KBBQ for Beginners

Joseph Jang, Jasper Lessiohadi, Raymond Ngo

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—Korean Barbeque is delicious cuisine, but can be slightly daunting to people who have never experienced it before. To aid beginners through this, we will be implementing a robotics system that can help KBBQ beginners to properly cook meats. The system will use computer vision and a robotic arm to scan various meats and automatically cook them, leading to food that is cooked well every time.**

***Index Terms*— Computer vision, design, inverse kinematics**

I. Introduction

Korean Barbeque (KBBQ) is a type of meal where people grill their own meats on a grill in the middle of the table. However, those who are new to Korean BBQ could have difficulty properly cooking the different varieties of meats that are served. Each ingredient has a different thickness and required internal temperature, and beginners who have little experience may not know the appropriate amount of time to leave them on the grill. This could lead to burnt food or food poisoning if the dishes are cooked improperly. To avoid this situation, our group proposes a robotics system that can help KBBQ beginners to properly cook meats. The system will use a robotic arm with four degrees of freedom to handle putting the meat on the grill, taking it off the grill, and flipping the pieces to cook both sides. We will also use computer vision to scan each ingredient as we put it on the grill to determine how long a certain dish has to be cooked, mainly based on thickness. When the meat needs to be flipped or taken off the grill, the system will use a robotic arm to do so. Meats will each be placed in four sections of the grill to avoid the possibility of them overlapping. There will also be an overhead camera feed of the grill on a touchscreen connected to the system. Users will be able to tap on individual sections to signal to the arm that they would like meat in that section to be flipped, if they so choose. By using this product, users will eliminate the risk of overcooking or undercooking their foods, resulting in Some possible dishes we will consider for this project are pork belly (sam-gyup-sal), marinated ribs (LA Galbi), thin beef slices, and marinated beef (bulgogi).

The reason we chose these strategies in our solution is because we felt that fully automating the cooking process was the best way to ensure beginners have well-cooked food every time without having to worry about a new, possibly overwhelming experience. The advantage of primarily using thickness, rather than the type of meat, to determine cooking time is the fact that it will be significantly easier to train our algorithm. If we wanted to consistently identify meat type, we would have to feed our algorithm much more data than we have, making this strategy unfeasible. Additionally, since the meats served in KBBQ are generally quite thin, the difference in internal temperature required to be fully cooked between beef and pork will be less of a concern. Furthermore, we decided to use a robotic arm with a ‘hand’ instead of something like a spatula because we felt that it would be more precise. A spatula would also be significantly worse at picking up the raw ingredients, since they would likely be arranged as a pile on a plate, rather than a single piece on a completely flat surface. Overall, we felt that these choices were the best solution to creating a smooth experience for the user.

II. Use-Case Requirements

To consider this a successful project, we have several requirements. The first is that our computer vision algorithm must be able to correctly identify the type of meat being cooked within 5 seconds, 80% of the time. It must also be able to identify the thickness of said meat within 1/16th of an inch of error. Failing to do those two would mean that we are feeding incorrect data to the cooking time algorithm, leading to poorly cooked meat. Additionally, the robotic arm must be able to touch points on the grill within 1/16th of an inch of error. Having precise control of the arm is crucial to the function of the system, since we need to reliably pick up small, thin pieces of meat in a quick and efficient manner. Next, the touchscreen UI must correctly display the camera feed of the grill, and always register touches which signal that the user wants the meat in the chosen section to be flipped or removed from the grill. Finally, the algorithm to determine the cooking time for a given piece of meat must have a failure rate of less than 10%, where failure is either undercooking (internal temperature is too low) or burning the meat. The reason for this requirement is obvious: cooking the meat incorrectly is the whole goal of our project.

The entire system needs to last the length of an average dinner to be effective, which is to say it needs to last anywhere from 20 to 45 minutes. The system needs to simultaneously handle multiple pieces of meat at once to ensure enough food is cooked for an enjoyable dining experience.

III. System Architecture and Principle of Operation

The image on the next page shows the basic setup of our operations. A metal heated grill will exist. On the left side of the grill, the user can place meats they desire to cook on a plate designated as the raw meat plate. On the right side of the grill, cooked meats will be placed on a dish designated as the cooked meats plate. In between the plates and in front of the grill, the robotic arm will be placed so that it can reach any part of the grill with its claw. Not

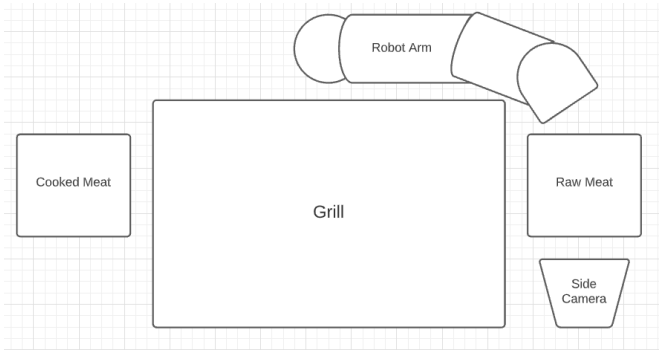


Figure 1: Robotic Arm system top down view.

shown in the image above is the top camera, which will be placed above the grill at a safe distance from the heat so that a bird's eye view of the grill is available. A side camera will be placed at the same level the grill is at. This operation setup will allow us to simplify some of the implementations of the robot's kinematics and allow the cooking of KBBQ to be efficient and safe.

The KBBQ robotic system will have four main subsystems, which are the Robotic Arm, Computer Vision, UI, and Software Controller. In the diagram below, each box represents one or more subsystems. The red and green boxes, which are the 4 Degree-of-Freedom Robotic Arm and Power Distribution wiring, respectively, make up the entire Robotic Arm subsystem. The inverse kinematics algorithm within the Jetson AGX Xavier (orange box) is also part of the Robotic Arm Subsystem. The blue box that contains both cameras and the software subsystem (rounded box) within the Jetson AGX Xavier box that is labeled "CV Algorithm" represents the Computer Vision Subsystem, which we will name the CV Algorithm subsystem. The Software controller box within the orange software block is the final subsystem, which is purely software.

