

Autonomous Solution for a Quadruped Robot

May 4th, 2022

Final Report

pathVision

Alex Domagala, Emily Hernandez, Jon Perthel, Pranay Singh

Faculty Advisors: Pranav Bhounsule, Matthew Alonso

Work Allocation (for Final Submission):

Alex: Conclusion, Task Allocation and Timeline Update, Final User Manual Update, Editing

Emily: Testing, Appendix update, Engineering Requirements Update

Jonathan: Formatting, Abstract, Final Design-Hardware, Bill of Materials

Pranay: Final Design, Editing

Table of Contents

I. Abstract	5
II. Overview	6
Project Goals	6
Needs Statement	6
Objective Statement	6
Background	6
Research Survey	7
User Survey	10
Marketing Requirements	12
III. Engineering Requirements	14
Objective Tree	16
Engineering Requirements Updates	16
IV. Engineering Design Alternatives	17
Design Alternative 1	18
Design Alternative 2	19
Design Alternative 3	20
Design Alternatives Comparison	20
V. Design Alternative Evaluation Criteria	22
VI. Selection of Design Alternative & Justification	24
VII. Preliminary Design	26
Level 1 Schematic	26
Circuit Design	27
Software Design	28
VIII. Testing	33
Camera Testing	34
Buck Converter Testing	34
Communication Testing	35
Integration Testing	35
Final Testing	36
IX. Final Design	37
Hardware Design	37

Software Design	38
Bill of Materials	39
X. Task Allocation and Timeline	41
Timeline	41
Contributions of Work	43
Alex	43
Emily	44
Jonathan	44
Pranay	45
XI. Lessons Learned	46
Alex	46
Jonathan	46
Emily	46
Pranay	47
XII. Conclusion	48
XIII. References	49
XIV. Appendices	51
Appendix A: Survey Answers	51
Appendix B: IEEE Standards	53
IEEE 1118-1990 - Standard for Microcontroller System Serial Control Bus	53
IEEE 1873-2015 - Robot Map Data Representation for Navigation	53
IEEE P2851 - A Landscape for the Development of Dependable Machines	53
IEEE P7009 - Standard for Fail-Safe Design of Autonomous and Semi-Autonomous Systems	53
Appendix C: Concept Map	54
Appendix D: Pairwise Comparison and Decision Matrix	55
Table D-1: Individual Pairwise Comparison	55
Table D-2: Average Weights of Pairwise Comparisons	56
Table D-3: Rating of Each Design Alternative Relative to Criteria	57
Table D-4: Decision Matrix	57
Appendix E: Datasheets	58
Unitree A1 Robot	58
Jetson Nano	70
DFR0379 - 20W Adjustable DC-DC Buck Converter with Digital Display	72
SWITCH TOGGLE SPDT 5A 120V	74
Intel Realsense	81

Appendix F: User Manual	87
Appendix G: Task Management Document	89

I. Abstract

Autonomous systems are becoming increasingly prevalent and have an expected market growth of ~20% annually through 2027. With advances in technologies such as computer vision and machine learning, autonomous systems can be applied to many different industries including automotive, manufacturing, and agriculture. UIC's Robotics and Motion Laboratory is looking to upgrade their Unitree A1 quadruped robot with an autonomous navigation system to assist with automated testing of the robot. We have designed a computer vision based autonomous system capable of detecting and avoiding objects in the robot's path of travel. This system mounts and interfaces directly with the quadruped robot. Testing showed that we exceeded our goals of identifying static and dynamic obstacles in the robot's path and maneuvering away from those obstacles to prevent collisions. As an added feature, we also implemented an object following algorithm where the robot can be set to follow an object of a certain color.

II. Overview

Project Goals

The project goal is to create an autonomous collision avoidance system for an Unitree A1 quadruped robot. The autonomous system will rely on computer vision and additional sensors to detect obstacles in order to avoid them. The autonomous system should transfer avoidance instructions to the robot control system to prevent collisions during operation. Additionally, the robot should be able to navigate an indoor environment while maintaining obstacle avoidance.

Needs Statement

The Robotics and Motion Laboratory at UIC, run by Professor Pranav A. Bhounsule, is looking to upgrade their Unitree A1 robot (quadruped, dog type robot) with an autonomous system capable of obstacle detection and avoidance. Furthermore, the laboratory requests that the autonomous system will be capable of operating in indoor environments and that it will be able to avoid static objects.

Objective Statement

For the project, the team will use a single board computer with a camera system to provide the necessary data to the autonomous system. The computer will process the image data and return information that can be used by obstacle avoidance algorithms. These algorithms will use the processed data to issue commands to the robot's control system and avoid collisions. The autonomous system may also use other ranging sensors such as LiDAR, in combination with the camera system, to develop enhanced maps of the operating environment.

Background

To properly develop the autonomous system, the team conducted research to better understand the capabilities of current technology and better understand where this system will fit into modern technology. First the team researched existing technologies to explore different types of autonomous systems and different applications. Additionally, the team interviewed the research lab to better understand their needs from the system.

1. Research Survey

For the research survey, the team looked at different autonomous systems and how they are currently used. In addition the team researched quadruped robots and how they can be used, particularly in combination with autonomous systems. Finally, the future of autonomous systems was researched.

Real World Autonomous Solutions

In recent years, there has been a large increase in the production of systems using autonomous technologies. We can see evidence of this across many different industries including agriculture, automotive, manufacturing, shipping, and defense. Autonomous solutions in these industries have led to an increase in safety, production, and even profits in many instances which in turn has led many companies and universities to continue research on creating smarter and more efficient systems.

There are many different approaches to implementing autonomous systems. In the automotive industry, there are many companies conducting research and actively using autonomous solutions. Perhaps one of the greatest challenges for autonomous systems exists in the self-driving space. About 94% of automotive accidents are caused by human error and cause over 35,000 deaths each year[1]. Given the dangers of driving, this has led many different companies to pursue research in autonomous solutions for cars ranging from basic antilock braking systems to advanced autonomous systems like Tesla's Autopilot. Systems such as the latter have consistently shown accident averages about seven times better than vehicles without Autopilot[2]. The Boston Consulting Group comments on the projected timeline in the push toward self-driving cars, "Level 4 self-navigation is fully autonomous with a backup system that can turn the machine off in ostensibly rare, unexpected situations, but no human involvement is required. We expect Level 4 capabilities to be perfected around 2030. When that occurs, we'll see the emergence of mobile machines that are self-driving in specific limited environments, such as room service robots in hotels or last-mile delivery robots[3]." These systems typically use a variety of sensors, including but not limited to, ultrasonic sensors, LiDAR, radar, and cameras. Then, they use the data from these sensors, often in conjunction with a form of artificial intelligence such as neural networks, to control the vehicle and autonomously respond to its surroundings[4]. While many of these systems have been proven to reduce accidents, there is still

a lot of mistrust in their safety as they do not 100% of the time respond to their surroundings properly due to limitations of the technology[5]. Artificial intelligence and computer vision applications are somewhat novel but typically very complex technologies. The main limitations of these systems are the fact that not much research has been done to collect enough data and fully train the artificial intelligence. In systems not using artificial intelligence, there can be limitations in how quickly a computer can process the image data, as well as properly being able to identify objects in the environment.

Autonomous Algorithms

Given image data and a powerful computing system, an important function of an autonomous system is successfully interpreting the data and actually issuing movement commands. This idea of interpreting data leads to the use of algorithms. A common family of algorithms specifically used for autonomous navigation are referred to as “Bug” algorithms[6]. This family of algorithms loosely model path planning on the capabilities of insects[7]. The algorithms assume that an autonomous system knows its position, a goal position, and can locally sense obstacles. The most simple algorithm in this family works as follows: the robot heads towards its goal. If an obstacle is encountered, the robot follows the obstacle until it can proceed towards the goal again. The preceding steps are repeated until the robot successfully reaches its goal. Due to their simplicity, bug algorithms are exposed to a wide range of cases in which their operation would seem illogical. An example of this would be entering an enclosed space and hugging a wall to exit and proceed around the obstacle. In more advanced autonomous systems like self-driving cars, machine learning algorithms are utilized to address more significant tasks that potentially carry safety concerns.

Existing Autonomous Quadruped Robots

Automated quadruped robots have become more prevalent in the commercial world largely due to companies like Boston Dynamics. One of their first designs was called BigDog and was designed to carry a payload through rough terrain that a wheeled robot wasn't capable of traversing[8]. This robot contains over 50 sensors and includes an onboard computer that performs both low level and high level controls. The BigDog robot was mainly manually operated, but it also included a stereo vision system. This system used stereo cameras, LiDAR,

and a computer to run computer vision software. The computer vision was used to map out the terrain and allow BigDog to follow a human leader without having to be manually controlled. Boston Dynamics has also recently developed a new version of the quadruped robot called Spot. This robot is fully autonomous and can maneuver through rough terrain and avoid obstacles. Spot also has the ability to be programmed for the specific needs of a business using the sensors and cameras that are a part of the robot[9]. Also, incorporated into Spot is the ability to detect a disturbance and the ability to determine if it needs to wait to avoid the object or to go around. This is described in Boston Dynamics' patent "Handling gait disturbances with asynchronous timing[10]" filed in July of 2019. Spot is designed to automate routine tasks, inspections, and collect a wide variety of data. Boston Dynamics has also filed a patent in January of 2021 for "Autonomous Map Traversal with Waypoint Matching[11]". This patent outlines a system that processes sensor data to trigger waypoint to waypoint mapping. This patent additionally outlines how a system can be implemented with three-dimensional cameras and LiDAR data. The design also uses cloud data to determine and set waypoints with the sensor data.

Computer Vision and Quadruped Robots

Overall, these features of Spot share some similarities with the pathVision design. However, the pathVision team is implementing autonomous features from scratch for the laboratory at a significantly lower price point. The team plans on initially developing a computer vision system for obstacle avoidance and high-level control. Once this is achieved, the team will be able to further program the robot for specific applications such as navigating through a maze or following an object. Sony has also worked on an autonomous quadruped robot using a similar camera system. Their robot had a primary focus on an entertainment application, which differs from the goals of the pathVision team. Although the end features may be different, Sony uses similar technology and computer vision approaches to the team. This robot uses a camera system for object detection and it uses ultrasonic sensors for depth perception. The vision system on the Sony Quadruped Robot was primarily used to automate testing of the control algorithms in a controlled environment[12]. While Sony may not be a direct competitor, the features they developed for their autonomous testing are similar to the needs of the RML lab for evolving their quadruped robot.

The Future of Autonomous Robots

As shown by the previous research, the prevalence of autonomous robotic systems will only continue to increase with improvements in technology. According to Boston Consulting Group, “As these technologies take hold, we believe many customers will shift from buying core robot systems, such as arm, controller, and end-of-arm tools, to purchasing broader, modular systems comprised of the core as well as edge controllers, machine vision software, and AI for smart and autonomous activities, among other emerging innovations[3].” It is known that the departure from traditional robotics is already well under way. As robots leave controlled environments of factories and warehouses they will have to increasingly rely on the special technologies mentioned above. Integration of robotics with the fields of machine vision and artificial intelligence will begin to bridge the gap that prevents full autonomous operation in dynamic environments. But, applications that require autonomous solutions are still waiting on the development of the supporting technologies.

Once the technological barrier has been surpassed, autonomous systems will be able to be incorporated into full production of a vast array of different applications. To illustrate the impact of autonomous systems, again the automotive industry can be used as a window. According to MarketWatch.com, “The global sale of autonomous vehicles was valued at 1.4 million units in 2019 and is projected to reach 58 million units by 2030, expanding at a CAGR of 40.3%, from 2020 and 2030[13].” Therefore, the metrics show that there is strong growth to be had in the autonomous vehicle industry. It is important to note that this is only one of the many areas that utilize autonomous systems. The total impact of autonomous systems will have a far greater reach. There will not only be economic effects, but also far reaching political and social consequences.

2. User Survey

The users of the senior design project will be individuals associated with the Robotics and Motion Laboratory. This laboratory falls under the Department of Mechanical and Industrial Engineering at the University of Illinois at Chicago. The laboratory develops cutting-edge robotic prototypes and control systems that are pushing forward current state-of-the-art systems.

Since this project is being done in collaboration with the laboratory, the team decided that interviews would be most appropriate in learning more about the project as well as the needs of

its users. Thus, the interview questions were split between two sections. The first grouping addresses why autonomous development is desired. The second focuses on technical details that will support the development of the project. Finally, it is important to note that two individuals were interviewed for this project. First, the team interviewed Professor Pranav A. Bhounsule. The Professor leads the Robotics and Motion Laboratory. The team also interviewed Jonathan Garcia, who works as an undergraduate research assistant in the laboratory and is involved with day-to-day use of the robotic system.

Being a mechanical engineering lab, research is primarily focused on low level control such as robot balancing and movement. The development of an autonomous system would improve the laboratory's capabilities in conducting testing as the robot would have to rely on its own sensor data rather than remote control. This is more representative of systems that would be deployed to the real world. Discussion then shifted to the topic of whether the autonomous system should be solely focused on complementing the capabilities of the Unitree A1 robot or if a more general approach should be taken. The professor advised the team that this type of project has two aspects. First, the cameras/sensors must work with a single board computer to indicate where obstacles are located. This could be incorporated into the development of any general system. It is important to note that how this data is used will depend on the capabilities of the robotic system it is being deployed to. The second aspect to consider is that in order to move forward with any autonomous deployment, work must be done to integrate with the control system of the given robot. Thus, it would make the most sense to keep the primary focus of this project on working with the Unitree A1 robot. Finally, challenges to the development of the autonomous system were discussed. The professor noted that in order to be successful, students need to have time on their side. With an early start, the team can become more familiar with hardware and software components. This allows for the identification of limitations and corner cases that could significantly impact the project. This would put the team in a much better position for the start of ECE 397. Other common challenges that were discussed included system integration, learning system hardware limitations, and debugging. Overall, these topics further emphasized the need for an early start with technical aspects of the project.

Technical discussion began with a focus on hardware. The team learned that the robot has its own battery that can power both an external single board computer as well as an internal microcontroller. The primary approach being considered is to use the single board computer for

imaging processing tasks. After performing the processing, the results would be passed to the internal microcontroller which is responsible for actual control of the robot. Continuing on, the robot has an Inertial Measurement Unit (IMU), position sensing capabilities, and power usage monitoring. It is important to note that there is no on-board GPS system, the robots actions must be based on data received from the environment. Professor Bhounsule explained that when collecting data from the environment, the team is not only limited to cameras. The Unitree A1 robot comes with a LiDAR module that can be easily attached/removed from the system. Other sensors to consider include radar, infrared, and ultrasonic types. Efforts should be made to make sure that the operating environment does not negatively impact sensor performance. During the meeting with Jonathan Garcia, the team had the opportunity to view the capabilities of the robot in-person. The robot had a very high build quality and was able to move through the testing environment with ease. The robot can lower itself towards the ground, but it cannot jump. One concern that was raised was that as the robot moved, it tended to shake. This will have to be considered when working with many different aspects of the system.

The final interview questions centered on the topics related to software. The team's primary concern was finding a manual/documentation that covers general code by which the team could interface with the robot. Jonathan Garcia was able to provide a Github repository that contains essential system code. With the concern addressed, discussion shifted to additional features that should be implemented. Jonathan Garcia and Professor Bhounsule both emphasized the importance of completing the obstacle detection system before any other systems can be added. Once this is completed, the team may develop a system to navigate the robot through a maze. The maze initially will be static, but the maze can also be made more complex to include dynamic obstacles. Additional features such as real time display of robot video, a user interface that allows control over autonomous operations, or object following would only be approached following completion of the initial objectives. Only if the core system is reliable, should the team move forward with additional features.

Marketing Requirements

In order to satisfy the requirements from the lab, the design must comply with the following marketing requirements. It should be noted that the sponsor has stated that the design must use cameras as a part of the system.

1. The system must use cameras.
2. The system must integrate into the onboard power and control system.
3. The system must detect objects blocking its path.
4. The system must navigate around or avoid obstacles.
5. The system must be able to turn on and off so the robot can enter manual control mode.
6. The system must be well documented to support its use within the laboratory environment.

III. Engineering Requirements

The following engineering requirements outline performance measures that can be evaluated to ensure that the needs of the user are identified and met. Each of the engineering requirements addresses at least one of the marketing requirements and has a justification either from the lab or an industry standard.

Marketing Requirements	Engineering Requirements	Justification
1, 3	The vision system must detect 90% of static objects within 0.5 to 5 meters in front of the robot.	The specified distances allow the autonomous system ample time to detect obstacles before coming into contact with them. The accuracy and range values come from technical specifications from Unitree [14]. The distances reflect a room scale environment. (IEEE 1873-2015 , IEEE P2851)
4	The system must navigate around any static obstacles that are detected. If the system cannot avoid an object or is unsure how to proceed, it must cease movement.	Requirement based on functionality requirements set by the laboratory. In addition, this enhances system reliability. (IEEE P2851)
1	The camera system must be able to measure distance to objects at a minimum of .3 meters and up to at least 5 meters away from the center of the robot.	To support object avoidance operations, the robot must be able to measure distances to objects. Furthermore, Unitree's documentation provides the noted distance values for the robot's on-board camera [14].
2, 4	All sensors and hardware must be able to mount to the robot without inhibiting mobility.	Any additional hardware must not interfere with the mechanical operation of the robot. (IEEE P2851)
2, 5	The processing computer for the autonomous system must	The Unitree documentation states that the robot has 4

	be able to connect and communicate with the robot's onboard control system.	USB connections and 2 ethernet connections. This is the sole way for the microcontroller to communicate with the onboard computer [14]. (IEEE 1118-1990)
6	Software must be well documented and accessible from a gitHub repository.	Request from the laboratory. Will assist with future use/maintenance of the system.
5	The system must have a control override that returns the robot to manual control.	Required by laboratory for safety purposes. (IEEE P7009)
2	The system must connect to the onboard power supply and be compatible with 5V, 12V, or 19V voltage supplies.	The robot provides the specified output voltages [14].
5	The system must have a physical kill-switch that cuts power to the robot preventing it from moving any further.	In case of an emergency where the autonomous mode cannot be disabled via software, kill-switch allows for rapid shut-off of the robot. (IEEE P7009)
	Costs for additional hardware must not exceed a value of \$250.00. This does not include additional hardware provided by the lab.	Maximum allotted budget from ECE 396.

The system does not have significant political, economic, social/cultural, environmental, or legal connections therefore we have excluded requirements in these categories. The system will only be used for the laboratory at UIC and will not be reproduced so manufacturability requirements are also not needed.

Objective Tree

The objective tree breaks down the functions of the autonomous system. This is broken down into two parts, the object avoidance system and the object detection system. The object detection system is responsible for identifying objects that are in the path of the robot. The object detection system may also be responsible for finding an alternate route around the object in the path of the robot. The object avoidance system is responsible for sending commands to the robot's microcontroller to adjust speed and change direction. This system will use the object detection system data in order to issue commands to move the robot.

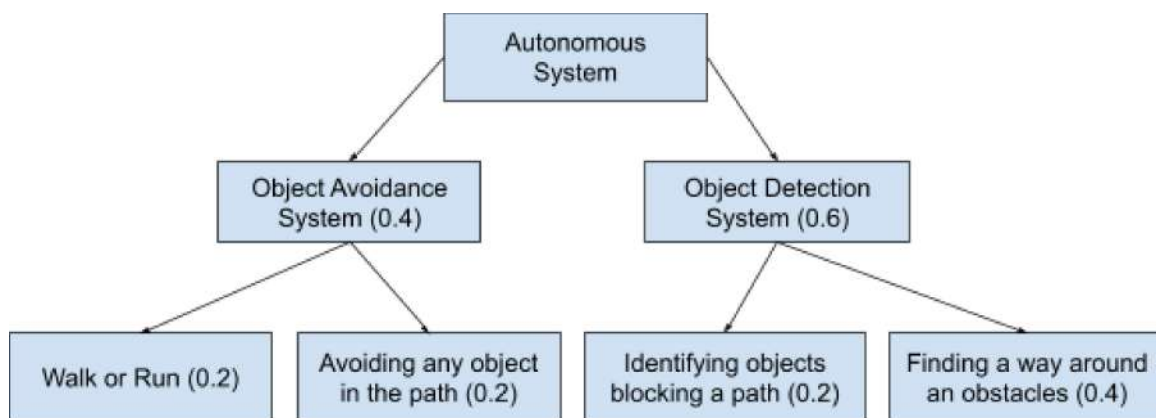


Fig 1. Objective tree

Engineering Requirements Updates

While there are no updates made to the engineering requirements, the team did add functionality to perform obstacle following as well. Obstacle following is not discussed in this paper as it was not originally part of the project but does utilize some of the same features and components. This functionality was requested by the UIC Robotics and Motion Laboratory.

