Novel Autonomous Subsequent-Hopper Pill-Sorting Solution with Lego® Mindstorms® EV3, Python 3 Implementation

-Design Proposal-

Nicholas Olson, Ty Brewen

Section 23, Team 05

Proposal Due November 8, 2018

Oregon State University

CS 160 / MIME 101

Fall Term, 2018-19

Table of Contents

Introduction **3**

Background **4**

Social Design Criteria **5-6**

- Listening 5
- Structural Conditions Causing Needs 6

Technical Descriptions **7-13**

- Required Hardware 7
  - Physical Construction and Structure 7
  - Driving and Interacting with Objects 7
  - Sensing and Decision-Making 7
- Hardware Overview 7
- Sorting System 8
- Dispensing System 8
- Driving System 9
- Software Overview 9
- Software Tasks (Functions) 10-13
  - Sorting Pills 10
  - Searching for Containers 10
  - Turning Toward Container 11
  - Driving to Container 11
  - Dumping Pills into Container 12
  - Reversing to Starting Position 12

Gantt Chart **13-16**

- Timeline and List Representation 14
- Spreadsheet List Representation 15
- Interactive List and Timeline 16

Design Economics **17-20**

- List of Materials and Costs 17-20
- Analysis of Costs 20

References **21**

Introduction

The ability for impaired persons, especially those persons with impaired vision, to safely differentiate between different pills of similar or identical shape and size is of great concern to the engineering and pharmaceutical community. In addition, the community also has concern for aging persons or those with generally limited cognitive processing power when it comes to differentiation of similar objects. Because of these concerns, a local non-profit company, AID Design, has tasked OSU students, to design a robot capable of accurately and efficiently sorting pills of different colors and depositing them into their respective containers. Thus requiring only that these visually impaired persons, or any other persons who might require such a solution, identify the proper container of sorted pills, without worry that they might ingest the wrong medication.

The team has set out to create a solution that is accessible to as many persons as possible. The team's solution utilizes the Lego® Mindstorms® EV3 education kit for physical construction to minimize cost and maximize accessibility. The robot will completely autonomously sort all pills that are deposited into its input hopper into two smaller hoppers, locate the target pill containers in the space around it, and deposit the correct pills in the correct containers, returning to its initial location, ready for another iteration of its task. This requires minimal input from the user, making it as accessible as possible.

Table 1. Pill Sorting Design Requirements

Design Criterion	**Criterion Description**	**Importance**
Size	The pill sorting robot must fit and be capable of navigating in a circular arena with a 5-foot diameter that contains two cylindrical containers with diameters of 6 inches.	Essential
Durability	The robot must sturdy enough to perform this task multiple times without risking breakage.	Great Importance
Materials	All materials used must be Lego Mindstorm, Cozmo, or 3D printed parts.	Essential

Background

As technology progresses, a major endeavor of the engineering community has become enabling impaired persons to safely cross streets, move between stories of a building, walk upright, read, speak, and so on. For centuries disabled persons and those who deal with some impairment (whether it be visual or related to motor-skills) have been underserved by society. Political institutions, engineers, and litigators all seek to increase the accessibility of everyday tasks and locales, and to generally increase the quality of life of these underserved peoples. Recently, a light has been shone on the trials that the visually impaired face. For the visually impaired, navigating a very visual world can be challenging. Although there are many ways companies have innovated their products to allow for better accessibility for visually impaired persons, the work is far from done. Especially in the pharmaceutical industry, there are strides still to be taken. Usually, unless different pills have radically different shapes or textures, which is never a certainty, pills must be distinguished visually. For pills that are already sorted into containers, this is not too great of an issue, but if they are mixed, this can pose a problem for visually impaired persons.

When searching for different solutions to this issue, an impaired person may have difficulty finding a product that can assist them effectively and reliably, while also being affordable or adaptable to any space. For a person who is not familiar with the sorting industry, they are likely to find only products, like the VMek Analytic Color Sorter, that are large, cumbersome, extremely expensive, or some combination of the three. For someone with limited space to set up such a device, limited funds to allot, or someone who doesn't need all the bells and whistles that drive up such a price, there aren't many products on the market. The team's solution seeks to address all these concerns. The team will minimize cost by using as few moving parts as possible and sticking to the basics of the problem: how to sort up to twenty pills efficiently and accurately. The team's solution also only requires that you have some open space for the robot to maneuver within, but as long as the pill containers are the two closest objects to the robot, it will still succeed, even if objects encroach on the space. The team's solution makes improvements to the industry by focusing essentially on accessibility, not just to those with impairment, but also to everyone in almost every socio-economic class.

For those persons who work with visual impairment or disability, the world is becoming more accessible, but is far from perfect. The team's solution for color-sorting of medication seeks to increase that level of accessibility on as many fronts as possible.