As the key on the top right of the diagram suffest, dotted lines within a colored box represent connections between the various software subsystems and algorithms. A solid line within a colored box represents physical electrical wirings within a subsystem. Blue lines are interconnections between different parts to the Jetson AGX Xavier, which is responsible for the software subsystems. Red Power lines represent physical electrical power lines from the power distribution block to the main subsystems. And finally, black solid lines that go past the dashed box, which represents our robotic system, are used to indicate inputs and outputs into and out of our system, such as vision and motion.

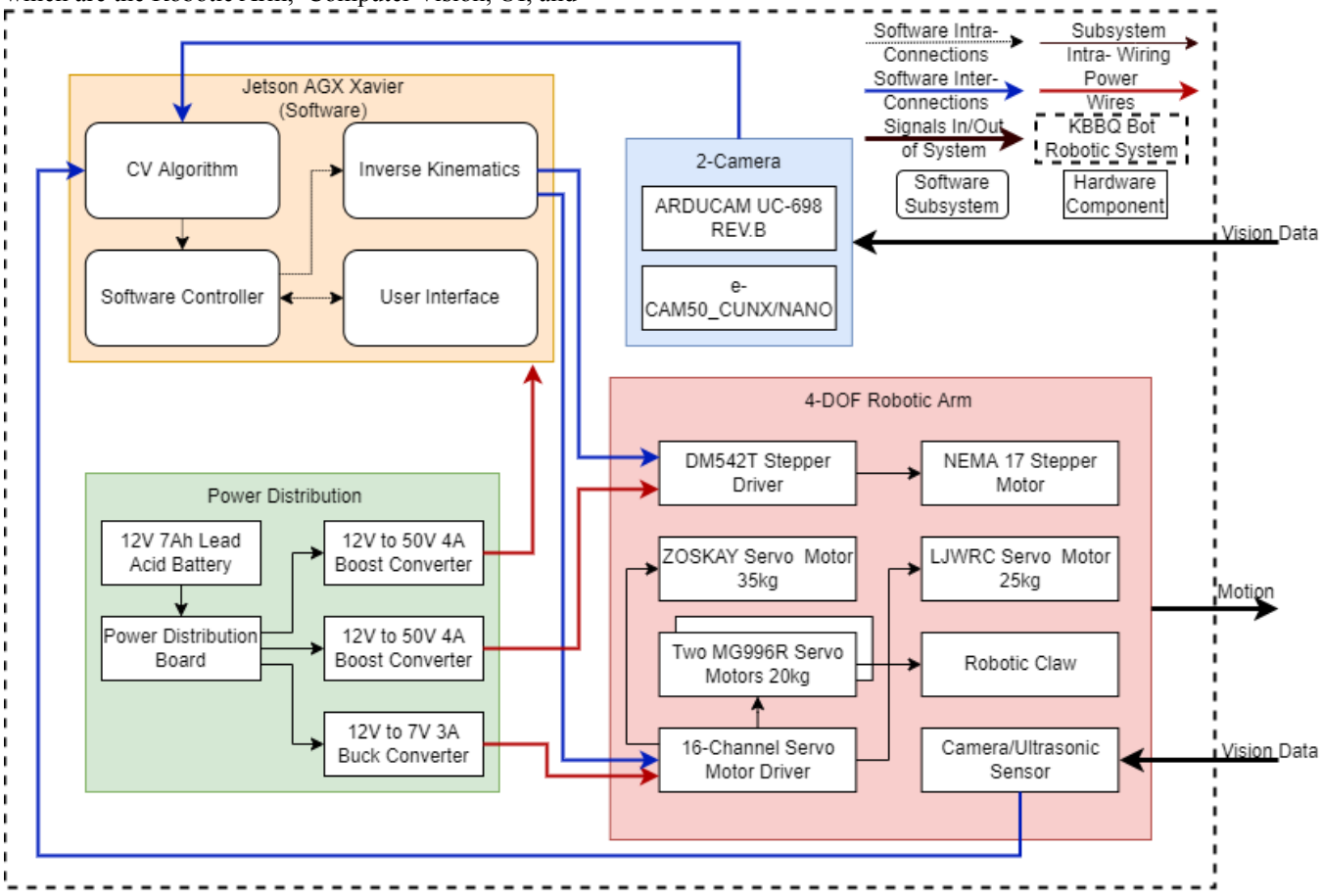


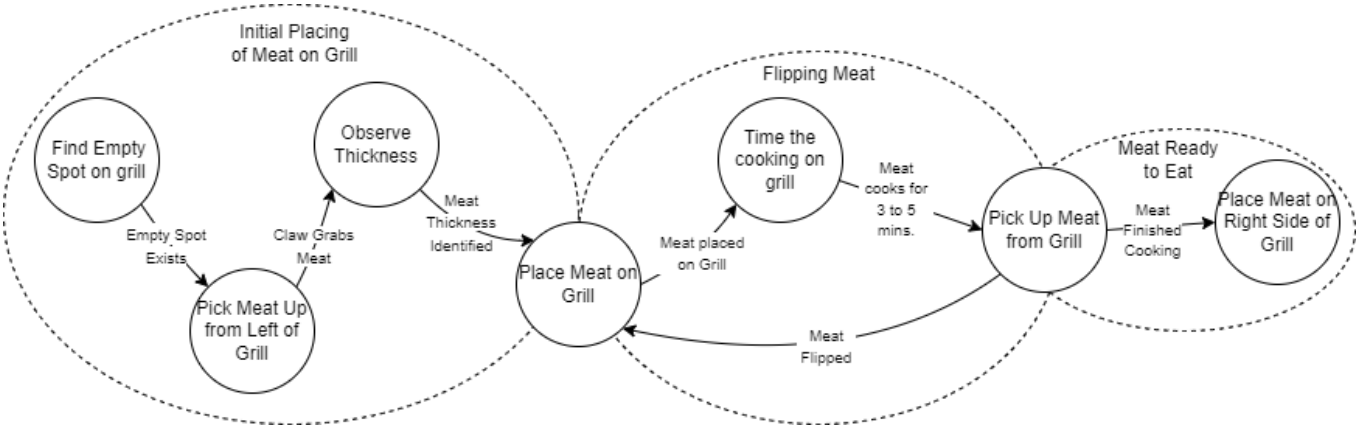
Figure 2: System architecture diagram.

