Team D1: YoServe

Authors: Isabel Murdock, Kashish Garg, and Matteo Longo: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— Our project is to design and build a robot that can autonomously navigate a room to serve appetizers to guests. While there are currently robots that autonomously clean floors and cut grass, we aim to bring robots to the cocktail hour scene. Utilizing thermal image processing and object detection, our YoServe Appetizer Bot will make serving guests as easy as loading up a tray and pressing go.**

***Index Terms*—** Appetizers, Human Detection, Motors, Object Detection, Robot, Thermal Camera, Ultrasonic Sensors

I. Introduction

Have you ever been at an event or cocktail hour hungry for some delicious appetizers yet engaged in a conversation without any opportunity for escape? Have you found yourself wishing the food might just come right to you? Well our project will leave your mingle time undisturbed and your stomach satisfied! Our project is to design and build a robot that can autonomously navigate a room to serve appetizers to guests. The domain of function is an open, smooth-floored room without furniture but full of people, who may be moving around themselves. We aim for it to operate at a human walking pace and safely approach guests. It will detect and stop operation when its tray is empty.

The food industry currently employs a variety of robotic technology. However, even in smaller restaurants and food locations, these robots are typically stationary and perform one repetitive task. Yet similar to our YoServe Bot, some companies have started developing mobile food service robots, such as PepsiCo's snackbot. Our robot will differ from theirs in that we will be serving food on an open tray that will increase accessibility for our guests. We will also be working in an indoor environment, which will require less reliance on outdoor paths and roads. Instead we will focus on finding heat sources within the room in order to directly approach humans. Finally, while other snack delivering robot projects have been documented, they are, in general, meant to carry small treats and designed for personal usage. Our robot will be large enough to support a number of appetizers and serve a variety of guests.

II. Design Requirements

In terms of mobility, the robot will move at a rate of 2 mph, average human walking speed. It will also be stable enough that food carried on its tray will not roll/fall off upon stopping or starting. We will test this by conducting a "room test". During this test, the robot will drive in an empty room and carry a completely full tray of food, accelerate to 2 mph, and then come to a full stop. We will then evaluate if any food has fallen off the robot.

The robot will be able to operate for an hour in its party environment. While power usage can vary depending on room situation, we can calculate a worst-case scenario where the robot is constantly moving and drawing current through the motors. Later calculations can be done by averaging idle current and moving current based on the percent time in each state.

For human detection, we require the thermal camera and image processing algorithm to detect people within 8 meters of the robot. We want to be able to detect just one guest all the way through a group of guests clustered together. We plan on testing this by positioning 1 person to the left, right, and center of the thermal camera's frame, at approximately 8 meters away and evaluating if the robot is able to detect the person. Then, we will repeat the test with a group of humans in the same positions.

For safety, we want the robot to be able to detect when humans are within 5 feet of the front of the robot and then stop by the time they are 1 foot away from the robot. To test this, we will conduct a stop test. This test will be conducted two different ways. In the first, a person will walk towards the front of the robot. We will measure both when it detects the person and when it finally stops. In the second way, we will put a person, within a 5 feet radius, to the side of the robot as it turns. The robot should be able to detect the human as it turns and make the decision to not travel in that direction. In addition to these safety tests, we will also have an emergency button on the back of the robot, which will immediately stop the motors. In this case, the stability requirement of not having food fall off the tray will not be enforced. Also, we will have the robot play music or make some sort of noise, so that its presence will be noticed by guests.

The robot shall detect when it has run out of food. We will complete this test by filling up the tray completely, taking all the items off, and then refilling it. The robot should stop moving when all of the food is removed and resume moving when the tray is refilled.

In terms of sensor and image processing, we want the time to find if an image contains a heat signature worth following to be under 100 milliseconds. We also want our software algorithm to know within 10 milliseconds if an object is detected.

III. Architecture and/or Principle of Operation

Our YoServe robot is designed to meet this use in several ways. It utilizes a Raspberry Pi for sensor data processing, and Arduino for motor control, and has a wooden frame suited for navigation and tight turns. As shown in figure 1, the Raspberry Pi takes in input from three different types of sensors: the ultrasonic sensors, the thermal camera, and the load sensor in order to determine the motion required from the robot. It then passes the driving instructions to the Arduino, which uses a motor shield to control the motors and navigate as per the instructions provided.