IV. Engineering Design Alternatives

In order to satisfy the engineering requirements, three different design alternatives were proposed. Each of these design alternatives addresses all of the engineering requirements. The level zero table describes the input, output, and functionality of the design, and it applies to all three of the design alternatives.

Table 1. Level Zero Table

Module	Quadruped Autonomous System
Inputs	Environment Data (eg. cameras, LiDAR, ultrasonic), Power (19V), Autonomous/manual mode select, Robot data (eg. IMU data)
Outputs	Movement Commands
Functionality	Use visual data, additional sensors, and robot data to detect objects in the path of the robot. The system will then interpret the input data sources and send motor commands to avoid objects.

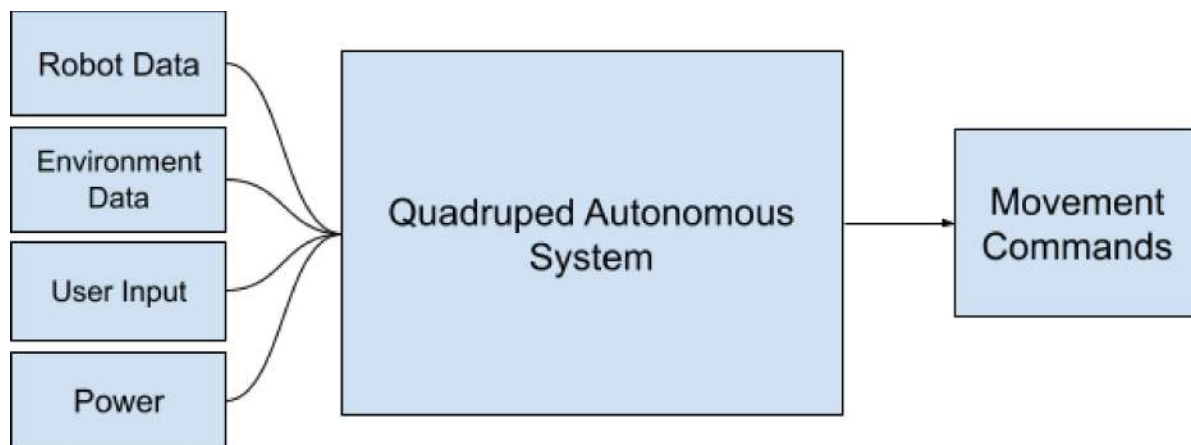


Figure 2: Level 0 Diagram

This level zero diagram is based on the engineering requirements. The diagram applies to all of the design alternatives as the inputs and outputs for the system will remain the same. The inputs include on-board power from the robot to power the autonomous system and user input that includes a kill-switch and input to set the robot into autonomous or manual modes. Next, there is robot data input that consists of IMU data and any other positional data given by the robot. Finally, there is environment data which includes all data collected by the cameras or any other

supplemental sensors. The autonomous system will use the user input, environment data, and robot data to generate movement commands to send to the robot. Below are three different design alternatives that list out options for hardware and software for the system.

Design Alternative 1

For the first design alternative, the robot utilizes a Bug 2 algorithm[11]. This algorithm is also referred to as a “Greedy” algorithm. It works in the following manner: assume that a robot is heading towards its goal in a straight line and note that there is only one straight line defined. If an obstacle is encountered, the robot will depart from the line and follow the obstacle until the line is re-encountered. The robot then resumes its progression toward the goal along the line. This process is repeated until the robot reaches its goal.

The unique hardware used in this alternative is LiDAR, ethernet connection, a physical autonomous mode switch, and a Jetson Nano. Adding LiDAR to the system will increase confidence in the depth perception of objects. LiDAR is known to be a reliable and fast sensor used to detect distances to objects. Additionally, there is a LiDAR module that is specifically intended for use with the robotic system, this allows for ease of use. The only downside to LiDAR is that the data can be difficult to analyze. This design will also rely on a hardwired ethernet connection to communicate movement commands with the robot’s main computer. Ethernet is a communication link that has been used due to its speed, efficiency, and reliability. For user input, this design alternative calls for a physical flip switch to switch between the autonomous and manual mode. This was selected to simplify the implementation, the only downside is that a user wouldn’t be able to switch modes unless they were near the robot. Finally, this system will use a Jetson Nano as the processing unit for the autonomous system. The Jetson Nano is a relatively new computer but is used quite frequently due to its affordability and low power consumption[15]. The Jetson Nano is also designed for object avoidance and image processing systems and has a variety of input ports to accommodate a wide range of sensors and other inputs.

As with the other design alternatives, this system will mostly rely on stereo cameras for depth perception as defined by the lab requirements. For the software, this system will use a combination of C++ and Python. This will be beneficial because while Python has many libraries and resources for computer vision applications, it is a slower language. Being able to use both

languages together will allow for ease of development while still maintaining high processing speeds so the system can quickly respond to its surroundings.

Design Alternative 2

For the second design alternative, the robot will use a bug 1 algorithm[11]. This algorithm is also referred to as an exhaustive search algorithm. It works in the following manner: Given that a robot is heading towards its goal, if it encounters an obstacle, the robot will depart from its approach towards the goal and instead circumnavigate and map out the obstacle. After completing the mapping, the robot will resume its progress towards the goal from the point of closest approach. This algorithm would be utilized for its dependability and consistency. However, the tradeoff is speed and live operation would not seem very intuitive for simple obstacles.

The unique hardware for this design is the ultrasonics sensor, Rock Pi N10, I2C communication, and a remote interface for controlling the autonomous mode. The system will also use stereo cameras and ultrasonic sensors to provide environment data to the single board computer. Ultrasonic sensors will provide extra information regarding depth perception to the autonomous system to increase confidence in the systems sensing. Ultrasonic sensors are used frequently and require little setup and processing but they are only effective in low noise environments and have limited range. This design will use the Rock Pi N10 single board computer which has a CPU, GPU, and Neural processing unit which are designed specifically for deep learning to process inputted sensor data [16]. The Rock Pi N10 is useful because it has interfaces for cameras and has various input/output pins to accommodate the ultrasonic sensor and the Inter-Integrated Circuit (I2C) communication protocol to communicate with the robot's main computer. This communication protocol is native to the Rock Pi N10 and has simple hardware implementations. This design will implement an interface on the robot's remote controller for initiating the autonomous system or for turning it off. This allows the system to be turned off while maintaining distance to the robot. This design will be implemented in the Python programming language which is commonly used in the computer vision field for its vast libraries and community resources. The downside of Python is that the language is interpreted so it runs slower than other compiled languages such as C++.

Design Alternative 3

The robot will use the bug 0 algorithm[12] which is the least complex of the three. The algorithm is defined as follows, the robot heads in the direction of a target location. If an obstacle is detected, the robot will follow the obstacle until it can head in the direction of its goal again. Note that this is different from Design Alternative 1 where the robot chooses to follow a single line towards the goal. Rather, the path is dynamically updated as the goal is sighted. These steps are repeated until the robot has successfully reached its target.

In this design alternative, there will be no supplemental sensor. This will simplify the overall implementation but will also decrease confidence and reliability. This design alternative will use the BeagleBone AI as the single board computer to process the input data. This SBC is designed for machine learning applications and has embedded-vision-engine cores which are optimized for computer vision tasks [17]. This is ideal for this design alternative which is only using stereo cameras. This device is also open source and many community resources are available. The system will communicate with the robot's main computer by Serial Peripheral Interface (SPI) which does not require processing overhead but can have more complex hardware. A voice command will be used to initiate and deactivate the autonomous system. This allows for the autonomous system to be turned on and off from a distance, without having to reprogram the robot's remote controller. This design will use the language C++ due to its versatility and speed. C++ has external libraries to assist with computer vision and image processing, it is also compiled and thus can be executed much faster than Python. The downside to C++ is that it can be complex to implement when compared to Python.

Design Alternatives Comparison

In selecting design alternatives, the team gave significant weight to the object avoidance algorithm. Depending on the algorithm that is selected, the team will take different approaches in the hardware that is used and the programming that is done. The choice to include or not include additional sensors reflects finding a balance between setup difficulty and system reliability. For example, alternative 1 would make use of LiDAR in addition to the camera system. Thus, distance measurements could be compared and validated, but this comes with an expense of an increase in complexity. Alternative 3 on the other hand does not call for the use of additional

sensors. This simplifies setup, but puts the system integrity on the capabilities of the camera. Continuing on, the different single board computers share similar capabilities, however, since they are at the heart of the autonomous system, the corresponding communication methods, programming language(s), and user interface will reflect options that are most suitable for each computer. Overall, the three design alternatives are viable options by which the project can be completed.

V. Design Alternative Evaluation Criteria

To select the best design alternative, the team chose five selection criteria and performed a pairwise comparison on a scale of 1-9 for all five criteria. From the pairwise comparison, each criteria was given a weight.

1. Time needed to complete design and development

Time is one of the greatest constraints placed on the group. Due to the relatively high technical complexity of the project, the group anticipates to allocate a significant amount of time. To help reduce the pressure of this constraint, the best design alternative should have hardware/software that is easy to work with. This will streamline the development process.

2. Hardware/Software Compatibility

Since the project goal is to create an autonomous system, it is extremely important that it is possible to integrate it with the designated platform, the Unitree A1 Robot. The team worked to verify that all design alternatives are compatible. However, some approaches may be easier to work with than others. The best design alternative should have robust interaction between sensors, processing algorithms, and control commands.

3. Group's Technical Knowledge

Through experiences in student organizations, internships, and classes, the group has a diverse set of technical skills. It would be beneficial if the group could rely on what they already know during project development. This saves time and gives more confidence when working towards a solution. However, due to the nature of the project, the group is expecting to adapt and learn to work with new tools/concepts. The best design alternative should have a strong overlap with the technical skills of the group.

4. Simplicity of Design

Some design alternatives use additional hardware or more complex software implementations. This criteria emphasizes the importance of having a design that has the fewest hardware components while still meeting all of the engineering requirements. By having fewer components, this means that the overall complexity of the system is reduced, and it can prevent compatibility issues between different components. This can reduce development time. Having a simpler software approach and fewer sensors ensures that the single board computer is able to perform all necessary calculations and ensure an accurate and fast response time.

5. Availability of Project Resources (documentation, datasheets, application notes, etc.)

The team is making use of both hardware and software for this project. The availability of documentation will allow for successful integration between the various components of the system and help fill any gaps in the group's knowledge. Furthermore, clear resources reduce development time as guesswork is removed. The best design alternative should have a strong set of resources to back both the hardware and software.

VI. Selection of Design Alternative & Justification

Each design alternative was reviewed in terms of the selection criteria on a scale of 1-5, one being does not meet criteria and five being completely meets criteria. The team then averaged and applied the weights from the pairwise comparison to choose the best design alternative.

Table 2: Decision Matrix

Criteria	Weights	Alternative 1	Alternative 2	Alternative 3
Development Time	0.23	4	4	4
Compatibility	0.32	5	3	4
Technical Knowledge	0.12	5	2	3
Simplicity of Design	0.10	4	3	3
Resource Availabilities	0.24	5	3	3
Score		4.67	3.11	3.55

The design selection process showed that design alternative 1 aligned most closely with the valued criteria. It had the highest score of 4.67 whereas alternatives 2 and 3 had scores of 3.11 and 3.55, respectively.

The most valued criteria is compatibility because without it, the system would not meet most of its engineering requirements. It is important that our system work specifically with the Unitree AI Robot to meet all of the requirements set by the team and the UIC Robotics Lab. With design alternative 1, we are able to communicate with the robot via the ethernet ports on the Jetson Nano and the robot. The robot command scripts are natively written in C++, therefore using C++ ensures that the autonomous commands are compatible with the robot.

Next, resource availability is the second-highest criteria since this is one of the largest aids in development and research. Given that both the Jetson Nano and mapping/localization algorithm strategies are widely used in the world of autonomous systems, there are many different resources available online to help with research, development, and debugging.

Development time is a significant constraint so it was also given considerable weight. Given that design alternative 1 has compatibility with the robot and many external resources, it should have a lower development time compared to the other design alternatives which use less

common microcontrollers and communication interfaces. Design alternative 1 also has the simplest implementation of the autonomous enable/disable functionality. With just a physical switch, this feature has a very fast development time as opposed to voice command and remote interface which can take weeks to implement properly.

While the team does have widespread technical knowledge, it is important that we have enough in order to complete the project in a timely manner. Given that many members of the team have worked on mapping algorithms, Jetson Nanos, and remote controls, design alternative 1 supports the team's technical knowledge. C++ is a language that is taught in many classes at UIC so the team will be comfortable using it. But, given that Python is typically used in this application, the team plans to use both Python and C++. A few members on the team have done considerable work in Python and given their collective coding knowledge in C, Matlab, and C++, they will be able to transfer that knowledge to using Python.

Finally, in terms of simplicity, design alternative 1 has features that are simpler than other designs, which make it easier to implement. Design alternative 1 uses ethernet to connect to the robot's microcontroller, which requires no additional hardware and allows for a high data transfer rate. Using the Jetson Nano also aids in simplicity because of the GPIO pins, 4 USB ports, ethernet port, and HDMI port. This prevents the need for additional converters to connect sensors to the Jetson Nano. This design also uses a physical switch to enable and disable the autonomous system. This only requires a single device and minimal software compared to the other design alternatives. This design does use LiDAR as a supplemental sensor, although an additional component is being added, this should increase the reliability of the autonomous system. Using python and C++ also simplifies the design because of the features of each language. Python has a large collection of libraries specifically suited for image processing and machine learning while the robot control algorithms are programmed in C++. By using these two languages together, the team is able to use the advantages of each language, which will make the software implementation easier.

VII. Preliminary Design

Based on the design alternative and the team's decision matrix, design alternative 1 was chosen as the most appropriate alternative. Below is a preliminary design based on design alternative 1, including a level 1 schematic, circuit designs, and software workflows and designs.

Level 1 Schematic

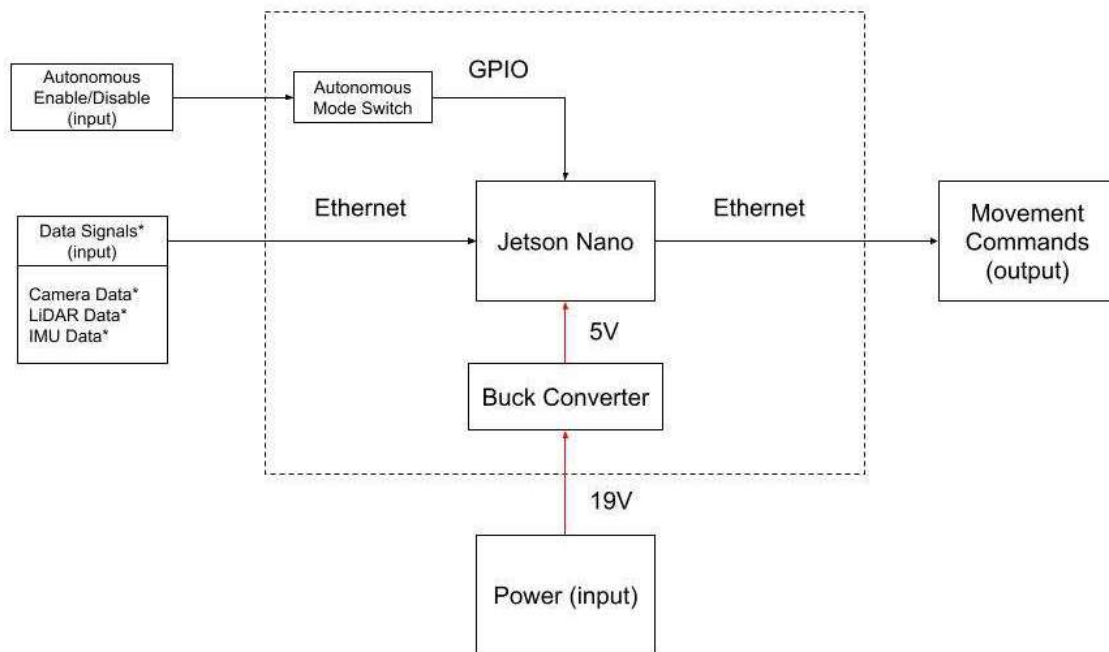


Figure 3: Level 1 Schematic

Figure 3, shown above, is a level 1 schematic for the autonomous system design. The main computer being used is a Jetson Nano, and it is receiving data via ethernet from the camera, LiDAR, and IMU located on the Unitree A1 robot. The Jetson Nano will receive power directly from the 19-volt source located on the Unitree A1 robot, which will be stepped down to 5 volts using a buck converter. From the user side, a switch on the controller will enable/disable the autonomous mode, which is read through a GPIO pin on the Jetson Nano. Finally, the Jetson Nano will communicate movement commands back to the robot via ethernet.

Circuit Design

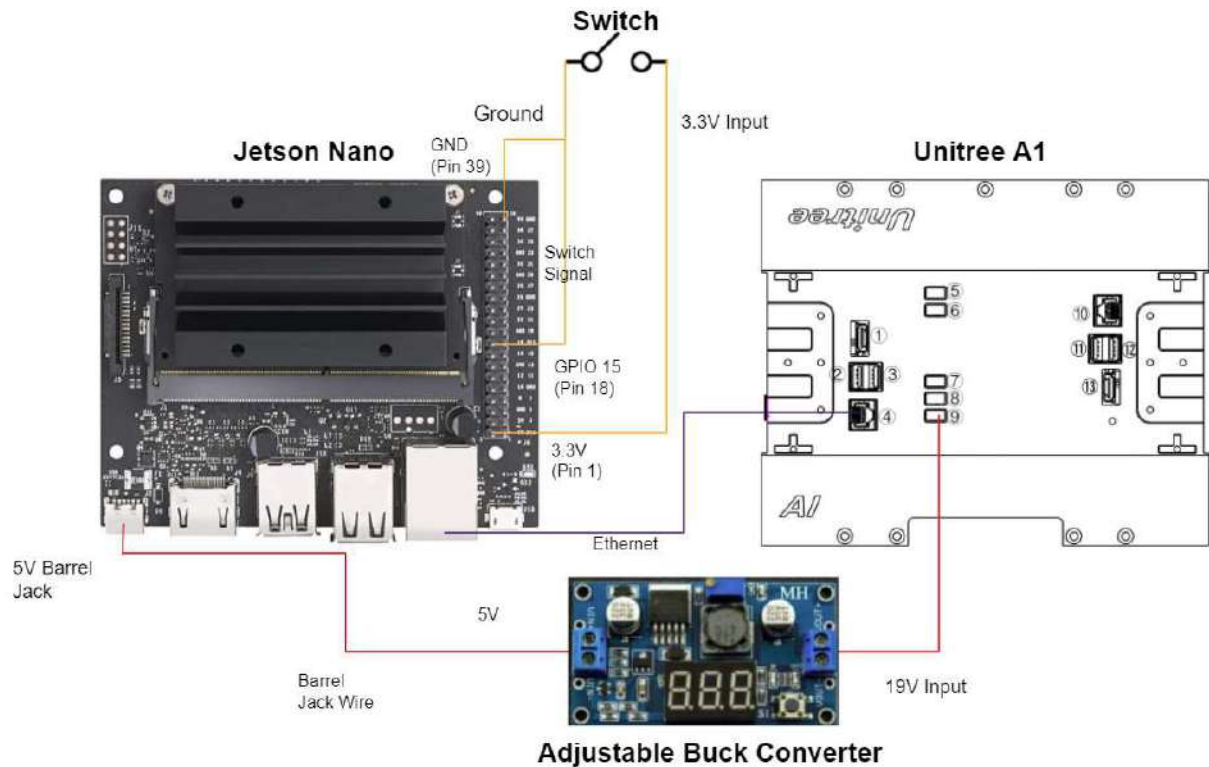


Figure 4: Circuit Diagram

Figure 4, shown above, is the circuit schematic for the pathVision project. The team is using a Jetson Nano as the single-board computer to process environmental data and detect objects in the path of the robot. The Jetson Nano is connected to the Unitree A1 robot using an ethernet connection directly from the Jetson Nano to the Unitree A1 Robot. This connection will send camera data, IMU data, and LiDAR data to the Jetson Nano and send commands to the Unitree A1 robot. The Jetson Nano is powered from its barrel jack port, which is connected to a buck converter that steps down the 19V supply from the Unitree A1 to 5V to power the Jetson Nano. The autonomous enable/disable will be the physical switch to enable and disable the autonomous system. This switch is connected to the 3.3V voltage supply pin on the Jetson nano, the other side of the switch is connected to GPIO pin 15 and GND. The datasheets for the Jetson Nano, buck converter, switch, and Unitree A1 Robot are in appendix E.