IV. System States

Figure 3 is a system state diagram for cooking a single piece of meat. Once an empty spot on the grill is found, the meat must be picked up from the left of the grill. The robotic arm will hold the meat to the side camera to observe its thickness, which is necessary to compute the cooking time. Next, the meat is placed on the grill at the specific empty location to cook. After time is kept track of for the first side of the meat, and placed on the grill (about 3 to 5 minutes), it is picked up from the grill, flipped, and placed back on the grill. After enough time passes for the second side of the meat, the meat is once picked up again and placed on the dish that is on the right side of the robotic arm and grill. At this point, the system state diagram ends for that single piece of meat.

The second state diagram (figure. 4) handles multiple pieces of meat. It shows prioritization for flipping meats and placing them back on the grill or the cooked meat dish, away from the grill, over placing meats on the grill. Hence, the "Initial Placing of Meat on Grill" states will only occur if an empty spot on the grill exists and uncooked meat exists on the left dish and if no meats are ready to be flipped or pick up from the grill (all meats on the grill are in the "Time the cooking on grill" state in the individual state diagram). These state diagrams will be used to help develop and verify that our robot is working as expected, first for one piece of meat on the grill, and then for multiple pieces of meat.

Figure 3: State diagram of flipping and placing single item.



V. Design Requirements

The most important design requirement is being able to properly cook the meat the user shows the robotic arm system. The first main step in this process is proper identification of the types of meats placed on the grill. We plan on having at minimum 80% classification accuracy in our computer vision systems. Highly trained networks are able to classify between thousands of categories of objects with 99% accuracy, however, as we are providing our own dataset, as well as our lack of relative expertise compared to researchers, we lowered our threshold for accuracy to 80%. Furthermore, this recognition needs to be under 5 seconds, as studies have found out humans are impatient and 75% of people give up within 5 seconds of no loading on a web page, which is a proxy for robotic arm onset movement time we have set.

Another important aspect of getting the proper cooking time besides classification accuracy is meat thickness precision. As some meats for Korean BBQ vary in thickness due to the nature of their cuts, it is important to obtain thickness measurements for certain cuts to obtain an accurate cooking time estimation for the food. As the thinnest cut of meat we plan on measuring is ⅛ of an inch thick, we expect the margin of error to be a maximum of 1/16 inch.

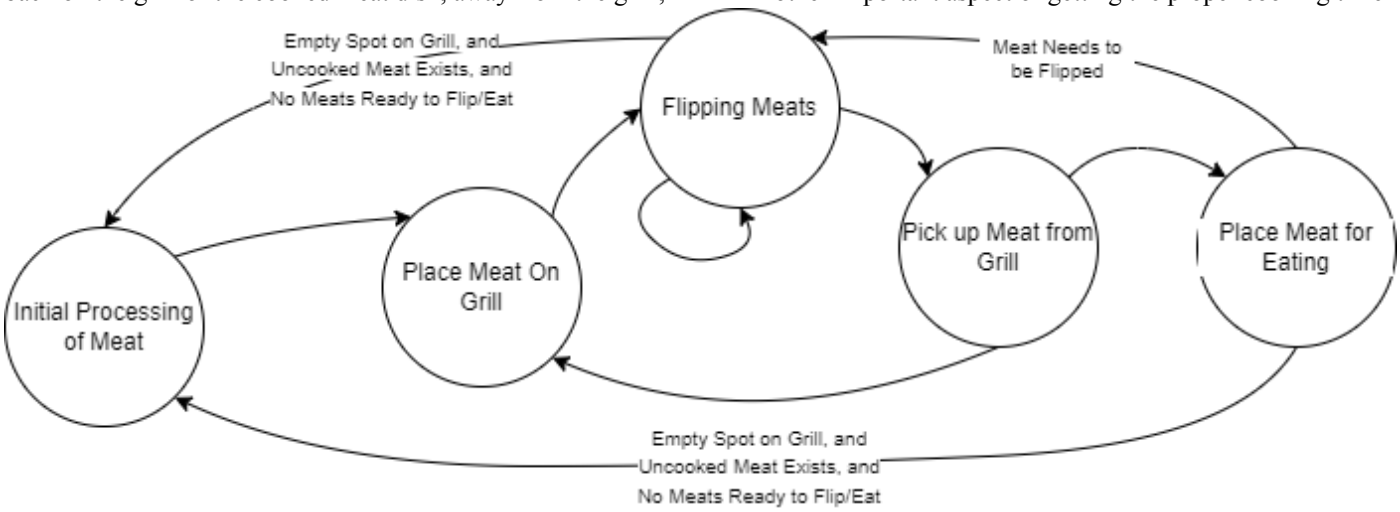


Figure 4. State diagram of flipping and placing multiple pieces of meat.

One important aspect of the entire solution, of course, is cooking. The cooking time is determined as a function of the type of meat as well as the thickness of the meat in certain scenarios. The function is not to rely on any input from the camera to ease project overcomplexity. The cooked meat should reach an internal temperature of at least 170 for beef and 145 for pork, and should not be burnt or undercooked. Our goal is to create an environment to lower the intimidation one faces when encountering Korean BBQ for the first time, so these requirements for internal temperature are necessary to decrease the risk of any potential issues such as food borne illnesses that decreases enjoyment factor.

The robotic arm must be able to reach a certain point with an accuracy of within 1/16 of an inch of a given selected point. It must have a maxim and minimum reach of at least 13" to 1" if the robot arm is placed right next to the grill. The electrical wires and components must be able to withstand grill temperatures of up to 400 F at any given moment or for a period of 1 minute. The robot must be able to complete a single action, such as picking up a piece of meat and placing it on the grill, within a 1 minute time frame. The robot must be able to move at least 12 cm/s. Lastly, the robot must be able to pick up pieces of meat that are at most 2 lbs. The battery power of the arm and the entire robotic system must last for at least 20 minutes to an hour.

VI. Design Trade Studies

A. Processing Unit

The computer vision algorithm running twice a second to detect objects necessitates a strong mobile computing platform. The Raspberry Pi, without a dedicated GPU, is excluded due to its lack of strength compared to a product in the Jetson family of devices. However, the rapidity of the detection, the variety of our computer vision algorithms, and the strict time constraints required for computation mean the Jetson Nano might not contain the computing power necessary for our desired application. The Jetson Xavier, while potentially containing much more power than we need, provides more computing power and gives us breathing room should any operation require more computing power than we expect. Furthermore, the ability to loan devices instead of acquiring it with our funds frees our budget as part of our risk mitigation plan in the case where one of our components fails. As a result, price was not a factor in our comparison.