Social Design Criteria

Listening

Stakeholder	Desires / Needs
Visually Impaired Patient	**Sorting:** The impaired patient will need the robot to sort with complete autonomy. It would be better for the robot to sort fewer pills than to be inaccurate, lest the patient consume the wrong pill because of a fault of the solution. **Locating:** The robot needs to be able to locate the correct pill containers on its own, as to not rely on a patient who may not be able to identify the containers accurately in the moment. **Reusability:** The robot must be reusable a number of times without excess input by the patient, such that the only input the solution requires is a first-time setup of colors to differentiate between and then for the pills to be deposited into the robot for each iteration. The robot should return to a defined starting point to simplify the patient experience.
AID Design	**Design:** AID requires that the robot system can efficiently and accurately sort two colors of pills into their respective containers.
Health Professionals	**Consistency:** The doctors of visually impaired patients must be able to trust that the system will operate consistently each time so their patient is safe from technical error. **Accuracy:** The doctors need the pills to be sorted accurately for their patient, so the patient can use the solution safely, without risk of ingestion of the wrong medication.

Structural Conditions Causing Needs

Structural Condition	**Description of Barrier**
Financial	Cost of Lego parts and plastic for the 3D printer could be costly, so the team must limit use of more costly pieces and streamline design. Fewer moving parts nets lower cost. No more than four motors.
Implicit difficulty of living as a visually impaired person	The visually impaired patient must not need to rely on their vision whatsoever to allow for any patient with any degree of impairment to be able to use the solution.

Technical Descriptions

Required Hardware

Physical construction and structure:

- 1 large hopper for unsorted pills
- 2 small hoppers for sorted pills
- 2 color sensors

Driving, interacting with objects:

- 2 large motors for operating wheels to drive and steer
- 1 large motor for operating dispensing mechanism
- 1 medium motor for operating sorting mechanism

Sensing, decision-making:

- 1 ultrasonic Sensor for distance computation
- 1 gyroscopic Sensor for steering computation
- 1 color sensor for operation of hopper sorting mechanism
- 1 color sensor for determining color of target pill containers
- 1 EV3 Mindstorms console for operation of pill-sorter

Hardware Overview

The technical solution is a three-hopper sorting system mounted on a mobile robot. Each pill will start in the main, largest hopper, then will be dispensed onto a ramp. The pill will slide to a stop over a color sensor, which will determine one of two directions to send the pill. A gear system will deposit the pill onto one of two ramps while simultaneously releasing the next pill from the hopper. Once on the second ramp, the pill will slide into its respective small hopper (one of two). When all of the pills have been sorted, the robot will determine the locations of the two pill containers and set a course towards one using its ultrasonic and gyroscopic sensors. A color sensor mounted at the front of the system will determine the color of the container, and the robot will rotate to position the respective small hopper over the container and then empty the hopper. Finally, the robot will back into the center of the field and repeat the search and emptying process.

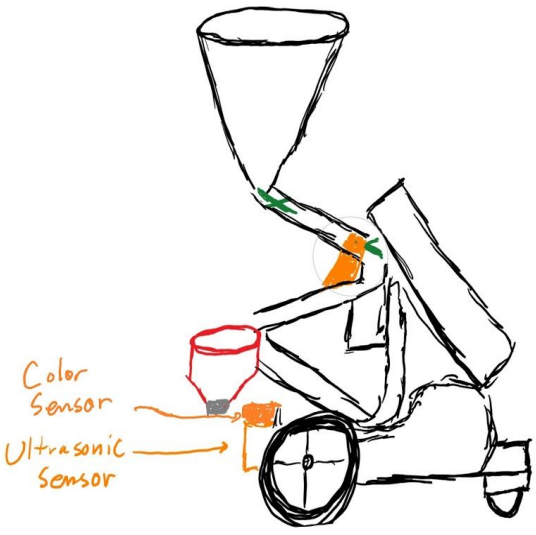


Figure 1

Sorting System

One of the key systems is initial pill dispensing and sorting, which can only be described as a single system. Primarily, the hopper will be a perfect conical design with a hole at the point allowing only one pill through at a time. A 4-tooth gear will be beneath the hopper at an angle aligned with the ramp.

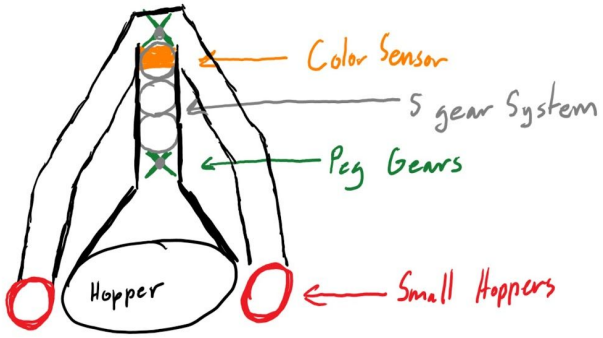


Figure 2

A 45 degree rotation will allow the pill to drop between the gear teeth, and another 180 degree rotation will slid the pill down into the second 4-tooth gear. The first of these gears will run directly to a medium motor, and between the gears is a system of five other gears (two 8-tooth and three-24 tooth in between), so both 4-tooth gears will move simultaneously. This allows for a perfectly smooth transition between each pill.

The color sensor, located directly underneath the spot the pill will stop, will determine the color of the pill. A 90 degree rotation in the medium motor will drop the pill onto a ramp, depending on which way it rotates. Another 90 degrees and process will repeat.

