Workspace Wizard

Mary Esenther, CE, Aryaman Ghura, CS+CE, Kavya Manchanda, CE, Ishaan Salian, CE[1]

Abstract— **In an increasingly fast-paced academic environment, students often struggle with time and project management. A messy workspace not only creates stress but also decreases the overall productivity of an individual. We propose to develop a personal solution in the form of a robot that can clean your workspace and act as a smart assistant. The Workspace Wizard is built around a distributed architecture consisting of an overhead computer for processing and a robotic body equipped with a plow arm for object manipulation. This project aims to enhance workspace organization without human intervention, ultimately reducing stress and improving productivity.**

I. INTRODUCTION

Knolling as a concept was introduced in 1987 by a janitor named Andrew Kromelow who worked at Frank Gehry's furniture-making shop. Each day, Kromelow would go to organize the tools in the shop that were left out from the day. He would arrange them on a flat surface, so they were at 90 degrees to each other. He called this knolling, since it reminded him of Florence Knoll's angular furniture pieces [1]. In our work-focused society, worker wellbeing is often placed below the main priority of productivity. Students and employees alike face incredible pressure to deliver promising results to higher-ups, often facing mental health issues in these efforts. Such issues, if left unaddressed, may eventually lead to burnout and depression. Our product, Workspace Wizard, seeks to alleviate these issues and reduce pressure by providing workers with the satisfaction of an organized desk after they return from breaks.

[1] Mary Esenther (mesenther@umass.edu)
Aryaman Ghura (aghura@umass.edu)
Kavya Manchanda (kavyamanchan@umass.edu)
Ishaan Salian (isalian@umass.edu)

A. Significance

Anxiety has always been a prevalent issue among students and workers alike. Spurred on by deadlines, toxic workplaces, loneliness, and more, it has been ranked the number one issue among the American workforce [2]. Along with these typical workplace-present stressors, there are several additional contributing factors: the COVID-19 pandemic is one example. But among these is a factor that often flies under the radar when observing this problem: messy workspaces.

Such a factor may seem trivial in the face of widespread economic troubles. However, its psychological effects can provoke a downward spiral in mental health, including emotional exhaustion, less motivation, and, of course, anxiety [3]. The reason for this lies in the stigma that only disorganized people have disorganized desks, a belief shared by 57% of Americans [4]. Not only might this trigger self-doubt and demotivation among workers upon seeing that their desk is messy, but it may also trigger social and professional consequences. Employers and coworkers are more likely than not to assume that someone is naturally messy if they see that their desk is messy, which may affect their level of respect for that person and, subsequently, their treatment of them. And given that a messy desk decreases motivation and increases emotional exhaustion, such a worker is less likely to clean up their desk, especially if they believe it to be a menial issue. This may produce a vicious cycle of social judgment and inaction.

Countless proposals have been made to battle the anxiety epidemic head-on: therapy treatments, fewer mandated working hours, and antidepressant distribution, to name a few. But perhaps it is time to address this issue by taking away the additive stressors making workers' lives harder than they need to be.

B. Context and Survey of Similar Solutions

Similar solutions, such as the Vector Robot and the Roomba Vacuum, do exist but fall short in cost and application of knolling. Anki's Vector Robot is small and compact, suitable for a desk setup. It has an in-built camera and is capable of self-charging, though it is not autonomous. It also has an optional integration with Amazon Alexa [5]. However, as of April 2019, Anki is

bankrupt, and features that were computed on the AWS cloud are now non-functional. It is only available for reselling or on Amazon for the heavy price of $879 [5]. It also does not organize your workspace. The Roomba Robot Vacuum is not made for a desk, but for an apartment floor. It does not have object or image recognition and only avoids obstacles. It has advanced navigation algorithms and automatically returns to its charging dock when the battery is low [8].

Lastly, among prototypes, Knolling Bot by Yuhang Hu et al. proposes a knolling system with a robotic arm and an RGB camera to organize items of various sizes and quantities on a table [7]. Their system can aesthetically organize items without the need for human supervision. However, it uses a transformer neural network model and Gaussian Mixture Models (GMM), which require significant computational resources, especially when scaling to large or real-time tasks. Since the arm can only reach a certain distance and the system is not portable, its usage is limited to certain desk sizes.

Our solution incorporates elegant algorithms for navigation as well as its own clean battery management system that allows for pass-through charging (charging while organizing the desk). Since all computation will be done by the overhead computer, there will be few compromises made in the size of the robot itself, making it ideal for a desk application. Moreover, we will have the system running offline, i.e., without needing Wi-Fi, so that all data is on-device and secure, in contrast to all the solutions mentioned above.

C. Societal Impacts

The target audience of our product consists of anyone with a desk that they regularly work on. This primarily includes office workers and students. With this product, the most immediate impacts on this audience are positive: neat organization of their workspaces may induce aesthetic appreciation, a sense of cleanliness, renewed motivation, and perhaps a fascination with robotic capabilities. If these are achieved, the clients are likely to experience less stress in their day-to-day life.

D. Goals, Specifications, and Testing Plan

The specifications, along with the corresponding testing plan for the Workspace Wizard, are shown below in Table 1.

Specification	Corresponding Testing Plan
Detect human absence	Robot movement starts only after 2 minutes of human absence in at least 98 out of 100 trials (reduced from 20 minutes for demo purposes)
Map viable paths for object placement	The robot should successfully navigate around obstacles to find viable paths, if one exists, without getting stuck.
Object Orientation	Orient objects such that their bounding boxes consume the least area.
Identify objects to avoid	The system should identify pre-defined objects to avoid, like coffee mugs and laptops, which may be dangerous for the robot to knoll.
Recognize table boundaries	No objects, nor the robot, should exceed the table boundaries. If either one of the ArUco markers is missing, or the robot is not within the space defined by the markers, the program will identify this and warn the user
Run for 30±5 minutes on one full charge	Run the robot for 30 minutes while monitoring battery levels.
Knoll up to 10 (varying) objects	The robot should knoll all 10 objects accurately in at least 95% of trials, if a knolling path exists.

Table 1: Specifications and Testing Plan

II. DESIGN

A. Overview

Workspace Wizard is composed of two distributed sub-systems, as seen in Figure 1, which communicate with one another to collaboratively form the complete design. This includes the overhead computer and the robot. The Overhead Computer is responsible for monitoring the desk, detecting objects, finding knolled arrangements for the detected objects, executing path planning to command the robot's actions for knolling, and performing course correction to

accommodate the hardware's inaccuracies. Meanwhile, the robot is responsible for pushing objects on the desk with a physical plow to knoll them following the commands received from the Overhead Computer. These systems are further explained in detail in the following sections.

