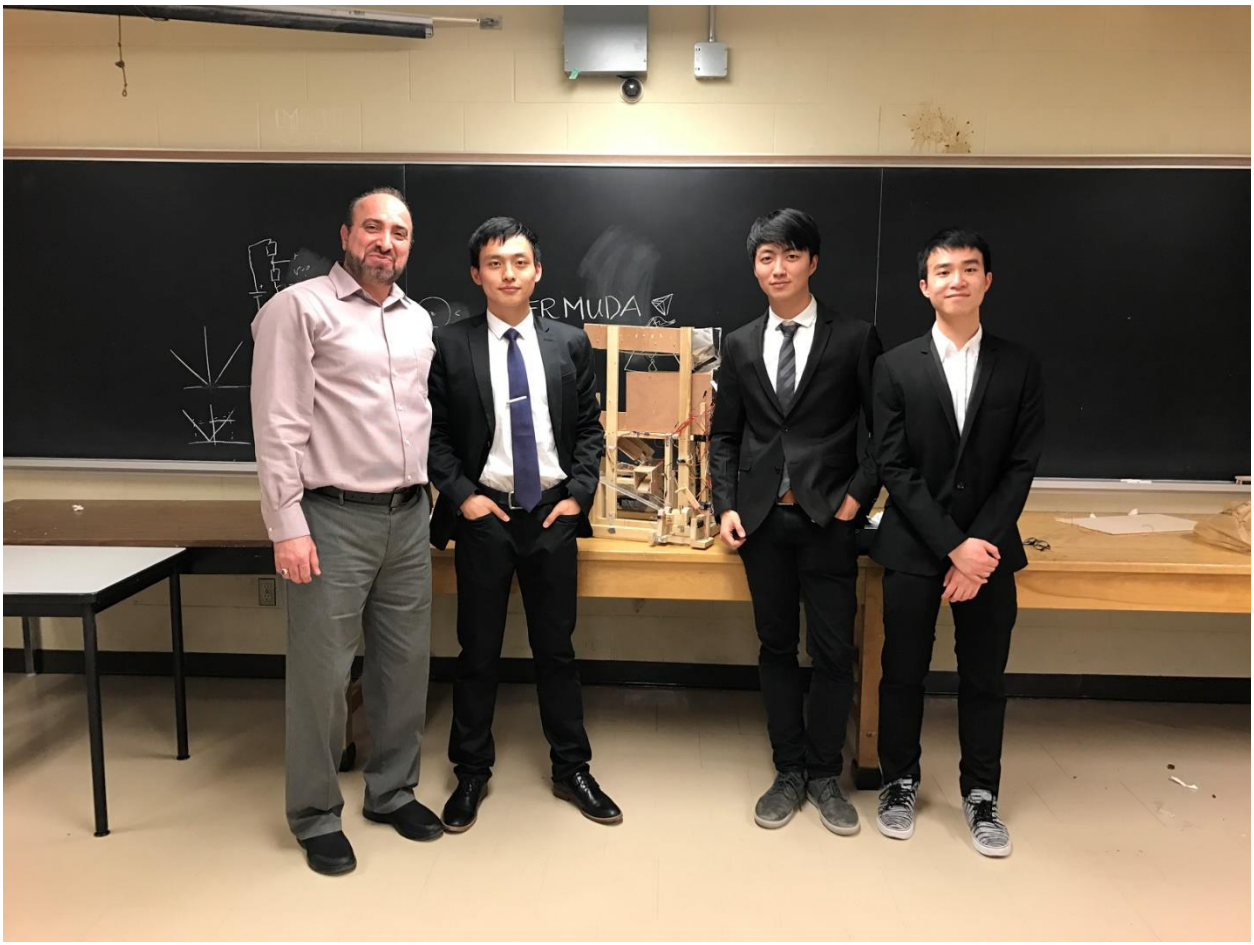
Meet the Team



From left to right:

Professor M. Emami

Leo (Shoujun) Feng (1001117506)

Litos (Hanze) Li (1002526493)

Alex (Shen) Lin (1001367193)

Acknowledgment

Our robot would not do a single thing if it wasn't for the help of our wonderful peers, teaching assistant and professors.

We would like to express our appreciation to professor M. Emami for coordinating the AER201 course and giving us this opportunity to gain experience in the engineering profession.

As well, we want to thank the following people for their valuable technical support, inspirations and mental support:

Nathan Cole, the teaching assistant, who help and contribute to the design with great patience
Colin Harry, machine shop technologist, we deeply thanks your effort and selfless help
Jingyi Han, engineering science, 1T9, provided invaluable mental support to Microcontroller member
Kuanpei Chen, lifescience, 1T9, with invaluable mental support to Electro-Mech member
Yueyang Zhang, engineering science, 1T9, with invaluable mental support to Circuit member
Bingruo Liu, engineering science, 1T9, the lady who is willing to share her great knowledge
Tongqi Zhu, engineering science, 1T9, lent her goggles to the team many times
Junhao Bi, engineering science, 1T9, helped bought 2 solenoids and gave great suggestions
Kaiyang Chen, engineering science, 1T9, who suggested the team to use some grease
Haobang Wu, engineering science, 1T9, accompanied the lonely Circuit member to solder
Zhuoran Lu, engineering science, 1T9, helped buy diodes with great quality

And a great list of people we cannot thank more about your help:

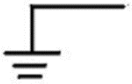
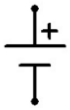
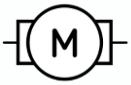
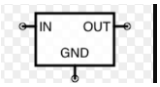
Yipeng Huang, engineering science, 1T9
Wei Hu, engineering science, 1T9
Ze Jia, engineering science, 1T9
Weixin Liu, engineering science, 1T9
Erkang Liu, engineering science, 1T9
Che Liu, engineering science, 1T9
Yiqing Luo, engineering science, 1T9
Xuran Ma, engineering science, 1T9
Ningrui Rui,engineering science, 1T9
Zhaocong Yuan, engineering science, 1T9
Ke Zhang, engineering science, 1T9
Tongyi Zhu, engineering science, 1T9
Kejie Zhao, engineering science, 1T9

Abstract

The problem presented by the RFP is to design and manufacture a fully autonomous machine that is capable of separating various batteries based on their type and charge condition. In order for the machine to be easy-to-access and portable, there are various constraints imposed by the RFP such as budget, dimension, weight, sorting speed et cetera. The “LAL sorter” by team 10 was designed to specifically target the requirement and constraints of the RFP, which is able to sort batteries very quickly with virtually no chance of jamming. The machine consists of a loading platform, primary and secondary sorting mechanism, and three voltage measuring areas for three types of batteries. “LAL sorter” integrated the electromechanical, circuitry and microcontroller subsystems, which will be discussed in great detail in the report. Background information, problem formation subsystem integration, budget, schedule, and areas for improvement are also presented. The performance of team 10 and the “LAL sorter” is also evaluated in a holistic and practical perspective. In the public demonstration on April 10, 2017, The “LAL sorter” was one of the five battery sorting machine to perform at least one “qualified” run out of the two run. Team 10 was very proud of the “LAL sorter”, in addition, the valuable lessons learned by the team members during the span of this project will be cherished for the rest of their life.

Table of Contents

1. Notations

Component	Symbol
Resistor	
Ground	
Transistor (NPN)	
Diode	
Switch	
Power Supply	
Motor	
Voltage Regulator	

2. Abbreviations

Abbreviation	Meaning

PIC board	Programming Intelligent Computer board, a type of microcontroller
Rpm	Rotation per minute
V	Volts
A	Ampere
Ω	Ohms
Team 10	The design team responsible for this project, author of the final report
DC	DC motors
Servo	Servo motors
9V	9V batteries
C	C batteries
AA	AA batteries
ADC	Analog Digital Converter
RTC	Real Time Clock
LCD	Liquid Crystal Display
PWM	Pulse Width Modulation
EEPROM	Electrically Erasable Programmable Read-only memory

3. Introduction

The chemical compositions of batteries differ depends on their types, but some heavy metals most commonly found in batteries are cadmium, lead, mercury, nickel and electrolytes [1]. If not properly disposed, these heavy metals can contaminate the landfills they are disposed, resulting land and water pollution. Such pollutions can then damage human health through birth defects, cancers and much more. Despite the potential harm of batteries disposal, only a small portion of batteries are being recycled due to various challenges and holdbacks. In 2012, Ontario's target recycling rate for all batteries is at 20.2%. In comparison, It is estimated that Ontario's recycling rate for water bottles is at around 66- 75%, at least three times higher than batteries. It is estimated that around 2000 tonnes of batteries are not being collected, but disposed in landfill. As a result, a new mean of properly recycling batteries is badly needed.

4. Project Concept and Design Parameters

4.1 Overall Objective

The machine requested by the RFP needs to sort a maximum number of 15 mixed batteries within 3 minutes. The batteries come in 3 types and need to be sorted into 4 categories: (1) charged AA batteries; (2) charged C batteries (3) charged 9V batteries and (4) uncharged batteries. A battery is defined uncharged if the voltage between its terminals is less than 85% of its nominal rating.

4.2 Requirements and Constraints

- Autonomous operation with no interaction, external PC, or remote control
- Dimension of the machine must be no more than 0.7 x 0.7 x 0.7 m^3
- Weight of the machine should not exceed 10 kg
- Cost of the machine should not exceed $230 CAD
- Must have an easily-accessible emergency STOP button that stops all mechanical parts immediately
- Should not damage the batteries sorted in any way

4.3 Division of the Problem

The scope of work for the battery recycling project include project planning, conceptualization, subsystem construction, subsystem integration, and debugging. The period of performance is 14 weeks beginning on 9 January, 2017 through 14 April, 2017. The specific tasks are broken down to three categories to each of the three team members:

4.3.1 Electromechanical member

In the development of the sorting machine for batteries, electro-mechanical member is in charge of fabrication and assembling of all mechanical parts of the machine. More specifically, this member will be responsible for the construction of the loading platform, passive sorting mechanisms, conveyor belts, frames, tracks, voltage measuring mechanisms and containers etc. The development of these physical mechanism is essential and is the foundation of the other two members' work which will directly affect the quality of the sorting machine as a whole.

4.3.2 Microcontroller member

Microcontroller member is responsible for all the dynamic control mechanisms of the whole system. More specifically, this includes time recording mechanism, counting mechanism, selective path control mechanism, start and stop mechanism, voltage measuring mechanism and any programmable additional featured mechanisms. These mechanisms are implemented with the help of circuit and electromechanical members.

4.3.3 Circuit member

Circuit member is responsible for the wiring of the whole machine, motor/solenoid driver circuit, circuits for sensors, protection circuits for the PIC board, voltage and current analysis, power consumption analysis and all other circuit components required in the project.

5. Perspective

5.1 Market Survey

There were three reference designs found in the market that serve as batteries sorting devices. They were analyzed by the design team for inspirations and also to learn from the shortcomings of other designs.

5.1.1 Design 1: Autonomous Battery Sorting Machine

Figure 5.1.1 Autonomous Battery

This fully autonomous machine [3], designed by a Chinese engineering team, was used to sort batteries based on their shape (cylindrical and square), and chemical composition (ZnC and ZnCl). Its sorting mechanism for cylindrical and cuboid batteries using a upward-moving conveyor belt was implemented as a final design concept and will be discussed in details in later sessions of this proposal [3]. This machine doesn't have the feature to sort batteries based on charging conditions. More importantly, this massive machine is built mainly using steels, which is very expensive. Thus it is safe to assume that this machine would not meet the constraints of size and budget. However this machine has a very clever way of sorting 9V and C batteries by placing them on an upward-moving inclined conveyor belt, with C batteries rolling down the conveyor belts and 9V batteries being transferred upward by the conveyor belt. This design concept was later implemented in the "LAL sorter"

5.1.2 Design 2: OBS 600 Battery Sorting Machine

Figure 5.1.2 OBS 600 Battery Sorting Machine

This battery sorter is used to sort batteries based on their chemical compositions (Alkaline, Zinc, Lithium etc) [4]. It utilizes high-speed cameras, together with various sensors to sort 600kg of batteries per hour. Most of the technologies implemented in this design are not

available to Team 10 due to budget constraints and limited capabilities of team members. However, this design used a brush above the moving conveyor belt to prevent any batteries from stacking on one another, and also light sensors for identification of sizes and location detection, which are simple but effective mechanism that are worth considering. One of the critical weakness of this design is its noise level, which is mainly caused by batteries smashing against metal containers at high speed. This reminded Team 10 that no excessive noises is one of the constraints imposed by the RFP.

5.1.3 Design 3: Opti-sort Battery Sort

Figure 5.1.3 Opti-sort battery sort

This machine utilizes high-end technologies to identify sort batteries, such as optical recognition and air jets. The overall design concepts are pretty similar to the previous reference design, except for the vibration mechanism and paddle blocking system used to prevent batteries from being stacked up and/or stacked up.

5.2 Circuit Survey

5.2.1 Full-wave rectifier

Rectifier is a electrical device that converts AC input into DC output, this is accomplished by allowing current to flow in only one direction. A rectifier also serves an important purpose of ensuring the voltage output to always be positive, regardless of the sign of the voltage input.

As shown in the figure below, the full-wave rectifier consists of four diodes and a power input. When the voltage input is positive, the diodes denoted as D1 and D2 are in series while D3 and D4 are in reverse biased. As a result, the current flows through the circuit as shown in the figure. Similarly, when the voltage input is negative, the current flows in a different path. But in both paths, the load voltage is positive. Thus the circuit could successfully rectify the input voltage into a positive voltage.

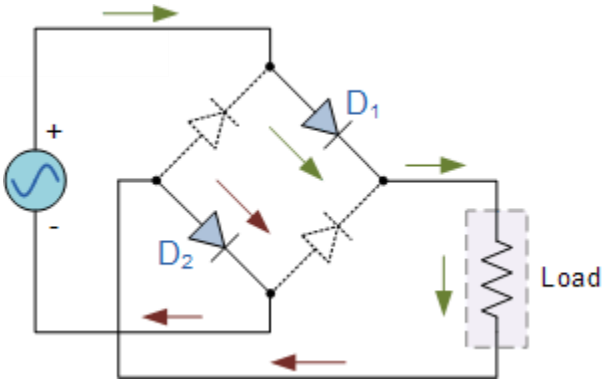


Figure 5.2.1 Full-wave rectifier

This is extremely useful for measuring voltages of batteries with PIC board, since the PIC board can only take in positive voltage signal. Thus a full-wave rectifier was implemented to every voltage-measuring circuit.

5.2.2 Voltage regulators

Voltage regulator is an electrical device that takes in a range of voltage input and supply out a constant voltage output by dissipating the excess power into the surrounding as heat. It is a powerful device with a lot of uses, but one of the potential hazard is overheating. The higher the difference between the voltage input and output, the more power dissipated as heat and the hotter the voltage regulator becomes. As a result, extra attention must be paid toward the temperature of the voltage regulator to prevent it from burning down, and a heatsink may be needed to help cool the voltage regulator down.

5.2.3 Voltage divider

A voltage divider is a linear passive circuit that gives voltage output as a fraction of the voltage input. A simple voltage divider consists of two resistors in series, with the voltage input applied across the two resistors, and the voltage output being the voltage between the ground and the second resistor, as shown in the figure below.

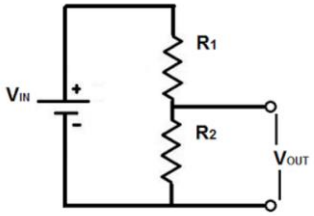


Figure 5.2.3 Voltage divider

The actual fraction in a resistor voltage divider is determined by the resistance ratio of the two resistor:

$$V_{OUT} = V_{IN} \frac{R_2}{(R_1 + R_2)}$$

Voltage dividers have wide range of uses, including to create reference voltages, or as signal attenuators at low frequencies, and most commonly, to reduce the magnitude of a voltage for it to be safely measured. One thing to keep in mind of is that a voltage divider's performance may change dramatically when it is under load, as shown in the figure below. In such cases, the resistance ratio is the ratio between R1 and the combined resistance of R2 and load resistance R3.

5.2.4 DC motors [2]

Relatively inexpensive, starting from $5. Requires simple circuitry only. The rotation of the motor can be reversed by simply reversing the signs of the voltage input or by using a H-bridge. It has fixed nominal voltages of 5V and 12V, voltage regulators may be needed to adjust for such voltages. It also has fixed upper limit for rotation per minute (rpm) and torque. Rotational speed may be increased by applying a higher voltage input. Does not allow for precise control of angular positions. A gearhead may be added to increase torque and decrease rpm. It is most commonly used in most scenarios where precise positioning is not required. However, since different DC motors have different nominal voltage and rpm, the amount of torque and moment of inertia needed for the motor should be calculated to decide on the most suitable motor.

5.2.5 Servo motors [3]

Servo motors are much more sensitive than DC and stepper motors, but can be very expensive, starting from $15. Servo motors are rated by their holding torque and response time, and can have both continuous rotation and rotation to a specific angular positions. Servo motors have fixed rpm. Increasing voltage for servo motors can increase both their torque and rpm. Servo motors included in the project kits have fixed nominal voltages of 5V and 12V, respectively. There are two general categories for servo motors: continuous and non-continuous. Continuous servo motors have rotation range of 0-360°, while non-continuous servo motors have rotation range of 0-180° but its rotational angle could be controlled much more precisely.

5.3 Electromechanical Survey

5.3.1 Loading Mechanism Survey

A reference design of loading mechanism was raised during the brainstorming stage: a rotating pole with three-layers platforms. Batteries will be separated into smaller cluster of 1 to 3 batteries such that the number and speed of batteries can be controlled. The possibility of jamming is also virtually eliminated, since batteries are no longer in a big cluster but separated into smaller groups. This design was saved as a backup design. [2]

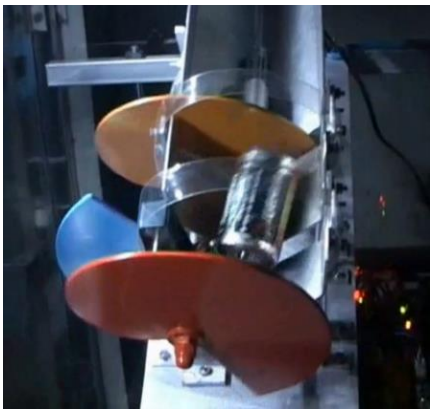


Figure 5.3.1 Three layers rotator

Another reference design of coin sorting machine was a good substitution for the loading mechanism. This loading mechanism utilizes rotational force and the difference in geometries for quick and effective sorting.[3]

Figure 5.3.1.1 Rotating disk

5.3.2 Sorting Mechanism Survey

This passive sorting mechanisms for coins found online utilized the differences in size of different coins. This simple method of passive coin sorting was implemented in various different design. Moreover, this mechanism has inspired the design of the primary sorting mechanism for "LAL sorter".[4]

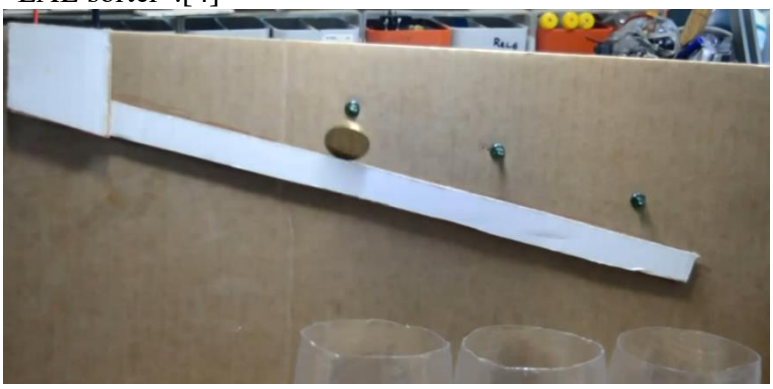


Figure 5.3.2 DIY Coin sorting machine

5.4 Microcontroller Survey

The C programming language was chosen to program the microcontroller PIC18F4620 from Microchip. The reason for the choice of PIC18F4620 and C language will be explained in detail later in this report. This section is about the survey on how to use C programming language to program the microcontroller efficiently and effectively.

Key functions that needed to be accomplished by the microcontroller in this project included analog input signal to digital input signal converter for measuring the voltage, real time clock for displaying time, keypad interface and keypad-LCD interaction, converting digital output signal to analog output signal.

During the survey, the sample code provided by Professor Emami was very helpful. The following paragraph will explain in detail how survey was conducted step by step. All of the codes related in these survey activities are included in Appendix 9.1.1.

5.4.1 Port test

The first thing that was studied by the microcontroller member was to load the program onto the microcontroller chip PIC18F4620 and to test all pins on the chip were functional by means of PIC Development Board Debug Module. This was instructed by Professor Emami during the experiment session. The main goal in this activity was to learn how to set specific pins to high and how to use the debug module in the PIC Development Board to see whether there exist bugs in the program.

5.4.2 Keypad LCD interaction

The second thing that was explored by the microcontroller member was how to interact with keypad and how to display information on the LCD screen. The interrupt function was used to detect the keypad presses and the part of the sample code regarding LCD display was utilized to display information on the LCD screen. Two small programmed chips on the PIC Development Board were taken advantaged of so that the low-level detailed implementation for programming keypad and LCD screen didn't need not to be written in this project. However, the microcontroller member needed to proficiently master how to utilize the given module. The learning activities were more complex than port test and involved more sample codes. After these activities, the microcontroller member was able to make the user interface without any actual implementation for the underlying functions such as measuring voltage and displaying real time.

5.4.3 Analog to digital input signal converter(ADC)

For measuring voltage and determining whether a battery is charged or not, the analog voltage input from the voltage reading pin needs to be converted into digital input which can be analyzed by the program. This is accomplished by utilizing the built in A/D Converter module in the PIC18F4620. The microcontroller member studied the related sample code and understood what conditions should be set before calling the A/D Converter module, how to set the reading channels, and how to read from bits after calling the A/D Converter. These are crucial for measuring voltage of batteries.

For measuring voltage, some experiments were conducted in order to explore the way PIC18F4620 responds to analog input. It was discovered that only input in the correct orientation can be read. In the opposite orientation, the reading was zero. Therefore, the microcontroller determined that the circuit of the machine must make sure signals received by microcontroller is in the correct orientation.

During the experiments, it was also discovered that the reading from the pins will fluctuate in the open air. The reading from exactly same situations might very likely not the same, but were very close. The power input to the microcontroller and circuit configuration would significantly influence the readings. These findings played an important role in later design improvement and calibration process.

5.4.4 Real time clock

One requirement of the project was to record the operation time. This can only be achieved by the RTC on the PIC Development Board. With the PIC development board, the PIC18F4620 chip uses an external clock(RTC) to keep track of time. The initial time needed to be set manually by loading a setting program onto the board, and later on RTC will keep going until the coin battery on the PIC Development Board runs out of power.

5.4.5 Digital to analog output

Servo motors used in this project required analog signals to operate. The voltage height and frequency requirements must be fulfilled in order for servo motors to function. The microcontroller, PIC18F4620 chip, is only able to output digital signals. However, as introduced by Professor Emami during the lecture sessions at the beginning of the project, the equipment will not be able to respond as fast as digital devices like PIC18F4620. Therefore, the idea was introduced by Professor Emami that periodic high and low signal output can achieve the same goal as the analog output. This idea was essential in later process of the project for controlling multiple servo motors.

5.5 Related theories

5.5.1 Torque [7]

Torque is a measure of how much a force acting on an object causes that object to rotate. Consider the example shown in the figure. If an object rotates about a point, such point is called the “pivot point”. If a force F is applied on the object at angle θ at distance r from the pivot point, such distance r is called the level arm. The torque on the object by the force is then calculated to be

$$\tau = r \times F$$

Calculation of torque is essential for this project, especially in motor selection. When selecting a specific motor, the most important criteria is for the motor to have enough torque given the conditions to rotate the particular shaft.

5.5.1.1 *Example of torque calculation on a conveyor belt [5]*

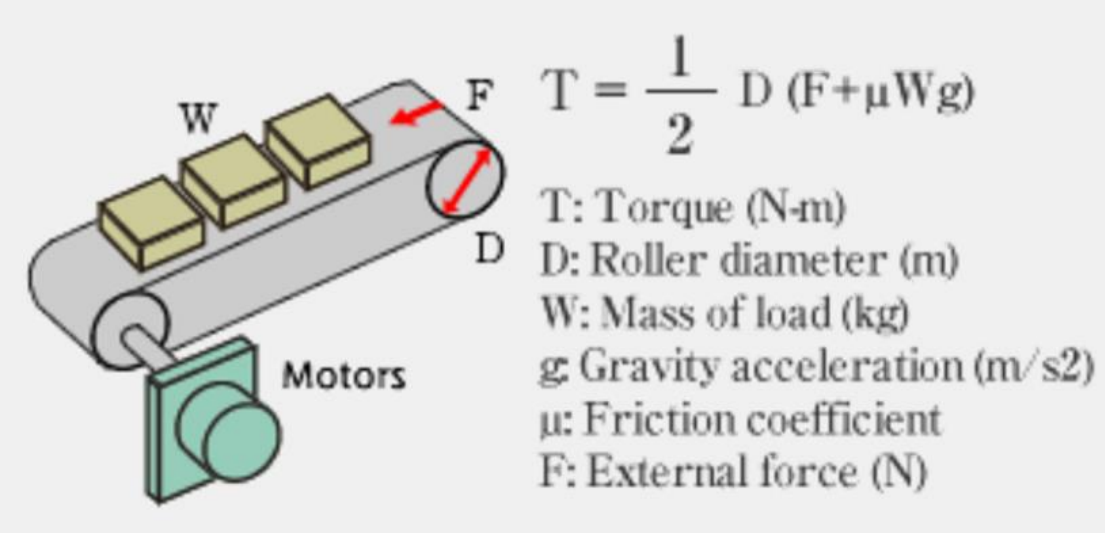


Figure 7.5.1.1 Torque Calculation on a conveyor belt

5.5.1.2 *Hooke's Law for Mechanical Arm Calculation [6]*

Hooke's Law was utilized specifically in the calculation of torque for the mechanical arm for 9V, since there were springs to be compressed and released during voltage measurement.

Hooke's Law specifies the relationship between the relationship between the deformation and the force of the spring. The equation is:

$$F = k\Delta x$$

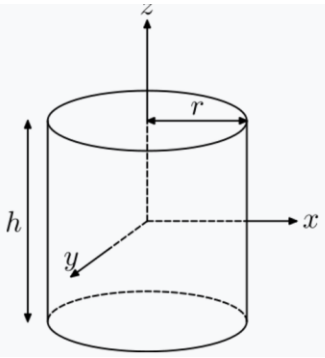
Where k is the stiffness of the spring and Δx is the change of length compared to the spring's initial length. However, this relationship works only in a certain range of compression or extension and does not count into the effect of friction etc. Therefore, the calculation was only a estimation for the force or torque needed for the mechanism.