Software Design

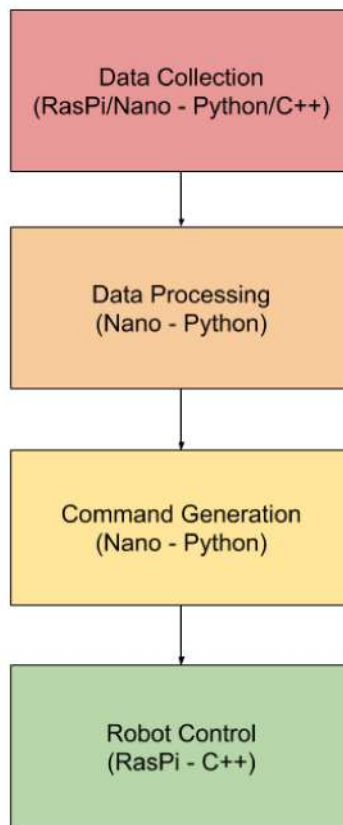


Figure 5: Software Workflow

The team anticipates that software development will be the most involved aspect of the project. For simplicity, the team has split the design between 4 different subroutines: data collection, data processing, command generation, and robot control. The core autonomous system as shown by the level 1 diagram is represented by the data collection and processing subroutines, as well as the command generation subroutine. The robot control subroutine is the link to physical implementation.

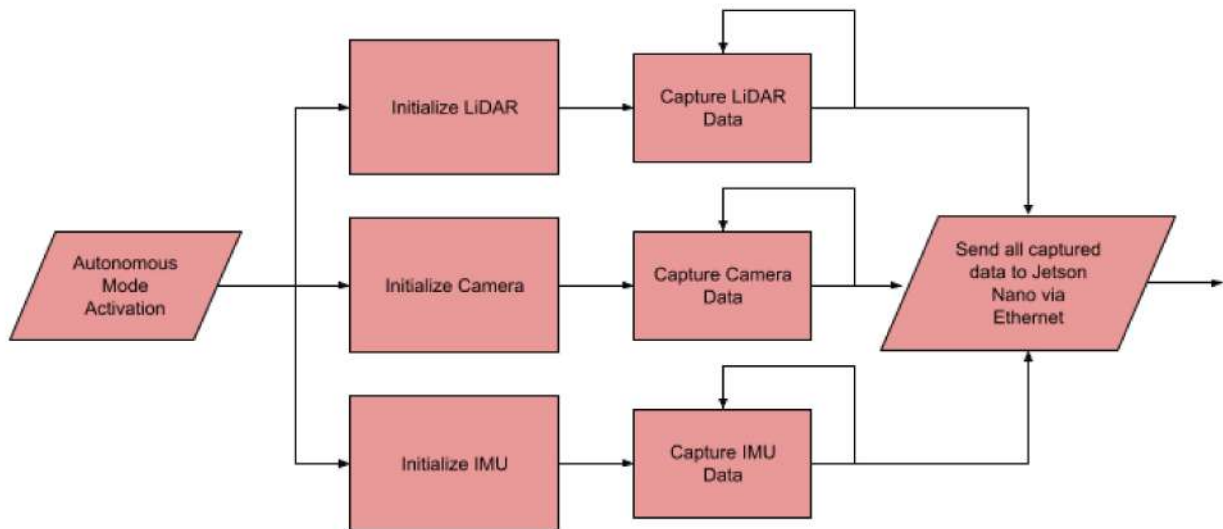


Figure 6: Data Collection Subroutine

The data collection subroutine shown above is relatively straightforward. All sensors must be initialized and then set to continuously feed data to the Jetson Nano. The following data is collected: camera, LiDAR, and IMU. As previously stated, all the data received from the robot will then be transmitted to the Jetson Nano via an ethernet connection.

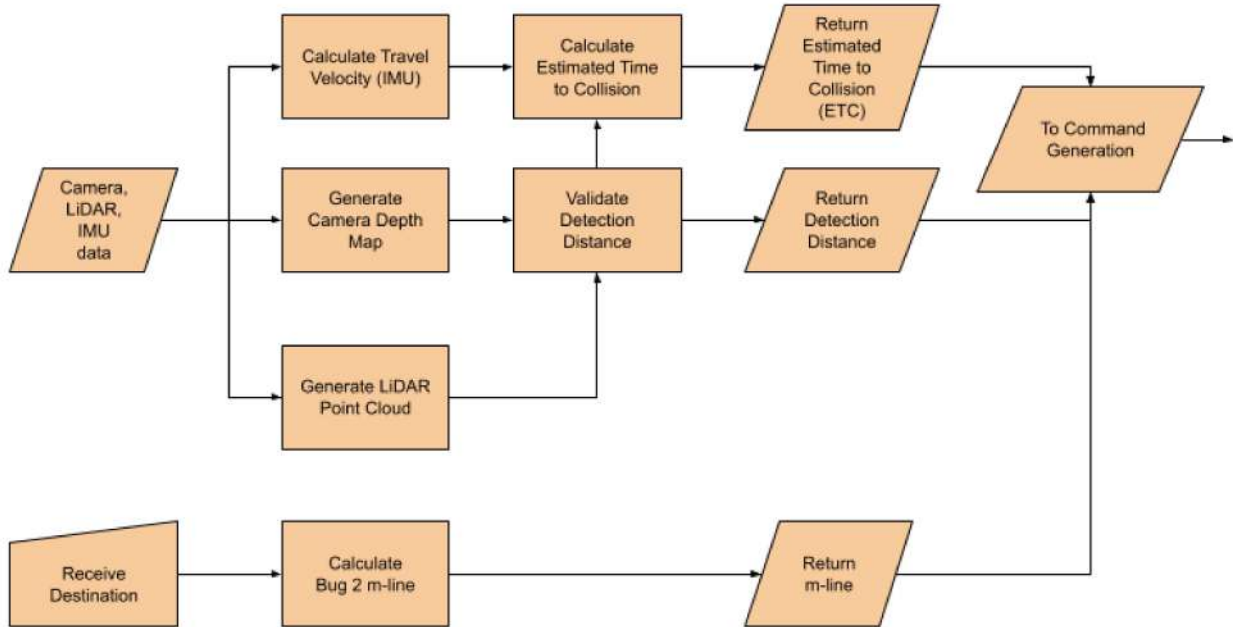


Figure 7: Data Processing Subroutine

After the data is collected, it is fed to the processing subroutine. The purpose of this subroutine is to calculate useful metrics that will support the generation of commands. Metrics to be calculated include the estimated time to a collision, distances to detected objects, and the m-line. The m-line is part of the bug 2 algorithm that is described in the design alternatives. For the purpose of this project, the m-line will be hardcoded but can also be modified by the lab. An example of this would be to set the robot to follow a line extending 5 meters forward. Progress along the line will be tracked using the IMU and the autonomous system will prevent collision as the robot progresses along the line.

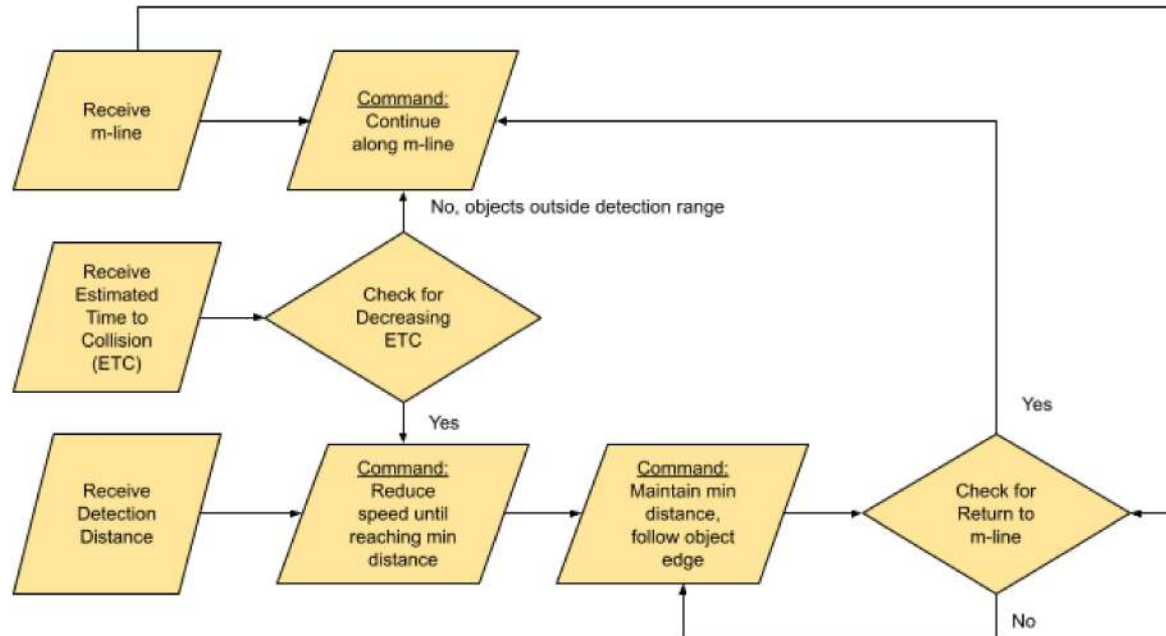


Figure 8: Command Generation Subroutine

The command generation subroutine is responsible for taking the processed data and converting it to commands that can be used to control the robot. First, a check is done to see if the estimated time to collision is decreasing. If this is true, then the robot is approaching an object. This means that the system will issue a control command that reduces the speed of the robot until it reaches a preset minimum distance away from an object. Next, the robot works through a loop of following the object's edge until it returns to the m-line. Once it returns, the robot will continue on the path towards the goal. For the other branch, if the time to collision is not decreasing, the robot will simply continue along the m-line. The implementation described above falls in line with the bug 2 algorithm. It should be noted that the group will make attempts to optimize the algorithm as they encounter and work through different corner cases or challenges to autonomous operation.

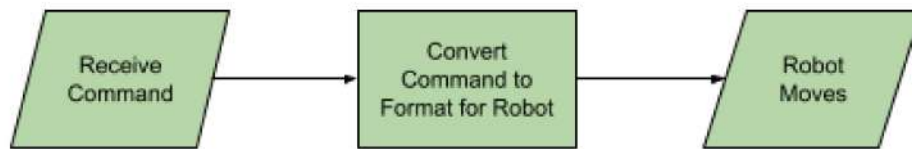


Figure 9: Robot Control Subroutine

The final step before having the robot move autonomously is to convert the actual movement command from the autonomous system, into direct instructions for the robot. The Unitree A1 Robot has a Raspberry Pi which is responsible for controlling the robot's movements. The movement scripts are written in C++. The idea is that general movement commands (eg. move forward, move left, etc.) would be taken and incorporated into the aforementioned movement scripts. The robot would then be capable of autonomous operation.

VIII. Testing

The team conducted various testing throughout the build process and has highlighted the major tests below. Also in the table are accounts of each engineering requirement and where the team confirmed testing for that engineering requirement.

Engineering Requirement	Was the engineering requirement met?
The vision system must detect 90% of static objects within 0.5 to 5 meters in front of the robot.	Yes, as described in the Final Testing section.
The system must navigate around any static obstacles that are detected. If the system cannot avoid an object or is unsure how to proceed, it must cease movement.	Yes, as described in the Final Testing section.
The camera system must be able to measure distance to objects at a minimum of .3 meters and up to at least 5 meters away from the center of the robot.	Yes, as described in the Camera Testing section.
All sensors and hardware must be able to mount to the robot without inhibiting mobility.	Yes, as described in the Integration Testing section.
The processing computer for the autonomous system must be able to connect and communicate with the robot's onboard control system.	Yes, as described in the Communication Testing section.
Software must be well documented and accessible from a gitHub repository.	Yes, as described in the Conclusion section.
The system must have a control override that returns the robot to manual control.	Yes, as described in the Integration Testing section.
The system must connect to the onboard power supply and be compatible with 5V, 12V, or 19V voltage supplies.	Yes, as described in the Buck Converter Testing section.
The system must have a physical kill-switch that cuts power to the robot preventing it from moving any further.	Yes, as described in the Integration Testing section.
Costs for additional hardware must not exceed	Yes, as described in the Final Design - Bill of

a value of \$250.00. This does not include additional hardware provided by the lab.	Materials section.
---	--------------------

Camera Testing

One of the first devices selected for the autonomous system was the camera. The team worked with a number of different cameras and ultimately found the Intel Realsense camera to work the best. The main requirement for the camera alone was to be able to detect objects from distances 0.3 to 5 meters away. This requirement was set in the early stages of the project and was later found to be partially unnecessary for the system. Ultimately the team found that the camera did need to measure objects in close proximity, and was able to fulfill the 0.3 meter minimum distance requirement but, the camera did not need to see up to 5 meters away as the final algorithm for the object detection did not need to process objects that were 5 meters away. Regardless, the camera used in the system does meet the full requirement of 0.3 to 5 meters of distance capture and the team measured this by visually testing being able to see objects that were in the distance requirements.

Buck Converter Testing

One of the first tests conducted was of the buck converter. This test was to ensure the converter was working properly and providing the accurate power for our system. The components we tested included the harness and buck converter itself. To begin the testing, before plugging anything into the buck converter and its harness, the continuity on each input and output wire was checked. This was done to ensure that the system was wired with the correct polarity and the system did not have any unexpected shorts or open circuits. Next, the system was plugged into the Unitree A1 robot and calibrated using the instructions provided. Then, the output voltage was tested to ensure it matched the expected output after calibration. The output voltage matched the required 5 V. The current was not tested as the system was not expected to draw current at this time. Finally, the system was plugged into the Jetson Nano, making it fully connected from the robot to the Jetson, and the Jetson was powered on and monitored for current warnings. After careful inspection, it was confirmed that the Jetson was receiving its full power and operating as expected. This concluded the testing for the buck converter and meets the engineering requirement of system compatibility and power consumption.

Communication Testing

Another vital test completed for this system was a communication test. The goal of this test was to ensure that the system could connect and communicate effectively with the robot. To complete this test, the system was connected to the robot via ethernet and TCP communication was started on both the robot and Jetson Nano. A message was sent from the robot to the Jetson and the Jetson sent a confirmation message back to the robot. This test passed as both devices were able to send and receive messages, meeting the effective communication requirements.

Integration Testing

Many of the system's components were tested and validated individually, but during the integration phase it was important to test compatibility and safety to protect the robot, the team, and the autonomous system. Testing during these stages helped ensure that each component of the system integrated with the others, and the system was safe to test as a whole. With safety being a priority, some of the first tests conducted were those pertaining to system override and shutdown. As documented in the engineering requirements, the system needed a sort of physical kill switch, which was fulfilled by manually lifting the robot off of the ground. This action causes the Unitree A1 robot to cease movement as it has sensors in its feet that detect when it is and is not in contact with the ground. For manual override, while connected to the autonomous system, one could terminate the autonomous scripts running on the Jetson Nano to cease the robot's autonomous movement. This automatically places the robot back into its manual control mode and can now only be moved with its joystick controller. The system now could be attached to the robot and the team could ensure safe operation and could intervene if necessary. Next, all of the autonomous system hardware was mounted and connected to the robot. A major requirement by the team and lab was for the autonomous system to not interfere with any movement functionality native to the robot. To meet this requirement the team created custom harnessing and a custom mount to attach the system to the robot. Once the system was fully attached and plugged into the robot, the team carefully moved the robot about and assessed how the entire autonomous system interacted with the robot and was able to determine that the autonomous system did not interfere with any of the robot's normal movements. This completed integration testing as now the autonomous system could fully attach to the robot, communicate

with the robot, receive power from the robot, and the team could respond appropriately in an unexpected situation.

Final Testing

Lastly, once the entire system was integrated with the robot, the team tested all final functionality to assess their completion of their goals for this project. The main goal of the final testing was to prove the autonomous system functionality with the robot. Here we set up our system fully connected to the Unitree A1 robot and allowed the autonomous system to run. We placed objects in the robots path to test if the robot would avoid the obstacles. As the system was running we took note of the objects it avoided and if the avoidance was successful. The team found that in the final round of testing, the autonomous system with the robot successfully detected and avoided all obstacles placed in its path after ten minutes of running continuously. The team ran this test a multitude of times and found the system's continuous performance to be consistent with the engineering requirements.

IX. Final Design

The final design for the robot includes two parts; the hardware design and the software design. The hardware design includes all of the physical components that are needed for the function of the autonomous system. Each component that is used is included in the bill of materials. In the software design, each piece of the software is outlined and how they interact together.

Hardware Design

The hardware design for the autonomous system is seen in figure 10. The design included the Jetson Nano, Intel RealSense Camera, and the buck converter. These components were then powered from the 19V battery on the robot. The ethernet port from the robot's raspberry pi was connected to the Jetson Nano. Note that the preliminary design discussed the use of LiDAR. This was dropped from the final design because of the added complexity. Additionally, the physical switch was removed from the robot since the autonomous system can be deactivated wirelessly from the user's keyboard. The robot also has built in safety features so that all movement can be stopped if the robot is picked up. Finally a mount for the autonomous system was designed to house the Jetson nano and the buck converter to secure them to the robot (figure 12). The Intel RealSense camera was then mounted to a built-in mount on the front of the robot (figure 11).

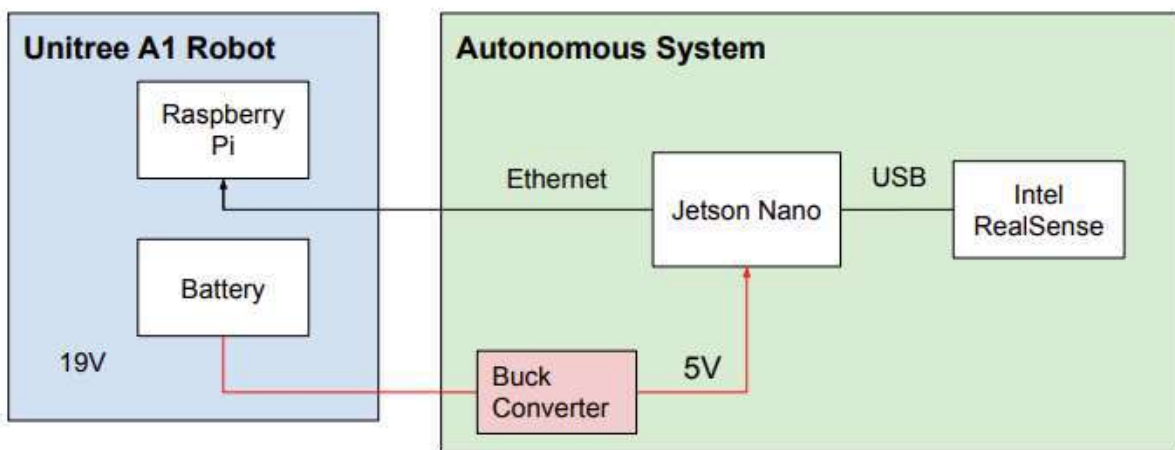


Figure 10: Hardware Design

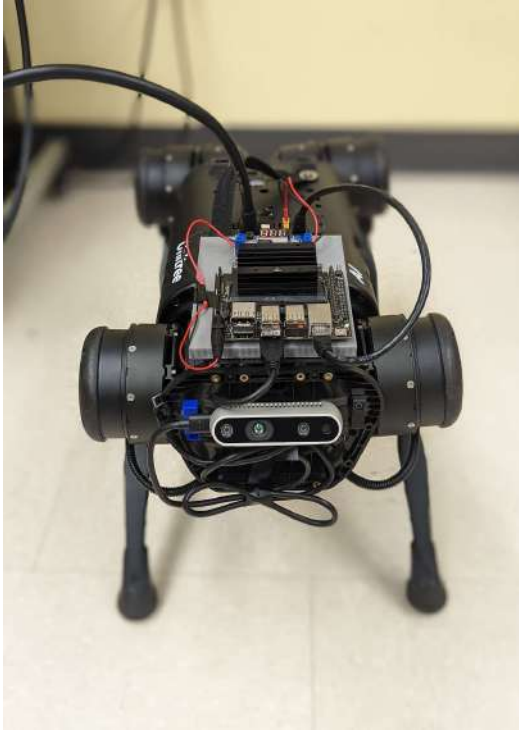


Figure 11: Robot Front View

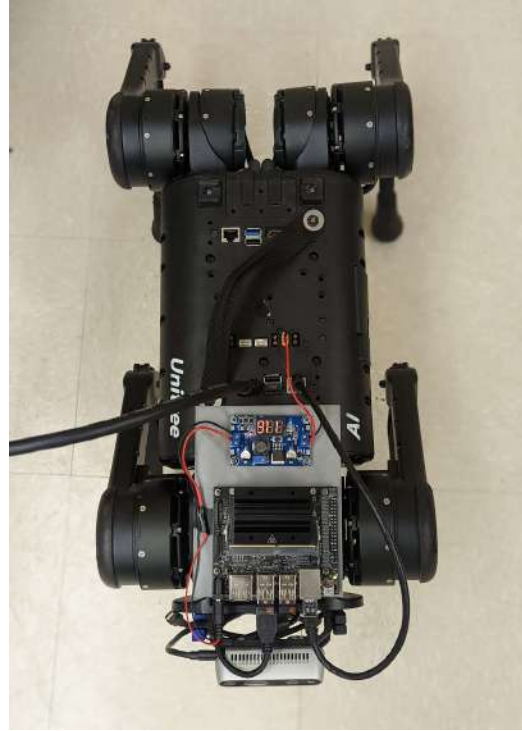


Figure 12: Robot Top View

Software Design

The software design for the autonomous robot is shown in figure 13. For the obstacle avoidance portion, the system collects the depth map data and checks if the object is within a certain threshold. Once the object is within the threshold, it looks for an alternate path. If an alternate path is found, the system sends a command to move the robot left or right. If an alternate path is not found, a stop command will be sent to the robot. For the object following system, once the data from the current frame and the depth map are collected, it is then interpreted using OpenCV in python. Then, the system locates the predetermined colored object in the frame and calculates the angle to recenter the robot in reference to the object. A command is then issued from the Jetson Nano to the robot's Raspberry Pi which contains the angle the robot needs to turn. Finally, the commands are converted to allow them to be sent over a TCP connection which is established over ethernet. All of this is processed on the Jetson Nano and then the commands are sent using ethernet to the robot's Raspberry Pi to translate them into movement.