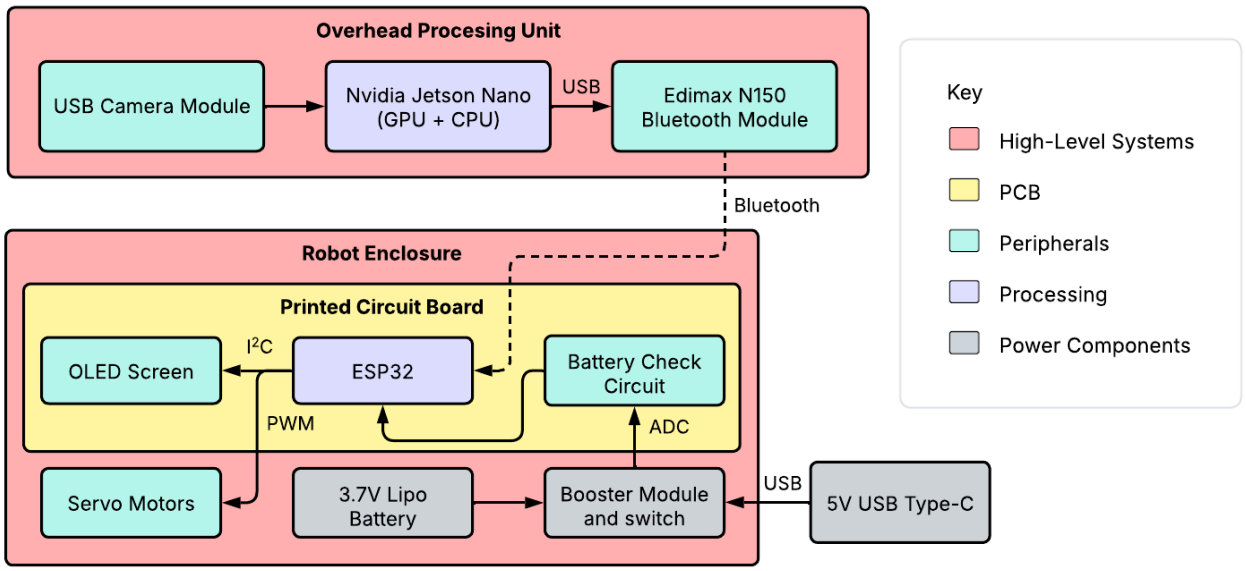


Figure 1: Hardware Diagram of all sub-systems

B. Overhead Computer and Software

The overhead computer, an Nvidia Jetson Nano, is responsible for monitoring the desk, running algorithms to detect and map objects to their final locations, transmitting instructional commands to the robot, and performing course correction when required. Input is received via a USB Camera module connected to the Nvidia Jetson, which records the desk in real-time, allowing the Jetson to detect objects and find a suitable path for their final locations. It then communicates with the robot by transmitting navigational instructions via Bluetooth, enabled via the Edimax N150 Bluetooth module. In case the robot moves in an unexpected manner, the Jetson accommodates the change in the following movement. The software being executed on the Jetson makes use of CUDA, a parallel computing platform and programming model that allows developers to leverage the power of NVIDIA GPUs for general-purpose computing, enabling faster processing in applications like deep learning, image processing, and scientific simulations. Thus, it enables us to execute computer vision workloads and path-finding

algorithms efficiently and successfully. A flowchart for the toolchain running on the Nvidia Jetson is illustrated in Figure 2.

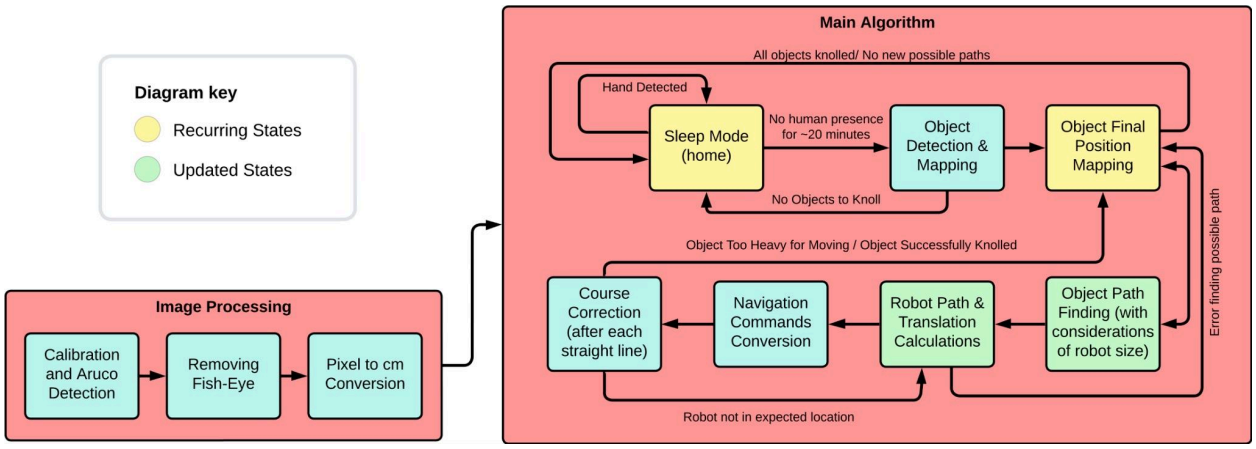


Figure 2: Jetson toolchain

The Jetson is capable of tracking a series of fiducial markers that represent the ends of the table as well as the robot itself. The connected USB camera is first calibrated using a 12x8 chessboard. Once a certain number of pictures are taken with this chessboard visible to the camera, the Jetson runs a program to remove the “fish-eye” effect of the camera based on the actual known measurements of the chessboard (2.1cm per square). Once this effect is removed, it becomes possible for the robot’s fiducial marker to be tracked accurately according to a real-world distance, and a direct mapping between the camera distance and actual distance can be made. Now, when the robot moves, the camera knows exactly where it is in relation to the desk. The software also keeps track of a “movement unit” variable - this stores the actual distance (in cm) that the robot moves when it receives a single command. The program sends the edited image of the desk to the path planning algorithm, which determines where the robot must go to knoll an object, and then this plan is followed through by converting each distance in the path to movement units associated with a particular direction. These are then converted into actual commands transmitted to the robot’s ESP32 board using the Bluetooth Low Energy (BLE) protocol.

Once the input is calibrated and cropped, the Jetson constantly monitors the live feed for the presence of human hands using the custom-trained instance segmentation modules, trained on

YOLOv8. If no hands have been detected for 20 minutes, the system assumes the user is absent and starts the knolling process. This is illustrated in Figure 3, given below.

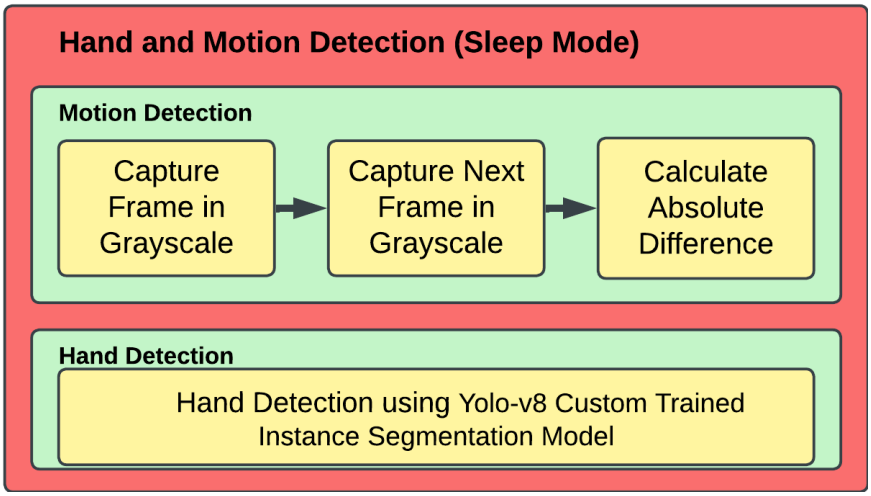


Figure 3: Hand and Motion Detection flowchart for "Sleep Mode"

The Jetson then performs object detection on the processed input image by using opencv2 and a custom YOLO8 segmentation model [6]. This custom model provides details about the objects' bounding boxes, point-to-point contours, and object types. This information is used to distinguish between objects that must be knolled and those that might be dangerous for the robot to propel, eg, coffee mugs. Thus, a predefined list of objects, such as cups and laptops, is avoided by the algorithm either because they are dangerous or have been tested to be too heavy for the robot to propel. An example of detected objects is illustrated in Figure 4, where dangerous objects are displayed in yellow (one knife), and the objects to be knolled are displayed in red (three scissors).

