SigSent

Autonomous Sentinel and Patrol Robot

Department of Electrical and Computer Engineering

University of Central Florida

Group 11

Joshua Lee Franco, CpE, EE, ME

John Millner, CpE, & EE

Jeff Strange Jr., EE

Richard Wales, CpE

Sponsor: Vision Land Service

Significant Contributors: Robotics Club, TI Innovation Lab

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

1 EXECUTIVE SUMMARY

The project explores a field of robotics, patrol and sentry duty, which has only recently become tractable through modern technological and scientific advancements. With its novel hardware platform and intelligent software, SigSent can assume a unique role in the field.

Producing a useful end product required developing new or implementing existing solutions for multi-terrain travel, efficient power and time management, and simple Human to Robot Interaction (HRI) for both the robot's supervisor and other people encountered during its operation. Improvements in battery storage density and computational power have lowered the cost of major components driving the design of such a robot.

SigSent demonstrated a capable platform which could substitute for a human in conducting routine patrol and sentry duties. These duties included following predefined paths in either smooth or rough terrains, reliably alerting the robot's supervisor to a potential intruder, and instructing a potential intruder on how to proceed.

The security services industry is a prime candidate for growth through human-robot cooperation. The Three Ds of Robotics: Dull, Dirty, and Dangerous, are applicable to security services due to the repetition of tasks, need for assured surveillance, and potential for hostile situations.

This document contains an analysis of the goals for this system, the requirements defining those goals, constraints imposed on accomplishment of those goals, research investigating avenues for the implementation of this project, design decisions which shaped our proposed solution, and the final prototype of SigSent. Additionally, possible future work is outlined.

2 Project Narrative

2.1.1 Goals and Objectives

The goal of SigSent is to prototype a robot capable of intelligently patrolling a predefined area and reducing the risk of harm to human sentries. SigSent can also enable security professionals to enact a more proactive security policy by freeing security guards from repetitive tasks.

By learning to work in a mixed terrain environment, the robot can effectively perform its job as a sentinel irrespective of the landscape in which it is placed.

With teleoperation functionality, operators can manually control the robot or direct it to enter an automatic sentry mode. The SigSent bot can stream a video feed of its perspective, enabling remote surveillance. With multiple SigSent units, a single operator could surveil a much larger area alone. When in sentry or patrol mode, SigSent can also alert the operator upon detecting activity of potential interest. This reduces the workload on security guards by freeing them from simultaneous supervision of multiple locations throughout the entirety of their shift. The SigSent robot should match the speed of an average person jogging so that it may pursue an intruder if deemed necessary. SigSent should be able to deter trespassers with vocal commands and also be able to record video of trespassers or events for later action by law enforcement.

In conclusion, SigSent should replace the main duties of a security guard and allow guards to perform higher-level tasks with less occupational hazard.

2.1.2 Requirements Specifications

Table 1: Requirement Specifications

Specification	Value	Units	Rationale	Ref.
Sentry Robot Specifications				
Weight	25	kg	OSHA limit of safe weight to lift	[1]
Durability	0.5	m	Survive a 0.5m fall. Internal benchmark to reach.	
Reliability	1	yrs	Based on Life cycle of parts (Servo motors, motors, etc.)	
Availability	75	%	Robot will need 25% availability for maintenance and repair.	
Speed Characteristics				
Wheel Top speed	12	mph	Average speed of a male human running ranges from 10 to 15 mph	[2]
Rough Terrain Top speed	1	mph	⅓ the normal walking speed of a male human. Internal benchmark to reach.	
Battery Life				
Static Monitoring Span	3	hrs	To be competitively better than our competitors, iPatrol, with a battery life of 1.5 hours	[3]

Walking Lifespan (3mph smooth)	30	mins	For basic, reasonable operation of the sentry bot. Internal benchmark to reach	
Jogging Lifespan (6mph smooth)	10	mins	Internal benchmark to reach	
Running Lifespan (12mph smooth)	3	mins	Internal benchmark to reach	
Rough Terrain Lifespan (1 mph)	15	mins	Internal benchmark to reach	
Accuracy Specifications				
GPS waypoint finding	5	m	Based on standard accuracy of smartphone GPS modules under open sky conditions	[4]
Communication distance	32	m	Based on the signal power limit allowed by FCC regulation for WiFi.	[5] [6]
Bandwidth	5	Mb/s	Based on industry accepted requirements for high definition video streaming.	[7]

2.1.2.1 Movement Specifications

The robot must have certain movement capabilities to be considered a multi-terrain and accessible device. This means the robot must be able to fit in common areas to do its functions. To satisfy this requirement the robot must be able to pass through a standard door size opening of 36 inches. This also includes the ability to travel across different smooth and rough surfaces/terrain.

Our definition of a smooth surface is: “Any continuous surface with no more than a 10-degree incline/decline”. This definition was created with reference from the National Highway Traffic Safety Administration and their road regulations for paved highways. The list of example smooth surfaces are tile, asphalt, and carpet.

Our definition of a rough surface is: “Any non-continuous surface with instantaneous raises/lower no greater than 6 inches and a max incline/decline of 15 degrees”. This definition was created with reference from the creation of the smooth surface definition. The list of example rough surfaces are forest, rocks, stairs, and sand.

2.1.2.2 Security Functionality

Being a sentry bot, this robot will require multiple security capabilities.

- Robot should be able to transmit full quality video to the base station upon request.
- Robot should be able to detect human movement from 10 meters away.
- Robot should be able to sound a siren heard at 60 dB from 10 meters away.
- Robot should be able to reliably operate during night. (at full moon, 0.01 ftcd, lighting)
- Robot should be able to be teleoperated from the base station.
- Robot should be able to have a path programmed into it.

2.1.3 House of Quality

Below in Figure 1: House of Quality is our house of quality with demo-able technical characteristics and their correlation to the characteristics seen by the user as important.

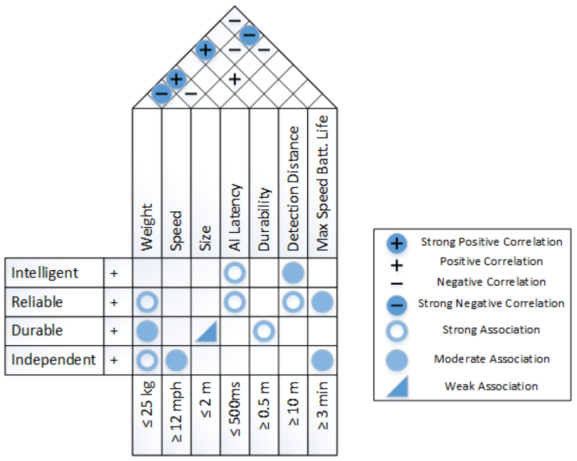


Figure 1: House of Quality

3 Design Constraints and Standards

3.1 Design Constraints

The constraints outlined below guided decisions on the overall design and marketing direction of the SigSent project.

3.1.1 Economic

In its final marketable configuration, SigSent will be strongly constrained by its total annual cost per unit. SigSent's value proposition is that it can supplement the utility of existing security personnel, and multiply their presence through a networked set of units. SigSent will only be a successful commercial product if it offers comparable surveillance capability as a conventional security guard at smaller recurring cost.

The median pay for a security guard in 2016 was $25,840 [8]. This is conceivably the uppermost constraint on the annual price an organization would pay per SigSent unit. If a SigSent unit were to cost greater than a guard for the same capability, the organization would likely hire the additionally employee instead. A SigSent unit's value should be comparable to that of a full-time security guard, since the unit offers additional capabilities that a conventional guard doesn't, such as the ready availability of a unit to record all of its visual and auditory observations. Additionally, SigSent units may be able to patrol a larger area than a conventional guard in the same time due to their greater speed in drive mode. SigSent must also match the scheduling availability of a guard as well, with full-time guards only working 40 hours out of 168 hours in a week, or ~ 24% availability.

With multiple SigSent units networked to one basestation, a single operator should be able to simultaneously monitor many areas of interest and respond to events as appropriate. The displaced cost of the potential additional guards enables businesses to afford SigSent units out of their current budgets.

By 2026, 70,000 more security guards are projected to be employed. These additional employees alone would cost $1.75 billion dollars annually [8].

3.1.2 Environmental

Because it is unable to open doors, SigSent is poorly-suited to operation in most indoor environments. Facilities with automatic doors could be suitable for SigSent's operation if the doors' trigger mechanisms were sensitive enough to detect the unit and open autonomously. Additionally, remotely-operated doors could be either intentionally triggered by a security professional. Networked door could communicate with a SigSent unit, enabling the unit to request the doors to open autonomously.

SigSent is not intended for use in environments lacking firm surfaces, such as swamps or soft snow. If a unit were to sink into a surface instead of walking atop it, it may not be able to remove itself. Additionally, SigSent is not intended for use on slippery surfaces, such as ice, because a unit may lose traction and then find itself stuck in isolation.

SigSent is not intended for use in weather producing poor visibility. Heavy rain, snow, or fog would limit an operator’s ability to see the environment surrounding a unit and may prevent the unit from successfully completing its mission.

For long-term survivability outdoors, SigSent’s final marketable design will need to be waterproof, protecting the electronics housed in its abdomen from ingress of water and its sensors from contact with moisture.

3.1.3 Social

For SigSent to be generally accepted for use in public spaces, its final marketable design will need to appear nonthreatening. This is especially critical in retail and hospitality environments where customers or clients should feel comfortable and willing to return.

SigSent’s final marketable design should look intentionally dissimilar from insects in order to avoid the uncanny valley which would diminish likability of the product to both customers and the general public [9]. Additionally, those suffering from entomophobia would be especially distressed by a unit highly similar to an insect [10].

Units should be reasonably quiet in order to prevent disturbing people nearby.

3.1.4 Political / Ethical

SigSent is not intended to be used as an offensive surveillance tool. Customers should be informed accordingly and reminded not to use the device for illegal recording activities.

With SigSent being a product tailored to the defense and security industry, it could potentially be subject to export control restrictions. Care would need to be taken to ensure relevant governmental agencies approve of foreign sale.

At least initially, SigSent’s customer base should be carefully examined to ensure any potential purchasers would not use the system for illegal or objectionable activities. Greater market acceptance may be hindered if any early adopter is shown using the system for a negative purpose.

3.1.5 Health & Safety

SigSent incorporates a high-capacity lithium-ion battery which could pose significant risk of harm to those nearby if handled incorrectly. Only appropriate chargers should be used with the system to ensure the battery is not overcharged.

SigSent’s limb joints could pose pinching hazards, and the system should not be handled by children.

At full speed, SigSent will possess considerable momentum and should not be directed into an obstructed path. A collision could incur significant damage to the SigSent unit, in addition to causing property damage or bodily harm.

To preclude any chance of optical damage, SigSent's LIDAR unit should be no greater than a class 1 laser device [11].

3.1.6 Manufacturability

SigSent is designed to leverage digital fabrication practices, enabling flexible lead times and manufacturing run scheduling. Fabrication techniques include 3D printing and laser cutting. These common techniques minimize overhead costs due to underutilized or specialized manufacturing equipment. Each unit is also fairly symmetrical, reducing the number of unique design elements, and enabling larger quantity production of the appendages.

Tolerances are not incredibly exacting, enabling high yield production rates for fabricated components.

Scaling SigSent larger would be relatively straightforward, with none of its structural components approaching typical limits of a readily-available workshop scale cutter or printer. SigSent could likely be scaled twice as large, but at greater material cost.

SigSent may not be able to scale down significantly without major design changes due to the demands for abdomen space and due to the relatively poor tolerance of the digital fabrication tools on market.

3.1.7 Sustainability

SigSent operates on electricity, which can be provided by renewable energy sources. SigSent can be manufactured from sustainable plastics and polymers. Its battery should be carefully disposed to prevent environmental degradation. If possible, the batteries should be recycled.

3.2 Hardware Standards

In the creation of SigSent, hardware standards used in the industry were followed to minimize possible error. These strict standards, set by large, successful companies, will force our team to work at the highest quality.

3.2.1 Soldering Standards

Soldering is a critical skill required to ensure that all electrical connections are electrically connected and have minimal impedance. Equally important for the system is that soldering creates a secure mechanical attachment of a component onto a PCB.

To ensure that our soldering techniques and processes are trustworthy, we will be loosely following Standard J-STD-001F [12] created by the National Aeronautics and Space Administration (NASA). This document goes over various soldering materials, supplies, definitions, preheating procedures, reflow procedures, what defines a good solder connection, and how to verify a proper soldering connection.

Notable sections that will be strictly followed are those in section 4.18: Solder Connection which defines and discusses the characteristics of a proper solder connection defined by wetting, angle, slope, and surface finish. Also defined in the report are the characteristics of an improper solder connection in section: 4.18.2: Solder Connection Defects which details how to identify bad solder connections. In Section 5.1: Wire and Cable Preparation the standard defines how to identify bad insulation and when a wire should be deemed unfit for use. Section 7.5.7 Flat Gull Wing Leads defines acceptable soldering dimension criteria on placement and solder fillet radii.

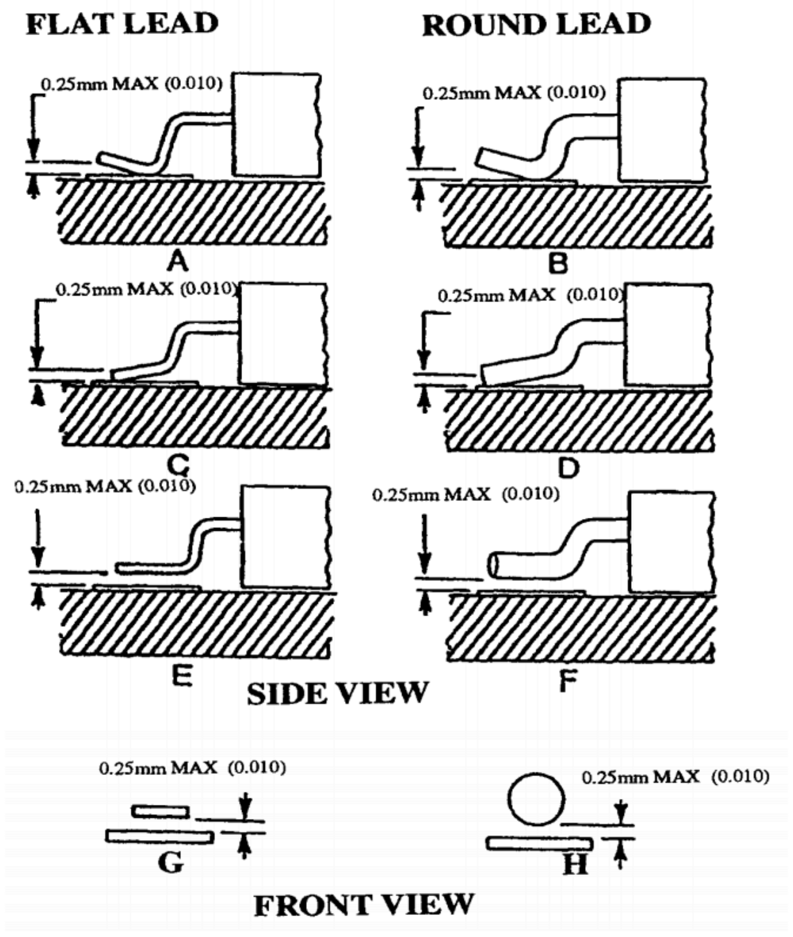


Figure 2 NASA Soldering Standard: Lead Height (Public Domain NASA)

3.2.2 PCB Design Standards

Proper printed circuit board (PCB) design is critical for ensuring that the PCB introduces only minimal noise and impedance to the circuits that the PCB holds and that the circuits work as expected. Outlined in the standard IPC-221A by the Association Connecting Electronics Industries (IPC) [13] these standards cover design and fabrication practices related to PCB design. This standard also highlights common problems and solutions that will help the SigSent team create a functional board. Key sections of the report that will be followed strictly are found in section 3.6.1 Board Layout Design, the entire chapter 6: Electrical Properties, and Section 7.2: Heat Dissipation Considerations. Following the standards set forth by IPC will better ensure that our circuit boards will work as expected and will be considered good craftsmanship.

3.2.3 IEEE 802.11g

802.11 Wi-Fi comprises multiple iterations of a standard that has evolved over time. The standard used in the implementation of SigSent is 802.11g. The 802.11g standard for Wi-Fi is an older standard from 2003. It utilizes the 2.4GHz band. Its average throughput is 22 Mbit/s with a maximum of 54 Mbit/s for forward error correction codes. 802.11g is backward compatible with 802.11b. 802.11g was quickly adopted by the market due to its increased speeds at the time of its release. We are using this standard as it is integrated with the router we are using on-hand. Without budget constraints, a higher fidelity router with a better resolution or signal strength could be used [14].

3.2.4 Inter-Integrated Circuit (I^2C) Version 6

I2C (I-two-C), also known as I^2C (I-squared-C), is the communication standard we will have to abide by when connecting our various sensors to the microcomputer device. I2C stands for Inter-Integrated Circuit. It has multiple masters and slaves. It was invented in 1982 by Philips Semiconductor. I2C is used for connecting ICs to microcontrollers/computers with a close relative locality. There are no licensing fees to use I2C. The only fee that you need to pay is for access to slave addresses that NXP (the new name for Philips Semiconductor) assigns. I2C is a design where there is a clock (labeled as SCL) and data line (SDA) with 7 bits of addressing. The master nodes generate the clock and start the communication process. The slaves receive the clock and respond to the master's requests. There can be any number of masters present. Masters and slaves can change places at will after a message transmission session has ended with a stop signal. The master kicks off the process by sending a start signal with the 7-bit address of the slave that it wants to work with. The master then sends a bit that specifies what mode it wants to enter with the slave (read/write). The slave responds to the master if it is on the bus and received the message. The signal that the slave sends is known as the ACK signal (acknowledgement). The bits are sent with the most significant bit first. The start of the bit stream is notated by a "high-to-low transition of SDA with SCL high" [15]. When the master is reading from the slave, it sends an ACK signal after every bit except for the last one, signaling that it is done receiving. Multiple messages can also be sent in I2C. A new start bit can be sent to signal a new message.

