**The Final Fig Newton**

**Final Design Report**

Group 11

Caroline Aitken, C. Tyler Booth, Usman Dauda, Claire Lizotte

**Executive Summary**

The group was tasked with designing a robot that would get a cube off a wall and place it under a pyramid without any human assistance. The problem was defined, then various solutions were brainstormed. Decision matrices selected the concepts for the design, then these concepts were refined and iterated in the building phase. Overall, the prototype was well designed. The final robot was designed with three arms, four wheels, a slide for the cube, and a few sensors to tell the robot what is happening around it. One of the arms is to “pickup” the pyramid, another to pick up the cube, and the last to assist in picking up the cube.

Not all the sensors that were decided upon were used on the prototype due to shipping time. So, compromises were made in their place. It was found that having a few extra sensors on the robot would of increase its capabilities and performance. Also, using a different microcontroller would allow all the inputs/outputs to not be divided up between two separate controllers. This would have increased the speed and reliability of the robot. Given more time the robot could be redesigned to increase the turning ability and allow for the desired sensors to be ordered.

Table of Contents

[Introduction 3](#_Toc479320090)

[The Task 3](#_Toc479320091)

[Background Information 3](#_Toc479320092)

[Design Specifications 3](#_Toc479320093)

[Discussion 4](#_Toc479320094)

[Overall Final Design 6](#_Toc479320095)

[Iterations and Their Effects on the Prototype 8](#_Toc479320096)

[Conclusions 10](#_Toc479320097)

[Recommendations 12](#_Toc479320098)

[Appendix 14](#_Toc479320099)

[Works Cited 22](#_Toc479320100)

**Figures and Tables**

[Figure 1 - Top View of Prototype 6](#_Toc479320101)

[Figure 2 - Angled View of Prototype 7](#_Toc479320102)

[Figure 3 - QFD Part 1 16](#_Toc479320103)

[Figure 4 - QFD Part 2 17](#_Toc479320104)

[Figure 5 - QFD Part 3 18](#_Toc479320105)

[Figure 6 - QFD Part 4 19](#_Toc479320106)

[Figure 7 - QFD Part 5 20](#_Toc479320107)

[Figure 8 - Drive System Decision Matrix 21](#_Toc479320108)

[Figure 9 - Cube Manipulation Decision Matrix 21](#_Toc479320109)

[Figure 10 - Pyramid Manipulation Decision Matrix 22](#_Toc479320110)

[Table 1 - Objectives That Were Not Met 10](#_Toc479320111)

[Table 2 - Objectives and Constraints 14](#_Toc479320112)

[Table 3- Engineering Specifications 15](#_Toc479320113)

# **Introduction**

The group was tasked with creating an autonomous mechatronic device that would locate a Power Pack and place it under the Power Pyramid in the Tesseract Power Co. plant. The robot will be required to manipulate the Power Pack and Power Pyramid, deal with collision avoidance, and traverse obstacles on the ground. This report summarizes the design process, states the results, and provides recommendations to further improve the robot.

## *The Task*

According to Marvel Cinematic Universe, a tesseract is a cube that contains a stone which has the ability to emit unlimited energy (Tesseract). However, this energy is toxic to humans, and direct contact made with the tesseract could result in death. Therefore, to harness this endless power, an autonomously controlled robot must approach the tesseract. Per the design specifications, the Tesseract Power Cooperation has been able to design a machine that can control and harness the power emitted by the tesseract. To do this, this machine – designed in the shape of a pyramid – must be lifted carefully, and have the cube placed underneath it in the space the company allocated for it.

## *Background Information*

Tesseract Power Co. Decided it was simplest to design the power pyramid to emit infrared (IR) light in specific and discrete packages: it may sometimes emit signals that equate to the letters 'A' and 'E'. At other times, it will emit the letters 'I' and 'O'. However, they will only be emitted in these pairs and no other combinations.

## *Design Specifications*

The best way to approach this task is to evaluate it in a systematic process. Ultimately, there are three major components of this assignment: get the cube, find the pyramid, and place the cube under the pyramid. However, other requirements for the autonomous robot arise from the constraints of the task.

As stated in the task, the robot must be completely autonomous. Therefore, its ability to function must be ensured. No fixes may be made on site during the procedure itself, as this will result in fatalities. Ultimately, this means that quality cannot be sacrificed: the most appropriate materials will be chosen, even if it results in a more expensive robot.

In addition to this point, the robot need not accommodate any humans. Therefore, it is possible to use as little material needed as possible to ensure the function of the robot. For example, a hub for the entire frame need not exist, as there is nothing alive that needs to be protected.

Another way that cost reduction will be implemented will be in attempting to combine the purposes of components in the robots. However, simplicity is also desired to reduce the chance of a part malfunctioning. So, there will be a balance between simplicity and using the same systems to do multiple tasks.