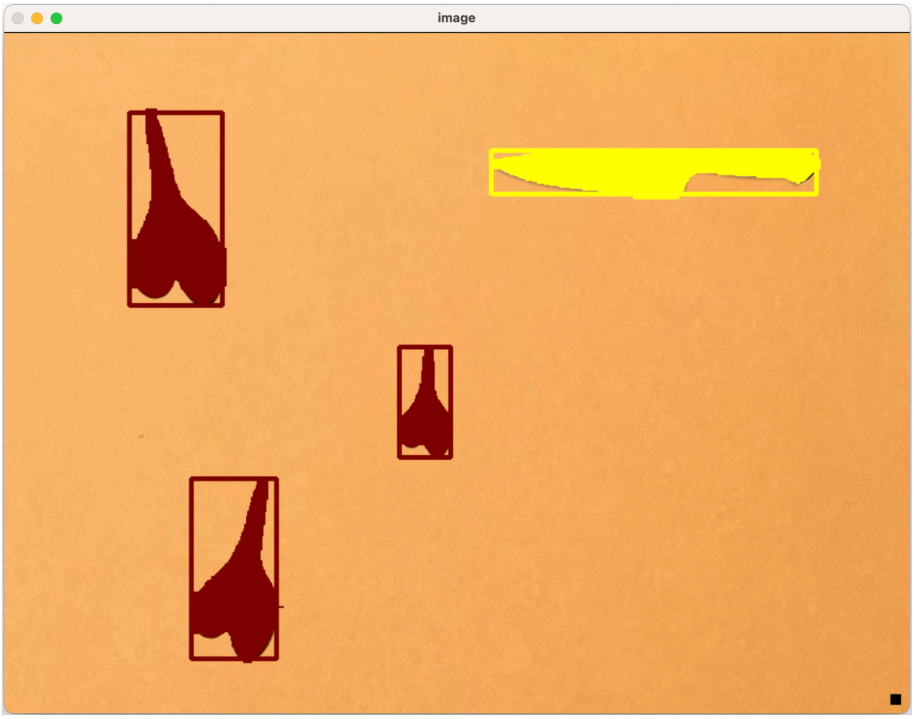


Figure 4: Object detection and distinction. Red objects indicate objects to be knolled, yellow objects indicate objects to be avoided (a knife for demonstration purposes), and black dot represents the robot.

Following the execution of the object detection algorithms, the code runs a knolling algorithm that attempts to find a final knolled location for each object, one at a time. The process starts with an object closest to one of the table's four corners, and attempts to find a location that may fit the detected objects' bounding box such that it is as close as possible to the left/right table edge and does not overlap with the positions of the other objects present. The final location of the first object to be knolled is displayed in Figure 5, where one of the scissors from Figure 4 was knolled to the top left corner as illustrated by the green bounding box. Once a location is finalized, a custom A-star algorithm is executed that attempts to find the optimal path for the object from its current location to the target location while avoiding any obstructions that might appear on the way. During its execution, the algorithm also ensures that the robot can fit behind the object to properly propel it without colliding with any obstructions. This is again illustrated in Figure 5, where the blue path is the object path from the initial to final location of the object's center, and the black path is the robot's path to propel the object from its initial to final location. In such a way, the custom A-star provides a viable path for both the object's movement as well

as the robot's movement so that it may propel the object through the finalized path. Since A-star is a complete algorithm, ie, it always finds a path if one exists, a failure indicates the path does not exist and the algorithm moves on to knoll a different object. Once all other objects have been knolled, the code attempts to find a new final location and path for the failed object under the impression that movement of other objects may now have cleared a path for the previously failed object.

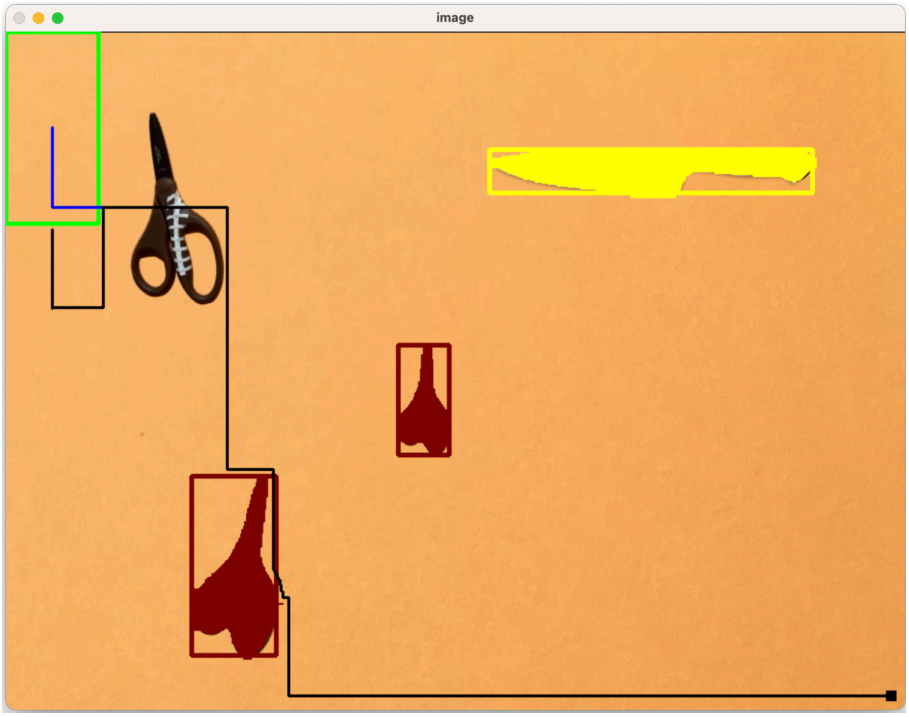


Figure 5: Knolled location and path planning commands. Red objects indicate objects to be knolled, yellow objects indicate objects to be avoided, and black dot represents the robot. Green bounding box represents the final location for the top-left pair of scissors, the blue line indicates the object path, and black line indicates the robot path.

If the custom A-star algorithm successfully outputs a path, the robot path is translated into navigational commands by converting the pixel distances to centimeters using the conversion factor calculated during initialization. These navigational commands are sent to the robot via Bluetooth, as discussed previously. A course correction algorithm checks if the robot is moving in an expected manner during each separate navigational command. For example, if the robot is expected to move 20 steps forward, the algorithm checks the robot's position after half the steps

(10 in this case) and compares the expected location to the actual location. Based on this metric, it accommodates the next half of the instructional command to ensure the robot moves in a reliable manner. Similarly, if the orientation of the robot is not aligned with the edges of the table, it attempts to correct the same within a tolerance of ±5 degrees due to limitations in accuracy of the servo motors. Furthermore, if the robot is notedly stuck and unable to propel a given object, the object in discussion is simply considered an obstruction and ignored by the robot due to weight limitations.

It may be noted that the robot position is easily retrieved using the fiducial marker atop the robot, which is detected by OpenCV2. The robot position and the knolled position for a scissor from the live camera feed are displayed in Figure 6.

