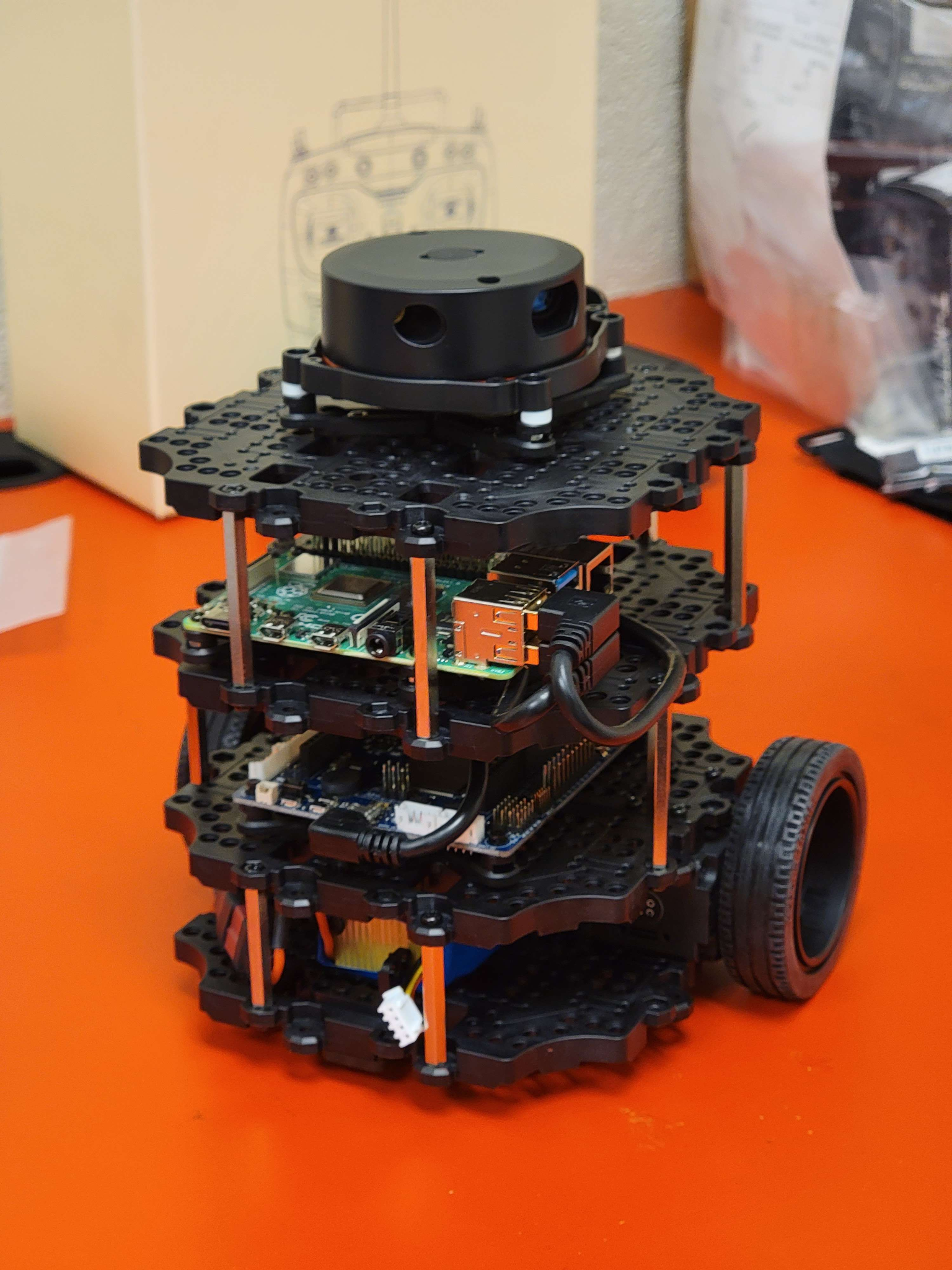
# 

# NibblesBot: Project Design

By Ethen Carrell, Charles Moertle, and Calab Reeder



## Table of Contents

Page #

**Project Name** **2**

**Team Member Names**  **2**

**Revised Requirements** **2**

**Design Description 5**

**Diagrams and Documentation 9**

**Appendix 10**

Block Diagram

Component Diagram

Message Documentation

Storage Documentation

Misc Documentation

## Project Name

The name *NibblesBot* comes from a pet turtle.