The area in which the cube will be found and the pyramid has been placed has been cordoned off in an area anywhere between 6x6 feet to 10x10 feet, in varying combinations of rectangular shapes that fit these parameters. Thus, the robot’s movement may be restricted upon its implementation. A response strategy must be created to avoid the halt of function of the robot. Ultimately, this will also impact how the robot turns and what the turning radius should be.

Due to restrictions of the facility, the volume of the robot itself is ultimately restricted as well. It must be able to fit within a dimension of 10 cm by 17 cm by 15 cm.

# **Discussion**

The first step in the design process was to develop an understanding of the problem and of the customer requirements.

Need and goal statements were constructed and are shown below.

*Need Statement:* The Power Pyramid needs to have a Tesseract underneath it.

*Goal Statement:* Design an autonomous mechatronic device that can obtain a tesseract power pack and place it under a pyramid.

The customer requirements were then translated in to design objectives, requirements, and constraints to further define the problem. Examples of these include:

* The product must fit in a locker (constraint)
* The product can get the cube under the pyramid (requirements)
* To have the product complete the task in under 5 minutes (objective).

For a complete list see Table 2 - Objectives and Constraints – in the Appendix.

Engineering specifications were generated from the list of requirements. Examples of the engineering specifications include:

* Have less than 15 moving systems
* Product and prototype must fit in a space of 10 cm by 17 cm by 15 cm
* Startup time of the product is less than 5 min

See Table 3- Engineering Specifications – in the Appendix for a complete list.

The requirements and engineering specifications were put into a Quality Function Deployment chart (QFD) to assist in the understanding of the problem.

Next, various concepts were brainstormed for each sub system. Decision matrices were used to form objective decisions of which concept would be utilized.

For the cube manipulation, it was thought that a grasper, an electromagnet, or a sweeper could be used. The sweeper was rejected because it would not be as consistent as the other two options. The decision matrix (see Figure 9 - Cube Manipulation Decision Matrix - in Appendix) indicated that the electromagnet was the superior concept, however it was rejected because the available electromagnet was not strong enough to consistently pick up the cube.

For the drive system, concepts included: Six wheels, tracks, and four wheels. Six wheels at staggered height, which was rejected because of the constraint of the overall size of the product. Tracked drive, which was discarded due to the objective to minimize mass. Four wheels, which was the superior concept per the decision matrix (see Figure 8 - Drive System Decision Matrix - in Appendix), was implemented using large wheels to overcome the power conduits and a chain that allowed the motors to sit in front of the wheels due to the size constraint.

There were four main concepts for pyramid manipulation. First, it could be pushed against a wall of the course and an arm could tip it. This idea was rejected because it is uncertain where in the course the pyramid would be and it may not be close to a wall and it could get caught on a power conduit. The second concept involved picking the pyramid up with a grasper. However, it was thought that depending on the angle the pyramid is approached it may not be able to pick it up. Third, it was thought that the pyramid could be raised using a forklift. However, there were uncertain if the robot could consistently get the forklift under the pyramid. Additionally, the space required to implement the forklift design would not fit the requirements for the overall size of the robot. The result of the decision matrix (see Figure 10 - Pyramid Manipulation Decision Matrix - in Appendix) was to tip the pyramid by pushing it against a wall that the robot places next to the pyramid. This fourth concept was implemented by having an arm on the front of the robot that lowers once the pyramid has been found. A set of wheels on the arm is then spun to tip the pyramid.

After the concepts were chosen, they were developed into more detailed models. These were made on SolidWorks, and a prototype was constructed.

Research on various parts and materials was conducted to choose and make components that are suitable to meeting the design requirements. For example, it was decided to use stamped aluminum for the frame because it is light, durable, and it can be used with sheet metal processes.

## *Overall Final Design*

The final design is a rectangular chassis with three arms (a back arm, cube arm, and pyramid arm), four wheels, a chute, IR distance sensor, limit switch, and IR communication receiver. The four wheels are inside the frame. The back two wheels are geared to motors using a chain, and the robot steers using skid steer.