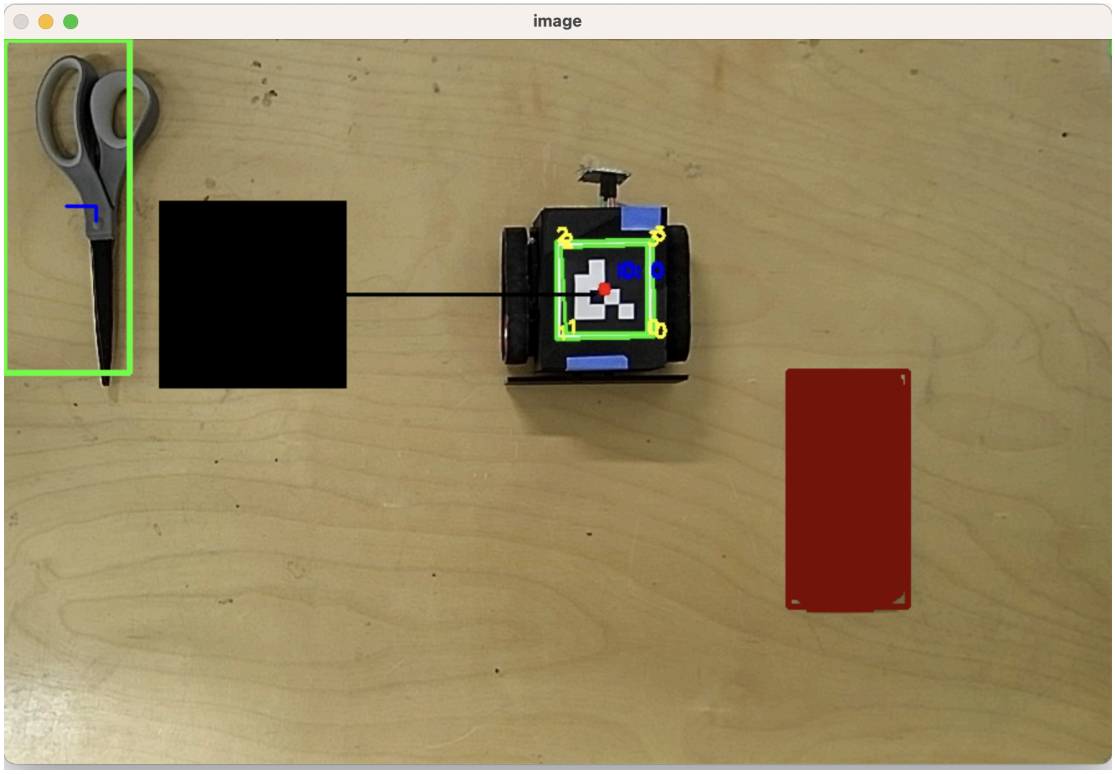


Figure 6: Knolled location and robot position from live feed. Red objects represent objects yet to be knolled, green bounding box visualizes final position for scissors, blue line represents object path, and black line/box represents object path.

C. Robot and Power Management

The robot is designed to knoll the objects found on the desk space using a simple plow mechanism and motorized tracks. The key considerations for the design include compactness, sufficient power, and robot precision to effectively knoll objects. The robot incorporates two servo motors that require 5V each for effective knolling. This voltage is provided using a 3.7-volt Lithium-Polymer battery in conjunction with a boost converter. This converter also functions as both a battery management system (therefore preventing the battery from overheating) as well as a charger. The connection to the battery through this unit is always the source of charging (a USB cable connected to the wall or computer) until this is disconnected, in which case the battery automatically begins to power the motors and ESP32 board. With this setup, the battery management system is capable of pass-through charging; this means that the robot can continue to function while charging. The only cases in which the robot is unable to function are when the booster unit recognizes that the battery is effectively dead, in which case, the system cannot be powered until proper charging, or the current draw is outside of a safe range. This ensures that battery depletion does not affect the safety of the robot nor the user.

III. THE PROTOTYPE

A. Prototype Overview

The Workspace Wizard system comprises two subsystems: the overhead computer and the robot, which together provide workspace monitoring and physical object movement in a desk environment.

The overhead computer is built around an NVIDIA Jetson Nano, which processes a real-time video feed from a connected USB camera. This system detects fiducial markers placed on the table and on the robot, allowing it to calibrate the camera view and convert pixel measurements to real-world distances using a scaling factor based on marker size. A custom-trained YOLOv8 instance segmentation model enables the system to detect and classify objects on the table, distinguishing between objects to be knolled and restricted items (such as mugs or laptops). When no human movement is detected for a specified time (2 minutes during the demo), the system generates final knolled positions and uses a custom A* path planning algorithm to

compute viable routes for the robot to move each object. The pathfinding mechanism was extensively optimized over the semester to reduce execution time from over 15 minutes to under 5 seconds. Robot movement is then controlled via BLE commands, and a basic course correction algorithm is used to check the robot's trajectory and adjust it when necessary.

The robot subsystem consists of a custom 3D-printed body that holds two Parallax 360 degree servo motors, an ESP32-based custom PCB, and a 3.7V 5000mAh LiPo battery. The power system consists of a boost converter module that steps the voltage up to 5V for motor and ESP32 operation. The robot's compact, tracked design (using a Pololu 22T Track Set) allows for smooth motion across various surfaces. The outside of the robot contains an OLED display for battery level and task status. A level shifter on the PCB allows for communication between the ESP32 and servos. The robot achieves over 1.15 hours of continuous runtime on a full charge, meeting the power under load requirement.

Together, these components create an end-to-end system capable of real-time desk monitoring, object classification, pathfinding, and knolling.

B. List of Hardware and Software

List of hardware components used in the prototype:

- 2x Parallax 360° Servo Motors
- 1x DWEII LiPo Boost Converter Charger Protection Module
- 1x 3.7V LiPo Battery
- 1x Computing Platform with GPU (NVIDIA Jetson Nano)
- 1x NVIDIA Jetson-Compatible USB Webcam
- 1x OLED Display for Status Updates

List of software components used in the prototype:

- Python OpenCV library with CUDA enabled, and Aruco library.
- YOLOv8 Instance Segmentation.
- Custom Object Detection and Instance Segmentation trained via Roboflow.

- Python numpy, math, os, copy, and time libraries.