Small Hopper Dispensing System

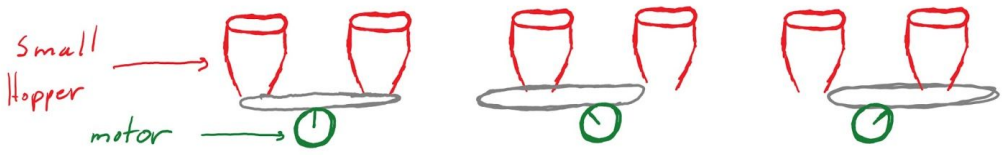


Figure 3

Once sorted, each pill will glide down its respective ramp into its corresponding hopper. These hoppers have a more liberal hole in the bottom that is stopped my a cross beam spanning both. A large motor attached loosely to the cross beam will rotate one way or the other, sliding the cross beam out from underneath one hopper while securely capping the other, as shown in *Figure 3*.

Drive System and Corresponding Sensors

The final component of the robot is its driving system. As shown in *Figure 1*, the robot will run one motor for each of the two wheels, and support its rear on the steel ball secured in the ball bearing. Two sensors will be at the front of the robot, the ultrasonic and second color sensors, and one, the gyro sensor, will be placed centrally. Ideally, the robot will spin 360 degrees, locating the two closest objects (the pill containers) with its ultrasonic sensor and noting what degree of its rotation they are located at with its gyro sensor. The robot will then turn towards the first container it located and approach it, using the gyro to keep it on corse, and then stop upon getting within 1 cm of the container. The color sensor will then detect the value of the container's color, rotate to position the correct small hopper directly above the container, and deposit the pills using the small hopper system. The robot will then reposition itself, back up to its original location, turn to the second degree coordinate, and repeat the process with the second container.

Overall Software Flow

1. Reset / calibrate all sensors, especially gyro and ultrasonic sensors.
2. Simultaneously if possible:
    - Search for pill containers in space around robot.
    - Sort all pills from large hopper into their respective hoppers.
3. Turn toward one of the located containers.
4. Drive up to this container (directly ahead of robot), return whether the color of the container matches the color value passed to the function.
5. Depending on which container it is, turn so that the corresponding hopper is above the container, and open the bottom of that hopper, releasing those pills. Face container again.
6. Reverse to original location of robot.
7. Turn toward the other located container (repeating step 3 with other desired angle to turn to).
8. Drive up to this container (repeating step 4).
9. Turn so that the still-full hopper is above the container, and open the bottom of that hopper, releasing those pills. Face container again (repeating step 5).
10. Reverse to original location of robot. (repeating step 6). Problem is completed.

Software Tasks (Functions)

- Sorting the pills in the large hopper.

○ This function is passed nothing, and returns nothing.

1. Only for the first pill, rotate the medium motor controlling the four-tooth gear, so that a pill rolls in front of the sorting color sensor.
2. Repeated twenty times for the twenty unsorted pills:
    i. Get the color of the pill from the color sensor.
    ii. Depending on which desired color this value is closest to, rotate the medium motor one way or the other to rotate the second four-tooth gear to move it onto the track for its color. This rotation also allows the next pill to enter the sorting chamber.
3. Pills have been sorted, return.

- Searching for pill containers.

○ This function is passed nothing, but returns a list or array containing two values that represent angles or bearing the robot must align with in order to face the two closest other objects in the field of the problem.

1. Create a two-element list / array for the two values to return that represent the angles that point toward the two closest objects in the 360 degree field around the robot, which should be the two pill containers.
2. Create a two-element list that contains the distances from the robot to each of these objects, either starting as arbitrarily high or both as the distance from the robot to whatever is in front of the robot on function call.
3. Create variables both for the initial bearing of the robot on function call and the current bearing of the robot (starts as value of this initial bearing).
4. Turn around, 360 degrees, in small segments (or ‘continuously’ by making this segment extremely small).
    i. At each stop (between each segment of rotation), measure distance from the front of robot to closest object ahead using the ultrasonic sensor.
    ii. If this distance value is lower than one of the lowest distances stored in the list, then replace that value with this new value and record the current bearing in the corresponding place in the area of angles.
5. Return gyroscopic sensor angles that point toward the two closest objects (two pill containers) as a list / array.

- Turning toward a container.

○ This function is passed the angle that the robot should turn to face, and returns nothing.

1. Create variable that represents current bearing, fill this variable with initial bearing of robot on function call.

2. Create variables for the amount the wheels will initially turn (starting with some arbitrary value), and the direction the robot needs to turn (initially fill by checking the current bearing against the desired bearing).

3. Repeated until the current bearing of the robot is within a certain threshold of the desired bearing.

    i. Depending on whether the desired bearing or current bearing is greater (more or less clockwise), rotate the left and right large motors (that are used for driving) in opposite directions to turn the robot in place some amount, to turn the robot toward that desired bearing.

        - If the direction of rotation ever has to switch, decrease the amount the robot turns. (Determine by checking which direction the robot had to turn on the last iteration against this direction, then filling that variable with the current true statement).

    ii. Re-record the current bearing of the robot.

4. The robot is now aligned with the passed angle, return.

- Going up to a container.

○ This function is passed nothing and returns the distance the wheels turned during the execution.

