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.

For planning and motion, the robot should spin at most 360° before deciding a direction to move in or making the decision to remain where it is. We will monitor this throughout all of our testing, as well as test the robot both in an empty room and with people completely surrounding it in order to ensure that it never continuously spins.

III. Architecture and/or Principle of Operation

Our YoServe robot will be designed to meet this use in several ways. It will utilize a Raspberry Pi for sensor data processing, and Arduino for motor control, and have a wooden frame suited for navigation and tight turns. As shown in figure 1, the Raspberry Pi will take in input from three different sensors: the ultrasonic sensors, the thermal camera, and the load sensor in order to determine the motion required from the robot. It will pass the driving instructions to the Arduino, which will use the motor shields to control the motors and navigate as per the instructions provided.

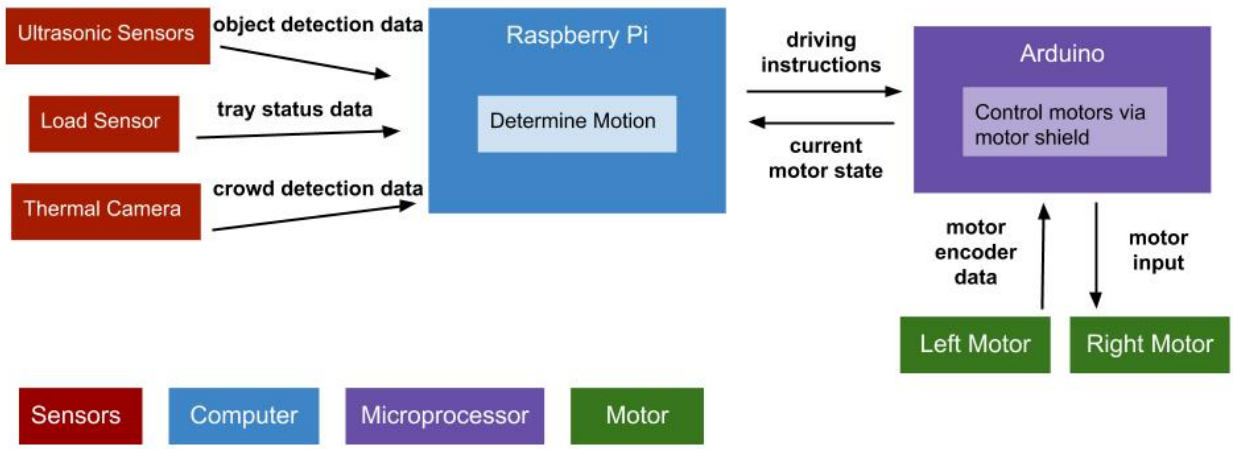

Figure 1: Overall system block diagram

A. Physical Structure

The overall shape of the robot, as shown in Figure 2 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 will provide additional stability to the robot as the rubber will absorb some of the oscillation in speeding up or slowing down. The frame will be 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 will make it more stable and less likely to fall in a specific direction. Additionally, the battery and hardware will be 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 will help to reduce any oscillation of the robot. We will also orient the battery so that its weight is balanced over the two wheels rather than the tennis balls so that the robot will not have to overcome as much static friction upon starting.

B. Motion Control

In order to further support robot stability, we will implement a trapezoidal velocity profile with a low constant of acceleration. This will cause less oscillation of the robot upon stopping and starting. We will conduct tests in order to determine the slope of the velocity profile. Additionally, if we continue to find oscillation in the robot, we will round the corners of the trapezoidal velocity profile until there is minimal oscillation. This, in combination with the physical structure, should keep the robot stable and prevent food from rolling/falling off of the tray.

C. Sensors Signal Processing

Looking closely at the sensor input processing, a Raspberry Pi 3 would be used to process the input from our three different sensors: the thermal camera, the ultrasonic sensors, and the load sensor. First, a load sensor is used to detect whether the food is tray empty. If so, the robot stops moving, and waits for the tray to be filled. Then we would use the thermal sensor to get a thermal image, which can be processed

by OpenCV in Python to figure out areas of crowd in the room. Using OpenCV's inRange function for identifying areas falling with a particular color threshold, we can get the general direction the robot needs to move in order to reach a person or a group of people. Then, we use the ultrasonic sensors to detect the distance from that crowd, and stop the robot when that distance becomes less than 1 foot. Once the distance and direction has been determined, the Raspberry Pi would then send over these driving instructions to the Arduino for controlling the motors.