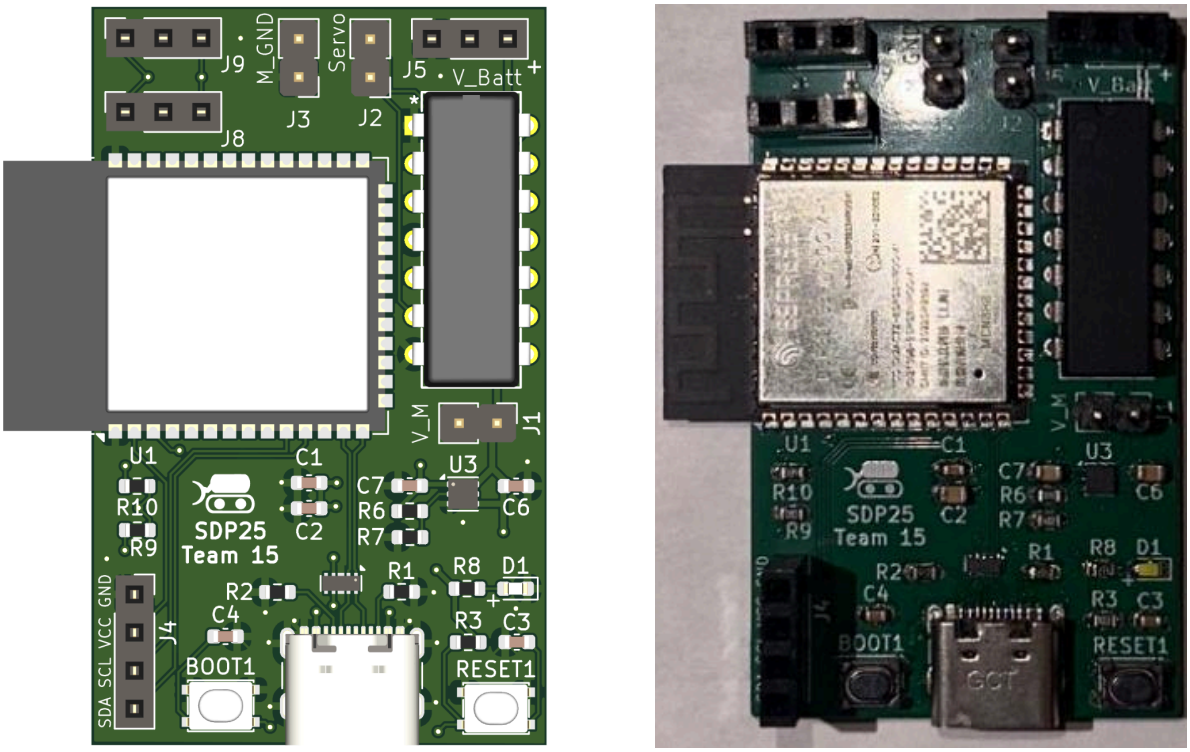
C. Custom Hardware

(a) *Front (while charging)* (b) *Back*

Figure 7: Custom Robot Body

The robot's body is a custom-designed, 3D-printed enclosure optimized to house two servo motors positioned adjacent to each other, the PCB, and the battery to power the robot. The body features a wide front plow for initial testing as seen in Figure 7 (a). There are also two 5mm holes at the back, Figure 7 (b), allowing easier fine-tuning of the zero position (0 degrees) or other specific angles, ensuring precise and accurate movement. Additionally, there are magnets to fit the lid in place, allowing for a satisfying click.

To ensure compatibility with the motors and maintain durability, precise motor mounting slots were integrated. Additionally, 3D-printed wheels were created for the servo to attach better to the track set.

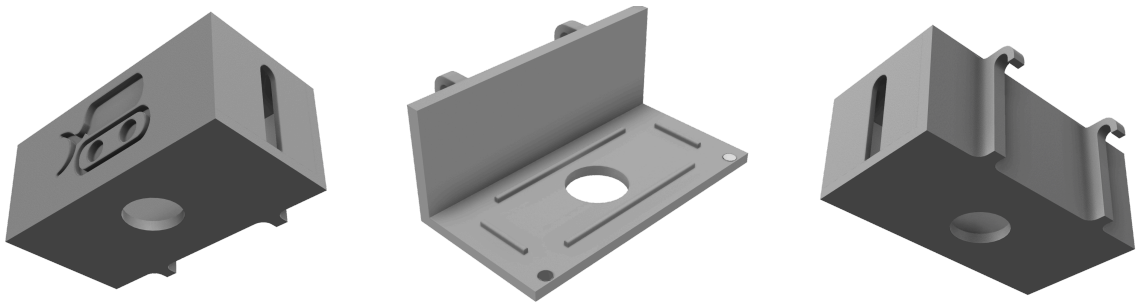


(a) *PCB 3D model* (b) *Populated PCB*

Figure 8: Custom PCB Model

The PCB is designed to integrate and manage the core components of the system efficiently. It includes an ESP32-S3 module, which serves as the central microcontroller, handling communication and control tasks via its built-in Bluetooth capabilities. This enables real-time operation of the Knolling Robot by the Overhead Computer to effectively and successfully complete tasks. The PCB also features power regulation components, ensuring stable voltage delivery to the ESP32 and other peripherals. A USB-C interface is included for programming and power input while debugging, while the onboard status LED provides visual feedback for debugging and system status. Two push-button switches are incorporated for user interaction, allowing manual resets or mode selection as needed. A level shifter is included to shift the 3.3V PWM signal from the ESP32 to 5V for the servo signal.

Additionally, a camera mount was designed to ensure that the webcam is always level with the tabletop. This helps reduce any human error in setting up the camera and ensures smooth calibration. We once again used magnets to ensure a satisfying click-fit.



(a) *Front-Bottom* (b) *Back-Bottom* (c) *Back-Top*

Figure 9: Overhead Camera Enclosure

D. Prototype Functionality

Key Achievements:

- The tracks function as expected, providing effective movement and mobility.
- Bluetooth commands are successfully received, enabling the robot to move and execute tasks in real time.
- The robot demonstrated the ability to push small objects with minimal delay. The delay is almost negligible, making the system responsive to its intended functionality.
- Human detection and motion detection successfully worked for “Sleep Mode.”
- Object classification of “unknollable” objects like coffee mugs, which may be considered dangerous, was demonstrated using the custom-trained YOLO model.
- Path planning and knolling algorithms were a success.
- Path planning was optimized extensively and substantially reduced planning times.
- We were able to run these algorithms as well as BLE on Jetson using the CUDA-optimized OpenCV library.
- The Power Delivery System was finalized; the robot is able to run for over 30 minutes on one full charge even with constant movement.
- All software subsystems were successfully integrated with the Jetson Nano.
- The new PCB design was functional - we incorporated an ESP32 module along with custom pins and a USB-C type charging connector. We now have an SMT PCB and use servo motors, which have their own drivers, instead of TMC2209 motor drivers.

- We implemented a custom BLE-based monitoring system using an ESP32 to track battery voltage every 5 minutes during a full discharge cycle of a 3.7V 5000mAh LiPo under a constant 1.45A load. Readings were transmitted to a computer script via BLE and logged in excel via a Python script. As seen in Figure 10, results revealed a drop from 4.2V to 3.6V over ~70 minutes. The battery management system (BMS) consistently ceased boosting below 3.6V, confirming it as the system's functional minimum voltage. The results can be seen below.

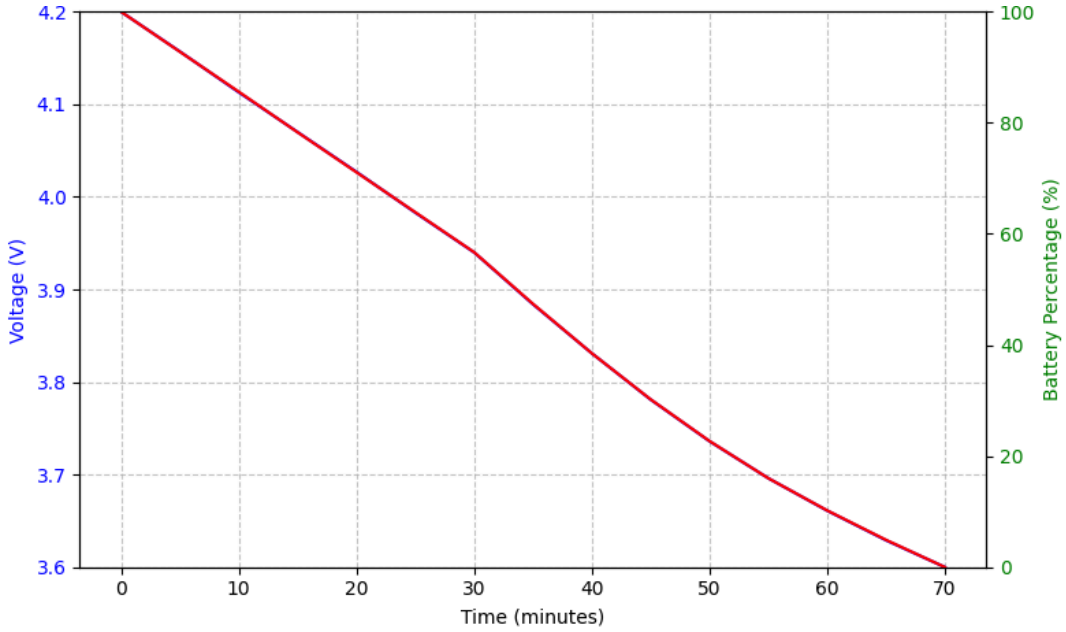


Figure 10: Voltage (V) & Battery Percentage vs. Time (minutes)