There are three formats for I2C messages: single message (master to slave), single message (slave to master), and combined (master has two reads/writes for each one for the slave). These dynamic formats make I2C a welcome solution for peripheral communication. We will primarily be operating under the slave-to-master single message format, where the microcomputer will request data from the sensors.

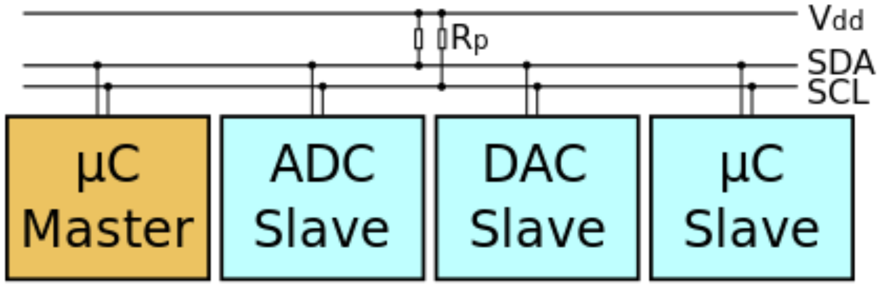


Figure 3 I2C example with one master, three slaves (CC license from https://upload.wikimedia.org/wikipedia/commons/3/3e/I2C.svg)

3.2.5 Universal Serial Bus (USB)

SigSent's design incorporates multiple USB peripherals communicating with its microcomputer. In order to troubleshoot potential communication errors, understanding the family of USB specifications is necessary.

The Universal Serial Bus (USB) set of specifications detail the protocol and physical hardware enabling interdevice communication [16]. Implementation of the specification is ubiquitous, and enables data and power transfer for many computer peripherals. The specifications define the physical form factor of connectors, parameters for reliable cabling, and the protocol for data transfer.

USB relies on a star topology with a host servicing multiple endpoint devices. USB permits branching hubs relaying host information further down line, allowing up to 127 devices to connect to a single host.

Over time, USB has received revisions increasing its bandwidth, with the most recent version, USB 3.2, boasting up to 20 Gb/s of data transfer. Additionally, the USB Implementers Forum, the organization governing the specification, has developed various different physical interfaces to connect USB devices. USB ports and their accompanying connectors come in multiple shapes and sizes as appropriate to the device's form factor.

3.3 Software Standards

In the software field, standards keep code readable and maintainable. To aid in the development of our software components, and expose our team to professional-grade development standards, we have outlined the following criteria.

3.3.1 Programming Languages

Python was used for the majority of SigSent's software modules. Python allows for quick prototyping and fast iterations due to its simple syntax and extensive standard library included in its distribution. ROS has full Python support and NEAT has a Python implementation that we have used in alternative projects before. With both the artificial intelligence and robotic system using the same back-end, they can work together with little alterations.

The Graphical User Interface (GUI) for the base station unit was built with Python and Pyside, a Python wrapper for Qt. Qt is a popular framework for GUI development, with an interface designer that features native components for the respective OS that it is run on, and a massive amount of documentation available. Qt is available for “student/academic purposes, ...[and] internal research projects without external distribution” under the GPL and LGPLv3 open source licenses [17]. When used commercially, Qt licenses can be fairly expensive.

Our project’s website was built using a framework more abstracted so that development time is not wasted building a comprehensive site that could instead be spent on bettering SigSent. Bootstrap templating was used to quickly produce a website where the project’s documents and progress can be publicly viewable. Python could be used to make a site using the Django framework, however this is not necessary, as the powerful templating engine from a dynamic backend language will not make much of a difference for a website as basic as ours.

3.3.2 Naming Conventions

The naming conventions we use follow the PEP8 Python Style Guide written by Python’s creator (and Benevolent Dictator) Guido van Rossum. Classes use PascalCasing. Functions and variables are underscore separated like_this. PEP8’s style guide is used commonly in other companies and projects as it has set itself as a standard for software engineers. We followed this standard to be consistent within our own project, and to better fit in with other software teams utilizing Python. We also used a linter, Pylint to enforce these design decisions. The code will emit warnings and fail to properly build when the code style guidelines are not followed.

ROS has a style guide where they suggest best practices and auto formatting techniques to easily set the aesthetic of the C++ portion of the ROS code. ROS is fairly popular among professionals and enthusiasts alike. Many programmers host their code as open source projects on repository websites for other robotics community members to improve upon and utilize in their own works. SigSent’s code fit this design criteria to have an equal contribution in the field while not disrupting the standard already set in the community.

3.3.3 Build Environments

SigSent’s software was built on the Raspberry Pi microcomputer under the Ubuntu Mate 16.04 distribution. The Jessie Raspbian distribution is available as well, however Ubuntu Mate has more support at this time. ROS officially supports this setup. Catkin is the build system made for ROS similar to CMake, with added support for distributed sets of packages that ROS projects have. The Python scripts unrelated to ROS will not need to be built under a specific regime as they are dynamically interpreted by a Python interpreter at runtime. This lowers performance, however the tradeoff for Python's portability and speedy development is worth it.

The code was developed and tested on Ubuntu 16.04 VM and desktop platforms. Programming on an environment other than an ARM Linux distribution brings some inconsistencies, however using Python and ROS, there will be no issues, as each platform we develop for is officially supported by each module we use in development. Python itself has built-in cross platform support. Unit tests and frequent testing on our build environment will ensure that our software is always functional on our microcomputer. Debugging will be done regularly manually as well as automatically with our own build scripts we will use in our development pipeline.

3.3.4 IEEE 802.11i-2004

In order to ensure the privacy and integrity of communication between a SigSent unit and its corresponding base station, WPA2 security was applied to the shared Wi-Fi network. IEEE 802.11i-2004 is the technical name for the standard implemented by the WPA2 protocol [18]. 802.11i relies on the Advanced Encryption Standard (AES).

In 2004, WPA2 superseded the earlier Wired Equivalent Privacy (WEP) mechanism which had been proven insecure by Fluhrer, Mantin, and Shamir [19].

WPA2 dictates a handshake procedure between an access point and its supplicant which relies on exchanging messages encrypted with a common key, but not the key itself. Mutually successful decryption of shared messages confirms that both participants know the password and should be trusted.

3.3.5 National Marine Electronics Association (NMEA) Message

SigSent relies on the Global Positioning System to help locate itself globally. After determining its coordinates, SigSent logs its position according to a standardized structure developed by the National Marine Electronics Association (NMEA) [20]. All GPS Fix data messages are stored in adherence with the following format:

1. $GPGGA, the sentence identifier for fix data
2. Time Stamp, in Coordinated Universal Time
3. Latitude
4. Longitude
5. Quality Indicator, a value between 1 and 5
6. Number of Satellites in view and used in localization
7. Horizontal Dilution of Precision
8. Altitude of the antenna
9. Altitude unit of measure
10. Geoidal Separation
11. Age of Correction
12. Correction Station

4 Research and Background Information

4.1 Similar Projects

Below are two projects that follow our use-case and robot design. Knightscope has a product line, all advertised for autonomous sentry work. Although their robots do not have a similar build as ours, they seek to solve the same patrolling problem. NASA's ATHLETE robot is detailed due to its similar hexapod platform.

4.1.1 Knightscope

Knightscope is an up and coming company manufacturing patrol robots that are large, functional, and aesthetically pleasing. Their products are being distributed in subscription based packages to clients that request demos at their location. Knightscope has two models that are releasing and being tested in 2018, being the K1 and K7 respectively [21].

The K1 model is a stationary product with a large array of sensors at its disposal. It is even able to detect weapons and radiation levels. The K1 has a weight of 150 lbs and dimensions of 62.4in x 28.8in x 11.2in. Until the product is released, this is the limited amount of information provided to the public as of now.

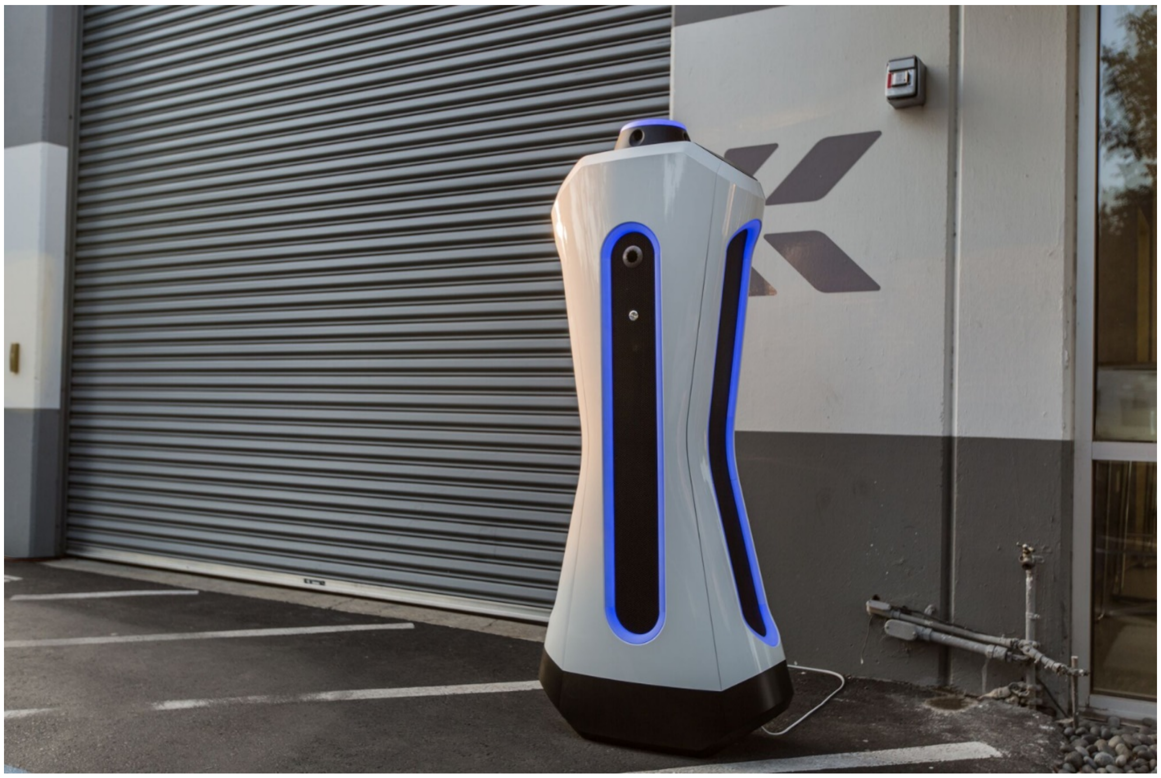


Figure 4: Stationary Knightscope K1 set outside of an office building [21]

The K3 is a mobile model that can move at up to 3mph. It is meant to be used indoors, "patrolling the interiors of businesses like sporting arenas, shopping malls, and warehouses" [21]. Its dimensions are 51in x 24in x 33in and it has a weight of 340lbs. The K3 sports a 360-degree high definition video feed that can be viewed by anyone with the proper permissions. Two-way audio allows communication from security personnel and people nearby the actual robot. Messages can be recorded beforehand as well to be played from any fleet of K3 robots. A thermal camera on the K3 allows thermal imaging on temperature critical devices at your location. The K3 can be used to alert the users if a predetermined temperature is reached to prevent damage or dangerous scenarios from playing out.

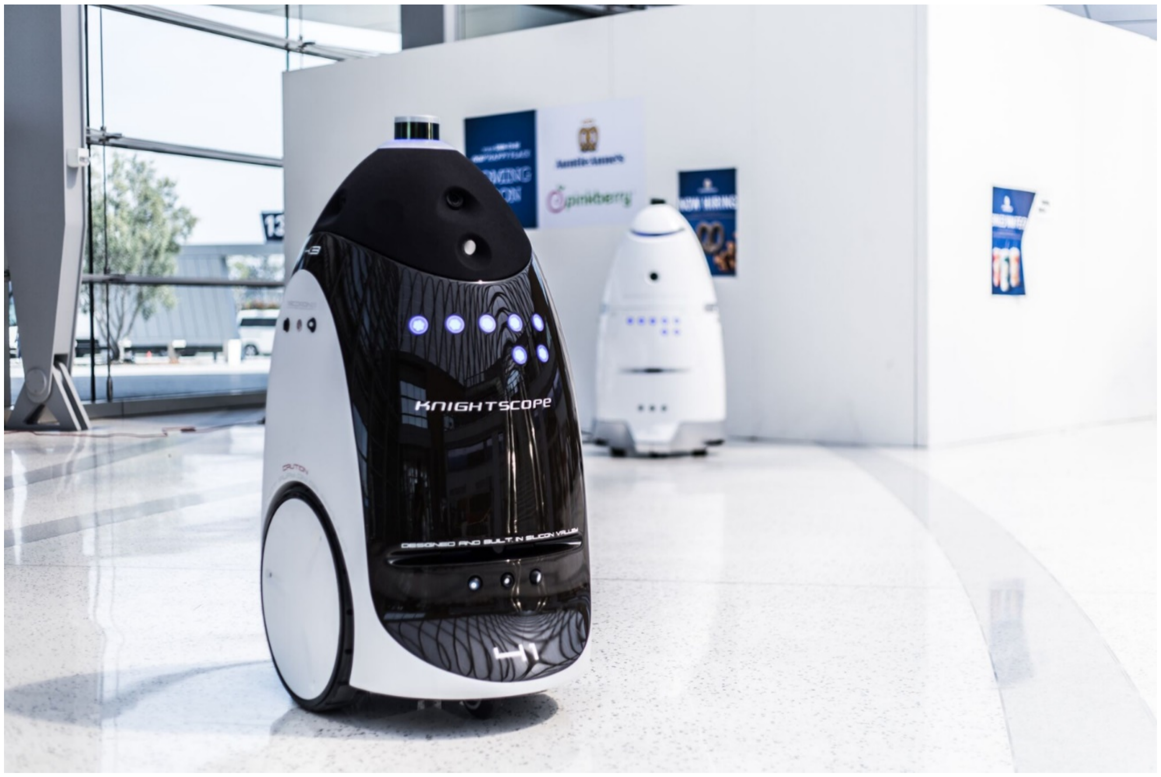


Figure 5: Knightscope K3 in an indoor environment [21]

The K5 is used to patrol outdoor areas. Knightscope says that it should be paired with a human element "to keep areas such as parking lots, corporate campuses and hospitals safe autonomously". The K5 appears to be a more durable, robust iteration of the K3. Its dimensions are 62.5in x 33.5in x 36in with a weight of 398 lbs. One of the selling points is its intimidating appearance. Knightscope compares it to having a marked police car sitting outside your location to deter criminals. The K5 can read up to 300 license plates per minute, checking for trespassers, blacklisted plates, and to track the usage of parking lots that it is supervising. The Knightscope K5 can also detect signals coming from routers and mobile devices to be aware of possible security penetrators in the nearby area [21].

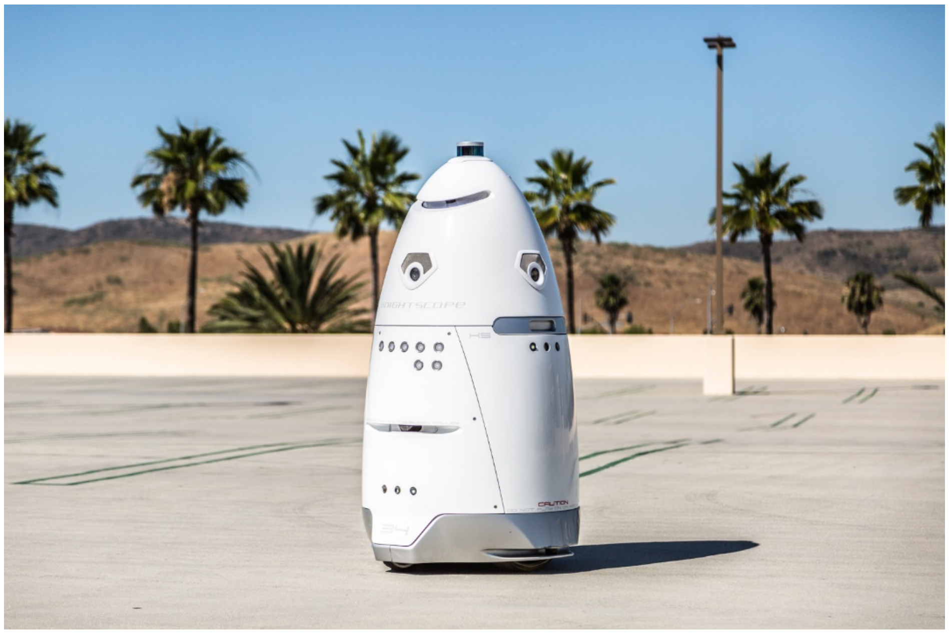


Figure 6: The Knightscope K5 patrolling a parking lot [21]

The final, and arguably most impressive Knightscope model, is the unreleased K7. This model has yet to enter a beta deployment, anticipated for 2018. The K7 is in the shape of a futuristic looking car. Its speed has not been announced, however it will most definitely exceed the current maximum speeds of the other Knightscope models. The K7 is a multi-terrain robot that has dimensions of 57.5in x 63.9in x 116in, weighing 770 lbs [21].