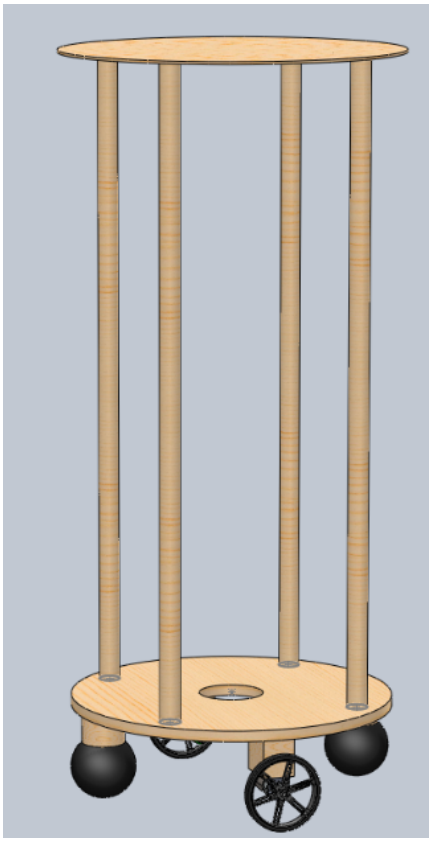

Figure 2: CAD design for the robot structure

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. One of the biggest advantages of the first approach is the availability of widely used frameworks that can be used for implementing human detection. This would have meant a lot of online support for bug fixes and improvements. The problem with this approach is the time it takes; each frame of the video feed would have to be fed into the neural net in order to detect a human. Kashish had previously worked on a similar project on her 2.4 GHz Macbook Pro, and it was quite slow. On a Raspberry Pi 3 B+ running at 1.47 GHz, this approach would be too slow for our robot to meet any system requirements. If we wanted to run the algorithm faster, we would need to buy much faster hardware options, which usually cost well more than our allocated budget. The second approach works very well in terms of speed, since it is only looking for pixels within a fixed range. On the 2.4 GHz Macbook Pro, this takes less than 10 milliseconds, so it should still be fast enough for our requirements on the Raspberry Pi. The drawback for this approach is 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 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. Since the second approach works for our requirements, and is well within our budget, that's the approach we chose to implement.

B. Sensors for Object Detection

For the object detection part of the project, we considered two different sensors: IR sensors, and ultrasonic sensors. IR sensors work by reflecting light waves to figure out the proximity or distance to objects. In general, they work better at defining edges of an area than ultrasonic sensors. However, IR sensors are more sensitive to variant light conditions. In particular, they do not work well in dark environments. Ultrasonic sensors, on the other hand, are usually insensitive to hindering factors like light. However, since they work by reflecting sound waves, they cannot really detect people wearing soft clothing like fur coats etc. For our application area of a cocktail hour, appetizer-serving robot, we can expect the lighting to be pretty dim in the room. We can also expect that, in general, we would be in an enclosed room, which would mean that people would not be wearing their coats. Therefore, we chose to go with the ultrasonic sensors.

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 may also be able to replace our current one. The new battery could supply around 5-10 Ah, which would still allow for desired duration, but also lower the total weight of the robot significantly. With a lower weight, the motors wouldn't have to work as hard, benefitting mobility.

V. System Description

A. Software Design

We use a Raspberry Pi 3 Model B+ to determine robot motion using input from the three different 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 3, first, a KNACRO load sensor will be used to detect whether the food tray has a weight higher than our calibrated threshold. If not, we would assume the tray is empty and the robot control module would be signaled to stop the motors. This check would be performed in a loop, and the robot would wait until the tray has been loaded again. Once we have determined that the robot is loaded with food, we would use an Adafruit AMG8833 8x8 Thermal Camera Sensor to capture the thermal image, and pass it in to our thermal image processing module.