5.5.2 Moment of Inertia [7]

Moment of Inertia was calculated for specific rotational components existing in the design. The calculation of moment inertia is given by the formulas:

Solid cylinder of radius *r*, height *h* and mass *m*.

This is a special case of the thick-walled cylindrical tube, with $r_1 = 0$. (Note: X-Y axis should be swapped for a standard right handed frame).



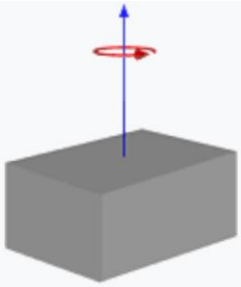
$$I_z = \frac{mr^2}{2} \quad [1]$$

$$I_x = I_y = \frac{m}{12}\left(3r^2 + h^2\right)$$

Figure 5.5.2.1 Moment of Inertia calculation for cylinder

Solid cuboid of height *h*, width *w*, and depth *d*, and mass *m*.

For a similarly oriented cube with sides of length *s*, $I_{CM} = \frac{ms^2}{6}$



$$I_h = \frac{m}{12}\left(w^2 + d^2\right)$$

$$I_w = \frac{m}{12}\left(d^2 + h^2\right)$$

$$I_d = \frac{m}{12}\left(w^2 + h^2\right)$$

Figure 5.5.2.2 Moment of Inertia calculation for solid cubic

5.5.3 Power consumption

Power consumption of an electrical component is the total power consumed and dissipated by that component due to an applied voltage and the amount of current flowing in the circuit containing the component. When selecting a component, a suitable component not only need to perform their intended task, but also need to safely survive under the given operating conditions. For the latter, the most common check is to ensure that the power consumption of a component is below its maximum power ratings. Otherwise the parts would quickly fail to operate as intended or even burned down due to overheating.

To calculate the power consumption of a component with applied voltage V, internal resistance R and current I passing through it is

$$P = VI = I^2R = \frac{V^2}{R}$$

5.5.4 Heat dissipation [11]

Heat dissipation of an electrical component is the process of that component losing heat to the ambient due to temperature difference. Heat dissipation is important because to dissipate more heat means that the components needs to have higher temperature, thus it is important to consider especially for when a component faces risk of overheating. The amount of heat

dissipated by a component is the difference between the amount of energy entering the component and amount of energy outputted by the component, or

$$\dot{Q} = P_{in} - P_{out}$$

The amount of output energy for a given input energy is different for every type of component. For example, a resistor has no ability to output power, and its heat dissipation equals to its power input, $P = VI$; a motor is able to output a certain amount of mechanical power, with any excess power input dissipated as heat; a voltage regulator outputs a constant voltage by dissipating the excess power as heat.

For most components, overheating shouldn't be a problem if they are operated within the maximum rating/conditions, but heat dissipation still needs to be performed if components are observed to have too high of temperature.

5.6 Problem Limitations

The potential solutions to the problem presented in the RFP have some inherent limits due to the characteristic of batteries and other non-human factors.

- With the current state of technology, there is always a certain degree of precision when performing a measurement. In the context of measuring voltages of batteries, such precision would be lowered due to the non-ideal electrical parts and other external factors in the voltage measuring process. As a result, there is always a possibility of sorting the batteries incorrectly if their voltages are very close to 85% of the nominal ratings.

- Some batteries may be broken or have really small voltages to be measured, this causes difficulty in determining whether the battery is actually broken or it is due to bad contact at the terminals.

6. Description of Overall Machine

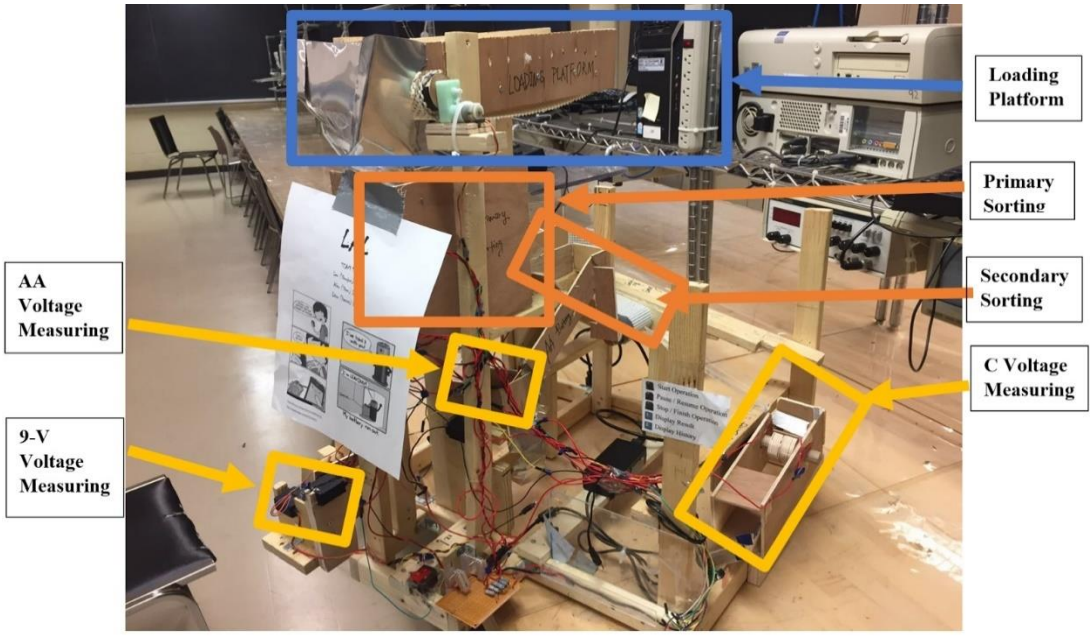


Figure 6(1) Front View

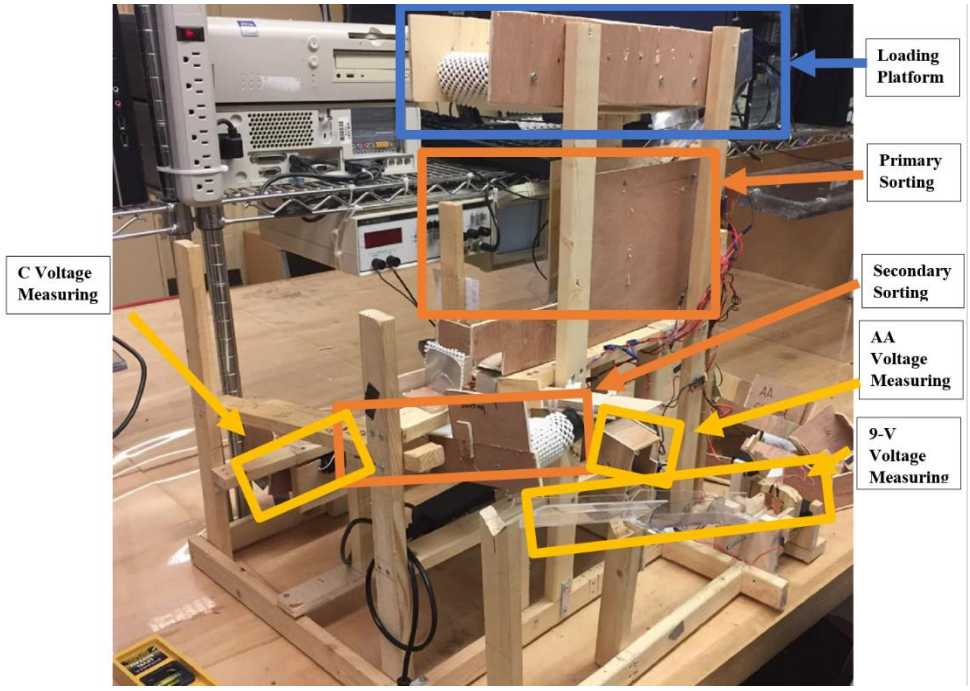


Figure 6(1) Back View

The **"LAL sorter"** was designed to be a fast and stable battery sorting machine for AA, C and 9V batteries with three operating processes: loading, sorting and voltage measuring. AA, C and 9V batteries can be separated based on their types and charging conditions by the LAL sorter. The machine does not use any sensors and has the advantages of low possibility of

jamming and high sorting speed with decent success rate. The name "LAL" came from the initials of Team 10's members (Litos, Alex and Leo), it is also short for "Laughing a Lot", which represents the mentality of the team members, to always face any challenges positively, no matter how tough the situation is.

In LAL sorter, the batteries are to be loaded on a loading platform consists of a large conveyor belt. After the start button on the microcontroller board is pressed, the conveyor belt will transport the batteries slowly into the primary sorting mechanism where AA batteries will be first separated for voltage measuring. This is done by a passive-sorting track that was specially designed to only allow AA batteries to pass through without obstacles based on their relatively small sizes. On the other hand, C and 9V batteries will slide into an adjacent track and drop to a conveyor belt which was mounted at an inclining angle. Then, the cylindrical C batteries will roll down while 9V batteries will be carried up by the conveyor belt into their corresponding voltage measuring areas.

Three voltage measuring areas were connected by tracks following the sorting process. AA and C batteries will slide to their positions onto a "water wheel", where their terminals will come in contact with copper coils for voltage measurements. Then the batteries will be sent into different bin by rotating the water wheel clockwise or anticlockwise depending on their charge conditions. 9V batteries will slide into a measuring box with 4 copper plate supported by springs. A mechanical arm will periodically press down the 9V battery to ensure an optimal contact of the battery terminals and the measuring plates. Similar rotating mechanism helps separate charged and drained 9V batteries after the voltage measuring. 6 bins were placed at corresponding places for collection sorted batteries.

7. Electro-mechanical subsystem

The electromechanical subsystem contains all passive sorting parts and supports of the machine. During the construction process, detailed designs such as buffer mechanisms and elongation of certain paths were made in response to unforeseen problems. These solutions were improved and refined iteratively in the construction and integration stages.

7.1 Introduction

The design of the ***LAL sorter*** has mainly three components: the Loading Platform, the Sorting Mechanisms and the Voltage Measuring Mechanism. The mechanical design of each functional part are to be first described followed by its actual construction and potential improvements. Important supporting documents or calculations are to be provided when necessary. Possible improvements about the general mechanical design and assessment of the current design are be presented in the end.

7.2 Frame and Connection Mechanism

7.2.1 Assessment of the general problem

The overall dimension of the machine was restricted by the frame to be under 0.7*0.7*0.7m^3 and the total weight was restricted below 10 kilograms. Also, the mechanical part of the machine was designed to allow space for the other two subsystems.

7.2.2 Material Selection and Construction

Wood was selected to be the base material due to its low cost and easiness to be joint with metal. Acrylic sheets were used for the construction of the holder for PIC board and tracks because it is aesthetic appealing and provides low friction coefficient with the contact batteries. A formal decision making process was made using Utility-based method, and is shown in Appendix 1.

More specifically, 1'' x 2'' inches wood beams were used to construct the overall cubic frame of the machine. Loading platforms, sorting mechanisms, measuring mechanism and paths connecting each components were to be mounted on the frame. Screws were used to connect the beams at the corners in such a fashion that three beams are packed together to increase the stability of the frames and eliminate moment.

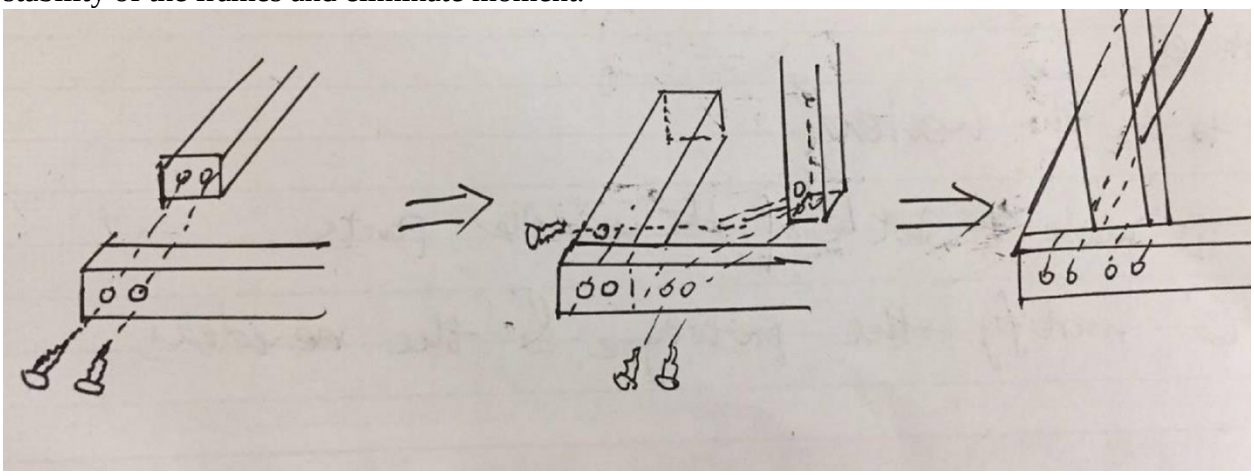


Figure 7.2.2 Base connection mechanism

7.3 Loading Mechanisms

7.3.1 Functionality of Loading Platform

A top loading platform was initially designed as a sliding-gate such that the batteries will drop during the movement of attached rack. An alternative design of conveyor belt was chosen such that the batteries can drop consistently at a controllable rate at one location.

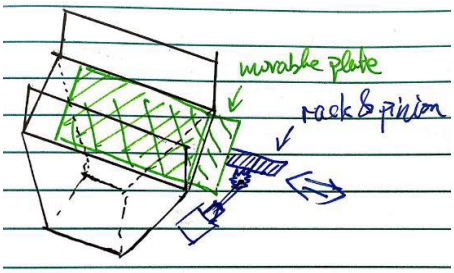


Figure 7.3.1 Alternative for top loading platform

Figure 7.3.2 Loading Platform

The goal of the final loading platform, when integrated with circuit and microcontroller subsystem, was to transfer the batteries slowly and steadily to minimize collisions between adjacent batteries in motion, which is a major external factor that could cause jamming or decrease the success rates for the rest of the sorting process.

Table 7.3.2 Experiment Value for Primary Sorting Mechanisms

Trial number	Number of batteries to be sorted	Dropping property	Rough time (s)	Sorting result*
1	11	All at once	1	6/11
2	13	All at once	N/A	Jam
3	13	All at once	N/A	Jam
4	2	All at once	1	2/2
5	3	All at once	1	3/3
6	3	All at once	1	2/3
7	5	Slow	5	5/5
8	5	Slow	6	5/5
9	8	Slow	6	8/8
10	13	Slow	6	12/13
11	13	Slow	5	13/13
12	13	slow	7	13/13

The result shows that the accuracy of the Primary Sorting Mechanism is the higher with lower speed and smaller number of batteries coming together.

7.3.2 Material and Construction of loading platform

The loading conveyor belt was made out of wood and insulation wrap around a hardwood pole. Rectangular wood blocks are placed between two shafts, inside the conveyor belt as supports for the loading platform. Non-slip shelf liner was used as the belt for the conveyor. The other conveyor belt that serves at secondary-sorting mechanism was construct in the similar way.

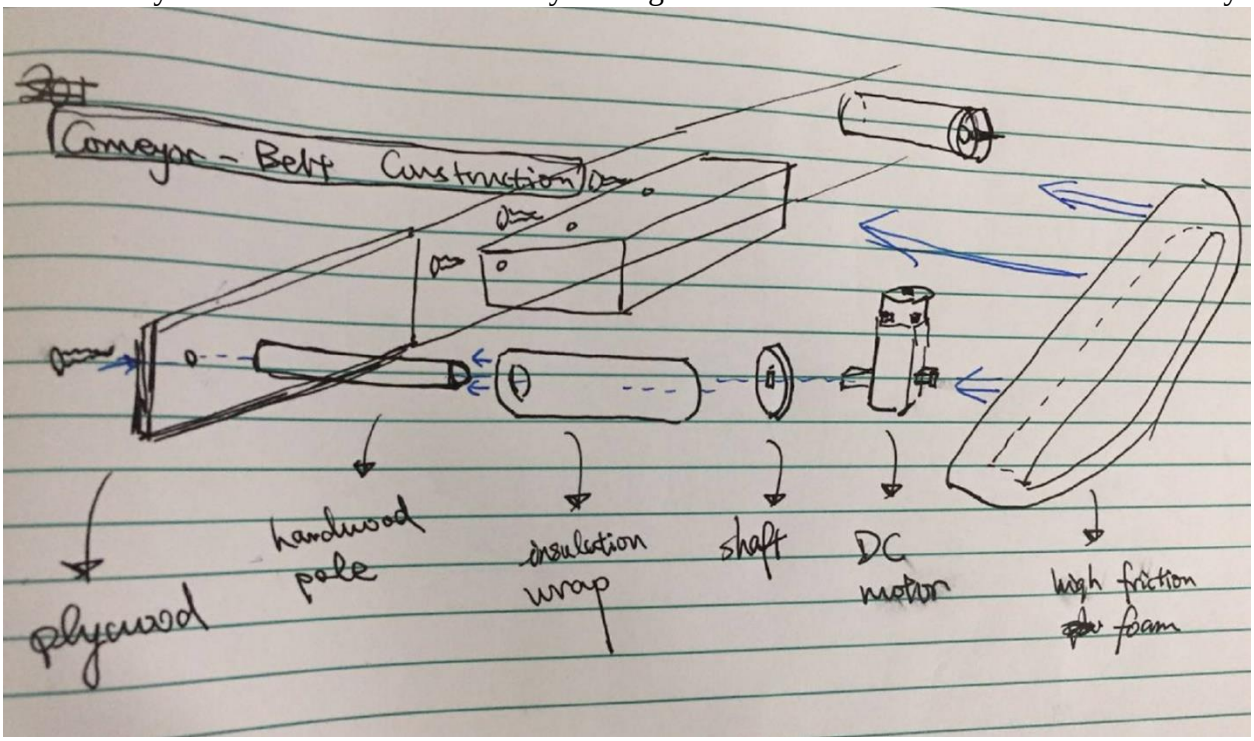


Figure 7.3.2(1) Construction of conveyor belt

Figure 7.3.2(2) Connection of pole and insulation wrap

Figure 7.3.2(3) Assembly of conveyor belt shaft

The motor was mounted using zip ties which are not only robust but also easy to be removed whenever the motor needs to be repaired or replaced.

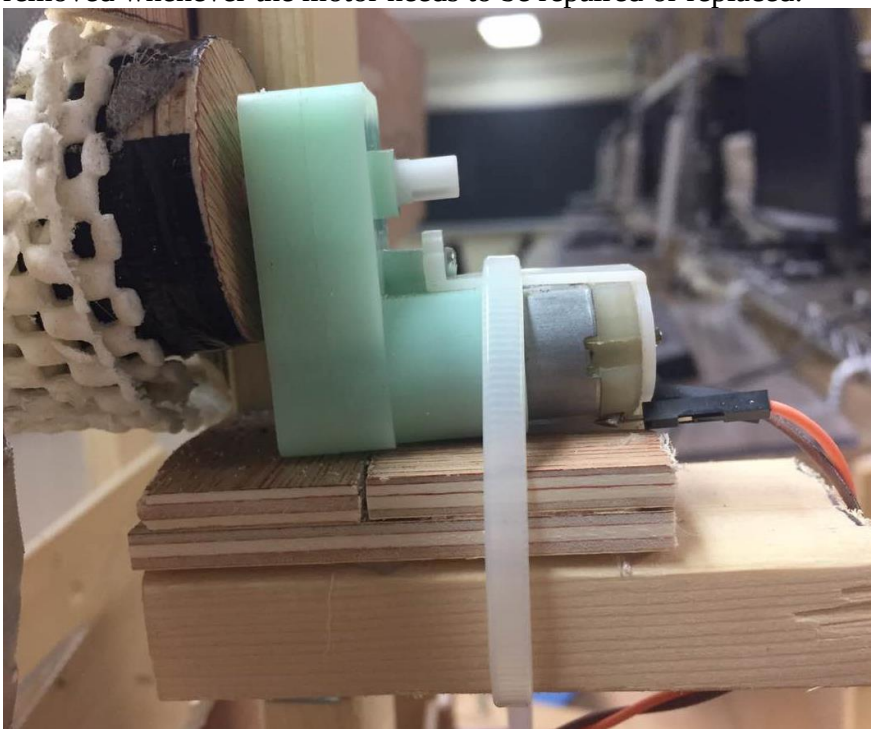


Figure 7.3.2(4) Mounting of conveyor belt

7.3.3 Calculation of the torque required for top loading platform

The calculation for the motor for operating the loading conveyor belt is shown below as a sample calculation.

The calculation assumed that there is no sliding anywhere between the conveyor belt and the battery, and the weight to be carried by the conveyor belt was assumed to be the maximum weight possible (15 C batteries since C are the heaviest among the three). The result showed a total torque of 1.47kg cm is required, the DC motor selected has stall torque of 4.01 kg cm, thus it is concluded to have more than enough torque to ensure smooth operation of the loading conveyor belt in all scenarios.

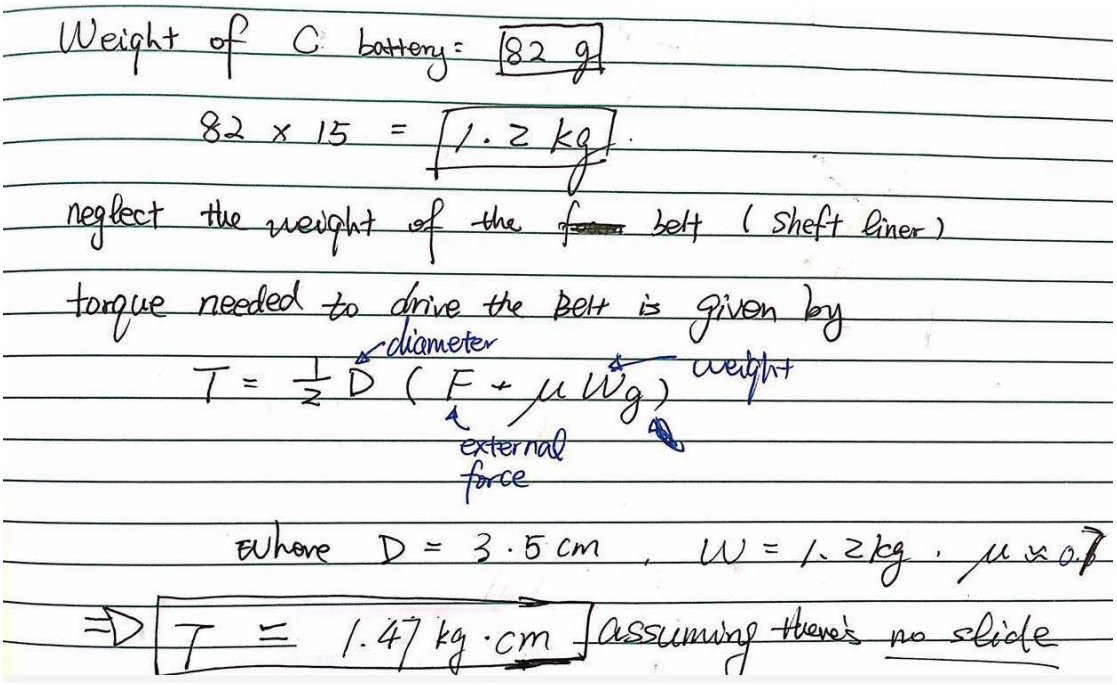


Figure 7.3.3(1) Calculation of Torque required for top conveyor belt

Cast iron on oak (parallel)	0.30-0.50	-
Cast iron on magnesium	0.25	0.133
Cast iron on steel, mild	0.23	-
Cast iron on tin	0.32	-
Cast iron on zinc	0.21	-
Earth on earth	-	-
Glass on glass	0.40	-
Hemp rope on wood	0.40-0.70	-
Nickel on nickel	0.53	0.12
Oak on leather (parallel)	0.30-0.50	-
Oak on oak (parallel)	0.48	0.16
Oak on oak (perpendicular)	0.32	0.07
Rubber tire on pavement	0.75-0.85	0.5-0.7*

Figure 7.3.3(2) Friction coefficient for Calculation [8]

7.4 Sorting Mechanisms

7.4.1 Primary-Sorting Mechanism

A passive-sorting mechanism was placed beneath the loading platform which utilized AA's small size. A small-stick was glued onto one side of the V-shape track made of acrylic sheet. The stick was designed such that only AA batteries can pass through without obstacle but the other two types of batteries will go into to the adjacent track which is followed by a secondary-sorting mechanism for further separation.

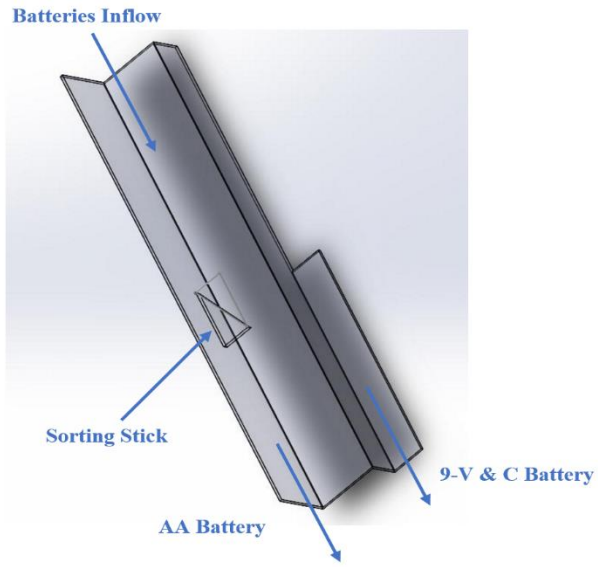


Figure 7.4.1(1) Primary Sorting Mechanism

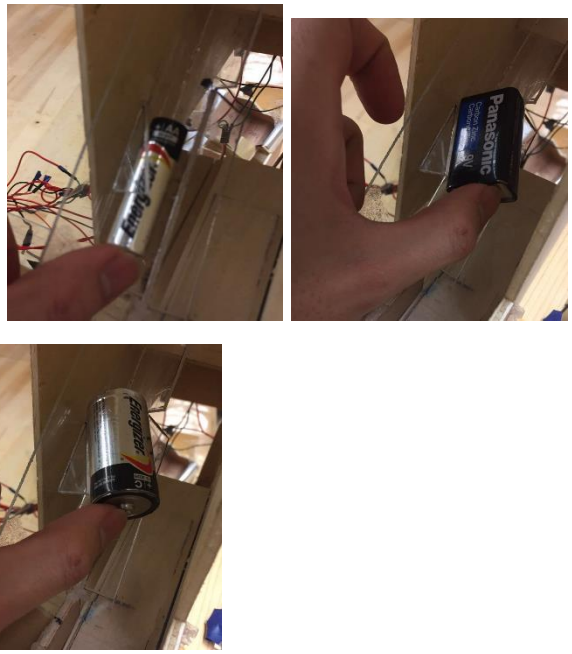


Figure 7.4.1(2) Only AA battery can pass through the stick

7.4.1.1 Problems and Resolutions

Certain extreme cases were recognized during which the primary sorting mechanism may fail, including:

1. Multiple batteries entering the primary-sorting mechanisms at the same time, causing jamming issues or mis-sorting:
    a. An AA battery sitting on top of a C/9V , causing the AA to be missorted.

   b. A combination of over 3 batteries entering the sorting area, causing mis-sorting of one or more battery or jamming issue.
   c. AA-battery stucked with the small-stick in the truck after it bounced from the bottom of the track due to high momentum.
2. Batteries getting pushed into non-intended tracks due to collisions with other batteries in high speed.