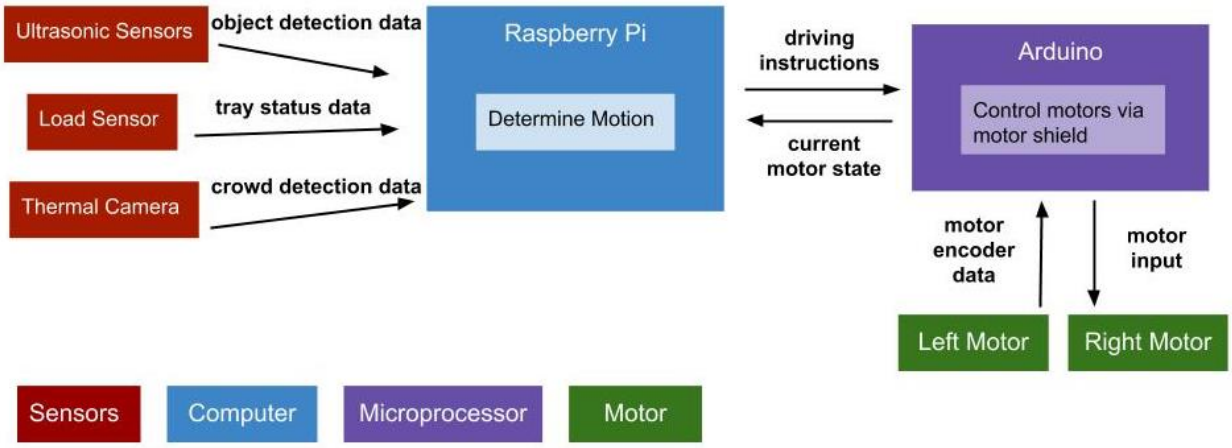

Figure 1: Overall system block diagram

A. Physical Structure

The overall shape of the robot, as shown in Figures 2 and 3 on the next page, is tall and cylindrical. The top level is approximately 38 inches high to ease the retrieval of food by standing patrons. The round chassis allows the robot to turn without changing its profile, which means turning is guaranteed to cause no collisions and robot will not get stuck in tight corners. The two wheels are centered on one axis, while two tennis balls provide orthogonal support and can slide as the robot moves across hard floor. The choice of tennis balls provides additional stability to the robot as the rubber will absorb some of the oscillation in speeding up or slowing down. The frame is made up of wood to ease in mechanical construction and keep the overall weight of the robot down, lessening the work done by the motors and minimizing momentum in the rare event of a collision. The overall balanced structure of the robot makes it more stable and less likely to fall in a specific direction. Additionally, the battery and hardware are stored on the lower circle of the robot. Since the battery is the heaviest part of the robot, placing it at the base of the robot helps to reduce any oscillation of the robot. We also oriented the battery so that its weight is balanced between the two wheels and the back tennis ball in order to reduce static friction upon accelerating.

B. Motion Control

In order to further support robot stability, we first implementing a trapezoidal velocity profile with a low constant of acceleration. This was because it would cause less oscillation of the robot upon stopping and starting. Upon testing, we discovered that for driving forward, we needed a higher initial speed in order to overcome the static friction. Once the robot started moving though, we could back off to a slower speed. Through these tests we determined the speed necessary to reliably get the robot moving and set that as the initial speed and then backed down to our desired forward driving. For stopping, however, we did implement an incremental, linear slope for decreasing the speed of the robot. This did help with robot stability and minimize the oscillation of the robot. This, in combination with the physical structure, helped keep the robot stable and prevent food from rolling/falling off of the tray.

C. Sensors Signal Processing

A Raspberry Pi is used to process sensor input from our three different sensors: one thermal camera, three ultrasonic sensors, and one load sensor. First, a load sensor is used to detect whether the food tray is empty. If so, the robot stops moving, and waits for the tray to be refilled. Then, we use the thermal sensor to get a thermal image of the room and detect any warm bodies. The raspberry pi sends turning instructions to the Arduino until the camera shows that the robot is centered towards a warm body. Then the ultrasonic sensors are used to determine the distance to the object, and driving instructions are passed on to the Arduino.