Challenges Encountered:

- The custom A* algorithm for path planning originally took a long time (>10 minutes) to find a viable robot and object path for each object. We have optimized the algorithms for both CUDA enabled environments as well as faster processing times. Now the robot is successfully able to find a path for both object and robot within a few seconds.
- The course correction algorithm is flawed and cannot fix every movement of the robot, as it relies on the camera grid having a more accurate representation of the actual table. It also causes significant delays since it stops the robot halfway into each command. As

such, we often removed it for demo purposes.

E. Prototype Performance

We were able to demo the following specifications drawn from Table 1.

Specification	Progress
Detect human absence	
Map viable paths for object placement	
Object Orientation	
Identify objects to avoid	
Recognize table boundaries	
Run for 30±5 minutes on one full charge	
Knoll up to 10 (varying) objects	Small inaccuracies in movement accumulated over time, leading to misalignment after placing just a few objects. There was also a spatial tradeoff between object size and table area, which became increasingly difficult as more space was occupied by objects.

Object Classification:

- We are able to distinguish between fixed objects (e.g., monitors), movable objects (e.g., phones, mice), and objects to avoid (e.g., mugs and liquids). This classification forms the foundation for effective organization and avoidance strategies.

Path Planning and Knolling Finding:

- We have successfully developed the knolling and custom a-star algorithms to find a viable path while avoiding obstacles.

Table Boundary Detection:

- We implemented table boundary detection using Aruco fiducial markers, four of which were placed at the corners of the table boundary to be considered. This ensures neither the objects nor the robot falls off the table boundary.

IV. CONCLUSION

At the conclusion of the semester, the Workspace Wizard project has evolved into a fully integrated system capable of intelligent, autonomous workspace organization. Major technical milestones were achieved across both hardware and software domains, resulting in a demonstrably functional robotic system.

From a hardware perspective, the robot was redesigned to incorporate a refined chassis, improved enclosure, and an updated custom PCB with integrated power management. The robot now features a compact form factor, Type-C charging, an onboard battery management system (BMS), and a reliable magnetic lid. These updates collectively enhanced structural integrity, usability, and runtime performance. The robot achieved a runtime of over 1.15 hours on a single charge during extended testing, validating power delivery and endurance under realistic operating conditions.

On the software side, significant breakthroughs were made. A complete redesign of the path planning algorithm reduced planning time from over 15 minutes to under 5 seconds, enabling near-instantaneous execution of movement commands during live demonstrations. The fiducial marker-based calibration system was streamlined from four markers to two, and a dynamic pixel-to-centimeter conversion mechanism was implemented to remove rigid placement constraints. Object detection and classification, including identification of restricted items, were achieved using a merged YOLO and custom-trained model. Furthermore, a baseline course correction algorithm was developed, enabling dynamic orientation adjustments during object manipulation.

The final system successfully demonstrated 6 out of the 7 formal FPR deliverables:

- Human absence detection via person-tracking over a defined timeout.
- Real-time path planning with obstacle avoidance and object delivery animations.
- Object orientation is aligned with workspace axes during knolling.
- Restricted item identification and avoidance, using classification logic.

- Table boundary recognition through ArUco marker detection.
- Sustained runtime of over 30 minutes on battery power.

While the goal of knolling all 10 unique objects was not fully met, this shortfall stemmed from design tradeoffs, primarily the use of compact servos that slightly limited actuation accuracy, and a limited time in tackling the issues associated with course correction.

In summary, the Workspace Wizard project demonstrates an integrated solution that balances form factor, runtime, and intelligent autonomy. Despite ambitious goals and significant technical complexity, the team delivered a working prototype backed by substantial algorithmic innovation, mechanical refinement, and end-to-end system integration.

V. ACKNOWLEDGMENT

We would like to express our deepest gratitude to our advisor, Professor Duarte, for his invaluable guidance and support throughout this project. His expertise and encouragement have been instrumental in shaping our progress and helping us address our concerns and setbacks.

We would also like to extend our sincere thanks to our evaluators, Professor Soules and Professor Ciesielski, for their constructive feedback during PDR, MDR, CDR, and FPR. Their insights and recommendations have been critical in refining our design and approach.

A special acknowledgment to Professor Soules for graciously offering us a workspace bench in M5.

REFERENCES

[1] Ana, “Always Be Knolling,” *The Well-Appointed Desk*, Nov. 12, 2018. Available: https://www.wellappointeddesk.com/2018/11/always-be-knolling/

[2] B. Robinson, “Anxiety Skyrockets To No. 1 Issue Among American Workers, New Study Shows,” *Forbes*, Mar. 28, 2024. Available: https://www.forbes.com/sites/bryanrobinson/2024/03/27/anxiety-skyrockets-to-no-1-issue-among-american-workers-new-study-shows/

[3] “The Psychological Consequences of Clutter,” *Psychology Today*, 2021. Available: https://www.psychologytoday.com/us/blog/still-procrastinating/202106/the-psychological-consequences-of-clutter

[4] L. Alton, “The Negative Relationship Between a Messy Desk and Productivity,” *Inc.com*, Feb. 16, 2017. Available: https://www.inc.com/larry-alton/waning-productivity-could-a-messy-desk-be-to-blame.html

[5]“Vector 2.0 AI Robot Companion, Smart Home Robot with Alexa Built-in,” *Digital Dream Labs*. Available: https://ddlbots.com/products/vector-robot

[6] Ultralytics, “YOLOv8,” *docs.ultralytics.com*, Nov. 12, 2023. Available: https://docs.ultralytics.com/models/yolov8/

[7] Y. Hu, Z. Zhang, R. Liu, P. Wyder, and H. Lipson, “Knolling bot: A Transformer-based Approach to Organizing a Messy Table,” *arXiv (Cornell University)*, Jan. 2023, doi: https://doi.org/10.48550/arxiv.2310.04566

[8] “Roomba Combo® 10 Max robot + AutoWash dock,” *Irobot.com*, 2024. Available: https://www.irobot.com/en_US/roomba-combo-10-max-robot-with-autowash-dock/X085020.html

APPENDIX

A. Design Alternatives

The first few iterations of the design involved a camera on the Workspace Wizard itself without any Overhead Computer. However, considering the required computational power to run computer vision models, the compactness of the robot itself, its battery power limitations, and its need to plan the final knolled layout, the team decided to move forth with a separate Overhead Computer enabling the execution of heavy computer vision models, unlimited supply of power, significant reduction of robot size, as well as full desk view at all times for planning in advance.