## Team Member Names

See **Figure 1** in Appendix

## Revised Requirements

1. Software Installation
   1. Installing Ubuntu
      1. One installation will be done for each team member’s laptop
         1. Oracle VirtualBox will be used to run the Ubuntu OS
      2. Another installation will be done for the robot’s Raspberry Pi 4
         1. Ubuntu will be the main OS so it will not need a virtual machine
   2. Installing ROS Noetic
      1. One installation will be done for each member’s laptop
         1. It will be installed in the virtual Ubuntu OS
      2. Another installation will be done for the robot’s Raspberry Pi 4
   3. Installing PyCharm as designated IDE
      1. Some needed libraries…
         1. SciPy
         2. OpenCV
2. Robot Hardware Modifications
   1. Install 3d printed scoop for moving blocks
   2. Mount camera to the front of the robot
      1. Camera will be facing downwards at an angle to capture what the robot is pushing
3. Verify all Hardware Integrity
   1. Confirm the robot’s wheel motors work properly
   2. Confirm the camera can properly read the top of each colored block
      1. This only concerns blocks immediately in front of or being pushed by the robot
   3. Confirm the battery holds charge for expected duration of time
      1. Expected operating time: 2h 30m
   4. Confirm the 3d printed scoop can accurately move the blocks…
      1. In a corner
      2. Next to a wall
      3. In the open
      4. Next to other blocks
4. Engineering the Environment
   1. External Camera with bird’s eye of the area
      1. The camera will be deployed on a tripod
   2. Build walls surrounding the activity area of the robot
      1. Demarcations will be made using black tape
5. Computer Vision (OpenCV)
   1. Create objects based around each colored block on the map
      1. The location of each block will be stored as a coordinate in the object
         1. Coordinates will be made using the bird’s eye view camera
      2. The letter or number at the top of the block will be stored as a string in the object
         1. Top of the block will be recorded using camera on the robot
   2. Constantly check orientation of robot
   3. Designate area to arrange the blocks
6. Arranging Children’s Blocks
   1. Arrange blocks in specified order
      1. Avoiding the use of Z and N, as well as M and W blocks
      2. Avoiding the use of yellow colored blocks
      3. Only using the distinctly colored sides of the blocks
      4. The order will be A-Z 1-9
         1. Map may not always contain both letters and numbers
      5. Duplicated letter blocks will be placed side by side in the normal order
7. Experiments
   1. Find block
      1. Find a block on the map
         1. Blocks will be in the same spot and orientation each test
         2. Robot will always start in the same corner
      2. Approach block and stop before bumping into it
         1. Does not have to be directly facing the block
      3. Read the symbol from the block
         1. Attempt to read the symbol from multiple orientations
      4. Find orientation of the block in relation to boundary
         1. Degrees will be documented in a similar format as Odometry: -180, -90, 0, 90, 180
   2. Check if wall boundary is working
      1. Robot cannot cross black tape
      2. Robot must turn if driving towards the corner of the boundary
   3. Approach block parallel from front (bottom of letter)
      1. First approach the block parallel to one of the faces
      2. Next attempt to approach block using identified front face
         1. Will require moving around block until on front faced side
         2. Must not accidently move the block in the process
   4. Push block
      1. Push blocks in a straight line
      2. Try rotating blocks
      3. Push blocks while turning
         1. Robot must avoid pushing blocks across the boundary
   5. Navigate around blocks
      1. Keep awareness of multiple blocks by not bumping into any of them
      2. Must be able to understand which movements are acceptable in all situations
         1. Acceptable in the sense a movement will not unintentionally move a block
   6. Push multiple blocks in the same test and arrange the blocks
      1. Keep awareness of block locations when pushing blocks to designated area
         1. Robot should not bump into any other block while pushing
         2. Robot should know which blocks have and have not been moved to designated area
         3. Robot should place the blocks in designated order