1. Create variable for current distance from the robot to the object, and a variable called distance travelled that records how much the wheels have turned to get the robot to that position, starting at zero.

2. Repeated until the loop is broken: (while True:)

    i. Drive straight ahead (turning the large driving motors forward at the same rate) some small distance.

    ii. Add the amount the wheels turned to the total distance travelled variable

    iii. Record current distance measured by ultrasonic sensor from front of robot to nearest object directly ahead.

    iv. If this value is under a certain threshold (almost touching the object), break out of the loop.

3. Robot has driven up to the object, return the distance the wheels turned during process.

- Dumping pills from appropriate sorted hopper.

○ This function is passed the two colors it is differentiating between for decision making, and returns nothing.

1. Create variable that represents which container the robot has found (start as empty, filled when the color is checked against parameters).
2. Check color of the object directly ahead of the robot using the color sensor attached to the front of the robot.
3. Depending on which desired color this value is closest to:
    i. Turn the robot in place by rotating the driving motors an arbitrary amount in opposite directions, such that the correct sorted hopper is directly above the target container.
    ii. Rotate the large motor that controls the dispensing mechanism, such that the appropriate hopper is opened and the pills may fall.
    iii. Wait arbitrary length of time for all pills to fall.
    iv. Rotate this large motor the same amount the opposite direction to close that hopper and reset the system to neutral.
4. Rotate the robot's wheel motors the opposite direction by the same arbitrary amount to face the container dead on again.
5. The pills have been dumped, return.

- Reversing back to center of field

○ This function is passed the value that was returned by the function that drove the robot to the container, and returns nothing.

1. Create variable for the amount the wheels will initially turn (starting with some arbitrary value, where negative represents backward).
2. Repeated until the current bearing of the robot is within a certain threshold of the desired bearing.
    i. Depending on whether the desired bearing or current bearing is greater (more or less clockwise), rotate the left and right large motors (that are used for driving) in opposite directions to turn the robot in place some amount, to turn the robot toward that desired bearing.
        - If the direction of rotation ever has to switch, decrease the amount the robot turns. (Determine by checking which direction the robot had to turn

on the last iteration against this direction, then filling that variable with the current true statement).

ii. Re-record the current bearing of the robot.

3. Repeat until the value of the distance to travel variable (passed) is within a certain threshold of zero.

   i. Rotate both large wheel motors by the arbitrary value the amount to turn variable, where a negative value represents moving the robot backward.

   ii. Check the value of the distance to travel variable (passed) plus the amount traveled in the step (the current value of that variable).

      - If this value now has an opposite sign to the current value of the distance to travel variable, then multiple the amount traveled in each step variable by -.5, to reverse the direction of travel and make it smaller (such that if the robot overshoots, it can hone in on the original position).

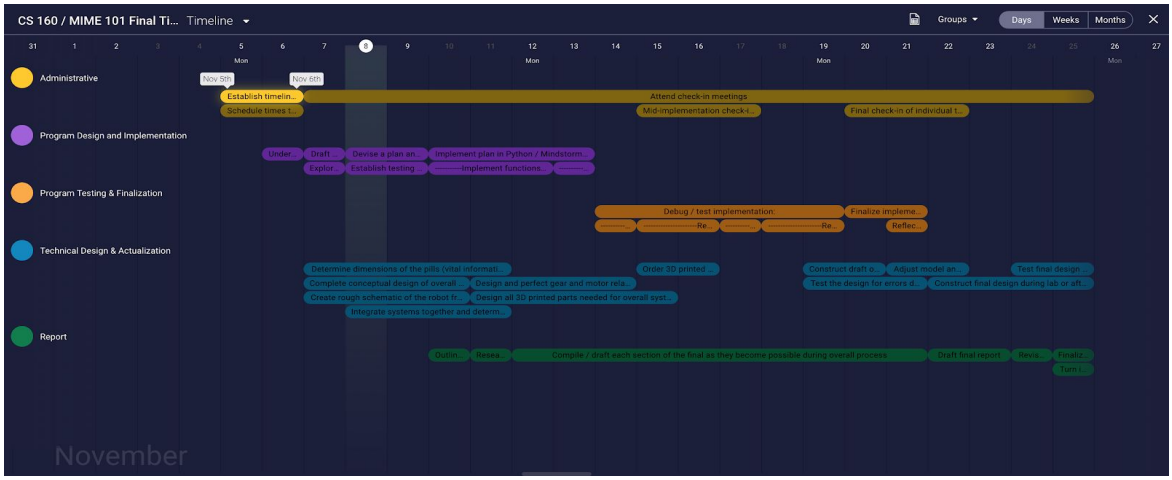
      - Set this value (the sum) as the new value of the passed distance to travel variable.

4. The robot is now extremely close to its original position, return

Gantt Chart

The gantt chart for this proposal was created using the tool “Monday,” a web-based platform for team management and timelining, among other things. Multiple representations of the team’s timeline follow on the next few pages, as well as a link to the interactive version of the timeline and list of tasks for project completion. The viewing of this interactive version may require the user to sign onto the board with an “@oregonstate.edu” email address.