Figure 7.4.1.1(1) Type 1.a problem

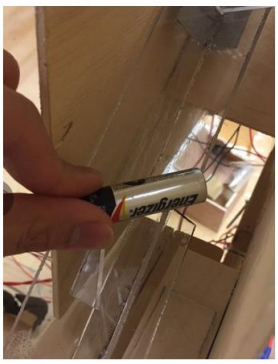


Figure 7.4.1.1(2) Type 1.c problem

Three general modifications were made in response of the above-mentioned circumstances:

1. Extend the sorting track such that:
   a. Batteries will have enough time to become stable at the bottom of the V-shaped sorting track before entering the sorting area, increasing the success rate of sorting.
   b. The chance of one battery sitting on top of another battery decreases since the top battery would have more time to roll down off the bottom battery when transported by the loading conveyor belt.
2. The problem caused by wrong orientation of the batteries was solved by gluing a small stick at the entrance of V-shaped track which is similar to the sorting-stick. Standing batteries would get knocked down when passing by stick.
3. Lowering the rate of batteries exiting the loading conveyor belt, which is discussed in details in session 11.4.1.

Figure 7.4.1.1(3) Extension of the primary sorting mechanism

7.4.2 Secondary-Sorting Mechanisms

An upwardly moving inclined conveyor-belt was used to separate the C and 9V batteries based on its geometry.

9V batteries will be carried up by the conveyor while C batteries can roll down the belt. The conveyor belt was placed beneath the Primary-Sorting Mechanism at an inclining angle of 15°. The construction of this conveyor belt is very similar to that of the loading conveyor belt.

Figure 7.4.2 Buffer area and orientation-restriction mechanism

7.4.2.1Buffer area and Orientation-Restriction Mechanism

Since batteries were dropping from the Primary-Sorting Mechanism at some height, they have some downward momentum that cannot be ignored. In real experiments, batteries were observed bouncing to the further side of conveyor belt, colliding to the next batteries coming to the sorting area and causing jamming issues. Moreover, C batteries can end up lying on the sorting conveyor belt while being completely parallel to it, preventing the batteries from rolling down as expected.

To solve this problem, a buffer mechanism was added using a piece of shelf liner located in the wall opposite to the exit of the Primary-Sorting Mechanism. Batteries with high speed will bump into the buffer area and significantly reduce its momentum before dropping to the conveyor belt,

minimizing the uncertainty of the sorting. Small pieces of wood were also attached around the buffer area to restrict the orientation of the batteries, increasing the stability of the batteries.

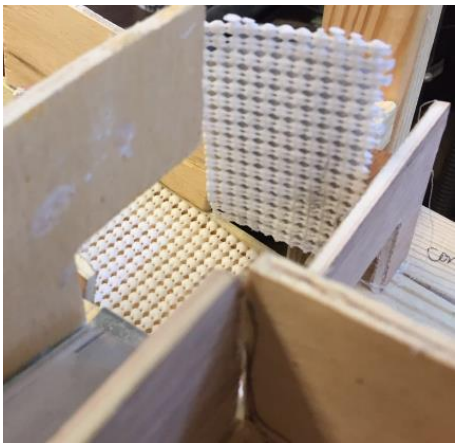


Figure 7.4.2.1 Buffer area and orientation-restriction mechanism

7.4.3 The order of sorting

The order of the sorting was decided such that AA batteries were to be sorted out from AA and C batteries. This order is very significant because the weight of AA batteries are much smaller than the other two kinds of batteries. Experiments and prototyping showed that if the AA batteries were not sorted out first, the success rate of the whole sorting process would dramatically decrease. As could be seen in the table in appendix 6

The reason could be the AA battery is light and is harder to maintain its orientation once dropping from a distance. The small weight also gives its smaller tolerance once the orientation is not as expected.

7.4.4 Compromise for mis-sorting

Since a jamming issue will cause the whole machine to stop operating, while a missorting only affect one or two individual batteries, it was decided by Team 10 to prioritize on preventing jamming instead of preventing mis-sorting. In addition, an evaluation was done to investigate the consequences of batteries entering the voltage measuring areas of other types of batteries. As shown in the table, ideally no jamming would occur even when batteries were missorted into tracks of other batteries, one of the reason for this is the fact that 9V and C batteries are very similar in heights.

batteries/Tracks	AA	C	9V
AA	Success Sort	AA batteries will not cause a jamming issue in the track of C batteries	AA batteries will not cause a jamming issue in the track of C batteries
C	C batteries rarely enter AA-batteries track due to the restriction of the sorting stick	Success Sort	C batteries will not cause a jamming issue in the track of 9V batteries but voltage cannot be measured.

9V	9 batteries rarely enter AA-batteries track due to the restriction of the sorting stick	9V batteries will not cause a jamming issue in the track of C batteries but voltage cannot be measured.	Success Sort

7.5 Voltage-Measuring mechanism:

After the sorting process, three types of batteries were expected to enter their corresponding measuring areas, where they would come in contact with copper coils coils or copper plates to measure batteries' voltages and send them to the PIC board. Charged and drained batteries based on the measured voltage were send into different bins by rotation of a servo motor.

7.5.1 C battery

7.5.1.1 Track and waiting area

The motion of C batteries, which rolled down from the conveyor belt, was converted to longitudinal motion at the entrance of V-shaped measuring track. This design was chosen because the longitudinal motion at the bottom of V-shaped track of a battery is more consistent and stable than that of rolling at a flat surface.

The end of the track was connected to a carefully designed measuring area such that only one battery can fit perfectly into the waiting area before rolling down to the measuring area at a time, followed by multiple batteries lining up at the waiting area and the end of the track.

Figure 9.5.1.1 Track and Waiting area

7.5.1.2 Measuring Design and construction

The voltage of C battery was measured at the measuring area surrounded by a water wheel shaped device with a 90° cut. The C battery fits into the space of cut, squeezed by copper coils fixed at both sides of the wall. Charged batteries were to be rotated clockwise and drained batteries were to be rotated contour-clockwise into different containers.

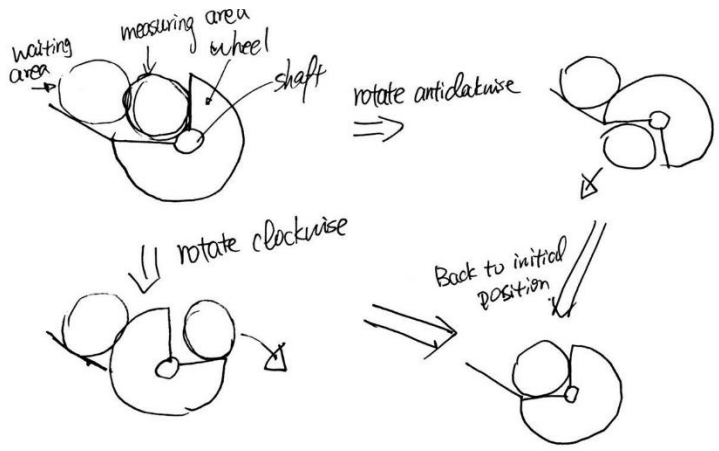


Figure 7.5.1.2(1) Cross section of C battery voltage measuring process

The construction of the measuring area composed of a hollow rectangular module with two circular holes at each side of the wall. A wheel that is fixed to the shaft went through the hole and was used to carry the batteries that goes inside the space of cut. The wheel would rotate back to its initial position to house the next battery in waiting area. Loose copper coils were attached at the appropriate location such that battery would fit perfectly into the space between copper coils with enough contact between the coils and terminal for voltage-measurement. The shaft was connected to a servo-motor and could be rotated clockwise and counter clockwise at will.

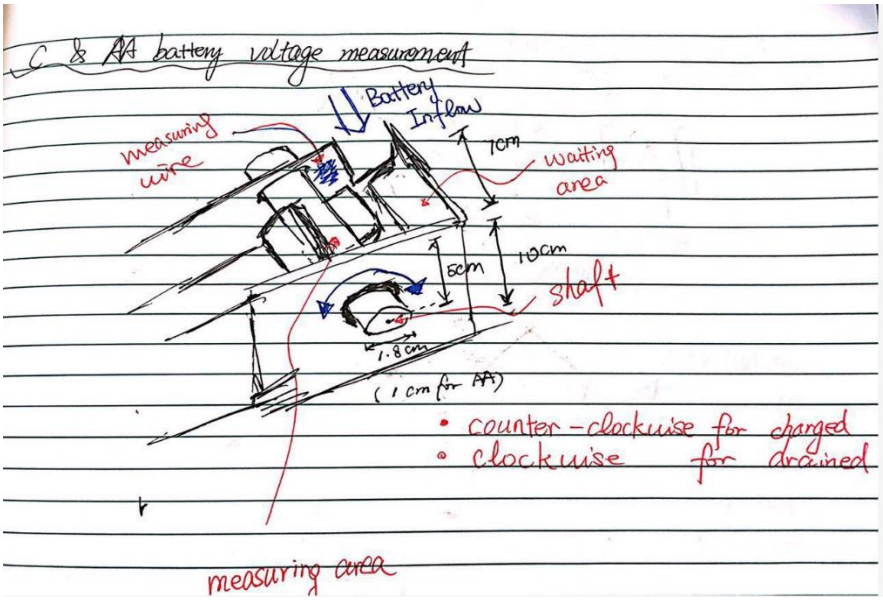


Figure 7.5.1.2(2) The C battery voltage measuring design

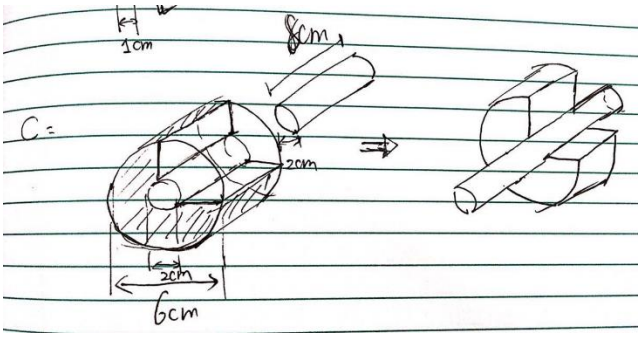


Figure 7.5.1.2(3) The construction of measuring wheel

7.5.1.3 Mounting of the servo-motor

The servo motor was mounted to a piece of wood by screws. The wood was cut to fit the servo motor and four screws were used to fix the servo physically. The shaft was connected to the servo motor by a special hat.

Figure 7.5.1.3 Mounting servo motor

7.5.1.4 Problems and potential improvements

It is discovered during experiment that the thickness of the copper coils were hard to determine such that batteries could fit inside without jamming and gain enough connection for voltage-measuring. Adjustment was made with small aluminum sheets that help direct the track of C batteries. It was considered by Team 10 to change the design altogether, since it was extremely difficult to successively measure voltages without batteries getting stucked. However, due to the time constraints, the team decided to maintain the current design. Potential improvements and alternative designs were being thought of and can be substituted given enough time in the future:

- Attach one side of the measuring coils onto a solenoid that presses to force the batteries into places to impose between connection between coils and batteries.
- Instead of converting longitudinal motion to rolling between the waiting area and the measuring area, use a direct method of measuring (similar to that of 9V battery measuring mechanism), eliminating extra uncertainty during the process.

.

7.5.2 AA battery

7.5.2.1 Track and waiting area

Since AA batteries were sorted directly from the Primary-Sorting Mechanism, a tailor-made track was placed beneath the sorting track and allow excess AA batteries to lineup outside of the measuring area.

Figure 7.5.2.1 Track and voltage measuring area for AA batteries

7.5.2.2 Measuring Design and construction

The measuring techniques of AA batteries are very similar to that of C batteries with carefully-made waiting area and wheel rotating back and forth to measure the voltage and separate the drained or charged batteries. .

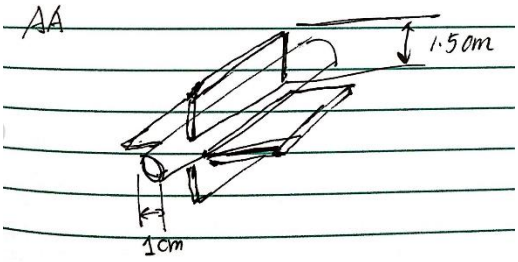


Figure 7.5.2.2 Construction of the measuring wheel for AA battery

7.5.2.3 Difference between C and AA battery voltage measurement mechanism

The main difference between AA batteries and C batteries measurement was that AA batteries used a four-leaves shaft that can measure the voltage more efficiently without jamming. While the first AA battery slides inside the measuring wheel, the following battery would stay in the waiting area. Once the first battery finished measuring, the wheel rotates to drop the measured battery and the next battery will directly fall into the measuring area between the adjacent leaves of the wheel. The process repeated till every battery in the waiting area and the track be sorted.

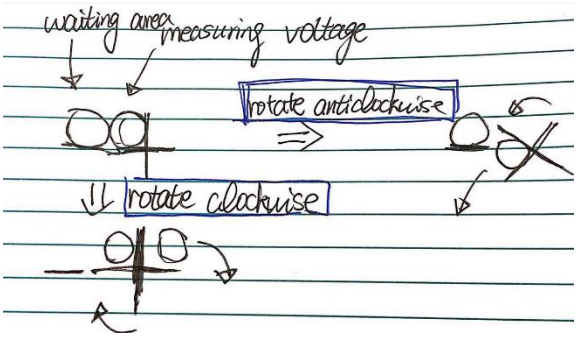


Figure 7.5.2.3 Cross section of the AA battery voltage measuring process

7.5.2.4 Problems and potential improvements

Similar problems in C batteries were recognized for AA batteries so similar adjustment using aluminum pieces are made to optimize the process.

Another Problem was that the AA battery voltage measuring mechanism was significantly inconsistent due to the loose connection between the shaft of the rotating wheel and the servo-motor. The existing mechanism was using a screw to connect the shaft and servo motor. However, since the rotation can occur both clockwise and anticlockwise, the torque of one direction of rotation may result in loose connection. The problem could be solved by using a better connecting mechanism with specific designed shaft.

Figure 7.5.2.4 Current Connecting mechanism for AA battery voltage measuring mechanism

7.5.3 9V battery

7.5.3.1 Track and waiting area

After 9V batteries were carried upslope by the conveyor belt, they fell into a V-shaped track followed with a flat track specifically designed for 9V batteries. The material and angles of the tracks were carefully designed such that there was enough gravity force for batteries to slide along the track even for batteries with no momentum due to blocking by batteries ahead, instead of stopping on the middle of the track.

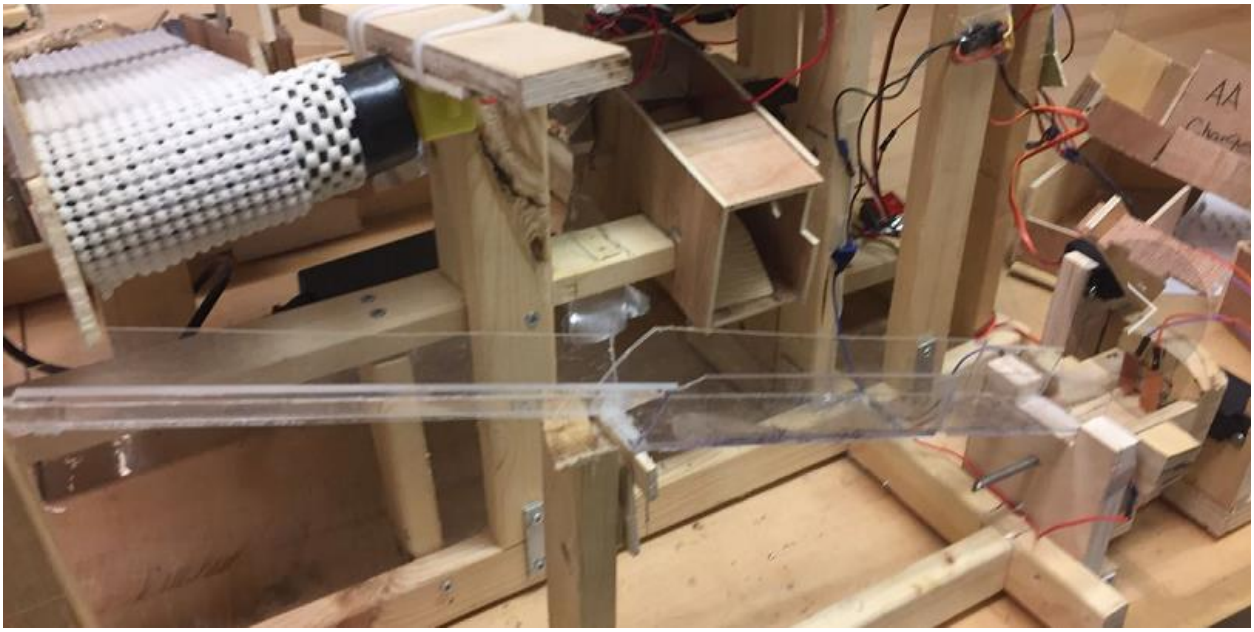


Figure 7.5.3.1(1) Track connecting the secondary sorting mechanism and 9-V measuring mechanism

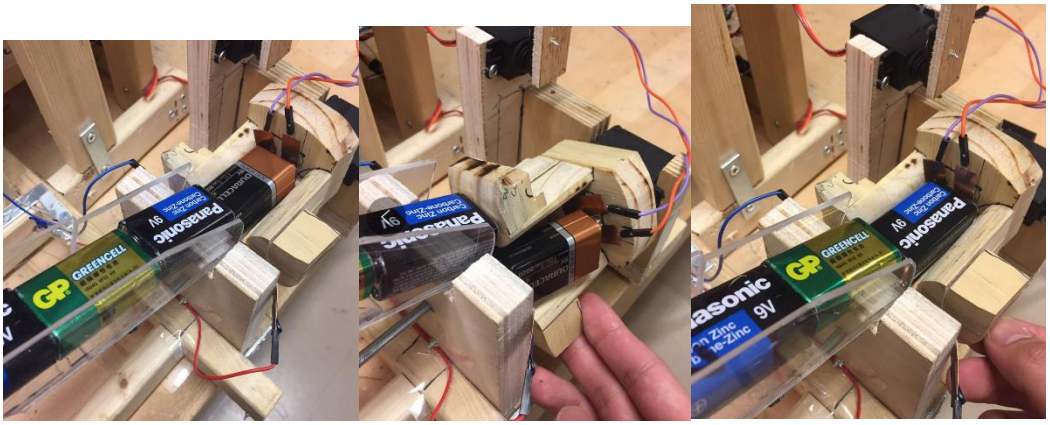


Figure 7.5.3.1(2) voltage measuring process at the measuring box

7.5.3.2 Measuring Design and construction

Voltage measurement mechanism of 9V batteries was different from that of the other two types of batteries due to the difference in geometry and location of terminals.
The measuring area of 9V batteries was designed to be a quadrate box with open top. The battery will slide into the box surrounded by polished wood and stopped by two copper plates with springs attached at the back of the box. Another two pieces of copper plates were fixed under the end of the track. The box was connected to a servo motor for transporting charged or drained batteries similar to the AA and C batteries.

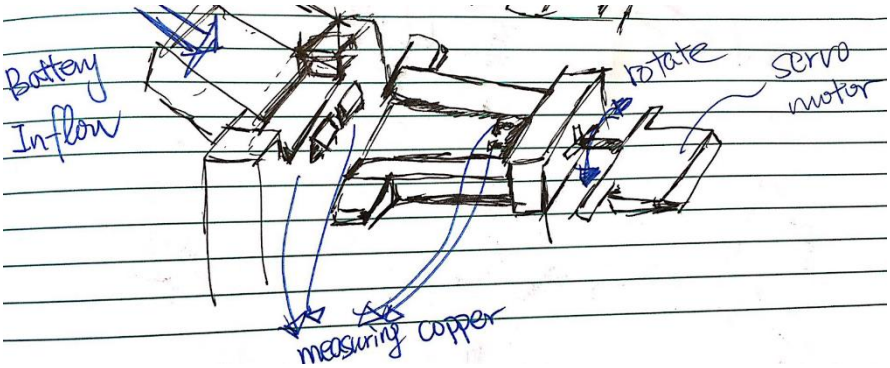


Figure 7.5.3.2(1) construction of measuring box for 9V batteries

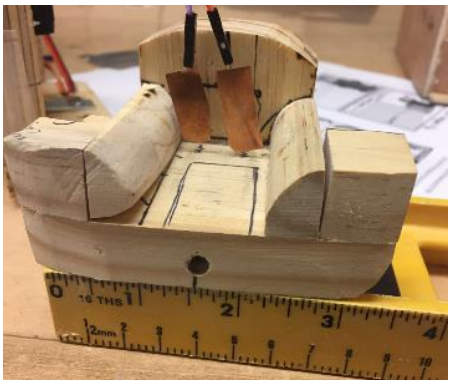


Figure 7.5.3.2(2) dimension of the measuring box

A mechanical beam connected to another servo motor to periodically push down the near-track-end of the battery so that the measuring sticks at either side of the 9V battery get enough connection for voltage-measurement.

Figure 7.5.3.2(3) the effect of mechanical arm acting on the 9V battery

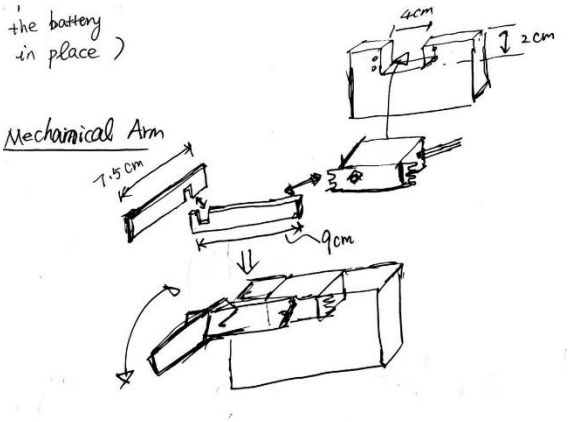


Figure 7.5.3.2(4) Construction of the Mechanical Arm

7.5.3.3 Spring for better connection and flexibility

The Copper Plates in the measuring box for 9V battery had been modified and changed a few times for sake of better connection between the batteries and the copper plate. The team finally decided to use a spring-supported copper plates such that the batteries can be pushed down in place and introduce pressure to the terminal of the batteries.

Figure 7.5.3.3 spring supported copper plate

7.5.3.4 Problems and potential improvements

Several problems were encountered when the team test the performance of the measuring:

- The 9V batteries sometimes lie on the flat track with unexpected side such that when it enters the measuring box, only one electrode of the 9V battery can connect to the measuring stick and fails to measure the voltage.
- C batteries were occasionally mis-sorted to the track of 9V batteries.

Adjustment of angles of the track allows for better accuracy in restricting the orientation for 9V batteries. Microcontroller subsystem also designed a periodic rotation of the measuring box such that batteries cannot be measured will be dropped after some time to allow measuring of other batteries.

7.5.4 Moment of Inertia Calculation

7.5.4.1 Moment of Inertia of shaft of measuring devices

The shaft of measuring stick can be modelled as a cylinder and the calculation shows the moment of inertia for C/AA/9V batteries are 1.8*10^(-5), 7.07*10^(-5) and 3.18*10^(-5) respectively (All units in $kg \cdot m^2$)

	AA	C			9-V
Radius	0.02	0.005		h	0.065
height	0.08	0.08		w	0.05
density	900	900		d	0.035
Volume	0.00010053	6.28E-06		volume	0.00011375
Mass	0.0904752	0.005655		density	900
Moment of Inertia	1.8095E-05	7.07E-08		mass	0.102375
				Moment of inertia	3.17789E-05

Figure 7.5.4.1 Excel table of calculation for moment of inertia

7.5.4.2 Torque needed for servo-motors:

The signal given by the microcontroller member to control the servo motor indicates the servo motor will rotate 90° per 196ms, which equals 8.01 radians per second.

The equation to calculate torque was given by

$$\tau = I \cdot w$$

, where I is the moment of inertia, w is the angular speed.

The results of torque are given below, in units of $kg \cdot cm$

Speed	8.010204082		
	C	AA	9-V
Moment of Inertia	1.81E-05	7.07E-08	3.18E-05
Torque	0.144944963	0.000566191	0.254555525

Figure 7.5.4.2 Excel Table of torque calculation for Servo Motors

7.5.4.3 Torque needed for Mechanical Arm

Another significant part of torque calculation besides the loading conveyor belt is the torque needed for the mechanical arm in the 9V battery voltage measurement area.

Due to the complexity of the situations in the measuring area, several assumptions were made when doing the calculations, and all figures used in the calculation were their maximum number possible.

Assumptions include:

1. The dimension of 9V battery is 5 cm by 3 cm by 2 cm
2. The cross-section dimension of the measuring box is 6 cm by 1 cm

3. The stiffness of the spring used to support the copper plate is 230 N/m (commercial spring used in pens)
4. The situation is ideal and the battery is perfect symmetry for both ends.
5. The mechanical arm is a rigid body and can transform toque without consumption from the servo-motor
6. The result shows that the maximum torque required to push the batteries in place is 2.2 kg cm which is safely under the torque provided by the servo motor the team used.

Detailed calculation steps can be seen in Appendix 4. The result shows a **2.2 $kg \cdot cm$** torque was required.