## 

## Design Description

### Hardware Design

The NibblesBot is a small mobile NibblesBot built on the open source TurtleBot3 Burger platform by ROBOTIS. The robot's dimensions are 138 x 178 x 192 (L x W x H mm) and is equipped with a Raspberry Pi 4B, an OpenCR board, a LDS-02 LiDAR system, two PID drive motors, and an 11.1V Li-Po battery giving an estimated 2 hours of operating time. In addition to the equipped sensors we are attaching a small usb camera for vision recognition. The Raspberry Pi will be installed with Ubuntu 20.04.05 LTS, ROS Noetic, and all applicable drivers for the LiDAR, TurtleBot3 ROS packages, and OpenCR board. The NibblesBot will have a linear top speed of 0.2 m/s and a max rotation speed of 2.8 rad/s.

The primary limitation of the Raspberry Pi is the processing power of the SoC built into the board. This will be addressed by a computer, also running Ubuntu 20.04.05 LTS and ROS Noetic, that will handle the logical operations of the NibblesBot remotely and transmit instructions back to the mobile frame. This is done using the computer as a ROS master node and the rostopics as a go between for data transmission. In this configuration the NibblesBot will only need to listen to the master node for movement instructions and transmit image data to the master node for image processing and logic processing. Further considerations will need to be made based on the rate at which the pictures can be sent as the network will be a limiting factor.

### Algorithm Design

For the search and movement algorithm of the robot, the search area will be enclosed by a black line that will work as the boundary. This boundary will be made out of black tape. The surface that our NibblesBot will be using to complete its task will be a smooth, hard surface that will be a solid color such as white or brown. The color matters in this experiment since we are using color recognition to identify blocks. There will be an assembly zone denoted in the environment that NibblesBot will take the desired blocks to so that they can be placed in order to solve the puzzle. Blocks will be placed far enough apart so that no two blocks will be in the view of the camera on NibblesBot at one time. No additional obstacles will be present besides the blocks being used for experimentation. The starting position of the NibblesBot will be set as a “zero” point and will work as the assembly zone that blocks should be brought to. The specified order of blocks will be stored in a list (order list) and the master node will store the positions of any blocks it encounters in another list (found list) along with their coordinates. When a block is found that corresponds to the next block in the order list the master node will store its current position and orientation and proceed to push the block to the assembly zone. After a block is successfully delivered the master node will check the order list against any block in the known list and if one is found it will navigate to the known coordinates and proceed to move the block to the assembly zone as it did the previous one. In the event that the next block is not a known position, the NibblesBot will return to the coordinates stored as the last searched position and resume the search pattern. If the field is completely searched and a block is not found the NibblesBot will search the field once more to prevent cases where a block is missed. If the second search pattern completes the NibblesBot will notify the user that a block could not be found and prompt if the user wishes to continue with retrieval. If the next block is also not in the found list the NibblesBot will continue to prompt the user until the order list has been completed or until a block is in the found list. If the block is in the found list the NibblesBot will proceed to retrieve the block. In the event the user responds to not continue or the order list is completed the NibblesBot should enter a safe state and terminate the process.