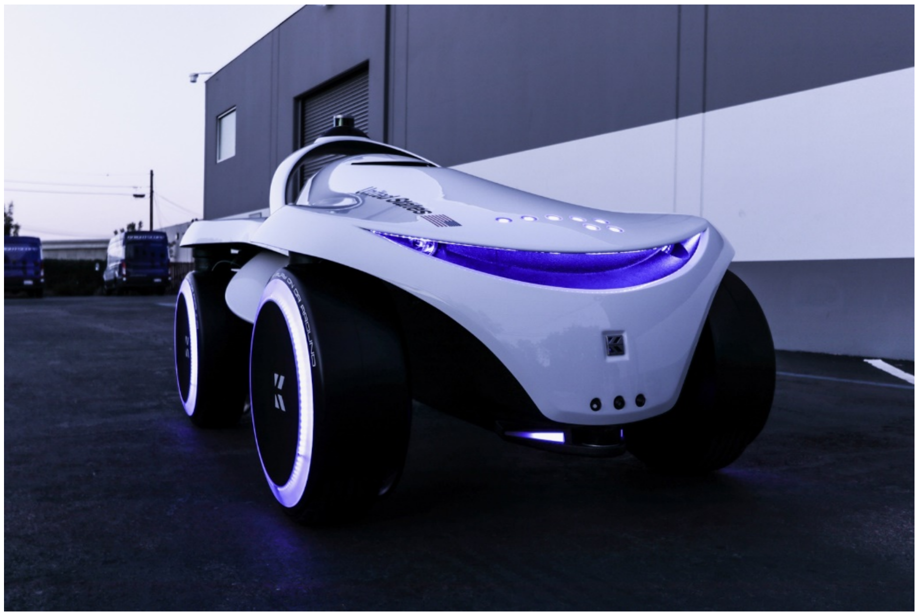


Figure 7: The K7 model in an outdoor environment [21]

4.1.2 ATHLETE (All-Terrain Hex-Legged Extra-Terrestrial Explorer)

NASA's ATHLETE rover is a wheeled hexapod platform proposed for use during extraterrestrial missions [22]. ATHLETE differs from SigSent in scale and form but shares the same principle of integrating motorized wheels with walking legs to efficiently navigate mixed-terrain. ATHLETE is 4 meters in diameter, stands at 4 meters, and can carry a load of 450 kg on Earth. In contrast with SigSent's combination of wheeled and wheel-less legs, all of ATHLETE's legs have wheels.

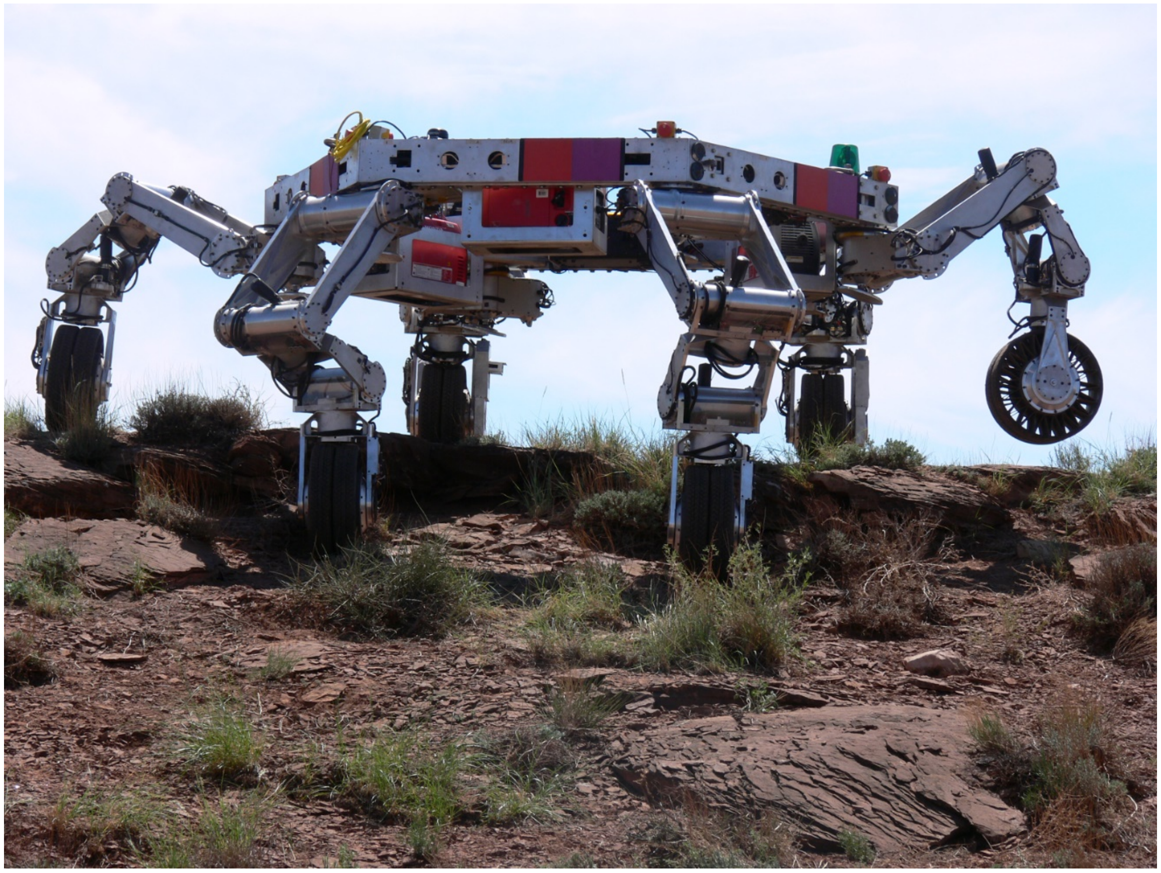


Figure 8: ATHLETE navigating rough terrain with wheels installed. Courtesy NASA/JPL-Caltech.

ATHLETE is designed to operate at up to 10 km / hr on smooth terrain, which is 100 times faster than its predecessor rovers on Mars. This would allow ATHLETE to survey a much larger area in the same length of time. It also makes ATHLETE useful as a cargo transporter in addition to basic observational roles.

ATHLETE is also able to perform various missions through the attachment of modules to its highly-maneuverable legs, offering a more generally useful platform than specialized rovers of the past.

4.2 Software Research

In deciding the entire software platform for SigSent, research was done for components at every level of development, starting with the microcomputer board's OS, and ending with the high-level artificial intelligence framework on which SigSent learns to act.

4.2.1 Operating Systems

The microcomputer and necessary software ran on SigSent added constraints to the operating system that was chosen. It had to support each framework and library used and be able to be run on a minimal, low-cost microcomputer.

4.2.1.1 Raspbian

Raspbian is a Linux distribution based on Debian, another popular flavor of Linux. Raspbian has been endorsed and provided by the makers of the Raspberry Pi (Raspberry Pi Foundation). This OS is directly made for use on the Raspberry Pi, meaning it has been stripped down to only contain what the Pi needs and can use. It uses a lightweight desktop environment called PIXEL (Pi Improved Xwindows Environment Lightweight) for optimized performance on the microcomputer. Raspbian is also supported for use with ROS, although with the newer tutorials on recent ROS distributions, they recommend that Ubuntu MATE is used, due to its more extensive package list for use on the Pi.

4.2.1.2 Ubuntu MATE

Ubuntu MATE is a FOSS version of Ubuntu that is able to run on popular architectures such as IA-32, x86-64, PowerPC, and ARMv7 (which the Raspberry Pi features). Ubuntu MATE was a possible candidate for our main OS that SigSent runs due to its support for the ARM architecture. Ubuntu MATE is a fully featured OS and has the support of Canonical's powerful Ubuntu system. Ubuntu MATE has better ROS support for newer distributions that Raspbian, and was so chosent to serve as our main OS for SigSent.

4.2.2 OpenCV

OpenCV is a widely used open source computer vision (hence the name) library. It has been ported over to many languages, including Python, the primary language used for our project. OpenCV was initially created by Intel to create a free framework that developers could read and use to build upon for advanced vision infrastructure. It was originally released publicly at the 2000 IEEE Conference on Computer Vision and Pattern Recognition. OpenCV has since been taken over by a non-profit organization at OpenCV.org. Now, OpenCV contains much more than a simple vision-based recognition system. They provide additional support for decision tree learning, Naïve Bayes classifiers, artificial neural networks, and deep neural networks (used extensively in frameworks such as TensorFlow, a deep learning framework made by Google).

OpenCV is commonly used in facial recognition, gesture recognition, robotics, object identification, motion tracking, and augmented reality applications.

OpenCV is written in C++. To spread the framework to multiple platforms, wrappers have been made in several languages so that developers in almost any project can utilize it in some way. Popular languages using OpenCV are Python, Java, MATLAB, and C#. OpenCV is also supported on most operating systems, including: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD, Android, iOS, Maemo, and even Blackberry 10.

To increase performance, OpenCV has added support for GPU acceleration in the image processing pipeline. CUDA support was added so that NVIDIA based cards can take advantage of GPU rendering speeds. OpenCL has been added as well, which is open source, but not as performant as CUDA in graphical applications. With embedded applications, using an NVIDIA board would enable higher performance under any vision-based project.

OpenCV was used for SigSent to recognize anomalies during its sentry routes. There are a myriad of built-in classifiers for object detection already. By simply enabling a classifier to detect people or movement, we can simply alert the user monitoring the SigSent unit, and highlight the activity on the video feed being streamed to the base station computer. The Histogram of Oriented Gradients (HOG) detector is used for pedestrian detection as it is already implemented in OpenCV. The Raspberry Pi that the SigSent is running off of does not perform the computer vision computation. The base station does the actual vision detection on the images transmitted from the Raspberry Pi's ROS node. If a connection to the base station is lost, the robot is still be able to capture images, but does not attempt realtime CV on them. The Raspberry Pi is powerful, but a microcomputer with better hardware, namely a GPU, would be necessary to do local computation. The NVIDIA Jetson is a pricier alternative.

4.2.3 SLAM

Simultaneous localization and mapping (SLAM) is a process which combines mapping an unknown area with localization. First created by R.C. Smith and P. Cheeseman in 1986, SLAM combines the creation of topological maps created from sensor data and Advanced Monte Carlo Localization (AMCL) to create an accurate relative map that is constantly expanded and refined as a robot moves around its environment. As the robot moves around sensor data is collected and creates a relative "frame" of a map, this frame is then matched with the robots last known location through AMCL to determine the robots new position within the map and the two maps stitched together to create a seamless map that the robot can later use for obstacle avoidance and object-based navigation goals.

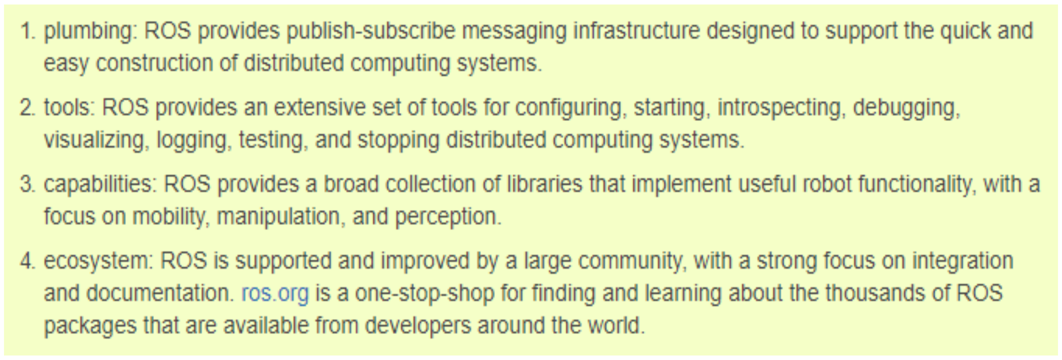
4.2.4 State Machine

A finite state machine (FSM) is defined as "a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a transition. An FSM is defined by a list of its states, its initial state, and the conditions for each transition." [23] FSM's are often used in robotics as a way to place the robot into a state of operation based on some number of inputs, A state might be something as simple as “sleep” or as complex as “search for ‘x’”.

4.2.5 ROS

The Robot Operating System (ROS) features new distributions on a constant release cycle of “Long Term Support” (LTS) on even numbered years and then short-term releases with a shorter lifespan on odd numbered years. LTS releases are recommended for mission critical applications. We will be using the ROS Kinetic Kame distribution released on May 23rd, 2016 with an End-of-Life (EOL) date of April, 2021.

“…[ROS] is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.” [24]. ROS is designed around four key concepts, Plumbing, Tools, Capabilities, and Ecosystem: [25]



1. plumbing: ROS provides publish-subscribe messaging infrastructure designed to support the quick and easy construction of distributed computing systems.
2. tools: ROS provides an extensive set of tools for configuring, starting, introspecting, debugging, visualizing, logging, testing, and stopping distributed computing systems.
3. capabilities: ROS provides a broad collection of libraries that implement useful robot functionality, with a focus on mobility, manipulation, and perception.
4. ecosystem: ROS is supported and improved by a large community, with a strong focus on integration and documentation. ros.org is a one-stop-shop for finding and learning about the thousands of ROS packages that are available from developers around the world.

Figure 9: ROS Key Concepts

The core usefulness of ROS is that it provides a standardized messaging and monitoring system where programs can easily and generically interact with other programs allowing for easy communication and modularization between high and low-level software’s with each other. This standardized messaging also allows for software’s to be genericized from individual hardware’s, where only the low level driver wraps a sensors output or a actors input in a ROS compatible message. This standardized messaging and monitoring allows for ROS to be very powerful by allowing programs to become very modular, and also allows for distributed computing natively by design.

ROS has a hierarchy where there is a ROS MASTER which is the main program that executes and manages all interactions and keeps track of everything happening within ROS. Individual programs are called nodes, nodes can be a publisher or/and a subscriber which sends or receives messages to/from a topic. A topic is a bus of messages and work as an abstracted message handler controlled by ROS MASTER. The general process is that you have a PUBLISHER which sends a MESSAGE to a TOPIC, a SUBSCRIBER is listening to the TOPIC and then receives the MESSAGE.

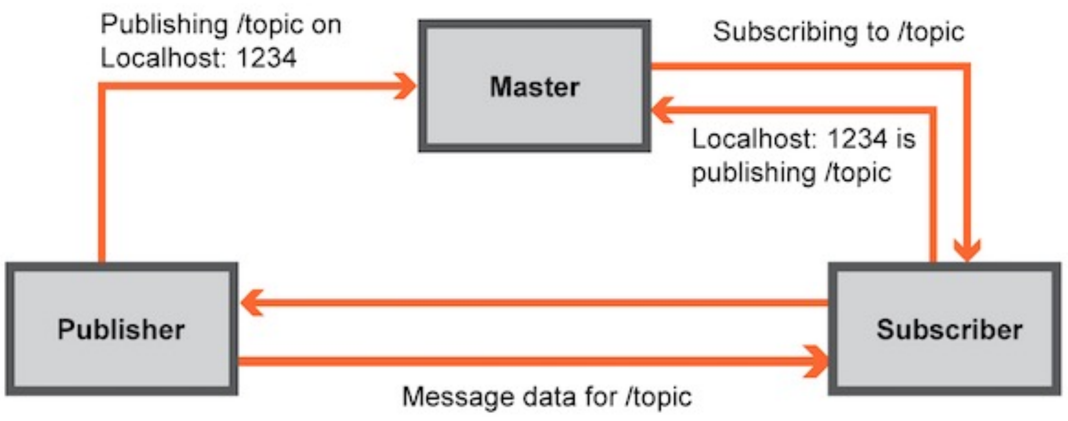

Publishing /topic on Localhost: 1234
Master
Subscribing to /topic
Localhost: 1234 is publishing /topic
Publisher
Subscriber
Message data for /topic

Figure 10: Example of a publisher and subscriber relationship in ROS [26]

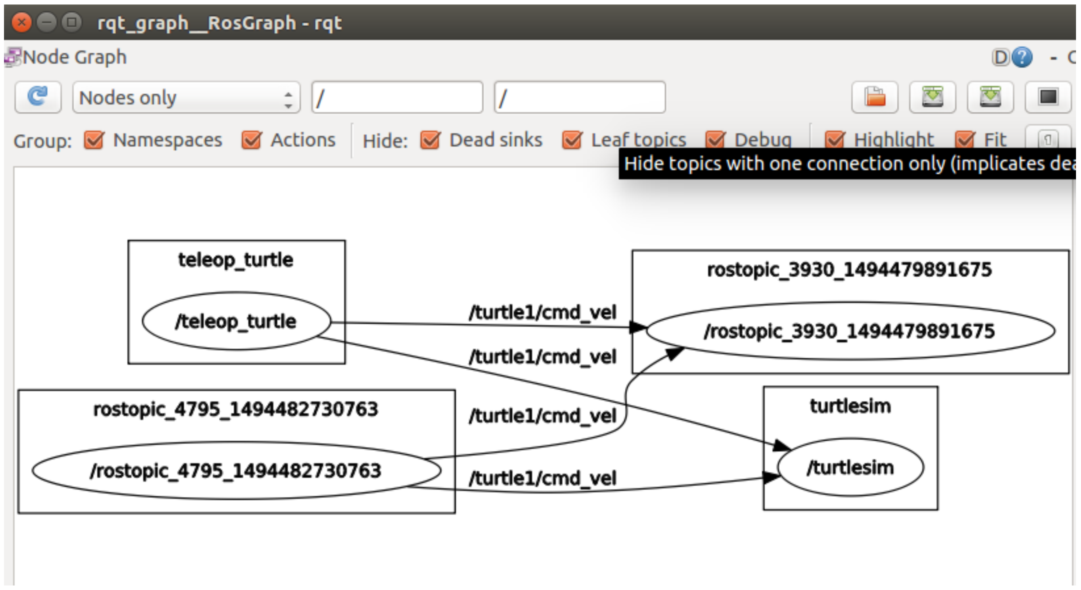


Figure 11: Visualization of Multiple Nodes and topics interacting through messages in ROS [27]

4.2.5.1 Gmapping

Gmapping is a Package that contains the node that runs SLAM (4.2.3). Gmapping uses lidar (4.3.3) data along with camera, IMU (4.3.7) and GPS (4.3.8) data to construct a map of its surroundings on the fly. Gmapping enables the robot to map its surrounding area and localize itself so that SigSent can quickly understand its local area and avoid obstacles and path itself along GPS waypoints.