Timeline and List Representation:



CS 160 / MIME 101 Final Timeline

Add View | Unsubscribe | Guests & Subscribers / 2

Administrative

	Person	Status	Timeline	Deadline	Progress
Establish timeline of milestones and requirements		Done	Nov 5 - 6	Nov 6	100%
Schedule times to meet		Done	Nov 5 - 6	Nov 6	100%
Attend check-in meetings		Working on it	Nov 7 - 25		0%
Mid-implementation check-in of progress, supporting each other (fresh eyes o...		Not started	Nov 15 - 17		0%
Final check-in of individual tasks		Not started	Nov 20 - 22	Nov 22	0%
					40%

Program Design and Implementation

	Person	Status	Timeline	Deadline	Progress
Understand problem		Done	Nov 6	Nov 6	100%
Draft out requirements and possible solutions		Done	Nov 7	Nov 7	100%
Explore options for implementaion (ev3dev's validity / feasibility)		Done	Nov 7	Nov 7	100%
Devise a plan and formulate pseudocode for approaches			Nov 8 - 9	Nov 9	100%
Establish testing plan for individual functions and for overall process		Working on it	Nov 8 - 9	Nov 9	0%
Implement plan in Python / Mindstorms' awful interface		Not started	Nov 10 - 13	Nov 13	0%
------Implement functions for searching, driving, checking color of a pill, chec...		Not started	Nov 10 - 12	Nov 12	0%
------Implement overall flow of program, connecting functions to pull functio...		Not started	Nov 13	Nov 13	0%
					50%

Program Testing & Finalization

	Person	Status	Timeline	Deadline	Progress
Debug / test implementation:		Not started	Nov 14 - 19	Nov 19	0%
------First check of functionality on ev3 robot		Not started	Nov 14		0%
----------Test implementation of individual functions in lab or lab after-ho...		Not started	-		0%
----------Test implementation with test-run using competition set-up, if po...		Not started	-		0%
----------Revise implementation as needed		Not started	Nov 15 - 16		0%
------Second chance to test functions, repeating sub-steps		Not started	Nov 17		0%
----------Re-test implementation of individual functions in lab or lab after-...		Not started	-		0%
----------Re-Test implementation with test-run using competition set-up, if...		Not started	-		0%
----------Revise implementation as needed		Not started	Nov 18 - 19		0%
Finalize implementation, check-in with partner one last time		Not started	Nov 20 - 21		0%
Reflect on process, journal on reflection		Not started	Nov 21	Nov 21	0%
					0%

Technical Design & Actualization

	Person	Status	Timeline	Deadline	Progress
Determine dimensions of the pills (vital information)	TB	Working on it	Nov 7 - 11		0%
Complete conceptual design of overall system	TB	Done	Nov 7 - 10	Nov 10	100%
Design and perfect gear and motor relationships	TB	Not started	Nov 11 - 14	Nov 14	0%
Create rough schematic of the robot from the top and side	TB	Done	Nov 7 - 10	Nov 10	100%
Integrate systems together and determine measurements for 3D modeled parts	TB	Not started	Nov 8 - 11	Nov 16	0%
Design all 3D printed parts needed for overall system	TB	Not started	Nov 11 - 15	Nov 15	0%
Order 3D printed parts	TB	Not started	Nov 15 - 16	Nov 16	0%
Construct draft of design during lab or after-hours	TB	Not started	Nov 19 - 20	Nov 20	0%
Test the design for errors during lab or after-hours	TB	Not started	Nov 19 - 21	Nov 21	0%
Adjust model and its technical solutions, accounting for errors	TB	Not started	Nov 21 - 22	Nov 22	0%
Construct final design during lab or after-hours	TB	Not started	Nov 22 - 23	Nov 23	0%
Test final design for new or repeated issues	TB	Not started	Nov 24 - 25	Nov 25	0%
					17%

Report

	Person	Status	Timeline	Deadline	Progress
Outline final report (before any implementation, keeping notes as we go)		Not started	Nov 10	Nov 10	0%
Research real-world equivalents to the problem and possible solutions to the p...		Not started	Nov 11	Nov 11	0%
Compile / draft each section of the final as they become possible during overal...		Not started	Nov 12 - 21		0%
Draft final report		Not started	Nov 22 - 23	Nov 23	0%
Revise final report		Not started	Nov 24	Nov 24	0%
Finalize final report		Not started	Nov 25	Nov 25	0%
Turn in final report		Not started	Nov 25	Nov 25	0%
					0%

Spreadsheet List Representation:

CS 160 / MIME 101 Final Timeline

Administrative

Name	Person	Status	Timeline - Start	Timeline - End	Deadline	Progress
Establish timeline of milestones and requirements		Done	2018-11-05	2018-11-06	2018-11-06	100%
Schedule times to meet		Done	2018-11-05	2018-11-06	2018-11-06	100%
Attend check-in meetings		Working on it	2018-11-07	2018-11-25		0%
Mid-implementation check-in of progress, supporting each other (fresh eyes on our indivi		Not started	2018-11-15	2018-11-17		0%
Final check-in of individual tasks		Not started	2018-11-20	2018-11-22	2018-11-22	0%
						40%