7.6 Potential Improvements

Other aspects that the electromechanical subsystem could be improved given enough time and under more careful thoughts include:

- The primary sorting mechanism had been elongated by simply gluing another piece of V-shaped track to the existing sorting track. This resulted in inconsistency of the primary sorting mechanism, whose success rate could be improved if the track was remade.
- The Secondary Sorting Mechanism should be better constructed and mounted to achieve higher success sorting rate would be achieved.
- The material of measuring tracks was acrylic sheet with low friction coefficient. However, the surface of this material were damaged due large amount of collisions with batteries fallen from the sorting conveyor belt. As a result, its friction coefficient increased a lot after large amount of testing and increased the possibility of jamming. The material should be changed for sake of robustness to withstand collisions or collisions should be reduced by decreasing the difference in the height of the two parts.
- There was barely enough room for bins to be placed for the sorted batteries. Better planning would need to be done in the early stages of design processes to better allocate space

8. Circuit Subsystem

The circuitry for this machine included circuits for two DC motors, one continuous motor and three non-continuous motors, one voltage divider for power supply, one voltage divider for

measuring the voltage of 9V batteries, and full-wave rectifiers for measuring voltages for all three types of batteries. Eventually all subcircuits were integrated in parallel with the same power input supplying a constant power input of 5V. Each subcircuit is analyzed in detailed below, including necessary calculations. The selection of specific components however, are analyzed in details in a separate session.

8.1 Circuit for the DC motor for sorting conveyor belt

The purpose of this subcircuit was to control a DC motor by sending a 5V signal through the PIC board. The DC motor operated an upward moving, inclined conveyor belt as part of the secondary sorting mechanism. To accomplish this, a simple DC circuit was designed.

The DC motor used is the Shenzhen DC Gearhead Motor (Stright) provided in the project kit. The circuit also consists of a 5V power input, a TIP142 transistor, a 1N4007 diode and a 1.1kΩ resistor in series with the base of transistor. A circuit diagram for this circuit is shown below.

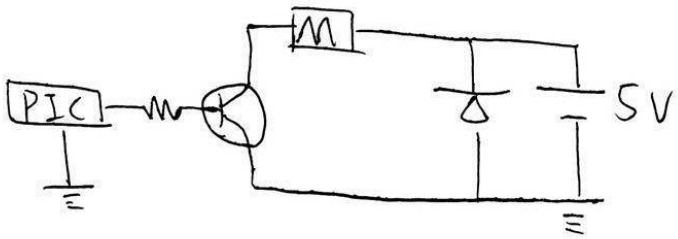


Figure 8.1.1 Circuit for the DC moor for sorting conveyor belt

The transistor was used as an electrical switch, controlled by the PIC board through the base terminal. Since a DC motor is partially an inductor, if the current through the DC motor is switched off suddenly by the transistor, there would still be inductive current flowing through the motor. The diode was then used as a safe path for such current to flow through without causing any harm. Without the diode, the current may flow into power supply and burning it. Similarly, the resistor in series with the base terminal and the PIC board was used to prevent the transistor from overheating. Through research it was found that the maximum rating of current through the base terminal was set to be 0.5A, and very small amount of current flowing into the base terminal was sufficient to send out the signal, thus a resistor with high resistance of 1.1kΩ was used to limit the current into the transistor.

The current flowing through this sub circuit was measured using a multimeter to be around 0.4A during operations. It was also found that the average loaded rpm of the motor was about 78, and the motor had enough torque for carrying at least six different batteries on the sorting conveyor belt.

8.2 Circuit for the DC motor for loading conveyor belt

The purpose of this subcircuit was to run a DC motor for the loading conveyor belt platform. One of the main objective of the loading conveyor belt was to transport batteries slowly and steadily, or even one battery at a time, if possible. The reason is that if one battery enters the sorting conveyor belt at a time, the chances of batteries being mistakenly sorted would be dramatically decreased since the motions of batteries wouldn't be interfered by other batteries. As a result, the rpm of the DC motor was to be minimized.

To achieve that, the DC motor used was changed from Straight to Angled Shenzhen DC Gearhead motor, and reasons would be analysis in the "Electrical Components Selection" Section. Also, in addition to the regular circuit, the 5V power input for the DC motor controlling

the loading conveyor belt was set to be in series with resistors such that the there was a sufficiently low voltage across the motor for it to rotate as slow as possible. However, the voltage across couldn't be too low, or else the DC motor wouldn't rotate at all.

After repetitive testing, it was found that the DC motor has internal resistance of around 10 ohms, and it was decided to use six 10 ohms resistors in parallel, resulting in 1.67Ω of combined resistance for this DC motor. The actual power input across this DC motor was 4.2V. At this voltage, the motor still had a pretty high rpm of 55 without load. However, after 15 batteries were loaded, rpm dropped dramatically to 20, and any voltage lower than 4.2V would result in the motor unable to rotate when loaded. The low input voltage combined with the PIC board controlling the motor to run and stop every one second interval, resulted in a low enough rate of batteries entering the primary sorting mechanism, and made the rest of the sorting process much smoother.

Shown below is the subcircuit design.

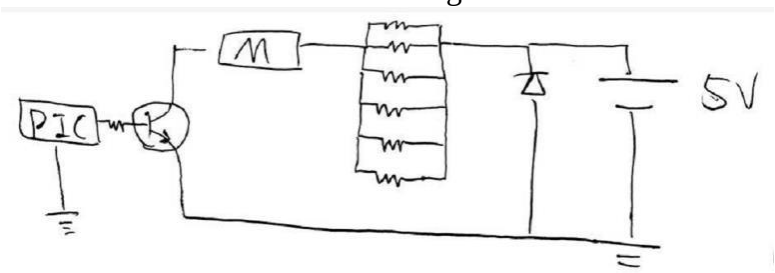


Figure 1 Circuit for the diagram C motor for loading conveyor belt

To find the current flowing in this sub circuit, the voltage across the 1.67Ω combined resistors was found by subtracting 4.2V across the motor from the 5V input to get 0.8V, and current flowing through the combined resistors was found to be 0.479A. The power consumption of each of the six 10Ω resistor was found using the equation $P = \frac{V^2}{R}$ to be 0.38W each. Resistors with power ratings of 1W were chosen for this subcircuit.

8.3 Circuit for servo motors

The circuitry for servo motors is very simple. Each servo motor has three pins, one is connected to the positive power input, one is connected to the ground, and one is connected to a pin on PIC board that was used to transmit signals to control the servo's operation. A diagram of circuitry for servo motors is shown below. Note that the stall current of all servo motors used is 0.8A, meaning that there can be maximum of 0.8A of current passing through the servo subcircuit.

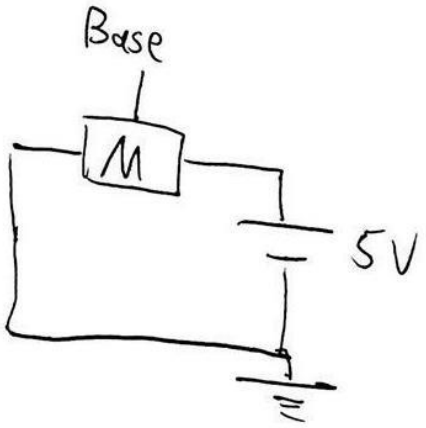


Figure 8.3.1 Circuit diagram for servo motors

8.4 Circuit for measuring voltages - AA/C

The purpose of this subcircuit was to connect to the terminals of batteries to measure the voltage across them, and send such signals to the PIC board.

Due to different possible orientations, the voltage measured across the iron chips could either be positive or negative, but the microcontroller could only recognize positive voltage input signals. Thus full-wave rectifiers were used to convert all negative voltage inputs into positive voltage outputs for all three types of batteries. All rectifiers were made using four 1N4007 diodes.

After testing it was found that the voltage signals received by the PIC board were very unstable, resulting in the PIC board unable to differentiate the signals. To solve the problem, a 1.1k Ω resistor was connected in parallel with each terminal. A diagram for the voltage measuring circuit for AA and C is shown below.

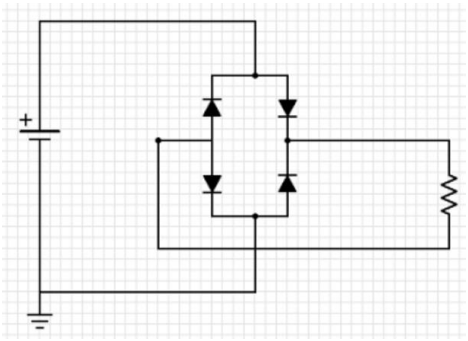


Figure 8.4.1 Circuit diagram for AA/C voltage measuring

The maximum reverse blocking voltage for each individual diodes is no more than 1.5V, the total voltage input.

The voltage across R1, the 15000Ω resistor is 5V, and its power consumption could be calculated to be 0.002W. A resistor with power rating of 0.25W was chosen. The power rating is two degrees of magnitude higher than the actual power consumption. This ensured a safe operation of the circuit.

8.5 Circuit for measuring voltages - 9V

The nominal voltages of AA and C batteries are 1.5V, which is much lower than the safety range of voltage input signal for the PIC board, namely 5V. However, nominal voltage of 9V batteries is 9V, thus a voltage divider was implemented to cut the voltage input by half to 4.5V. The voltage divider consisted of two 550Ω resistors. The complete circuit diagram for the 9V batteries measurement is shown below.

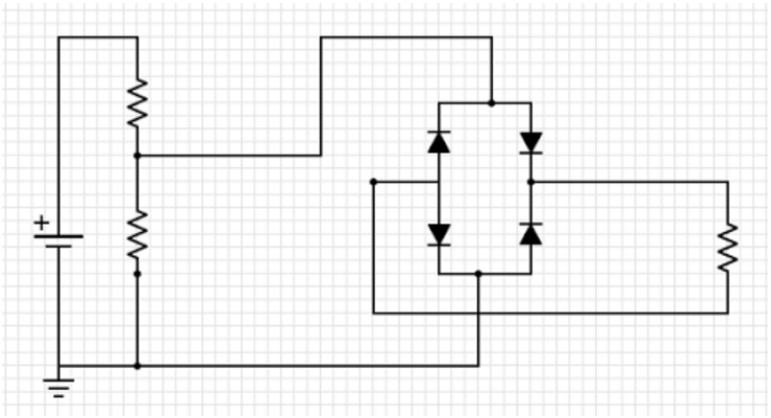


Figure 8.2 Voltage measuring for 9V

The maximum reverse blocking voltage for each individual diode is no more than 9V, the total voltage input.

The resistance of R1, R2 and R3, are 550Ω, 550Ω and 15000Ω, respectively 550Ω. The combined resistance of R2 and R3 is 530Ω using equation $R_{parallel} = \frac{R_1 R_2}{R_1 + R_2}$. As a result, the voltage across R1, R2 and R3 for a 1.5V voltage input are 4.578V, 4.416V and 4.416V, respectively using Ohm's Law. And the corresponding power consumption for R1, R2 and R3 are 0.038W, 0.035W and 0.001W using equation $P = \frac{V^2}{R}$. Resistors of power rating of 0.25W were chosen for this circuit, which is at least one degree of magnitude higher than the actual power consumption. This ensured a safe operation of the subcircuit.

8.6 Circuit for the voltage divider for power supply

The power supply for this machine was needed to supply power to not only the actuators but also the PIC board. The PIC board required a voltage of 12V, as proven by its original power supply

Figure 8.6 Power supply to PIC board proof

On the other hand, the servo motors and dc motors required voltage input of around 5V. Thus a voltage divider was needed. However, through testing it was discovered that the internal resistance of DC motors is not constant, but changes throughout the operation. Thus if the voltage divider was consisted of simple resistors, the voltage across the DC motors would always be changing and so would their rpm, which was not desired.

As a result, two L7805 voltage regulators were used instead, to provide a steady 5V, 2A power input to all the actuators. To prevent voltage regulators from overheating, a heat sink was attached to each of the voltage regulators. The amount of heat dissipated by each voltage regulator into heat sink and ambient was calculated to be $Q = (V_{in} - V_{out})I_{load} = (12-5)V * 1A = 7W$. After a heat sink was added, assuming a case-to-air thermal resistance of 10 °C/W for the heat sink, along with junction-to-case thermal resistance given in voltage regulator's datasheet, the resulting thermal resistance would be (3+10) = 13 °C/W, resulting in 91°C of temperature rise. Assuming ambient temperature of 25°C, the temperature of voltage regulators with maximum current drained is 116°C, which is below their maximum operating temperature, 125°C.

Note that in actual operations, the maximum possible current drained by all actuators was actually $I_{DC,loading} + I_{DC,sorting} + I_{servo} = 0.479A + 0.5A + 0.8A = 1.779A$, at any moment in time, since only one servo could be rotating at the same time(refer to session 9.6.4). At this current the temperature of voltage regulators would be at 100°C. Also the current used for servo motors is the stall current, but in normal operation the current drawn by servos would be about half of their stall current or even less than that.

8.7 Emergency Stop

It was required by the RFP to implement an emergency stop for the machine that is independent from the PIC board. Once the stop is pressed, it should safely cut off power supply for the whole machine completely.

A switch was obtained and the circuit was designed as follows: the positive end of the power supply was connected to the one end of the switch, which is in series with all positive power input of subcircuits, while the negative end of the power supply connected to the ground. A sketch of the emergency stop was shown below, with the bulb representing the actuators for

different subcircuits. A sketch of the emergency stop was shown below, with the bulb representing the actuators for different subcircuits.

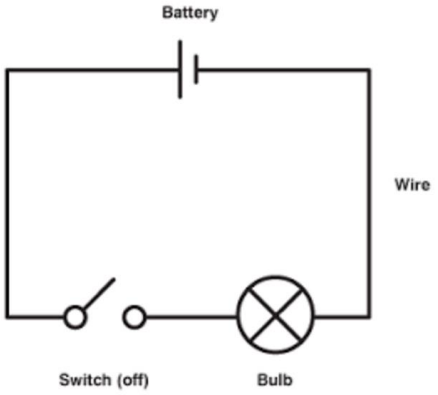


Figure 8.7Circuit for emergency stop

8.8 Electrical Components Selections

8.8.1 DC Motors

Two DC motors were required for the machine, they were used to control the two conveyor belts. The DC motors provided in the project kit were Shenzhen DC Motors (Angled), Shenzhen DC Motors (Straight) and Zheng DC Gearhead Motor. These three types of motors were the first ones to be analyzed by Team 10 to see if they are suitable for controlling the particular uses of the machine, and are denoted as motors #1, #2 and #3 respectively in the following discussion.

From torque calculation in session 7.3.3, it was determined that motors #1 and #2 provide more than enough torque to keep the conveyor belts running at desired rpm, while having much lower power dissipations compare to motor #3. As a result motor #3 was eliminated. Through testing and prototyping, it was determined that the loading conveyor belt required higher torque and lower rpm, since it needed to transfer 15 batteries, and low rpm minimizes the interaction of batteries in motion thus minimizes missorting. On the other hand, the sorting conveyor belt required lower torque since it only needed to transfer five or less batteries during operations, and faster rpm could potentially eliminate the bug where a C gets pushed up the conveyor belt by a 9V behind it. It was found that motor #1 provides higher torque but lower rpm under the same condition compare to motor #2. Also, motor #1 was easier to be mounted for the loading conveyor belt. As a result, it was decided that motor #1 to be used for the loading conveyor belt and motor #2 to be used for the sorting conveyor belt.

The specifications for motor #1 and motor #2 are given in the table below, found from the AER201 Lecture notes.

Motor	**Voltage (V)**		**No load**		**Stall**	
	Operational Range	Nominal	Rpm	Current(A)	Torque (Nm)	Current (A)
#1	3 - 12	3.0	50	0.2	0.393	2.15

#2	3 - 12	4.5	90	0.2	0.14	1.25

8.8.2 Servo Motors.

There were in total of four servos used in this machine: three for each of the three voltage measuring areas and one for the mechanical arm for 9V. The servos included in the design kits were non-continuous HD-3001HB and continuous S4303B. Another smaller, non-continuous servo UMTRS-001053 was also considered to be used. These three servos were denoted as #3, #4 and #5 respectively in the following analysis.

Through prototyping and testing, the difference between continuous and non-continuous motors were analyzed and generalized, as a result it was suggested for the C water wheel to be turned using a non-continuous motor, while using a continuous motor for a water wheel. Detailed analysis of the matter is included in session 10.4.3. Calculation of torque in session 7.5.4.2 (for three other servos) and Appendix 4 (for the servo controlling the mechanical arm) showed the torque required for the four servo were 0.15, 0.0006. 0.25 and 2.2 kg cm, which are all less than maximum torque provided by motor #3 and #4, 3.5 kg cm and 3.3 kg cm. On the other hand, motor #5 was very easy to be damaged when there were being mounted onto the machine. Thus it was decided that motor #3 was used for 9V and C wheel, as well as the pushing mechanism for 9V; while motor #4 was used for the AA wheel.

The specification for motor #3 and #4 are given in the table below. With links to their datasheets included in the Reference session. [13] [14]

Motor	**At 4.8V**		**At 6V**	
	rpm	Torque (Nm)	rpm	Torque (Nm)
#3	3 - 12	3.0	50	0.2
#4	3 - 12	4.5	90	0.2

8.8.3 Transistors

Two transistors were used in this machine, both were used as a switch in the DC motor circuits. Circuit diagram of one of the circuit is shown again for the convenience of the discussion. Both transistors needed to be NPN since a signal would be sent from the PIC board to the DC motor to turn it on/off. A commonly used and easy-to-obtain type of NPN transistor is the TIP142 transistor. The link to the datasheet for TIP142 transistor is included in the reference session. [15]

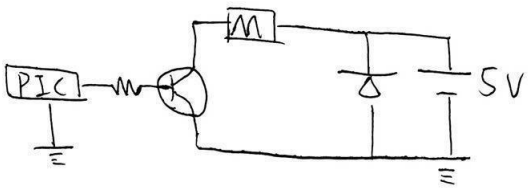


Figure 2 Circuit for DC motor

To calculate the voltage difference between any two of the E, C, and B terminals, circuit member recognized the fact that with voltage input of 5V, the voltage across any of the two terminals would always be no more than 5V. Furthermore, the current entering the Base terminal

were calculated using the 11kΩ resistor in series with it to be 0.45 mA. It is evident that the voltage across and the current passing through each terminal was well below the maximum rating, by at least one degree of magnitude. This ensures that the transistor could be safely operating for a long period of time with no risk of it being damaged, which would then possibly causes damages to the PIC board, power supply and other electrical components. Thus it was decided to use TIP142 transistor as switches for the two DC motors.

8.8.4 Voltage Regulator

In the machine, voltage regulators were used to provide a steady 5V voltage input to all the actuators, at the same time providing a steady 12V voltage input to the PIC board. The only type of voltage regulator that provides 5V of voltage output in both Creation and Home Hardware is L7805 regulator, which has power output of 5V 1A. The datasheet for this model of voltage regulator is included in the reference session [16]. From the circuit calculations in session 10.6, it was determined that less than 2A of current would be drawn by all the actuators. As a result, two voltage regulator were connected in parallel to provide a 5V, 2A power input to all actuators.

8.8.5 Power Supply

After testing with the PIC board, it was found that the PIC board requires 12V of power input, for any power input less than 12V, the LED light would be too dim for users to see. From calculations in session 10.6, maximum of 1.779A of current would be required for operating all actuators. From the original power adaptor of the PIC board (refer to Figure 8.6.1), it could be easily deduced that 1A of current would be drawn for the PIC board to operate. Even though it was advised to use a 1.2 safety factor for choosing the power supply using the calculated power dissipation, such factor was not used, since all the values used in the calculations use the highest power consumption possible, but in normal operations, the actual power consumption of the actuators would be much lower than the calculated values. As a result, a 12V, 3A power supply was obtained and used to power all subcircuits as well as the PIC board.

8.8.6 Diode

Diodes were used in this machine for two purposes: (1) to protect the DC circuits from inductive currents, and (2) to construct full-wave rectifiers for voltage measuring circuit. A package of diodes with different models were purchased and tested individual to find the most optimal ones. The specifications for the different models are shown in the table below. Note that the maximum DC blocking voltages for (1) were the voltage inputs for the DC motors, 5V and 4.2V, and for (2) is the voltage across the particular types of batteries, thus 1.5V for C and AA and 9V for 9V batteries, which are well below the maximum ratings for all models. After repetitive testing, it was found that all 7 models of diodes work perfectly for both (1) and (2). Since all models are very cheap, ranging from $0.13 to $0.04 per unit, it was decided to use 1N4007 diodes since they have the highest maximum ratings among the rest, thus the lowest possibilities of being damaged during operations.

Shown below is the specifications for the different diode models [17].

MAXIMUM RATINGS (T_A = 25 °C unless otherwise noted)									
PARAMETER	**SYMBOL**	**1N4001**	**1N4002**	**1N4003**	**1N4004**	**1N4005**	**1N4006**	**1N4007**	**UNIT**
Maximum repetitive peak reverse voltage	V_{RRM}	50	100	200	400	600	800	1000	V
Maximum RMS voltage	V_{RMS}	35	70	140	280	420	560	700	V
Maximum DC blocking voltage	V_{DC}	50	100	200	400	600	800	1000	V
Maximum average forward rectified current 0.375" (9.5 mm) lead length at T_A = 75 °C	$I_{F(AV)}$	1.0							A
Peak forward surge current 8.3 ms single half sine-wave superimposed on rated load	I_{FSM}	30							A
Non-repetitive peak forward surge current square waveform T_A = 25 °C (fig. 3) t_p = 1 ms	I_{FSM}	45							A
Non-repetitive peak forward surge current square waveform T_A = 25 °C (fig. 3) t_p = 2 ms	I_{FSM}	35							A
Non-repetitive peak forward surge current square waveform T_A = 25 °C (fig. 3) t_p = 5 ms	I_{FSM}	30							A
Maximum full load reverse current, full cycle average 0.375" (9.5 mm) lead length T_L = 75 °C	$I_{R(AV)}$	30							μA
Rating for fusing (t < 8.3 ms)	I^2t (1)	3.7							A^2s
Operating junction and storage temperature range	T_J, T_{STG}	- 50 to + 150							°C

Note
(1) For device using on bridge rectifier appliaction

Figure 8.8.6 Diode models

8.9 Subsystem Possible Improvements

8.9.1 Follow proper soldering procedures

As a beginner in soldering, it was very important for the circuit member to strictly follow the proper safety soldering procedures online. However, such procedures were not followed since they are too time consuming. As a result, lots of inconvenience was caused, such as short circuiting due to excess terminals not cut away; short circuiting due to exposed joints, not warped by heat wrap which should have been performed. It turns out that it is less time consuming if the procedures were followed as more time would be saved from debugging.

8.9.2 More organized circuitry

A lot of small techniques could be used to make the entire circuitry for the machine more organized, and easier to be debugged. Some of the techniques include using the same colour of wire for a subcircuit, or carefully labeling the subcircuit each wire belongs to; position wires to be on surfaces of soldering boards instead of hanging in the air, if possible [Figure]; allocate the paths of long wires such that multiple wires could be braided together for aesthetic and more efficient use of space.

9. Microcontroller Subsystem

9.1 Assessment of the problem:

The problem is to recycle batteries and categorize batteries into four types: 9V, AA, C, and uncharged battery. For microcontroller subsystem, the goal is to control different parts of the machine operating according to the circuit and electromechanical specifications.

The following topics are going to be discussed in detail.

- Processor choice
- Programming language
- Control method
- Overall flow of operation
- Detailed implementation
- Standards for determining charged or uncharged
- Specific pin assignment on PIC board
- Potential improvement

9.2 Processor choice

Several options were available for choosing the processor. CPU was considered as a poor choice of processor for this project because of its poor heat dissipation ability due to its small size; a professional ARM processor was also seen as not suitable due to budget constraints. Because the control mechanism of this battery recycling machine is simple and the cost requirement of the RFP is rigorous, the microcontroller satisfies the need for low cost operation. After consulting with Professor Emami, arduino family and PIC family remained in the candidate pool. In the end, PIC development board and PIC18F4620 chip were chosen to be the microcontroller part for this project, with the detailed reasoning shown below.

Some criteria for choosing the processor was considered and applied. The processor should be accompanied by a supplement device to assist heat dissipation. The microcontroller part should also have the ability to simulate the program outputs. This functionality will greatly simplify the debugging process. PIC Development Board has a debugging module which can achieve this goal. In addition, microcontroller can handle digital to analog conversion for input and output without external accessories. It also has a small footprint which is a huge benefit due to the limitation on the size of the machine. The real time clock(RTC) on the PIC board is also convenient for the project. Plus the consideration of the memory storage, PIC18F4620 was selected because this chip possesses relatively large memory storage to ensure the program can be completely loaded onto the chip.

9.3 Programming Language

For PIC18F4620 from Microchip, C and assembly programming language can both be utilized to program the chip. Microchip provides users with relatively reliable compiler for programming C onto the PIC18F4620 chip. This feature is convenient for this project because C requires less learning curve than assembly language. All circuits in the machine were self-designed and self-assembled, bugs were prone to occur. Although the recycling process is simple

for individual battery, there are three subprocesses, due to three kinds of batteries, requiring many available pins from microcontroller chip. Due to the large amount of pins required, easiness to debug was a critical factor that was considered in choosing the programming language.

Compared to assembly language, C language is more readable and more similar to high level programming language. With the xc-8 compiler provided by Microchip, low level implementation such as interrupt can be easily achieved in C language. With a more comprehensive library that C has, it should be more convenient to manipulate data structure like strings. As mentioned above, C code are generally easier to debug and allow for extra time to work on other parts.

Given the above considerations, C language was chosen in this project.

9.4 Control Method

All actuators are controlled by signals from the PIC Development Board. Actuators include two DC motors driving two conveyer belts, three servos driving three water wheels at three voltage measuring boxes and one servo controlling one mechanical arm at the 9V battery voltage measuring box.

9.4.1 DC Motor for first conveyer at the top

The low possibility of jamming is one of the advantages of our battery recycling machine. When a relatively large number of batteries, more than four, entered the primary sorting mechanism at the same time, jam was prone to occur. This problem was solved by controlling the top loading conveyer belt to turn periodically. Each turn would last for 0.3 second and would move batteries on the loading conveyer belt forward by a small distance. An average of two centimeters per run was finally achieved on the conveyor belt. Due to the width of the conveyer belt, at most three batteries will fall into the primary sorting mechanism. This avoids jamming in the primary sorting mechanism and increases the success rate of the operation.