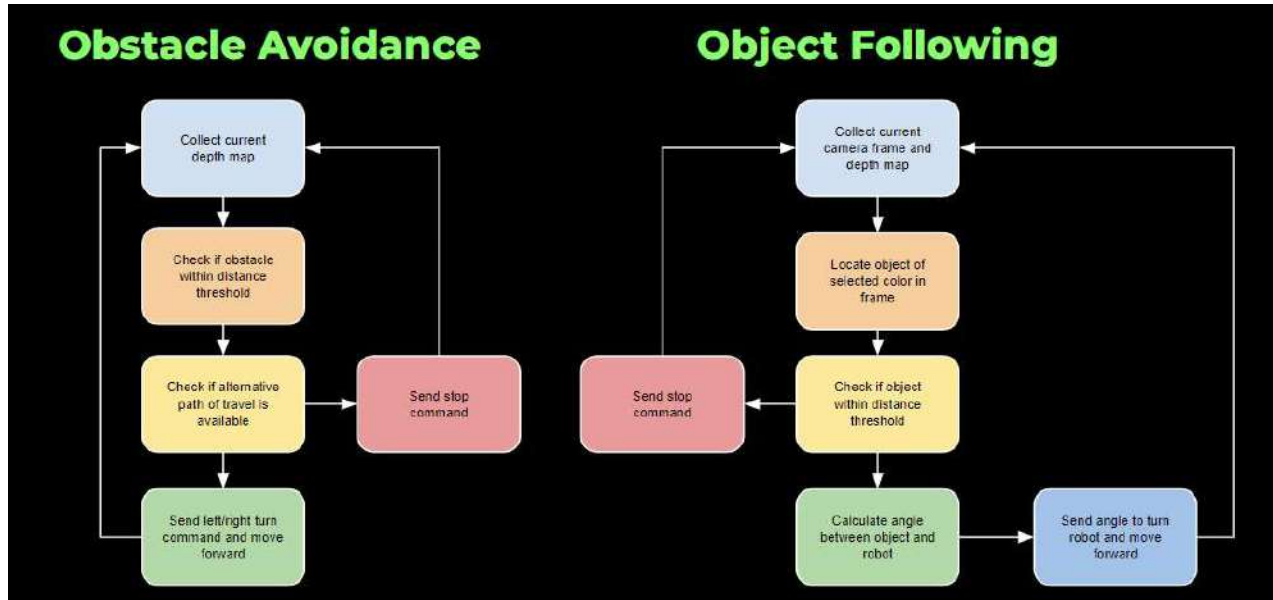


Figure 14: Software Design

Bill of Materials

The bill of materials for the autonomous system consists of five components for a total cost of \$313.57. The team has chosen to not include the Unitree A1 Robot, or any of its included components, into the product budget because the autonomous system will be connected to the Robots’ Laboratory robot as a separate system. Due to materials provided by the UIC Robotics and Motion Laboratory, including the Intel Realsense camera, the team was able to remain under the \$250 senior design budget. When this system is added to the robot, the total cost comes out to \$3,113.57.

Part Name	Vendor	Quantity	Cost per Unit	Total
Jetson Nano	UIC Robotics Laboratory	1	\$99.00	\$99.00
DFR0379 - 20W Adjustable DC-DC Buck Converter with Digital Display	DFRobot	1	\$4.90	\$4.90
Barrel Jack Converter	Digi-Key	1	\$3.81	\$3.81
Wifi Adapter	Walmart	1	\$10.86	\$10.86

Intel Realsense Camera	UIC Robotics Laboratory	1	\$195.00	\$195.00
			Total:	\$313.57

X. Task Allocation and Timeline

At the beginning of the semester, the team planned the building process for the entire autonomous system. Included in this section is a week by week breakdown of the tasks completed and the contributions of each team member.

Timeline

Below is the approximated time allocated to completing the project. The team defined tasks for each week of the Spring 2022 semester that include implementation, testing, and presenting at the UIC Engineering Expo event.

Start Date	Finish By	Number of weeks	Week	Lead	Task	Description
01/10/22	01/17/22	1	2	A	Software configuration, conduct research for prototyping and purchases.	Configuring Jetson Nano, researching different ways of using computer vision and purchasing critical components.
01/17/22	01/24/22	1	3	J	Begin prototyping code for different subsystems.	Begin development of prototypes for communications, computer vision, and movement.
01/24/22	01/31/22	1	4	E	Achieve basic communication functionality, judge prototyping results to determine best computer vision methods.	Send messages over TCP to verify communication functionality. Test Neural Network Approach.
01/31/22	02/07/22	1	5	P	Begin stereo depth map generation attempt, analysis on converting sent data to movements.	Set up dual webcams that support depth map generation functionality. Explore how simple commands can be transformed into robot movements.
02/07/22	02/14/22	1	6	A	Modify depth map script to support better depth map generation. Test Alternate object following approaches.	Low quality of depth maps from dual webcams will try to be improved through modifications. Begin Testing OpenCV Object Following.

02/14/22	02/21/22	1	7	J	Purchase IntelRealsense camera, begin developing command generation prototype, power Jetson Nano from robot. Create a basic movement script.	IntelRealsense must be purchased because of continued difficulties with other depth map approaches, command generation can be prototyped using static images, hardware integration begins with receiving power from the robot. Basic movement script allowed us to understand how to program the robot.
02/21/22	02/28/22	1	8	E	Learn about use of IntelRealsense, Develop general software outline.	Code samples were collected so depth mapping can be quickly implemented once the camera arrives. General software will be useful before full system integration begins.
02/28/22	03/07/22	1	9	P	Modifications to communication script (on RasPi in C++), implement depth mapping on IntelRealsense, determine how to transmit robot angle (for object following).	Communication script was modified to facilitate transmission of data. Accurate depth maps were created using the IntelRealsense. Angle is based on the difference between the detected object and the current robot orientation.
03/07/22	03/21/22	2	10	A	Implement depth map generation on Jetson Nano, Integrate socket script with movement code on RasPi.	Tasks to be completed before system integration, each module can now fully accomplish its individual task (communications, command generation, movement)
03/14/22	03/28/22	2	11	J	System Integration	Perform integration of all subsystems, work to ensure that every subsystem is working properly. Verify that stated performance objectives are fulfilled.
03/21/22	03/28/22	1	NA	NA	Continued System Integration	Perform integration of all subsystems, work to ensure that every subsystem is

						working properly. Verify that stated performance objectives are fulfilled.
03/28/22	04/04/22	1	12	E	Final System Integration, Testing, Expo Preparation	System integration is completed. Robot is functioning as desired. Complete all expo items.
04/04/22	04/18/22	1	13	P	Expo Presentation	UIC Engineering Expo Event

Contributions of Work

For the project, the team broke it up into three parts: robot motion, communication, and computer vision. Pranay was responsible for programming the robot to move, Emily was responsible for creating the communication link between the autonomous system and the robot, and Alex and Jonathan were responsible for the computer vision portion. The Computervision system was also broken up into two parts: depth mapping and object tracking.

Alex

My work on the project was centered around the computer vision objectives. Early on in the project I performed research on different methods of computer vision that could be used to fulfill the objectives of the project. I then moved to implementing a prototype for generating depth maps. This included a traditional stereo camera, dual web cameras, and the IntelRealsense. The latter option was selected because of the high quality depth maps it generated and its overall ease of use. The first step in working with the IntelRealsense involved installing the appropriate drivers in order to interface with the camera in a programming environment on the Jetson Nano. As this process was ongoing, I developed a prototype command generation script that takes depth information and outputs collision avoidance commands. As soon as depth information was available from the IntelRealsense on the Jetson Nano, I integrated with the prototype command script to complete the computer vision module. During system integration I worked on adding depth map capabilities to the object following and collision avoidance algorithms. Finally, I helped debug and ensure that all system components tied to computer vision were functioning as expected. I also created and edited the expo video from the recorded parts from my teammates and myself.

Emily

As mentioned above my main priorities for this project were the communication system as well as the hardware setup on the robot. The communication link was set up with an ethernet cable and this design decision was made in the previous semester based on our research. I implemented python and C++ scripts on both the robot and Jetson Nano to allow both systems to send data and communicate. I then helped integrate these scripts into the code that Pranay and Jonathan created. I also created all of the electrical harnessing for our system and created a 3D mount for our hardware. This 3D mount was designed to hold the Jetson Nano, and the buck converter, while attaching to the robot. I then 3D printed the mount and did some post processing to ensure the mount properly fit and securely held the components. I also mounted the Intel Realsense and secured all components and harnessing onto the robot for operation. Finally, during our integration phase I helped with script integration and testing. This included being in the lab and assisting with code changes and any errors we encountered. For expo and our final report, I gathered all of our extra materials for expo and helped create the expo slides. For the final report I wrote the testing section and created the table comparing our engineering requirements. I also added to the appendix including part of the intel realsense datasheet as well as our task management document.

Jonathan

My primary focus was on the computer vision aspects of the design. My primary focus was on object detection and following. Originally, I researched implementations for line following using neural networks. However, after I conducted initial testing with these neural networks, I decided to switch to using OpenCV to detect objects. I was able to develop two different systems; one for tracking a designated object and one for line following. The line following script was not used in the final design, but it could have been combined with the depth mapping system in the future. After completing the object following system, I worked closely with Pranay and Emily to link up the object following to robot motion. I also worked with Alex to track the distance to the object we were following. During integration testing, I also helped troubleshoot issues that occurred with the different systems of the robot. In addition to making the autonomous system, I outlined the expo video and made the slides.

Pranay

My main focus was with the robotic movement itself. I had to understand how the robot worked and how to program the remote so that the team and I could use it to receive commands. Initially, my primary focus was to read into the libraries and understand the prewritten functions for the high level code of the robot. Then, I had to implement a test script to see if the research I have done actually allows the robot to move. This was the hardest part for me as I was getting a lot of errors and was not able to get the robot to move. Once I got the test script working I created the final script which was able to be ready for testing with the object following and object detection code. Finally, I was working on integration itself, making sure that all elements of everyone's code worked flawlessly, especially my part.

XI. Lessons Learned

Alex

The senior design experience illustrated many of the nontechnical aspects of engineering. The team had to conduct interviews/surveys to understand users needs, schedule meetings, manage many different deadlines, etc. Although these tasks can be less interesting than technical development, a project is less likely to succeed if they aren't addressed. On the project management side, I realized the importance of leaving plenty of time available for system integration and testing. This was the most laborious part of this project as it involved a lot of troubleshooting and adjustments. The team was able to successfully complete the project due to the strong schedule that was set up. Overall, senior design reflected the importance of teamwork, organization, and strong communication with all parties involved.

Jonathan

From this project I was able to learn the full engineering process. In most of my prior experiences the problem that I was trying to solve was well defined and I had limited options to solve the problem. However, in senior design, I was able to learn how to properly define a goal and identify the needs of a project. Then I was able to learn how to translate these needs into technical requirements. From this I learned better communication skills that I believe helped me to improve my ability to describe an engineering problem. In addition, I learned a lot from the integration of our system. It helped me to understand how important good communication and planning is when making a design to make all of the parts work together. Ultimately, the senior design project helped me to fully understand the engineering process and gain valuable technical experience.

Emily

My senior design experience has taught me many technical and non-technical skills which I am grateful for and look forward to continuing building throughout my career. As far as technical skills, I was able to work on the networking and hardware sides of the project, two

areas of which I have little to no experience in. Through our project I was able to learn about ethernet communication between two devices while also building my python and C++ skills. I was also able to assemble hardware after writing and completing testing on the devices. Outside of the technical aspects of the project I was able to build on my communication skills, time management, organization, and teamwork. Through all of the in-class presentations and discussions as well as the weekly forms, I really was able to refine organizing my time and effectively communicating in regard to our project. When working with my team, I was also able to work on communicating in a healthy way and contributing to the project in ways that best helped the team. While a lot of our work was done individually, when we all came together for testing and integration, it was clear how well we worked together and that made things go very smoothly in a project stage that is notoriously difficult. All in all my senior design experience was a positive one and I am proud of myself and everyone on my team for all of their hard work this semester!

Pranay

What this project and this whole senior design class have taught me is the thought process of a project like this in the real world. It gave me an insight into how I need to brainstorm, prototype, and develop a product with a team. One field I am particularly interested in is robotics itself, and this project catered to my interest and let me expand on my knowledge of quadruped robots. Specific technical skills that I greatly improved on while working on the Unitree A1 robot are coding in C++ and understanding the movement of robots. Also, one of the major hiccups of this project was integration, and though we did discuss what we needed beforehand, I severely underestimated the amount of time it would take in order to get all the systems to be working together flawlessly. Which shows how nothing will work on the first try and help me understand it is a laborious process to ultimately get what we are looking for. Overall, this senior design project helped me learn a lot and I am delighted by all that my team and I have accomplished.

XII. Conclusion

The final product successfully achieved the goals of obstacle avoidance and object following. Each team member contributed a component that was vital to the fulfillment of these goals. In order to complete the handoff of the project, the team collected and uploaded all necessary code files to gitHub for use by members of the Robotics and Motion Laboratory. For future improvements, two possible changes stand out. First, the startup process of the autonomous system could be streamlined. In the current form, a set of specific actions must take place in order to activate the autonomous functionality. The addition of scripts that simplify this process would allow for more rapid testing/use of the system. Second, the team noticed that the robot would slightly deviate from its path. This was likely caused by the native software/mechanical components. The team experimented with counteracting this drift through the use of software, but the issue continued to persist. Streamlining startup and eliminating drift could be achieved with future work. Overall, the team is satisfied with the final product and thankful for the learning experiences during the project.

XIII. References

- [1] National Highway Traffic Safety Administration, “Automated Vehicles for Safety,” U.S. Department of Transportation, Washington, D.C., 2020.
- [2] Tesla Motors, Inc., “Tesla Vehicle Safety Report,” Tesla Motors, Inc. Palo Alto, CA, 2021.
- [3] R. Lässig et al. Robotics outlook 2030: How intelligence and mobility will shape the future. Boston Consulting Group Boston, MA. Boston. 2021.
- [4] Autopilot Review, Elon Musk on Cameras vs LiDAR for Self Driving and Autonomous Cars.(April 27, 2019). Accessed: Sep. 28, 2021. [Online Video]. Available: <https://www.youtube.com/watch?v=HM23sjhtk4Q>
- [5] M. Dikmen and C. Burns, "Trust in autonomous vehicles: The case of Tesla Autopilot and Summon," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, pp. 1093-1098, doi: 10.1109/SMC.2017.8122757.
- [6] “Bug Algorithms and Path Planning”, <https://spacecraft.ssl.umd.edu/academics/788XF14/788XF14L14/788XF14L14.pathbugsmapsx.pdf> (accessed Oct. 28, 2021).
- [7] “Robotic Motion Planning: Bug Algorithms”, https://www.cs.cmu.edu/~motionplanning/lecture/Chap2-Bug-Alg_howie.pdf (accessed Oct. 28, 2021)
- [8] M. Raibert et al., “BigDog, the Rough-Terrain Quadruped Robot,” in *Proceedings of the 17th World Congress*, Seoul, Korea, Jul. 2008
- [9] Boston Dynamics, Inc.. “Spot,” [BostonDynamics.com](https://www.bostondynamics.com/spot#id_third). https://www.bostondynamics.com/spot#id_third (accessed Oct. 1, 2021).
- [10] K. Blankespoor, B. Stephens, and M. Silva, “Handling gait disturbances with asynchronous timing,” 12-Jan-2021
- [11] D. Jonak, M. da Silva, J. Chestnutt, and M. Klingensmith, “Autonomous Map Traversal with Waypoint Matching,” 13-May-2021
- [12] G. S. Hornby, S. Takamura, T. Yamamoto and M. Fujita, "Autonomous evolution of dynamic gaits with two quadruped robots," in IEEE Transactions on Robotics, vol. 21, no. 3, pp. 402-410, June 2005, doi: 10.1109/TRO.2004.839222.

- [13] MarketWatch, “Autonomous Vehicle Market Share 2021 Global Trend,” segmentation, size, Business Growth, top key players analysis industry, opportunities and forecast to 2030,”*MarketWatch*, 21-Jul-2021.
- [14] “Unitree - A1”, Unitree. <https://www.unitree.com/products/a1/> (accessed Oct. 17, 2021).
- [15] “Jetson Nano Developer Kit”, Nvidia.<https://developer.nvidia.com/embedded/jetson-nano-developer-kit> (accessed Oct. 28, 2021).
- [16] “Rock PI N10”, Radxa . <https://wiki.radxa.com/RockpiN10> (accessed Oct. 28, 2021).
- [17] “BeagleBone AI”, BeagleBoard. <https://beagleboard.org/ai> (accessed Oct. 28, 2021).
- [18] *Standard for Microcontroller System Serial Bus*, IEEE Standard 1118.1, 1990. Available: <https://ieeexplore-ieee-org.proxy.cc.uic.edu/document/159173>
- [19] *Standard for Robot Map Data Representation for Navigation*, IEEE Standard 1873-2015. Available: <https://ieeexplore-ieee-org.proxy.cc.uic.edu/stamp/stamp.jsp?tp=&arnumber=7300355>
- [20] "A Landscape for the Development of Dependable Machines," in *A Landscape for the Development of Dependable Machines - White Paper* , vol., no., pp.1-34, 30 June 2021. Available: <https://ieeexplore-ieee-org.proxy.cc.uic.edu/document/9468432>
- [21] *Standard for Fail-Safe Design of Autonomous and Semi-Autonomous Systems*, IEEE Standard P7009, 2017. Available: <https://standards.ieee.org/project/7009.html>

XIV. Appendices

Appendix A: Survey Answers

Survey questions addressed to Professor Bhounsule and Jonathan Garcia

1. What are the main uses of the robot?
 - The main uses of the robot are to support research on low level control and to act as a testbed for developing autonomous systems.
2. What type of environment will the robot typically operate in?
 - The environment is indoors with consistent lighting.
3. How would having an autonomous system benefit the lab and/or contribute to research?
 - Having an autonomous system will improve the laboratory's capabilities in conducting testing and be a better representation of systems being deployed to the real world.
4. Would it be beneficial if the autonomous system could be run on different robots?
 - An autonomous system is a collection of hardware and software and can be deployed to different robots. The best result would be achieved when working with a single system.
5. What are common challenges teams face while developing autonomous systems?
 - Common challenges are time constraints but, an early start with the technology allows teams to make progress and get around any roadblocks that may be faced.
6. Have any corner cases been identified with this particular robotic system?
 - Corner cases will be identified as the team works through the project, it is up to the team to decide how they should be handled.
7. What model robot is the autonomous system being designed for?
 - Unitree A1 Quadruped
8. What are the general capabilities of the stock setup of the robot?
 - The robot can move through a flat environment with ease, it can lower itself to the ground, but cannot jump.
9. What are the robot's on-board power capabilities?
 - The robot has its own battery that can power an external Jetson Nano as well as any other sensors we choose to add to the system.