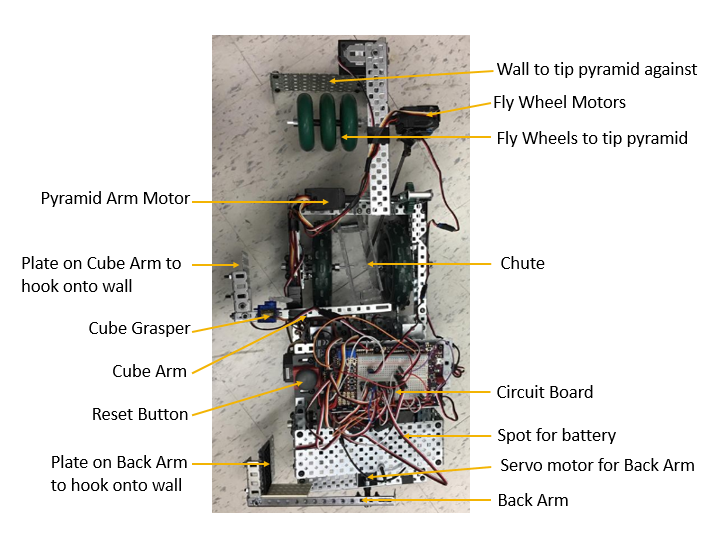


Figure 1 - Top View of Prototype

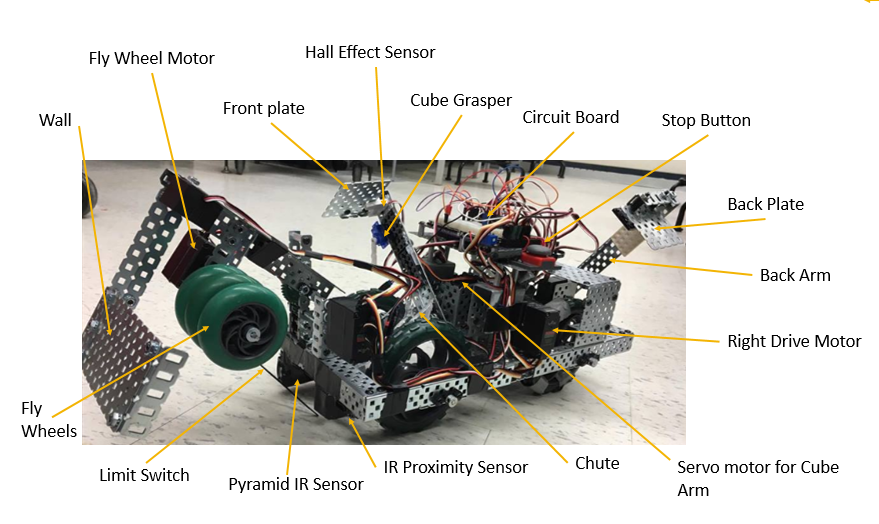


Figure 2 - Angled View of Prototype

The back arm is at the very back of the robot and is actuated with a servo motor to move in the vertical plane. It has a plate on the end of it to hook onto the wall to prevent the robot from drifting away from the wall as it tries to find the cube. Also, if the cube is initially behind the cube arm, the back arm will push it along the wall. The robot can reverse and then get the cube with the cube arm.

The cube arm also moves in the vertical plane through a servo motor. It is about one third the length of the robot from the front. On the end of the arm there is a hall effect sensor to locate the cube, a servo motor with an arm to pick up and hold the cube, and a plate to hook onto the wall.

The pyramid arm is actuated by a motor that will raise and lower it. Attached to this arm there is a set of wheels (also actuated by another motor) and the fake wall. The wheels tip the pyramid against the wall and hold it in place while the cube is put under it.

The cube is put under the pyramid by the cube arm dropping the cube down a chute, to consistently drop the cube in the same spot with respect to the robot. The robot then backs up so that the cube is under the pyramid while the pyramid arm is holding the pyramid tilted. The set of wheels reverses, allowing the pyramid to fall over the cube, to finish the task.

The pyramid is located through the IR communication signal it emits. There is also a limit switch that hangs from the pyramid arm to indicate when the pyramid is near the robot.

The IR distance sensor was place on the front left corner of the robot. This sensor was used to determine if there was a wall in front of the robot. Used in the initial cube acquisition and while locating the pyramid, the robot would know how it was from any wall directly in front of it.

## *Iterations and Their Effects on the Prototype*

Throughout the prototyping and iteration phase, several aspects were changed. Some of the main iterations were:

* The first major iteration was to add a third arm to the back of the robot. This was done to increase the overall range that the robot can locate the cube. Now, if the cube is behind the cube arm, the back arm will push it along the wall, so the robot can then back up and then obtain the cube. This allows the robot to obtain the cube if it is in the front corner as well because the robot could perform a 180’ turn and lower the back arm on the other side to bring the cube to a point where the cube arm could grab it. Due to time constraints, the prototype was not programmed to perform such a maneuver.
* Originally, the wall to tip the pyramid was being deployed from the side of the robot. This was changed to have it come from above the robot to have a larger clearance for the cube. However, this increased the weight of the arm, so the servo motor was replaced with a motor.
* Also, related to the tipping of the pyramid, the original concept was to have an arm to push the pyramid. This was changed to a set of wheels because of the spacing inside of the robot already being tight. The wheels were easier to construct and code. Additionally, the wheels worked well for tipping the pyramid as it has a high coefficient of friction and could hit the pyramid at multiple different angles and still successfully tip it.
* The location of the pyramid IR sensor was moved from underneath the middle of the chassis to the front right corner to allow it to pick up the IR signal more frequently. The sensor was originally underneath the robot to have the sensor be in line with the desired placement of the pyramid. It was moved because the sensor would often not read the signal when it was in front of the pyramid because the signal was blocked by the chute and the wheels. The disadvantage of this iteration is that now when the pyramid is located, it lines up with the right side of the robot instead of the center. This is accounted for by having the robot slightly turn right, so it lines up with the pyramid arm, before the robot manipulates the pyramid.
* The IR sensor for the pyramid only tells if the robot is facing the pyramid or not. It does not indicate how close the robot is to the pyramid. The original plan to determine when the robot reached the pyramid was to have a light sensor pointing at the ground from the pyramid arm. Since the robot would have to operate in various lighting conditions, it was decided to use an IR distance sensor instead. Unfortunately, we were unable to obtain a second IR proximity sensor, so a limit sensor is being used. A zip tie extends the limit switch, as it is light, firm enough to move when the pyramid hits it, yet flexible enough that it does not break when the pyramid arm lowers. This affected the design because the limit switch is very finicky and frequently triggers before the robot is at the pyramid. The solution to this would be to obtain an IR proximity sensor, however due to time constraint that was not feasible.

After most of the prototype was assembled, coding the prototype began. The base code was taken from the provided line following code because there were many functions of that code that the team liked and wanted to implement in the prototype, such as the ability to calibrate the motors and have a resting mode and a run mode.

This was a very iterative process. Both microcontrollers on the board were used to control the robot. The general structure of the code was a switch statement with the task broken up into many smaller sections. This allowed the microcontrollers to alternate which one was the action was dependent on.

For the first section of code (to pick up the cube), the code was written as it was tested, to simplify debugging. The key issues with this section involved the microcontrollers not communicating properly. There is a bug somewhere that causes a 10-12 second delay when the slave is communicating to the master. Many different methods were implemented to try to remove this delay, however the exact cause of the delay was not found.

There were many issues and iterations with the locating the pyramid code. The fundamental problem was that the robot would not register that it was facing the pyramid. To fix this the pyramid IR sensor was moved to the slave microcontroller, however this introduced the 10-12 second delay, which was not optimal. The slave easily read when it was facing the pyramid and not, but this information was not communicated to the master microcontroller quickly enough to respond to the signal. The pyramid IR sensor was returned to the master microcontroller in the hope of getting that to work. It was decided to move on to coding the tipping of the pyramid and come back to locating the pyramid, however due to the time constraint, this section of code was not made to work.

If the robot is assisted in locating the pyramid, it can tip the pyramid and getting the cube underneath. This was coded similarly to obtaining the cube, where tasks rotate between the master and slave.

If there was more time, then attempts would be made to get the delay in the communication removed, which would make it much simpler to locate the pyramid.

# **Conclusions**

*Objectives We Met*

* The number of moving systems is under 15 (there are 7): By designing versatile components for the robot, multiple subsystems were created that could perform more than one of the small tasks required of the robot. An example of this would be the flywheel system designed to pick up the pyramid. By using round wheels that spun at a common axle, it was avoided having to use multiple parts to align and tip the pyramid.
* The cost of the prototype was under budget: the robot made use of available components in the lab, and only a few components were custom made or ordered for design purposes.
* The number of user set up functions is under 5: the only user function to be performed is motor calibration.
* Number of successful runs to pick up the cube is over 80%: By creating an arm that also served as a rail when it came down, the robot could effectively “latch” to the wall to avoid the robot drifting away from the wall. This not only removed the need for a sensor to monitor the distance from the wall, it enabled us to pick up the tesseract in the same location on our robot each time.
* The product fits in a locker: To fit the robot inside the locker, the components were designed to arm to fold and start in an upwards position. This made use of the vertical space of the locker, so all components of the robot could fit within the locker space.
* It is capable of driving on various surfaces. The implementation of performance wheels that have variable tread lets us change the material of the tread to best suit the different surfaces the robot would have to drive over.
* It is capable of driving over the power conduits: The 6” wheels of the robot provide enough clearance for the robot to drive over the conduits, and the gear ratio of the drive train allows for enough torque to power the robot over the conduits.
* The mass of the product is under 5kg: From complete bill of materials of the components used in the Solidworks model, the robot has a total weight of just over 4kg.
* The accuracy of locating the cube on the first pass is greater than 80%: When the arms latch on to the side wall, the robot will almost always follow the wall, and will be able to detect the cube along the wall.
* Can distinguish between the 4 different IR frequencies the pyramid emits: The robot has different modes for picking up both “A/E” and “I/O” frequencies.
* Weight of objects that can be lifted/tipped is greater than 500g: The flywheel has a relatively high coefficient of kinetic friction (about 0.4) and when forced against the wall at the end of the arm, will be able to generate enough consistent frictional force to lift an object.