In particular, for the thermal image processing module, we decided on using OpenCV to identify areas of crowd, and get their direction. First, we de-noise 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

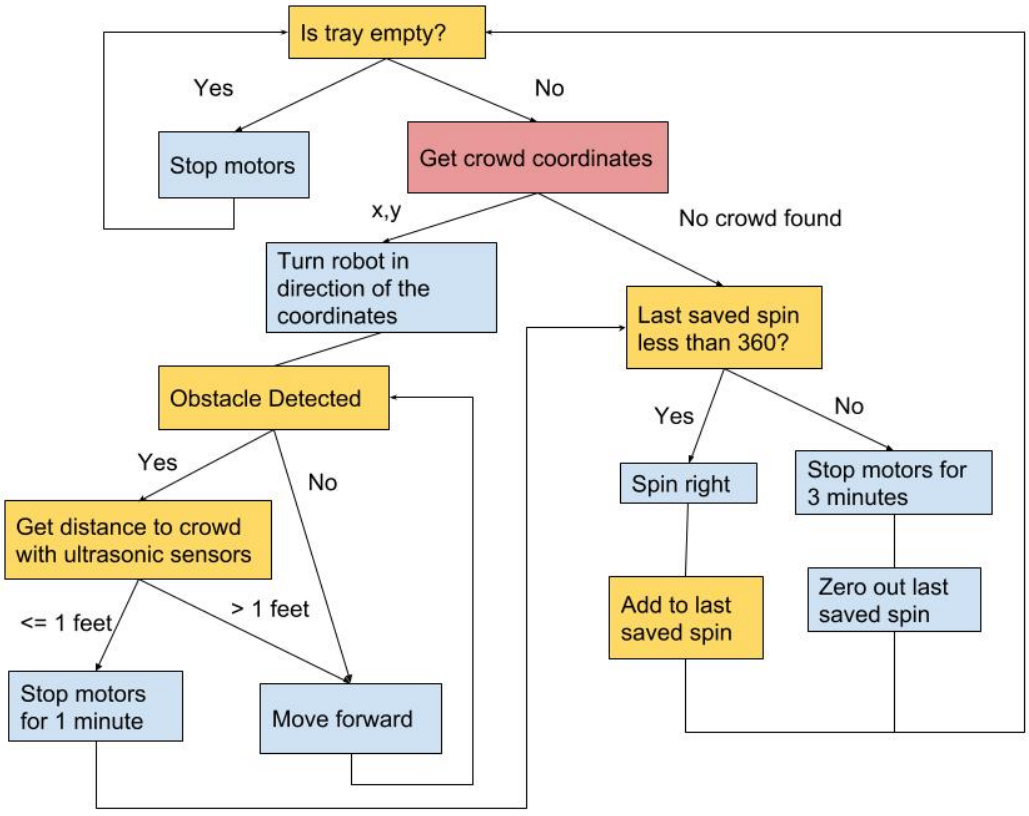

Figure 3: Software System Flow Chart with blue representing the robot control module, red representing the crowd detection module, and yellow representing the load and ultrasonic sensor processing module

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 a negative value is returned from this module. However, if more than one contour is present, we get the centroid for the leftmost contour. This is done because our robot always spins to the right, and we want to move towards the person nearest to the robot in angle. Using the centroid (x,y), we can calculate the direction the robot needs to head in. Figure 3.1 shows a flow chart summarizing this module.