10. What type(s) of microcontrollers are used to control the robot?
 - The robot has two computers, the one we will be using is the onboard Raspberry Pi 4b+.
11. What internal data can be received from the robot to complement sensor data?
 - The robot has an Inertial Measurement Unit (IMU), position sensing capabilities, and power usage monitoring.
12. What type(s) of external sensors are included with the robot?
 - The robot has a LiDAR system that we may be able to work with. Otherwise we will be using off-the-shelf sensors.
13. Will the environment inhibit the use of certain sensors?
 - Sensors should be carefully selected so that the robot can effectively and efficiently operate in a pre-selected environment.
14. What types of documentation are available to gain familiarity with the robot?
 - We were supplied with a manual for the robot as well as a GitHub repository with code examples on how to control the robot.
15. Should an interface be developed that allows users control over autonomous operations?
 - A user interface is not something that is required but may be implemented for ease of use and testing.
16. Should the robot camera feed be displayed in real time?
 - This requirement is not needed for the lab but we may implement it to make testing easier.
17. What type of obstacles should the robot be able to avoid?
 - A maze would be a good way to showcase the object avoidance capabilities of the robot, another approach would be to develop a more general system that avoids any object that is moving towards the robot.
18. Does obstacle avoidance include maneuvers like ducking/jumping?
 - The robot does not have the ability to jump but it can duck. We can choose to implement these but they are not necessary.

Appendix B: IEEE Standards

IEEE 1118-1990 - Standard for Microcontroller System Serial Control Bus

IEEE 1118-1990 is relevant to the project because it describes the importance of having a control bus that incorporates microcontrollers and other devices using a communication protocol. This standard was published in 1990 and is now inactive, but the standard describes high level standards for using a control bus with a microcontroller. This standard emphasizes the importance of selecting a communication bus that is optimized for the data acquisition system, the control devices, and for testing purposes.

IEEE 1873-2015 - Robot Map Data Representation for Navigation

IEEE 1873-2015 applies to the engineering requirements because it brings formal definition to the operating environment. Specifically, the standard outlines three different types of operating environments for autonomous systems. First, examples of the *Room scale* environment include homes, offices, or shops. *Vista scale* environments include factories, terminals, farms, and campuses. Finally, *the Environment scale* covers broad areas such as towns, districts, and cities. Per the user's needs, the autonomous system will operate in the room scale environment.

IEEE P2851 - A Landscape for the Development of Dependable Machines

IEEE P2851 applies to the engineering requirements because it specifies activities for autonomous systems and the dependability of intelligent machines. This standard addresses the safety engineering issues of creating dependable autonomous systems. This applies directly to our requirements regarding functionality. The system will be attached to the robot in a way that ensures it will not affect the movement or normal functionality of the robot. Also, in the event where the system is unsure how to proceed, it will cease movement.

IEEE P7009 - Standard for Fail-Safe Design of Autonomous and Semi-Autonomous Systems

IEEE P7009 is related to the engineering requirements because it creates a fail-safe mechanism in the case of an emergency or unintentional behavior. This standard describes a system test and rates the fail safely scale from weak to strong. This is important for our

requirements since we are working with a quadruped robot, and if something unintentional happens we need to take into account the safety of the people near the robot.

Appendix C: Concept Map

Concept Map of Design Alternatives

Obstacle Avoidance Algorithm	Supplemental Sensor Module(s)	Autonomous system connection to robot	Programming Language	Single Board Computer	Autonomous Mode Enable/Disable
Bug 2 (Greedy) Algorithm	LiDAR	Ethernet	Python/C++	Jetson Nano	Physical Switch
Bug 1 (Exhaustive) Algorithm	Ultrasonic	I2C	Python	Rock Pi N10	Remote Interface
Bug 0 (Simple) Algorithm	None	SPI	C++	BeagleBone AI	Voice Command

The concept map is broken up into six categories. The obstacle avoidance algorithm defines the type of approach that will be used to avoid obstacles in the path of the robot. The supplemental sensor modules will be used along with the camera system and the robot data as the inputs into the autonomous system. The autonomous system connection to the robot is the communication method that will send commands to the robot and receive data from the robot (imu, motor encoder data). The single-board computer will be receiving all of the input data and processing it to detect objects in the path of the robot. Then it will send commands to the robot's microcontroller. The autonomous mode enable/disable will be the system that will turn the autonomous system on and off.

Appendix D: Pairwise Comparison and Decision Matrix

Table D-1: Individual Pairwise Comparison

Level of importance

1.0 equal weight

3.0 moderate

5.0 strong

7.0 very strong

9.0 extreme

Alex	Development Time	Compatibility	Technical Knowledge	Simplicity of Design	Resource Availabilities	Geometric Mean	Weights
Development Time	1	3	1	1	1/3	1.00	0.19
Compatibility	1/3	1	2	1/2	5	1.11	0.21
Technical Knowledge	1	1/2	1	5	3	1.50	0.29
Simplicity of Design	1	2	1/5	1	1	0.83	0.16
Resource Availabilities	3	1/5	1/3	1	1	0.72	0.14

Emily	Development Time	Compatibility	Technical Knowledge	Simplicity of Design	Resource Availabilities	Geometric Mean	Weights
Development Time	1	1/5	2	1	1/5	0.60	0.09
Compatibility	5	1	5	4	1	2.51	0.39
Technical Knowledge	1/2	1/5	1	1/2	1/5	0.40	0.06
Simplicity of Design	1	1/4	2	1	1/2	0.76	0.12
Resource Availabilities	5	1	5	2	1	2.19	0.34

Jon	Development Time	Compatibility	Technical Knowledge	Simplicity of Design	Resource Availabilities	Geometric Mean	Weights
Development Time	1	1	5	7	1/5	1.48	0.21
Compatibility	1	1	9	9	1	2.41	0.34
Technical Knowledge	1/5	1/9	1	1/3	1/5	0.27	0.04
Simplicity of Design	1/7	1/9	3	1	1/5	0.39	0.05
Resource Availabilities	5	1	5	5	1	2.63	0.37

Pranay	Development Time	Compatibility	Technical Knowledge	Simplicity of Design	Resource Availabilities	Geometric Mean	Weights
Development Time	1	3	7	7	1	2.71	0.41
Compatibility	1/3	1	9	7	3	2.29	0.34
Technical Knowledge	1/7	1/9	1	1	3	0.54	0.08
Simplicity of Design	1/7	1/7	1	1	1	0.46	0.07
Resource Availabilities	1	1/3	1/3	1	1	0.64	0.10

Table D-2: Average Weights of Pairwise Comparisons

	Alex	Emily	Jon	Pranay	Weights
Development Time	0.19	0.09	0.21	0.41	0.23
Compatibility	0.21	0.39	0.34	0.34	0.31
Technical Knowledge	0.29	0.06	0.04	0.08	0.12
Simplicity of Design	0.16	0.12	0.05	0.07	0.10
Resource Availabilities	0.14	0.34	0.37	0.10	0.24

Table D-3: Rating of Each Design Alternative Relative to Criteria

- 1 - Does not meet the criterion
 3 - Partially meets the criterion
 5- Completely meets the criterion

	Development Time	Compatibility	Technical Knowledge	Simplicity of Design	Resource Availabilities
Alternative 1	4	5	5	4	5
Alternative 2	4	3	2	3	3
Alternative 3	4	4	3	3	3

Table D-4: Decision Matrix

Criteria	Weights	Alternative 1	Alternative 2	Alternative 3
Development Time	0.23	4	4	4
Compatibility	0.32	5	3	4
Technical Knowledge	0.12	5	2	3
Simplicity of Design	0.10	4	3	3
Resource Availabilities	0.24	5	3	3
Score		4.67	3.11	3.55

Appendix E: Datasheets

Unitree A1 Robot

Product Description

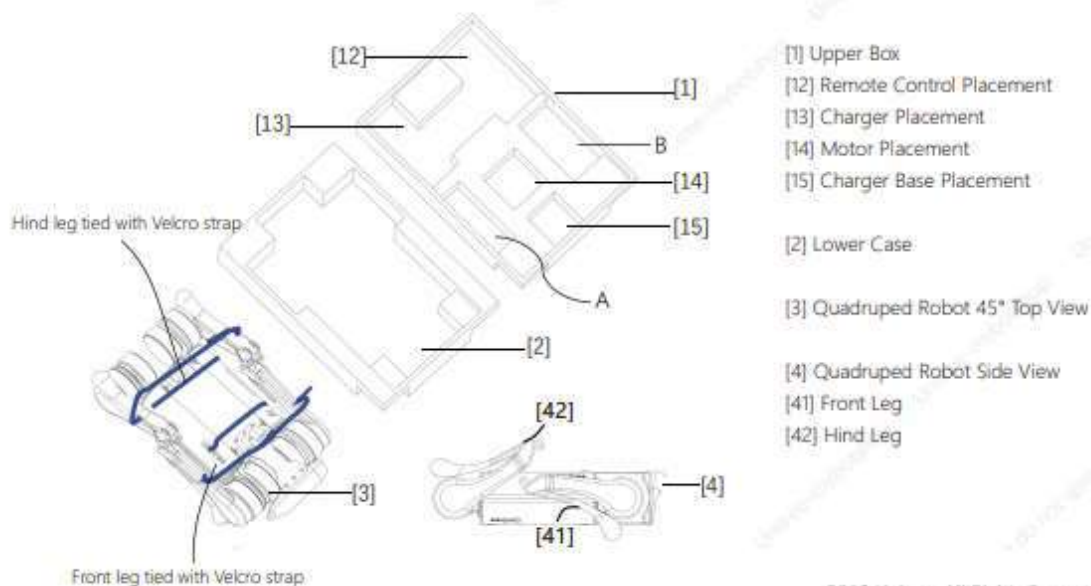
Introduction

The high performance A1 quadruped robot consists of a quadruped robot, a remote control and a remote control software. The whole machine has 12 DOF (composed of 12 high-performance servo motors), uses force control technology to perform compound control of force and position for each joint. Adopt streamlined mechanical structure to reduce manufacturing difficulty and improve machine reliability. A1 have reached domestic and international leading levels in terms of structure, kinematics and cost.

Feature Highlights

The optimal design of joint parts makes the quadruped robot not only reduce the cost, but also greatly improve the motion performance and service life. In Sport mode, the maximum joint speed of 21rad/s allows A1 to speed up to 3.3m/s in an instant, possess excellent balance ability; and the torque of 33.5NM allows A1 to easily achieve backflip; the joints can be quickly disassembled and easy to maintain. The addition of a multi-eye depth camera allows the quadruped robot platform to have intelligent applications such as real-time image transmission (image transmission quality 720P / 30fps) and character following; and supports secondary development. With optional lidar, extended functions such as dynamic obstacle avoidance, navigation planning, autonomous positioning, and map construction can be completed; optional NVIDIA TX2 can be used for visual SLAM and gesture recognition. At the same time, A1 supports APP control for Android and IOS.

Unpacking and Packing



Unpacking


Place the box on the flat ground according to the placement requirements (Face up), then open the upper box and lift the whole robot out, as shown in the figure above. Remove the robot and remote control, charger, etc. from the box, place the robot on the flat ground, untie the velcro strap of the robot's leg, and then prepare for the boot.

Before Packing

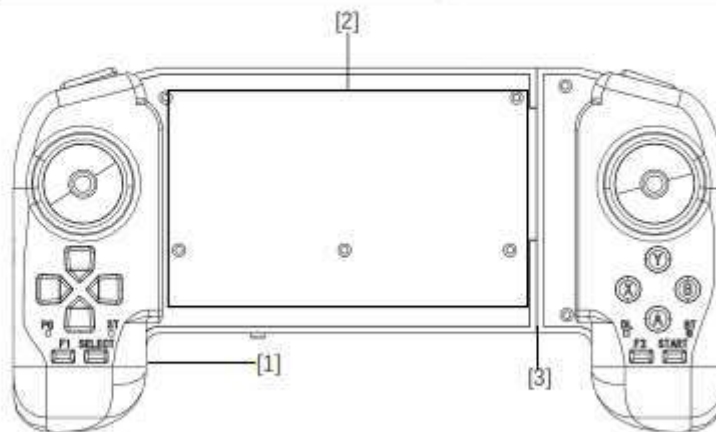
First rotate the legs of the robot to the position shown in Figure [4] above (the rear leg is folded up: rotate the rear leg hip motor so that the rear thigh is placed in the position shown in the above figure [4] above, and the lower leg is closed. Place it in the position shown in Figure [4] above, and tie the left rear leg and the right rear leg together with a Velcro strap, as shown in Figure [3] above. The front leg is folded up: similar to the hind leg, as shown above Figure [3], [4] above placed and bundled).

Packing

After completing the preparation work before packing, load the robot into the lower case [2] in the direction shown in the figure (note that the two rear thighs of the robot are respectively engaged in the A and B slots during the loading process. After the robot is loaded into the lower case, place the remote control, charger, etc. that came with the product into the corresponding positions in the upper case and fix them with elastic straps, make sure the above parts do not fall when the upper case is closed.

 •if not using the robot for a long time, take out the battery pack and put it into battery box.

Part Name of the Remote Control

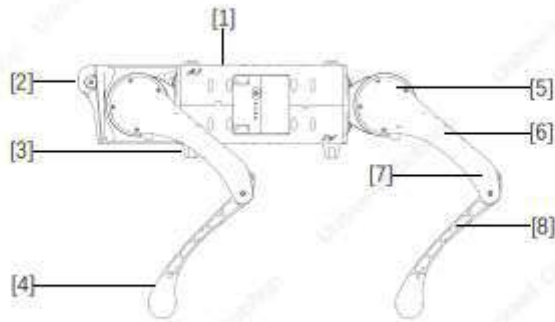


[1] Joystick

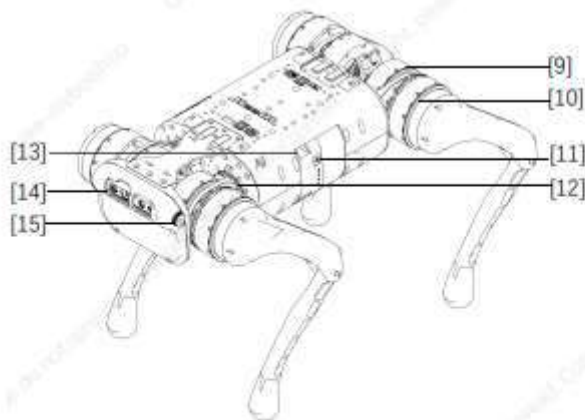
[2] Mobile phone

[3] Rocker (pull out the telescopic rod when using, take it out, and **put it back when it is transport or not used for a long time**)

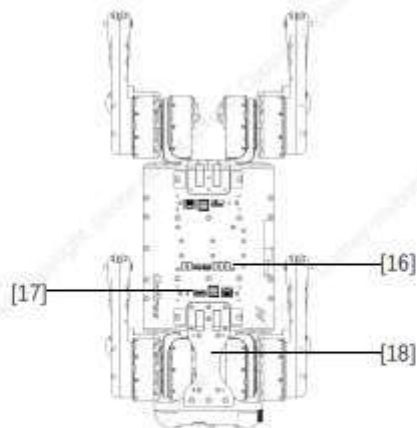
Robot Part Name



- [1] Body
- [2] Front cover
- [3] Abdominal support pad
- [4] Foot assembly
- [5] Hip joint
- [6] Thigh
- [7] Knee joint
- [8] Calf



- [9] Thigh motor & Reducer
- [10] Calf motor & Reducer
- [11] Power switch
- [12] Body motor & Reducer
- [13] Battery clasp
- [14] Multi-Eye depth camera
- [15] Camera angle adjustment knob



- [16] Developing reserved power interfaces
- [17] Developing reserved communication interfaces
- [18] Front suspension plate (Reserved threaded hole, can be equipped with laser radar and other components)

Prepare Before Starting Up

Install the Battery Pack

Insert the battery pack into the battery slot from the side of the robot, and pay attention to the installation direction. If the battery cannot be completely inserted, please adjust the battery direction and do not press it forcibly to avoid damaging the battery interface and Clasp.

💡 • Recommended that the battery be fully charged before use robot.

Body Placement

Horizontal boot: please make sure that the robot is placed on the leveling ground before starting the machine. The robot's abdominal support pad should be flat on the ground. The body level is not tilted on the ground. The robot calf is fully stowed (As shown below) , make sure that the robot's thighs and calves are not pressed by the body, otherwise the robot may fail to boot.



Connect the remote control module

The remote control module includes a joystick and mobile phone. First press the power button for a short time, and then press and hold the power button for more than 2 seconds to turn on the joystick. The joystick corresponds to the robot data transmission module one by one, and it can be automatically connected after starting. The data transmission signal light on the left side of the handle is fully lit to control the robot; at the same time, the Bluetooth of the mobile phone can be turned on, and the handle can be connected through the APP to view the connection status of the handle and the robot.

The robot dog launches 5G WIFI hotspot for image transmission connection. Enter the phone settings, turn on the WLAN, and find the corresponding WIFI hotspot, namely **UnitreeRoboticsA1-XXX**, XXX corresponds to the robot dog code. Click to connect, enter the **password 00000000**, the image transmission function is opened after the connection is successful, and the characters can follow and autonomous navigation operations.

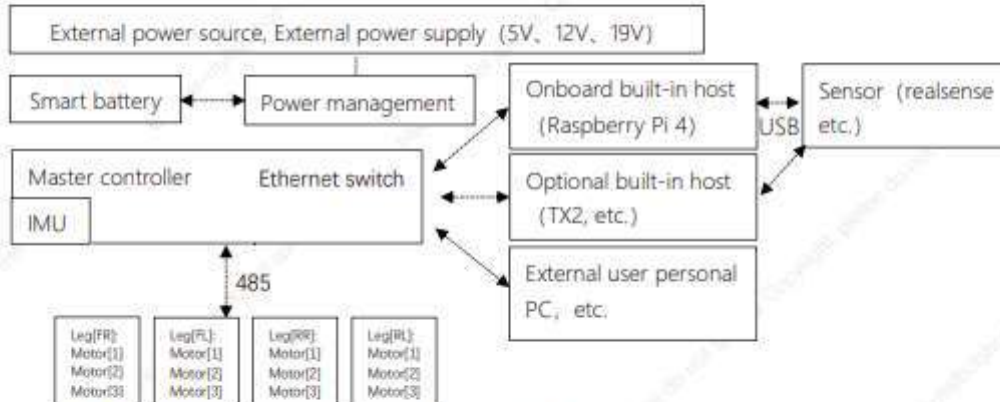
A1 Quadruped Robot

Overview

The A1 consists mainly of control systems, communication systems, power systems, and battery packs. This chapter will detail the functions of the various components of the robot.

Robot Operating Mode

A1 uses a new hardware architecture and control system, control system as follows:



supports the following operating states and operating modes:

Static Standing State:

The static standing state refers to the initial height of the robot body position after the start-up, the body level, and the joystick has no operation state. In this state, the power consumption of the whole machine is the smallest, and the longest battery life can be achieved.

A Mode (3-Axis Attitude And 3-Axis Position Control While Standing) :

When the robot is standing, the joystick can be used to control the robot in 3-axis attitude and 3-axis position. Including the body pitch, body roll, body yaw, continuous squat, jog-type squat, continuous standing, jog-up, and various combinations.

W Mode (Walking) :

When the robot is in W mode, the robot can realize the step by step without manipulating the joystick; the joystick can realize the forward and backward movement, the left and right side shift, the in-situ turning, and the walking according to certain rules on the flat ground (straight line, circle, arc, Rectangular), crawl forward, up and down slope/step, large-scale Push Recovery, etc., also has super adaptability to irregular terrain, the maximum walking speed is 3.3 m / s.

Sport Mode :

When the robot dog is standing (in A mode), holding down the L2 key, single-click the B key, the robot dog squats and lays on the ground, enters the damping mode; then hold down the L1 key, single-click

the START key, start the robot dog. In sport mode, there will be obvious current sound, and the robot dog will stand up again; release the L1 key and click START once to enter the sport mode. In the motion mode, the robot has a strong adaptability to irregular terrain, and the maximum motion speed is 3.3 m/s.



• Since the actual control personnel have different levels of control proficiency, in order to be reliable and stable, please use it in an open and flat environment. When operating the robot, be careful to avoid steps above 5cm, slopes greater than 25°, and obstacles that may cause the robot to fall. When the robot is walking on a terrain with a certain undulation or slope, the controller should reduce the walking speed of the robot.

• The robot has certain requirements for the ground to walk. Do not use robot on the ground with very low friction, such as ice. Do not use robot on soft ground, such as thicker sponge floors. For use on smoother floors, such as glass, tiles, etc., carefully and compliantly control the robot to exercise, avoid strenuous exercise, and reduce the walking speed of the robot to prevent the robot's foot from slipping and falling.



• Through the START key on the joystick, the robot can switch between static standing mode and W mode. The L2 + START key enables the robot to switch between static standing mode and trotting mode.