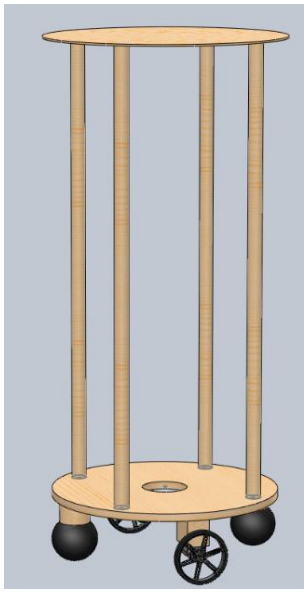

Figure 2: CAD for the robot structure

Figure 3: Final design of robot

IV. Design Trade Studies

A. *Crowd Detection*

For the crowd detection part of the project, we considered two general approaches: training a neural network using input video feed from a camera, and using OpenCV color thresholds to detect humans from thermal camera imagery. As discussed in the design report, on a Raspberry Pi 3 B+ running at 1.47 GHz, the first approach would be too slow for our robot to meet any system requirements. So we chose to implement the second approach, by looking for pixels within a fixed temperature threshold that we calibrated based on the environment. The drawback for this approach was the limitation of a thermal camera in identifying humans. For example, with a thermal sensor, any heating in the walls could also be detected by our algorithm as a human. However, this is not particularly troublesome for our application area, because on detecting a wall as a human, our robot would simply stop for a little bit, and then turn away and move on to another heat-emitting-object. This case was validated during our testing, when we were in a room that had heaters near the walls. The robot detect the warm "body", but successfully stopped away from the heater, and then turned away after some time. We also performed tests with having one person/group stand at different locations in the camera frame (at the very edge, about one-third of the way in, or completely centered) to ensure that this module detects people as expected. Table 1 to the right shows some of the timing metrics for this implementation. These were calculated by using python's built-in time() functionality to get the exact time metrics.

Hardware Type	Module tested	Expected Time (ms)	Actual Time (ms)
2.5 GHz Macbook Pro	Original (leftmost)	10	15.7
	Final (is_centered)	n/a	9.6
Raspberry Pi 3 B+	Original (leftmost)	100	1226
	Final (is_centered)	n/a	881

Table 1: Timing metrics for the crowd detection module

The original design involved returning the coordinates of the leftmost warm body in thermal image. However, because the thermal camera had a narrower range than we expected, and the robot needed some time to stop turning. So, we changed our final approach to just use the image to determine if the robot was almost centered towards a warm body, and then turn in increments as required. Of course, this module reduced the amount of computation required, and therefore, resulted in some speed-up.

B. *Sensors for Object Detection*

As discussed in our design report, we considered two different sensors: IR sensors, and ultrasonic sensors. We decided to go with ultrasonic sensors, because IR sensors do not work well in dark environments, and our application area is of a cocktail hour reception, which is usually dimly lighted. However, since ultrasonic sensors work by reflecting sound waves, we faced some difficulty in detecting people wearing soft clothing. We tested our object detection module by having people and objects stationed 10 cm to 300 cm distances away, for each of the three ultrasonic sensors. We also tested the side ultrasonic sensors by testing the case where people could be standing diagonally from the robot, and the robot could potentially hit them, if it tried to go forward. By calibrating our thresholds to be lower than the thresholds for the center sensor, we passed these tests. Overall, we expected this entire module to be take about 10 ms to return the object distance. Unfortunately, it took much longer than that, because we had setup the pins for communication with the raspi for each of the three different sensors. To improve our performance, we decided to only check the center sensor first, and then if there is no obstacle up to 120 cm in front of it, we would check the other two sensors. This significantly reduced our time for the case when obstacles are far away. However, we would still need to check all three sensors in order to determine if the robot needs to stop.

Module	Time Taken (s)
get_dist_all	1.54
get_dist_mid	0.54

Table 2: Timing metrics for the object detection module

C. Motors

We originally considered selecting CIM motors for their power and wide selection of gearboxes. However, upon further calculations and a lower weight estimate for the robot, we ultimately decided to purchase 12V Pololu motors. There were a number of reasons why the Pololu motors were more appealing. First of all, they were simpler because they came with encoders and gearboxes already integrated with the motors. This also meant that the Pololu motors were cheaper than the original CIM motors/gearboxes. We saved roughly $70 per motor by not having to purchase separate gearboxes. In addition, the Pololu motors can be controlled by a single motor shield that works directly with an Arduino. The library is straightforward with basic example code provided from the manufacturer. This is much simpler than the process that the CIM motors would have required. While the weight of the battery may cause the motors to not be strong enough to drive the whole robot, we can cheaply acquire a lighter/less powerful battery that should still be able to power the Pololu motors since they require far less power than the CIM ones. The only downside to selecting the Pololu motors is that they will not be able to overcome as many environmental factors, such as bumps and ramps, since they are not as strong as the CIM motors. Yet, due to all of their benefits and the scope of our project, the Pololu motors were ultimately the better choice.