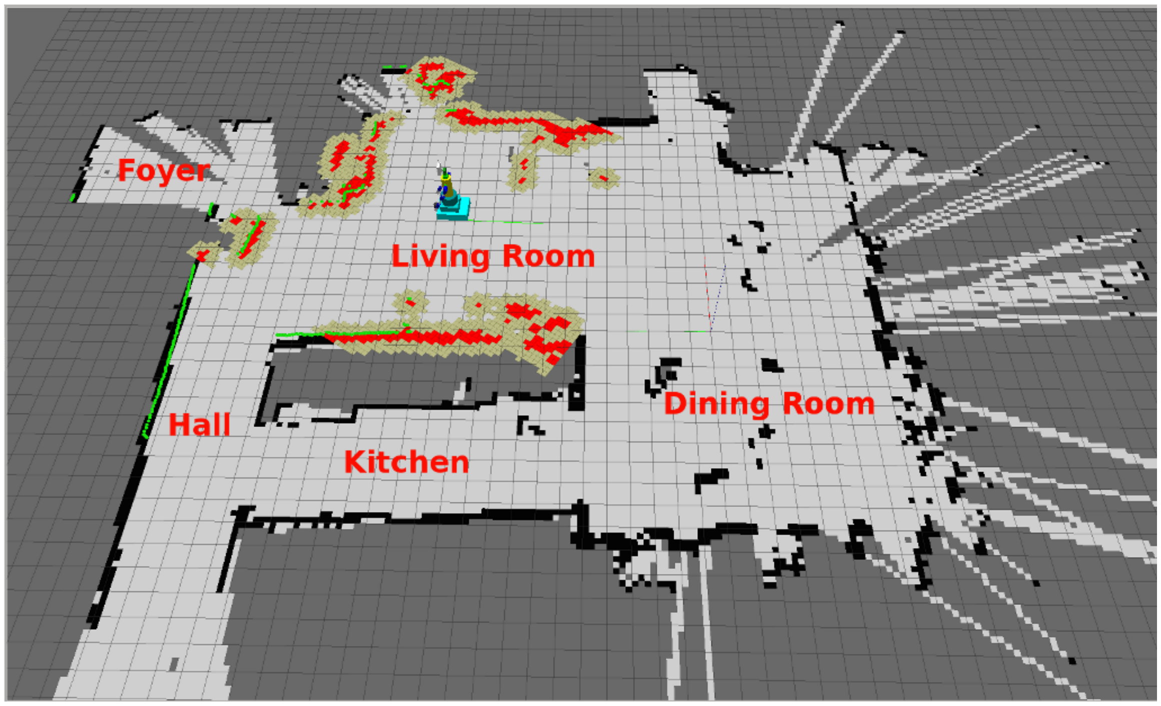
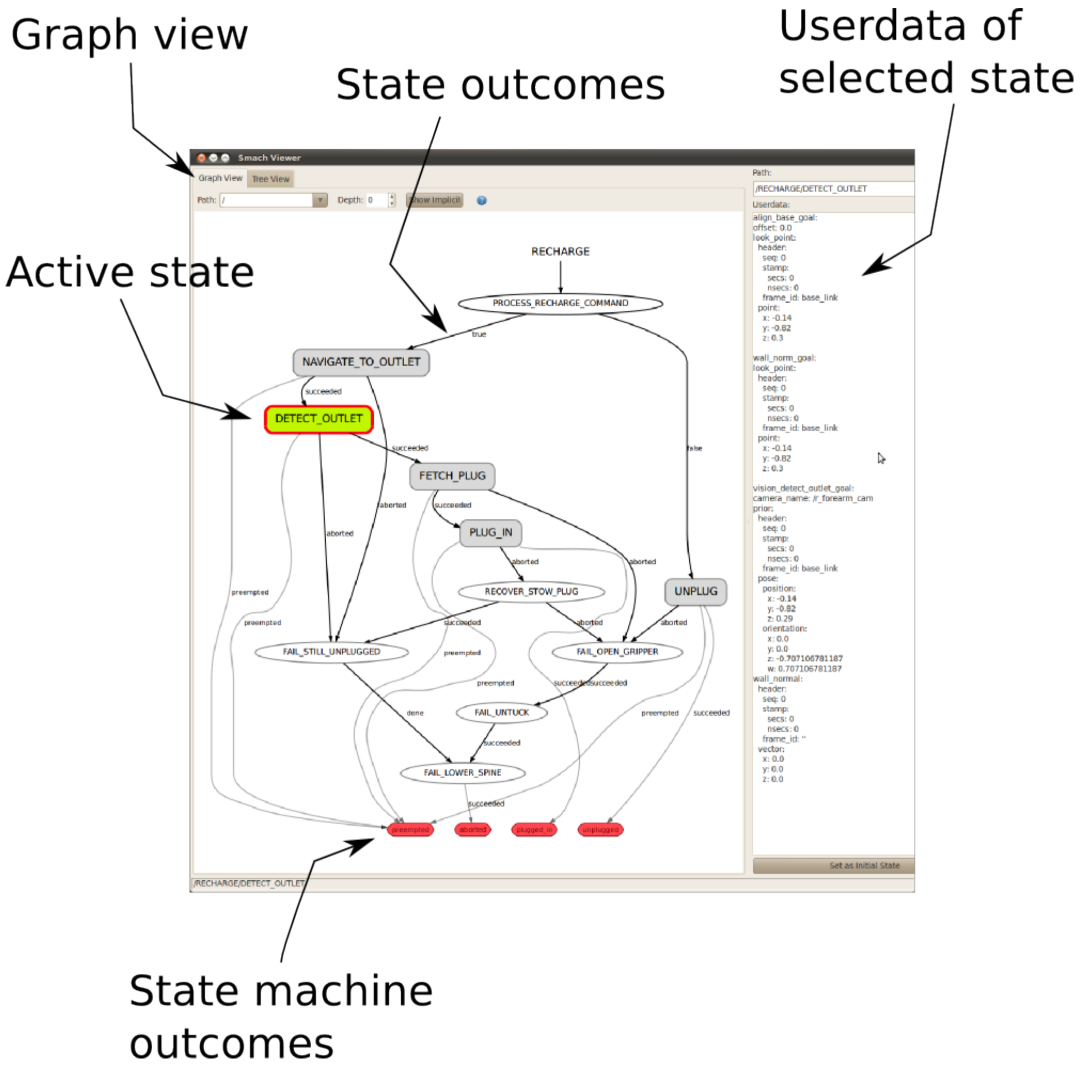


Figure 12: Representation of Gmapping and Lidar Data (highlighted in red) [28]

4.2.5.2 SMACH

SMACH or State MACHine, is "…a task-level architecture for rapidly creating complex robot behavior. At its core, SMACH is a ROS-independent Python library to build hierarchical state machines. SMACH is a new library that takes advantage of very old concepts in order to quickly create robust robot behavior with maintainable and modular code." [29]

SMACH allows for a streamlined and integrated way to create a state machine within ROS which efficiently directs SigSent to its short and long term goals as well as switch states quickly on the detection of an intruder or human controlled teleoperation.



Path:

/RECHARGE/DETECT_OUTLET

Userdata:

```
align_base_goal:
offset: 0.0
look_point:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: base_link
  point:
    x: -0.14
    y: -0.82
    z: 0.3

wall_norm_goal:
look_point:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: base_link
  point:
    x: -0.14
    y: -0.82
    z: 0.3

vision_detect_outlet_goal:
camera_name: /r_forearm_cam
prior:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: base_link
  pose:
    position:
      x: -0.14
      y: -0.82
      z: 0.29
    orientation:
      x: 0.0
      y: 0.0
      z: -0.707106781187
      w: 0.707106781187
wall_normal:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: ''
  vector:
    x: 0.0
    y: 0.0
    z: 0.0
```

Figure 13: A Visualization of a SMACH State Machine [30]

4.2.6 Intelligent Systems

Before implementing the artificial intelligence portion of SigSent, extensive research was done on the various learning methods available to us. After reviewing their advantages and disadvantages, we settled on using reinforcement learning. Reinforcement learning is used in a variety of algorithms. We researched some of the most popular algorithms and decided to use NEAT (NeuroEvolution of Augmenting Topologies) due to its impressive track record and presence at the University of Central Florida, as well as prior knowledge of the algorithm's inner workings and extensive work in genetic algorithms by the team's artificial intelligence programmer, Richie Wales.

4.2.6.1 Learning Methods

In AI, there are diverse ways a system can artificially "learn" to perform a task. Research was done on the three main methods so that the most optimal one would be further explored and then used for SigSent's intelligence platform.

4.2.6.1.1 Supervised Learning

When the desired output is known to the programmer, supervised learning is used to push the intelligent system to provide the necessary function that results in this output. The output necessary for a given input is sometimes called the *supervisory* signal. Supervised learning is useful when there is a clear behavior that should be propagate.

The learning takes place on a training set of data. This set is handpicked by the programmer. An additional data set is necessary to test the derived function after learning. This test set determines how effective the training period was. This test set must contain unique elements in it that were not included in the training period to provide sufficient evidence of a generally learned behavior. This tests how general the learned function has become. The test set needs to be broad enough to represent the data fairly as it occurs naturally.

The inputs into the function should be minimized to lower the complexity on the learning process. Having too many inputs will require optimizations of all of those attributes. A common phenomenon is the “curse of dimensionality” [31]. This issue refers to having too many dimensions of data to optimize for, where the search space grows much too large. With every added dimension, the number of enumerations possible for each parameter increases by a multiplicative of each additional input. For machine learning, this means you need to have an even greater number of training data points such that you fairly represent the desired output for a large region of the search space. If you do not have enough data to represent each parameter’s changes, the function will not learn how to process each variation effectively. The Hughes Phenomenon is a relative of the curse of dimensionality specifically targeted towards pattern recognition described by Gordon F. Hughes in his paper, “On the Mean Accuracy of Statistical Pattern Recognizers” [32]. In his conclusions, he states that there is a maximum acceptable complexity associated with a problem domain. In his pattern recognition experiment, he found that after some threshold, the increase in input dimensionality did not lead to a significant improvement in creating his classifier. Hughes presents ideas on how to accurately predict the necessary input size. He suggests using statistical techniques, like “Shannons’ information measure” or “Kullbacks’ divergence measure” to prune the number of possible input sizes. He finishes his paper stating that further work must be done on these ideas to develop a better idea on how an optimal search space can be decided.

The learning algorithm that is chosen for the task should be problem specific. If the data can be easily represented in a specific data structure or programmatic manner, the algorithm should be chosen to fit that domain. If the hardware that the learning algorithm is taking place on is optimized for a specific data representation, that should also be considered. Embedded machines with limited memory would have to utilize a method that takes this into account. Perhaps the execution time is more important than the space complexity; This would lead the programmer to seek an algorithm optimizing for speed by sacrificing memory usage. A humorous theorem in mathematics known as the “No free lunch theorem” [33] covers this problem. No algorithm will be able to solve every function.

There is always a tradeoff associated with it. Because machine learning and artificial intelligence has become a more matured field over time, there is no shortage in possibilities though.

Supervised learning, while effective, is all about curve fitting. Sometimes however, there is no “desired” behavior that we hope to elicit. In the case of SigSent, we have a type of behavior that we hope to see propagate, however there is no exact functionality that we want to impose on the robot’s mechanics. If we knew exactly how its movement should be performed for very specific terrain environments, the training set could encompass what moving mechanism and mobility methods are used for very specific conditions. In the case of our arachnid inspired device, the movement type and behavior is complex. The learning algorithm finds some optimal, or at the very least, well-performing functionality that solves the problem of mixed mobility that we present to it. Instead, we choose a learning method that strives to achieve the programmed goal by whatever means available to the software/hardware. In this case, an unsupervised learning method is used.

4.2.6.1.2 Unsupervised Learning

In unsupervised learning, the machine learning algorithm attempts to find a function that classifies the given data with no direct comparison between objective, desired outputs. The algorithm has no guidance during the training, however it must find a way to group the data presented to it. The data is known as being “unlabeled”. They are strictly discrete values that have no classification or association given to the algorithm. Two popular domains that this style of algorithm is used for hope to solve classification and association problems. Clustering involves grouping data through some sort of classifier where data points share similar attributes. Associations are found through relationships between input parameters.

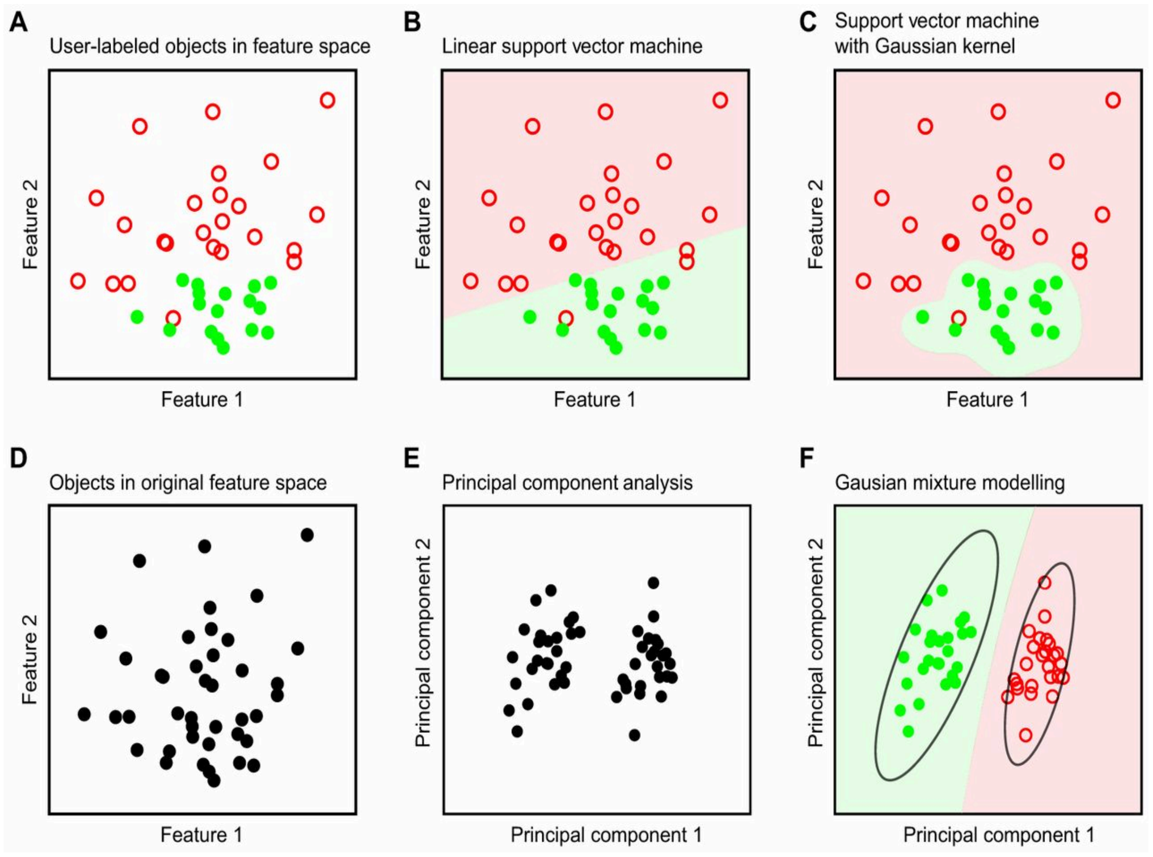


Figure 14: Supervised and unsupervised classification performed for Dr. Sommer's research on using machine learning for phenotype recognition

In the figure above, Dr. Sommer shows how supervised and unsupervised learning methods took place on labeled and unlabeled data respectively. In subfigures A-C, the colored data points show that they are labeled. In subfigure B, the data was classified to some decent sense of accuracy as most of the green points are clearly sectioned away from the red. This classifier was done with somewhat linearly separable data, as a single divider was able to separate most of the data. In subfigure C, using a Gaussian kernel, the data was able to be classified more accurately in a circular region. Additional extensions like this to the classic classifier allow for more accurate separations of classified regions to more accurately model the desired function. Subfigures D-F show data being classified under an unsupervised algorithm as they are unlabeled (shown as black data points). Subfigure E shows the data being grouped by analyzing the properties of the data. This is problem specific and can be as simple or complex as the use case it is performed on. In figure F, the grouped data is then classified by the unsupervised learning algorithm very easily, as the grouping that was performed before has easily separated the data into two distinct groups [34].

4.2.6.1.3 Reinforcement Learning

Like its name suggests, reinforcement learning uses the idea of a reward system to reinforce behaviors that are performing as desired. The reward is decided by the programmer, but should follow the problem statement closely. In the case of SigSent, this reward could be based on the distance the robot travels, the speed of its mobility, and/or the smoothness of the journey. This reward is tracked throughout the learning trial, taking sensor data as inputs to decide how well the device’s mobility mechanism performed. Reinforcement learning is used in a broad range of fields due to its easy extensibility. All the programmer has to modify is its reward mechanism and how its data is represented in the learning algorithm. Since the desired behavior is not exactly known, but the overall results that the programmer seeks is known, simply rewarding functionality that meets those requirements can cause any sort of behavior to propagate that meets them. This allows the computer to discover methods that were either not thought possible, or not initially envisioned. Giving the device that freedom can lead to interesting results. Markov Decision Processes (MDPs) are popular in machine learning practices.

Since reinforcement learning hopes to optimize some functionality without a direct input/output to compare to, there is a direct trade-off between the exploration of the search space and the exploitation of the current knowledge that the algorithm has discovered. This dichotomy has been researched heavily in learning algorithms to help optimize their performances. The multi-armed bandit problem is a prime example of this issue. In casinos, slot machines can be referred to as one-armed bandits, given it has a single arm and seeks to steal all of your money. The multi-armed bandit problem states, if you are given a slot machine with multiple arms that award different payouts and you have a limited amount of lever pulls available to you, how do you maximize your gains? You must explore the search space by trying out the levers presented to you, figuring out their probabilities as best as you can. The problem is, this exploratory period means that you are spending lever pulls on sub-optimal machines that will not net you the highest gain, but you must do this so you can discover which machine has the best reward. After some time, it is in your best interest to commit to the lever that you believe has the highest payoff. There are many modified versions of this problem and suggested solutions to it as well. The most optimal solution has been proposed in the paper, “Asymptotically efficient adaptive allocation rules” [35].