Startup and Shutdown

Startup

After placing the robot according to the requirements in the "Preparing Before starting up" section, start the following steps: short press the power switch once, then press and hold the power switch for more than 2 seconds to turn on the battery (when the battery is turned on, the indicator light is the green light is always on and the indicator shows the current battery level). Then the robot will perform the power-on self-test. If the self-test is successful, the robot will stand up to the initial height of the body, and the boot is successful. If the robot is not stand up during the above process, the robot fails the self-test. If the boot fails, the robot can't stand up. At this time, you need to re-start the body according to the two steps of "body placement" in the "Preparing Before Starting Up" section.

Shutdown

Before shutting down, please make sure that the robot stands on the level of the ground, make sure that the robot is in Static Standing State (the height of the robot body is at the initial height after starting up, the body level, the joystick has no operation, the state when standing statically). Press and hold the handle L2 button, then click the A button three times, the robot will then complete the squat, stand up, and lie down; then hold down the handle L2 button, and then click the B button twice, the robot then completes prone (Damping), prone (undamped) action; after the robot enters the prone (undamped)

state, press the power switch once, then press the power switch for more than 2 seconds to turn off battery. When the battery is turned off, the indicators are off. After shutting down, please adjust the size of the robot's size legs and hips according to the requirements in the "Preparation before starting" section, and prepare for the next boot.

After shutting down, please adjust the size of the robot's size legs and hips according to the requirements in the "Preparation before starting" section, and prepare for the next boot.



- The step of shut down will be described in the "Joystick Operation" section.
- Please pay attention to Unitree's official website - service and support - technical support - firmware update or contact Unitree staff. We will develop a one-button shutdown function as soon as possible, and upload the firmware. At that time, the customer can update the firmware to make the robot have a one-button shutdown function.

Battery Pack

Introduction

The battery pack is designed for the A1 with a capacity of 4200 mAh and a voltage of 21.6 V with charge and discharge management. The battery pack features a high-performance battery and uses the advanced battery management system developed by Unitree to provide sufficient power for the A1. The battery pack must be recharged using a dedicated charger from Unitree.



- Be sure to fully charge the battery before using it for the first time.

Battery Pack Function

The battery pack has the following features:

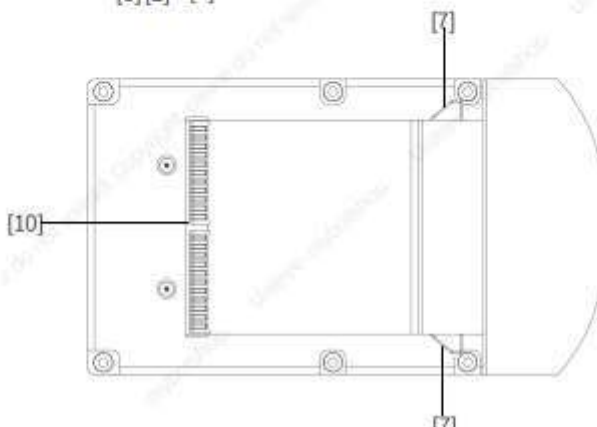
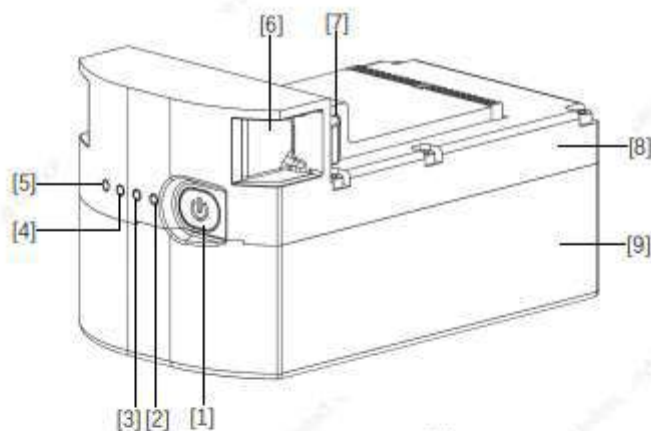
- 1. Battery Display:** The battery has its own battery indicator, which can display the current battery level.
- 2. Battery Storage Self-discharge Protection:** When the power of battery is higher than 65% After 10 days without any operation, the battery can start self-discharge to 65% to protect the battery. Each self-discharge process lasts for about 1 hour. There is no LED light indication during discharge, and there may be slight heat, which is normal.
- 3. Balanced Charge Protection:** Automatically balance the internal battery voltage of the battery to protect the battery.
- 4. Overcharge Protection:** Overcharging can seriously damage the battery, our battery can automatically stop charging when it is fully charged.
- 5. Charging Temperature Protection:** Charging will damage the battery when the battery temperature is

below 5 °C or above 55 °C. At this temperature, the battery will trigger charging abnormality.

- 6. **Charging Current Protection:** High current charging will seriously damage the battery. When the charging current is greater than 4A, the battery will stop charging.
- 7. **Over-discharge Protection:** Over-discharge will seriously damage the battery. When the battery is discharged to 18V, the battery will cut off the output.
- 8. **Short Circuit Protection:** When the battery detects a short circuit, the output will be cut off to protect the battery.
- 9. **Battery Load Detection Protection:** When the battery is turned on, if no powered device is connected, the battery will automatically shut down after 3 seconds.

⚠️ •Please read and strictly follow requirements of Unitree in this manual, disclaimer, the sticker on battery pack surface and dedicated charger surface before using the battery pack. The consequences of failure to use as required are borne by the user.

Battery Pack Part Name



[1] Power switch

[2] LED1

[3] LED2

[4] LED3

[5] LED4

[6] Buckle

[7] Clip

[8] Battery pack panel

[9] Battery pack base plate

[10] Battery interface:

Battery pack charger interface.

Turn the Battery On/Off

Turn on the Battery Pack: In the off state, first press the power switch once, then press and hold the power switch for more than 2 seconds to turn on the battery. When the battery is turned on, the power indicator is steady green and the battery indicator shows the current battery level.


Turn off the Battery Pack: In the on state, short press the power switch once, then press and hold the power switch for more than 2 seconds to turn off the battery. When the battery is turned off, the indicators are off.

Precautions For Use:

1. The battery pack should be used between 5 °C and 40 °C, and the temperature is too high (above 45 °C), which may cause the battery pack to catch fire or even explode. If the temperature is too low (below 0 °C), the battery pack life will be seriously damaged.
2. Do not use the battery pack in strong magnetic or static environments. Otherwise, the battery pack protection board will malfunction, causing the battery pack and the robot to malfunction.
3. When the battery pack charge is less than two compartments, stop using the robot as soon as possible, replace the new battery pack or charge the battery pack.
4. Before inserting or removing the battery pack into the robot battery compartment, make sure that the battery pack is closed, otherwise the battery pack or the robot may be damaged.


View Battery





































When the battery pack is off, press the battery switch once to view the current battery level.

 The battery indicator can be used to display the battery level during charging and discharging of the battery pack. The indicators are defined as follows.

 Indicates that the LED light is always on

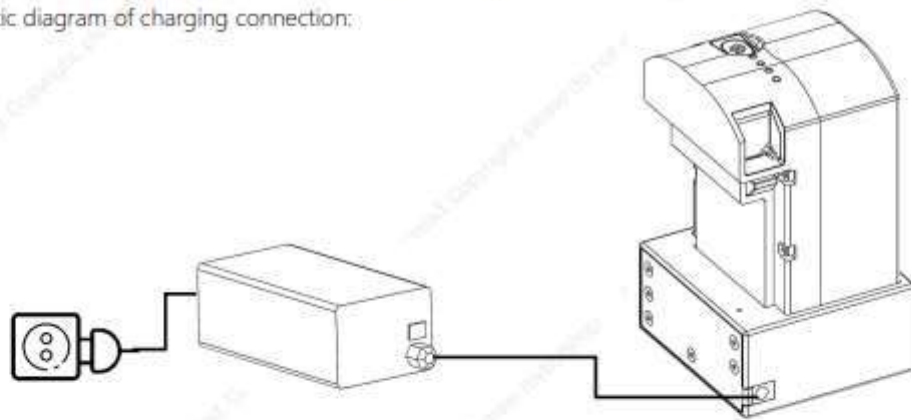
 Indicates that the LED light is flashing

 Indicates that the LED light is off

Battery Level LEDs				
LED1	LED2	LED3	LED4	Remaining Battery
				87.5%-100%
				75%-87.5%
				62.5%-75%
				50%-62.5%
				37.5%-50%
				25%-37.5%
				12.5%-25%
				0%-12.5%
				=0%

Charging

1. Connect the charger to an AC power source (100-240V, 50/60Hz). Before connecting, you must ensure that the external power supply voltage matches the rated input voltage of the charger, otherwise the charger will be damaged (the rated input voltage of the charger is indicated on the charger nameplate).
2. Before charging the battery, the charger is connected to the AC power supply before the battery is connected.
3. Before charging the battery, please make sure the battery pack is off, otherwise it will damage the battery and charger.
4. Under the charging state, battery indicator will flash in a cycle and indicate the current battery level.
5. When the battery indicator is off, the battery pack is full. Please remove the battery pack and charger to complete the charge.
6. After the robot is running, the battery pack temperature may be high. The battery pack must be charged after the battery pack temperature drops to room temperature.
7. Schematic diagram of charging connection:



Charging Indicator				
LED1	LED2	LED3	LED4	Current Battery
●	○	○	○	0%-25%
●	●	○	○	25%-50%
●	●	●	○	50%-75%
●	●	●	●	75%-100%
				Full

Charging Protection Indication

The LED can display information about battery protection triggered by abnormal charging.

Charging Indicator					
LED1	LED2	LED3	LED4	Display Rule	Protection Project
○	⦿	○	○	LED2: 2 times/sec	Excessive charging current
○	⦿	○	○	LED2: 3 times/sec	Short circuit
○	○	⦿	○	LED3: 2 times/sec	Overcharge causes battery voltage be too high
○	○	⦿	○	LED3: 3 times/sec	Charger voltage is too high
○	○	○	⦿	LED4: 2 times/sec	Charging temperature is too low
○	○	○	⦿	LED4: 3 times/sec	Charging temperature is too high

Troubleshoot (charge current is too large, charge short circuit, charge overcharge causes battery voltage is too high, charging voltage is too high), please re-plug the charger to resume charging. If the charging temperature is abnormal, please unplug the charger first. After the charging temperature returns to normal, plug in the charger and recharge.



- For safety reasons, the battery needs to be discharged during transportation. The discharge mode is divided into active discharge and passive discharge:
 - 1.Active discharge: Run the robot until the battery is at a low battery (eg 50% or less).
 - 2.Passive Discharge: The battery is stored in self-discharge protection. For details, please refer to the chapter "Battery Pack" - "Battery Pack Function".

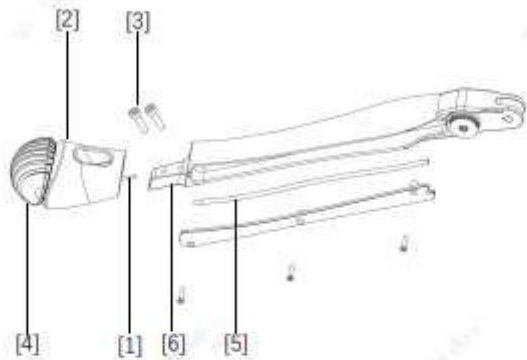
Foot Assembly

Introduction

The foot assembly adopts a new design. The movement of the robot will compress the air in the foot pad and send the pressure signal through the air pipe to the pressure sensor in the shoulder, so as to judge the environment the robot is in and adjust the movement of the robot accordingly.

Foot end components include foot end base, bottom curved rubber pad, air needle and other components. It can increase the friction of the foot, avoid the damage of the foot, and reduce the impact on mechanical parts.

The foot end components are consumables, and the service life is generally 2-6 months (depending on the use frequency, duration and working condition). Especially in the rough ground running wear will be more serious, such as obvious foot pad wear, damage, or found that the robot walking on the ground significantly increased impact noise, please replace the foot components in time, so as not to damage the foot, resulting in the robot movement disorder.



- [1] Air Needle
- [2] Foot Protection Cover
- [3] Foot Cap Screws
- [4] Rubber Foot Pad
- [5] Rubber Trachea
- [6] Air Needle and Trachea Connection Slots

Foot Assembly Replacement Method

Remove the [3] foot cap screws, and gently rotate the [1] air needle and [5] rubber trachea connected in the [6] connection slots; Attach the new foot assembly air needle to the rubber trachea (make sure the connection does not loosen easily), and then install the foot end component in the corresponding position of the lower leg (note that the foot end component is divided into left and right parts, the directions are different), and then tighten the screws.

Jetson Nano



Bringing AI to millions of new systems at the edge

The NVIDIA® Jetson Nano™ module is opening amazing new possibilities for edge computing. It delivers up to 472 GFLOPS of accelerated computing, can run many modern neural networks in parallel, and delivers the performance to process data from multiple high-resolution sensors—a requirement for full AI systems. It's also production-ready and supports all popular AI frameworks. This makes Jetson Nano the ideal platform for developing mass market AI products such as AIoT gateways, smart network video recorders and cameras, consumer robots, and optical inspection systems.

The system-on-module is powered by the NVIDIA Maxwell™ GPU with 4 GB of memory, which allows real-time processing of high-resolution inputs. It offers a unique combination of performance, power advantage, and a rich set of I/Os, from high-speed CSI and PCIe to low-speed I2Cs and GPIOs. Plus, it supports multiple diverse sets of sensors to enable a variety of applications with incredible power efficiency, consuming as little as 5 W.

Jetson Nano is supported by NVIDIA JetPack™, which includes a board support package (BSP), Linux OS, NVIDIA CUDA®, cuDNN, and TensorRT™ software libraries for deep learning, computer vision, GPU computing, multimedia processing, and much more. The comprehensive software stack makes AI deployment on autonomous machines fast, reduces complexity, and speeds time to market.

The same JetPack SDK is used across the entire NVIDIA Jetson™ family of products and is fully compatible with NVIDIA's world-leading AI platform for training and deploying AI software.

KEY FEATURES

Jetson Nano module

- 128-core NVIDIA Maxwell GPU
- Quad-core ARM® A57 CPU
- 4 GB 64-bit LPDDR4
- 16 GB eMMC 5.1
- 10/100/1000BASE-T Ethernet

Power

- Voltage Input: 5 V
- Module Power: 5 W-10 W

Environment

- Operating Temperature: -25 °C to 80 °C*
- Storage Temperature: -25 °C to 80 °C
- Humidity: 85% RH, 85°C [non-operational]
- Vibration: Sinusoidal 5 G RMS 10 to 500 Hz, random 2.88 G RMS, 5 to 500 Hz [non-operational]
- Shock: 140 G, half sine 2 ms duration [non-operational]

* See thermal design guide for details.

NVIDIA JETSON NANO MODULE TECHNICAL SPECIFICATIONS

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	16 GB eMMC 5.1
Video Encode	4K @ 30 4x 1080p @ 30 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 (H.264/H.265)
CSI	12 (3x4 or 4x2) lanes MIPI CSI-2 D-PHY 1.1
Connectivity	Gigabit Ethernet
Display	HDMI 2.0, eDP 1.4, DP 1.2 (two simultaneously)
PCIe	1x1/2/4 PCIe Gen2
USB	1x USB 3.0, 3x USB 2.0
Others	I ² C, I ² S, SPI, UART, SD/SDIO, GPIO
Mechanical	69.6 mm x 45 mm 260-pin SODIMM Connector

Learn more at <https://developer.nvidia.com/jetson>

© 2019 NVIDIA Corporation. All rights reserved. NVIDIA, the NVIDIA logo, CUDA, Jetson, Jetson Nano, NVIDIA Maxwell, and TensorRT are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated. ARM, AMBA and ARM Powered are registered trademarks of ARM Limited. Cortex, MPCore and Mali are trademarks of ARM Limited. All other brands or product names are the property of their respective holders. "ARM" is used to represent ARM Holdings plc; its operating company ARM Limited; and the regional subsidiaries ARM Inc.; ARM KK; ARM Korea Limited; ARM Taiwan Limited; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Germany GmbH; ARM Embedded Technologies Pvt. Ltd.; ARM Norway AS and ARM Sweden AB. JUL19

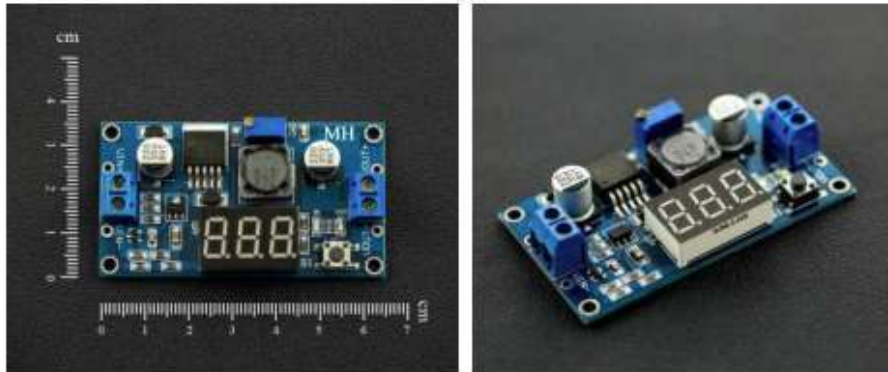


DFR0379 - 20W Adjustable DC-DC Buck Converter with Digital Display



20W Adjustable DC-DC Buck Converter with Digital Display

SKU:DFR0379



INTRODUCTION

This is a 20W adjustable DC-DC buck converter module with digital display. It is based on LM2596 3A step-down voltage regulator and supports an input of 0-40V DC with an accuracy of $\pm 0.05V$.

On a regular buck converter there is no display and you have to measure the output manually with a multimeter, which can be slow and inefficient. This buck converter has a display with the output voltage readout integrated right in to the board. You can change the output by adjusting a screw potentiometer that is also integrated on to the board. Simple!

This module can be used in DC applications such as batteries, power transformers, DIY adjustable power supplies, 24V vehicle power supplies, industrial equipment, 12V to 3.3V, 12V to 5V, 24V to 5V, 24V to 12V, 36V to 24V and so on.

The on-board voltage meter supports self-calibration mode. You only need to calibrate it once and the value will be stored automatically. The method is as follows:

1. Hold the button for 3 seconds. Release the button to enter input voltage calibration mode ("IN" is ON) ; Hold the button for 3 seconds and release the button to enter output voltage calibration mode ("OUT" is ON); hold the button for 3 seconds, and release the button to exit calibration mode, all parameters will be save automatically.
2. In calibration mode, click the button to adjust the value.

FEATURES

- Supports self-calibration function to provide high-precision voltage output. (Recommended input voltage is maintained at 4.5V or more)
- Touch the button to switch the measurement input or output voltage, and an indicator shows which voltage is being measured.
- The display can be disabled if necessary. Hold the button for 2 seconds, and release the button to turn off the display
- With wire terminals, no soldering is necessary
- The input voltage is 4.0 – 40V. (The input voltage must be 1.5V higher than the output voltage)
- Continuously adjustable output voltage range of 1.25V – 37V. (The input voltage must be 1.5V higher than the output voltage)
- The Maximum output current is 3A, it is recommended to use within 2.0A, higher currents will need a heatsink to dissipate heat.
- The output power is 20W. For more than 15W a heatsink is recommended.
- The unit offers high conversion efficiency, with an average of 88%
- The unit includes reverse polarity protection, overheating protection and short circuit protection

SPECIFICATION

- Input Voltage: 4.0 – 40V
- Output Voltage: 1.25V – 37V
- Output Power: 20W
- Output Current: 3A
- Mounting Dimensions: 6.1 * 3.1cm/ 2.4 * 1.22 inches (L x W)
- Dimension: 6.6 * 3.6 * 1.2cm/ 2.59 * 1.42 * 0.47 inches
- Weight: 22g

SHIPPING LIST

- 20W Adjustable DC-DC Buck Converter with Digital Display x1

SWITCH TOGGLE SPDT 5A 120V

SERIES 100 SWITCHES TOGGLE SWITCHES - MINIATURE



SPECIFICATIONS	
Contact Rating:	See contact material options
Life Expectancy:	40,000 make-and-break cycles
Contact Resistance:	10 mΩ max. typical initial @ 2-4 VDC 100 mA for both silver and gold plated contacts
Insulation Resistance:	1,000 MΩ min.
Dielectric Strength:	1,000 V RMS @ sea level
Operating Temperature:	-30° C to 85° C

MATERIALS	
Case:	Diallyl Phthalate (DAP)
Toggle Handle:	Brass, chrome plated
Switch Support:	Brass or steel, tinplated
Bushing:	Brass, nickel plated
Housing:	Stainless steel
Contacts / Terminals:	Silver or gold plated copper alloy

- | | |
|--|--|
| FEATURES & BENEFITS <ul style="list-style-type: none"> ▶ Up to 4 poles available ▶ Variety of switching functions ▶ Miniature ▶ Multiple actuator & bushing options | APPLICATIONS/MARKETS <ul style="list-style-type: none"> ▶ Telecommunications ▶ Instrumentation ▶ Networking ▶ Medical equipment |
|--|--|



HOW TO ORDER

SERIES	MODEL NO.	ACTUATOR	BUSHING	TERMINATION	CONTACT MATERIAL	SEAL	HARDWARE
100	SP1	T1	B1	M1	Q - Silver	E - Epoxy Sealed at Base of Terminal	H - Hardware
	SP2*	T2	B2	M2	R - Gold		
	SP3	T3	B3	M3			
	SP4*	T4	B4	M4			
	SP5*	T5	B5	M5			
	DP1	T6	B6	M6			
	DP2*	T7	B7	M7			
	DP3	T8	B8	M8			
	DP4*	T9	B11	V2			
	DP5*	K1	B12	V3			
	DP6	K2	B13	V3*			
	DP7*		B15	M4*			
	DP8*	Optional Toggle Cap (T1 Only)	B16	V21*			
	SP1	T100-1 - White	B25**	V31*			
	SP2	T100-2 - Black	B26**				
	SP4*						
	SP5*						
	DP1						
	DP2*						
	DP3						
	DP4*						
	DP5*						
	DP6						
	DP7*						
	DP8*						



Example Ordering Number
100-SP1-T4-B2-M1-R-E

Notes: * Not available with the K1 and K2 actuator options.
** Available only with the K1 and K2 actuator options.