D. Power

The 18 Ah battery selected for our design was initially to support stronger motors. However, since the motors were switched for much cheaper counterparts, the battery can supply power to the system for much longer than anticipated. Jumping from about 30 min of movement to over 1.5 hrs, we can easily sustain operation for the duration desired in our scope. A smaller battery, a 11.1V 5Ah Gens ace LiPo battery, was found to replace our original one. The new battery supplies 5Ah, which still allows for our desired duration, but also lowers the total weight of the robot significantly. With a lower weight, the motors don't have to work as hard, benefitting mobility. We also used a separate battery, a 3.7V 3800mAh battery pack, to power the Raspberry Pi so that it could be powered independently from the motors.

In testing, we found some non-ideal factors caused runtime to be slightly lower than the calculated 5 hrs (in worst case of constant movement). As the battery depleted, the voltage would drop, causing the motors to become slightly weaker. This is only noticeable after about an hour of performance, when the motors begin to slow when going over uneven parts of the floor. Even still, the robot can easily have its main battery replaced along with food, or be recharged during downtime

.

V. System Description

A. Software Design

We use a Raspberry Pi 3 Model B+ to determine robot motion using input from the three different types of sensors. This particular microprocessor was chosen, because the Raspberry Pi 3 Model B series works well with OpenCV, and it is one of the fastest options running at 1.47 GHz. Fast processor speed is crucial for our project given our time requirements for crowd detection. Looking closely at Figure 4, first, a HX711 load sensor is used to detect whether the food tray has a weight higher than our calibrated threshold. If not, we assume the tray is empty and the robot would wait and do nothing until it is refilled again. Once we determine that the robot is loaded with food, we use an Adafruit AMG8833 8x8 Thermal Camera Sensor to capture the thermal image, and pass it in to our thermal image processing module.

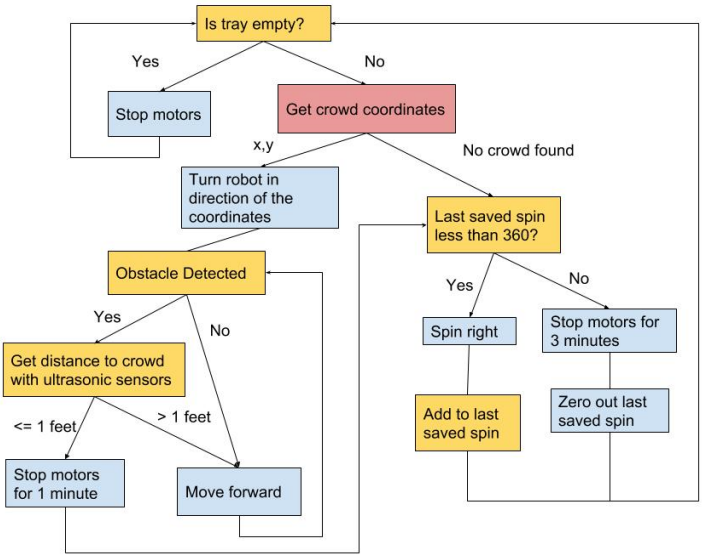

Figure 4.1: Software System Flow Chart with blue representing the robot control module, red representing the crowd detection module, and yellow representing the object detection module

The 8x8 image produced by the thermal camera is very difficult to process. So, the thermal image processing module first applies bicubic interpolation to convert the small image into a 1024 pixel (32x32) image. This smooths the input image and makes it easier to detect crowds using OpenCV. First, we denoise the input frame by applying a Gaussian blur. Then, we convert the RGB (red-green-blue) pixel values into HSV (hue-saturation-value), and apply a predetermined threshold such that, the result is a black and white image in which all pixels within the threshold are white, and the rest are black. Then the

erode function is applied to the black and white image in order combine the white areas that are very close to each other. Once this is done, we find the enclosed contour for each white area. If no contour is present i.e. the image has no pixels that fall within our threshold, then it is determined that no crowd is present, and the human_centered function returns false. Note that because our thermal camera has a very narrow input range, we do not have to worry about two groups of people being seen in the same thermal image frame. Therefore, we can just get the centroid of the contour with the largest area. If the x-value of this centroid is less than 20 pixels into the frame from either the left or the right, the robot is not centered to the robot, and again a false value is returned. Otherwise, the function returns true. Figure 4.2 shows a flow chart summarizing this module.