Program Design and Implementation

Name	Person	Status	Timeline - Start	Timeline - End	Deadline	Progress
Understand problem	Nicholas Olson	Done	2018-11-06	2018-11-06	2018-11-06	100%
Draft out requirements and possible solutions	Nicholas Olson	Done	2018-11-07	2018-11-07	2018-11-07	100%
Explore options for implementaion (ev3dev's validity / feasibility)	Nicholas Olson	Done	2018-11-07	2018-11-07	2018-11-07	100%
Devise a plan and formulate pseudocode for approaches	Nicholas Olson	Done	2018-11-08	2018-11-09	2018-11-09	100%
Establish testing plan for individual functions and for overall process	Nicholas Olson	Working on it	2018-11-08	2018-11-09	2018-11-09	0%
Implement plan in Python / Mindstorms' awful interface:	Nicholas Olson	Not started	2018-11-10	2018-11-13	2018-11-13	0%
'----------Implement functions for searching, driving, checking color of a pill, checking col	Nicholas Olson	Not started	2018-11-10	2018-11-12	2018-11-12	0%
'----------Implement overall flow of program, connecting functions to pull functions togethe	Nicholas Olson	Not started	2018-11-13	2018-11-13	2018-11-13	0%
						50%

Program Testing & Finalization

Name	Person	Status	Timeline - Start	Timeline - End	Deadline	Progress
Debug / test implementation:	Nicholas Olson	Not started	2018-11-14	2018-11-19	2018-11-19	0%
'----------First check of functionality on ev3 robot	Nicholas Olson	Not started	2018-11-14	2018-11-14		0%
'----------Test implementation of individual functions in lab or lab after-hours	Nicholas Olson	Not started				0%
'----------Test implementation with test-run using competition set-up, if possible	Nicholas Olson	Not started				0%
'----------Revise implementation as needed	Nicholas Olson	Not started	2018-11-15	2018-11-16		0%
'----------Second chance to test functions, repeating sub-steps	Nicholas Olson	Not started	2018-11-17	2018-11-17		0%
'----------Re-test implementation of individual functions in lab or lab after-hours	Nicholas Olson	Not started				0%
'----------Re-Test implementation with test-run using competition set-up, if possil	Nicholas Olson	Not started				0%
'----------Revise implementation as needed	Nicholas Olson	Not started	2018-11-18	2018-11-19		0%
Finalize implementation, check-in with partner one last time	Nicholas Olson	Not started	2018-11-20	2018-11-21		0%
Reflect on process, journal on reflection	Nicholas Olson	Not started	2018-11-21	2018-11-21	2018-11-21	0%
						0%

Technical Design & Actualization

Name	Person	Status	Timeline - Start	Timeline - End	Deadline	Progress
Determine dimensions of the pills (vital information)	Ty Brewen	Working on it	2018-11-07	2018-11-11		0%
Complete conceptual design of overall system	Ty Brewen	Done	2018-11-07	2018-11-10	2018-11-10	100%
Design and perfect gear and motor relationships	Ty Brewen	Not started	2018-11-11	2018-11-14	2018-11-14	0%
Create rough schematic of the robot from the top and side	Ty Brewen	Done	2018-11-07	2018-11-10	2018-11-10	100%
Integrate systems together and determine measurements for 3D modeled parts	Ty Brewen	Not started	2018-11-08	2018-11-11	2018-11-16	0%
Design all 3D printed parts needed for overall system	Ty Brewen	Not started	2018-11-11	2018-11-15	2018-11-15	0%
Order 3D printed parts	Ty Brewen	Not started	2018-11-15	2018-11-16	2018-11-16	0%
Construct draft of design during lab or after-hours	Ty Brewen	Not started	2018-11-19	2018-11-20	2018-11-20	0%
Test the design for errors during lab or after-hours	Ty Brewen	Not started	2018-11-19	2018-11-21	2018-11-21	0%
Adjust model and its technical solutions, accounting for errors	Ty Brewen	Not started	2018-11-21	2018-11-22	2018-11-22	0%
Construct final design during lab or after-hours	Ty Brewen	Not started	2018-11-22	2018-11-25	2018-11-25	0%
Test final design for new or repeated issues	Ty Brewen	Not started	2018-11-24	2018-11-25	2018-11-25	0%
						17%