	Jetson Xavier	Jetson Nano	Raspberry Pi 4
CPU	8 Core Nvidia Carmel Cores (1.37 Ghz)	Quad Core A57 (1.4 Ghz)	Quad Core A53 (1.5Ghz)
GPU	512 core Volta GPU with 64 Tensor Units	128 core Maxwell GPU	None
Memory	16GB (136 GB/s)	4GB(26 GB/s)	2GB
Connectivity	3x USB 3 (2 of them USB-C)	4x USB 3	2x USB 2 2x USB 3

B. Computer Vision Algorithms

For the edge detection algorithm, the Canny edge detector[1] in the OpenCV[2] library showed itself best for our purposes. It accurately detects edges and has manipulable parameters. In addition, the use of built in functions allows us to avoid having to spend time creating edge detection algorithms from scratch using gradients.

For the object detection algorithm, the simpleblobdetector function presented similar benefits for the Canny edge detection in its simplicity compared to its competitors. One alternative we considered was using the classification algorithm to also detect the presence of objects, however that was ruled out due to the need to train for such a scenario. After brief preliminary testing, it would seem some form of image manipulation would need to be made to accommodate such a function. We plan to isolate red elements of the image to improve accuracy in object presence detection.

We needed a way to classify our images, and we had a selection of choices. Selecting a neural network was not among our top choices due to the size of the data set needed. However, many other methods we looked at that involved feature extraction requires fine tuning of parameters, and that is extremely difficult when classifying images that have extremely similar characteristics, such as, for instance, color and shape. As a result, we came to the conclusion that a neural network that learns the features could, despite the tradeoff in dataset size, ease the process in feature selection. Upon researching, we opted to just use an artificial neural network with single dimensional inputs as opposed to more popular and more accurate convolutional neural network based approaches. The reason, even though it would seem to be a tradeoff at first, is because our team has no experience with convolutional neural networks, and given how we are working with multiple different computer vision algorithms, we felt a reduction in complexity by sticking with approaches we are most knowledgeable in is the best course of action.

C. Software Controller Operations

For the cooking mechanism, we settled on a function that takes in the type of meat and its thickness and outputs out cooking time, which is fed to the robotic arm queue. Flipping time occurs halfway into the output cooking time. One key assumption made is the grill maintains temperatures around 500F. This is a key assumption made because KBBQ grills

usually exist around that temperature range. Furthermore, temperature sensors from our research of different options max out their operating temperatures at or below 500F, which makes a grill that exceeds that temperature extremely dangerous for the sensor, which unfortunately rules out most sensors that fit within our desired budget. A computer vision algorithm to detect burning and meat doneness based on color was discussed, however such a solution would take time to train or tune. Most importantly, external meat color does not correlate with meat doneness. Meat can appear burnt but still not have a safe internal temperature.

VII. System Implementation

Our project will be arranged such that the robotic arm will be on one side of the grill, a plate with raw meat will be on its left, and a plate for cooked meats will be on its right side. There will be a camera attached near the end of the arm to help guide it to its destination. A side view camera will be next to the raw meat so that we can use blob detection to see how thick the meat is when the robotic arm picks it up. A third camera will be placed above the grill and its data will be funneled into the user interface. We will also have a movable touch screen that can go wherever the user chooses. All of the processing involved will be done using a Jetson Xavier, which will be more than enough processing power for our needs.

A. Computer Vision Algorithm

The computer vision algorithms we plan on using involve an edge detection algorithm, an object detection algorithm, and an object classification algorithm. The edge detection algorithm uses a bounding rectangle around the dimensions of the meat to determine the pixel width, and by extension, the thickness of the object. Because the robot holds the meat up for the detection, the variance of the meat's distance away from the camera is bounded by the width of the meat where the robotic arm can grab.

The object detection algorithm uses a blob detection algorithm from openCV. Before the activation of the blob detection algorithm, the input image's color space is converted to HSV, then filtered to only highlight red portions of the image. After dilation and floodfill, any object with a red outline (in this case white), would be white, and every other object in the image would be black. This makes the job easier for the object detection algorithm by specifically highlighting red objects, which in this case would be meat.

The classification algorithm would try to find out which type of meat was placed in front of the camera. The classification is composed of a neural network and the dataset is created by ourselves. Images would be flattened and downscaled before being placed into the neural network. One of three classifications would emerge as a result, as shown in the figure.

On the side camera, the object detection algorithm runs twice a second to detect if the user has placed meat in front of the camera. On detection of the meat, the classification algorithm takes an image from the camera and classifies the meat into several categories highlighted in Fig _. On classification, if the meat is classified as short rib or any type of meat where thickness is a necessary factor in cooking, the robotic arm will lift the meat vertically in front of the camera for the edge detection algorithm to detect the thickness of the meat.

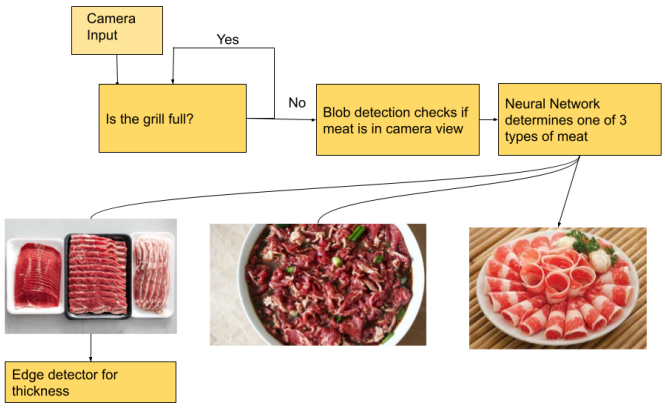


Figure 5 : Computer Vision algorithm flowchart. Note only one classification of meat requires checking for thickness.