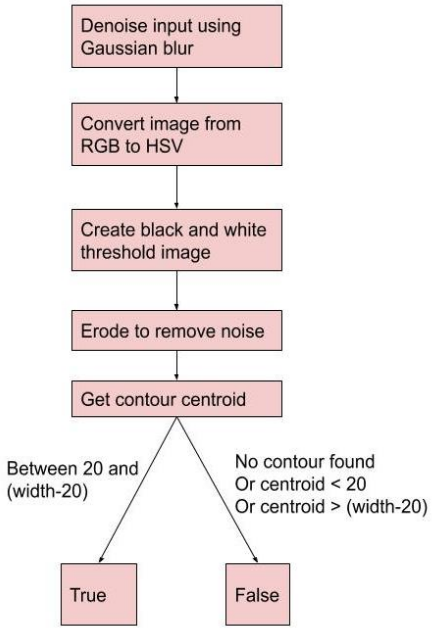

Figure 4.2: A flow chart zooming into the crowd detection module

If the crowd detection module's centered function returns false, the raspberry pi sends a turn signal to the arduino, and then takes a picture again after it has turned by an increment. If the robot is centered, we then process the three HC-SR04 Ultrasonic Sensors to detect the distance from that crowd, and any other obstacles if applicable. These sensors were used because they have a good range for our requirements (i.e. greater than about 8 feet), and are pretty standard-use when working with Arduino or Raspberry Pi. Unfortunately, these sensors only have a range of 15 degrees, so we have to process data from all three of them in order to be sure that the robot would not run into obstacles. First, we get the distance to an obstacle from the ultrasonic sensor located at the center of the robot. If that distance is greater than a fixed threshold (currently 120 cm), we check the side sensors (with a threshold of 50 cm) to be sure that the robot can move forward without hitting an object. If the crowd is out of range of the ultrasonic sensors, or the crowd is nearby but further than 120 cm away, the robot control module would be signaled to move forward. Once the ultrasonic sensors detect a distance less than 120 cm, the robot control module is signaled to stop the motors. As the robot slows down, it ends up stopping about 1 foot away from people. Currently we have set a stop time value of 5 seconds to give people enough time to pick up their food. Once the 5 seconds time is up, the robot would spin again to find the next warm body, and perform all of the aforementioned tasks again.

As an additional feature, the robot also makes auditory announcements when completing certain actions. This is controlled in the software, which plays specific announcements before the robot begins spinning or driving forward in order to alert any guests close to the robot. It also invites guests to take food when it stops close to it. These announcements help to make humans near the robot aware of its presence and helped with debugging.

.

B. Motor Control and Communication Design

The two motors are controlled by an Arduino via a Pololu Dual VNH5019 Motor Shield. This motor shield is specifically designed to interface with the Pololu motors and comes with an easy to use Arduino library for controlling the motors. In order to facilitate the robot's motion, we wrote functions to handle each of the primary driving instructions: forward, turn, and stop.

The Arduino communicates through a serial connection to the raspberry pi. Upon powering up, the Arduino sends a serial message to the raspberry pi indicating that it is ready to begin receiving instructions and waits for a response from the raspberry pi. Once the pi ensures its systems are ready, it waits for and then receives the initial message from the Arduino. At which point, it begins the process outlined in the previous section and decides which driving instruction should be executed. Once the instruction is determined, the raspberry pi writes the instruction to the serial line and waits a fixed amount of time for the instruction to be executed. Meanwhile, the Arduino reads the message, updates its driving state, and executes the specified command.

For forward, the Arduino sets the motors speed to a defined maximum speed. This speed was determined through extensive testing and is the minimum speed required to reliably overcome initial friction and get the robot moving forward. After a brief time at this speed, the robot slowly backs down to a slightly smaller speed. Driving at this speed gives the robot more time to sense obstacles and stop before hitting them. For this state, the Arduino checks for incoming commands from the raspberry pi after each change in speed in order to catch all stop commands as quickly as possible.

For stopping, the Arduino incrementally decreases its operating speed until the robot is no longer moving. This

happens without any checking for incoming commands from the raspberry pi in order to make sure the robot stops as quickly as possible to prevent hitting people or obstacles. We decrease the speed incrementally in order to minimize the oscillation of the robot upon deceleration.

For turning, we set the motors to a defined speed for a fixed amount of time and then set them to zero. Due to the construction of the robot and the placement of the tray on the load cell, we were able to make these immediate changes in acceleration without causing significant jarring of the food tray. The time interval used for turning was based on experimentation and results in the robot turning 30-45 degrees each time. The speed was determined through testing as well and is the minimum amount of speed required to reliably cause rotation of the robot. Once a turn is completed, the Arduino waits for the next command from the raspberry pi.