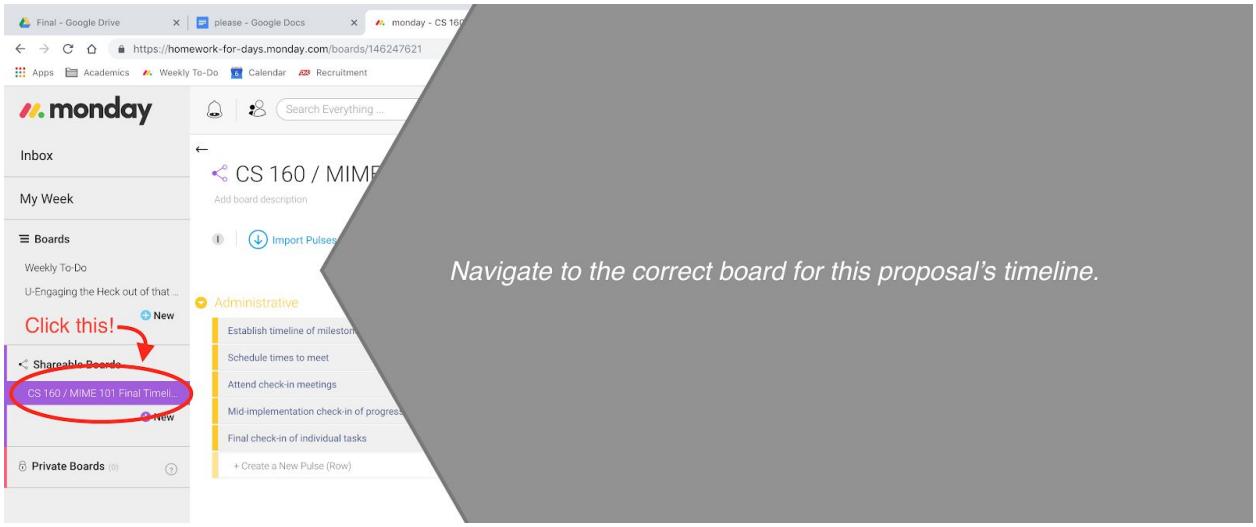
Report

Name	Person	Status	Timeline - Start	Timeline - End	Deadline	Progress
Outline final report (before any implementation, keeping notes as we go)		Not started	2018-11-10	2018-11-10	2018-11-10	0%
Research real-world equivalents to the problem and possible solutions to the problem		Not started	2018-11-11	2018-11-11	2018-11-11	0%
Compile / draft each section of the final as they become possible during overall process		Not started	2018-11-12	2018-11-21		0%
Draft final report		Not started	2018-11-22	2018-11-23	2018-11-23	0%
Revise final report		Not started	2018-11-24	2018-11-24	2018-11-24	0%
Finalize final report		Not started	2018-11-25	2018-11-25	2018-11-25	0%
Turn in final report		Not started	2018-11-25	2018-11-25	2018-11-25	0%
						0%

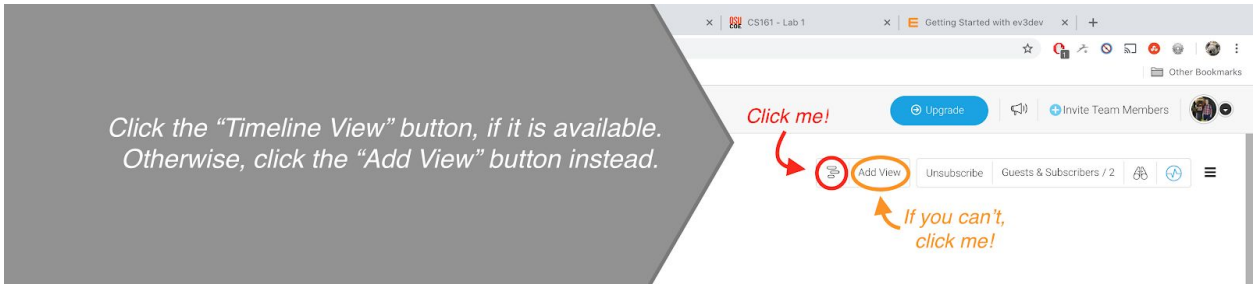
Online Interactive Board and Timeline:

https://homework-for-days.monday.com/boards/146247621

To view interactive board and timeline, one may need to sign on or register with an "@oregonstate.edu" email address. This board hosts multiple tasks lists for the student Nicholas Olson and various group projects he is a part of. Once one has logged on, the only board one need concern themself with is the one titled "CS 160 / MIME 101 Final Timeline." If one finds themself looking at a different board, they need only click on the button labelled for this timeline on the left sidebar of the browser window:



Then click on the "Timeline View" button near the top right of the window. If one cannot see this button, they must click instead the "Add View" button to add the timeline view as an option, then they may open this interactive timeline:



Then the interactive timeline view will be an option for the viewer to open and sort through. The list that follows is the complete list of tasks the team has laid out for the process, including deadlines, timelines, the progress of each task and its category, and whether each task is completed.

Design Economics

List of Materials and Costs

Item Name	Item Part Number	Unit Price	Estimated Quantity	Resulting Cost
Bushing, 1/2 - module, yellow	4239601	$0.06	6	$0.36
Bushing, 1 - module, gray	4211622	$0.06	10	$0.6
Connector peg with friction, 2-module, black	4121715	$0.06	24	$1.44
Connector peg with axle, 2-module, beige	4666579	$0.06	4	$0.24
Connector peg, 3-module, beige	4514554	$0.06	2	$0.12
Connector peg with friction/axle, 2-module, blue	4206482	$0.06	10	$0.6
Connector peg with friction, 3-module, blue	4514553	$0.06	16	$0.96
Axle with stop, 4-module, dark gray	4560177	$0.20	2	$0.4
Axle with stop, 8-module, dark gray	4499858	$0.25	2	$0.5
Axle, 3-module, gray	4211815	$0.06	4	$0.24
Axle, 5-module, black	4211639	$0.13	6	$0.78
Axle, 7-module, gray	4211805	$0.14	6	$0.84
Pointer, 3-module, white	4173941	$0.09	4	$0.36
Beam, 3-module, green	6007973	$0.09	4	$0.36
Beam, 3-module, red	4153718	$0.09	4	$0.36