Specifications subject to change without notice.



SPDT

KEYWAY
25°
POS 3
POS 2
POS 1
115 DIA
2.92#
Epoxy Seal

410 350 505 12.83
10.41 8.89 3.50 0.30
12.83 8.89
185 4.70
0.76

KEYWAY THIS SIDE

Model No.	POS 1	POS 2	POS 3
SP-1	ON	NONE	ON
SP-2	ON	NONE	ON
SP-3	ON	OFF	ON
SP-4 (ON)	OFF	OFF	ON
SP-5	ON	OFF	ON
Term. Conn.	2-3	OPEN	2-1

() - Momentary

2 Comm. 5
1 2 3

SCHMATIC

DPDT

KEYWAY
25°
POS 3
POS 2
POS 1
115 DIA
2.92#
Epoxy Seal

410 350 505 12.83
10.41 8.89 3.50 0.30
12.83 8.89
185 4.70
0.76

KEYWAY THIS SIDE

Model No.	POS 1	POS 2	POS 3
DP-1	ON	NONE	ON
DP-2	ON	NONE	ON
DP-3	ON	OFF	ON
DP-4 (ON)	OFF	OFF	ON
DP-5	ON	OFF	ON
DP-7	ON	ON	ON
DP-8 (ON)	ON	ON	ON
Term. Conn.	2,3,5-6	OPEN	2,1,5,4

() - Momentary

2 Comm. 5 Comm.
1 2 3 4 5 6

SCHMATIC

3PDT

KEYWAY
25°
POS 3
POS 2
POS 1
115 DIA
2.92#
Epoxy Seal

410 350 571 0.60
10.41 8.89 14.50 1.52
4.15 0.30
10.70 0.76

KEYWAY THIS SIDE

Model No.	POS 1	POS 2	POS 3
3P-1	ON	NONE	ON
3P-2	ON	NONE	ON
3P-3	ON	OFF	ON
3P-4 (ON)	OFF	OFF	ON
3P-5	ON	OFF	ON
Term. Conn.	2,3,5-6 8-9	OPEN	2,1,5,4 8,7

() - Momentary

2 Comm. 5 Comm. 8 Comm.
1 2 3 4 5 6 7 8 9

SCHMATIC

4PDT

KEYWAY
25°
POS 3
POS 2
POS 1
115 DIA
2.92#
Epoxy Seal

410 350 571 0.60
10.41 8.89 14.50 1.52
4.15 0.30
10.70 0.76

KEYWAY THIS SIDE

Model No.	POS 1	POS 2	POS 3
4P-1	ON	NONE	ON
4P-2	ON	NONE	ON
4P-3	ON	OFF	ON
4P-4 (ON)	OFF	OFF	ON
4P-5	ON	OFF	ON
4P-7	ON	ON	ON
4P-8 (ON)	ON	ON	ON
Term. Conn.	2,3,5-6 8-9 11-12	OPEN	2,1,5,4 8,7 11-10

() - Momentary

2 Comm. 5 Comm. 8 Comm. 11 Comm.
1 2 3 4 5 6 7 8 9 10 11 12

SCHMATIC

TACT. SWITCHES

NAVIGATOR SWITCHES

POSITION SWITCHES

TOGGLE SWITCHES

ROCKER SWITCHES

SLIDE SWITCHES

EMERGENCY ACTION SWITCHES

DIP SWITCHES

RELEASER SWITCHES

ALARM SWITCHES

DEFECTOR SWITCHES

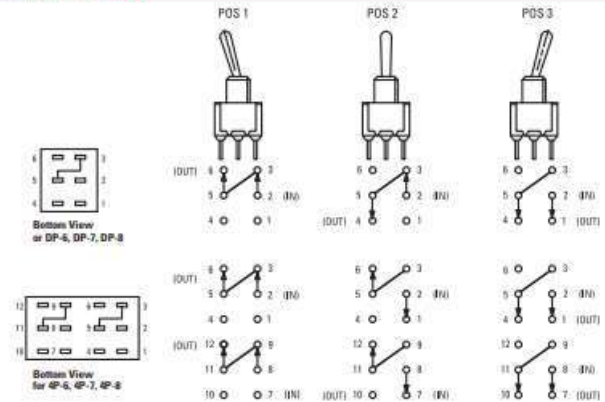
LEAP OPTIONS

SERIES 100 SWITCHES

TOGGLE SWITCHES - MINIATURE

- FACT SWITCHES
- NAVIGATION SWITCHES
- MISDICTION SWITCHES
- TOGGLE SWITCHES**
- ROCKER SWITCHES
- SLIDE SWITCHES
- EMERGENCY SWITCHES
- DP SWITCHES
- SYSTEM SWITCHES
- ROCKY SWITCHES
- DESTROY SWITCHES
- CAP OPTIONS

3 - WAY WIRING DIAGRAM SCHEMATICS



ACTUATOR OPTIONS

<p>T1⁺ STANDARD</p>	<p>T2⁺</p>	<p>T3⁺</p>	<p>T4⁺</p>
<p>T5⁺</p>	<p>T6 FLATTED WITH BUILT IN ANTIROTATION</p>	<p>T7</p>	<p>T8</p>
<p>T9⁺ PLASTIC BLACK</p>	<p>K1</p>	<p>K2</p>	

*Add .070 (1.78) for B3, B4 bushings, subtract .020 (0.51) for B6 bushing.

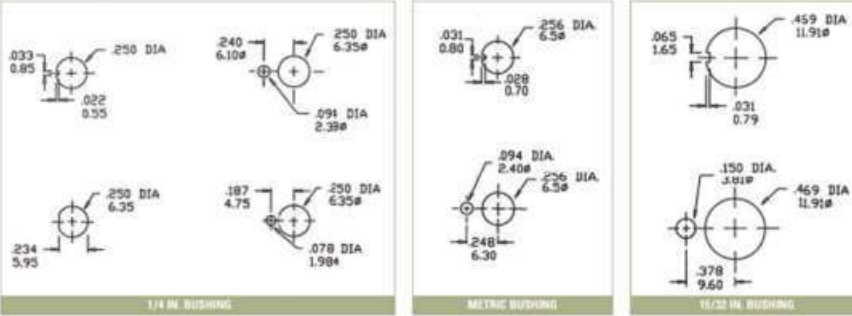


SERIES 100 SWITCHES

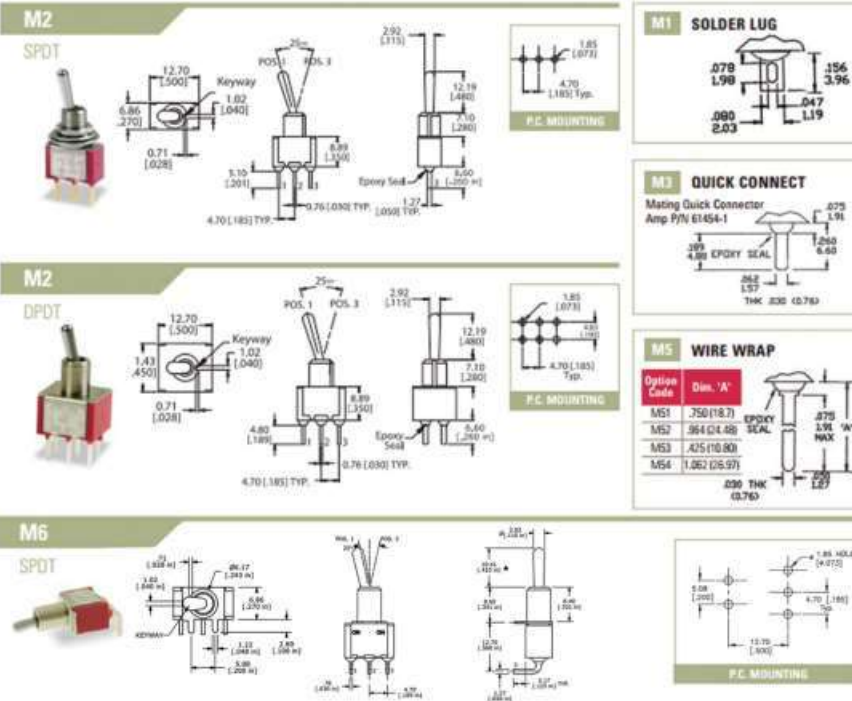
TOGGLE SWITCHES - MINIATURE

FACT SWITCHES
 NAVIGATION SWITCHES
 POSITIONER SWITCHES
TOGGLE SWITCHES
 ROTARY SWITCHES
 SLIDE SWITCHES
 MAP-ACTION SWITCHES
 DIP SWITCHES
 REVERSE SWITCHES
 ROTARY SWITCHES
 DETECTOR SWITCHES
 GAP OPTIONS

BUSHING OPTIONS - PANEL MOUNTING


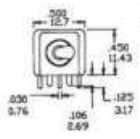
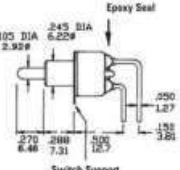
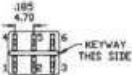
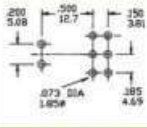

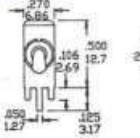
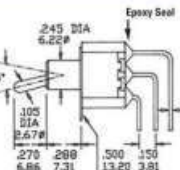

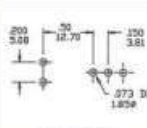

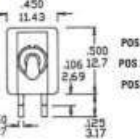
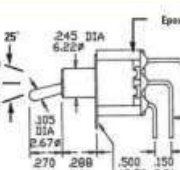
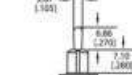
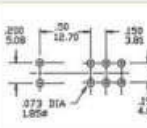

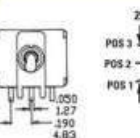
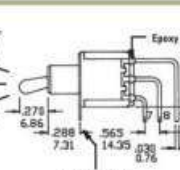

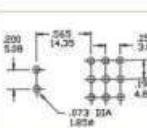

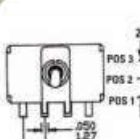
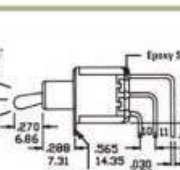
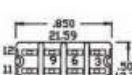
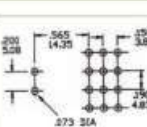


TERMINATION OPTIONS





E-SWITCH®

<p>M6</p> <p>DPDT</p> 	  <p>Switch Support</p>	 <p>KEYWAY THIS SIZE</p> <p>TERM. NOS. FOR REFERENCE ONLY</p>	 <p>PC MOUNTING</p>	TACT SWITCHES NAVIGATION SWITCHES FULLBRIGHT SWITCHES
<p>M7</p> <p>SPDT</p> 	  <p>Switch Support</p>	 <p>KEYWAY THIS SIDE</p> <p>TERM. NOS. FOR REFERENCE ONLY</p>	 <p>PC MOUNTING</p>	TOGGLE SWITCHES MOBILE SWITCHES
<p>M7</p> <p>DPDT</p> 	  <p>Switch Support</p>	 <p>EPOXY SEAL</p> <p>1.27 X 0.70 THK (0.00 X 0.00) (3 PL)</p>	 <p>PC MOUNTING</p>	SLIDE SWITCHES SNAP-ACTION SWITCHES
<p>M7</p> <p>3PDT</p> 	  <p>Switch Support</p>	 <p>TERM. NOS. FOR REFERENCE ONLY</p>	 <p>PC MOUNTING</p>	DIP SWITCHES REVERSE SWITCHES
<p>M7</p> <p>4PDT</p> 	  <p>Switch Support</p>	 <p>TERM. NOS. FOR REFERENCE ONLY</p>	 <p>PC MOUNTING</p>	MEMORY SWITCHES DETECTOR SWITCHES CAP OPTIC

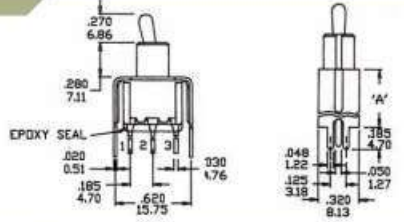
SERIES 100 SWITCHES

TOGGLE SWITCHES - MINIATURE

- FACT SWITCHES
- NAVIGATION SWITCHES
- POSITION SWITCHES
- TOGGLE SWITCHES
- BUCKER SWITCHES
- SLIDE SWITCHES
- SEMI-ACTION SWITCHES
- IMP SWITCHES
- KEYLOCK SWITCHES
- MEMORY SWITCHES
- DETECTION SWITCHES
- CAP OPTIMIZER

VS2-VS3

SPDT

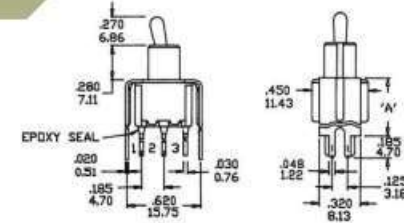


Option Code	Dim. 'A'
VS2	.460 (11.68)
VS3	.530 (13.50)

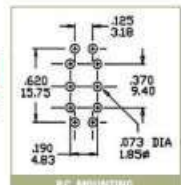


VS2-VS3

DPDT

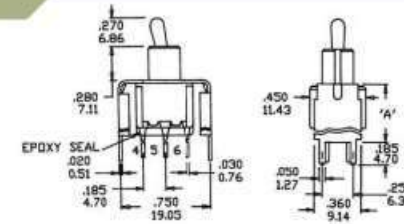


Option Code	Dim. 'A'
VS2	.460 (11.68)
VS3	.530 (13.50)

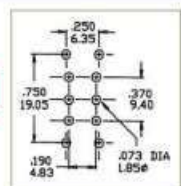


VS5

DPDT

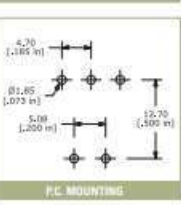
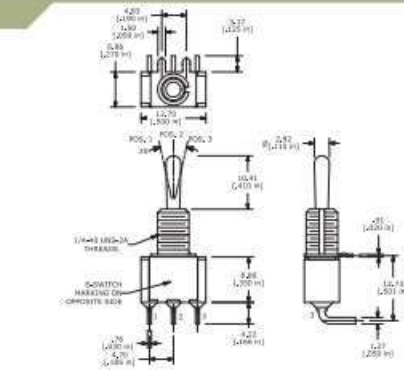


Option Code	Dim. 'A'
VS5	.530 (13.50)



M61

SPDT





HARDWARE

<p>HDW2 1/4 IN. NUTS</p> <p>1/4-40UNS</p> <p>Supplied standard with B1, B3, B5, B6 and B9 options</p>	<p>HDW3 BLIND DRESSNUT</p> <p>1/4-40 UNS THD</p> <p>Supplied standard with B8 metric bushing</p>	<p>HDW6 METRIC NUT</p> <p>M6P 0.75 THD</p> <p>Supplied standard with B8 metric bushing</p>	<p>HDW11 LOCK WASHER</p> <p>Supplied standard with B1/B3 bushing</p>
<p>Optional Rubber Hood Sealing E1</p> <p>E1- Inch (Std.) for actuator T1, T2 and bushings B1, B3, B5, B6 Optional rubber boot sealing</p>	<p>HDW4 15/32 IN. NUTS</p> <p>15/32-32 UN</p> <p>Supplied standard with B13 bushing</p>	<p>HDW12 LOCK WASHER</p> <p>Supplied standard with B13 bushing</p>	<p>HDW15 LOCKING RING</p> <p>Supplied standard</p>
<p>HDW16 LOCKING RING</p> <p>Supplied with B5 bushing</p>	<p>HDW17 LOCKING RING</p> <p>Supplied with B13 bushing</p>		

FACTORY ASSEMBLIES
 MANIPULATOR SWITCHES
 PULLMOTION SWITCHES
TOGGLE SWITCHES
 MOBILE SWITCHES
 BLIND SWITCHES
 SIGNAL ACTION SWITCHES
 2P SWITCHES
 KEYLOCK SWITCHES
 ROTARY SWITCHES
 DETECTOR SWITCHES
 CAP OPTICS

Intel Realsense

(*Limited pages were chosen due to file size)



Intel® RealSense™ D400 Series Product Family

Datasheet

Intel® RealSense™ Vision Processor D4, Intel® RealSense™ Vision Processor D4 Board, Intel® RealSense™ Depth Module D400, Intel® RealSense™ Depth Module D410, Intel® RealSense™ Depth Module D415, Intel® RealSense™ Depth Camera D415, Intel® RealSense™ Depth Module D420, Intel® RealSense™ Depth Module D430, Intel® RealSense™ Depth Camera D435, Intel® RealSense™ Depth Camera D435i

Revision 005

January 2019



Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at Intel.com.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

Intel and the Intel logo, Intel® Core™, Intel® Atom™, trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2019 Intel Corporation. All rights reserved.

Description and Features

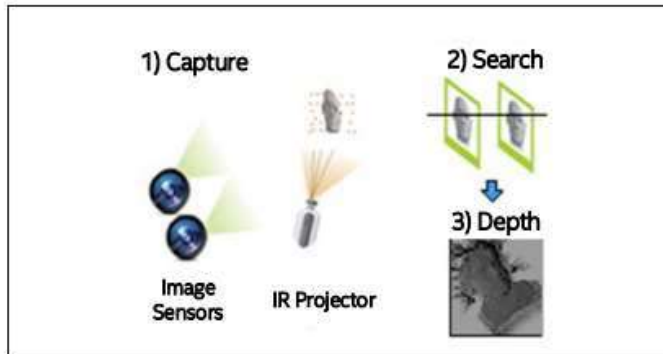


1 Description and Features

<p>Description</p> <p>The Intel® RealSense™ D400 series is a stereo vision depth camera system. The subsystem assembly contains stereo depth module and vision processor with USB 2.0/USB 3.1 Gen 1 or MIPI¹ connection to host processor.</p> <p>The small size and ease of integration of the camera sub system provides system integrators flexibility to design into a wide range of products.</p> <p>The Intel® RealSense™ D400 series also offers complete depth cameras integrating vision processor, stereo depth module, RGB sensor with color image signal processing and Inertial Measurement Unit² (IMU). The depth cameras are designed for easy setup and portability making them ideal for makers, educators, hardware prototypes and software development.</p> <p>The Intel® RealSense™ D400 series is supported with cross-platform and open source Intel® RealSense™ SDK 2.0</p> <p>Features</p> <ul style="list-style-type: none"> • 2nd Generation Stereo Depth Camera System • 2nd Generation dedicated Intel® RealSense™ Vision Processor D4 with advanced algorithms • Infrared (IR) Laser Projector System (Class 1) • Full HD resolution Image sensors • Active Power Management • Selection of Stereo Depth Module options to meet your usage requirements <ol style="list-style-type: none"> 1. MIPI is not currently supported. Please contact your Intel representative on MIPI enablement timelines. 2. Camera SKU dependent 	<p>Usages/ Markets</p> <ul style="list-style-type: none"> • Drones • Robots • Home and Surveillance • Virtual Reality • PC Peripherals <p>Minimum System Requirements</p> <p>USB 2.0/USB 3.1 Gen 1 Ubuntu*16.xx/Windows*10</p> <p>Intel® RealSense™ Depth Camera D415 Features</p> <ul style="list-style-type: none"> • Intel® RealSense™ Vision Processor D4 • Up to 1280x720 active stereo depth resolution • Up to 1920x1080 RGB resolution • Depth Diagonal Field of View over 70° • Dual rolling shutter sensors for up to 90 FPS depth streaming • Range 0.3m to over 10m (Varies with lighting conditions) <p>Intel® RealSense™ Depth Camera D435/D435i Features</p> <ul style="list-style-type: none"> • Intel® RealSense™ Vision Processor D4 • Up to 1280x720 active stereo depth resolution • Up to 1920x1080 RGB resolution • Depth Diagonal Field of View over 90° • Dual global shutter sensors for up to 90 FPS depth streaming • Range 0.2m to over 10m (Varies with lighting conditions) • Intel® RealSense™ Depth Camera D435i includes Inertial Measurement Unit (IMU) for 6 degrees of freedom (6DoF) data
--	--

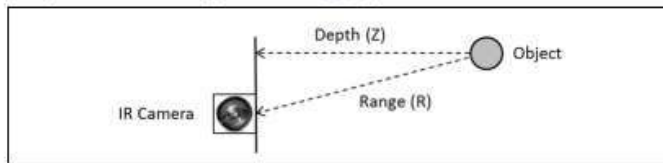


Figure 2-1. Active Infrared (IR) Stereo Vision Technology



The depth pixel value is a measurement from the parallel plane of the imagers and not the absolute range as illustrated.