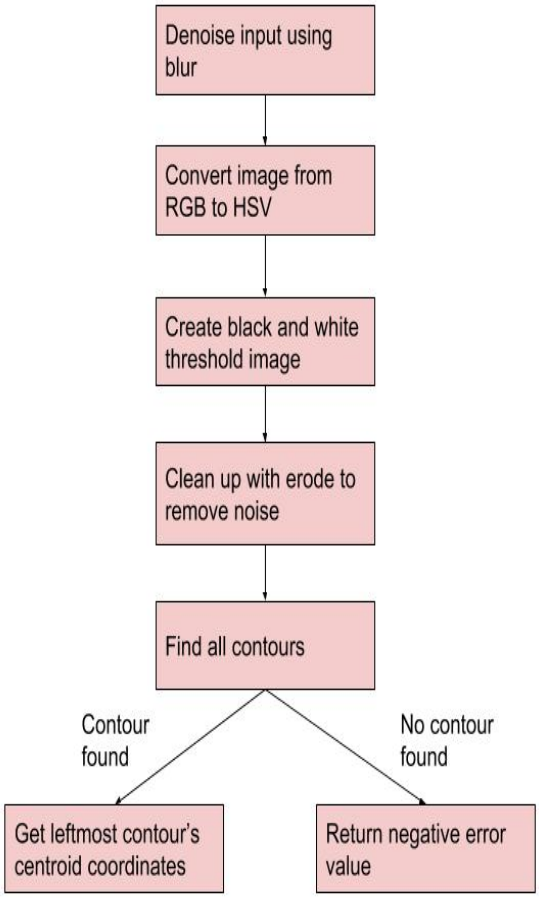

Figure 3.1: A flow chart zooming into the thermal image processing module

Once we have the direction from the thermal image processing module, it would be passed on to the robot control module that would spin the wheels accordingly. Then, we use a HC-SR04 Ultrasonic Sensor to detect the distance from that crowd. 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. In this case, because we are connecting them to the Raspberry Pi, we would use the CircuitPyton library (free and easy to install) to interact with the sensor output. If the crowd is out of range of the ultrasonic sensors, or the crowd is nearby but further than 1 foot away, the robot control module would be signaled to move forward. This is a wait loop similar to the one used for the load sensor, in that, once the ultrasonic sensors detect a distance less than 1 foot, the robot control module would be signaled to stop the motors for a predetermined time threshold. This threshold can be changed as necessary, but has currently been set to 1 minute, giving people enough time to pick up their food.

Once the minute is up, or in the case where the image processing module return an error, we want to add a random spin to our robot a little, so it doesn't detect the same person/group again. However, we do not want our robot to keep spinning in case there are no people around it, so we keep a counter to check its rotation, and stop for a predetermined time, in this case 3 minutes, before checking for the tray weight again.

B. Circuit Design

All sensors will be run by the Raspberry Pi, while the Arduino controls the motor shields and thus the motors. The motor shields will be controlled by how much current passes through them, separate from the Arduino so as to not fry the board; the motors pull by far the most current. With an 18 Ah battery, the robot can function for over 1.6 hours. Since both boards and motors need 12V to operate, we will wire them in parallel. The battery may show 13V between nodes, so a small resistor can be placed at either the anode or cathode. See Figure 4 for circuit diagram attached at the end of this document.

VI. Project Management

A. Schedule

The schedule is included at the end of the document labeled Figure 5. Each task in our schedule is marked with a task number T1 through T28, and its dependencies are listed in parentheses. For example, "T28. Test empty tray" is dependent on "T15. Build tray for food", and "T27. 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.

B. Team Member Responsibilities

Responsibilities were easily divided based on course backgrounds. Kashish is taking charge of software development and writing scripts for human detection, movement, etc. Isabel is leading efforts in the physical structure of the robot, from the chassis to the motors and putting it all together. Matteo is handling power constraints, circuit design, and any peripheral features, such as the load cell and playing music. The whole team is responsible for budget, appearance, and any course-related tasks.

C. Budget

Product Name	**Qty**	**Price/unit**	**Total**
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
Power Supply	2	24	48
Thermal Sensor	1	40	40
Ultrasonic	2	5	10
Load cell	1	8.5	8.5
Tennis balls	2	3	5.99
Raspberry Pi	1	35	35
Arduino	1	16.98	16.98
Shipping Adafruit	1	8	8
Shipping Pololu	1	6.72	6.72
Shipping BatteryShark	1	26	26
		Total:	**327.39**

Table 1: Parts Ordered and Budget Tracking

D. Risk Management

Should there be significant delays in the ultrasonic sensor ping and the robot response, the robot would face risks in stopping in time. There could even be instances where the ultrasonic sensors fail to detect an object at all, especially if it's covered in loose cloth and has no hard surface. To accommodate this risk, we have chosen to keep a low maximum speed for the robot. Even in the event of a collision, the robot of this weight class and low speed will do little to no harm.