Angular beam, 2x4-module, red	4141270	$0.21	2	$0.42
Angular beam, 2x5-module, gray	4211713	$0.21	2	$0.42
Angular beam, 2x5-module, white	4585040	$0.21	4	$0.84
Beam, 5-module, gray	4211651	$0.15	4	$0.6
Beam, 7-module, gray	4495930	$0.21	2	$0.42
Beam, 9-module, gray	4211866	$0.32	2	$0.64
Beam, 13-module, gray	4522934	$0.32	2	$0.64
Beam, 15-module, white	4542578	$0.34	6	$2.04
Angular beam, 4x4-module, white	4509912	$0.25	2	$0.5
Angular beam, 3x7-module, gray	4211624	$0.25	2	$0.5
Double angular beam, 3x7-module, white	4495412	$0.50	4	$2
Frame, 5x7-module, gray	4539880	$0.62	1	$0.62
Frame, 5x11-module, gray	4540797	$0.90	1	$0.9
Double connector peg, 3x3-module, gray	4225033	$0.15	2	$0.3
Cross block, 3-module, dark gray	4210857	$0.08	2	$0.16
Double cross block, 3-module, black	4121667	$0.08	2	$0.16
1/2 triangle beam, 5x3-module, gray	6009019	$0.13	4	$0.52
3-spoke angular block, 3x120 deg, gray	4502595	$0.21	2	$0.42
Angular block 1, 0 deg, black	4107085	$0.08	2	$0.16

Angular block 2, 180 deg, black	4107783	$0.08	2	$0.16
Gear, 8-tooth, dark gray	4514559	$0.15	2	$0.3
Gear, 24-tooth, dark gray	4514558	$0.35	3	$1.05
Worm gear, gray	4211510	$0.18	2	$0.36
Gear, 4-tooth, black	4248204	$0.25	4	$1
Hub, 43.2x26 mm, gray	4634091	$0.28	2	$0.56
Low profile tire, 56x28 mm, black	6035364	$0.34	2	$0.68
Steel Ball, silver metallic	6023956	$1.00	1	$1
Ball bearing, dark gray	4610380	$0.85	1	$0.85
EV3 Brick	6009996	$197.99	1	$197.99
Rechargeable Battery	6012820	$88.99	1	$88.99
Large Motor	6009430	$26.99	3	$80.97
Medium Motor	6008577	$21.99	1	$21.99
Color Sensor	6008919	$41.99	2	$83.98
Ultrasonic Sensor	6008924	$31.99	1	$31.99
Gyro Sensor	6008916	$31.99	1	$31.99
Cable, 25 cm / 10 in.	6024581	$2.83	4	$11.32
Cable, 35 cm / 14 in.	6024583	$2.83	3	$8.49
Cable, 50 cm / 20 in.	6024585	$2.83	1	$2.83

USB Cable	6036901	$6.99	1	$6.99
			Total Cost	$593.91

Analysis of Costs

The most significant problem with this cost estimate is the lack of information about the cost of 3D printed items. Without this information to compare against the cost of Lego parts, it difficult to decide whether to 3D print more parts (such as the ramp system) or primarily use Lego. Thus, this analysis we detail only the costs of Lego parts.

By far, the most costly items are the EV3 Brick, the rechargeable battery, and the leading sensors. A significant amount of money would be saved if the second color sensor were removed and a cheaper alternative for determining which cup has been located was discovered. Another proposal was to cut out a motor by powering the sorting system with a wheel motor. Consequently, the robot would spin in a circle while sorting, but this would not dramatically impact the robots function. Alternatively, replacing a medium motor, perhaps in the pill dispensing system, could also reduce costs.

References

Hellem, Amy. “Resources for the Visually Impaired.” *All About Vision*, Oct. 2016, www.allaboutvision.com/lowvision/resources.htm.

“Disability Overview.” *National Disability Authority*, www.nda.ie/Disability-overview/.

Bourne RRA, Flaxman SR, Braithwaite T, Cicinelli MV, Das A, Jonas JB, et al.; Vision Loss Expert Group. “Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis.” *Lancet Glob Health*. 2017 Sep;5(9):e888–97.

Fricke, TR, Tahhan N, Resnikoff S, Papas E, Burnett A, Suit MH, Naduvilath T, Naidoo K, “Global Prevalence of Presbyopia and Vision Impairment from Uncorrected Presbyopia: Systematic Review, Meta-analysis, and Modelling, Ophthalmology.” 2018 May 9

“Vision Impairment and Blindness.” WHO, World Health Organization, 11 Oct. 2018, www.who.int/en/news-room/fact-sheets/detail/blindness-and-visual-impairment.

“Blindness and Vision Impairment.” *Gateway to Health Communication & Social Marketing Practice*, Centers for Disease Control and Prevention, 15 Sept. 2017, www.cdc.gov/healthcommunication/toolstemplates/entertainmented/tips/Blindness.html.

“Metrix Analytic Color Sorter.” *Metrix Analytic Lab Color Sorter*, VMek Sorting Technology, vmek.com/metrix.html.