# NibblesBot

Ethen Carrell

Charles Moertle

Calab Reeder

## Abstract

Our team will use the TurtleBot3 Burger to rearrange several children’s blocks into a specified order. To accomplish this task, we will assemble the mobile robot and modify it to suit the assignment. Several of the modifications planned include: attaching a camera to allow the use of computer vision algorithms, as well as attaching a 3d printed scoop on the front of the robot to facilitate better movement of the children’s blocks. We will perform additional modifications as needed.

The mobile robot will operate in a flat, stable environment with no obstacles, the only obstructions being the various blocks scattered around the area. At first, the robot will scan the environment using its LiDAR sensor to give an accurate mapping of the area. Using this map, the robot will then navigate to each block and record the letters and numbers for further processing. After having an accurate understanding of everything involved, the robot will then proceed to sort the various blocks. Some issues that may arise include: insufficient motor power to push the children’s blocks into the correct places, hardware issues inherent to the TurtleBot3 Burger, and inaccurate mappings using the onboard LiDAR. Rigorous testing will allow us to detect and prevent these issues early on in the assignment.

## 

## Description

Using the TurtleBot3 Burger equipped with a Raspberry Pi 4, a small camera, Light Detection and Ranging (LiDAR), Robot Operating System (ROS), and two drive motors with encoders, we will program the mobile robot to use computer vision to identify markings on children’s blocks and move them into a specified order while avoiding potential obstacles. The blocks to be moved will be placed in a flat, enclosed area, with no other obstructions until the robot can successfully arrange the blocks. Using a combination of LiDAR and camera feeds, the robot will navigate around the enclosed area and map the locations of the various blocks and store the identity of the block in an array. After knowing the location of the target blocks, the bot can then calculate how to move without bumping into any blocks unintentionally. Finally, after collecting all needed information, the robot will compute a safe path in which to move the blocks in the correct, specified order before moving to accomplish this goal. While moving along the predetermined routes, the LiDAR will remain active in detecting the current coordinates of the bot and any large obstructions that need to be avoided. With smaller obstructions that may not be detectable by the LiDAR, the bot will be able to check the change in measured coordinates from LiDAR against the estimated change in coordinates using the speed setting of the TurtleBot in ROS and checking if the bot is moving in real space or if it is stuck. If the bot is stuck, the bot should back away from the block and attempt to move it from another angle of attack. Once the blocks are in the specified order the bot should back away, stop all motion, and await further input.

To construct the robot, the base waffle plate was first fitted with wheels as well as the lithium-polymer battery, before being secured by a similarly sized plate directly above the base. On this second waffle plate, the motor control circuit board is secured before connecting wires to the battery and wheel motors, a third waffle plate is then placed on top of the second. The third waffle plate houses the Raspberry Pi 4, which is then connected to the motor control board via two different cables, and to the USB2LDS chip via a third cable. A fourth and final waffle board is placed on top of the third plate, this final plate houses the LiDAR which is connected to the USB2LDS chip. Aside from the physical construction of the mobile robot, we also have to install the Ubuntu Operating System (OS) as well as the ROS Noetic Integrated Development Environment (IDE) into the 16 GB SD card which will be slotted into the Raspberry Pi 4 computer before we can begin using the robot. Installing Ubuntu and ROS is not only required for the Raspberry Pi 4 to work, but we also need to install it on a PC to interact with the robot. In order to install Ubuntu on our computers, we first installed Oracle’s VirtualBox, a virtual machine that would allow the use of a different operating system on top of the one already running our computers. After getting the VirtualBox working, we installed an instance of Ubuntu 20.04 as the latest version, 22.04, will not work with the ROS version we have chosen. Once the initialization of Ubuntu completed, we began the process of installing ROS Noetic by configuring our Ubuntu repositories to allow for restricted processes. Next, we set up our computers to accept software from ‘packages.ros.org’, this allows us to set up our ‘sources.list’. In addition to setting up the ‘sources.list’, we also needed to set up our keys, which included installing curl if it was not already installed. The last requirement before picking which ROS to install was to make sure the Debian package index is up to date. Once everything is up to date, we can choose how much of ROS to install. Our team chose the full installation of ROS as it came with additional libraries and features that may be needed in our development process. The last step of this process was to set up the environment by using the ‘source /opt/ros/noetic/setup.bash’ command in each bash terminal we used. At this point, ROS Noetic should be fully operational, and ready to be linked up with the Raspberry Pi 4 attached to the robot.