Depending on the problem and the environment that the agent is operating in, the learning agent will be given either complete or partial visibility of its surroundings. In the case of an agent implemented completely in software, its vision is boundless. There are no physical restrictions on what information is provided to it. The only reason to limit its vision would be to lower computation time and dimensionality of the domain. In the case of SigSent, and other physical implementations, the amount of information given to the device is limited by sensor specifications, data latency, and what is actually visible or present in the physical environment. The actions of the agent are measured in some sort of time tick decided by the programmer and the algorithm. Realistically, there would be some sort of loop executed in some discrete time step where the environment’s state is passed to the agent, the agent takes some action based on this information, and the algorithm/programmer interprets its action to provide the necessary reward to promote or discourage the behavior it saw. Depending on the implementation of the algorithm, the agent can attempt to maximize or minimize this reward. If the agent continuously performs some action that tends to be a boon to its reward, then it will continue to exhibit similar behaviors. It may do something radically different to explore more of the search space, but if that change was not helpful, the agent can easily fall back on the previous, performant action. This is where the exploration versus exploitation problem comes into effect. This search can be dynamic as well to offer better search optimizations. In the beginning of the learning process, exploration is very important. In a higher order function, there can be many hills and valleys in its search space to throw off the algorithm. In a hill climbing exercise for optimization, the algorithm should not settle for one simple curve with a positive gradient. The top of that hill could be suboptimal, local extrema. The exploitative part of the algorithm will continue to climb this hill, however in the later stages of the algorithm, exploration should still be possible to search for global maxima present in the space.

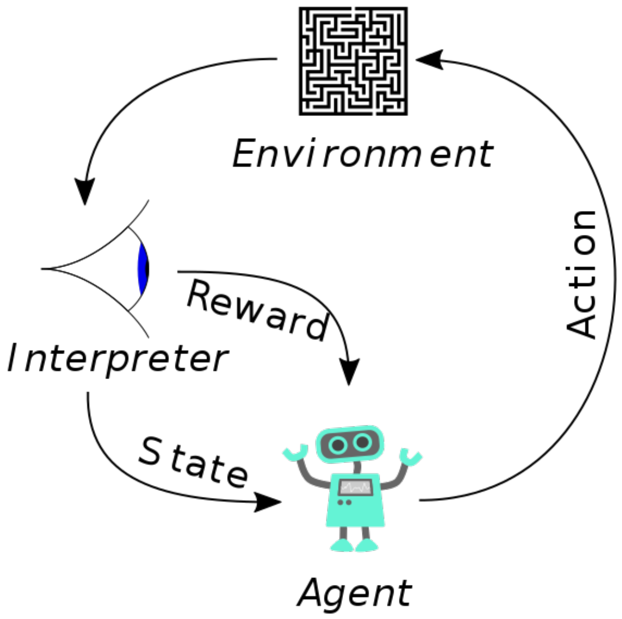


Figure 15: Representation of Reinforcement Learning (Under CC license at [36])

4.2.6.2 Reinforcement Learning Implementations

Of the three learning methodologies, reinforcement learning techniques were the natural choice for SigSent. True intelligence stems from the robot making the best decisions on its own. If we were to give it training data for very specific environments and expect certain mobility responses, we would be better off programming detailed sensor thresholds to trigger the mobility changes. The learning method would be cumbersome for little additional gains. Furthermore, the large input size from SigSent's sensor array would make the learning process very complex. We hope to enable SigSent to adequately learn optimal movement techniques by discovering it on its own, undefined by our team. Having a unique robot with several limbs, and limbs of different types, defining the proper movement method would be difficult anyway. When dealing with something radically different, letting the computer explore the different options available to it will provide a much clearer picture on what works and what does not, unbeknownst to us [36].

4.2.6.2.1 Q-Learning

Q-Learning is a policy-based learning algorithm that decides what action should be taken to result in the best utility value. An “action-value” function is learned by the learning method such that for any inputs, the highest scoring action is chosen to be enacted. A major disadvantage to using Q-Learning is that all of the states and actions must be known beforehand [37]. Since our robot will be used in many different environments, there is no way we can accurately model every single possible location as a designed state. Before the learning takes place, an exhaustive search of all of the various possible states would need to be found experimentally, which is infeasible for SigSent.

4.2.6.2.2 Genetic Algorithms

Genetic Algorithms (GA) take techniques from Darwin’s method of natural selection to effectively search through a space for optimal solutions. A population of individuals are randomly created initially, where each individual is essentially a “solution” to the problem. These individuals are tested in some environment, specific to the problem, and have a fitness score assigned to them. The fitness is the same reward mechanic in any reinforcement learning mechanism. The most fit individuals are then used to generate a new population through crossover and mutation operators. Crossover takes parts of two solutions and combines them into one. Mutation has a rarer occurrence, selecting pieces of a solution and randomly modifying them with no regard for the consequence of the change. This mutation is used to increase exploration of the search space. The specific operators, and how the individuals are selected for reproduction, are up to the implementation of the genetic algorithm. GAs are a way to speed up an exhaustive search by adding a sort of “implicit parallelism” by having individuals tackle the search space at many different angles (dependent on the size of the population), honing in on regions that have high fitness values.

Genetic algorithms can fall flat when the search space, or the relationship between the inputs and outputs, are not suited for the GA. If the search space contains many hills, the GA will have a difficult time trying to find the most optimal value as it will get stuck in local optima. The genetic operators (crossover/mutation) assume that solutions that are adjacent in the search space have similar fitness values such that small movements around those solutions will provide an increasing fitness. If there is no such relationship, the GA will be about just as effective as a random, exhaustive search throughout the space. Also, as is the case with our robot, there is a high order of dimensionality, which causes the search space to be absurdly large. Representing the solution in the GA can also be a major concern. Binary strings are the easiest items to manipulate in a GA, however with all of the sensors and motors utilized by SigSent, these could not be easily represented.

4.2.6.2.3 Neuroevolution

Neuroevolution uses a GA to evolve a neural network to solve a problem. Neural networks act as black boxes. You pass some inputs to them, they provide some hidden computation in the background, and then you retrieve the outputs and use them for your problem. Neural networks are able to find meaning from complex data that humans can not intuitively find. Neuroevolution provides this technique while also evolving the neural network with a GA such that the programmer's involvement in the learning process is very minimal. Normally, you must write some sort of methodology to change the weights on the neural network (where the topology is constant), or somehow change the structure of the network over time. Using a neuroevolution algorithm takes care of this work for you, altering the network based on its performance (as it is assigned a fitness).

4.2.6.3 NEAT

NEAT (NeuroEvolution of Augmenting Topologies) is a direct implementation of neuroevolution by Dr. Kenneth Stanley [38]. He found that starting with a minimally structured network and having a GA add complexity over time resulted in a simple network that had an optimal performance value. NEAT uses speciation to continue with exploration, while also not sacrificing exploitation by keeping networks grouped by similar topologies so that if one network structure was performing well, others were not discounted for further investigation. There are several new versions of NEAT that have been released as extensions for different use cases as well, so the community is very active. Dr. Stanley's lab, the Evolutionary Complexity (EPlex) lab, is housed at the University of Central Florida, giving us access to possible mentoring opportunities for the project.

The NEAT module for SigSent was not able to be implemented in its final form due to the limitations on robot walking experienced. The servos that were selected did not meet their torque requirements as advertised and the material that the robot was constructed out of was not strong enough to keep SigSent walking correctly without ripping itself apart. To prevent damage to the robot, SigSent had its gait and movement functionality demoed while suspended in their on a platform. Due to this limitation, SigSent could not gather IMU data to train the NEAT ANN. The research and code is still provided for a future implementation of SigSent if the reader chooses to do so.

SigSent would ideally use NEAT in a training phase in both simulation and physical environments. The simulation period will be used to refine the parameters for the algorithm as it relates to NEAT's GA values and the initial neural network's starting structure. The input and output representations need to also be modified into some optimal format, which is only known through empirical study. The simulations will be done in Gazebo, a test environment created to interface easily with ROS. If the tests in Gazebo run well, we will transfer that knowledge over into the training phase when working with the actual robot in real location settings. The studies done in the simulation may not have a direct application in the physical tests, however it gives us a good starting point and also allows us to code up the algorithm's implementation while parts are arriving, and the robot is still being built.

Our inputs into NEAT would be the angular velocity and linear acceleration from the IMU as well as the current mode of transportation (0/1). An artificial neural network would be trained to be able to classify the terrain type that is currently being traveled over depending on the current movement mode and the IMU data being supplied. Data would be collected that has the movement type used and IMU values labeled as being rough or smooth. After training the classifier to correctly identify the terrain, this ANN would be used in operation to tell the operator if they are using the correct mobility type for what terrain they are moving over. It would alert them to change if necessary.

4.2.7 Gazebo Simulation

To test the robot's mobility mechanisms in a realistic environment, and begin debugging the ROS code before the robot is built, a proper test simulation was needed. Gazebo is a popular robot simulation tool that is free to use and boasts a well-designed environment to "rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios" [39].

A model of a robot, designed in the SDF file format, can be imported into the simulation, describing every detail of the vehicle needed for simulation. ROS is easily integrated with Gazebo to allow for ROS message passing and processing within the simulation. The test environment that the robot simulation runs on can feature different types of terrain to validate the functionality of the mobility switching of the intelligent system. Through the various tests that the simulation is put under, the algorithms could be tuned and perfected to meet the criteria of the desired product.

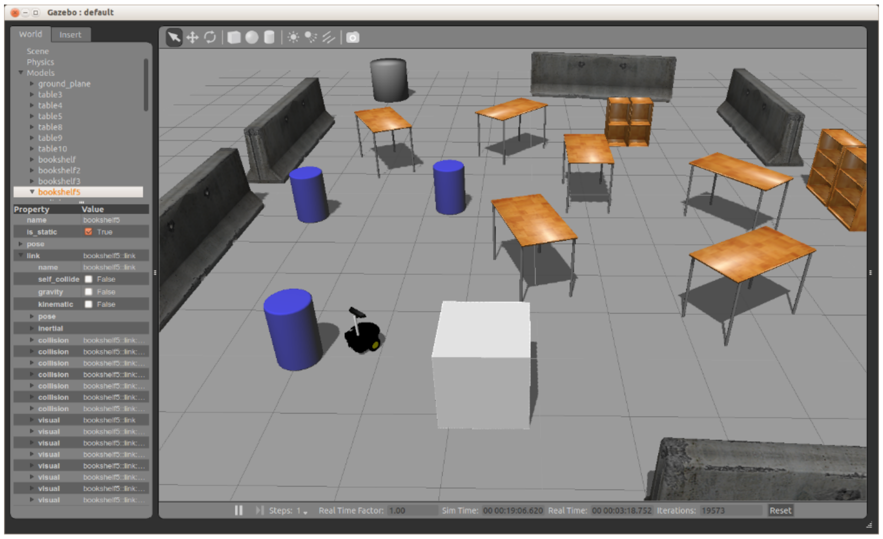


Figure 16: An Example of a Gazebo Simulation [40]

Work on the simulation was done up to creating a base environment in Gazebo from a drawn heightmap, however it was pruned from the project as too much work to have a fully functional hexapod robot simulated was necessary. The goal of the Gazebo simulation was to parallelize the work being done on the software and the hardware, but to work on the simulation, the hardware model needed to be completed to be imported into the simulation. Since the actual hardware model was a significant portion of the project, too much time was spent waiting on its completion and other key software tools were being put on hold. Due to this, the simulation was neglected and full attention was given to implementing code on the physical robot instead as it became available.

4.3 Hardware Research

In addition to the hardware constraints and standards outlined in section 0, every piece of hardware involved in the development of SigSent had particular part considerations necessitating research to find the best choice in each category. Some parts had scores created to evaluate their objective value to our project. Other parts have clearly defined specifications that were used to decide which part was the most optimal for SigSent.

4.3.1 Microcomputer

A microcomputer in the generic sense was chosen over a microcontroller due to the complexity of the robot. SigSent has to manage computer vision, LIDAR data, SLAM, state machine, control system, wireless communications, and diagnostic information constantly in order to properly complete its goal. Completing these goals on a microcontroller, while feasible, was determined to not be a time efficient solution since an operating system capable of handling multitasking would complete much of the juggling required to complete all those tasks simultaneously, in the same stroke there are many programs which could be used on an operating system (OS), most likely a linux derivative, that will also streamline the development of the project within our time scope. Libraries and programs such as OpenCV, ROS, i2c-tools, bash, ssh, and python allowed for the SigSent team to quickly develop the novel features of our robot while not reinventing the wheel, spending precious time developing and testing already heavily standardized features and libraries.

4.3.1.1 Microcomputer under consideration

The microcomputers defined below were serious considerations due to their popularity and computing power. Their strengths and weaknesses are displayed in the scoring table: Table 2 Microcomputer Comparison.

4.3.1.1.1 Raspberry Pi 3

The Raspberry Pi 3+ was under consideration for this project due to its low price, impressive computing power, number of USB ports, and its significant community support and documentation.

4.3.1.1.2 Raspberry Pi Zero W

The Raspberry Pi Zero W was under consideration for this project due to its extremely low price, low power consumption, and the significant community support and documentation

4.3.1.1.3 Beaglebone Black

The Beaglebone Black was under consideration for this project due to its good community support, and impressive amount of GPIO pins, and its efficient use of power.

4.3.1.1.4 Gumstix DuoVero™ Zephyr COM

The GimStix DueVero Zephyr COM was under consideration for its professional, more industrial design approach outside of the hobby/maker market like the above microcomputers. Where the gumstix lacks in other specifications and price point, it makes up for in customer service and reliability through tested development.

4.3.1.1.5 Nvidia Jerson MK1

The Nvidia Jetson MK1 was under consideration for this project on the possibility receiving sponsorship for the microcomputer. The Jetson MK1 has one of the most powerful GPU's in the embedded computer market which would enable the robot to crunch through the heavy math calculations easily where other microcomputers would struggle.

4.3.1.2 Specifications

To compare each microcomputer objectively, important specifications were decided upon which the highest scoring in total would be decided to be used.

4.3.1.2.1 Price

Price is a self-explanatory constraint, as the price of the microcomputer increases this relates linearly with the team's will to implement it due to our limited sponsored budget.

4.3.1.2.2 Frequency

The faster the clock cycle is on a RISC processor (all microcomputers under consideration are ARM based) we could safely assume that the more instructions will be executed per second. Being able to crunch those numbers more time efficiently means that our robot is not be bottlenecked by the CPU and allows for near continuous operation of the robot.

4.3.1.2.3 Cores

The more cores there are in the CPU the more threads our robot can run, once again allowing for a more continuous, less bottlenecked operation of the robot.

4.3.1.2.4 RAM

RAM is very important for our robot as processing images through OpenCV can be very memory intensive as multiple images need to be loaded, processed, and acted upon as soon as possible for our robot to operate functionally, the more RAM that is available to us, the lower the risk of SigSent being RAM bottlenecked is.

4.3.1.2.5 Average Power Consumption

Power Consumption is of critical importance to the vehicle overall, the less power the microcomputer consumes or wastes the longer the vehicle can move, patrol, and report.

4.3.1.2.6 USB, GPIO, I2C, WiFi

USB Ports, GPIO, I2C, and WIFI functionality are crucially important to the vehicle since our sensors require USB and I2C, our simple outputs and simple transducers rely on GPIO, and our communication with SigSent's base station relies on WiFi (pending change). A valid board would require all of these features.

4.3.1.3 Score

A score was calculated based on a formula that maximizes the value for specifications that are positively valuable and minimizes over specifications that undesirable.

4.3.1.3.1 Formula

In order to quantifiably determine the relevance of one microcontroller over another a simple formula was devised after analyzing the available specifications found in the documentation for each of the microcontrollers.

$$Relavence = \frac{Speed * Cores * RAM * USB\ Ports * GPIO\ Pins}{Cost * Average\ Power\ Consumption}$$

Equation 1: Formula for Microcomputer Comparison Score

4.3.1.3.2 Specification Comparison and Score Output

Table 2 Microcomputer Comparison

Name	price (USD)	Processor	Speed (GHz)	Cores	RAM (kB)	Avg PWR (mW)	#USB	#GPIO	I2C	WiFI	Score
rPi 3+	35	BCM2837	1.2	4	1024	5120	4	40	T	T	4.39
rPi Zero W	10	BCM2835	1	1	512	900	0.5	28	T	T	0.8
BBB	68.75	AM3358 Sitara	1	1	512	2500	1	92	T	T	0.27
Zephyr	200	OMAP4430	1	2	1024	3960	1.5	70	T	T	0.27
Jetson	129	Quad ARM® A57/2 MB L2	1.9	4	4096	4700	1	20	T	T	1.03

4.3.1.3.3 Selection Rationale

Going from the score, and corroborated with group consensus, chose the Raspberry Pi 3 Model B. We chose this microcomputer due to its combination of low cost, high CPU performance, low power consumption, large amount of USB ports, and reputable amount of GPIO pins allowing for easy expandability should the project have required unforeseen changes. The Raspberry Pi 3 also has native support for USB, I2C, GPIO, and Wifi, as well as some other nice options such as an ethernet port, speaker output, and HDMI output and USB output. And above almost all else, the Raspberry Pi organization has a significant amount of documentation, support, and community driven forums that aided in quickly being able to understand, develop, and troubleshoot any problems or questions that our team had about using this specific microcomputer.

4.3.2 Microcontroller

A microcontroller was used to complement the microcomputer on the robot. The two systems are on two separate discrete boards to reduce load on the system and provide a more reliable system that controls the actual robot's movement. If the artificial intelligence and control systems logic were placed on a single board, the AI latency could cause the movement scheme to suffer, and vice versa. A microcontroller being used to interface with the physical robot needs to be cheap, low-power, have enough input and output ports to interface with the necessary sensors, and have enough computational power to reduce delays in processing the movement of the robot. These three parameters will be discussed for the popular architectures and implementations below.

4.3.2.1 Atmel megaAVR

Atmel chips are popular for hobbyist projects and for low-power needs. Arduino boards which are ubiquitous in the embedded world today use the Atmel Atmega IC. This chip can be taken off of the development board and placed in a PCB very easily. For prototyping purposes, the chip can remain on the Arduino (or third party) board until the PCB has been created and the functionality has been verified to be correct. Below are discussions on the major Atmel chips used today.

4.3.2.1.1 ATmega328

This Atmel chip is 8-bit with 32KB of ISP flash memory that can "read-while-write". It also has 1KB EEPROM, 2KB SRAM, 23 General Purpose Input/Output (GPIO) lines, 32 General Purpose Registers (GPRs), three timer/counters, internal/external interrupts, USART serial programming, SPI serial port, 6-channel 10-bit ADC, watchdog timer, and power saving modes. The ATmega328 has an operating voltage of 1.8-5.5V. From this operating voltage range, a range of clock speeds can be achieved as seen in the table below.