B. Robotic Arm

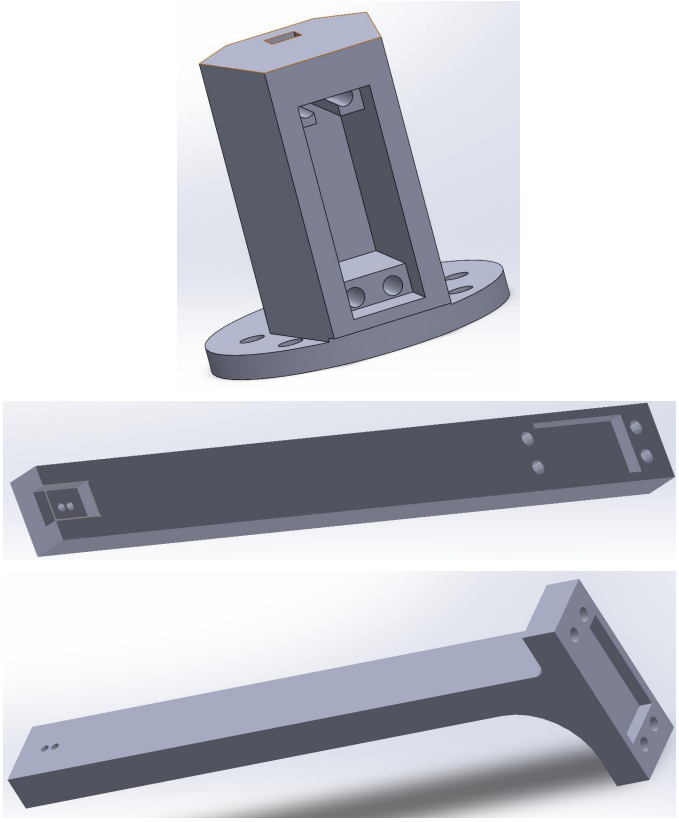


Figure 6: Robotic arm linkage parts (links 1,2,3 in order from top to bottom)

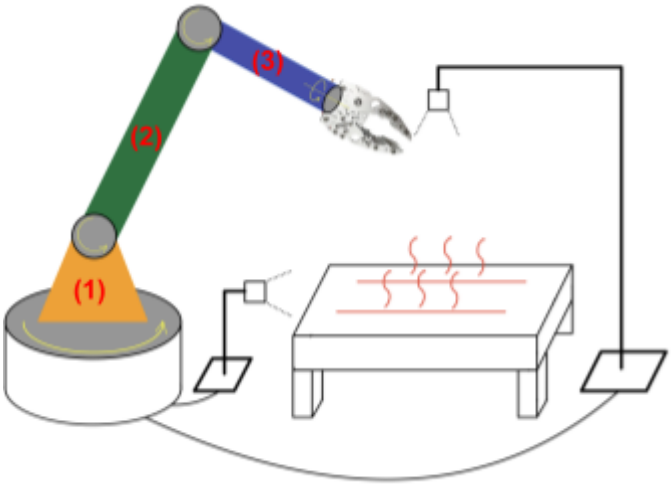


Figure 7: Robotic Arm fully assembled mockup.

Figure 8: Robotic Arm CAD Model

The robotic arm will have 4 Degrees of Freedom. Therefore, it will have 3 links and 4 joints. The first DOF is the base joint. This is the yaw movement, and is capable of freely rotating (360). The robot will have two elbow joints, which will account for the Pith movement of the robotic arm. Each joint will be capable of 270 of movement. Finally, the wrist of the robot, or the joint where the last link 3 is connected to the robotic claw, will be the Roll (4th DOF), and will be able to rotate 360. Link 1, 2, and 3 stand at a height of 3, 11, and 9 inches respectively. The claw is 4 in. long and can open 180 degrees. The CAD models of each link have been created in Solidworks in millimeter format. These links can now be 3D printed in Techspark using a Dremel 3D40 or F170 Printers. We will use an infill of 20% and a precision of 0.01mm to print the 3 parts.

The assembly of the physical robot should not be difficult. The base will be created using pieces of scrap wood from Techspark. We will create a sort of box where the lazy susan bearings, stepper motor, and link 1 will be attached to. Then, the 35kg-cm servo motor will be bolted onto the four holes provided in link 1 using four 8/32 by 2 in. bolts and nuts. These same bolts will be used to secure the 25kg-cm servo motor to link 2, and the MG996R 10kg-cm servo motor to link 3. A 25-teeth aluminum servo horn metal steering arm will connect the ends with the smaller grooves of link 2 and 3 to the 2 strong servo motors. A 25-teeth disc horn will then connect the MG996R servo on link 3 to the claw, which will also be powered by a MG996R servo motor.

For the electrical wiring of our robotic arm, a 12V 7Ah Lead acid battery will be used to power our system. An electrical power terminal block that can handle 30 Amps and 18 AWG wires will be used to power various components of the system. Two boost converters will power the Jetson AGX Xavier (needs 19V, and 65W) and the DM542T stepper driver and motor (needs around 20-50V). A buck converter will then be used to power the I2C servo motor controller, which requires a voltage of around 7 V and supports 16 servo motors (we will only use four of these triple pins). All signal wires from the servo motor controller and the stepper driver will be connected to the 40 pins of the Jetson AGX Xavier. All three cameras will also be connected to this computer. ATC fuses and fuse holders will be used to protect electrical components. A heat sleeve, which is usually used to protect wires in car engines that could reach temperatures as high as 1000F, will be used to cover all servo motors and wires so that no melting or electrical damage can occur. An electrical diagram is provided at the end of this document.

The robotic arm stepper and servo motors can all be controlled using PWM and I2C. For the stepper motor, we will implement a PID controller to accurately control the base of the robot, but for the servo motors we believe it can be controlled using the I2C servo motor controller we will buy. As for the camera that will be mounted on the robot, it will be mainly used for depth perception when the robotic claw must pick, place, or flip a piece of meat. Although we believe we have planned and designed enough to avoid any major critical

failure, if the mechanical or electrical design of our robotic arm fails, we will buy a $100 robotic arm on Amazon.