#### Computer Vision

One of the integral computer vision algorithms of our project is the ORB (Oriented FAST and Rotated BRIEF) feature detection algorithm. Additionally, the FAST in ORB stands for “Features from Accelerated Segment Test”, and BRIEF stands for “Binary Robust Independent Elementary Features”. ORB feature detection is an algorithm that is often used in object detection, using FAST as a means to extract feature points from some image and then using BRIEF to combine these feature points so they can locate an object. The features from FAST are calculated using pixel brightness, if a certain number of pixels surrounding some selected pixel (keypoint) are brighter or darker than the intensity threshold, then that pixel keypoint and location will be taken to be a ‘feature’ of the image. FAST in this case derives its name from the fact that, while comparing the pixel brightness surrounding some selected pixel, the algorithm first checks a pixel in each of the cardinal directions and the rest of the pixels will only be checked if at least three of the four cardinal pixels satisfy the intensity threshold we are looking for. After scanning the entire image for pixels that qualify as features, BRIEF will begin computing descriptors for each detected feature. Each descriptor will act as an ID, allowing the software to record any interesting information about a feature’s surrounding pixels. The gathered descriptor information is then converted into a binary feature vector in order to represent an object. At this point, the algorithm is done scanning and compiling information about what to detect in an image. The next step is to begin analyzing a second picture for the object we are looking for. In order to achieve more accurate results, the second image will need to be blurred as the BRIEF algorithm is very noise sensitive. The results will vary depending on the level of blur applied to the secondary image, so independent testing is needed to achieve an appropriate level of success to blurred image ratio. After blurring the image, the algorithm creates a similar binary feature vector as was created from the initial image using features gathered from the new image. The final step is to use a matching algorithm to compare the features and descriptors of both images in order to find the object from the first image in the second image. The matching algorithm we plan to use in our version of ORB is a brute force matcher that will compare each pair of descriptors for the object we are looking for, before drawing a square around the desired object.

In our project, we will use the ORB algorithm to differentiate blocks from their surroundings, which may only be the map itself including the surrounding walls, or it may be several blocks located in close proximity to each other. Before using this algorithm, we must first compile a set of training images that will be used in tandem with the testing images (images taken directly from the robot’s camera). These images will be used to identify the symbol on each block, further experiments with obfuscating block images will allow a singular image to determine blocks at a distance, although this has yet to be tested.

A second computer vision algorithm integral to the success of our project is centroid detection. Centroid simply means center of an object. All though in our case, the centroid will be the center of a collection of pixels which have been masked based on a particular color we are looking for. In order to calculate the center, the moments about the x and y axis need to be calculated. Each moment is derived by calculating the weighted average of image pixel intensities in some masked image where each pixel is a binary representation of the original image. Furthermore, once the moments have been calculated, we can compute the x and y component of the centroid by dividing the moments, thus giving us the coordinate of the center of our image.

Combining the computer vision algorithms discussed so far, the plan is to use ORB feature detection to not only differentiate the blocks from one another, but also to allow dynamic cropping of the robot’s camera feed. Dynamic cropping in this instance refers to cropping each successive image received from the robot’s camera so that the block we are focused on is always the only one that appears in the robot’s view. Doing so will allow the use of centroid detection on the top face of the block, we will then relate the cropped image back to the original so that the robot can use the centroid coordinates as a reference for which direction to move in order to head directly to the block. A similar approach will also be made in order to push the blocks to the assembly area. First we will approach the block using the previously described combination of ORB and centroid detection. Next, we will continue isolating the image of the block from the rest of the image while the robot begins to push the block, all the while the centroid detection will keep the robot aligned with the block. This process will continue until the robot manages to push the block into its correct position in the assembly area

The third integral computer vision algorithm has yet to be decided, although we know that it must be capable of calculating the orientation of an object relative to some other object. We currently plan to use the walls as a stationary reference point to use in relation to the robot and the blocks. Using the wall will allow the robot to know which direction to approach each block in order to rotate them accurately, and thus allow for the blocks to be assembled in a legible way once they are pushed to the assembly area.

### Experimental Design

Our process for developing NibblesBot will have a variety of experiments that get more complex as we complete each experiment.