Table 3: Operating speeds at voltage ranges

Clock Speed	Operating Voltage
0-4MHz	1.8-5.5V
0-10MHz	2.7-5.5V
0-20MHz	4.5-5.5V

4.3.2.1.2 ATmega1280

The Atmega1280 is a higher performance Atmel chip that is still low-power. It has 128KB ISP flash memory, 8KB SRAM, 4KB EEPROM, 86 GPIO lines, 32 GPRs, real time counter, six timer/counters, PWM, 4 USARTs, SPI, 16-channel

10-bit ADC, and a JTAG interface. The ATmega1280 has a performance of 16 MIPS at 16 MHz with an operating voltage of 2.7-5.5V.

Table 4: Operating speeds at voltage ranges

Clock Speed	Operating Voltage
0-8MHz	2.7-5.5V
0-16MHz	4.5-5.5V

4.3.2.1.3 ATmega2560

The ATmega2560 chip employs a larger 256KB ISP flash memory, 8KB SRAM, 4KB EEPROM, 86 GPIO lines, 32 GPRs, six timer/counters, PWM, 4 USARTs, SPI, 16-channel 10-bit ADC, and a JTAG interface. It has similar specifications to the ATmega1280, however has double the flash memory. Its operating voltage is a narrower range of 4.5-5.5V with a clock speed ranging from 0-16MHz.

4.3.2.2 MSP430

The MSP430 is produced by Texas Instruments (TI). It is a group of 16-bit CPUs that are built for low power and are sold at cheap prices. As noted by the different families of chip implementations below, TI follows a naming pattern for each group of MSP430 chips. MSP430 signifies that it belongs to that specific architecture. The next letter indicates the memory type or its specific application. Flash memory chips use a “F” to identify themselves. A “G” is used to denote items used for medical instrumentation. One chip that does not follow this naming convention though is the MSP430FG2xx family. Below are some popular implementations of the MSP430 architecture produced by TI that we researched for use as the primary microcontroller in SigSent.

4.3.2.2.1 MSP430x1xx

This series of MSP430 chips is very basic, not including an embedded LCD controller. They can use flash (1-60KB) or ROM (1-16KB) based memory, and 128 B -10KB of RAM. They have a performance score of 8 MIPS. They have an operating voltage of 1.8-3.6V. The x1xx series includes 14/22/48 GPIO lines, 10/12-bit SAR (Successive Approximation) ADC. They have several integrated peripherals. To name a few key items (and similar ones to the Atmel chips), two 16-bit timers, a watchdog timer, brown-out reset, USART, 16x16 multiplier, and a temperature sensor. These chips have three different operating modes that have low levels of current draw. In order from least to greatest current draw: RAM retention mode (0.1 µA), real-time clock mode (0.7 µA), and MIPS active (200 µA). The x1xx chips have a wake-up time from standby under 6 µs.

4.3.2.2.2 MSP430F2xx

The F2xx series adds more performance at a lower power usage than the x1xx series. It includes a very-low power oscillator (called the VLO). The F2xx chips feature 1-120KB of flash, 128B-8KB of RAM, 10/11/16/24/32/48 GPIO lines, 10/12-bit SAR ADC, and 16/24-bit Sigma Delta ADC. In addition to the peripherals from the x1xx series of chips, the F2xx family has I^2C support and operational amplifiers. Its power modes from least to greatest current draw are: RAM retention (0.1 µA), standby using the VLO (0.3 µA), real-time clock (0.7 µA), MIPS active (220 µA). These chips have a wake-up from standby time under 1 µs.

4.3.2.2.3 MSP430G2xx

The G2xx series are considered “Ultra-Low Power”. They feature the same 16 MIPS performance, VLO, 1.8-3.6V, and I^{2}C in a smaller package, with less GPIO pins. The power modes are similar to the F2xx series, except the VLO mode draws 0.4 µA instead of 0.3 µA. The device specifications are as follows: 0.5-56 KB flash, 128 B – 4 KB RAM, 10/16/24/32 GPIO lines, and 10-bit SAR ADC. The differing peripherals are: three 16-bit timers (one higher than the other series) and capacitive touch I/O.

4.3.2.2.4 MSP430x3xx

The x3xx series includes an LCD controller, increasing its portability. EEPROM memory was not included in this series, instead using one-time programmable EPROM. They operate from 2.5-5.5 V. The x3xx specifications include: 2 – 32 KB ROM, 512 B – 1 KB RAM, 14/40 GPIO lines, 14-bit SAR ADC, and an integrated LCD. Their power modes are: RAM retention (0.1 µA), real-time clock (0.9 µA), and MIPS active (160 µA), and a wake-up time of under 6 µs.

4.3.2.2.5 MSP430x4xx

This series is said to be “ideal for low power metering and medical applications” [41]. It has a low operating voltage of 1.8-3.6V. These chips include Frequency Locked Loop (FLL) and Supply Voltage Supervisor (SVS) for better clock synchronization. Its specifications are: 4 – 120 KB flash/ROM, 256 B – 8 KB RAM, 14/32/48/56/68/72/80 GPIO lines, 10 – 12-bit SAR ADC, and 16-bit Sigma Delta ADC. The x4xx chips have a CPU speed of 8 MIPS. Its unique peripherals unavailable in the aforementioned series are a 32x32 multiplier, ESP430, and SCAN_IF.

4.3.2.2.6 MSP430x5xx

The x5xx series chips have a higher maximum clock rate of 25 MHz while still operating with low-power constraints and putting out 25 MIPS. It has an operating voltage of 1.8 – 3.6V. Its specifications are: up to 512 KB flash, up to 66 KB RAM, 10/12-bit SAR ADC, 29/31/47/48/63/67/74/87 GPIO lines, high resolution PWM, and a backup battery switch among other similar peripherals from its sister families. Its power modes are: RAM retention (0.1 µA), real-time clock (2.5 µA), and MIPS active (165 µA). The wake-up time is less than 5 µs.

4.3.2.2.7 MSP430x6xx

The x6xx series chips can also run up to 25MHz with 25 MIPS. It operates at 1.8 – 3.6V as well. It features a special power management module for better power consumption and the USB integrated in it. Its specifications are: up to 512 KB flash, up to 66 KB RAM, 12-bit SAR ADC, 74 GPIO lines, USB, LCD, power management module, and real-time clock (RTC). Its power modes are: RAM retention (0.1 µA), real-time clock mode (2.5 µA), and MIPS active (165 µA). It has a wake-up time of under 5 µs.

4.3.2.2.8 RF SoC (CC430)

The RF SoC board integrates an RF transceiver at under 1 GHz, with a 1.8 – 3.6V. Its specifications are: 20MHz, up to 32 KB flash, up to 4 KB RAM, 12-bit SAR ADC, 30/44 GPIO lines, and peripherals similar to the previous models (LCD, power management module, RTC, etc.). The power modes are: RAM retention (1 μA), real-time clock (1.7 μA), MIPS active (180 μA).

4.3.2.2.9 FRAM Series

This series of chips features memory access speeds that are 100 times faster than the traditional flash memory times. FRAM does not require power for writes, so if power is lost, writes can still be finished. FRAM can be written over 100 trillion cycles. EEPROM is not needed because of this resilience. Its specifications are: 8 – 24 MHz speed, 4 – 128 KB FRAM, 0.5 – 2 KB RAM, 10 or 12-bit SAR ADC, 17 – 83 GPIO lines. It features peripherals rom its lower level sister series and also a new Extended Scan Interface, AES (Advanced Encryption Standard), and IR modulation. Its power modes are: RAM retention (.320 μA), real-time clock (0.35 μA), and MIPS active (82 μA).

4.3.2.2.10 Low Voltage Series

There are two microcontrollers in the Low Voltage Series. They are the MSP430C09X and MSP430L092. Their low operating voltage range is 0.9 – 1.65V. Its specifications are: 4 MHz speed, 1 – 2 KB ROM, 2 KB SRAM, 8-bit SAR ADC, 11 GPIO lines, two 16-bit timers, SVS, comparator, and the other basic peripherals available to all MSP430 implementations. Its power modes are: RAM retention (1 μA), real-time clock (1.7 μA), and MIPS active (180 μA).

4.3.2.3 Selection Rationale

From the microcontrollers to consider and their features enumerated in 4.3.2.1 & 4.3.2.2, the best choice for our microcontroller, considering the different characteristics of memory on the chips, GPIO, PWM outputs, USART pins, and I2C pins available, as well as the critical feature of ease of use and integration into the project with the microcomputer selection in mind, then the choice was clearly the ATmega328 chip. This has enough output ports and memory to handle the code size needed without providing too much over head on the project for this sub-system. The availability to integrate the Arduino framework into the project also made for a very easy way integrate the sub-system into the other systems with the needed outputs and inputs.

This microcontroller allowed us to quickly, cheapy, and easily integrate a controls sub-system onto our robot and allowed more time and money to focus on developing new and novel concepts that this robot is attempting to accomplish without reinventing the microcontroller system since the Arduino framework has become ubiquitous in the amateur hobbyist community.

4.3.3 GPIO extenders

Since SigSent has numerous input and output devices connected to the microcontroller it is a wise decision to make use of a GPIO extender/extension.

This would be an IC chip that would interface with the microcontroller over some communication protocol and handle commanding a portion of the outputs. This frees up the GPIO pins on the microcontroller allowing for less computation or processing by the microcontroller and more focus on commanding of the I/O devices.

4.3.3.1 Pulse width modulation extender

The majority of the I/O devices that interface with the microcontroller are the servo motors that drive the movement of the system. This can really hinder our performance as the microcontroller is much slower in comparison to the clock speeds that the microcomputer runs at. This means efficiency and latency is of the utmost of importance when new commands from the microcomputer are being sent to the microcontroller. To ensure a clean command of the 18 servo motors, a pulse width modulation (PWM) extension integrated circuit (IC) is used. This frees up commands and allows for more inputs if they are necessary.

4.3.3.1.1 PCA9685

This is a 16 channel LED controller, with each channel having a 4096-step PWM brightness control. The PCA9685 has a programmable frequency output from 24Hz to 1526Hz. This chip uses I^2C as a communication protocol. The PCA9685 has an operating voltage of 2.3-5.5V with inputs and outputs being 5.5V tolerant. This chip has a driving current capability of up to 25mA. This IC also a fast-mode that allows it to 1MHz on the I^2C bus. It also has the option for an external clock input that will accept up to 50MHz, instead of the internal 25MHz oscillator, allowing for synchronization of multiple devices.

4.3.3.1.2 TLC5940

This is a 16 channel LED driver, with each channel having a 4096-step grayscale PWM brightness control. It uses serial communication and has a data transfer rate of 30MHz. The TLC5940 has an operating voltage of 3-5.5V. This chip has a driving current capability of up to 60mA on less than 3.6V and up to 120mA on greater than 3.6V. This IC also has thermal protection in the form of an error flag that is thrown out the error handling pin. This does need a clock signal to shift incoming serial data for output.

4.3.3.1.3 TLC5947

This is a 24 channel LED driver, with each channel having a 4096-step grayscale PWM brightness control. It uses serial communication and has a data transfer rate of 30MHz. The TLC5947 has an operating voltage of 3-5.5V. his has a driving current capability of up to 30mA. This IC also has thermal protection in the form of an automatic shutdown at over temperatures and restarts under normal temperatures again. This does need a clock signal to shift incoming serial data for output. This chip has an internal oscillator of 4MHz.

4.3.3.1.4 SN3218

This is an 18 channel LED driver, with each channel having a 256-step PWM brightness control. This chip uses I^2C as a communication protocol with a maximum clock frequency of 400kHz. The SN3218 has an operating voltage of 2.7-5.5V. his has a driving current capability of up to 23mA. This IC also has thermal protection in the form of an error flag that is thrown out the error handling pin.

4.3.3.2 Selection Rationale

From the pulse width modulation extenders to consider and their features enumerated in 4.3.3.1, the best choice for our pulse width modulation extenders, considering the distinctive characteristics of number of available channels, step size, output frequencies, communication protocols, operating voltage and output current for each channel, then the choice was clearly the TLC5947 chip. This has enough output ports channels to handle the amount of servo motors needed without providing too much overhead on the project for this sub-system. While this has the largest number of channels from the chips put under comparison, it does have enough channels for all the leg movements on SigSent while leaving some available to add a pan and tilt to the camera onboard if need be. Also with this amount of PWM outputs all control commands can be sent to this single chip from the microcontroller allowing for slimming of code sizes and the need for only one command to be sent to this board to begin movements of the servo motors.

This pulse width modulation extender allowed us to quickly, cheapy, and easily integrate extended output to the controls sub-system for the microcontroller onto our robot and allowed us more time and money to focus on developing new and novel concepts that this robot is attempting to accomplish without reinventing the output capabilities for microcontrollers or having a niche microcontroller with extended PWM capabilities but risk the ease of use and integration into the system for it.

4.3.4 Force/Pressure Sensor

In order for SigSent to have an active suspension system, the system would need to know whether it is touching the ground or not and if it is, how planted or hard is the leg pressing into the ground. This would give a relatively accurate reading from all the legs to be able to tell if SigSent has a good stance/footing at the current moment.

This especially becomes vital when the system is traversing over rough terrain that is bumpy in nature. SigSent needs some feedback as to whether it is touching the ground and if it is currently holding itself up on the current legs that are touching the ground or is off balance and is about to become unstable. The plan for the force/pressure sensor would be to place it on the end effectors of the two middle legs and also in between the bearing and the holder that the motors are mounted to. This would give an accurate reading on the end effectors and their contribution to stability as well as the other four legs. This coupled with the inertia measurement unit reading would show what the system needs to do to remain stable or to actively stabilize itself from a position. Due to the proposed locations/placements of the force sensors, it is a requirement that the force sensors are as thin/flat as possible, above most other specification, to not hinder the design or the solutions to the kinematics equations by adding size to the end effector, etc.

4.3.4.1 Force Sensors under consideration

The following force sensors are those that have properties viable to our project and will be objectively compared such that the best option under our constraints is chosen for use in SigSent's development.

4.3.4.1.1 SingleTact Capacitor force sensors

This force sensor is capable of measuring up to 100lbs of force while being 0.35mm thick. This force sensor is capable of being three times more sensitive than a resistive force sensor. The sensing area of this sensor is 8mm or 15mm in diameter. It has an I^2C interface making it easy to set up with microcontrollers. The SingleTact can also operate in temperatures up to 200^oC with a temperature sensitivity of $0.2\%/^oC$. This sensor has a repeatability performance of less than $\pm2.5\%$, response time of less than 1 millisecond, and drift of less than 2% per logarithmic time scale. This is only an analog sensor and would require an amplifying circuit and analog to digital converter to properly measure the reading from the force sensor. This product family does have an accompanying electronic circuit that amplifies and outputs it as a voltage signal or converts the signal to an I2C signal for direct reading of the measurement.

4.3.4.1.2 Interlink electronics FSR 400 Series

This force sensor is capable of measuring up to approximately 5lbs of force while being 0.3mm thick. The sensing area of this sensor is 5 to 13mm in diameter. The FlexiForce can also operate in temperatures up to 85^oC. This force sensor is also flexible and relatively easy to implement only needing an op amp circuit to get the output. This sensor has a repeatability performance of less than $\pm 2\%$, response time of less than 3 microseconds. This is only an analog sensor and would require an amplifying circuit and analog to digital converter to properly measure the reading from the force sensor.

4.3.4.1.3 Tekscan FlexiForce ESS301

This force sensor is capable of measuring up to 100lbs of force while being 0.203mm thick. The sensing area of this sensor is 9.53mm in diameter. The FlexiForce can also operate in temperatures up to 85^oC with a relative humidity of up to 95%. This force sensor is also flexible and relatively easy to implement only needing an op amp circuit to get the output. This sensor has a repeatability performance of less than $\pm2.5\%$, response time of less than 5 microseconds, and drift of less than 3.8% per logarithmic time scale. This is only an analog sensor and would require an amplifying circuit and analog to digital converter to properly measure the reading from the force sensor.

4.3.4.2 Selection Rationale

From the force sensors to consider and their features enumerated in 4.3.4.14.3.3.1, the best choice for our force sensor, considering the distinctive characteristics of capable weight able to be measured, sensor size, sensor thickness, communication protocol, operating temperature and humidity, and sensor measurement repeatability, then the choice is clearly the SingleTact Capcaitor Force sensors. This force sensor, while very similar to the other force sensors, has one clear advantage of coming with the amplification circuit already set up and ready to output data. As well as the added advantage of the ability of outputting I2C data directly to a microcontroller. This would save a lot of overhead and possible wasted hours calibrating the amplification circuitry to try and get a readable and reliable output.

This force sensor would allow us to quickly and easily integrate a feedback input for the controls sub-system to the microcontroller onto our robot and allow us more time to focus on developing new and novel concepts that this robot is attempting to accomplish without reinventing the amplification circuits needed for the proper outputs. Due to limited budgeting and higher priorities in sensors or actuators, this sensor was tabled for a later date to integrate into the controls sub-system with the microcontroller, if time and money allowed for it.