Table 1 - Objectives That Were Not Met

|  |  |  |
| --- | --- | --- |
| **Objective We Did Not Meet** | **Why It Was Not Met** | **Steps That Could Be Taken** |
| The operational runtime less than 5 minutes | The robot is not capable of locating the pyramid. | Perform more tests and iterations on the sections of code that focus on locating the pyramid. |
| Delays During Operations are less than 3s | There is a bug in the code that causes a delay when the two microcontrollers are communicating. | Spend more time debugging the code to find and remove what is causing the delay.  The delay is due to additional serial communication channels. |
| Turning Radius is less than 10 cm | The robot is long which results in sliding while turning | Iterate the physical design to have the robot be shorter or change the drive system |
| Accuracy of locating the pyramid on the first pass | The robot can occasionally locate the pyramid. There is a delay with the communication. | Debug the code. |
| Obtain the cube if it is in the very corners | The arms that manipulate the cube can only move in the vertical plane. | Redesign the arms to be able to rotate, or have the back arm on angle so it can pull the cube forward if it is in the corners. |

The outcome of the project is a product that can complete most of the tasks of getting the Tesseract under the Power Pyramid. The prototype can locate and obtaining the cube and, if assisted in locating the pyramid, it can manipulate the cube and pyramid to get the cube under the pyramid. Many of the objectives were met, such as the size constraint, the ability to get the cube, the weight objective, not cause damage to the operational area, and the ability to drive over the power conduits. The few objectives the team was unable to meet in the time limits are in Table 1 - Objectives That Were Not Met; such as operational time, turning radius, and obtaining the cube in the corners. The objectives could be met if there was a larger period to allow for the code, debugged, and slight modifications to the design. For example, the prototype has the potential to locate the pyramid, however it is unable to do so due to bugs in the code.

The use of two separate microcontrollers communicating across a serial connection caused a delay in the code, roughly 10 to 12 seconds. The delay appeared to be with the main board reading the secondary’s current state. This slowed down the overall timing of completing the different tasks. For example, the cube acquisition was done on the second board. When the cube was acquired, the robot should have stopped in place before moving onto the next task. Instead, it continued to drive forward reaching the bottom of the course. The robot would be stopped by its IR distance sensor, as that was on the main board with the drive motors, causing it not to run into the wall. Additionally, the software files used for the secondary serial connection caused the motors and servos to periodically twitch. This caused problems because the robot would move in ways that it should not.

During the iterations, it was decided to use a second IR distance sensor to locate when the pyramid was in front of the robot. However, the additional IR sensor was not able to be ordered and shipped in time. Therefore, a limit switch with a lightweight flexible extension on it was used in its place. The plan was to have the limit switch trigger when the robot came close to the pyramid, using the first IR distance sensor, located on the front left, to tell the robot that it is not next to a wall. Once the limit switch was triggered, the robot would stop and preform a slight turn to the right to accommodate the placement of the limit switch in relation to the pyramid arm. This ended up being a miserably daft choice. The limit switch would trigger when the robot went over a conduit, either because it would catch or due to the robot jerking around with the motor spasms.

The overall length of the robot worked well for the weight distribution but was not advantageous for the turning radius. The robot could turn by driving one wheel forwards and the other backwards, causing it to skid across the ground. The issues that arose from this were caused by much of the weight being towards the front, reducing the traction at the rear tires. Additionally, the operational area was faintly covered in a layer of dust. The dust would get picked up by the tires, further reducing the amount of grip the tires had with the ground. The motors also had a slight delay in starting, the left would start before the right. The delay in combination with the periodic motor spasms and loss of traction caused many undesirable actions in the robot’s performance.

The robot was designed with two arms to hold onto the wall when finding the cube and an IR distance sensor on the front to stop it from driving into the wall in front of it. It was found that the robot lacked enough information about its surroundings. Due to the nature of the steering, when the robot tried to turn it would also move backwards a slight amount. This caused the robot to back into walls if it was too close to the edge of the course. Additionally, if the robot did not find the cube on the wall in the first pass, the plan was to program it to turn around to check the bottom corner of the course. This was because roughly the first third of the robot was not reachable by the cube arm. In turning around, we could execute the same process as the top of the course to get the cube out of this area. However, the robot would have no way of knowing that it was beside the outside wall or against the bottom wall. The addition of three extra IR distance sensors would have allowed the robot to know exactly how far any wall was from any side of the robot. This would have increased the capabilities and functionality of the robot.