Inverse Kinematics is when given a point in 3D space and the lengths of each link, the inverse kinematics algorithm can calculate the appropriate angle each joint must be at so that the robotic arm can properly move its end effector to that point in space. With the dimensions of each link and the claw, the robotic arm will be able to have a maximum reach of at most 24 in. and a minimum reach of at least 4 in around itself (think of a circle with a radius of 4 in. and 24 in.). This is plenty of range, because the KBBQ grill will be a circle with a diameter of 13 in. If we place the grill and plates 4 inches in front of the robotic carm, the robot should have no problems with the range of motion possible. To solve this inverse kinematic problem, there are many available resources. We have looked in ROS, Matlab, and Python libraries that have inverse kinematics solvers. Among the best three, which are IK Fast, IKPy, and tinyik, we will use the ROS Kinetic tutorials to implement IKFast kinematics solver in our own robotic system. There are quite a few tutorials and resources available for using IKFast effectively for our unique application.

C. *User Interface*

Our UI will involve a touch screen which displays an overhead camera feed of the grill, along with some other information. The camera feed will be split into four sections, and the user will be able to tap on any section to manually signal to the robot to either flip or remove the meat on that section, depending on which action needs to be done. Information regarding the remaining amount of cooking time will also be displayed so that the user knows how long they will have to wait until their food is ready.

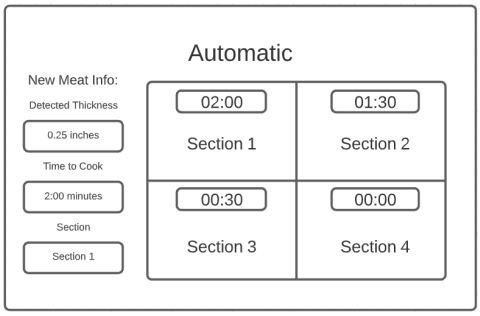


Figure 9: Layout of UI design. The mentioned camera feed is on the right, while information on the newest meat to be put on the grill is on the left.

D. *Software Controller*

This subsystem will be the main subsystem to integrate all theo other subsystems - CV, Robotic Arm, and UI. It will take in information from the CV Algorithm, and display CV data to the UI. It will need to keep track of multiple meats on the grill and display their cooking times. It will need to calculate information for each meat and make decisions for each meat. The software controller must also be able to make decisions, such as providing a desired action and position to the inverse kinematics algorithm. The controller will prioritize flipping and getting meats off the grill at the appropriate time over putting meats on the grill. It will use two queue implementations. One queue holds events related to robotic arm actions meat on the grill, while the other holds events related to robotic arm actions related to meat about to be added to the grill. The queues are separate to ensure actions relating to meat already on the grill take priority over meat yet to be cooked. The controller will also understand that the dish of uncooked meats that need to be grilled will be at the left of the robotic arm and grill, while cooked meats will be placed on a dish at the right of the robotic arm and grill. In essence, the software controller is truly the brains of our entire robotic subsystem. It will be where all the other subsystems integrate with one another and must also work with one another.

VIII. Test, Verification and Validation

Creating a successful system requires a lot of testing. Due to the many different subsystems, the plan is to test each subsystem separately, with the expectation that the different subsystems working individually will make the integration process smoother.

A. *Tests on Computer Vision Subsystem*

To test the classification strength of the object, as well as its ability to predict in real time, meat would be placed in front of the camera, and the network's prediction class and its prediction time would be outputted. Using this method of testing, a desired accuracy of 80% and a desired processing time of under 5 seconds would need to be obtained.

To test the accuracy of the measuring system, a section of a table will be marked, a person (or a hanging clip) will hold the meat vertically in front of the camera. The distance the meat is from the camera does not matter so long as the clip or hand remain the same distance from the camera between tests. The predicted measurement result is compared against the actual measured thickness.

B. *Tests on cooking time*

The testing of this portion of the software controller would happen independently of the testing of the computer vision systems and the testing of the robotic arm system. In this test, the cooking time function would be called with a specific meat type and thickness as the manual input, as opposed to the final system where the computer vision algorithm provides the function inputs. We would then cook the meat on the grill in accordance with the time output from the function, flipping at the exact halfway point. After cooking, a temperature probe would be inserted into the meat, and a visual inspection of the meat would be performed to determine the internal temperature and the amount the meat was burnt. Tuning of function parameters would happen at the conclusion of testing if the cooking result is not desired.

C. Tests on Robotic Arm

A simple test we will conduct on the finished robotic arm is we will define a point in space or on the grill (without any heat) and see if the robot can move the tip of the claw to that point on its own. This will be a test of the movement accuracy of our system. The robot must use computer vision feedback and precise motor control to correctly touch the point. It should only have an movement error of 1/16 inch. Next, we will test the grasping and flipping capability of our robotic arm's claw, by placing different sized and weighted objects on the non-heated grill. This will not only test the robot, but also the vision algorithm and software controller subsystems. This is a test to see if the robot can properly pick stuff up from a plate, pick meats from the grill and flip, and if the robot can properly place meats back onto a plate. A meat should not drop or be placed out of bounds from the plate or grill during these tests. After these two smaller tests, we will start real testing of the robotic arm using a heated grill with real meats. For safety purposes, we will likely use a heated electrical stove instead of a flame to heat the grill. We will also have to buy cuts of meat for this testing phase.

Other smaller tests we will conduct on the robot will be a heat test of whether the heat sleeve can adequately protect our wires and electronics from melting. We will also test each electrical component to see if the proper amount of power is flowing into the Jetson AGX Xavier and motors, and if the proper amount of power is flowing out of components like the voltage converters. Another small test to consider are battery tests to see if the proper amount of current and voltage is available to the entire robotic system. And finally, the robot's strength will also be tested. No piece of meat will be greater than 2 lbs heavy, so we expect that our robot will be able to easily pick and and move the meats and other objects that are around 5 lbs.

IX. Project Management

While all of us are managing our respective subsystems in a timely manner, Joseph Jang has been responsible for the parts list and schedule updates.. Raymond Ngo has been in charge of making sure deadlines for presentation, reports, and status updates are met. Jasper Lessiohadi has been responsible for making sure each of us understands our own responsibilities concerning each phase of the project and risk mitigation.

A. Schedule

At the end of the document is an updated version of our schedule. Our design phase is 100% complete. Now, we are in the development phase, and we plan to start development of the robot as soon as possible. We have also added slack time in our schedule. The first is during spring break so that we could finish any miscellaneous design questions and start development early. The second slack time is a week between system integration and the end of the development phase, which we will use to wrap up subsystem development and start integration early. We have also reduced the time for subsystem development for some parts, such as the UI and CV algorithm, while extending the dates for system integration and the inverse kinematics algorithm development. Please take a look at our schedule at the end of the document.

