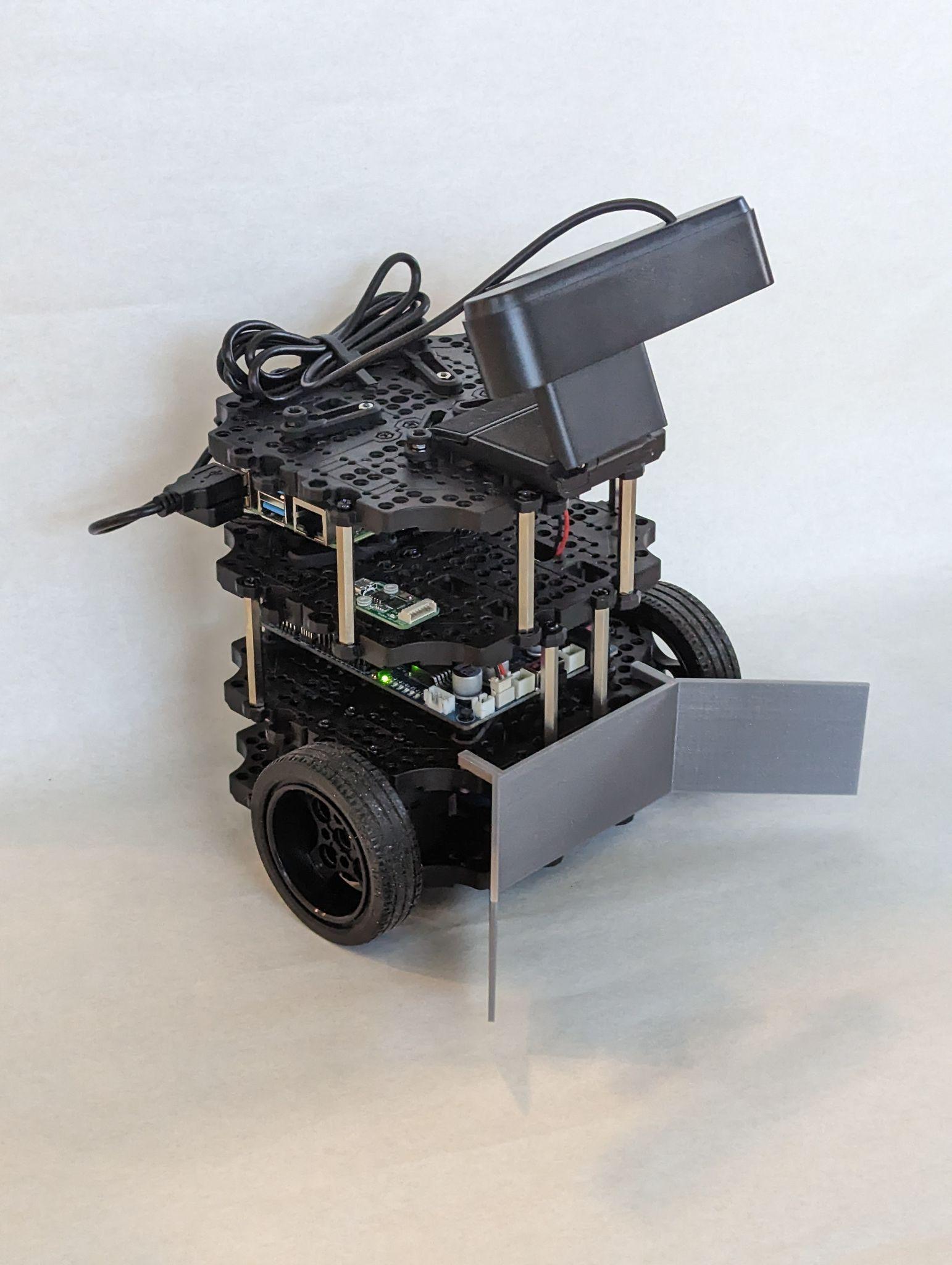
# NibblesBot: Project Update

By Ethen Carrell, Charles Moertle, and Calab Reeder



## CHANGES/DEVIATIONS

There were several changes made to the initial design of the robot in an effort to improve or rectify unforeseen issues. The first change required the removal of the LiDAR mounted to the top of the robot in order to make room for a front-facing camera. Additionally, the overhead camera was deemed unnecessary due to several design decisions, such as using odometry to understand the position of the robot approximate to its origin, and creating a staging area parallel to the left wall in order to gather the blocks for further processing. The third change involved the removal of dynamic cropping through the use of ORB and polylines, and instead switching to using Canny edge detection. In this instance, Canny edge detection was far more accurate and its stability allowed for repeated use of dynamic cropping without the worry that the wrong block will be selected as was the case when using ORB. Another deviation was the use of three datasets based on color (red, green, blue) instead of a single consolidated dataset. We split the dataset in order to increase the accuracy of the ORB feature detection, lowering the chance that similar looking blocks were mistakenly identified. The final change from the initial design resulted from the Oracle Virtual Machine (VM) being unable to provide a bridged connection on our laptops. Due to the ip address of a non-bridged VM not being the same as that of the laptop itself, we were unable to update the .bashrc file with the correct ip address, and thus were unable to send commands to the robot. The solution to this problem required the use of a dual boot, which directly booted the laptop on the Ubuntu Linux distro.