# **Recommendations**

After the testing done by the group and the performance in the challenge, several design changes could be implemented. There were issues with: code delays and board communications, pyramid location and acquisition, turning, and overall sensory input of surroundings.

The issues with the delays in the code were chiefly caused by using two microcontrollers communicating over an additional serial communication. The software files used to create the additional non-native serial connections caused the delay in the code and the servos and motors to jerk causing unwanted movements. These issues could be solved in one of two ways; write original code for the additional serial communication ports or use a larger microcontroller. The latter would prove to be the most efficient way of achieving this goal. If we wrote original code, this could cause further bugs in the code to appear. Using a larger microcontroller would increase the number of I/O ports available and remove the need of a second controller. Additionally, since more sensory input are required, the expansion of more I/O port would accommodate this need. Bringing everything onto a single microcontroller would increase efficiency in the codes since one board would not be waiting for instructions from another.

As discussed in the results section, the pyramid acquisition was done by using a limit switch. This was not the optimal choice and a IR distance sensor was preferred. However, due to timing, it was unable to be obtained. This sensor would have been mounted on the pyramid arm, being held at a viewpoint looking at where the arm would come down. With the sensor mounted in this setup, the robot would be able to look for the top third of the pyramid and not get faulty reading when going over conduits.

Redesigning the casing for the pyramid IR sensor so that it is smaller would help because then it could be mounted in line with the chute at the front of the robot, instead of on the right corner. This would remove the need for a correction turn after the pyramid has been located.

During the performance of the robot turning was a major issue. When testing, the robot could turn, but achieved this by sliding around. One way to help fix this would be to reduce the overall length of the robot by redesigning the complete layout. This would not be the best option as it would only increase the turning ability slightly. A second option would be raising all the components up, to allow the addition of gears and chains attaching the back-drive wheels to the fronts. By connecting the back and front wheels, the whole side of the robot would be used to drive. This would greatly improve the turning capability of the robot since this would not have the front of the robot as semi-fixed point. If one side drives forwards and the other backwards, the robot would be able to turn while keeping in the same relative position.

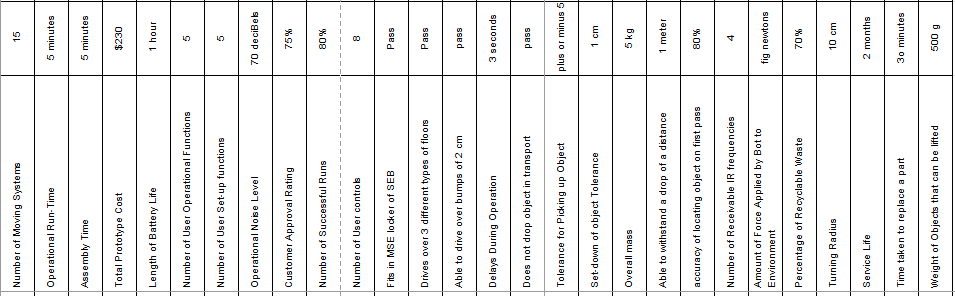
During the building and testing of the robot the group was focused on getting the cube off the wall and locating the pyramid. There was an oversight of what happens if the robot gets stuck in a corner or need to turn the bot on the wall. The cube arm and back arm were designed to work on both sides of the robot. However, the robot would still need to turn completely around and back it into the corner to get the cube out of the last section of the wall at the bottom of the course. Due to the time constraints, not much thought was put into this. Nevertheless, during the competition, it was realized that the robot was not receiving enough sensory input about the surroundings. The additional inputs required would be IR distance sensors on each side of the robot. These inputs would then be use so the robot would know how much room is on each side of it at any giving point in time. Furthermore, these sensors could be implemented to help locate the pyramid. As the robot traverses down the first wall, it would take readings from the opposing side. It would be able to note when it came across the pyramid in comparison to where it found the cube. Then through some simple trigonometric calculations, the robot would know how far it should turn before it needed to start scanning for the pyramid.

# **Appendix**

Table 2 - Objectives and Constraints

|  |  |
| --- | --- |
| * Move Forward * Get over the Power Conduits * Turn * Not slip * Minimize size and mass * Consistently locate the tesseract * Can manipulate the tesseract * Consistent operation * Durable * Fits in a locker * Easy user interface * Low maintenance * Does not damage environment | * Does not drop the tesseract * Accurately locates the tesseract * Consistently locate the pyramid * Can manipulate the pyramid * Does not drop the pyramid * Accurately locates the pyramid * Can expose the bottom of the pyramid * Consistently successful * Minimum number of moving systems * Competes tasks quickly * Cost effective * Recyclable parts |