B. Team Member Responsibilities

Our team has a good spread of abilities that cover every aspect of the design of this project. Joseph is in charge of the robot arm, meaning he is building it and designing the IK for it so that we can move the meat swiftly and precisely. Jasper is handling the UI and making sure it will interact well with the other subsystems. Raymond is handling the CV, making sure that blob detection can consistently find the thickness of the meat being passed in front of the camera. Finally, all of us will be working on the software controller and, of course, helping each other out where we can.

C. Bill of Materials and Budget

We have first created a parts list for the electrical and mechanical needs of the overall system, but specifically for the robotic arm. IF we borrow the parts listed and use parts that our group member, Joseph Jang, already has, we were able to save around $200. Therefore, that brings us to an estimated parts cost of around $213.22 and will all be ordered from Amazon. We estimate that 3D printing parts will cost us around $50 to $100 dollars. Even after ordering parts and 3D printing, that leaves us with around $300, which we could use as an emergency fund.

D. Risk Mitigation Plans

There is an obvious risk involved with this project, which is the heat of the grill. This high temperature could possibly melt the plastic that we are using for the robotic arm. To mitigate this risk, we plan to only have the arm go over the grill when it needs to interact with the meat there, and have its resting position be off to the side. This will make the robotic arm less exposed to the heat and less likely to be damaged by it. However, if the arm we are building doesn't work, we will buy one that we have already found on Amazon.com.

Another risk is serving meat that hasn't been fully cooked. We will manage this by exhaustively testing our timing algorithm to make sure that it only has a failure rate of less than 10%, as was mentioned in the use-case requirements. Manually probing the meat with a food thermometer will allow us to determine whether it has been fully cooked. So, through a lot of testing, we will be able to determine a suitable algorithm for finding the amount of time to cook a given ingredient.

X. Related Work

A previous capstone project called Hot Pot Bot was similar

to what we are trying to accomplish, so we have looked to that as a bit of inspiration for some aspects of our project.

XI. Summary

Overall, our project aims to assist those who would like to try delicious KBBQ, but may get overwhelmed by such a new and different experience. To do this, we have designed a system to automate the cooking process so that users can enjoy the company of those they are eating with, rather than stress about how long to leave ingredients on the grill. A robot arm will handle flipping the meat, along with moving it on and off the grill. Using CV, we will be able to reliably determine the correct amount of time to cook the ingredients presented. In the case that the time is calculated incorrectly, though, users will have the option to use the touchscreen UI with a camera feed of the grill to manually tell the system to flip or remove a specific piece of meat. In this way, we can give users an efficient and streamlined way to enjoy KBBQ.

In the process of implementing our design, though, we expect to have some challenges. Having IK for the robot arm and blob detection for our CV that are precise enough to fulfill our design requirements will not be easy, however we are confident that we will be able to find ways to make them work at a satisfactory level. In terms of the cooking time algorithm, we will need to perform a lot of time consuming testing to make sure that we do not serve undercooked or burnt meat. Despite these challenges, though, we know we can produce an effective system that will help anyone who would like a more stress-free experience with KBBQ.

Glossary of Acronyms

CV - Computer Vision
DOF - Degrees of Freedom
IK - Inverse Kinematics
KBBQ - Korean Barbeque
UI - User Interface

References

[1] J. Canny, "A Computational Approach to Edge Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[2] *OpenCV.* url: https://opencv.org/.