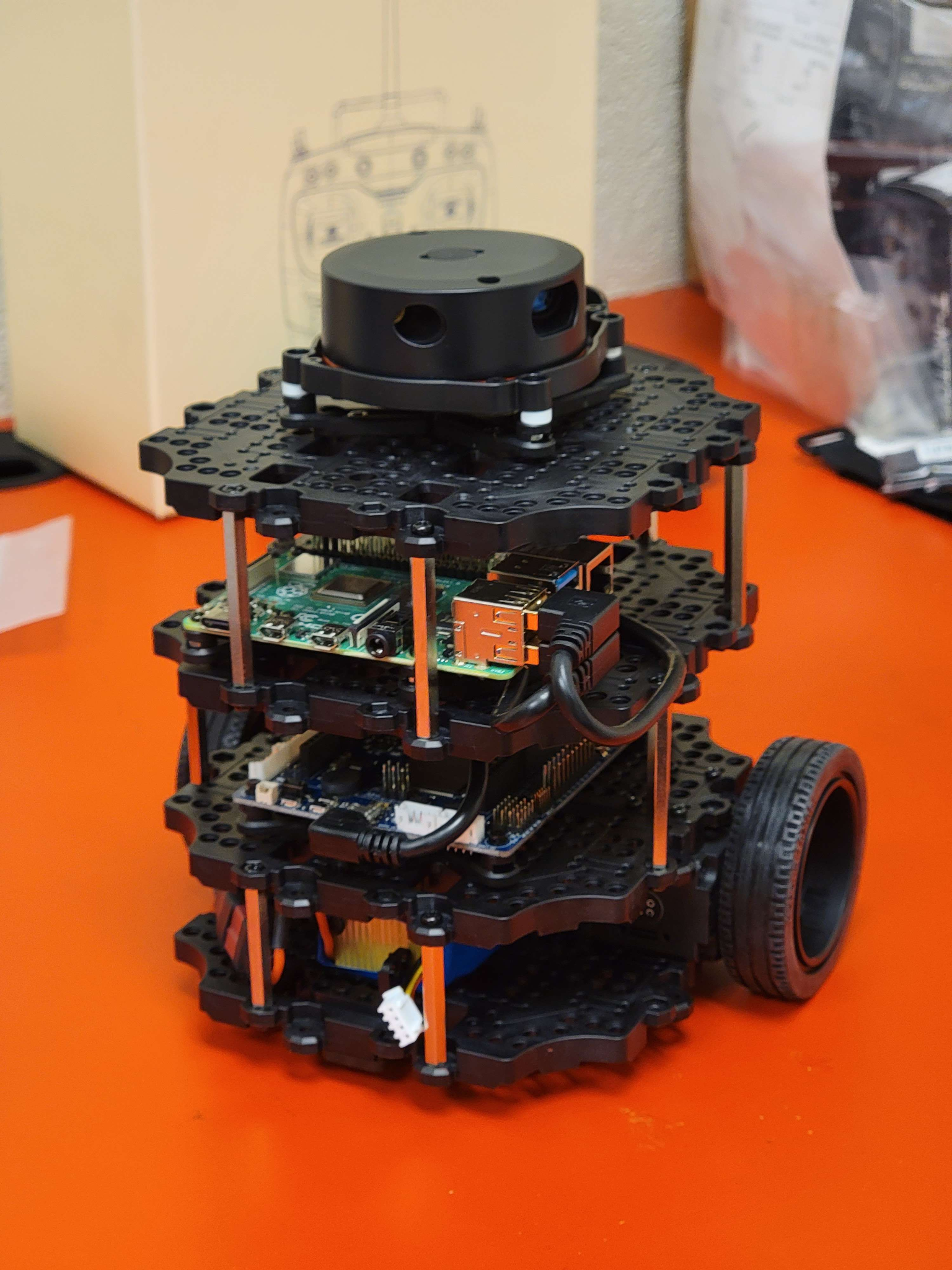


Figure 1. This is an image of the TurtleBot3 in its default configuration with the LiDAR at the top and the Raspberry Pi on layer 3.

Two of the software that this project will depend on are OpenCV and SLAM. OpenCV will receive its images via a camera mounted on the front of the TurtleBot3 Burger. This will allow the use of computer vision algorithms to identify the blocks needing to be arranged. The LiDAR sensor on the top of the robot will be responsible for feeding sensory information to the SLAM node in ROS, the information will be processed until a working map of the surrounding area can be made. To delve deeper, an array of tuples will be used to keep track of the blocks using a grid space. The grid space will be constructed using the input from the camera and LiDAR sensor to create identifiable objects on the working map. Each object will receive its own coordinates and additional information including images and dimensions will be attached to said coordinates.

Some of the modifications planned for the TurtleBot3 may require significant changes to the structure of the robot itself. One change to be made would be moving the LiDAR sensor lower to the ground so that the blocks can be properly sensed. Moving the LiDAR would allow for a more accurate mapping of the environment and remove our dependence on the camera for avoiding obstacles. While this idea would introduce several blind spots in the LiDAR’s sensor range, this can be accounted for with proper pathing of the robot. Another potential modification in the event the mobile robot is unable to properly move the blocks will be to install a 3d printed scoop. This scoop will ressemble a bulldozer front, which will be used to pick up blocks and dump them in the correct order. The TurtleBot3’s design allows for modifications of this level, it would be a simple alteration due to the modularity of the base plates and extra parts supplied from the robot kit. The third planned modification will be to attach a camera to one of the supporting pillars of the mobile robot. It will be attached via a padded clip and secured with zip ties to prevent undesired camera movement due to slip or vibrations.