Table 3- Engineering Specifications



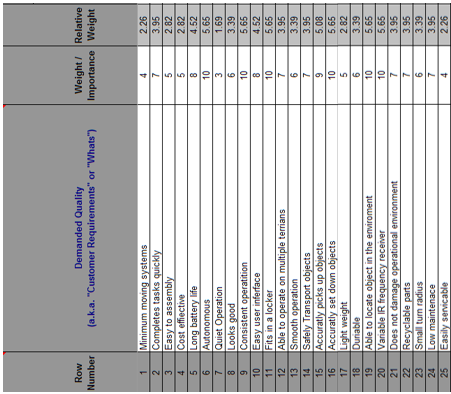


Figure 3 - QFD Part 1

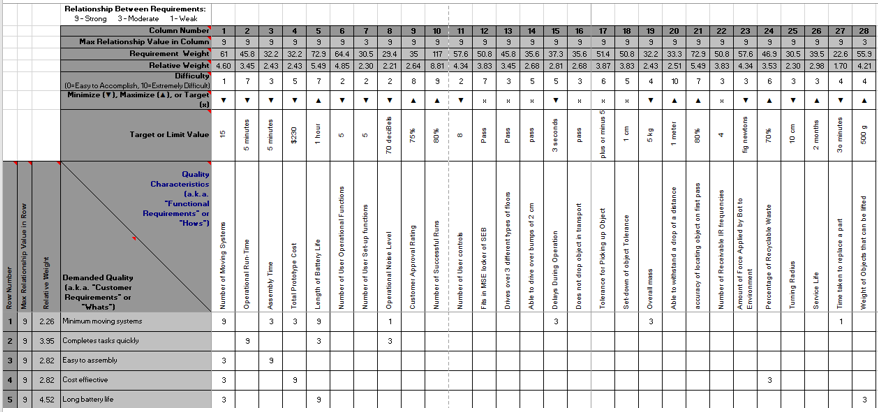


Figure 4 - QFD Part 2

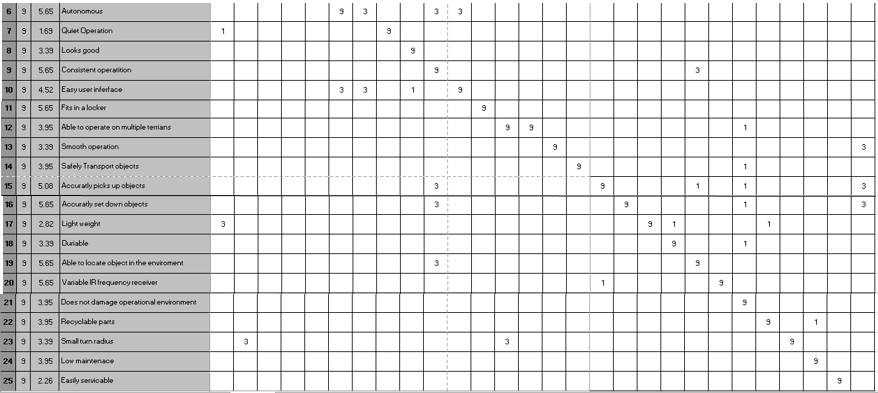
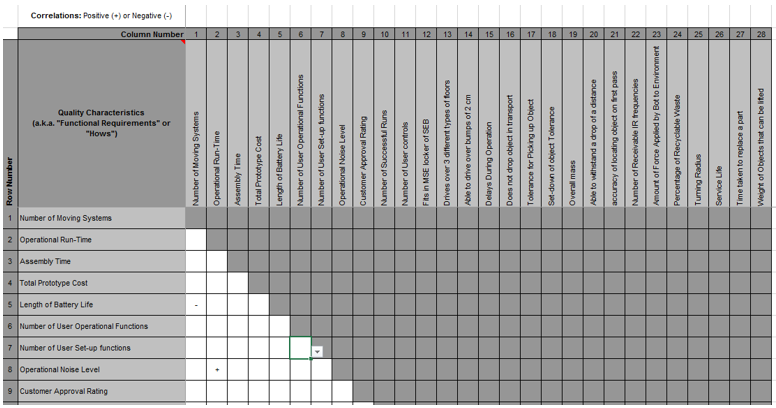