#### 1st experiment

The first test will be to find and identify a block on the map. The specifications for this experiment is that the block will remain in the same position and orientation during each test, and Nibblesbot will always start in the corner of the testing environment. This is to make our beginning experiment repeatable and allows us to get consistent results. The first goal of this experiment is for NibblesBot to find and detect the block. NibblesBot should then approach the block and stop before touching the block. The next step in the experiment is for NibblesBot to read the symbol on the block from a single orientation each time – more orientations of the block will be tested later once consistent results are created. The software to be used in calculating the orientation is still being researched, further testing will be performed after substantial software has been created.

#### 2nd experiment

The second experiment will test if the boundary of the testing environment is working correctly. NibblesBot should treat the black tape boundary as a wall. Upon approaching the boundary, NibblesBot should stop or turn before crossing over onto the boundary. We will test this by having NibblesBot drive toward the boundary and observe to see if it does the intended behavior. Upon a few successful trials where NibblesBot performs as intended, the next test will be allowing NibblesBot to find and identify blocks without crossing the boundary.

#### 3rd experiment

Our third experiment will be approaching the block parallel from the front of the block (bottom of the letter).The first test for NibblesBot when it comes to approaching the block is to approach the block from one of the blocks faces. Any of the faces of the desired block will do because this is a test to verify that the correct block is being chosen. The next step in this experiment is to have NibblesBot to approach the desired block from the front-face of the block. NibblesBot should move around the block until the front-face is found and orient itself to be prepared to push the block. NibblesBot must complete this without bumping into the block and changing the block's position.

#### 4th experiment

The fourth experiment will have NibblesBot begin to push the blocks. The first step in this experiment is to have NibblesBot push a block in a straight line. This test will consist of putting a block in front of NibblesBot and seeing if it is able to both push the block while also correcting the position of the block in motion. This is to make sure that NibblesBot does not lose sight of the block or leave the block behind while moving. The next step is to try rotating the blocks so that the desired face is oriented towards NibblesBot before pushing towards the assembly zone. This is to handle situations where NibblesBot is unable to orient itself to properly face the front of a block – an example would be a block that is too close to the boundary for NibblesBot to move behind. The final step is to make sure that NibblesBot does not push any blocks onto or outside of the boundary since this would make the blocks unavailable to NibblesBot. This testing is to prevent the puzzles that NibblesBot is trying to solve from becoming impossible.

#### 5th experiment

The fifth experiment will involve NibblesBot navigating around multiple blocks at the same time without moving any of them. The first few tests will use widely spaced blocks to allow the robot plenty of room to navigate, slowly shrinking the gap until the robot will have an inch gap between its wheels and two blocks if one were placed on either side of NibblesBot. The robot will be made to drive in between blocks, and circle around the blocks multiple times to ensure no issues appear after prolonged testing. These tests will thoroughly check to see how accurate the robot’s spatial awareness of multiple blocks is, as well as check how thoroughly the robot understands the movements it can make in all situations.

#### 6th experiment

The sixth experiment involves pushing multiple blocks in the same test. Awareness of the locations of the blocks is critical. NibblesBot will keep track of the position of the blocks as it comes across them. If NibblesBot comes across an unnecessary block, it will mark the block's coordinates and move on to searching. Once a desired block is found, NibblesBot will orient the block so that the front-face of the block is being pushed towards the assembly area. It is important to note that NibblesBot will not bump into blocks while pushing a block to the goal area. NibblesBot will keep track of the desired blocks that are brought to the assembly zone and continue to search for all the blocks until all necessary blocks are found. The final step should be that NibblesBot should order the blocks into the proper order.