## COMPLETE ACCOMPLISHMENTS

Our team achieved several significant milestones during our project. First, we built the TurtleBot3 Burger and installed the necessary software, including the Ubuntu Linux distro, ROS, and the ROS-Noetic packages for TurtleBot3. In addition to constructing the base robot model, we also modified the robot by removing the LiDAR on the top plate of the robot and instead installed an Argmao camera to allow for computer vision. The second modification involved designing and 3D printing a cow catcher that was attached to the front of the robot in order to manipulate the blocks in the area more easily. The next big accomplishment involved the computer vision side of things. We managed to accurately locate blocks using Canny edge detection, getting the approximate center of each block and using the portion of the frame the centroid fell in to determine where the block was relative to the robot. Canny was also used to dynamically crop the image of the block from the rest of the frame, and then pass the cropped image to ORB (Oriented FAST, Rotated BRIEF). Before ORB begins feature selection on the image, it is run through a color masking of red, green, and blue to identify which data set of image needs to be used. Once the appropriate dataset is selected, ORB can then be used to identify the letter on the block.

After we created most of the individual algorithmic components for this project, we began combining them in the test area. Starting simple, we created a pseudo state machine that had the objective to drive towards a block and stop once it was in range. With this initial success, we continued to scale up the project through steps like returning to the starting position from anywhere on the test area, bringing a block back to the starting position, bringing multiple blocks back to the starting position, and many more. Through these tests, we were able to identify other unique features of the TurtleBot3, such as the fact that the coordinate plane of the robot is not aligned in a traditional manner, instead being rotated ninety degrees to the left. We also discovered the orientation of the robot is calculated solely by the gyroscope located in the raspberry pi 4, and measured the approximate light levels needed to operate the computer vision algorithms. Combining all the previous tests, we were finally able to implement a search that was capable of locating all the blocks in the area and bringing them back to the staging area. For this part of the process, we found it best to keep the actual search algorithm of the robot simple, and thus only use a back and forth pattern where the robot drives forward until it reaches the limit of the area before performing a ninety degree turn in an appropriate direction, moves forward a slight amount, then performs another ninety degree turn.

## PARTIALLY COMPLETE ACCOMPLISHMENTS

Two partially completed portions of this project are the organization of blocks at the staging point and moving the implementation of our robot’s code to a state machine. At the moment, the robot is not able to order the blocks in an order other than the order in which it found the blocks. ORB will be able to solve this issue given more time. The state machine implementation of our project is under development, and its logic is currently not working correctly to be used.

## FUTURE IMPLEMENTATIONS

There are several features that still need to be implemented to the current robot design. The first feature is the use of the camera frame positioned inside the cow catcher. This frame is to be used to determine whether a block has been captured or not, to place the block more accurately, and finally determine and manipulate the orientation of the block. The next unimplemented feature is the use of image masking to keep the robot within the map boundary. While some parts for this feature have already been constructed, the final combination still requires logical statements to determine what degree of detected pixels are appropriate to determine when we are at a wall, and the logic to handle when and where to turn in order to avoid crossing the boundary at particular sections of the wall.

Other future implementations include training machine learning models to more accurately and efficiently locate and identify the blocks. The first model we would like to train is Haar cascade, an object detection algorithm that would allow us to solely detect the blocks in a frame and completely ignore the cow catcher and any other objects that are inside the test area. This would be possible by supplying the model with a multitude of positive and negative images, positive images meaning images that pertain to the objects we wish to detect, and negative images being things we do not wish to detect. So in our case we would pass in a large selection of images of the blocks for positive images, and for negative images we could supply the model with pictures of the cow catcher, the white background, and the black tape border. The second machine learning model we would like to implement is PyTesseract, which works to identify characters in an image. Training a PyTesseract model would remove the need for our mask and ORB combination which we currently use, and would be far more accurate as we could supply the model training process with blocks under different lighting conditions. Once we have trained a new font for PyTesseract based on the blocks we have, we could then combine our two machine learning algorithms by using Haar cascade to find and crop objects out of the main frame, and then running that cropped image through PyTesseract to identify what letter it is.

## UNRESOLVED ISSUES

Throughout the project there were several issues that we were unable to resolve. The first of these unresolved issues is in regards to the lighting for our experiment. Uneven lighting resulted in the ORB frequently misidentifying the blocks, it often caused the masks for the color blue to mistakenly think green blocks belong to the blue category, and uneven lighting also caused issues for the canny edge detection which frequently struggled in shadowy areas. Each issue has the potential to majorly impact the problem as they are integral to the searching and sorting of the robot. The next unresolved issue is the fact that canny cannot handle two blocks in the same frame under certain conditions. As we are currently relying on canny to flag a block as soon as it is found, and then drive towards that block, if a second block is introduced into the equation the robot will rapidly alternate its heading, trying to approach both blocks at the same time. Since this problem can be avoided for the moment by appropriately spacing the blocks, this only has a minor impact on the project’s success. The final unresolved issue is in regards to properly orienting the blocks relative to other blocks. As it stands, the blocks are merely moved into the appropriate position regardless of their orientation. Given that the orientation does not prevent the robot from properly organizing the blocks, this has only a minor impact on the overall project.