Figure 5 - QFD Part 3



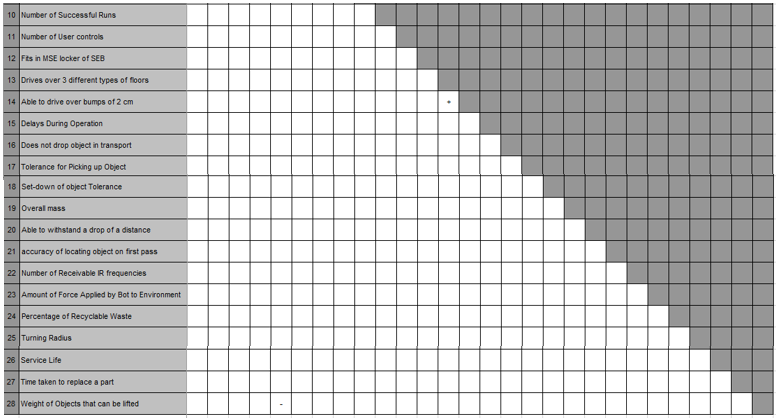


Figure 6 - QFD Part 4

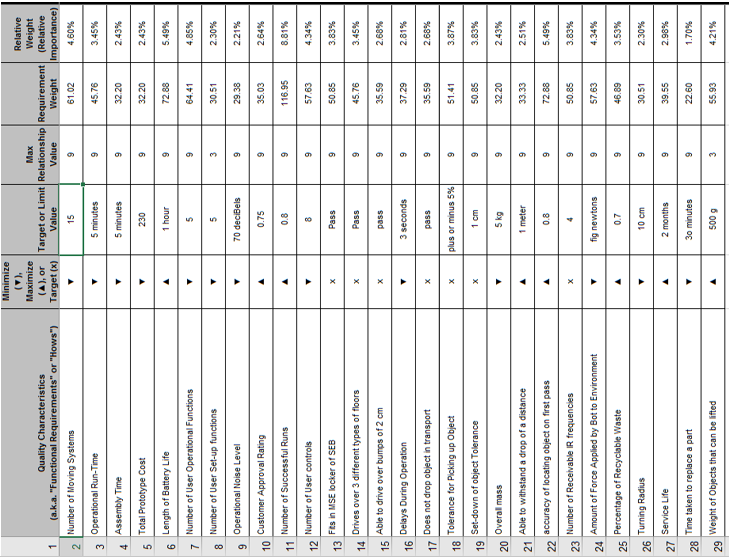


Figure 7 - QFD Part 5

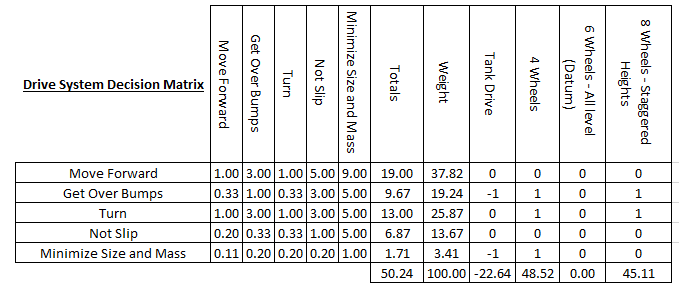


Figure 8 - Drive System Decision Matrix

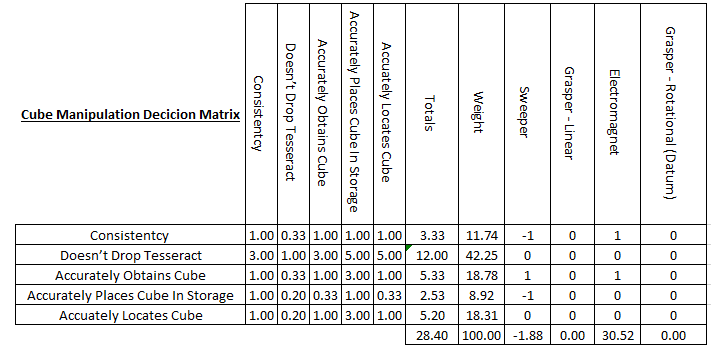


Figure 9 - Cube Manipulation Decision Matrix

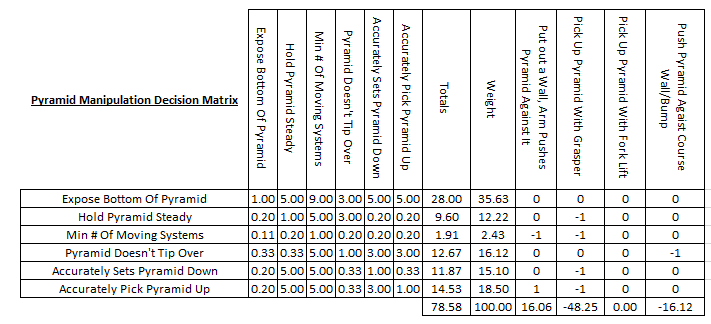


Figure 10 - Pyramid Manipulation Decision Matrix

# Works Cited

Marvel Cinematic Universe Wiki. *Tesseract*. Retrieved from

<http://marvelcinematicuniverse.wikia.com/wiki/Tesseract>

(August 2014). *Hall Effect Sensor*. Retrieved from <http://www.electronicstutorials.ws/

electromagnetism/hall-effect.html>

*What materials reflect infrared light?*Retrieved from*<*https://www.reference.com/science/

materials-reflect-infrared-light-80087ca386b2d887>