## Diagrams and Documentation

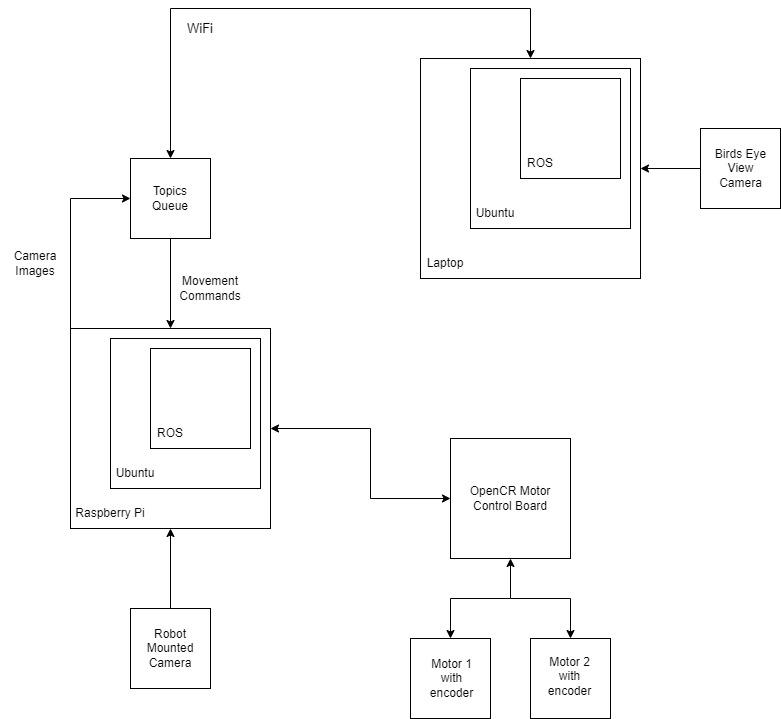
1. Block Diagram (See **Figure 2** in Appendix)
2. Component Diagram (See **Figure 3** in Appendix)
3. Message Documentation (See **Figure 4** in Appendix)
4. Storage Documentation (See **Figure 5** in Appendix)
5. Misc Documentation (See **Figure 6 & 7** in Appendix)

## Appendix

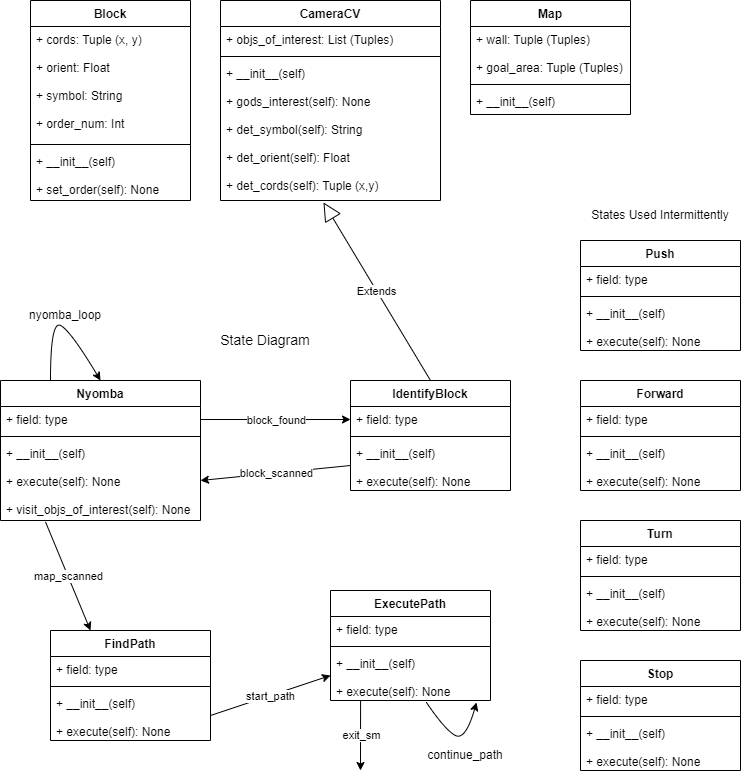
### Figure 1: Team Member Names

| Name | Role |
| --- | --- |
| Ethen Carrell | Researcher, Programmer, Software Tester |
| Charles Moertle | Researcher, Programmer,  Technician |
| Calab Reeder | Researcher, Programmer, Project Manager |