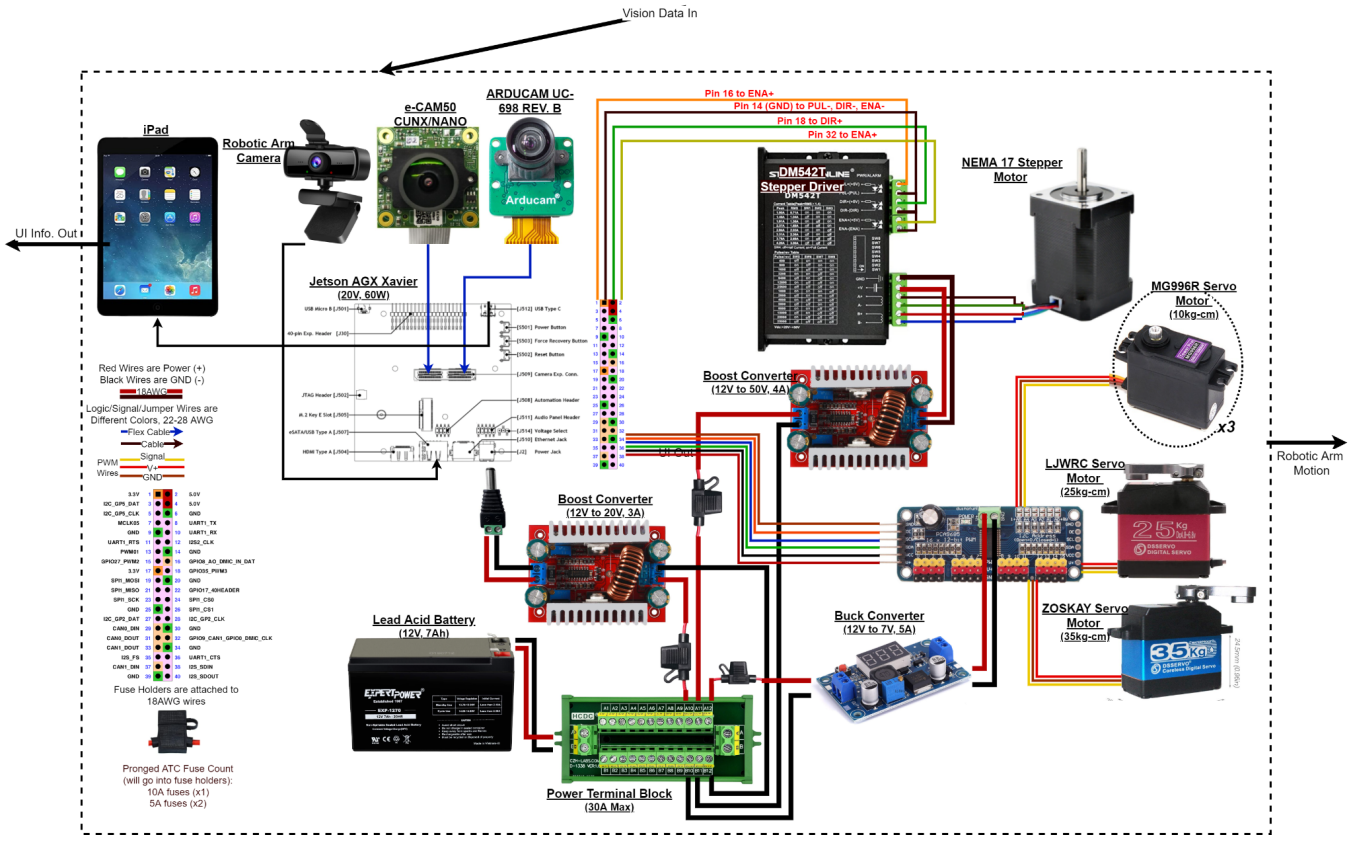


Figure 10: Electrical Diagram for entire System

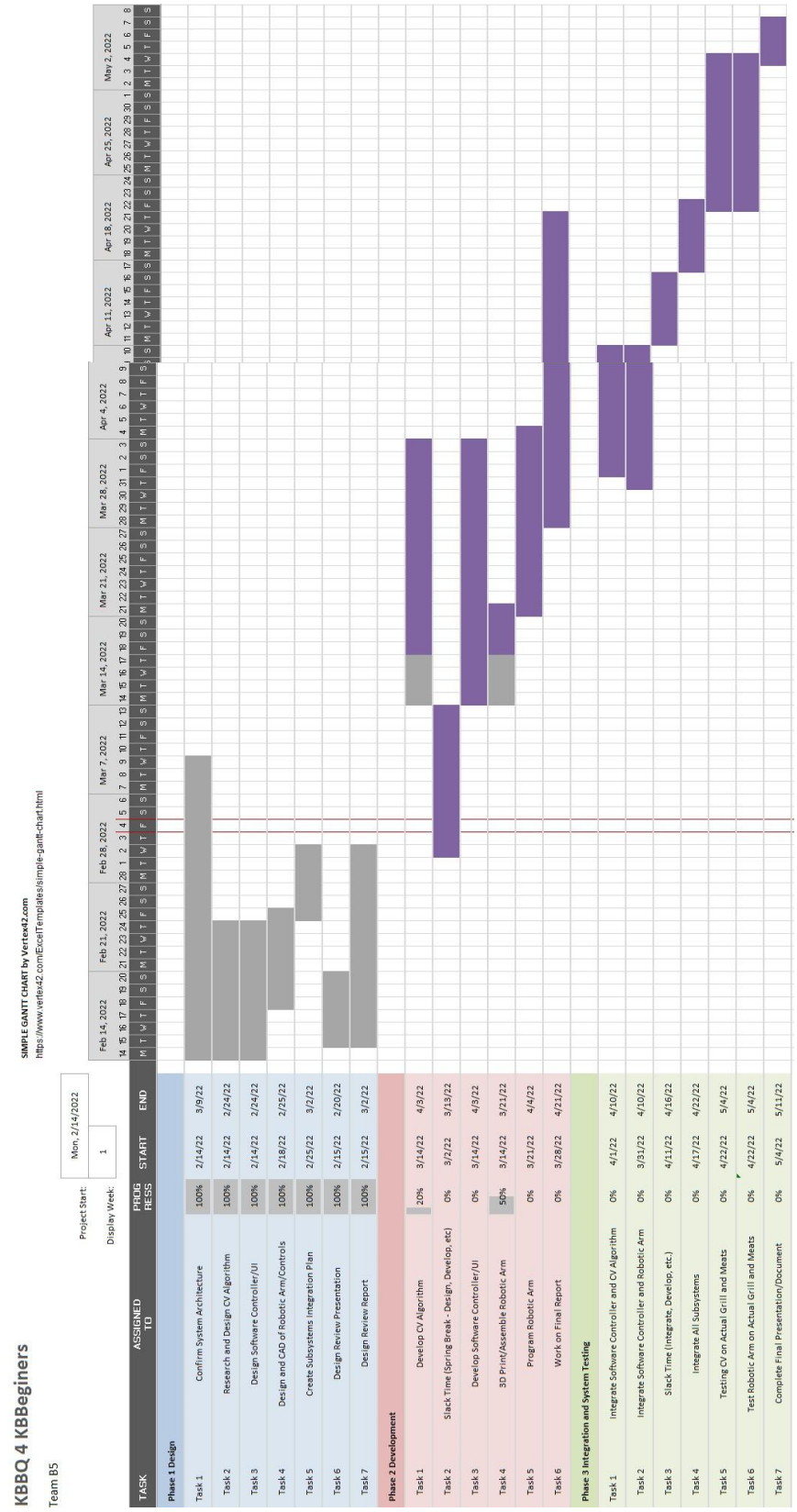


Figure 11: Schedule

Part Name	Quantity	Status	Cost
Stepper Driver DM542T	1	Borrowed	-
NEMA 17 Stepper Motor	1	Borrowed	-
Lazy Susan Bearing	1	Owned	-
Wood, Nuts, Bolts	Several	Borrowed	-
Stepper Motor Mounting Connector	1	Owned	-
Servo Motor 35kg	1	Buy	$32.99
Servo Motor 25kg	1	Buy	$18.99
25T Disc Horns	1 (pack of 5)	Buy	$6.99
Robotic Claw	1	Buy	$27.99
KBBQ Grill	1	Buy	$29.98
Jetson AGX Xavier	1	Borrowed	-
Camera 1 e-CAM50_CUNX/ NANO	1	Borrowed	-
Camera 2 ARDUCAM UC-698 REV. B	1	Borrowed	-
Power Terminal Block	1	Owned	-
12V 7Ah LEad Acid Battery	1	Buy	$24.14
Boost Converter	2	Buy	$14.29
Buck Converter	1	Buy	$15.88
Heat Sleeve	1	Buy	$10.99
3D Prints	***Cost: ~$50-$100***	***Total***	***~$213.22***