Figure 2-2. Depth Measurement (Z) versus Range (R)

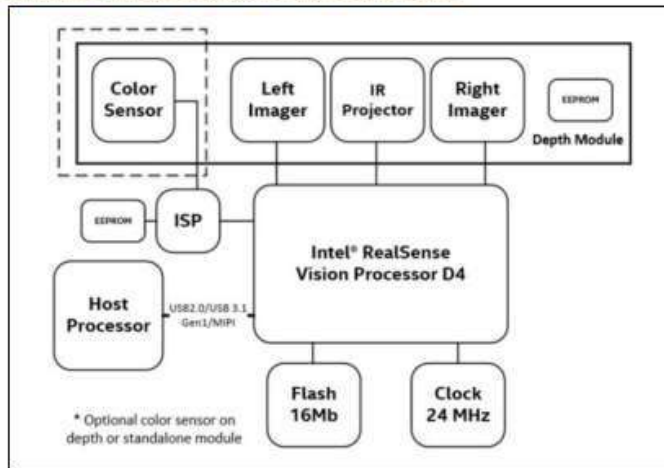




2.4 Camera System Block Diagram

The camera system has two main components, Vision processor D4 and Depth module. The Vision processor D4 is either on the host processor motherboard or on a discrete board with either USB2.0/USB 3.1 Gen1 or MIPI connection to the host processor. The Depth module incorporates left and right imagers for stereo vision with the optional IR projector and RGB color sensor. The RGB color sensor data is sent to vision processor D4 via the color Image Signal Processor (ISP) on Host Processor motherboard or D4 Board.

Figure 2-3. Vision Processor D4 Camera System Block Diagram



3 Component Specification

3.1 Vision Processor D4 Camera System Components

Table 3-1. Component Descriptions

Component	Description
Host Processor	Host Processor that receives Depth and other data streams from Vision Processor D4
Vision Processor D4 (DS5 ASIC)	Depth Imaging Processor with USB 2.0/USB 3.1 Gen 1 or MIPI interface connection to Host Processor
Clock	24MHz clock source for Vision Processor D4
Serial Flash Memory	SPI 16Mb Serial Flash memory for firmware storage
Stereo Depth Module	Camera module with left and Right Imager, Color Sensor ^(†) , IR projector ^(†) enclosed in a stiffener
Power Delivery	Circuitry on motherboard/Vision processor D4 Board to deliver and manage power to Vision Processor D4 and Stereo Depth Module.
Stereo Depth Connector and Interposer	50 pin connector on motherboard/Vision Processor D4 Board and Stereo Depth module with interposer for connection

(†) SKU dependent

3.2 Host Processor

The host processor interface to Vision Processor D4 is either USB 2.0/USB 3.1 Gen 1 or MIPI. To ensure the best of quality of service, the Vision Processor D4 must be connected to a dedicated USB 3.1 Gen 1 root port within the host processor system.

3.3 Intel® RealSense™ Vision Processor D4

The primary function of Vision Processor D4 is to perform depth stereo vision processing. The Vision Processor D4 on Host Processor motherboard or on Vision Processor D4 Board communicates to the host processor through USB2.0/USB 3.1 Gen 1 or MIPI and receives sensor data from stereo depth module. The Vision Processor D4 supports MIPI CSI-2 channels for connection to image sensors.

3.3.1 Vision Processor D4 Features

- 28nm Process Technology.

Appendix F: User Manual

- I. System Activation
 - A. Power on the Unitree A1 Robot. This will power the Jetson Nano which will have a green LED enabled.

- II. System Setup
 - A. Connect the computer to Jetson Nano via HDMI.
 - B. Adjust Parameters (for Jetson Nano) using laptop computer.
 1. Autonomous Operation Mode: user specifies whether they want the robot to perform obstacle avoidance or object following.
 2. Minimum Obstacle Distance: user specifies the minimum distance that the robot must maintain between itself and an obstacle.
 - C. Connect the computer monitor to the robot via HDMI and prepare to execute the movement script in the command terminal. Use a wireless keyboard and mouse to interface with the robot when connected via HDMI.
 - D. Remove all external connections so the robot is ready for autonomous operation.

- III. Autonomous Operation
 - A. Orient the robot so that it will be able to travel without immediately facing any obstacles.
 - B. Enable autonomous operation by executing the movement script.
 - C. Monitor system as it proceeds toward destination.

- IV. System Shutdown
 - A. Use wireless keyboard to stop execution of autonomous operation.
 - B. Power down the Unitree A1 Robot.

- V. Troubleshooting
 - A. Unable to power on Unitree A1 robot
 1. Check battery charge
 2. Check robot connections
 - B. Unable to activate Jetson Nano
 1. Observe Jetson Nano LED

2. Check power connection from buck converter
3. Check battery charge

C. Obstacle avoidance failure

1. Toggle killswitch (lift robot in air)
2. Check camera readouts from Jetson Nano
3. Check camera connections

Appendix G: Task Management Document

Team Name	pathVision
Week	2
Manager	Alex Domagala

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Find wifi adapter	Create purchase order for wifi adapter	Emily Hernandez	Complete
2	Install critical libraries/software for Jetson Nano	Installation of openCV, vscode, pytorch on Jetson Nano	Alex Domagala	Complete
3	Contact Jon G regarding buck converters	Determine if the lab has buck converters, and if it doesn't, submit order form	Jonathan Perthel	Complete
4	Contact Robotics lab for access to robot	Have dates where we can go in for testing	Pranay Singh	Complete
5	Establish ethernet connection between jetson and raspi	Send pings between Jetson and Pi	Emily Hernandez	Complete
6	Research stereo camera algorithms and implementations	Have a script started and ready to test by week 4	Emily Hernandez	Complete
7	Research single camera algorithms and implementations	Have a script started and ready to test by week 4	Pranay Singh	Complete
8	Research neural network algorithms and implementations	Find 2 example programs, and create a sample script to test week 4	Jonathan Perthel	Complete
9	Research stereo camera algorithms and implementations	Find at least 2 successful examples of a heatmap generated using stereo cameras	Alex Domagala	Complete
10	Setup github repository	Create organized repository to upload code, ensure teammates can upload code	Alex Domagala	Complete

Team Name	pathVision
Week	3
Manager	Jonathan Perthel

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Finish stereo camera script from previous week	Complete python test script so it runs and is ready to test on hardware for week 4	Emily Hernandez	Incomplete
2	Establish simple data transfer (strings) between jetson and pi using ethernet	Create script to print data on the pi sent over ethernet from the Jetson	Emily Hernandez	Complete
3	Complete single camera script from previous week	Implementation of online scripts and create own for next week	Pranay Singh	Partially Complete
4	Familiarize with robot movement	Go through robot github and understand movement commands and create documentation	Pranay Singh	Partially Complete
5	Learn about the camera capabilities of the robot	Create research document about the camera that is available on the Unitree A1 Robot	Alex Domagala	Complete
6	Stereo Depth Map Generation attempt	Connect stereo camera and generate heatmap on Jetson Nano, document any roadblocks and possible	Alex Domagala	Partially Complete
7	Verify that all required libraries are installed on jetson for testing in week 4	Read requirements for all written scripts and verify that all necessary libraries are installed on Jetson	Jonathan Perthel	Complete
8	Gather training data for neural network prototype 1 (blocked/free)	Gather 200 total images using data. Collection script for training neural network	Jonathan Perthel	Partially Complete
9	Begin training neural network for prototype 1 (blocked/free)	Using the collected images train the model to be tested week 4	Jonathan Perthel	Partially Complete
10				

Team Name	pathVision
Week	4
Manager	Emily Hernandez

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Test stereo camera script(s) from previous week	Make determination on moving forward with current script depending on how well it works	Emily Hernandez	Complete
2	Setup ethernet connections on lab robot	Send pings between Jetson and Pi located on lab robot	Emily Hernandez	Complete
3	Implementing single-camera script from last week	Test the object detection script on Jetson and determine if its feasible to use on robot	Pranay Singh	Complete
4	Add movement files to Github	Finish documentation and add compatible movement files to Github	Pranay Singh	Complete
5	Develop plan for Training environment	Detailed writeup for line follower environment (where training occurs, how will line be drawn)	Alex Domagala	Complete
6	Research obstacle avoidance with line following program	Detailed writeup of how the robot will break off the line and return when obstacle is encountered	Alex Domagala	Complete
7	Order Webcam	Submit Purchase order for webcam	Jonathan Perthel	Complete
8	Finalize Line following data acquisition script	Complete script for gathering training data of following line, add to github and download to jetson	Jonathan Perthel	Complete
9	Begin gathering training data for line following	Test data acquisition script using training environment by collecting 20-30 images	Jonathan Perthel	Incomplete
10				

Team Name	pathVision
Week	5
Manager	Pranay Singh

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Setup repo on robot and get socket scripts working	Send strings between Robot Raspi and Jetson	Emily Hernandez	Complete
2	Find a way to connect to UIC wifi or find a reliable alternative to connect the jetson to a wifi network when in the lab in order to use	Connect Jetson to UIC wifi or find alternative	Emily Hernandez	Complete
3	Get movement command layout	Work with Pranay and Jon to understand documentation	Emily Hernandez	Complete
4	Work on Movement script	Take in the data being fed and convert that to movements	Pranay Singh	Complete
5	Set up Gazebo for the robot	Use the Gazebo files given to try to set up the simulator	Pranay Singh	Partially Complete
6	Stereo Camera test on Jetson using Logitech Cameras	Jetson Nano connected to (2) Logitech Cameras, generates depth map using sd	Alex Domagala	Complete
7	OpenCV manual line following script development	Upload Python script that places bounding boxes on a detected line in image	Alex Domagala	Complete
8	Reflash Jetson	Reflash the Jetson SD Card with Jetbot code preinstalled	Jonathan Perthel	Complete
9	Get Jupyter notebooks to work with jetbot code	Be able to run the jetbot line following scripts	Jonathan Perthel	Complete
10	Get Logitech Camera display working	Use jupyter notebooks to be run one of the jetbot scripts to view camera	Jonathan Perthel	Complete

Team Name	pathVision
Week	6
Manager	Alex Domagala

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Raspi on Robot is not communicating over ethernet properly. Work with Jon G to get the robot working properly	Get ethernet connection fixed/working on robot	Emily Hernandez	Complete
2	We may not be able to get ethernet working so we should look into alternatives so we can implement them asap	Find an alternative to ethernet communication	Emily Hernandez	Complete
3	Testing basic movement script	Script handling basic movement such as moving forward and turning	Pranay Singh	Complete
4	Autonomous ON/OFF switch	See if its possible to dedicate a switch on the controller to change modes	Pranay Singh	Partially Complete
5	Improve stereo camera image feed	Modify current depth map scripts to allow for reliable viewing of polygons	Alex Domagala	Partially Complete
6	Research movement command generation based on Depth map	Document with at least 2 approaches for specifying if object is detected	Alex Domagala	Complete
7	Explore line following using opencv	Use Alex's line following code on jetson. Begin testing functionality at home	Jonathan Perthei	Complete
8	Research Intel Realsense Camera	Research functionality, Research compatibility, decide whether or not to use	Jonathan Perthei	Complete

Team Name	pathVision
Week	7
Manager	Jonathan Perthei

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	The jetson will be powered by the robot. We need to definitively set up a circuit diagram on where we are tapping into the robot to get access to the power and then include the converter as well as decide how to connect to the jetson.	Set up circuit diagram to connect the jetson to the robot and get it reviewed by Jon G. Should also include connector types. Gather materials to prepare to build setup and create setup and test procedure.	Emily Hernandez	Complete
2	There will be manual mode, autonomous, startup, idle, etc. states for the robot, we would enumerate them as well as create an FSM so we can easily switch between them for safety purposes as well as ease of use.	Create FSM diagram for robot states	Emily Hernandez	Complete
3	Ensure the ethernet comms between the devices work consistently. Should be sending strings.	Test communication on the robot and jetson.	Emily Hernandez	Complete
4	Implementation of test movement scripts	Test movement script from previous week	Pranay Singh	Complete
5	Finalize autonomous ON/OFF switch	Continue trying to get an ON/OFF for the autonomous mode working	Pranay Singh	Partially Complete
6	Use research to develop sample command generation script	Develop prototype script that generates commands from depth map	Alex Domagala	Complete
7	Test depth map script on alternate camera module	Document test results, if camera trouble, inquire about intel realsense to Professor Bhounsule	Alex Domagala	Complete
8	Configure opencv line following script	Adjust the opencv script so that it accurately follows a line at home and outputs direction changes	Jonathan Perthei	Complete
9	Begin script to send commands to robot	Using the opencv output, convert outputs to robot specific motion. Begin experimenting with simple left/right	Jonathan Perthei	Partially Complete

Team Name	pathVision
Week	8
Manager	Emily Hernandez

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	From the circuit diagram reviewed last week, build and test the harness to power the Jetson from the robot so we can completely attach the two according to the diagram in the final report. This includes soldering/attaching all connectors and programming the buck converter. In testing we should ensure we are getting the expected voltage output from the robot before plugging in the Jetson.	Build and test the harness for connecting the Jetson to the robot.	Emily Hernandez	Complete
2	Review Alex's research, work with Alex to put together a plan for movement generation workflow, work on overall software architecture/outline.	Develop software outline	Emily Hernandez	Complete
3	Communication Between Python and C++ code	Using the Pi find away for the python code to be read in by the C++ movement code	Pranay Singh	Partially Complete
4	Jetson movement command output.	Create a key for the exact data that is being sent from the Jetson to the Pi	Pranay Singh	Complete
6	Learn about the Intel RealSense Library: librealsense	Collection of sample code that can be deployed once camera order arrives	Alex Domagala	Complete
7	Formalize prototype command generation script and share results with team	Change command generation script from notebook file (.ipynb) to python file (.py) for use on Jetson	Alex Domagala	Complete
9	Test out line following script in the lab	Test to see how the system works in the lab, and ensure reliability of system. Make changes if necessary	Jonathan Perthei	Partially Complete
10	Create basic movement script using line following	Create a script that reads the output from opencv and converts to left/right/straight and magnitude, and edge	Jonathan Perthei	Complete

Team Name	pathVision
Week	9
Manager	Pranay Singh

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Rewrite server script in C++, probably the easiest way to allow the c++ movement code to receive the data being sent over ethernet, script should allow for easy integration into the movement C++ files	Rewrite server script in C++	Emily Hernandez	Complete
2	Test the above script to ensure it is working properly	Test C++ script	Emily Hernandez	Complete
3	Talk with Jon G about mounting the jetson and camera onto the robot, discuss options	Put together a plan to mount the camera and jetson	Emily Hernandez	Partially Complete
4	Camera order arrived, stream depth map from intel realsense	Script that collects depth map data	Alex Domagala	Complete
5	Run depth map data through command generation script	Modified command generation script that takes a camera stream	Alex Domagala	Partially Complete
6	Start writing final movement code	Given the map and the constant string translate that into movement commands	Pranay Singh	Partially Complete
7	Work on remapping Controller for stop	Figure out how to remap the controller using high level code	Pranay Singh	Partially Complete
8	Plan how to send commands using communication link	Contact Emily to discuss how to transmit angle, and work with Pranay to control robot	Jonathan Perthel	Partially Complete
9	Develop classes for object tracking	Restructure object following script to be in class format to be used with other files	Jonathan Perthel	Partially Complete
10				

Team Name	pathVision
Week	10
Manager	Alex Domagala

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Currently the script to run the socket to collect data over ethernet and the script to move the robot are separate. We need to integrate them so they live in one script and can be ran together	Integrate socket script with movement code on pi	Emily Hernandez	Complete
2	Using the line following script, integrate the socket scripts so we can continuously send data from the algorithm over ethernet to the pi	Integrate socket script with line following code	Emily Hernandez	Complete
3	Start writing final movement code	Continue working on final script	Pranay Singh	Partially Complete
4	Work on remapping Controller for stop	Continue remapping the controller using high level code	Pranay Singh	Partially Complete
5	Test Intel Realsense with Jetson Nano	Depth Map generation script functioning on Jetson Nano	Alex Domagala	Complete
6	Improve command generation script, begin integration	Modified command generation script that can pass commands via ethernet	Alex Domagala	Partially Complete
7	Plan how the gain can be modified for movement commands	Identify how the gain can be changed to impact the magnitude of the movement commands	Jonathan Perthel	Complete
8	Prepare code for systems integration testing	Prepare object tracking code to work with entire system to test over spring break	Jonathan Perthel	Complete
9				

Team Name	pathVision
Week	11
Manager	Jonathan Perthel

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	We need to mount the jetson nano onto the robot	Create 3d model for mount	Emily Hernandez	Complete
2	Finish integration	Communication system properly sending all generated commands	Emily Hernandez	Complete
3	Test movement code	Once intergration is done wait start working on bugs	Pranay Singh	Complete
4	Test remapping Controller code	Test out the script for the key mapping	Pranay Singh	Complete
5	Finish integration	Basic command generation yields actual movement from robot	Alex Domagala	Complete
6	Testing & Improvements to Command generation	Update command generation script as testing reveals corner cases	Alex Domagala	Complete
7	Finish integration	Integrate object following and line following with communication system	Jonathan Perthel	Complete
8	Full system Testing	Fine tune system to ensure all system requirements are met	Jonathan Perthel	Complete
9				
10				

Team Name	pathVision
Week	12
Manager	Emily Hernandez

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	Hardware needs to be attached to the robot	mount hardware	Emily Hernandez	Complete
2	work on expo video	make final edits to expo video	Emily Hernandez	Complete
3	Integration	Constant testing of code to make sure movement is flawless	Pranay Singh	Complete
4	Expo Video	make final edits to expo video	Pranay Singh	Complete
5	Review Testing Results	Verify which requirements are met following testing	Alex Domagala	Complete
6	Work on expo video	Make final edits to expo video	Alex Domagala	Complete
7	Review Testing Results	Verify which requirements are met following testing	Jonathan Perthe	Complete
8	Finalize Expo Video	Make final edits to expo video	Jonathan Perthe	Complete
9				
10				

Team Name	pathVision
Week	13
Manager	Pranay Singh

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	work on final paper	finish final paper draft	Emily Hernandez	Complete
2	meet with expo judge	answer any questions for expo judge	Emily Hernandez	Complete
3	Meeting with expo judge	answer any questions for expo judge	Pranay Singh	Complete
4	Revising expo draft	Editing and revising expo draft	Pranay Singh	Complete
5	Final report draft	complete and edit final report draft	Alex Domagala	Complete
6	meet with expo judge	answer any questions for expo judge	Alex Domagala	Complete
7	Schedule Appointment with Judge	Schedule q&a with expo judge	Jonathan Perthe	Complete
8	Begin revising expo draft	Start adding comments to sections that need	Jonathan Perthe	Complete

Team Name	pathVision
Week	14
Manager	Alex Domagala

Topic:

Task #	Description	Deliverable	Assigned To	Status
1	complete final report	Final report	Emily Hernandez	Complete
2	Review Final Report	Make sure report is ready and finalized	Pranay Singh	Complete
3	Address Final report comments	Use the comments to change the final report	Pranay Singh	Complete
4	Address Final report comments	Make any necessary content changes to final report	Alex Domagala	Complete
5	Review Final Report	Make any necessary edits to final report	Alex Domagala	Complete
6	Address Final report comments	Address all comments on final report	Jonathan Perthe	Complete
7	Review Final Report	Review that all content is up to date and check grammar	Jonathan Perthe	Complete