## Feature List

### Features to be completed by the end-of-semester:

* Puts children’s blocks in specified order
* Use LiDAR to map environment and record blocks

### Features to be completed if there is time:

* Obstacle avoidance

### Features we want to implement but cannot be completed this semester:

* Getting the robot to stack blocks
* Making the burger move quickly

## Technology

* Platform: Linux
* OS: Ubuntu 20.04
* IDE: ROS Noetic
* Programming Languages: Python
* 3rd party libraries and tools: OpenCV, SciPy
* Server Software: The Construct
* Communication Software: Discord

## Server Information

We have accounts made on The Construct which will run our software.

## Data Sources

* https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
* https://youtube.com
* https://get-help.robotigniteacademy.com
* http://wiki.ros.org/Installation/Ubuntu

## Team Member’s Background

### Ethen

Approximately four months using ROS in an applied project where software was created to make a drone fly in designated patterns based on shapes that are scanned through a camera. Additionally, this person helped assemble the TurtleBot3 Burger which will be used in this project.

Main responsibilities include: writing software, maintenance of the robot, testing software.

### Charles

### Approximately four months using ROS in an applied project where software was created to land a drone on an elevated platform. Additionally, this person assembled the TurtleBot3 Burger which will be used in this project.

Main responsibilities include: installing any additional modifications to the robot, writing software, testing hardware.

### Calab

Approximately four months using ROS in an applied project where software was created to land a drone on an elevated platform.

Main responsibilities include: turning in assignments, keeping members on task, writing software and documentation.

## Dependencies, Limitations, and Risks

### Dependencies

* ROS issues
* Construct server issues

Both dependencies can be mitigated by working ahead of time, and being aware of how long we spend in the construct (the construct limits daily use to eight hours).

### Limitations

* Battery life of robot
* Unfamiliarity with hardware may present issues

Battery life can be resolved easily as we have access to two robots, we can also make sure the battery is always charged after use.

Unfamiliarity with hardware will be resolved by research and by asking others who may know more about the subject.

### Risks

* Robot assembled incorrectly
* Inherently faulty hardware
* The robot uses a lithium polymer battery, which may explode…violently

If the robot is assembled incorrectly we can always reassemble it or ask for assistance from an engineering professor.

If any faulty hardware is detected we will have to order new parts, but more likely we will be stealing Adam and Austin's robot as new parts may not arrive in time.

Weekly checkups on the hardware will mitigate the chance of any potential explosions. In the event of an explosion, participants of the project will duck and cover, disregarding the potential loss of equipment in favor of keeping all limbs intact.

## Timeline

### Ethen

Week 1 - Installing Ubuntu and ROS

Week 2 - Installing Ubuntu and ROS as well as getting the robot operational

Week 3 - Setup Camera

Week 4 - Resolve any additional hardware issues

Week 5 - Build 3d printer

Week 6 - Build 3d printer

Week 7 - Build 3d printer scoop model and other further modifications

Week 8 - Object detection in openCV with camera

Week 9 - Object detection in openCV with camera

Week 10 - Lots of testing for potential bugs and hardware issues

Week 11 - Lots of testing for potential bugs and hardware issues

Week 12 - Create presentation

Week 13 - Modify presentation as needed

Week 14 - Testing and improving all features

Week 15 - Testing and improving all features

Week 16 - Testing and improving all features

### Charles

Week 1 - Installing Ubuntu and ROS

Week 2 - Installing Ubuntu and ROS as well as getting the robot operational

Week 3 - Setup Camera

Week 4 - Complete hardware modifications

Week 5 - Build 3d printer

Week 6 - Build 3d printer

Week 7 - Install 3d printed scoop

Week 8 - Create array representation where blocks are in LiDAR map

Week 9 - Create array representation where blocks are in LiDAR map

Week 10 - Lots of testing for potential bugs and hardware issues

Week 11 - Lots of testing for potential bugs and hardware issues

Week 12 - Look over presentation to present

Week 13 - Give more feedback on where presentation can be improved

Week 14 - Testing and improving all features

Week 15 - Testing and improving all features

Week 16 - Testing and improving all features

### Calab

Week 1 - Installing Ubuntu and ROS

Week 2 - Installing Ubuntu and ROS as well as getting the robot operational

Week 3 - Begin researching LiDAR sensor and mapping algorithms

Week 4 - Continue researching LiDAR mapping algorithms

Week 5 - Continue researching LiDAR mapping algorithms

Week 6 - Build 3d printer

Week 7 - Help build 3d printer scoop model and other further modifications

Week 8 - Implement additional features if time permits

Week 9 - Debug any added features

Week 10 - Lots of testing for potential bugs and hardware issues

Week 11 - Lots of testing for potential bugs and hardware issues

Week 12 - Look over presentation to present

Week 13 - Give feedback on where presentation can be improved

Week 14 - Testing and improving all features

Week 15 - Testing and improving all features

Week 16 - Testing and improving all features