C. Circuit Design

All sensors are controlled by the Raspberry Pi, while the Arduino controls the motor shields and thus the motors. The Arduino is powered by the Raspberry Pi since the Pi communicates to the Arduino through the serial connection anyway. The motors are powered by the external 12V battery through the motor shield. The motor shield is controlled by the Arduino which determines how much current passes through shield. The motors by far draw the most current. With a 5Ah battery, the robot can function for well over 1 hour. The motors draw 0.5A each when driving forward or turning, and 2mA when idle. The motor shield connects the motors in parallel since both motors need 12V to operate. The battery used is rated for 11.1V, but when fully charged is measured at 12V. See Figure 5 for circuit diagram.

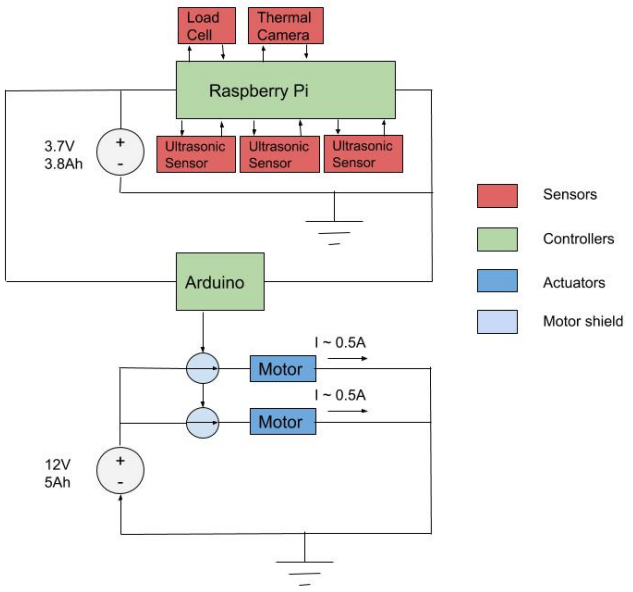

Figure 5: Block diagram for robot circuit

VI. Project Management

A. Schedule

The schedule is included at the end of the document labeled Figure 6. Each task in our schedule is marked with a task number T1 through T51, and its dependencies are listed in parentheses. For example, “T32. Integrate the load sensor with the rest of the code” is dependent on “T7. Assemble tray with load cell”, and “T30. Write framework for empty tray”. Our schedule also shows the tasks by person. Matteo is in green, Isabel is in blue, and Kashish is in orange. Purple tasks are team efforts where multiple members worked to complete it.

B. Team Member Responsibilities

Responsibilities were divided based on course backgrounds. Kashish took charge of software development and writing the program for human detection, movement, etc. This involved all work with configuring and communicating with the Raspberry Pi and developing our movement control algorithm. Isabel lead efforts in the physical structure of the robot, from the chassis to the motors and putting it all together. In addition, she worked on the Arduino code for controlling the motors and communicating with the Raspberry Pi. Matteo handled power constraints, circuit design, and the selection of peripheral features, such as the load cell and music speaker. The whole team was responsible for budget, appearance, and any course-related tasks.

C. Budget

A list of all the parts we bought is attached at the end of this report in Table 3.

D. Risk Management

Significant delays in the ultrasonic sensor ping and the robot response introduced risks in the robot stopping in time to avoid hitting obstacles. There were even be instances where the ultrasonic sensors fail to detect an object at all, especially when it was covered in loose/soft clothing and had no hard surface. To first accommodate this risk, we decided to drive the robot as the slowest speed possible that would still reliably result in the robot moving in the desired direction. This ended up being roughly 1 ft/s. This gave us the maximum amount of time possible to get feedback from the sensors and act on it before colliding with the obstacle. We also increased the rate of deceleration for stopping to shorten the time required to completely stop. Additionally, we designed the robot to make auditory announcements, so that patrons would be aware of the robot’s presence and its actions.

Even with these risk reduction measures, we encountered an issue with the ultrasonic sensors not detecting certain types of

clothing. To mitigate this during the demo, we had participants hold an 8”x8” wooden board at the height of the ultrasonic sensors to ensure they would detect the human. We also put stop buttons on the front and back of the robot that immediately cut off power to the motors, in case we, or any guests, needed to shut it off immediately. All this considered, even in the event of a collision, the robot did little harm due to its light weight and slow speed. It would come to a stop with minimal applied resistance.