Furthermore, the design initially featured a Raspberry Pi instead of an Nvidia Jetson Nano, but we soon realized the limitations and infeasibility of the Pi running computer vision models due to limited storage space and a slow computational processor.

The plow of the robot also improved over several iterations to allow for a narrow arm that attaches to the plow to allow for potential pulling capabilities by the robot. Furthermore, we considered adding bristles to the bottom of the plow to allow the robot to push thin objects like paper without significantly increasing the friction due to the plow touching the desk surface.

We also moved away from the self-charging mechanism using 15V LiPo's and the stepper motors with their drivers. Since none of the team members had experience with self-charging systems before, it was wise for us to remove that aspect and add a Type-C charging mechanism for the 3.7V LiPo's. That way, we were also able to program the ESP32 module through our laptops when needed during testing. We switched to servo motors, as they come with their drivers - this reduced the robot and the PCB size significantly. We managed to address step correction and orientation via software.

In terms of software, we moved away from the research and code presented in [7] due to a lack of positive results in implementing their functions. The software was reworked from the ground up by developing custom knolling and path finding algorithms to meet our custom needs. Although a variety of different Python modules were tested, including PyBullet, OMPL, and

more, a long delay in their initialization and setup motivated us to develop our own software natively using basic Python libraries.

B. Technical Standards

Our project design incorporates several examples of standardized hardware and software to ensure technical reliability. Below are examples of the standardized components used in our design and the underlying IEEE standards they adhere to:

Bluetooth Communication (BLE)

- The ESP32 module is used for Bluetooth Low Energy (BLE) communication.
- BLE operates under the IEEE 802.15.1 standard, which defines the specifications for wireless personal area networks (WPAN). This standard ensures reliable, low-power communication between devices, making it ideal for our robot's control and sensor data exchange.

PCB Design (Electrical Components)

- Our PCB design follows industry-standard practices for component placement and electrical routing and passes the Design Rules Checking without any warnings.
- The design incorporates components adhering to IEEE 315, which defines graphical symbols for electrical and electronic diagrams. This ensures clarity and consistency in schematic representations.

C. Testing Methods

Robot Abilities:

- Ensured stable BLE communication over various distances and environments. The Jetson sends simple character commands such as 10F (10 steps forward), R (90-degree turn), N (~10-degree turn), etc., to the robot. These steps are taken by the robot in cms using the calibrated pixel-to-cm ratio.

- To evaluate the performance and reliability of the robot, we conducted battery life tests by running the system under typical operating conditions. This involved sending continuous commands from the Jetson via BLE to the ESP32 on our custom PCB, while the robot moved lightweight objects without stalling. This test helped us assess both the endurance of our power system and the reliability of wireless communication. As shown in Figure 10, the robot was able to operate for over 70 minutes on a full charge. After this duration, the battery voltage dropped to around 3.6V. At this point, the Boost Converter within our Battery Management System (BMS) could no longer maintain a stable output. As the servos attempted to draw high current during movement, the voltage would sag below the BMS's minimum input threshold. This caused the BMS to enter a protective shutdown state to prevent damage or instability, effectively turning the system off.

Software Testing:

- Human presence was demonstrated with a camera pointing at the study desk, and the custom-trained model successfully detected the human's hands or fingers in 98 out of 100 trials. This was also demonstrated to be able to detect motion and highlighted all the changes captured by the algorithm in real time.
- Object Detection was demonstrated with a camera pointing at the desk, having three objects: a pair of scissors, a remote, and a mobile phone. The software was able to detect the locations and shapes of two of these objects during the final demonstration. Other tests have also successfully demonstrated the algorithm detecting 5+ objects: pencils, sharpeners, erasers, cups, a cell phone, a laptop, and bottles.
- Object Classification was demonstrated with a camera pointing at the desk, having three objects: a pair of scissors, a remote, and a mobile phone. The program was coded to input the remote as an unsafe object and should therefore have ignored it in the final output image of detected objects. This was also successfully demonstrated in the demo.
- The pathfinding algorithm was demonstrated in conjunction with the knolling algorithm, with the software outputting a path for both the object and the robot while avoiding obstructions in order for the robot to propel the object from its initial to final position. Furthermore, a separate video demonstrated the path-finding algorithm successfully

outputting the path for the robot in a table layout consisting of a pair of pliers, scissors, and a remote.

- The orientation mechanism was demonstrated through a video and live demo of the robot successfully orienting a remote controller initially positioned at an inclined angle on the table.
- Recognition of table boundaries was shown both through a video and a live demo consisting of the calibration technique, as well as the cropped view of the table made possible by the fiducial markers.
- The knolling algorithm and system-level integration were demonstrated through a live demo of the robot successfully knolling a remote controller placed randomly on the table. A similar video with the robot knolling a box of tissues was also provided.

D. Project Expenditures

Table 2 displays the total development cost for the project.

Item No.	**Item Description**	**Quantity**	**Cost (USD)**
1	Rubber tracks	2	22.90
3	Servo Motors[2]	2	0
4	Jetson Nano USB Camera[3]	1	0
5	Nvidia Jetson Nano[4]	1	0
6	MicroSD cards - 64GB	2	19.32
7	PCB + shipping for 1st and 2nd iteration	10 PCBs	156.85
9	PCB components for 1st and 2nd iterations	various	29.90
10	Battery Management System - Type-C Step-up Power Module Boost Converter	2 (Pack of 10)	23.98
11	3.7V LiPo Battery 5000mAh with JST connector	2	28.98

2 Acquired from M5

3 Personal camera of a team member, Mary Esenther

4 The Jetson Nano was acquired from the SDP Lab and the final product will not include the cost of the same.

12	3D printed material	~500 g	0[5]
	Total		**281.93**

Table 2: Project Development Expenditure

Table 3 displays the total production cost for the project.

Item No.	**Item Description**	**Quantity**	**Cost (USD)**
1	Rubber tracks	2	22.90
3	Servo Motors	2	39.90
4	Jetson Nano USB Camera[6]	1	0
5	Nvidia Jetson Nano[7]	1	0
6	MicroSD card - 64GB	1	9.65
7	PCB printing + delivery - final iteration	1	31.30
9	PCB components for the final iteration	1	9.63
10	Battery Management System - Type-C Step-up Power Module Boost Converter	2 (Pack of 10)	23.98
11	3.7V LiPo Battery 5000mAh with JST connector	2	28.98
12	3D printed material	~200 g	3.0
	Total		**169.34**

Table 3: Project Development Expenditure

E. *Project Management*

Our team is structured with clear roles assigned to each member:

- Aryaman worked on **developing and troubleshooting the software** for knolling and path finding algorithms, **optimizing** path finding and fiducial tracking, and **integrating** all

5 Acquired from M5

6 Personal camera of a team member, Mary Esenther

7 The Jetson Nano was acquired from the SDP Lab and the final product will not include the cost of the same.

software sub-systems for seamless performance. When progress stalled, he also took over the responsibilities of **configuring the Jetson environment** and **developing the course correction** algorithm to account for basic lapses in hardware accuracy.