4.3.5 Lidar

Lidar or Light Imaging, Detection, And Ranging or LIght raDAR is a "surveying method that measures distance to a target by illuminating that target with a pulsed laser light, and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to make digital 3D-representations of the target." [42]. In our robot we used a 2D lidar supplied by the Robotics Club at the University of Central Florida [43]. This Lidar is a Hokuyo UTM-30LX lidar which is capable of seeing 30 meters in day or night with a 270* view. The Lidar outputs a long vector of measurements in millimeters for each individual point, having a data point once every .25* for a total of 1440 steps per full revolution of the laser assembly. [44] This data could be easily represented as an absolute depth data that can be input into Gmapping (4.2.5.1)

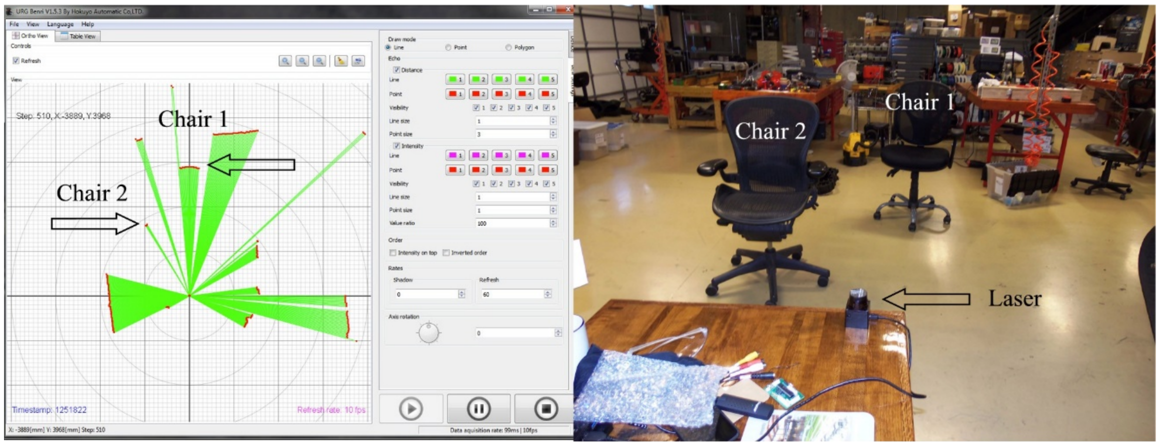


Figure 17: Representation of Lidar output compared to Image [45]

4.3.6 Camera

To facilitate the computer vision in SigSent, a camera that is versatile independent on the time of day, has a resolution that is high enough for performant image classification, and meets our pricing standards was chosen.

4.3.6.1 CCD v CMOS

There is an important distinction to be noted between CCD and CMOS based camera sensors, while CMOS has become popular in consumer cameras due to its very low price and small size, it is at the price of a significantly higher noise in image quality. CCS, while an order of magnitude more expensive than CMOS sensors has a much lower noise, creating significantly more reliable images that will lead to less errors in the robot’s computer vision.

4.3.6.2 Day Vision versus Night Vision

Since SigSent is designed to be operated during both daytime and nighttime operations, care must be taken into how SigSent’s cameras will take in light both during the daytime, where the Sun could easily wash out images, and the nighttime where there could be very minimal lighting. This allows for three options for the robot to operate in both daytime and nighttime operations without human intervention. The first is to have a normal camera and equip the robot with a powerful light to flood the area in front of the camera with enough light for the robot to gain enough data to determine if there is a human out of place - however this makes the robot's position very easily known and the robot easier to avoid. The second is to have an IR camera that can detect IR wavelengths, giving the camera a form of night vision, however this means that the cameras would be at a disadvantage during daytimes as some colors would be washed out due to the sun broadcasting IR light. The third option is to have a camera similar to the prior option but with an automated IR-CUT filter that could operate during the daytime, allowing the camera to filter out IR light during daytime and remove that filter during nighttime allowing the camera to detect the IR light again. Each camera has some combination of these features and each could theoretically be modified to work in both day or night (by adding or removing an IR-CUT filter) or adding a flashlight, depending on the features and their determined usefulness one of the above options had to be chosen.

4.3.6.3 Cameras Under Consideration

The following cameras are those that we considered due to their compatibility and performance.

4.3.6.3.1 Raspberry Pi Cameras

As the Raspberry Pi serves as our robot's microcomputer, cameras designed to work with the hardware are important to distinguish.

4.3.6.3.1.1 Infrared 500W Focus Adjustable Night Vision Camera Module - BLACK

This CMOS cameras has a dubious claim of having a 500 watt IR LED to light its surrounding, capability of detecting IR light, and a significant lack of documentation. This camera coming with an IR led built in is a significant feature which reduces the amount of modification needed for the camera to operate at night, however the lack of documentation for this camera significantly increases the risk of this product being unreliable and hard to integrate into the system.

4.3.6.3.1.2 Raspberry Pi Infrared Camera Module (NoIR) V2

This is a CMOS camera developed by Raspberry Pi and thus has workable amount of documentation associated with it as well as a more trustable brand to trust in. This camera uses the Sony IMX 219 PQ CMOS sensor with its IR Blocking filter removed allowing it to see at night when IR led's are present, however with the IR filter removed images during the daytime may be washed out, requiring the team to modify a IR filter to cover the lens during daytime operations. This camera has a high resolution and a high FPS as well as automatic exposure control, automatic white balance and automatic black level calibration allowing for us to easily retrieve more color accurate and more up-to-date images from the sensor. This camera however lacks an option to adjust focus which means that the camera may be blurry if an object of interest is too close or too far away from the camera.

4.3.6.3.1.3 Raspberry Pi Camera Module w/ Adjustable Focus and Night Vision

This CMOS camera developed by Waveshare comes with an adjustable focus an am Omnivision OV5647 sensor with its IR filter removed. This sensor can capture 720p images at 60FPS or 640p images at 90FPS which is ideal so that we are always processing the most up to date images for SigSent. This camera also comes equipped with IR LED's to help see at night, however there is no IR-CUT filter on it so we would need to modify that filter on to prevent images being washed out during the daytime.

4.3.6.3.1.4 Raspberry Pi Camera Module w/ Fisheye Lens and Night Vision

This CMOS Camera with the same sensor as in 4.3.6.3.1.3. Is fitted with a fisheye lens with its IR filter removed. However this camera is not equipped with IR leds. These LED's will need to be mounted with the camera, and the an IR filter installed to activate during daytime operations. The Fisheye lens will be very useful as it will allow the robot to see more of its surroundings while moving less, conserving overall system energy.

4.3.6.3.1.5 Raspberry Pi Camera Module w/ IR Cut Filter

This CMOS Camera with the same sensor as in 4.3.6.3.1.3. Is fitted with an IR-CUT filter that we can use to automatically switch between nighttime and daytime activity without need for modification or human interaction. This camera also includes IR LEDs allowing for the camera to see during night without additional LEDs needing to be installed.

4.3.6.3.2 BlackBird 2 3D FPV Camera

This 3D camera created by FPV3DCAM uses a custom onboard IC that can send 680p images at 60hz in a variety of standard 3D formats. This camera has an impressive field of view and a high signal to noise ration for a CMOS sensor (45db). The camera also uses low power at 1.8W. According to the camera's documentation, we may need to install a heatsink on the camera to prevent overheating. This camera has an IR filter installed by default and does not appear to be modifiable, therefore for this camera to operate at night we would have to install a large flood light on the vehicle.

4.3.6.3.3 Pixy CMUcam5 Image Sensor

The Pixy CMUcam5 has a large community support and uses the Omnivision OV9715 CMOS sensor, outputting 720p at 30fps or 640p at 60fps. What distinguishes this camera from the others is that it has an onboard microcontroller which can handle basic image recognition, outsourcing some of the computational power from the microcomputer controlling the robot to within the camera module itself. This camera comes with a significant amount of documentation and community support reducing risk if there is an error or problem with the board or in our attempts to integrate it into our system. This camera does come with a non-modifiable IR filter installed, so in order for this camera to accomplish SigSent’s goals, we would need to install a large flood light on the vehicle.

4.3.6.3.4 Logitech C920

The Logitech C920 is a COTS web camera designed for video calling, while this camera is not specifically the best choice for this project on paper, this camera has by far the easiest implementation since it uses generic USB drivers that work immediately with Linux being used on our microcomputer. Another benefit of the Logitech C920 is that the camera has a significant amount of community support both in the general market and in the hobbyist fields.

4.3.6.3.5 FLIR Point Gray: Firefly MV 0.3 MP Color USB 2.0 (Aptina MT9V022)

This CCD camera is a professional grade camera designed specifically for computer vision in an industrial environment. It captures images at 752x480 at 60FPS and uses a standardized CS-mount lens allowing for a custom (yet pricey) ideal lens to be selected in the future. This camera will be able to sense IR wavelengths since there is no internal IR filter built into it. However, it would require IR LEDs to illuminate an object of interest and an IR-CUT filter added for daytime and nighttime operation. This camera has by far the best documentation and sensor sensitivity compared to the other sensors in folds and uses, like the Logitech c920 in 4.3.6.3.4. A generic USB driver that is easy to integrate into a Linux environment on SigSent's microcomputer.

4.3.6.4 Specifications

The important specifications encompassed by a computer vision camera are outlined below to be compared among each candidate camera.

4.3.6.4.1 Price

Price is a self-explanatory constraint, as the price of the camera increases this relates linearly with the team's will to implement it due to our limited sponsored budget.

4.3.6.4.2 Image Quality

Image Quality is an almost qualitative measurement between these cameras since there is a significant variability between each of the camera's reported specification and how they choose to both measure (or not measure) them. Due to this image quality is an amalgamation of the resolution, the number of megapixels, the Signal to Noise Ratio, Field of View, and focal length of each camera. Unfortunately, assigning some number to the above would never be an accurate representation.

4.3.6.4.3 Frames Per Second

Frames per second is a crucially important specification because this both determines how quickly our cameras are able to capture a situation but also can help make up for poorer image quality. The higher the FPS is, the more likely we should choose that camera.

4.3.6.4.4 Night Vision

Since the goal of SigSent is to be able to effectively work in both nighttime and daytime environments, it is important to weight into the decision on whether or not the camera has the ability to natively see IR light (the easiest/cheapest way to achieve night vision). If the cameras can see IR light then the camera will need a way to block IR light during the daytime to prevent the camera's images being washed out by the sun, while if the camera cannot see IR, the robot will need to have a flood light installed on it so that that visible wavelengths can be seen at night.

4.3.6.5 Results

The results obtained below helped determine which camera was chosen for the SigSent. They are outlined in tabular form to be easily compared.

4.3.6.5.1 Comparison

Table 5: Comparison of Cameras

Name	Cost (USD)	Resolution	Megapixel	S/N (db)	Frame Rate	IR (bool)	Docs (bool)
PiCam IR Adj w/ LED	18.67	NA	5	NA	NA	T	F
PiCam IR Adj Official	24.99	720	5	44.56	60	T	T
PiCam IR Adj	21.59	1080	5	36	30-120	T	F
PiCam IR Fisheye	32.99	1080	5	36	30-121	T	F
PiCam IR Adj Cut w/ LED	27.99	1080	5	36	30-122	T	F
Blackbird 2 3D	179	3d = 680*512	NA	45	60	F	T
Pixy CMUcam5	67	1280x800	NA	39	50	F	T
Logitech C920	55.68	1920x1080	3	NA	30	F	T
Firefly	275	752x480	0.3	52	60	T	T

4.3.6.5.2 Selection Rationale

From the cameras to consider and their features enumerated in 4.3.6.3 and in the comparisons in the table aboe, the best choice for our camera, considering price, image quality, frames per second, signal to noise ration, modifications to work in night and day, and the amount of documentation/support for each product there is a decently clear winning of 4.3.6.3.1.5 (PiCam IR Adj Cut w/ LED) which has an IR-Cut filter and LEDs already installed, uses the Raspberry Pi dedicated camera point, and the well documented Omnivision OV5647 sensor. This sensor has the modest price of $26.29 and requires the fewest modifications to work with our scenario while capitalizing on Raspberry Pi's hardware (decided in 4.3.1.3.3) and all the documentation and community support that is associated with Raspberry Pi. This Camera allowed us to quickly, cheapy, and easily integrate vision onto our robot and allowed us more time and money to focus on developing new and novel concepts that this robot is attempting to accomplish without reinventing the wheel on already established technology.

4.3.6.5.3 Prototype Changes

While assembling all of the hardware for SigSent, the Raspberry Pi was swapped out for a newer model that did not support the kernel module for the PiCam we selected. For our prototype build, we instead went with the integrated camera on the Playstation Eye which was being used for its microphone. This meant we did not have to purchase a new component and did not add anymore weight to the build. Future builds should use the PiCam if possible for the integrated selectable IR filter.

4.3.7 IMU

An IMU is used to detect the rotational acceleration of the SigSent robot and also keeping track of the vehicle's orientation. Various IMU units are detailed below to be compared and scored such that the most optimal was selected for the SigSent.

4.3.7.1 IMU's under consideration

The IMU's that were researched below were under consideration for use in SigSent. There specifications were then found and compared.

4.3.7.1.1 MPU-9250

This IMU excels in power efficiency and a very high refresh rate. This IMU has an average sensor stability in its price range but at its high refresh rate this IMU shines above others.

4.3.7.1.2 LSM9DS1TR

This IMU has one of the best gyroscope sensors within its price range, and very detailed documentation, however with its refresh rate being far below average even with its price range this sensor is almost nonviable.

4.3.7.1.3 Sparton AHRS-8

This is a complete Attitude and heading reference system (AHRS) unit with extremely accurate sensors, however its price makes this sensor cost prohibitive, however the company that makes these sensors has been known to sponsor projects.

4.3.7.1.4 VectorNav VN-100

The VectorNav VN-100 boast an extremely high refresh rate of 400hz and sensors comparable or even better than the AHRS-8, however once again its price makes this sensor cost prohibitive, however the company that makes these sensors has been known to sponsor projects.

4.3.7.2 Specs

Specifications important in choosing an IMU for SigSent's use are detailed below such that each different unit can be objectively compared so that the best use-case for our project would be chosen that meets our demands and fits within our constraints.

4.3.7.2.1 Price

Price is a self-explanatory constraint, as the price of the IMU increases this relates linearly with the team's will to implement it due to our limited sponsored budget.

4.3.7.2.2 Degrees of Freedom

IMU's are incredibly important for autonomous navigation and control systems since they measure vital information such as absolute heading, acceleration in the 3 linear dimensions and acceleration in the 3 rotational dimensions. This totals to 9 degrees of freedom (absolute X,Y,Z rotational directions from the magnetometer, the X,Y,Z linear accelerations, and roll, pitch, and yaw with the gyroscope.) Additionally there can sometimes be an added 10th degree of freedom in the implementation of an absolute or relative altimeter. For our platform to operate as expected we will need the 9 degrees of freedom to keep our vehicle autonomous and functioning appropriately.

4.3.7.2.3 Average Power Consumption

Power consumption is of utmost importance within the project as a whole since the less powered used overall increases the overall lifetime of the robot on a singular charge.

4.3.7.2.4 Accelerometer Stability Scale Factor

Accelerometer Stability Scale Factor (SSF) is a quantifiable way to measure the accuracy of a Accelerometer by way of measuring the ratio of the sensors output compared to the input (placing the sensor under various linear G Forces), as the input and output is changed the linearity of this is measured as SSF.

4.3.7.2.5 Gyroscope Stability Scale Factor

GyroScope Stability Scale Factor (SSF) is a quantifiable way to measure the accuracy of a gyroscope by way of measuring the ratio of the sensors output compared to the input (placing the sensor under various rotational G Forces), as the input and output is changed the linearity of this is measured as SSF.

4.3.7.2.6 Refresh Rate

Refresh rate for the IMU is critical to the usefulness of an IMU as the more measurements that the IMU is able to produce, the more data we can provide to our sensor integration algorithm (potentially a Kalman Filter) with the GPS to get more and more accurate results that will help our navigation and path planning algorithms.

4.3.7.3 Scores

The scores were calculated such that the positive value of refresh rate boosted the unit's score, and the undesirable specifications would lower a unit's score. Based on the final values, the highest performing unit was chosen for use in the project.

4.3.7.3.1 Formula

In order to quantifiably determine the relevance of one GPS Unit over another a simple formula was devised after analyzing the available specifications found in the documentation for each of the microcontrollers.

Equation 2: Formula for IMU Comparison Score

$$Relavence = \frac{Refresh\ Rate^2}{Cost * Average\ Power\ Consumption * Accelerometer\ SSF * Gyroscope\ SSF}$$

4.3.7.3.2 Specification Comparison and Score Results

Table 6: IMU Comparison Table and Score Output

Name	Cost	DOF	Comm protocol	Voltage	AVG Power	Gyro SSF	Accelerometer SSF	Sample Rate	Overall Score
MPU-9250	10.63	9	i2c	3.3	3.7	16.4	0.061	200	1016.6
LSM9DS1TR	6.33	9	i2c	3.3	4.6	8.75	0.061	80	411.79
AHRS-8	1350	10	USB	5	82.5	0.18	0.023	100	21.69
VN-100	800	10	USB	3.3	45	0.16	0.04	400	694.44

4.3.7.3.3 Selection Rationale

From the consensus of the score and the team, the MPU-9250 is the best option for SigSent based on its excellent refresh to cost, terrific power efficiency and acceptable sensor errors.

4.3.8 GPS

A GPS is necessary to keep track of the absolute position of the SigSent vehicle as it navigates its route. Research had to be done on hardware that would provide the best performance under the constraints allotted to us.

4.3.8.1 GPS's under consideration

The GPS units below were possible considerations for use in the SigSent vehicle. They have their specifications described below to be objectively compared.

4.3.8.1.1 SkyTraq Venus638FLPx

This GPS unit has the best GPS refresh rate of all GPS units under consideration at 20hz. That refresh rate is key because of inherent flaws to the

IMU the GPS having a fast refresh rate means that navigation and mapping errors will be reduced significantly. This sensor however has a lower sensitivity at -165dB meaning that obstructions in the way such as buildings or atmospheric events will have an impact on performance.

4.3.8.1.2 LocoSys LS20031

This sensor has lackluster documentation and a higher cost but with the benefit of using less power than the SkyTraq Venus638FLPx.

4.3.8.1.3 Maestro A2135-H

This GPS unit has a significant amount of documentation, and has the lowest price by far while also having the highest sensitivity of the bunch. However, it has a below average refresh rate of only 5hz and uses the SPI bus while most sensors are using either I2C or USB.

4.3.8.1.4 Linx RXM-GNSS-TM-B

This GPS unit uses a UART interface and has a respectable refresh time with a high sensitivity receiver like in the SkyTraq Venus638FLPx.

4.3.8.2 Specifications

The specifications below encompass the necessary items we value in the performance of the SigSent vehicle. Each of the GPS units that were under consideration have had their details listed out such that the specifications below could be easily compared between each one.

4.3.8.2.1 Price

Price is a self-explanatory constraint, as the price of the GPS increases this relates linearly with the team's will to implement it due to our limited sponsored budget.

4.3.8.2.2 Refresh Rate

The Refresh rate of a GPS unit is how often it is able to contact, calculate, and send out a stable GPS coordinate. This is a critical specification because as the refresh increases the inaccuracies inherently introduced to our navigation and path planning by the IMU are reduced significantly.