There was also the risk of the robot being unbalanced and shaking during acceleration. While the robot did oscillate upon changes of acceleration, the physical construction of the robot, with the tennis balls on the front and back, prevented the robot from falling in either direction. They also acted as small shock absorbers in those instances. In addition, we incrementally decreased the speed of the robot during deceleration which decreased the overall oscillation of the robot. We also attached the battery and hardware to the base to prevent top-heaviness. We further stabilized the tray with foam to prevent it from wobbling. In the end, the food did not shake off the tray during the demo and stayed securely in place.

VII. Related Work

There are several products in market, such as the Rumba, that have mobility similar to our robot. Many such products also are capable of object detection through the use of various sensors. Even within our design class, there were projects dealing with floor mapping and crowd detection, all using a similar set of sensors. Should the projects be combined, we could design a robot with smarter pathfinding through groups of guests, as well as more functions for interacting with guests, such as taking pictures.

VIII. Summary

Overall, our system was able to meet the general expectations we had for it. That being said, we were not able to meet all of our design specifications. In terms of mobility, we decided to have the robot move at a speed of roughly 1 mph instead of 2 mph in order to increase safety. Our robot was able to move at 2 mph, however. Also, YoServe was able to meet the specification of not having food fall off upon starting, turning, or stopping.

In terms of performance, the robot was able to operate for an hour. For human detection, the thermal camera is able to detect people, both individuals and groups of people, within 8 meters of the robot, however it does need to be calibrated for the temperature of the environment in order to do so.

With respect to safety, while the ultrasonic sensors can detect hard objects within 12 feet of the front of the robot, this is not reliable, especially for humans and different types of soft obstacles. If an object is detected while it is 4 feet away from the robot, it is able to stop before it is 1 foot away from the obstacle. Although, this is not guaranteed if the obstacle is detected much closer than 4 feet. We did meet the requirement of having an emergency button on the front and back of the robot which immediately stop the motors. We also managed to have the robot make auditory announcements so its presence was noticed by guests, however the speaker ended up being too quiet to be heard in the demo space.

In terms of sensing, the load cell can detect food items on the tray and recognize when it is empty. When it does so, it stops moving until the tray is refilled. We were not able to meet our sensing time requirements. Our original goal for taking and processing a thermal image was 100ms. It ended up taking us roughly 900ms to detect humans. As for the time taken for the software algorithm to know if an object is detected, we originally hoped for it to be 10ms and it turned out to be ~1.5s. While not addressed in the specifications, the ‘start moving forward’ command takes ~0.9s to execute and the motor stop sequence takes 1.2s for the robot to come to a complete stop.

A. Future Work

While we do not plan to continue this project, we have thoughts on how we would do so. We would hope to find solutions to shorten the time needed to check surroundings as the robot turns to look for more people. To do this, we would use higher quality sensors and a processor faster than our Raspberry Pi, or at least find a way to thread sensor processing to do it in parallel.

B. Lessons Learned

Integration of tasks, both hardware and software, takes time, often more than is anticipated. Budget the schedule for lots of debugging. Sometimes this requires joint effort and can’t be done in parallel, tasks that may seem independent are often related and need to work together at some point. Everyone needs to be involved in order for this process to go as smoothly as possible. Additionally, purchase extra boards and materials ahead of time, as they will probably break. Make sure to communicate and know what your partners are doing and working on, you never know when you might be able to help them with an issue to need to fill in for them. Also, it just helps to make you feel like a team and prevent animosity.

REFERENCES

[1] Fernández-Caballero, A., Castillo, J., Martínez-Cantos, J. and Martínez-Tomás, R. (2010). Optical flow or image subtraction in human detection from infrared camera on mobile robot. *Robotics and Autonomous Systems*, 58(12), pp.1273-1281.

[2] Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6), pp.1153-1160.