9.4.2 DC Motor for second conveyer belt separating 9V and C battery

The second conveyer belt ran at a constant speed because it was supposed to carry the 9V batteries upwards and let C batteries roll down dynamically. Thus, this conveyer belt was made to run constantly until the end of the operation.

9.4.3 Three water wheels, Comparison between continuous and non-continuous servo motors

When choosing continuous and non-continuous servo motors, the main factor to consider was the accuracy of the rotation. For continuous servo motors, the time and the speed of the rotation were manipulated to control the angle of rotations. For example, the speed of the rotation was controlled by the averaged voltage height received by the servo motor.
In contrast, the non-continuous servo motors can rotate an accurate angle because the angle of rotation is strictly related to the averaged voltage height received by the servo motor. Thus, if the accuracy of the rotational angle is a crucial element in operation's success, non-continuous servo is preferred.

For continuous servo motors, the rotation can be continuous so that the process is faster than the non-continuous servo motors, because for non-continuous servo motors, the motor has to first rotate by 90° in one direction to move the battery out of the box and rotate back by 90° in

order to take the next battery. This process was clearly illustrated above in the electromechanical subsystem.

For 9V batteries and C batteries, the size of the batteries is relatively large. The size of the leaves on the rotary axis is larger than that of AA battery. Accuracy of rotary angle iss required for measuring voltage because of the design of the voltage measuring boxes. Because if a small angle of rotation is missed, it will cause a relatively large displacement of arc length. With multiple turns, this displacement will accumulate and will result in potential failure of the operation. Thus, non-continuous servo motors were selected for 9V battery and C battery voltage measuring boxes.

Since the size of the AA battery is relatively small, the size of the leaves in the AA voltage measuring boxes was also designed to be smaller, causing a smaller error in the displacement of the arc length given the same rotary angle. Thus, continuous servo motor was selected to optimize the operation by making this sub-process faster.

The different choice of the servo motors resulted in different designs of the water wheels in the voltage measuring boxes. For AA battery voltage measuring box, the wheel was chosen to be a crossing water wheel. For C battery voltage measuring box, the wheel was designed as a similar wheel. For 9V battery voltage measuring boxes, the wheel was made as a measuring box, with its shape specifically design for 9Vs.

9.4.4 Push arm at 9V battery voltage measuring box.

Because the design of the voltage measuring pins in the 9V voltage measuring box has moderately tensile springs underneath, it often required small force to push the battery into the position so that voltage can be detected. Thus, a mechanical push arm was designed to accomplish this goal.

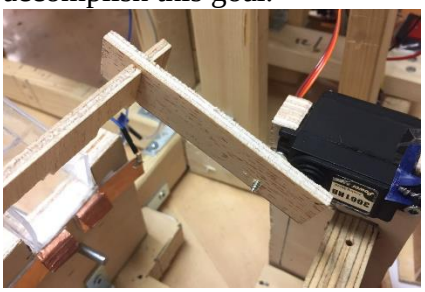


Figure 9.4.4 Mechanical Arm for 9V battery voltage measurement

The voltage readings from the three voltage measuring boxes are read every second, which will be discussed in detail in the following section, 11.6. The mechanical push arm will push down every time before reading the voltage from 9V voltage measuring pin and will raise the push arm after reading, whether there is voltage detected or not. This will guarantee the push arm will not block the subsequent batteries sliding into the voltage measuring box, and also guarantee that the mechanical push arm will not block the way of the measuring box when it rotates.

9.5 Overall flow of operation

The detailed operation flow can be visualized by the flowchart in the Appendix 2. The flow is monitored by the states of the operation. There are three states of the operation: standing

state, which is also called finished state, running state, and paused state. An emergency stop was also designed which stopped the operation once the button was physically pressed. Different actions can be performed in different states. Several extra features were also included such as real time clock(RTC).

9.5.1 Standing / finished state:

Actions that can be taken in this state include starting the operation, checking results from last run, checking history of the last four runs. This state is the default state when the PIC board is turned on. This state can also be reached by pressing key C, or waiting the operation to finish automatically.

9.5.1.1 Starting the operation

When the key A on the keypad is pressed down, the operation starts. The PIC will send signal to the transistor which is in charge of the power supply of the whole circuit and the whole circuit will be powered after this. Although all the circuit element will be powered up, they still need signals to control their operations. Transistors were used for these control signals to function.

9.5.1.2 Checking result from last run

When key 1 is pressed down, the first line of the LCD will display the result of the last run line by line. An example is provided below.

The result is stored in six lines in the following form:

Uncharged: # of Uncharged (8 in this example)
Nine Volt: # of Nine Volt batteries (3 in this example)
AA Batteries: # of AA batteries (1 in this example)
C Batteries: # of C batteries (3 in this example)
Total count: # of total count (15 in this example)
Optime: # of seconds of operation time (2 minutes and 10 seconds)

If there were last run, pressing key 1 for the first time will give the following LCD display:

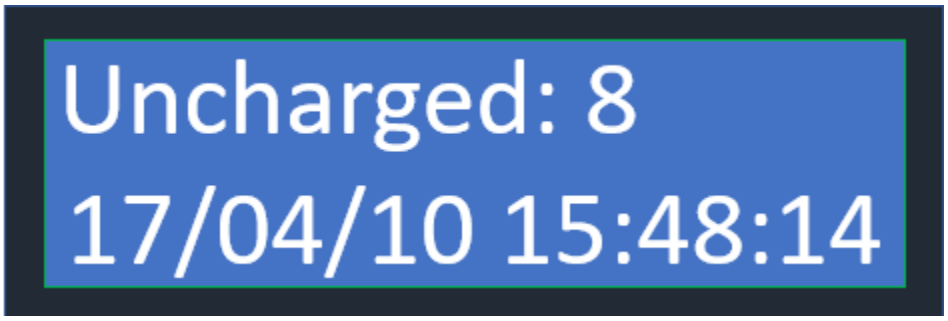


Figure 9.5.1.2(1) Uncharged Result

Pressing key 1 for the second time will give the Following LCD display:

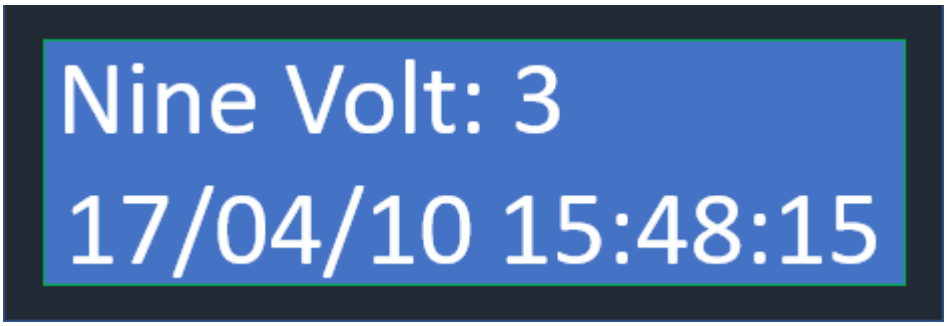


Figure 9.5.1.2(2) 9V Result

Pressing key 1 for the third time will give the Following LCD display:

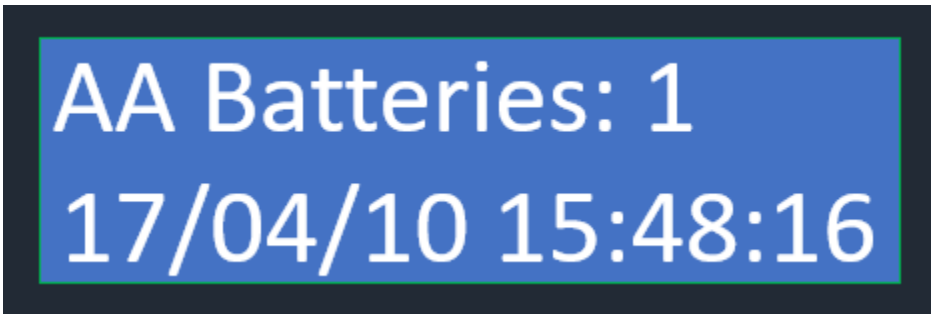


Figure 9.5.1.2(3) AA Result

Pressing key 1 for the fourth time will give the Following LCD display:

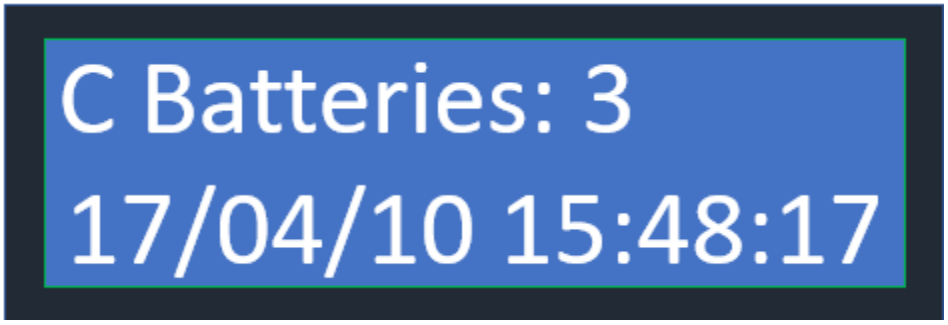


Figure 9.5.1.2(4) C Result

Pressing key 1 for the fifth time will give the Following LCD display:

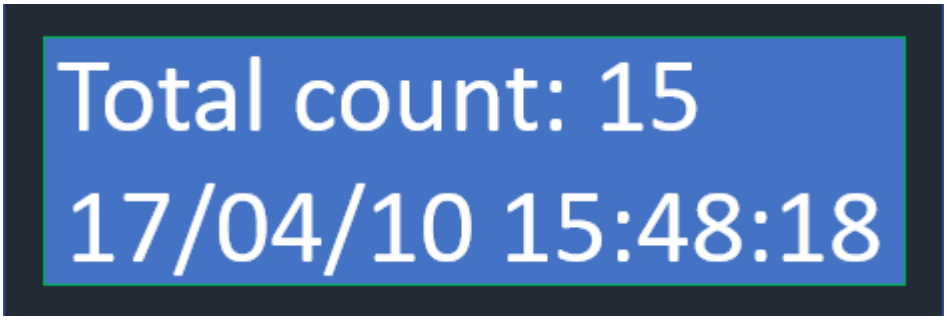


Figure 9.5.1.2(5) total count Result

Pressing key 1 for the sixth time will give the Following LCD display:

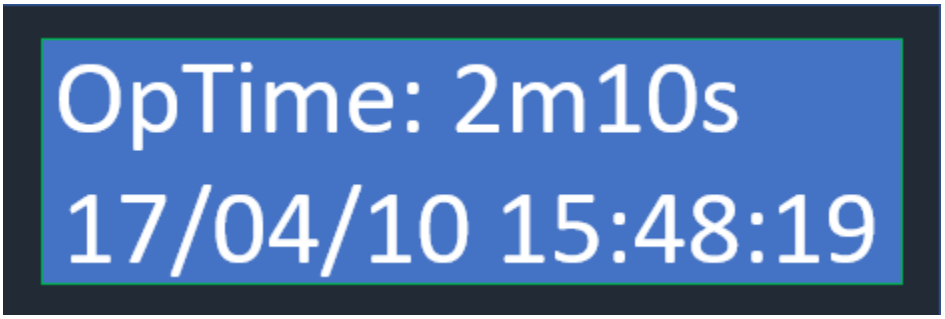


Figure 9.5.1.2(6) Operational Time

Pressing key 1 for the seventh time will give the Following LCD display:

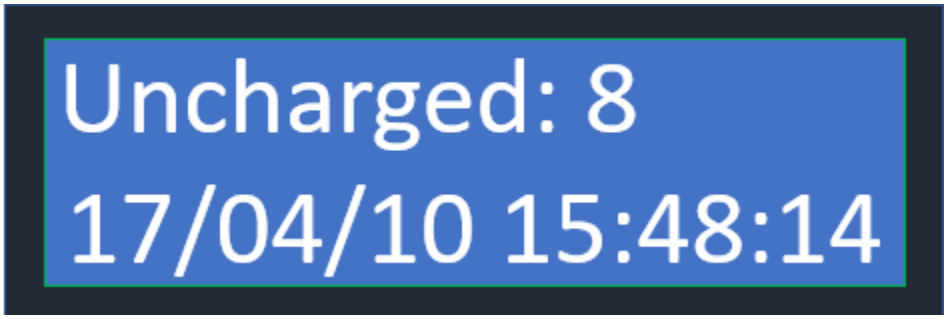


Figure 9.5.1.2(7) Uncharged Result

As shown in the example above, pressing the key 1 will display the result of the last run line by line and repetitively. If key 1 were pressed more than 6 time, the LCD will simply display the result of the last run again.

If there were no last run, the LCD display will be the following no matter how many times the key 1 is pressed.

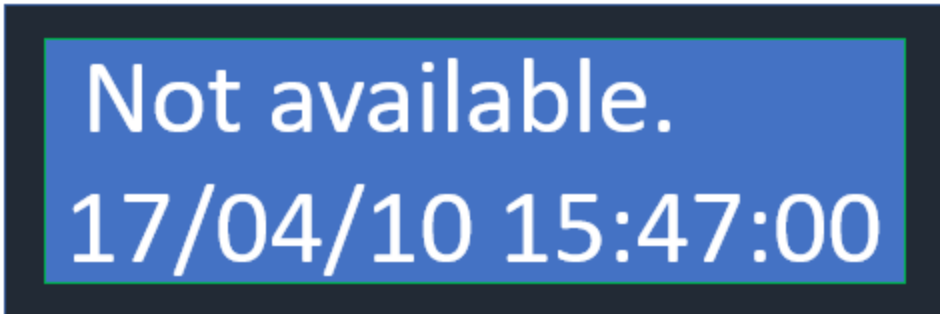


Figure 9.5.1.2(8) Not available Result

9.5.1.3 Checking history of the last four runs

When key 2 is pressed, the history of the last four runs will be displayed in the first line of the LCD line by line to the similar approach as displaying the result. There are maximum 4 past runs stored. When first pressing the key 2, text "History: " will be displayed on the first line of LCD to indicate the history is being displayed. The run indexed by the largest number is the latest run. For instance, if there is history of 3 runs, the third run is the latest run, and the second run is the second latest run. The following is an example of how history will be stored in the PIC board.

History:
Run#: 1
Uncharged: 5
Nine Volt: 6
AA Batteries: 2
C Batteries: 1
Total Count: 14
OpTime: 1m20s
Run#: 2
Uncharged: 4
Nine Volt: 2
AA Batteries: 3
C Batteries: 1
Total Count: 10
OpTime: 0m46s

In this example, the second run labeled by "Run#: 2" is the latest run, and the first line of LCD will display one line at one time starting from "History: ". If the bottom line is reached and the key 2 is still pressed, the whole history will be displayed again, line by line in the same way it is displayed the first time.

9.5.2 Paused state:

This state is entered only when the user presses key B during the running state. This state was designed for the user to fix some easy problems during the operation. Two actions can be taken in this state. The user can choose to resume to the original run after fixing small problems, or the user can choose to manually terminate the operation by pressing key C.

9.5.3 Running state:

This is the state when the machine is running. Measuring voltage, counting the number of batteries of each categories, displaying the time and calculating operation time are all achieved in this state. Two activities can occur in this state. The user can choose to pause the operation by pressing key B, or the user can manually terminate the operation by pressing key C.

9.6 Detailed implementation

All program code is included in Appendix 9. Global variables were used for essential communications between different functions, because this program is small and is not used in conjunction with other program. Due to the limitation of the time requirement of the project, the program must be easy to debug as discussed in earlier section. The choice of using pointers to communicate between functions was abandoned because the null pointer and wrong address allocation bugs are prone to be very time-consuming. Therefore, global variables were used for communicating among functions in the program.

In order to maintain the readability and maintainability of the code, the names of variables and functions were carefully chosen so that no excessive comments were required to make the code readable to a third person who would have no previous knowledge of what the code was trying to accomplish. In addition, there were descriptions for almost every function. Many implementation details were considered during the actual coding process. However, only major considerations are discussed in this section. The major sections include user interface, measuring voltage, controlling DC motors, controlling servo motors and setting real time.

9.6.1 User interface

A long string was chosen to contain all results of the last run. This decision was made because of the design of the user interface. For usability, the microcontroller member decided that it would be best to minimize the number of keys required to interact with the machine. Therefore, the result is displayed on one line of the LCD display and the user only needs to press one key to view all the results. This was designed for the user to view all results by pressing one key several times. A long string was considered to be the best choice to contain all the result information. Each line of the result is separated by a new-line operator and the program just needs to keep looping through the string when the user requests to view the result. A global index was also created for keeping track of which line had already been displayed.

A user manual was chosen to be displayed by default in the same way as the result. This idea was introduced by teaching assistant Nathan. This design will assure that the user can easily use the machine even without the hard-printed manual.

As discussed in previous section, the history was also chosen to be displayed in the same way for the purpose of simplicity and consistency.

9.6.2 Measuring voltage

Due to the observation from the survey process that the reading would fluctuate when the reading pins are in open air. It was found in contrast that when there were batteries connected between the pins, the reading was stable as long as the battery remains stable between the voltage measuring terminals. If the battery were removed and placed back again, the reading would be a little different from the previous reading even for the same battery placed in the same voltage measuring box in the same orientation for a second time, however, this slightly different reading would remain stable as long as the battery was not moved.

This observation helped the microcontroller member find a method to detect the existence of batteries in the voltage measuring box. The microcontroller chip PIC18F4620 would read from the same channel for three consecutive times. If these three consecutive readings were the same, then it was determined that there was a battery in the voltage measuring box, whether charged or not. By analyzing the digital numbers converted from the analog input, program could determine if the batteries being measured was a charged battery or not.

However, this method was found to be ineffective after integrating with the circuit due to interference. The improved approach method which can successfully deal with stable voltage measuring will be discussed in detail in subsequent integration section, Section 12, Integration of individual subcircuits and PIC Board.

9.6.3 Controlling DC motors

From the experiments and online researches, it was found that DC motor only requires constant voltage and current to rotate. Therefore, it was determined to use transistors which receive signals from the PIC Development Board to control DC motors. This decision avoided driving DC motors directly from the PIC board, which could damage the PIC Development Board.

Later in the project before the integration, it was decided that periodic movement of the loading conveyer on the top would increase the accuracy of the operation. Changes were made in the code around the display time function so that the first conveyer belt will move one second for every two second.

This amended approach was also found not effective enough. The final approach will be discussed in the integration part of this report as it was spirited by the process of integration. Although there are several different versions of controlling DC motors, the main idea of the control is to simply set the pin high and let the transistor receive the signal. Thus, the PIC Development Board controls the DC motor by means of transistor.

9.6.4 Controlling servo motors

There are two PWM output which can be utilized to control two sets of servo motors simultaneously. However, this specific design required four servo motors and they are all operating independently. Two output pins for controlling servo motors are obviously not enough. During the early process of design, the microcontroller member found that besides using PWM to control servo motors, manually setting the pin high and low could also control the motor to turn as if the servo motors were receiving PWM signals. Because there were enough pins available, the microcontroller member decided to manually control the servo motors. The result was that only one servo will operate at one time, as manual control is achieved with the assistance of delay function and when delay function was called, nothing else could be done until the end of the delay function, except the key interrupt.

9.6.5 Setting real time

The microcontroller member learnt how to use the sample code to set the initial time immediately after finishing the user interface. However, the microcontroller member understood that the initial only needs to be set once later.

After setting the initial time, the function of setting initial time must be removed, otherwise, the time will be reset to initial time every time the PIC board is turned on. The microcontroller member achieved this by first loading the code with the function of setting initial time onto the board. Let the program set the initial time, then comment out the part in the codes which have the time-setting function. Finally, the codes were loaded without time-setting function onto the board to overwrite the previous code.

9.6.6 Preparing and process result and history

As mentioned earlier, the result and history were all stored in the form of a long string for the simplicity of implementation and maintainability. Several helper functions were written for preparing and processing the result and history. The output of the preparation and processing is the string which is ready to print. This design decision made the program easy to debug because the other part of the program only used the string, which was the output of the processing. Thus, the modification of the preparation and the processing does not affect other part of the code. This resembles the modularity of other programming language like Java and Python.

9.7 Standards for determining charged or uncharged

If a battery is of the voltage 85% more than its full voltage, it is considered as a charged battery, otherwise, the battery is considered as an uncharged battery. Because of the resistance of different elements in the circuit, the actual value which is received by the PIC board will be smaller than the value measured directly from the battery. Therefore, for every different circuit, the digital value representing the same voltage may be different and need to be experimentally determined.

In the calibration process, the relationship between the voltage and digital number shown on the PIC Development Board was assumed to be linear. Experiments were done to collect data and predict the linear relation between the value on PIC Development Board and the actual voltage the battery possesses. Some processing was done on the data points before plotting the linear fit. Because there are multiple possible orientations for batteries and values obtained from the same battery but different orientations were different, the values shown on the PIC Development Board were averaged before they were used to plot.

The plots based on the collected data are listed below with raw data listed in the Appendix 7.

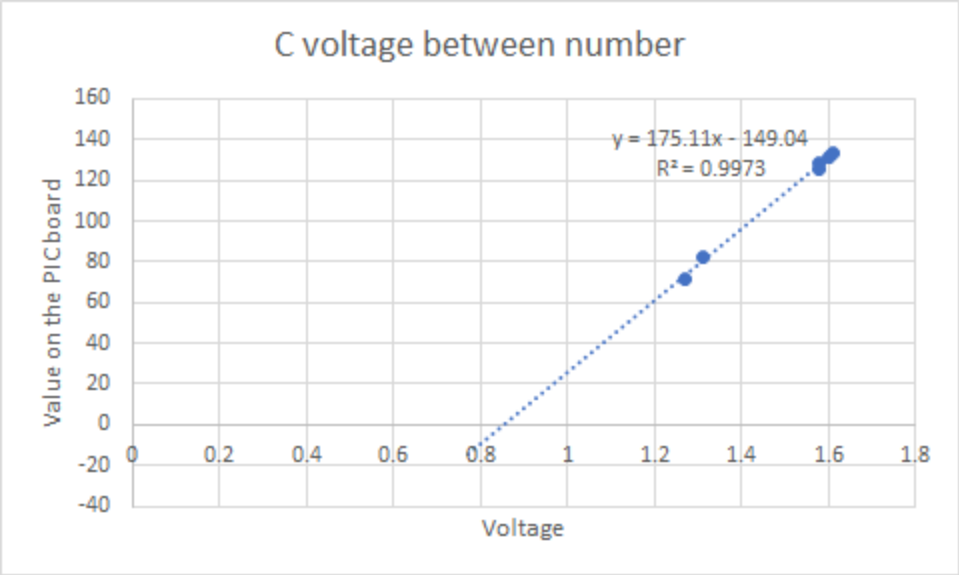


Figure 9.7(1) Plot for C battery

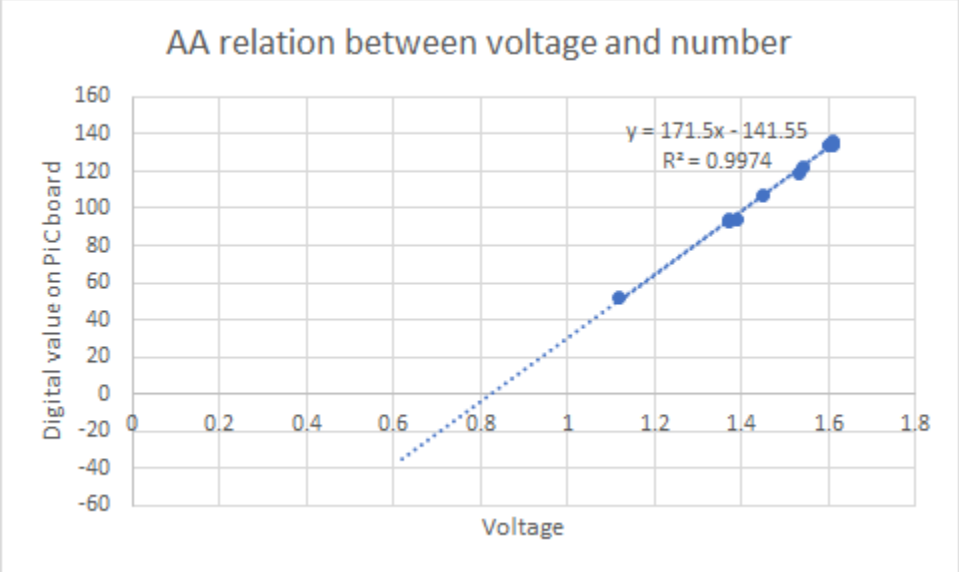


Figure 9.7(2) Plot for AA battery

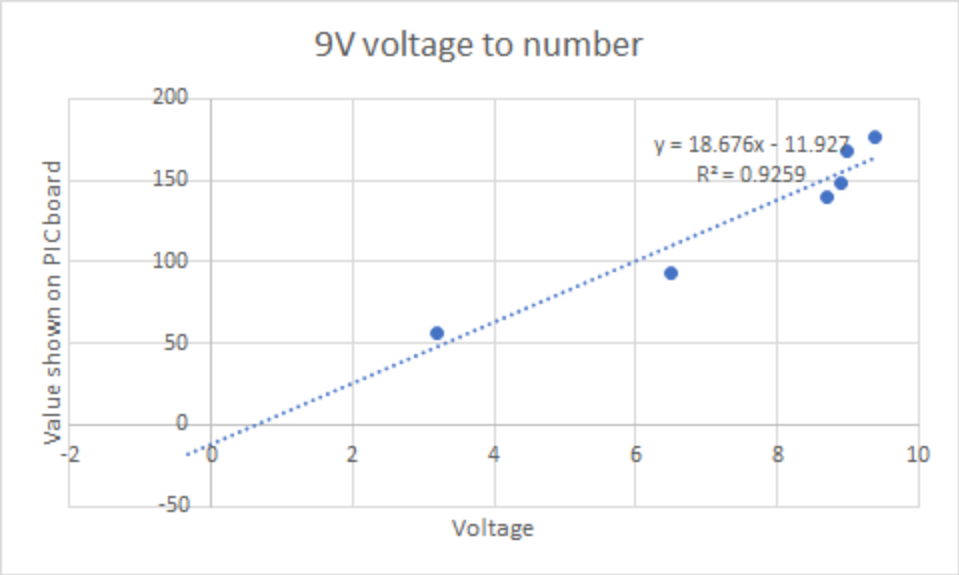


Figure 9.7(2) Plot for 9V battery

9.8 Specific pin assignment on PIC board

Table 9.8: C battery

Pin label	Pin function
RB1	Keypad interrupt detection
RB4,5,6,7	Keypad input reading
RA1	Voltage reading input from 9V battery voltage measuring box
RA2	Voltage reading input from C battery voltage measuring box
RA3	Voltage reading input from AA battery voltage measuring box
RC0	Turning signal for the second conveyer belt which separates 9V and C battery
RC1	9V voltage measuring box servo motor turning signal
RC2	C voltage measuring box servo motor turning signal
RC3,4	Reading input from external real time clock
RC5	AA voltage measuring box servo motor turning signal
RC6	Periodic signal moving the loading conveyer belt forward by a short distance
RC7	Pushing down and raising up signal for push arm at 9V voltage measuring box.