- Kavya takes the role of **Budgeting**, **Model Training,** and the initial setup of the Jetson. She installed OpenCV on the Jetson to run human detection models on it with the live feed from the USB camera. She also modified a 3D model of the Jetson case and took over some of the logistical tasks such as room booking and managing communication with M5.
- Ishaan designed and developed a **custom ESP32 dev board PCB**, integrating key hardware components and power management features. He also worked on **mechanical integration**, ensuring all hardware components fit together in the **custom designed 3D-printed enclosure**. Additionally, he implemented **BLE communication on the Jetson**, allowing wireless communication between the two systems. Beyond technical contributions, he played a key role in **managing project deadlines and coordinating communication between advisors and evaluators** to keep the project on track.
- Mary developed and troubleshooted the **power delivery systems** while also integrating **fiducial markers** for boundary detection, removal of the fish-eye effect, and image processing/pixel to cm conversion. She also assisted with the development of the **course correction algorithm** to fix the robot's location.

The team works well overall, with each member bringing unique expertise to the table. Ishaan and Aryaman's prior experience in Mechatronic Design Teams has been particularly helpful, as it provides the foundation for our robot's development.

Collaboration has been a key strength, with team members stepping up to assist each other as needed. For example, Aryaman helped Ishaan troubleshoot ESP32 communication issues and contributed valuable insights for the design process as well as working together on soldering the PCB. Ishaan was able to step in and develop a 3D prototype when there were technical setbacks faced by other members. Mary collaborated with Ishaan extensively to develop a successful power delivery that would suit our needs. Furthermore, Ishaan and Mary worked together on the CDR demo showcasing automatic command following. Aryaman and Ishaan also helped with installing the necessary libraries on the Jetson, managing to run all existing code on the Jetson.

The team has faced multiple challenges since the inception of the project, especially finding a common time for working collaboratively, given each member's hectic calendar. However, we have addressed this often and have mostly been able to find suitable times during the weekends. Communication has been established on WhatsApp and Discord, and has been one of the strong points of the team. The sense of teamliness was extensively observed during CDR week when we were facing various challenges and met with multiple setbacks in all aspects of the project. During this week, almost every member worked long hours and was seen supporting one another in the SDP lab.

F. Beyond the Classroom

Mary

"This year, I learned a lot about power system design as well as software detection of fiducial markers. The decision to change the motors to servos reduced the voltage of the system significantly - from 15V to 5V. Without a need for such high voltages, I worked on simplifying the power delivery as much as possible using any parts I could find - this led me to a solution involving a low-power LiPo and boost/charger module.

Once that was finished, I took up the part of fiducial tracking. Thankfully this was not too difficult - there are plenty of resources and libraries online designed for fiducial detection, and it did not take long to find out that OpenCV had dedicated functions for it.

Working on the course correction algorithm was more of a struggle. Since there were fewer resources available for such a specific application, this involved more collaboration with Aryaman and creativity in developing our existing code. Even though the result was not as efficient as we wanted it to be, I still think that the learning experience taught me how to think critically in regards to code."

Ishaan

"Initially, I focused on PCB and hardware design, but I had to take over the 3D modeling process midway, which was a new challenge. Though I'd never designed an entire project from scratch, this experience came as a fun challenge for me. I learned to approach complex

workflows systematically and adapt to issues beyond computer engineering principles.

Beyond designing the hardware layout, I developed a fully custom ESP32 dev board PCB from scratch, incorporating documentation guidelines and custom components. The design went through multiple iterations, each informed by testing and early-stage failures. I quickly realized that chasing perfection required shifting from just meeting functionality to prioritizing robustness, modularity, and design clarity. I realigned my goals to reflect a higher standard for quality and delivered a final design that exceeded our original expectations in both performance and integration.

This was also my first experience with SMT soldering, which I completed alongside Aryaman. Through that, I gained valuable experience handling delicate components and understanding the manufacturing side of prototyping. I also took charge of mechanical integration, ensuring multiple components, both electrical and structural, fit together seamlessly. This was a critical improvement from our MDR stage, where we lacked cohesive mechanical planning.

Setting up BLE communication on the Jetson itself was another critical task I worked on, helping bridge gaps in system functionality. I took on such responsibilities beyond my initial scope, stepping in to complete tasks that were delayed or left unfinished. This flexibility allowed us to stay on track while minimizing bottlenecks in the development timeline.

Internet forums, like GitHub discussions, were invaluable for troubleshooting and finding design insights. Personal connections with seniors and peers guided me in planning and executing non-technical aspects like timelines and resource management. This exposure not only helped me find technical solutions but gave me confidence in navigating ambiguous problems,which I find to be a core skill in real-world hardware development.

I also took up a managing role, ensuring team communication, tracking deadlines, and keeping the project aligned with our goals. This included handling discussions on key matters, such as quality concerns, technical disagreements, and advocating for the team when necessary. My experience reinforced the importance of ownership, cross-domain awareness, and high personal standards when delivering a full-stack hardware solution.”

Aryaman

“My primary contributions this semester centered around the development and integration of

the system software stack, with a strong focus on algorithm design and performance optimization. Early in the semester, I took the lead on developing our custom knolling and path planning algorithms after third-party libraries like PyBullet and OMPL presented extensive installation and dependency challenges on both my M1 Mac and the team's Jetson Nano. These issues, combined with poor compatibility and extended setup times, led me to design both algorithms from scratch using lightweight Python libraries such as NumPy.

The initial implementation of our path planning system required up to 30 minutes to compute a valid trajectory. Through iterative redesign, optimization, and testing, I successfully reduced the computation time to approximately 5 seconds, allowing near-instantaneous robot actuation after command issuance. This significant improvement was demonstrated during the FPR demo, where the robot's immediate response validated the successful deployment of the optimized planning pipeline.

In parallel, I overhauled the camera calibration system, reducing the number of fiducial markers from four to two and implementing a dynamic pixel-to-centimeter conversion mechanism based on marker size. This eliminated the need for strict, fixed-distance marker placement and improved the flexibility and robustness of our visual localization system.

After CDR, I also assumed responsibility for the software setup and successfully deployed all core modules (object detection, marker calibration, path planning, robot control) on the Nvidia Jetson. This effort involved integrating and debugging multiple codebases to ensure reliable, end-to-end functionality within a constrained time frame.

Furthermore, I took over the task of course correction later in the semester. In the time available, I implemented a baseline course correction system, allowing the robot to dynamically adjust orientation while pushing objects. While not fully meeting our original ambitions for robust feedback control, the system enabled consistent object placement and laid the foundation for future refinements.

On the hardware side, I mainly assisted with SMT soldering, submitted 3D print jobs to the Makerspace, and provided hands-on help in resolving unexpected mechanical issues during integration."

Kavya

"Going on from CDR to FPR, my main challenges included perfecting the instance

segmentation model for contours, bounding boxes, and object detection. I ran into many issues during the process, including false positives in identifying objects, misidentification, and errors because of the camera's distance from the bench. Since the images I trained on were singular, zoomed-in versions of the objects, the real-life size was over or underestimated. Other times, items like pens and pencils were seen as the same because of how similar they look. Modifying the dataset was a challenge, and I had to change where I was looking for the pictures. I started with simple Google images and soon progressed to pictures taken on our bench. This made the model robust for real-life as well as our particular application.

I also gained a lot of experience in 3D printing and designing while modifying the case for the Nvidia Jetson. Modeling and editing such a high-level design was challenging. Learning the use of a new OS and computer, such as the Jetson Nan,o was a challenge as well. To install the latest version of Python, opencv2, and other libraries/modules, we had to build them from source, including all the dependencies.

Overall, this project allowed me to explore several areas and interests, including programming, improving code to make it faster for the Jetson environment, 3D modeling and printing, and training models using Machine Learning and plotting their confidence values. there were failures, and I promised to improve on them to get better results.”