Product Name	Qty	Price/unit	Total	Notes
Motors + Gearboxes	2	36.95	73.9	
Wheels	1	9.95	9.95	
Motor Shield	1	49.95	49.95	
Brackets for motors	1	7.45	7.45	
Hubs	1	6.95	6.95	
12V 18 Amp HR battery	2	23.98	47.96	Bought but not used
Thermal Sensor	1	39.95	39.95	
Ultrasonic	3	3.95	11.85	
Load cell	1	8.5	8.5	
Tennis balls	2	3	5.99	
Raspberry Pi	1	35	35	
Raspi battery pack	1	17.99	17.99	Newly acquired
Arduino	2	16.98	33.96	
SD Card for Raspi	1	4.99	4.99	Newly acquired
9V batteries	1	6.88	6.88	Bought but not used
11.1V battery	1	36.99	36.99	Newly acquired
Extra Arduino - 2 pack	1	17.49	17.49	Newly acquired
Ultrasonic Brackets	3	0.93	2.79	Newly acquired
Extra Raspberry Pi	1	39.99	39.99	Newly acquired
Fabric	2	5.99	11.98	Newly acquired
Speaker	1	11.99	11.99	Newly acquired
Push Button pack	1	6.99	6.99	Bought but not used
Wood/brackets/buttons	N/A	25.00	25.00	Acquired from Roboclub and/or previous projects
Shipping Adafruit	1	7.81	7.81	
Shipping Pololu	1	6.72	6.72	
Shipping BatteryShark	1	25.99	25.99	
Exp. Shipping Amazon	1	7.98	7.98	
		Total:	562.99	

Table 3: Parts Ordered and Budget Tracking

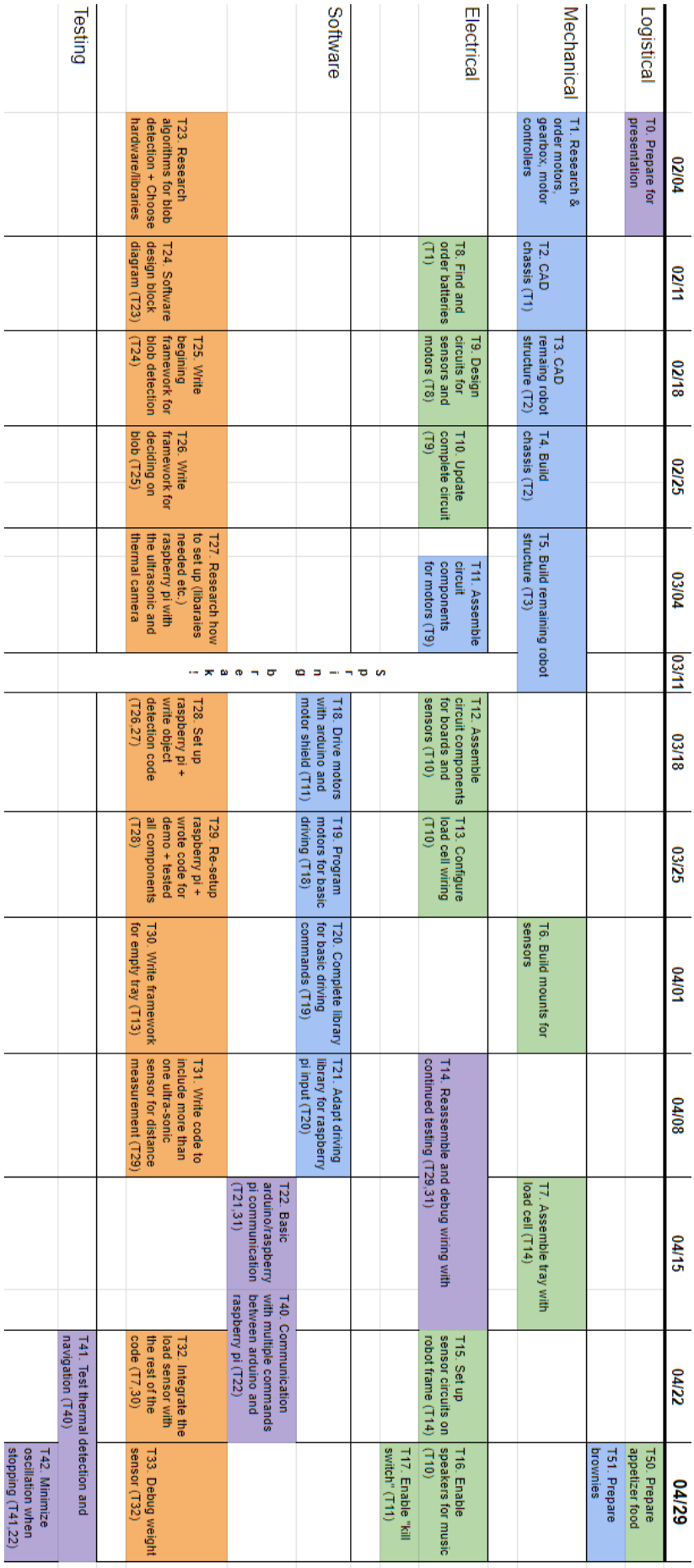

Figure 6. Complete Schedule