9.9 Potential improvement

The final version of all operation code was included in one single main.c file, with several helper functions placed in the same folder as the main.c file. These helper functions were given in the sample code, so all code specifically for this project was put into the main.c file, which resulted in the large main.c file consisting of 946 lines.

This is the part where improvement might be needed. Code related to one process of the machine should be grouped together and should be put into the one file, separately from the main file. In the main file, it should only call the functions from the other files so that the main program will be short and the logic will be clearer.

10.Integration and Testings

In the following sessions, the different stages of integrations are listed in chronological order, any problems arose from integrations are described in details, along with their solutions under the subheading "Integration". There were also problems that arose not from integration itself, but from the extensive testings performed after the systems were integrated. These problems and their solutions are also included in the following session under the subheading "Testings".

In the context of this project, the circuit subsystem acts as an intermediary between the microcontroller subsystem and the electromechanical subsystem. To integrate the latter two subsystem, they would need to be connected by the circuits to operate the actuators with the PIC boards. Thus for integrations, it is more logical to discuss the integrating process between microcontroller and circuit, and then discuss the integration between electromechanical subsystem and the already integrated circuit-microcontroller system.

10.1 Integration between microcontroller and circuit subsystem:

10.1.1 Integration

10.1.1.1Integration of individual subcircuits and PIC Board

- The PIC board was connected to the individual actuators subcircuits on the solder boards. Then it was tested to see if the operations of the actuators could be started/stopped by pressing "A"/"C" on the keypad. This process was performed smoothly, with all actuators controlled by the PIC board as intended using the subcircuits on the solder boards.

- For the AA/C/9V voltage measuring circuit, corresponding batteries were connected to the individual circuit inputs to see if the PIC board could actually pick up the voltage signal from the batteries.

    - Problem: the voltage signals received by the PIC board fluctuates a lot, even though the battery's position is stabilized. This bug was unexpected since it didn't appear in prior testings of both individual subsystems, and would only appear after the two subsystems were integrated.

- Solution: After repetitive testing and trial and error, it was discovered that the voltage signal would become steady if a resistor was connected in parallel with the output terminals. Moreover, voltage signals detected by the PIC board would drop dramatically if the resistor connected has small resistance. This is due to the internal resistances of the diode used in the circuit. To minimize the voltage drop, the resistance of the resistor was set to be very large at 15k. This way, the internal resistance of diodes would seem insignificant compared to the resistance at the terminal, thus there would be very little voltage drop between the actual voltage of the batteries and the voltage measured by the PIC board.

10.1.1. 2Using the power supply for the PIC board

- The circuit was used to power the PIC board, using the same power supply that supplied power to all the actuators. Before this process, the PIC board was powered using Shoujun's laptop.

    - Problem: Before the integration, the power supply was considered to power the actuators only. As a result, the power supply for the circuit subsystem was set to be 5V, 3A. However, it was discovered that voltage input of 5V was insufficient to power the PIC board, as the LED interface of the PIC board would be too dim to read.

    - Solution: The power supply was changed to 12V, 3A, and a voltage divider circuit was needed to lower the voltage input to 5V for all the actuators. Also, a 2.1mm DC Barrel (M) Wire Assembly was obtained so that the negative and positive end of power input to the PIC board could be separated and conveniently connected to the circuit.

10.2 Integration of all three subsystems

10.2.1 Integration

10.2.1.1 Mounting/fixing actuators/parts

- 2 DC motors were mounted at corresponding conveyor belts. Jumper wires were soldered to the terminal of DC motors for later connection.
- Servo motors were mounted at corresponding voltage measuring areas for later connection
- Stabilize all solder boards and power supply on the physical structure of the machine, as required for the "Team Evaluation - Integration"

    - Problem: After all solder boards were stabilized, one of the DC motor circuit appeared to be short circuited. After detailed investigation, the cause was found to be the excess terminals of components on the back of the solder board (as shown in the Figure below). Originally these terminals were not touching each other. However, after the boards being fixed on the machine, there terminals were pressured and thus accidentally touched each other. Fortunately, none of the electrical components were damaged due to short circuits.

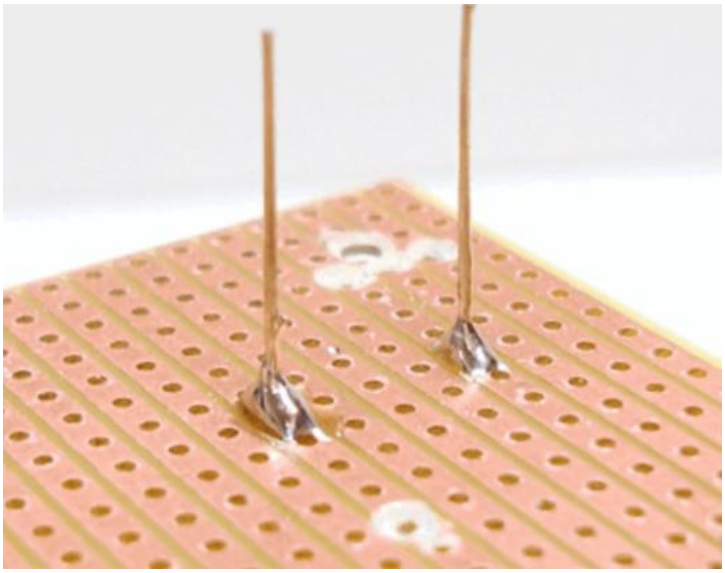


Figure 10.2.1.2.1 Excess terminals from soldering

- Solution: All excess terminals were cut using a pair of sharp pliers and carefully examined to ensure no possibility of short circuiting. In addition, all exposed wire to wire soldered joints were covered using electrical types. These are actually necessary soldering steps that needed to be followed every time after soldering a joint, yet they were overlooked by the inexperienced circuit member Shen until a safety hazard was found.

10.2.2 Organizing Wires

- Wires travelling across the machine were taped along the frames of the machine to prevent tangled wires. Wires from the same subcircuit were braided to prevent interferences between subcircuits and make the machine more organized and aesthetically appearing.

10.2.2Testings

10.2.2.1Testing the actuators and their voltage dividers

- The new 12V, 3A power supply was tested along with the voltage dividers for all actuators.

  - Problem: Initially the voltage divider was consisted of various resistors, and a divided steady voltage of around 5V input was reached when checked using a voltmeter. However, after testing it was found that all actuators would not have a constant rpm during operation without load. Instead, sometimes the actuators would operate normally as if there was 5V of voltage input; other times under the same conditions, the actuators would randomly start rotating rapidly for a period of time, during which times the voltage across the actuators were measured to be around 12V. There was no sign of what caused this bizarre phenomenon, and it

was impossible to predict when such phenomenon would happen. After thorough investigation, there was no sign of bugs in the circuit nor in PIC board's code.

- Other design teams also experienced similar situations, and the most plausible explanation for them was that the internal resistance of actuators are not constant, but fluctuate between a certain range. This has an effect on the resistance ratio of the voltage divider, thus the voltage received by the actuators would also fluctuate.

- Solution: To fix the problem, the voltage divider was changed to be made of voltage regulators instead resistors. Unlike the latter design, a voltage divider supplies a constant voltage output regardless of resistances in the circuit, thus the alternating internal resistances would not have an effect on actuators' performances. Indeed, the problem was solved with the usage of voltage regulators.

10.2.2.2Testing the loading platform

- The performance of the loading platform was tested repetitively with various amount of batteries being loaded on the platform.
  - Problem: During the testing, it was recognized that the torque provided by the DC motor was enough to carry the conveyor belt with more than 15 batteries sitting on it. However, sometimes the friction between the shaft and the belt was not large enough and may cause motor idling.

  - Solution: This problem was resolved by wrapping another sheet of shelf liner around the insulation wrap of the shaft. This would increase the friction coefficient between the shaft and the belt to prevent sliding between the two.

Figure 10.2.2.2 Another Wrap to prevent sliding

  - Problem: Originally, the microcontroller subsystem controlled the loading conveyer belt to move every one or two second. Sometimes, the loading conveyer belt would transport many batteries into the primary sorting mechanism at one time in that one to two seconds interval. This would likely lead to the jamming of the primary sorting mechanism and would very likely cause errors in the secondary sorting mechanism.

- Solution: The microcontroller subsystem changed the controlling method to let the loading conveyer belt to move forward for 0.3 second every one or two seconds. This made sure at most three batteries could enter the primary sorting mechanism at one time.

10.2.2.3Testing the voltage measuring mechanism

- The copper plates used in voltage measuring mechanisms were replaced by copper coils and tested for stable contact between the coils and batteries' terminals. The copper coils were made from stranded wires into coil shapes to have optimal contact with batteries' terminals.

Figure 10.2.2.3 Measuring coils

  - Problem: After testing, it was found that there was always a significant voltage drop for voltage measured by the PIC board. This was not only due to the internal resistance of diodes and other components in the circuit, but also poor connection between coils and batteries' terminals.

  - Solution: Such voltage drop was nearly impossible to avoid. But an experiment was performed to find out the relationship between the voltage of the battery being measured and its measured value on the PIC board. It was found that their relationship is linear positive, and was thus taken into account by the PIC board when using the digital value obtained to determine whether the battery is charged or not.

  - Experiment data: The raw data and subsequent analysis are discussed in detail in section 11.7.

- The performances of water wheels and voltage measuring box driven by servo motors were tested.

- Problem: The water wheel was required to rotate clockwise and counterclockwise exactly by 90° to transport the charged and drained batteries into separate bins. This required the codes from PIC board to be very specific to ensure that the servo always rotate exactly 90°.

- Solutions: repetitive testings were performed in a trial and error matter, until the PIC boards codes were adjusted to the correct value for the servo to always rotate to the desired degrees.

- Problem: Moreover, since C batteries are relatively heavy, the collisions between adjacent C batteries produce high enough momentum to affect the rotating angles of the servo, the impact got accumulated if multiple C batteries enter the area, resulting in significant inaccuracy of angle rotated after multiple few rotations.

- Solution: As mentioned above in Electro-Mechanical and Microcontroller Subsystem (8.5.2.3 and 11.4.3), a non-continuous servo motor was used instead of a continuous one for C with better accuracy while AA battery voltage measuring mechanism remained the same.

11. System Improvement

11.1 Measuring voltages before sorting based on types

Instead of sorting out the batteries based on their types first and then measuring their voltages separately, another approach would be to measure the voltages of all batteries in one place and sort them by types afterward. The major advantage of this approach is that this way only one voltage measuring device was required, instead of three, which reduces the number of parts of the machine significantly, thus reducing the time required to manufacture/integrate/debug the additional parts. The voltage measuring device would need to be more complicated to account for differences in the three types of batteries, but it is definitely achievable.

11.2 Utilizations of actuators

The design of "LAL sorter" uses five different motors, and it might be more efficient to use more actuators to replace some of the passive mechanisms. For example, when measuring voltages for C and AA, copper coils were positioned to measure voltages by passively come in contact of incoming batteries' terminals, but this is unreliable because: (1) copper coils' shape could change with time due to gravity, and more significantly change due to the batteries passing by and (2) motions of batteries could also be changed dramatically due to collision with other batteries, and many other factors. As a result, every voltage measuring was a game of probabilities, and countless hours were spent to position the copper coils into optimal

positions/shapes after a few trials of operations. Instead, it would be much reliable to use a actuator to manipulate the positions of batteries/copper coil to ensure a necessary contact for voltage measuring. In addition, instead of pressing down the mechanical arm periodically, it might

11.3 Utilizations of passive mechanicals

The two-stage Sorting mechanisms have to potential to be combined together as a purely passive sorting mechanism which can sort different batteries based on its geometries. The success rate is expected to be higher without the current design of conveyor belt for separation of C and 9V batteries. References design can be referred to and has already been mentioned in the Electro-mechanical idea survey section.

11.4 Better Utilization of space

It was realized by Team 10 that LAL sorter was huge in terms of dimensions and very inefficient in space usage. One major reason is the over-usage of linear transportation of batteries. Instead, rotational force could be a better substitution to linear motion. Reference designs of coin separation machine also provided practical possibility to this idea as well which in session 7.3.1.

Initial and Accomplished Schedule

The initial schedule made by Team 10 for this project in Proposal and the actual accomplished schedule for all three team members as Gantt Charts are included in Appendix 8 and compared to evaluation Team 10's progress. The first half of the schedule was followed pretty well by Team 10, as the initial and accomplished schedules were pretty similar. However, the team rushed through the system integration process in about a week, instead of one month as purposed by the initial schedule. This is mainly due to the team rushing for the "Team Evaluation - Integration" deadline. The result of rushing for the deadline was poor, as expected, since the Team only focused on integrating all the parts together, ignoring the fact that almost none of the parts were functional as intended, and some could never be functional without major changes. After that, Team 10 entered one month of painful redesigning and debugging. In most intensive weeks, each of the team member would work on the project for as much as 60 hours. In the end, our extra effort finally paid off with the machine being almost completely functional at the day of the public demonstration

12. Conclusion

In the public demonstration, the "LAL sorter" was able to score one "qualified" run and one "completed" run out of the two runs. The "completed" run was also very close to be qualified, if only Team 10 had one more day to replace the water wheel for AA. Considering the fact that only 5 out of 24 teams for battery sorting had at least one "completed", it was quite an accomplishment.

For future improvements of AER201, Two major aspects should be considered for changes for this particular project. The first one is to increase the amount of time giving to the design teams for brainstorming and finalizing designs. As mentioned, the deigning process is very crucial for a project, thus more time and resources should be given to teams for coming up with better, more feasible design concepts to carry on to the next stages. The second one was to reallocate the amount of work required from each of the three subsystem members more evenly. A lot of design concepts arose from the given RFP required significantly higher efforts from the electromechanical member comparing to the other two members. As a result, at some stages of the project, a lot of teams had all three members more or less working on the electrical mechanical subsystem. This was definitely an inefficient usage of human resources, since in general a subsystem member is more familiar with their own subsystem, and it would only be the most efficient for them to only work on their own subsystems prior to the integration stage.

The most limiting specification of the RFP was the fact that at most 15 batteries would be loaded onto the machine all at once. This specification not only required the loading platform to be very robust to withstand the high combined weight, but also immune from jamming, because a cluster of heavy objects with different geometries are very possible to block the path of one another in any containing space, if not carefully handled.

The journey of designing and building the "LAL sorter" was one of the most remarkable and memorable experience for Team 10. It was not only the first engineering design project for Team 10, but also a valuable experiments that inspired the team to their engineering philosophies and enabled the team to gain practical knowledge on engineering designs, professional communications and system integrations which are important assets in real-life engineering world.

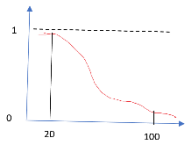
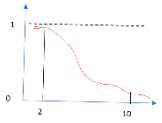
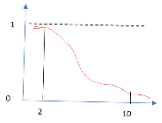
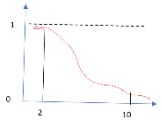
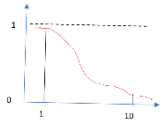
13. Budget

Budget			
Name	**Unit Price**	**Amount**	**Sub Total**
Micro-controller Subsystem			
PIC board	50	1	50
2.1mm DC Barrel (M) Wire Assembly	2.99	1	2.99
Circuit Subsystem			
Servo Motor SM-S4306B	9.9	3	29.7
Servo Motor SM-S4306R	9.05	1	9.05
DC motor	5	2	10
Emergency Stop Button	0,5	1	0.5
2.1mm DC Barrel Jack Breakout	3.5	1	3.5
silicon wires	5.4	1	5.4
resistors set	3	1	3
heat sink	2	2	2
L7805 voltage regulators	0.39	2	0.78
Solder Board	1.1	1	1.1
TIP142 transistors	2.19	2	4.38
12V 3A Power Supply	18.8	1	18.8
1N4007 Diode	0.13	14	1.82
Electro-Mechanical Subsystem			
1X2X8 WOOD	3.99	6	23.94
Plywood 1/4" 1'X2'	7.6	3	22.8
High impact acrylic sheet - .093Inch X 8 Inch X 10 Inch	7.6	1	7.6
Aluminum Sheet	5	1	5
2" x 1/2" Zinc Mending Plate*4	0.59	4	2.36
1-1/2" Zinc Flat Corner Plate * 2	0.96	2	1.92
3/4" Zinc Corner Brace*8	0.39	8	3.12
30cm x 150cm White Non Adhesive Grip Shelf/Drawer Liner	3.29	1	3.29
Screws WDZPFL SK 4X 1/2" 10PK	2.19	1	2.19
Insulation pipe wrap	0.79	1	0.79
Dowel, Hardwood, 5/8X4'	3.79	1	3.79
Total Price	**204.62**		

Receipts and evidences will be provided upon request.

Appendix 1. Frame material selection using utility-based method

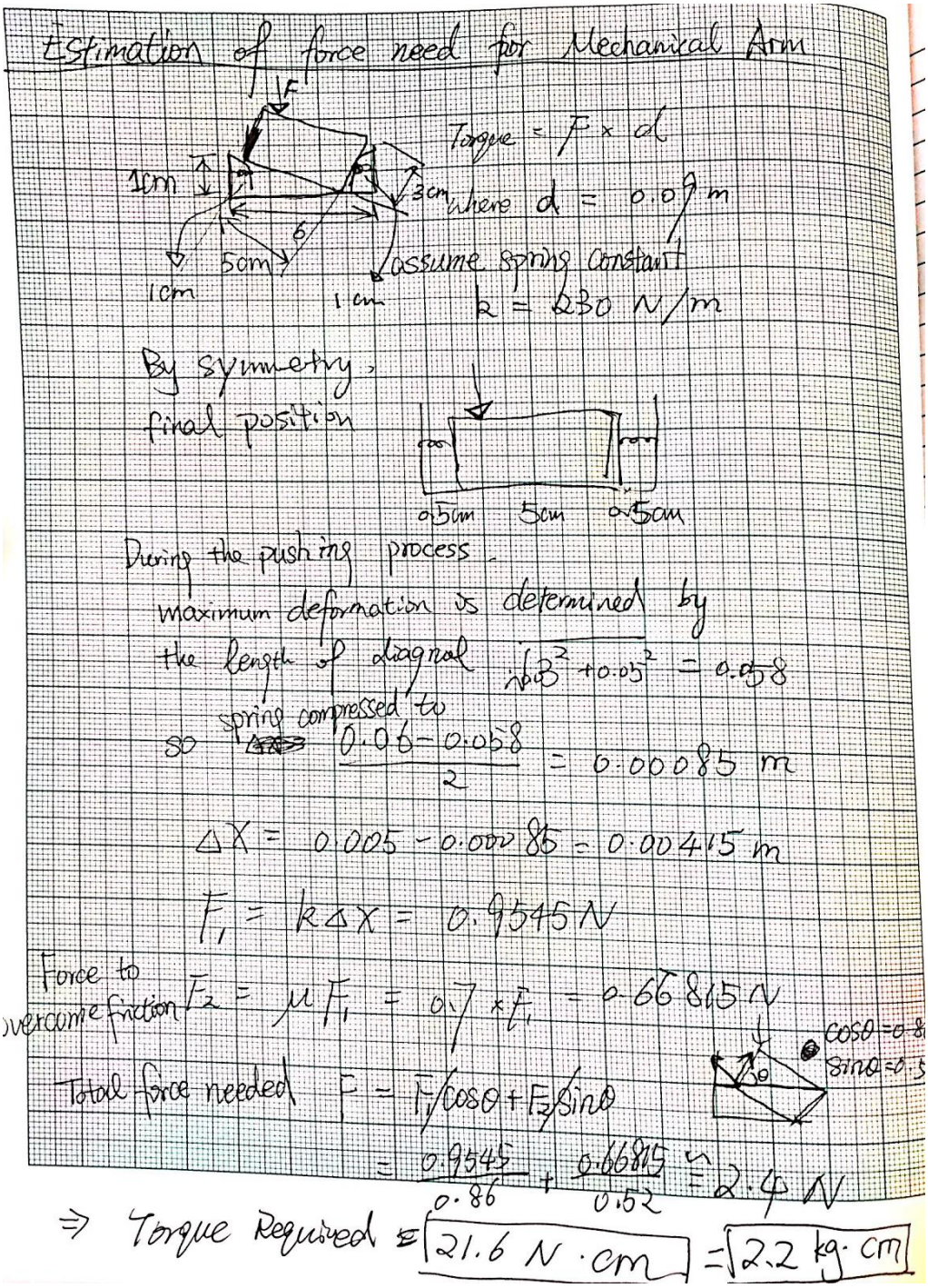
Table A1. Frame material selection

Objectives	Weight	Steel		Plywood		Aluminum		
		Estimated Val.	Utility Val.	Estimated Val.	Utility Val.	Estimated Val.	Utility Val.	Utility
Inexpensive	2	50	0.5	25	0.73	35	0.6	
Light	2	5	0.5	3	0.65	4	0.58	
High Manufacturability	3	8	0.25	4	0.8	5	0.7	
Elegance	2	6	0.7	5	0.6	6	0.7	
High robustness	4	8	0.8	7	0.7	5	0.5	
Utility Weighted Sum		7.35		9.16		7.86		
Normalized Total		0.30		0.38		0.32		

Appendix 2: Operation flow chart for machine operation

Appendix 3: Control flow chart for microcontroller program

Appendix 4: Torque Calculation for Mechanical Arm



Estimation of force need for Mechanical Arm

Torque = F × d

where d = 0.09 m

assume spring constant k = 230 N/m

By symmetry, final position

During the pushing process, maximum deformation is determined by the length of diagnal $\sqrt{0.03^2 + 0.05^2} = 0.058$

so spring compressed to $\frac{0.06 - 0.058}{2} = 0.00085$ m

$\Delta X = 0.005 - 0.00085 = 0.00415$ m

$F_1 = k\Delta X = 0.9545$ N

Force to overcome friction $F_2 = \mu F_1 = 0.7 \times F_1 = 0.66815$ N

Total force needed $F = F_1\cos\theta + F_2\sin\theta$

$= \frac{0.9545}{0.86} + \frac{0.66815}{0.52} \approx 2.4$ N

⇒ Torque Required = 21.6 N·cm = 2.2 kg·cm

Appendix 5: Raw data for determining charged or uncharged batteries

Table A5.1: C battery

Battery Type	Voltage	Number in one direction	Number in another direction	Average of two direction
C	1.58	125	132	128.5
C	1.31	79	85	82
C	1.6	129	134	131.5
C	1.27	70	74	72
C	0.82	9	11	10
C	0.5	4	3	3.5
C	1.61	131	136	133.5
C	0.82	4	4	4
C	1.58	131	120	125.5

Table A5.2: AA battery

Battery Type	voltage	number in one direction	number in another direction	average of two direction
AA	1.61	136	131	133.5
AA	1.37	94	92	93
AA	1.12	53	51	52
AA	1.6	133	135	134
AA	1.6	133	135	134
AA	1.37	92	96	94
AA	0.06	7	7	7

AA	1.61	135	137	136
AA	1.54	121	123	122
AA	1.45	105	108	106.5
AA	1.53	119	119	119
AA	1.61	134	137	135.5
AA	1.39	94	95	94.5

Table A5.3: 9V battery

Battery Type	voltage	number in one direction	number in another direction	Average of two direction
9V	3.18	55	57	56
9V	6.28	0	0	0
9V	9.38	179	173	176
9V	8.73	80	82	81
9V	4.4	14	17	15.5
9V	8.99	168	170	169
9V	7.5	60	62	61
9V	8.7	140	138	139
9V	6.5	93	95	94
9V	8.9	148	152	150

Appendix 6: Sorting Experiment

Table A6.1: Experiment Value for Primary Sorting Mechanisms

Trial number	Number of batteries to be sorted	Dropping property	Rough time (s)	Sorting result*
1	11	All at once	1	6/11
2	13	All at once	N/A	Jam
3	13	All at once	N/A	Jam
4	2	All at once	1	2/2
5	3	All at once	1	3/3
6	3	All at once	1	2/3
7	5	Slow	5	5/5
8	5	Slow	6	5/5
9	8	Slow	6	8/8
10	13	Slow	6	12/13
11	13	Slow	5	13/13
12	13	slow	7	13/13

The result shows that the accuracy of the Primary Sorting Mechanism is the higher with lower speed and smaller number of batteries coming together.

Table A6.2: Experiment Value for Secondary Sorting Mechanisms with 3 types of Batteries

Trial number	Number of batteries to be sorted	Rough time (s)	Sorting result*	Battery Type that causes mis-sorting
1	11	N/A	JAM	N/A
2	13	N/A	JAM	AA
3	13	N/A	JAM	AA
4	2	1	2/2	N/A
5	3	1	2/3	AA
6	3	1	1/3	AA

7	5	5	1/5	AA&C
8	5	6	2/5	AA&C
9	8	6	JAM	N/A
10	13	6	JAM	N/A
11	13	5	JAM	N/A
12	13	7	JAM	N/A

Table A6.3: Experiment Value for Secondary Sorting Mechanisms with C&9-V Batteries

Trial number	Number of batteries to be sorted	Rough time (s)	Sorting result*	Battery Type that causes mis-sorting
1	2	2	2/2	N/A
2	3	2	3/3	N/A
3	3	2	3/3	N/A
4	4	3	4/4	N/A
5	4	4	4/4	N/A
6	8	N/A	JAM	C
7	7	10	7/7	N/A
8	8	12	7/8	C
9	8	10	8/8	N/A
10	13	N/A	JAM	N/A
11	13	17	11/13	C
12	13	12	13/13	N/A

NOTE: All batteries were dropping from a track with 3 cm width such that at most 1 C battery or 1 9-V battery or 2 AA batteries can pass at a time.

The result shows the fact that the second mechanism (sorting AA batteries first) has a significantly higher success rate but C batteries are still the main concerns in separating all three types of batteries.