If those undetected obstacles are people, there will be further safeguards. The robot will be continuously playing pleasant music, so patrons will be made aware that the robot is approaching. If the robot was to get too close, patrons would be able to move out of the way. There will also be an emergency stop button on the robot itself. When pressed, this button will cut power to the motors and stop the robot immediately.

There is also a risk of the robot being unbalanced and shake during acceleration. We currently have no reason to believe this shakiness will result in food spillage, since the base has four solid points of contact. However, the battery and hardware will be attached at the base to prevent top-heaviness. If testing shows that the robot still shakes too much, we will look into methods of providing feedback to the motor system.

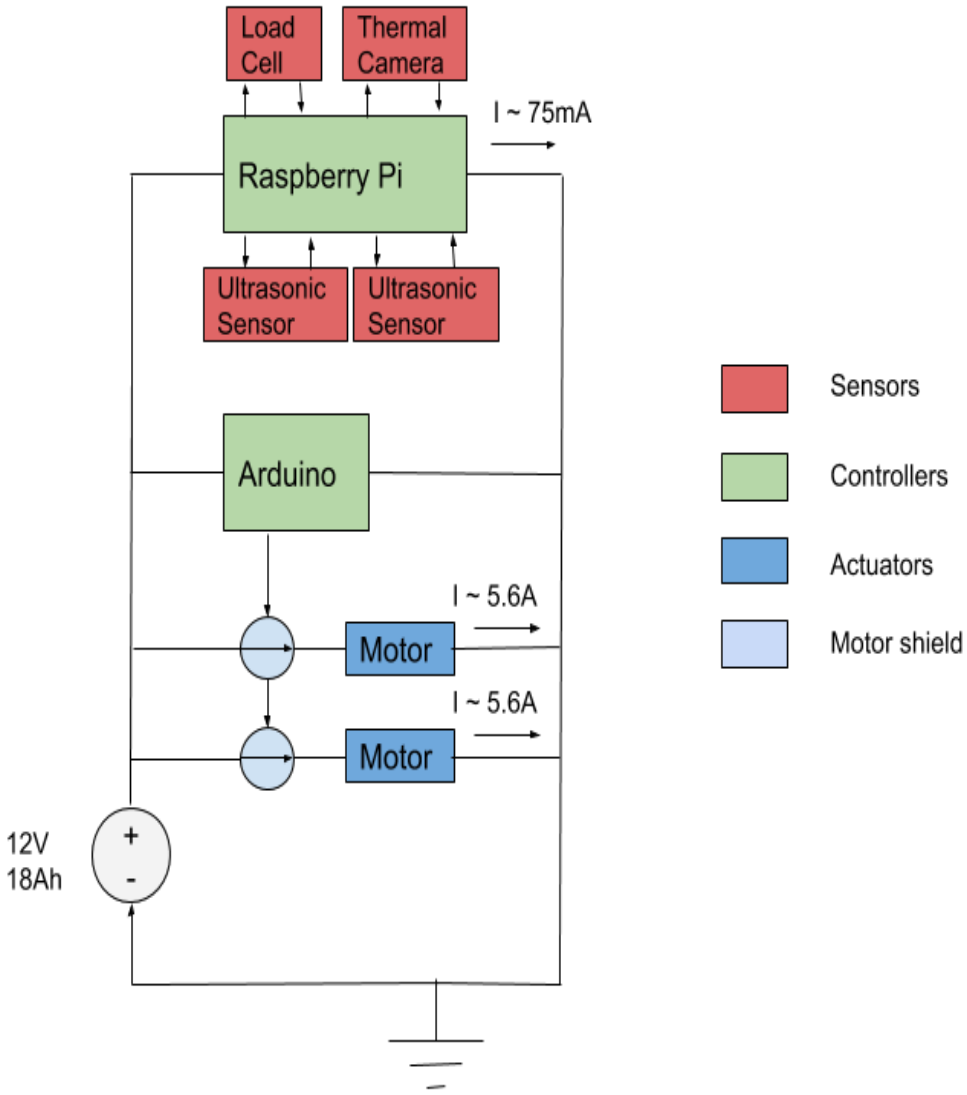

Figure 4: Block Diagram for robot circuit

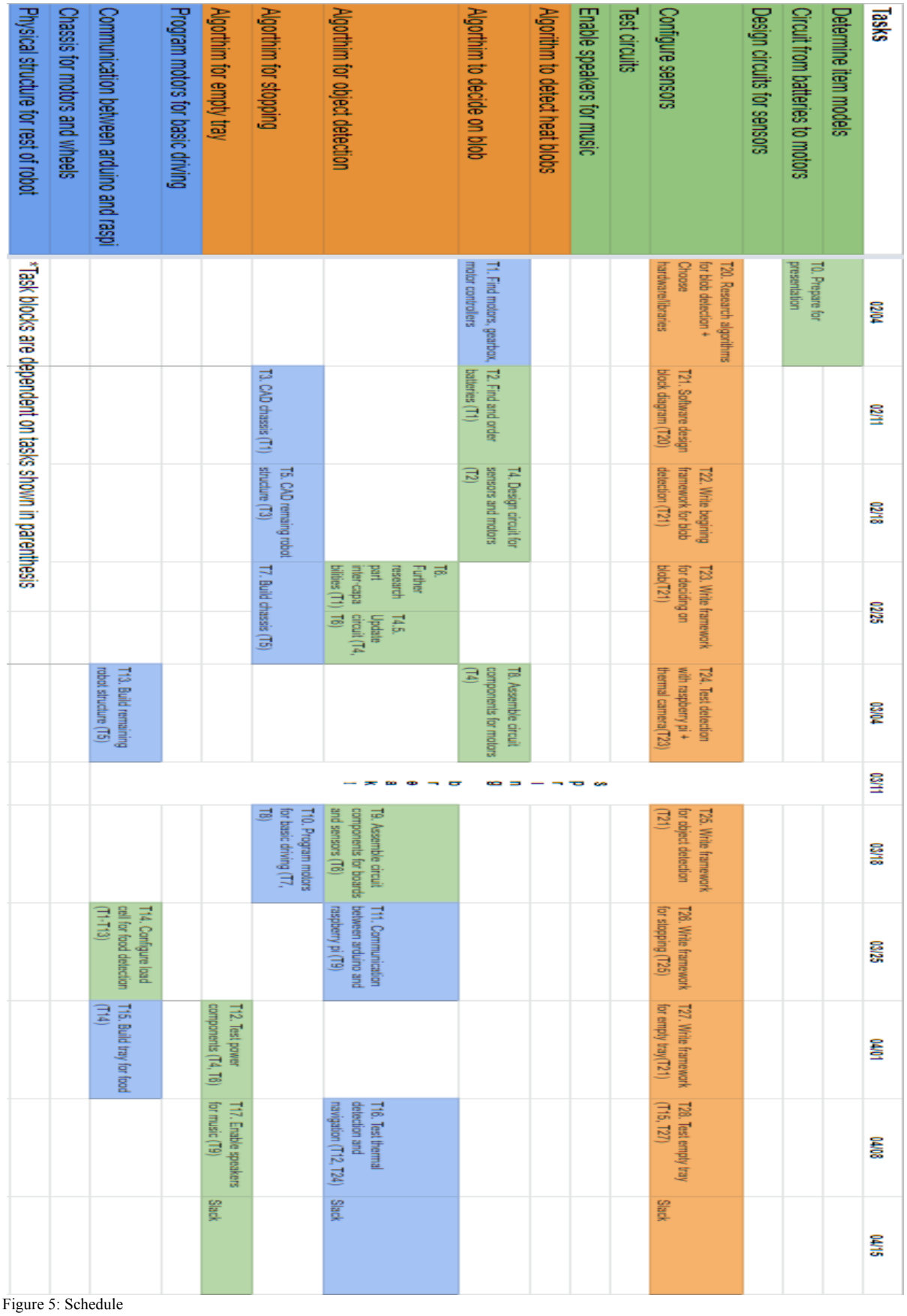

Figure 5: Schedule