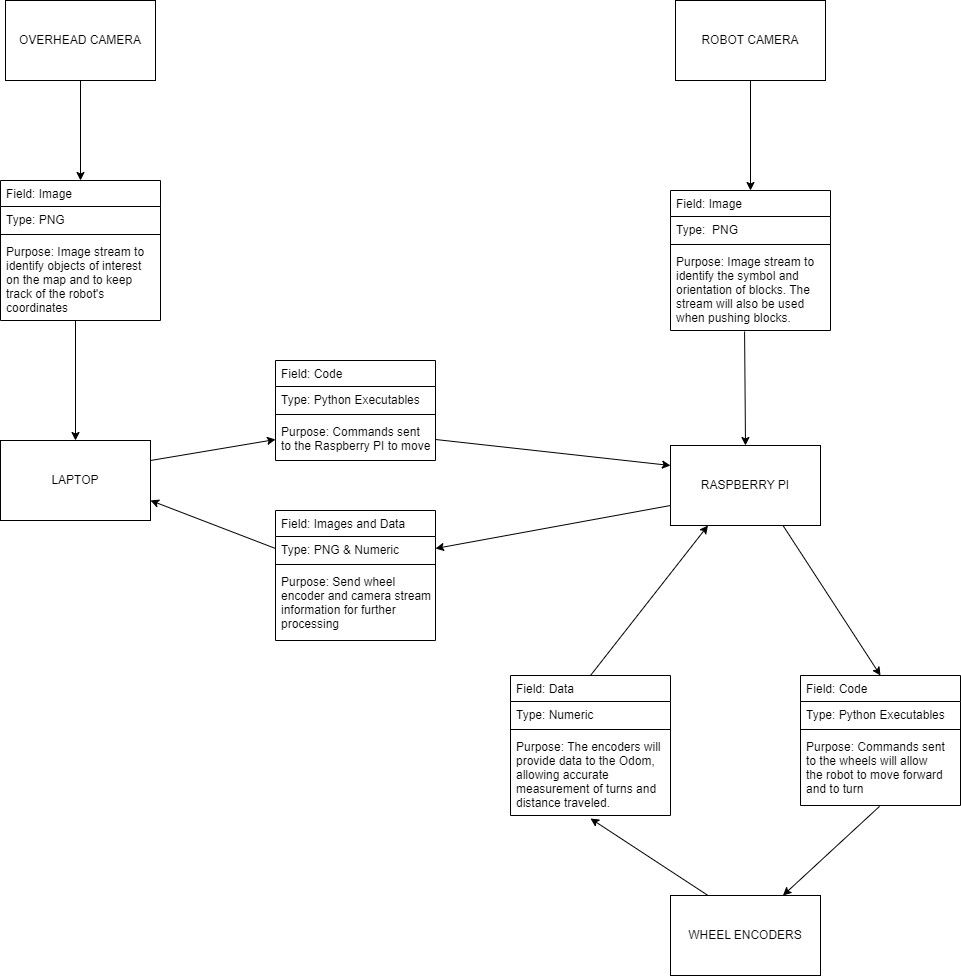
### Figure 2: Block Diagram



### Figure 3: Component Diagram



### Figure 4: Message Documentation



### Figure 5: Storage Documentation

The only storage planned as of now includes a folder containing clear images of each block we plan to use in the experiments we will conduct. This folder will be saved on each members’ laptop and in a google cloud drive to prevent the possibility of loss. Each image will be stored as a PNG and accessed as needed.

### 

### Figure 6: Hardware Dimensions

### Schematic of TurtleBot3 from Robotis documentation

### Figure 7: Hardware Layout Diagram

### 

### Schematic of TurtleBot3 from Robotis documentation