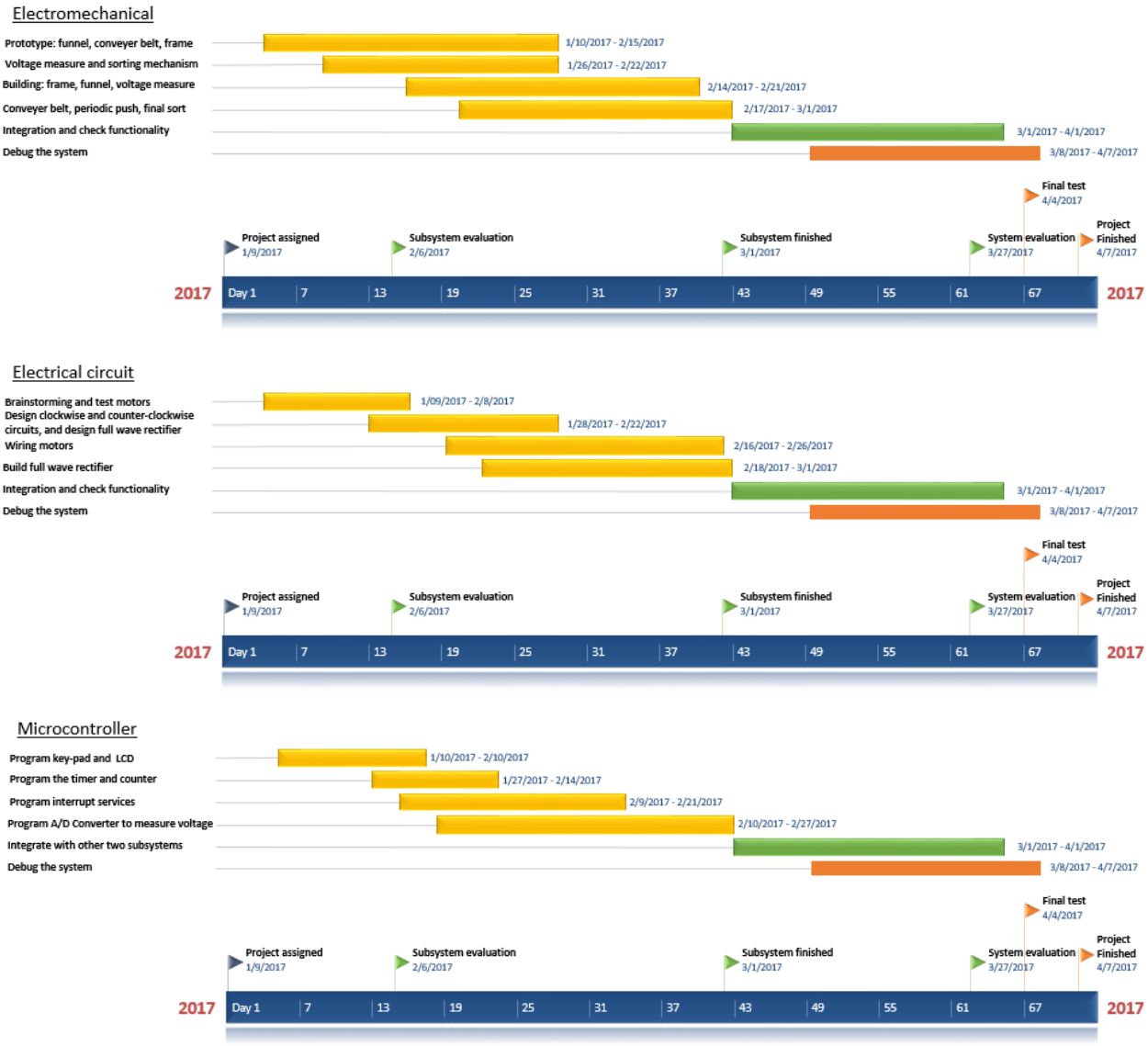
Appendix 7: Final Testing

Table A6: Final testing result

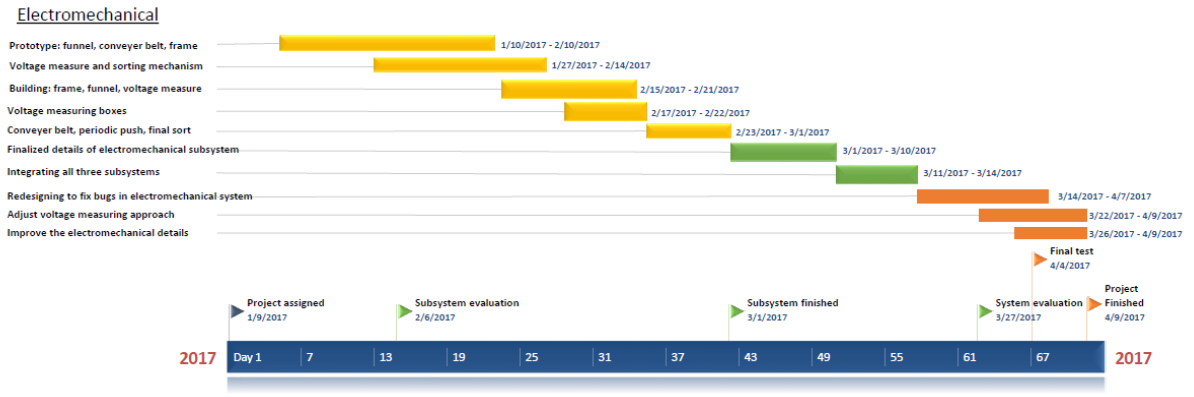
test number	AA correctly sorted	AA incorrectly sorted	C correctly sorted	C incorrectly sorted	9V correctly sorted	9V incorrectly sorted	Total number battery	Total Correct	Total Incorrect	Success rate
1	3	2	3	1	4	1	14	10	4	0.714
2	0	0	3	3	4	1	11	7	4	0.636
3	4	1	3	1	5	1	15	12	3	0.800
4	4	2	4	1	2	2	15	10	5	0.667
5	5	1	3	2	2	2	15	10	5	0.667
6	3	1	4	2	3	0	13	10	3	0.769
7	4	2	2	3	3	1	15	9	6	0.600
8	5	0	2	0	4	3	14	11	3	0.786
9	2	3	4	1	2	1	13	8	5	0.615
10	3	1	4	0	3	1	12	10	2	0.833
Total	33	13	32	14	32	13	137	97	40	0.708

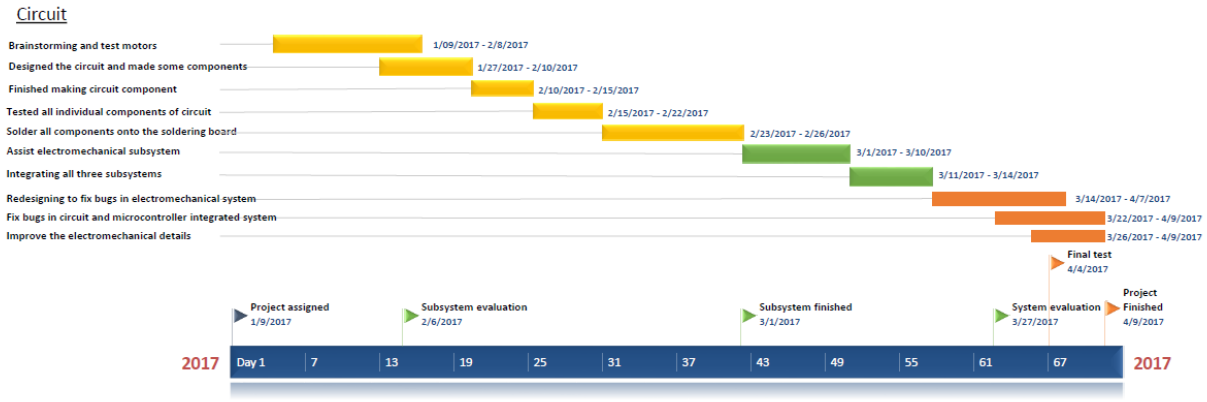
Appendix 8: Gantt chart

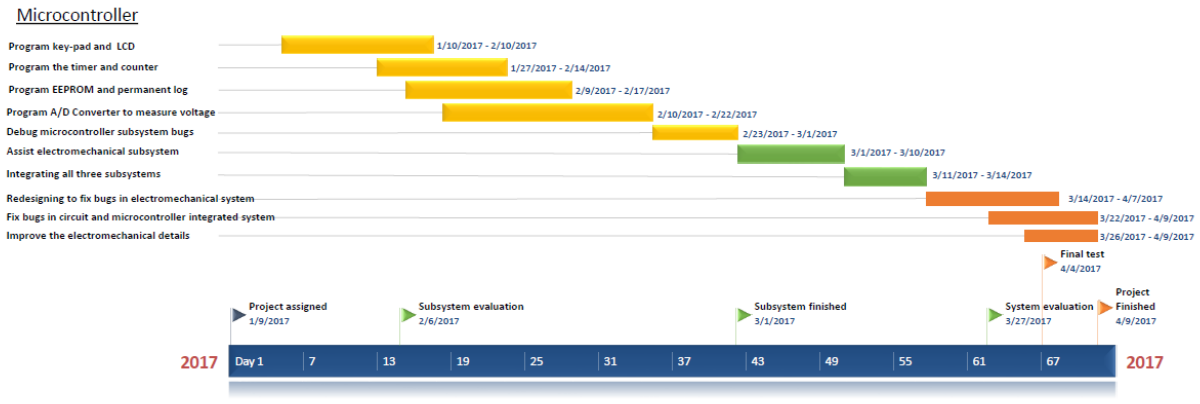
8.1 Initial Schedule



8.2 Accomplished Schedule







Appendix 9: Programming codes

9.1 Sample code studied in the Survey

9.1.1 Sample code used for port test

main.c

```
/*
 * File:   port_test.c
 * Author: Michael Ding
 *
 * Created on July 14, 2016, 10:25 AM
 */


#include "configBits.h"
//<editor-fold defaultstate="collapsed" desc="Variable Defs">
int i = 0;
unsigned char tmp = 0x00;
//</editor-fold>



void delay_10ms(unsigned char n) {
        while (n-- != 0) {
            __delay_ms(10);
        }
    }


void main(void) {


    // Disable Interrupt
    INTCON = 0x00;

    // SET ALL PORTS TO OUTPUT
    TRISA = 0x00; // PORT A OUTPUT
    TRISB = 0x00; // PORT B OUTPUT
    TRISC = 0x00; // PORT C OUTPUT
    TRISD = 0x00; // PORT D OUTPUT
    TRISE = 0x00; // PORT E OUTPUT

    // SET ALL PINS TO OUTPUT LOW
    LATA = 0x00;
    LATB = 0x00;
    LATC = 0x00;
    LATD = 0x00;
    LATE = 0x00;
```

main.c

```
45      // Infinite loop, cycle state on each pin
46      while(1){
47          // Same process for all other ports
48          // Repeated forever
49          tmp = 0x02;
50          for (i = 0; i<=1; i++){
51              LATE = tmp;
52              tmp >>= 1;
53              delay_10ms(10);
54          }
55          LATE = 0x00;
56
57          tmp = 0x80;
58          for (i = 0; i<=8; i++){
59              LATA = tmp;
60              tmp >>= 1;
61              delay_10ms(10);
62          }
63          LATA = 0x00;
64          tmp = 0x80;
65          for (i = 0; i<=8; i++){
66              LATB = tmp;
67              tmp >>= 1;
68              delay_10ms(10);
69          }
70          LATB = 0x00;
71
72          tmp = 0x80;
73          for (i = 0; i<=8; i++){
74              LATC = tmp;
75              tmp >>= 1;
76              delay_10ms(10);
77          }
78          LATC = 0x00;
79
80          tmp = 0x80;
81          for (i = 0; i<=8; i++){
82              LATD = tmp;
83              tmp >>= 1;
84              delay_10ms(10);
85          }
86          LATD = 0x00;
87      }
88  }
```

9.1.2 Sample code used for Keypad LCD interaction

main(1).c

```
/*
 * File:   main.c
 * Author: True Administrator
 *
 * Created on July 18, 2016, 12:11 PM
 */


#include <xc.h>
#include <stdio.h>
#include "configBits.h"
#include "constants.h"
#include "main.h"


void initLCD(void) {
    __delay_ms(15);
    // For reference, LCD instructions can be found in the controller datasheet
    // https://www.sparkfun.com/datasheets/LCD/HD44780.pdf
    lcdInst(0b00110011);
    lcdInst(0b00110010);
    lcdInst(0b00101000);
    lcdInst(0b00001111);
    lcdInst(0b00000110);
    lcdInst(0b00000001);
    __delay_ms(15);
}

void lcdInst(char data) {
    RS = 0;
    lcdNibble(data);
}

void putch(char data){
    RS = 1;
    lcdNibble(data);
}

void lcdNibble(char data){
    // Send of 4 most sig bits, then the 4 least sig bits (MSD,LSD)
    char temp = data & 0xF0;
    LATD = LATD & 0x0F;
    LATD = temp | LATD;
```

main(1).c

```
34  void putch(char data){
35      RS = 1;
36      lcdNibble(data);
37  }
38
39  void lcdNibble(char data){
40      // Send of 4 most sig bits, then the 4 least sig bits (MSD,LSD)
41      char temp = data & 0xF0;
42      LATD = LATD & 0x0F;
43      LATD = temp | LATD;
44
45      E = 0;
46      __delay_us(LCD_DELAY);
47      E = 1;
48      __delay_us(LCD_DELAY);
49
50      data = data << 4;
51
52      temp = data & 0xF0;
53      LATD = LATD & 0x0F;
54      LATD = temp | LATD;
55
56      E = 0;
57      __delay_us(LCD_DELAY);
58      E = 1;
59      __delay_us(LCD_DELAY);
60  }
61
62
63  void main(void) {
64      OSCCON = 0xF2; // Set internal oscillator to 8MHZ, and enforce internal oscillator operation
65      TRISD = 0x00;
66
67      initLCD();
68      printf("Hello World! :)");
69
70      while(1){
71          // Operation terminated, do noting
72      }
73
74      return;
75  }
76
```

9.1.3 Sample code used for A/D Converter

main(2).c

```
/*
 * File:   main.c
 * Author: True Administrator
 *
 * Created on July 18, 2016, 12:11 PM
 */


#include <xc.h>
#include <stdio.h>
#include "configBits.h"
#include "constants.h"
#include "lcd.h"
#include "macros.h"

void readADC(char channel);

void main(void) {

    // <editor-fold defaultstate="collapsed" desc=" STARTUP SEQUENCE ">
    OSCCON = 0xF0;  //8MHz

    TRISA = 0xFF; // Set Port A as all input
    TRISB = 0xFF;
    TRISC = 0x00;
    TRISD = 0x00; //All output mode for LCD
    TRISE = 0x00;

    LATA = 0x00;
    LATB = 0x00;
    LATC = 0x00;
    LATD = 0x00;
    LATE = 0x00;

    nRBPU = 0;

    initLCD();

    ADCON0 = 0x00;  //Disable ADC
    ADCON1 = 0x0B;  //AN0 to AN3 used as analog input
    CVRCON = 0x00; // Disable CCP reference voltage output
    CMCONbits.CIS = 0;
    ADFM = 1;
    //</editor-fold>
```

main(2).c

```
34
35      nRBPU = 0;
36
37      initLCD();
38
39      ADCON0 = 0x00;  //Disable ADC
40      ADCON1 = 0x0B;  //AN0 to AN3 used as analog input
41      CVRCON = 0x00; // Disable CCP reference voltage output
42      CMCONbits.CIS = 0;
43      ADFM = 1;
44      //</editor-fold>
45
46      while(1) {
47          readADC(1);
48          __lcd_home();
49          printf("%d %d", ADRESH,ADRESL);
50          printf("    ");
51          __lcd_newline();
52          readADC(1);
53          printf("%d %d", ADRESH,ADRESL);
54
55
56          printf("     ");
57          __delay_1s();
58      }
59
60  }
61
62  void readADC(char channel){
63      // Select A2D channel to read
64      ADCON0 = ((channel <<2));
65      ADON = 1;
66      ADCON0bits.GO = 1;
67     while(ADCON0bits.GO_NOT_DONE){__delay_ms(5);}
68
69
70  }
71
```

9.1.4 Sample code used for Real Time clock

main(3).c

```
/*
 * File:   main.c
 * Author: True Administrator
 *
 * Created on July 18, 2016, 12:11 PM
 */


#include <xc.h>
#include <stdio.h>
#include "configBits.h"
#include "constants.h"
#include "lcd.h"
#include "I2C.h"
#include "macros.h"

void set_time(void);

const char keys[] = "123A456B789C*0#D";
const char happynewyear[7] = {  0x20, //45 Seconds
                                0x39, //59 Minutes
                                0x11, //24 hour mode, set to 23:00
                                0x07, //Saturday
                                0x08, //31st
                                0x04, //December
                                0x17};//2016

void main(void) {

    // <editor-fold defaultstate="collapsed" desc=" STARTUP SEQUENCE ">

    TRISA = 0xFF; // Set Port A as all input
    TRISB = 0xFF;
    TRISC = 0x00;
    TRISD = 0x00; //All output mode for LCD
    TRISE = 0x00;

    LATA = 0x00;
    LATB = 0x00;
    LATC = 0x00;
    LATD = 0x00;
    LATE = 0x00;

```

main(3).c

```
43
44      ADCON0 = 0x00;  //Disable ADC
45      ADCON1 = 0xFF;  //Set PORTB to be digital instead of analog default
46
47      nRBPU = 0;
48
49      //</editor-fold>
50
51      initLCD();
52      unsigned char time[7];
53
54
55      I2C_Master_Init(10000); //Initialize I2C Master with 100KHz clock
56      di(); // Disable all interrupts
57
58      set_time();
59
60      while (1) {
61
62          //Reset RTC memory pointer
63          I2C_Master_Start(); //Start condition
64          I2C_Master_Write(0b11010000); //7 bit RTC address + Write
65          I2C_Master_Write(0x00); //Set memory pointer to seconds
66          I2C_Master_Stop(); //Stop condition
67
68          //Read Current Time
69          I2C_Master_Start();
70          I2C_Master_Write(0b11010001); //7 bit RTC address + Read
71          for(unsigned char i=0;i<0x06;i++){
72              //printf("%d ", i);
73              time[i] = I2C_Master_Read(1);
74          }
75
76          time[6] = I2C_Master_Read(0);       //Final Read without ack
77          I2C_Master_Stop();
78          __lcd_home();
79          printf("%02x/%02x/%02x%02x:%02x:%02x", time[6],time[5],time[4],time[2],time[1],time[0]);    //Print date in YY/MM/DD
80  //          __lcd_newline();
81  //          printf("%02x:%02x:%02x", time[2],time[1],time[0]);    //HH:MM:SS
82          __delay_1s();
83
84      }
85      return;
86  }
```

main(3).c

```
70          I2C_Master_Write(0b11010001); //7 bit RTC address + Read
71          for(unsigned char i=0;i<0x06;i++){
72              //printf("%d ", i);
73              time[i] = I2C_Master_Read(1);
74          }
75
76          time[6] = I2C_Master_Read(0);       //Final Read without ack
77          I2C_Master_Stop();
78          __lcd_home();
79          printf("%02x/%02x/%02x%02x:%02x:%02x", time[6],time[5],time[4],time[2],time
80  //          __lcd_newline();
81  //          printf("%02x:%02x:%02x", time[2],time[1],time[0]);    //HH:MM:SS
82          __delay_1s();
83
84      }
85      return;
86  }
87
88  void set_time(void){
89      I2C_Master_Start(); //Start condition
90      I2C_Master_Write(0b11010000); //7 bit RTC address + Write
91      I2C_Master_Write(0x00); //Set memory pointer to seconds
92      for(char i=0; i<7; i++){
93          I2C_Master_Write(happynewyear[i]);
94      }
95      I2C_Master_Stop(); //Stop condition
96
97
98  }
99
```

9.2 Final version operating code

main(4).c

```c
/*
 * File:   main.c
 * Author: Shoujun Feng
 *
 * Created on February 5, 2017, 10:30 AM
 */


#include <xc.h>
#include <stdio.h>
#include "configBits.h"
#include "constants.h"
#include "lcd.h"
#include "I2C.h"
#include "macros.h"
#include "stdint.h"
#include <string.h>
#include <stdlib.h>

#define __delay_1s() for(char i=0;i<100;i++){__delay_ms(10);}
#define __lcd_newline() lcdInst(0b11000000);
#define __lcd_clear() lcdInst(0x01);
#define __lcd_home() lcdInst(0b10000000);

const int STANDING = 0;
const int RUNNING = 1;
const int STARTED = 1;
const int FINISHED = 0;
const int DISPLAYED = 1;
const int NONDISPLAYED = 0;
char keys[] = "123A456B78tC*0#D";
char results[100];
char history[400];
int firstSecondsRun = 3;
int firstSecondsPaused = 0;

const char happynewyear[7] = {  0x10, //10 Seconds
                                0x20, //10 Minutes
                                0x15, //24 hour mode, set to 14:00
                                0x02, //Sunday
                                0x10, //27th
                                0x04, //February
                                0x17};//2017

```

main(4).c

```
46  int status = 0; // Initial status is standing.
47  int started = 0;
48  char AACount = 0;
49  char Ccount = 0;
50  char NineVoltCount = 0;
51  char UnchargedCount = 0;
52  char totalCount = 0;
53  int printLine;
54  int printLineHis;
55  int display;
56  int interruptDuringDisplay = 0;
57  unsigned char time[7];
58  int secondsConsumed = 0;
59
60  int secondsNotMeasured9V = 0;
61  int secondsNotMeasuredC = 0;
62  int secondsNotMeasuredAA = 0;
63
64  int secondsWithoutMeasure = 0;
65
66  int TOLERANCE = 2;
67  int NINE_CHARGED_CYCLE = 224;
68  int NINE_UNCHARGED_CYCLE = 390;
69  int C_CHARGED_CYCLE = 180;
70  int C_UNCHARGED_CYCLE = 412;
71  int AA_CHARGED_CYCLE = 140;
72  int AA_UNCHARGED_CYCLE = 380;
73  int pushServoPeriod = 2;
74  int nonContinuousRotFeq = 30;
75
76  void powerUp();
77  void powerOff();
78  void getResults(char* results);
79  void storeToEEPROM();
80  void set_time(void);
81  void measureVoltage(void);
82  void readADC(char channel);
83  void showManual();
84  void diaplayTime();
85  void clearLine();
86  void initialize();
87  void stopOperation();
88  uint8_t Eeprom_ReadByte(uint16_t address);
89  void Eeprom_WriteByte(uint16_t address, uint8_t data);
```

main(4).c

```
88  uint8_t Eeprom_ReadByte(uint16_t address);
89  void Eeprom_WriteByte(uint16_t address, uint8_t data);
90  void prepareHistory();
91  void showHistory();
92  void storeAccordingToNumberOfRun(uint8_t numberOfRuns);
93  void shift();
94  void toDigit(char* digits, char two_digit_number);
95
96
97  void initialize() {
98      results[0] = '*';
99      TRISA = 0xFF; // Set Port A as all input
100     TRISB = 0xFF;
101     TRISC = 0x00;
102     TRISD = 0x00;   //All output mode
103     TRISB = 0xFF;   //All input mode
104     TRISE = 0x00;
105
106     ADCON0 = 0x00;  //Disable ADC
107     ADCON1 = 0xFF;  //Set PORTB to be digital instead of analog default
108
109     nRBPU = 0;
110     INT1IE = 1;
111     ei();
112     I2C_Master_Init(10000); //Initialize I2C Master with 100KHz clock
113 }
114
115 void main(void) {
116
117     initialize();
118     initLCD();
119
120     LATA = 0x00;
121     LATB = 0x00;
122     LATC = 0x00;
123     LATD = 0x00;
124     LATE = 0x00;
125
126     while(1){
127         if(started == STARTED) {
128             measureVoltage();
129             if(secondsWithoutMeasure > 20) {
130                 stopOperation();
131             }
```

main(4).c

```
130                     stopOperation();
131                 }
132             }
133             if(!started && display == NONDISPLAYED){
134                 showManual();
135             }
136             if(secondsConsumed > 180) {
137                 stopOperation();
138                 secondsConsumed = 0;
139             }
140             diaplayTime();
141         }
142         return;
143     }
144
145     void turnNineVoltCharged(){
146         // Set to 0 degrees.
147         for(int f = 0; f < nonContinuousRotFeq; f++){
148             LATC1 = 1;
149             __delay_us(20);
150             LATC1 = 0;
151             __delay_ms(18);
152         }
153         __delay_ms(1000);
154         // Set to 90 degrees.
155         for(int f = 0; f < nonContinuousRotFeq; f++){
156             LATC1 = 1;
157             __delay_us(1350);
158             LATC1 = 0;
159             __delay_ms(17);
160         }
161     }
162
163     void turnNineVoltUncharged(){
164         // Set to 180 degrees.
165         for(int f = 0; f < nonContinuousRotFeq; f++){
166             LATC1 = 1;
167             __delay_us(3000);
168             LATC1 = 0;
169             __delay_ms(17);
170         }
171         __delay_ms(1000);
172         // Set back to 90 degrees.
173         for(int f = 0; f < nonContinuousRotFeq; f++){
```

main(4).c

```
172         // Set back to 90 degrees.
173         for(int f = 0; f < nonContinuousRotFeq; f++){
174             LATC1 = 1;
175             __delay_us(1350);
176             LATC1 = 0;
177             __delay_ms(17);
178         }
179     }
180
181     void turnCCharged(){
182         // Set to 0 degrees.
183         for(int f = 0; f < nonContinuousRotFeq; f++){
184             LATC2 = 1;
185             __delay_us(20);
186             LATC2 = 0;
187             __delay_ms(18);
188         }
189         __delay_ms(1000);
190         // Set to 90 degrees.
191         for(int f = 0; f < nonContinuousRotFeq; f++){
192             LATC2 = 1;
193             __delay_us(1500);
194             LATC2 = 0;
195             __delay_ms(18);
196         }
197     }
198
199     void turnCUncharged(){
200         // Set to 180 degrees.
201         for(int f = 0; f < nonContinuousRotFeq; f++){
202             LATC2 = 1;
203             __delay_us(3000);
204             LATC2 = 0;
205             __delay_ms(17);
206         }
207         __delay_ms(500);
208         // Set back to 90 degrees.
209         for(int f = 0; f < nonContinuousRotFeq; f++){
210             LATC2 = 1;
211             __delay_us(1500);
212             LATC2 = 0;
213             __delay_ms(18);
214         }
215     }
```

main(4).c

```
217  void turnAACharged(){
218      // turn clockwise
219      for(int f = 0; f < 196  ; f++) {
220          LATC5 ^= 1;
221          __delay_ms(1);
222      }
223  }
224
225  void turnAAUncharged(){
226      // turn counterclockwise
227      for(int f = 0; f < 98; f++){
228          LATC5 ^= 1;
229          __delay_ms(2);
230      }
231  }
232
233  void setInitialPosition(){
234      // Set all non-continuous servo initial conditions
235      for(int f = 0; f < nonContinuousRotFeq; f++){
236          LATC2 = 1;
237          __delay_us(1500);
238          LATC2 = 0;
239          __delay_ms(18);
240      }
241      // Set 9V to initial position
242      for(int f = 0; f < nonContinuousRotFeq; f++){
243          LATC1 = 1;
244          __delay_us(1350);
245          LATC1 = 0;
246          __delay_ms(17);
247      }
248  }
249
250  void pushForMeasure(){
251      // set to 180 degree
252      for (int f = 0; f < nonContinuousRotFeq; f++) {
253          LATC7 = 1;
254          __delay_us(3000);
255          LATC7 = 0;
256          __delay_ms(17);
257      }
258  }
259
```

main(4).c

```
259
260 void releasePushServo(){
261     // Set back to 90 degrees.
262     for(int f = 0; f < nonContinuousRotFeq; f++){
263         LATC7 = 1;
264         __delay_us(1100);
265         LATC7 = 0;
266         __delay_ms(17);
267     }
268 }
269
270 void measureVoltage() {
271     /*
272      * RB1, 4, 5, 6, 7 are reserved for keypad.
273      * RA0 for measuring 9V one direction
274      * RA1 for measuring 9V in another direction
275      * RA2 for measuring C
276      * RA3 for measuring AA
277      *
278      * This function reads four different pins to determine whether the process
279      * is still running and determine the battery is or not, increment the count
280      * of different batteries.
281      */
282     ADCON1 = 0b00001010;  //AN0 to AN3 used as analog input
283     CVRCON = 0x00; // Disable CCP reference voltage output
284     CMCONbits.CIS = 0;
285     ADFM = 1;
286     int thereIsBattery = 0;
287     int charged9V = 0; // 0 uncharged, 1 charged.
288
289     pushForMeasure();
290     // Read 9V battery in another direction.
291     readADC(1);
292     int nineVoltTest = ADRESL;
293     if (ADRESL > 0) {
294         thereIsBattery = 1;
295         totalCount++ ;
296         secondsNotMeasured9V = 0;
297         if(ADRESL > 140){
298             __lcd_home();
299             NineVoltCount++;
300             charged9V = 1;
301         }
302         else{
```

main(4).c

```
301         }
302         else{
303             UnchargedCount++ ;
304             charged9V = 0;
305         }
306 
307         if (charged9V) {
308             releasePushServo();
309             turnNineVoltCharged();
310         } else {
311             releasePushServo();
312             turnNineVoltUncharged();
313         }
314     }
315     releasePushServo();
316 
317     // read C battery
318     readADC(2);
319     int cBatteryTest = ADRESL;
320     int chargedC = 0; // 0 uncharged, 1 charged.
321     if (ADRESL > 8) {
322         thereIsBattery = 1;
323         totalCount++ ;
324         secondsNotMeasuredC = 0;
325         if(ADRESL > 89){
326             __lcd_home();
327             Ccount++ ;
328             chargedC = 1;
329         }
330         else{
331             UnchargedCount++ ;
332             chargedC = 0;
333         }
334         if (chargedC) {
335             releasePushServo();
336             turnCCharged();
337         } else {
338             releasePushServo();
339             turnCUncharged();
340         }
341     }
342 
343     // Read AA battery.
344     readADC(3);
```

main(4).c

```
343     // Read AA battery.
344     readADC(3);
345     int aaBatteryTest = ADRESL;
346     int chargedAA = 0; // 0 uncharged, 1 charged.
347     if (ADRESL > 15) {
348         thereIsBattery = 1;
349         totalCount++ ;
350         secondsNotMeasuredAA = 0;
351         if(ADRESL > 92){
352             __lcd_home();
353             AACount++ ;
354             chargedAA = 1;
355         }
356         else{
357             UnchargedCount++ ;
358             chargedAA = 0;
359         }
360         if (chargedAA) {
361             turnAACharged();
362         } else {
363             turnAAUncharged();
364         }
365     }
366     if(thereIsBattery) {
367         secondsWithoutMeasure = 0;
368     }
369     if(secondsNotMeasured9V > 4) {
370         releasePushServo();
371         turnNineVoltUncharged();
372         secondsNotMeasured9V = 0;
373         UnchargedCount++;
374         totalCount++;
375     }
376     if (secondsNotMeasuredC > 4) {
377         turnCUncharged();
378         UnchargedCount++;
379         secondsNotMeasuredC = 0;
380         totalCount++;
381     }
382
383     if (secondsNotMeasuredAA > 4) {
384         turnAAUncharged();
385         UnchargedCount++;
386         secondsNotMeasuredAA = 0;
```

main(4).c

```
385             UnchargedCount++;
386             secondsNotMeasuredAA = 0;
387             totalCount++;
388         }
389         initialize();
390     }
391
392     void interrupt keypressed(void) {
393         /*
394          * Key A represent start/restart.
395          * When A is pressed, status will be checked.
396          * If status is standing, clear the counts, clear the LCD screen
397          * and start the process by powering up all the elements. Display status on
398          * screen.
399          * If status is running, don't do anything.
400          *
401          * Key B represents pause/continue.
402          * When key B is pressed, status is checked.
403          * If status is running, stops the process and change the status into
404          * standing, display status on screen.
405          * If status is standing, continue the process and change the status into
406          * running.
407          *
408          * Key C represents end, change the status into standing and get the result
409          * stores the counts into the EEPROM.
410          *
411          * Key D represents display which displays the results on LCD and  Several
412          * lines will be displayed so that each time
413          * key D is pressed, a new line of the result will be displayed.
414          */
415         if(INT1IF){
416             interruptDuringDisplay = 1;
417             while (PORTBbits.RB1);
418             unsigned char keypress = (PORTB & 0xF0) >> 4;
419             if (keys[keypress] == 'A'){
420                 if (status == STANDING || started == FINISHED){
421                     // Clear all counts.
422                     started = STARTED;
423                     AACount = 0;
424                     Ccount = 0;
425                     NineVoltCount = 0;
426                     UnchargedCount = 0;
427                     totalCount = 0;
428                     display = NONDISPLAYED;
```

main(4).c

```
427                 totalCount = 0;
428                 display = NONDISPLAYED;
429                 secondsConsumed = 0;
430                 secondsWithoutMeasure = 0;
431                 // Start timing here.
432                 status = RUNNING;
433                 __lcd_home();
434                 printf("Status: Running.");
435                 powerUp();
436                 setInitialPosition();
437                 results[0] = '\0';
438             }
439         }
440         if (keys[keypress] == 'B'){
441             if(started == STARTED){
442                 display = NONDISPLAYED;
443                 if (status == STANDING){
444                     status = RUNNING;
445                     powerUp();
446                     __lcd_home();
447                     printf("Status: Continue.");
448                 }
449                 else{
450                     status = STANDING;
451                     powerOff();
452                     __lcd_home();
453                     printf("Status: Standing.");
454                 }
455             }
456         }
457 
458         if (keys[keypress] == 'C'){
459             stopOperation();
460         }
461         if (keys[keypress] == '1' && started == FINISHED){
462             display = DISPLAYED;
463             __lcd_home();
464             printf("                    ");
465             __lcd_home();
466             int i = 0;
467             int curPrint = 0;
468             if (results[0] == '*') {
469                 __lcd_home();
470                 printf(" Not available ");
```

main(4).c

```
466                 int i = 0;
467                 int curPrint = 0;
468                 if (results[0] == '*') {
469                     __lcd_home();
470                     printf(" Not available.");
471                     __lcd_home();
472                 } else {
473                     while(results[i] != '\0'){
474                     if(curPrint == printLine){
475                         if(results[i] != '\n'){
476                             putch(results[i]);
477                         }
478                     }
479                     if(results[i] == '\n'){
480                         curPrint ++;
481                     }
482                     if(curPrint > printLine)
483                     {
484                         printLine ++;
485                         break;
486                     }
487                     i++;
488                 }
489 
490                 if(printLine == 6)
491                 {
492                     printLine = 0;
493                 }
494                 }
495             }
496 
497             if (keys[keypress] == '2' && started == FINISHED) {
498                 // Display history here.
499                 prepareHistory();
500                 showHistory();
501             }
502 
503             INT1IF = 0;     //Clear flag bit
504         }
505     }
```

main(4).c

```
505  }
506
507  void powerUp(){
508      /*
509       * This function sends signals to power up elements in system so that the
510       * process is started.
511       */
512      LATC |= 0b00000001;
513      return ;
514  }
515
516
517  void powerOff(){
518      /*
519       * This function sends signals to power off elements in system so that the
520       * process is stopped.
521       */
522      LATC &= 0b11111110;
523      return ;
524  }
525
526  void toDigit(char* digits, char two_digit_number){
527      /*
528       * This function converts n to a string digits.
529       */
530      if (two_digit_number > 100) {
531          digits[0] = 9 + 48;
532          digits[1] = 9 + 48;
533          digits[2] = '\0';
534      }
535      if(two_digit_number >= 10){
536          digits[0] = two_digit_number / 10 + 48;
537          digits[1] = two_digit_number % 10 + 48;
538          digits[2] = '\0';
539      }
540      else{
541          digits[0] = two_digit_number + 48;
542          digits[1] = '\0';
543      }
544
545  }
546
547  void getResults(char* results){
548      /*
```

main(4).c

```
547  void getResults(char* results){
548      /*
549       * This function processes the counts and returns a string containing all
550       * results.
551       */
552
553      // Following is the fake results
554      strcat(results, "Uncharged: ");
555      char number[3];
556      toDigit(number, UnchargedCount);
557      strcat(results, number);
558
559      strcat(results, "\nNine Volt: ");
560      toDigit(number, NineVoltCount);
561      strcat(results, number);
562
563      strcat(results, "\nAA Batteries: ");
564      toDigit(number, AACount);
565      strcat(results, number);
566
567
568      strcat(results, "\nC Batteries: ");
569      toDigit(number, Ccount);
570      strcat(results, number);
571
572
573      strcat(results, "\nTotal count: ");
574      toDigit(number, totalCount);
575      strcat(results, number);
576
577      strcat(results, "\nOpTime: ");
578      char opTimeStr[6];
579      char minutes = secondsConsumed/60 + 48;
580      char secondsd, secondst;
581      if ((secondsConsumed%60) < 9) {
582          secondsd = secondsConsumed%60 + 48;
583          secondst = 48;
584      } else {
585          secondst = (secondsConsumed%60)/10 + 48;
586          secondsd = (secondsConsumed%60)%10 + 48;
587      }
588      opTimeStr[0] = minutes;
589      opTimeStr[1] = 'm';
590      opTimeStr[2] = secondst;
```

main(4).c

```
589         opTimeStr[1] = 'm';
590         opTimeStr[2] = secondst;
591         opTimeStr[3] = secondsd;
592         opTimeStr[4] = 's';
593         opTimeStr[5] = '\0';
594         strcat(results, opTimeStr);
595         strcat(results, "\n\0");
596     }
597
598
599 ▼   void shift() {
600         for (int i = 0; i < 3 * 48; i += 8) {
601             Eeprom_WriteByte(i, Eeprom_ReadByte(48 + i));
602         }
603     }
604
605
606
607 ▼   void storeAccordingToNumberOfRun(uint8_t numberOfRuns) {
608         Eeprom_WriteByte(numberOfRuns*48, UnchargedCount);
609         Eeprom_WriteByte(numberOfRuns*48 + 8, NineVoltCount);
610         Eeprom_WriteByte(numberOfRuns*48 + 16, AACount);
611         Eeprom_WriteByte(numberOfRuns*48 + 24, Ccount);
612         Eeprom_WriteByte(numberOfRuns*48 + 32, totalCount);
613         Eeprom_WriteByte(numberOfRuns*48 + 40, secondsConsumed);
614
615     }
616
617
618
619 ▼   void storeToEEPROM(){
620         //This function stores the current counts to EEPROM
621
622         uint8_t numberOfRuns = Eeprom_ReadByte(4 * 48);
623 ▼       if (numberOfRuns != 0 && numberOfRuns != 1 && numberOfRuns != 2
624             && numberOfRuns != 3 && numberOfRuns != 4) {
625             numberOfRuns = 0;
626         }
627 ▼       if (numberOfRuns == 4) {
628             shift();
629             storeAccordingToNumberOfRun(3);
630         } else {
631             storeAccordingToNumberOfRun(numberOfRuns);
632         }
```

main(4).c

```
631         storeAccordingToNumberOfRun(numberOfRuns);
632     }
633     if (numberOfRuns != 4) {
634         numberOfRuns++ ;
635     }
636     Eeprom_WriteByte(4 * 48, numberOfRuns);
637     return ;
638 }
639 
640 
641 void set_time(void){
642     I2C_Master_Start(); //Start condition
643     I2C_Master_Write(0b11010000); //7 bit RTC address + Write
644     I2C_Master_Write(0x00); //Set memory pointer to seconds
645     for(char i=0; i<7; i++){
646         I2C_Master_Write(happynewyear[i]);
647     }
648     I2C_Master_Stop(); //Stop condition
649 }
650 
651 
652 void readADC(char channel){
653     // Select A2D channel to read
654     ADCON0 = ((channel <<2));
655     ADON = 1;
656     ADCON0bits.GO = 1;
657    while(ADCON0bits.GO_NOT_DONE){__delay_ms(5);}
658 }
659 
660 void firstMotorRun() {
661     LATC6 = 1;
662     __delay_ms(300);
663     LATC6 = 0;
664 }
665 void diaplayTime(){
666     I2C_Master_Start(); //Start condition
667     I2C_Master_Write(0b11010000); //7 bit RTC address + Write
668     I2C_Master_Write(0x00); //Set memory pointer to seconds
669     I2C_Master_Stop(); //Stop condition
670 
671     //Read Current Time
672     I2C_Master_Start();
673     I2C_Master_Write(0b11010001); //7 bit RTC address + Read
674     for(unsigned char i=0;i<0x06;i++){
```

main(4).c

```
673     I2C_Master_Write(0b11010001); //7 bit RTC address + Read
674     for(unsigned char i=0;i<0x06;i++){
675         time[i] = I2C_Master_Read(1);
676     }
677     time[6] = I2C_Master_Read(0);       //Final Read without ack
678     I2C_Master_Stop();
679     __lcd_home();
680     __lcd_newline();
681     printf("%02x/%02x %02x:%02x:%02x", time[5],time[4], time[2],time[1],time[0]);
682     __lcd_home();
683 
684     __delay_ms(1000);
685 
686     if(status == STARTED) {
687         if(secondsConsumed <= 75) {
688             firstMotorRun();
689         }
690 
691         if (firstSecondsPaused > 0) {
692             LATC6 = 0;
693             firstSecondsPaused--;
694         } else if (firstSecondsRun == -1) {
695             firstSecondsRun = 1;
696         }
697     }
698 
699     secondsConsumed++ ;
700     secondsWithoutMeasure++ ;
701     secondsNotMeasured9V++;
702     secondsNotMeasuredC++;
703     secondsNotMeasuredAA++;
704 }
705 
706 void showManual(){
707     __lcd_home();
708     printf("                    ");
709     __lcd_home();
710     printf("A to Start.");
711     if(!started)
712     {clearLine();
713     if(interruptDuringDisplay){
714         interruptDuringDisplay = 0;
715         return;
716     }
```

main(4).c

```
715         return;
716     }
717     printf("B to Pause.");
718     }
719     if(!started)
720     {   clearLine();
721         if(interruptDuringDisplay){
722             interruptDuringDisplay = 0;
723             return;
724         }
725         printf("B to Continue.");
726     }
727     if(!started){
728     clearLine();
729     if(interruptDuringDisplay){
730         interruptDuringDisplay = 0;
731         return;
732     }
733     printf("C to Finish.");
734     }
735     if(!started)
736     {clearLine();
737     if(interruptDuringDisplay){
738         interruptDuringDisplay = 0;
739         return;
740     }
741     printf("1 to Display");
742     }
743     if(!started){
744     clearLine();
745     if(interruptDuringDisplay){
746         interruptDuringDisplay = 0;
747         return;
748     }
749     printf("the results.");
750     }
751     if(!started){
752     clearLine();
753     if(interruptDuringDisplay){
754         interruptDuringDisplay = 0;
755         return;
756     }
757     printf("2 for History.");
758     }
```

main(4).c

```
760  void clearLine(){
761      diaplayTime();
762      if(interruptDuringDisplay){
763          return;
764      }
765      __lcd_home();
766      printf("                    ");
767      __lcd_home();
768  }
769
770  void stopOperation() {
771      display = DISPLAYED;
772      status = STANDING;
773      started = FINISHED;
774      powerOff();
775      __lcd_home();
776      printf("Status: Finished.");
777      results[0] = '\0';
778      getResults(results);
779      printLine = 0;
780      printLineHis = 0;
781      storeToEEPROM();
782      LATC6 = 0;
783  }
784
785
786  //! @brief      Reads a single byte of data from the EEPROM.
787  //! @param      address     The EEPROM address to write the data to (note that not all
788  //!                         16-bits of this variable may be supported).
789  //! @returns    The byte of data read from EEPROM.
790  //! @warning    This function does not return until read operation is complete.
791  uint8_t Eeprom_ReadByte(uint16_t address)
792  {
793
794      // Set address registers
795      EEADRH = (uint8_t)(address >> 8);
796      EEADR = (uint8_t)address;
797
798      EECON1bits.EEPGD = 0;       // Select EEPROM Data Memory
799      EECON1bits.CFGS = 0;        // Access flash/EEPROM NOT config. registers
800      EECON1bits.RD = 1;          // Start a read cycle
801
802      // A read should only take one cycle, and then the hardware will clear
803      // the RD bit
```

main(4).c

```
802     // A read should only take one cycle, and then the hardware will clear
803     // the RD bit
804     while(EECON1bits.RD == 1);
805 
806     return EEDATA;              // Return data
807 
808 }
809 
810 //! @brief      Writes a single byte of data to the EEPROM.
811 //! @param      address     The EEPROM address to write the data to (note that not all
812 //!                         16-bits of this variable may be supported).
813 //! @param      data        The data to write to EEPROM.
814 //! @warning    This function does not return until write operation is complete.
815 void Eeprom_WriteByte(uint16_t address, uint8_t data)
816 {
817     // Set address registers
818     EEADRH = (uint8_t)(address >> 8);
819     EEADR = (uint8_t)address;
820 
821     EEDATA = data;          // Write data we want to write to SFR
822     EECON1bits.EEPGD = 0;   // Select EEPROM data memory
823     EECON1bits.CFGS = 0;    // Access flash/EEPROM NOT config. registers
824     EECON1bits.WREN = 1;    // Enable writing of EEPROM (this is disabled again after the write completes)
825 
826     // The next three lines of code perform the required operations to
827     // initiate a EEPROM write
828     EECON2 = 0x55;          // Part of required sequence for write to internal EEPROM
829     EECON2 = 0xAA;          // Part of required sequence for write to internal EEPROM
830     EECON1bits.WR = 1;      // Part of required sequence for write to internal EEPROM
831 
832     // Loop until write operation is complete
833     while(PIR2bits.EEIF == 0)
834     {
835         continue;   // Do nothing, are just waiting
836     }
837 
838     PIR2bits.EEIF = 0;      //Clearing EEIF bit (this MUST be cleared in software after each write)
839     EECON1bits.WREN = 0;    // Disable write (for safety, it is re-enabled next time a EEPROM write is performed)
840 }
841 
842 
843 void prepareHistory() {
844     uint8_t numberOfRuns = Eeprom_ReadByte(4*48);
845     if (numberOfRuns == 255) {
```

main(4).c

```
844     uint8_t numberOfRuns = Eeprom_ReadByte(4*48);
845     if (numberOfRuns == 255) {
846         return;
847     }
848     history[0] = '\0';
849     strcat(history, "History: ");
850     for (int i = 0; i < numberOfRuns; i++) {
851 
852         uint8_t data;
853         char number[3];
854         strcat(history, "\nRun# :");
855         toDigit(number, i + 1);
856         strcat(history, number);
857 
858         strcat(history, "\nUncharged:");
859         data = Eeprom_ReadByte(i*48);
860         toDigit(number, data);
861         strcat(history, number);
862 
863         strcat(history, "\nNine Volt:");
864         data = Eeprom_ReadByte(i*48 + 8);
865         toDigit(number, data);
866         strcat(history, number);
867 
868         strcat(history, "\nAA Battery:");
869         data = Eeprom_ReadByte(i*48 + 16);
870         toDigit(number, data);
871         strcat(history, number);
872 
873         strcat(history, "\nC Battery:");
874         data = Eeprom_ReadByte(i*48 + 24);
875         toDigit(number, data);
876         strcat(history, number);
877 
878         strcat(history, "\nTotal Count:");
879         data = Eeprom_ReadByte(i*48 + 32);
880         toDigit(number, data);
881         strcat(history, number);
882         strcat(history, "\nOpTime :");
883         uint8_t secondsConsumedHis = Eeprom_ReadByte(i*48 + 40);
884         char opTimeStr[6];
885         char minutes = secondsConsumedHis/60 + 48;
886         char secondsd, secondst;
887         if ((secondsConsumedHis%60) < 9) {
```

main(4).c

```
886             char secondsd, secondst;
887             if ((secondsConsumedHis%60) < 9) {
888                 secondsd = secondsConsumedHis%60 + 48;
889                 secondst = 48;
890             } else {
891                 secondst = (secondsConsumedHis%60)/10 + 48;
892                 secondsd = (secondsConsumedHis%60)%10 + 48;
893             }
894             opTimeStr[0] = minutes;
895             opTimeStr[1] = 'm';
896             opTimeStr[2] = secondst;
897             opTimeStr[3] = secondsd;
898             opTimeStr[4] = 's';
899             opTimeStr[5] = '\0';
900             strcat(history, opTimeStr);
901
902
903         }
904         strcat(history, "q");
905         strcat(history, "\0");
906     }
907
908
909
910     void showHistory() {
911         uint8_t numberOfRuns = Eeprom_ReadByte(4*48);
912         display = DISPLAYED;
913         __lcd_home();
914         printf("                    ");
915         __lcd_home();
916         if (numberOfRuns == 255) {
917             printf("No history!");
918         }
919         int i = 0;
920         int curPrint = 0;
921         while(history[i] != '\0'){
922             if(curPrint == printLineHis){
923                 if(history[i] != '\n' && history[i] != 'q'){
924                     putch(history[i]);
925                 }
926             }
927             if(history[i] == '\n'){
928                 curPrint ++;
929             }
```

main(4).c

```
913         __lcd_home();
914         printf("                    ");
915         __lcd_home();
916         if (numberOfRuns == 255) {
917             printf("No history!");
918         }
919         int i = 0;
920         int curPrint = 0;
921         while(history[i] != '\0'){
922             if(curPrint == printLineHis){
923                 if(history[i] != '\n' && history[i] != 'q')
924                     putch(history[i]);
925                 }
926             }
927             if(history[i] == '\n'){
928                 curPrint ++;
929             }
930             if(history[i] == 'q') {
931                 printLineHis ++;
932                 break;
933             }
934             if(curPrint > printLineHis)
935             {
936                 printLineHis ++;
937                 break;
938             }
939             i++;
940         }
941
942         if(printLineHis == 7 * numberOfRuns + 1)
943         {
944             printLineHis = 0;
945         }
946     }
947
```

Appendix 10: Bibliography

[1] C. R. Malavika, "Environmental Effects Associated with Battery Disposal," Frost Sullivan, [Online]. Available: https://www.frost.com/sublib/display-market-insight-top.do?id=20759887.

[2] G. D. M. Controls. [Online]. Available: http://www.geckodrive.com/motor-pros-cons.

[3] F. Reed, "How do Servo Motors Work," [Online]. Available: http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html.

[4] "废旧电池回收机," 2012. [Online]. Available: http://v.youku.com/v_show/id_XMTU4NjE5NTU0NA==.html?from=s1.8-1-1.2. [Accessed 13 4 2017].

[5] Youtube, "Coin Sorter Machine," Youtube, [Online]. Available: https://www.youtube.com/watch?v=ZfdSQdXPd30. [Accessed 13 4 2017].

[6] Youtube, "DIY money separator," Youtube, [Online]. Available: https://www.youtube.com/watch?v=jFKRCCk6JgA. [Accessed 13 4 2017].

[7] HyperPhysics, "Torque Calculation," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/torq2.html.

[8] Anon, "Torque Calculaion on Conveyor Belt," [Online]. Available: http://www.nidec.com/en-NA/technology/calc/torque/conveyor_belt/. [Accessed 13 4 2017].

[9] Study.com., "Hooke's Law & the Spring Constant: Definition & Equation - Video & Lesson Transcript | Study.com.," study.com, 2017. [Online]. Available: http://study.com/academy/lesson/hookes-law-the-spring-constant-definition-equation.html.

[10] Hyperphysics.phy-astr.gsu.edu., "Moment of Inertia," Hyperphysics.phy-astr.gsu.edu., [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html.

[11] Ecnerwal, "When to use a heatsink for a voltage regulator," 16 Jan 2017. [Online]. Available: http://electronics.stackexchange.com/questions/210198/when-to-use-a-heatsink-for-a-voltage-regulator.

[12] Engineersedge.com, " Coefficient of Friction Equation and Table Chart - Engineers Edge.," 2017. [Online]. Available: http://www.engineersedge.com/coeffients_of_friction.htm.

[13] springrc, "43R Servo Specification," [Online]. Available: https://www.sparkfun.com/datasheets/Robotics/servo-360_e.pdf.

[14] Pololu, "HD-3001HB.PDF," [Online]. Available: https://www.pololu.com/file/download/HD-3001HB.pdf?file_id=0J728.

[15] O. Semiconductors, "TIP142 Silicon Power Transistors," [Online]. Available: https://www.onsemi.com/pub/Collateral/TIP140-D.PDF.

[16] J. Electronics, "L7805 5V 2% Regulator," [Online]. Available: http://datasheet.octopart.com/L7805CV-STMicroelectronics-datasheet-7264666.pdf.

[17] VISHAY, "1n4001.pdf," [Online]. Available: http://www.vishay.com/docs/88503/1n4001.pdf.