4.3.8.2.3 Average Power Consumption

Power consumption is of utmost importance within the project as a whole since the less powered used overall increases the overall lifetime of the robot on a singular charge.

4.3.8.2.4 Sensitivity

Sensitivity is another critical specification, this is the ability of the GPS unit to properly detect and receive the packets of information coming from the GPS satellites around Earth. Measured in decibels, a single unit increase in the positive direction is equivalent to a significant increase in sensitivity.

4.3.8.2.5 Accuracy

Accuracy is a measurement of average tolerance for the GPS device measured in meters. Accuracy says: given a GPS coordinate the true position of the device is within the accuracy given. For all of the GPS units under consideration, the accuracy is given as 2.5m, this is a fairly standard unit and is more of a limitation of the satellites than of the sensors.

4.3.8.3 Scores

To choose the best fitting GPS, they were scored positively based mostly on their refresh rate, as well as their sensitivity, and had their scores negatively affected by their cost, power consumption, and accuracy error.

4.3.8.3.1 Formula

In order to quantifiably determine the relevance of one GPS Unit over another a simple formula was devised after analyzing the available specifications found in the documentation for each of the microcontrollers.

Equation 3: Score for calculating optimal GPS unit selection

$$Score = \frac{Refresh\ Rate^2 * Sensitivity\ as\ a\ linear\ ratio}{Cost * Average\ Power\ Consumption * Accuracy}$$

Table 7: GPS Comparison Table and Score Output

4.3.8.3.2 Specification Comparison and Score Results

Name	Cost	Refresh	Voltage	Power	Comm	Sensitivity	(Linear Ratio)	Accuracy	Total Score
Venus638FLPx	49.95	20	3.3	60	I2C	-165	562.34	2.5	30.02
LS20031	60	5	3.3	41	TTL	-165	562.34	2.5	2.29
A2135-H	20.9	5	3.3	31	SPI	-163	707.95	2.5	10.93
Linx RXM-GNSS-TM-B	34.33	10	3.3	30	UART	-165	562.34	2.5	21.84

4.3.8.3.3 Selection Rationale

From the Score which sums up that the Venus638FLPx is the best option due to its extremely high refresh rate alone. The Venus638FLPx despite its higher power consumption, slightly lower sensitivity, and slightly higher price is the best decision for the team because a high refresh rate for the GPS will ensure that our robot can more accurately keep track of its position and accurately follow a GPS weight point and map its surroundings.

After significant tested after selecting the Venus638FLPx we learned that it may have been niave for us to have placed such a high weight on refresh time and instead to have more heavily weighted or considered the device's sensitivity as

SigSent had significant trouble gaining an accurate GPS lock, especially indoors, and struggled meeting its 2.5m accuracy outdoor, sometimes being as inaccurate as over 10m. A more accurate system with higher sensitivity, or one that has DPGS or some version of a ground based GPS would yield significantly better and more reliable results.

4.3.9 Servo motors

Servo motors will be in control of the legs of SigSent. Precisely a set of three servo motors will be used to operate the movements of each leg with control signals sent to it by the main processor in SigSent's architecture. Various servo motor units are detailed below to be compared and scored such that the most optimal is selected for the control SigSent.

4.3.9.1 Servo motors Under Consideration

The servos motors that were researched below were under consideration for use in SigSent. There specifications were then found and compared.

4.3.9.1.1 HS-755HB Servo

This servo motor is priced at $27.99 – 47.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 4.8-6.0V with the stall torque being 183oz-in(13.2kg-cm) at the max voltage of 6.0V. This servo motor has a speed of 0.23sec per 60^o at the max voltage of 6.0V. Lastly the gear material of the servo motor is made up of Karbonite, with the weight being 3.88oz (110g)

4.3.9.1.2 HS-755MG Servo

This servo motor is priced at $39.99 – 59.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 4.8-6.0V with the stall torque being 200oz-in(14kg-cm) at the max voltage of 6.0V. This servo motor has a speed of 0.23sec per 60^o at the max voltage of 6.0V. Lastly, the gear material of the servo motor is made up of metal, with the weight being 4.12oz (117g).

4.3.9.1.3 HS-765HB Servo

This servo motor is priced at $39.99 – 59.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 4.8-6.0V with the stall torque being 183.31oz-in(13.2kg-cm) at the max voltage of 6.0V. This servo motor has a speed of 0.23sec per 60^o at the max voltage of 6.0V. Lastly, the gear material of the servo motor is made up of Karbonite, with the weight being 3.6oz (102g).

4.3.9.1.4 HS-5646WP Servo

This servo motor is priced at $54.99 – 74.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 6.0-7.4V with the stall torque being 179oz-in(12.9kg-cm) at the max voltage of 7.4V. This servo motor has a speed of 0.18sec per 60^o at the max voltage of 7.4V. Lastly, the gear material of the servo motor is made up of three Metal Gears and one Nylon Gear, with the weight being 2.15oz (61g).

4.3.9.1.5 D645MW Servo

This servo motor is priced at $39.99 – 59.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 4.8-7.4V with the stall torque being 180.1oz-in(12.9kg-cm) at the max voltage of 7.4V. This servo motor has a speed of 0.17sec per 60^o at the max voltage of 7.4V. Lastly, the gear material of the servo motor is made up of metal, with the weight being 2.11oz (60g).

4.3.9.1.6 S9470SV Servo

This servo motor is priced at $99.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 6.0-7.4V with the stall torque being 191.7oz-in(13.8kg-cm) at the max voltage of 7.4V. This servo motor has a speed of 0.09sec per 60^o at the max voltage of 7.4V. Lastly, the gear material of the servo motor is made up of metal, with the weight being 1.90oz (54g).

4.3.9.1.7 HS-8330SH Servo

This servo motor is priced at $89.99, depending on the features included at the time of purchase (if increased rotation oranime continuous rotation is wanted). The operating voltage of this is 6.0-7.4V with the stall torque being 180.53oz-in(13kg-cm) at the max voltage of 7.4V. This servo motor has a speed of 0.07sec per 60^o at the max voltage of 7.4V. Lastly, the gear material of the servo motor is made up of steel, with the weight being 2.32oz (66g)

4.3.9.1.8 DynaMixel AX-12A

This servo motor is priced at $44.99, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 9.0-12V with the stall torque being 212.41 oz-in(15.296 kg-cm) at the max voltage of 12.0V. This servo motor has a speed of 0.07sec per 60^o at the max voltage of 12.0V. Lastly, the gear material of the servo motor is made up of steel, with the weight being 1.88oz (54.6g)

4.3.9.1.9 DynaMixel AX-18A

This servo motor is priced at $94.89, depending on the features included at the time of purchase (if increased rotation or continuous rotation is wanted). The operating voltage of this is 9.0-12V with the stall torque being 254.90oz-in(18.355 kg-cm) at the max voltage of 12.0V. This servo motor has a speed of 0.07sec per 60^o at the max voltage of 12.0V. Lastly, the gear material of the servo motor is made up of steel, with the weight being 1.88oz (54.6g)

4.3.9.2 Specifications

The specifications below encompass general constraints on servo motors. The price, operating voltage, torque performance at these voltages, and speed are all items to consider in comparing the servos.

4.3.9.2.1 Price

Price is a self-explanatory constraint, as the price of the servo increases this relates linearly with the team's will to implement it due to our limited sponsored budget.

4.3.9.2.2 Max Voltage

The maximum voltage of the servos is an important consideration, ideally the servos would be able to be run directly off the batteries, which are 4s (12V-16.8v) however that is not the industry standard, most consumers off the shelf (COTS) servos are between 4.8V-6V with some capable of going up to 7.4V. With this consideration in fact, we want to choose a servo with a higher voltage because that will lead to less losses when regulating the DC power of the batteries down to the voltage required by the servo.

4.3.9.2.3 Torque at Max Voltage

The toque at max voltage is the critical spec for the servos as a large amount of torque will be required to properly move the legs of SigSent, the higher the torque the better, we choose to compare the torque at max voltage over the minimum voltage since the voltage being sent to the servos will be regulated, and regulated at the maximum compatible voltage the servos can take. This also allows us to more accurately anticipate the amount of torque that the servos can provide as torque varies with voltage.

4.3.9.2.3.1Minimum Required Torque

Due to the design of the legs of the robot, SigSent will have a critical requirement of minimum required torque in order for the hexapod legs to carry the weight of the robot in a stationary position but also in an active suspension or walking configuration. To calculate the stationary position a free body diagram needs to be constructed to calculate the moment arm from the main body to the joints of the legs on sig sent. As seen below in Figure 18 & Figure 19 & Figure 20:

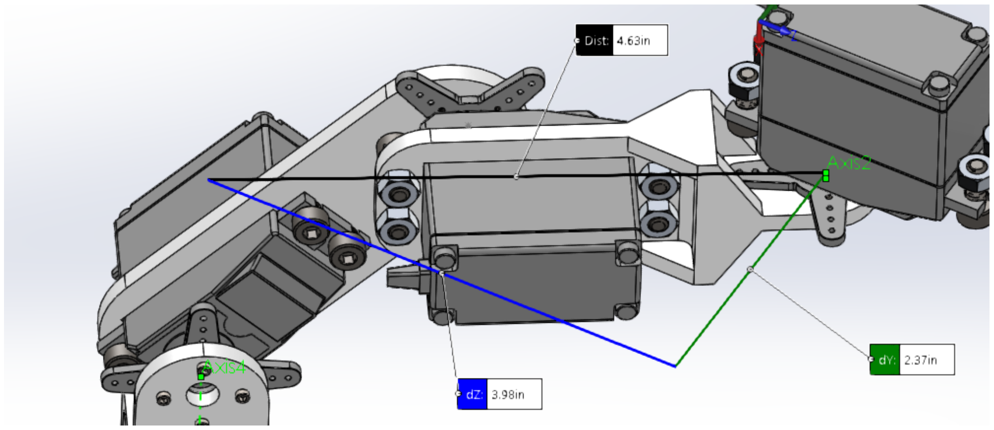


Figure 18: Moment arm for stationary extended position at 135^{0}(degree) position

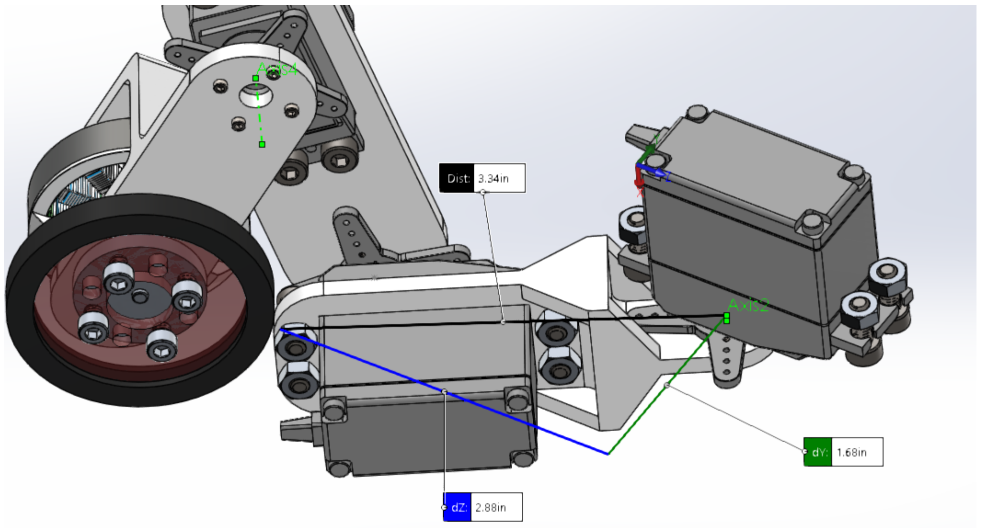


Figure 19: Moment arm in 105º degree angle position

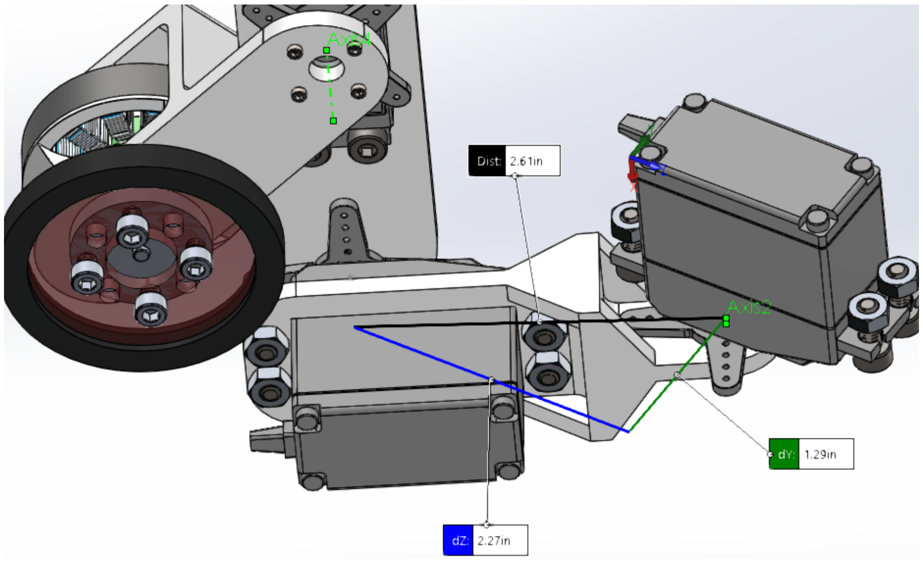


Figure 20: Moment arm in 90º degree angle position

Since a model of the current mechanical design has already be created pulling the measurements from the design configuration to calculate the toque on the joints created from the robot's body/weight. From this it was determined in the standard stationary configuration shown in Figure 18: Moment arm for stationary extended position at 135^0(degree) position that the minimum torque needed for movement of the body is 217.75 oz-in with an approximation of total weight of the system at 4kg. This makes the leg configuration shown in Figure 18 to be not viable as this leaves no margin of safety for the servo motors to operate. Thus, making a need for a different leg configuration, this make the configuration in Figure 19: Moment arm in 105^o degree angle position more viable as the required torque for this position is 157.087 oz-in. This leaves a margin of safety of approximately 35% making the configuration very feasible. The last configuration in Figure 20: Moment arm in 90^o degree angle position makes a required torque of 122.75oz-in. This leaves a margin of safety of approximately 73%, which allows SigSent to have a range of motion within its minimum required torque for certain configurations.

4.3.9.2.4 Speed at Max Voltage

The speed at max voltage is a measurement of the amount of time it takes for a servo to move 60 degrees under no load, we need the speed of the servos to be as high as possible so that the legs can move in a quick and responsive fashion to properly support SigSent and the various multi-terrain environments that it must operate in.

4.3.9.2.5 Weight

Since the servos are within the legs, the servo weight must be a minimum to reduce the amount of torque required for the servos to push and pull while in walking mode or in suspension during driving.

4.3.9.3 Scores

The scores below give a positive weight to better torque and speed. A negative weight is given to cost and weight of each servo.

4.3.9.3.1 Formula

In order to quantifiably determine the relevance of one motor over another a simple formula was devised after analyzing the available specifications found in the documentation for each of the motors.

Equation 4: Score for Servo Motors

$$Score = \frac{Torque * Speed}{Cost * Weight}$$

4.3.9.3.2 Comparison and Score Results

Table 8: Specification Comparison of Servo Motors

Description	Weight	Stall Torque	Speed	Price	Score
HS-755HB	3.88oz (110g)	183	0.23	$28.00	$0.00
HS-765HB	3.6oz (102g)	183	0.23	$40.00	$0.00
HS-755MG	4.12oz (117g)	200	0.23	$40.00	$0.00
HS-5646WP	2.15oz (61g)	179	0.18	$55.00	$0.27
D645MW	2.11oz (60g)	180	0.17	$40.00	$0.36
S9470SV	1.90 oz. (54g)	191.7	0.09	$100.00	$0.09
HS-8330SH	2.32oz (66g)	180	0.07	$90.00	$0.06
DynaMixel AX-12A	2.32oz (66g)	180	0.1695	$45.00	$0.29
DynaMixel AX-18A	2.32oz (66g)	180	0.1031	$95.00	$0.08

4.3.9.3.3 Selection Rationale

From the servo motors to consider and their features enumerated in 4.3.9.1, the best choice for our servo motors, considering the distinctive characteristics of weight, stall torque, speed, and price, then the choice is the DynaMixel AX-12A. This component needed major consideration and comparison as it is a critical requirement for the hexapod to even move its legs in a walking configuration let alone standing upright under its own weight. The weight of this servo compared to its stall torque puts it in a league above most others. The final reasoning was that DynaMixel sells these servo motors in bulk orders allowing for a massive price reduction that fall within our projected budget without putting that component at its limits of pricing.

This servo motor will allow us to quickly, cheapy, and easily create and integrate the controls sub-system for the microcontroller onto our robot and allow us more time and money to focus on developing new and novel concepts that this robot is attempting to accomplish without reinventing or redesigning the whole leg system and its movement scheme.

4.3.10 Motors

The Motors act as SigSent's end effector and source that powers the rotation of the wheels moving SigSent forwardly while in driving mode efficiently. Motor selection is important to both minimize weight and maximize speed.

4.3.10.1 Motors Under Consideration

- AX-4114C 330KV
- 4114-320KV Turnigy Multistar
- Turnigy Aerodrive SK3 - 4250-410KV
- Turnigy Aerodrive SK3 - 4250-350KV
- Quanum MT Series 4012 400KV

4.3.10.2 Specifications

The specifications below encompass the necessary items we value in the performance of the SigSent vehicle. Each of the motor units that were under consideration have had their details listed out such that the specifications below could be easily compared between each one.

4.3.10.2.1 Price

Price is a self-explanatory constraint, as the price of the motor increases this relates linearly with the team's will to implement it due to our limited sponsored budget.

4.3.10.2.2 Max Voltage

The maximum voltage the motors is an important consideration to take note of as it will impact both the amount of power the motors can output, the resultant RPM of the wheel, and the choice of battery and ESC. The choice was made to have a motor voltage to be compatible with a 4S battery (12-16.8) as that was considered a viable compromise between power efficiency, the RPM needed to spin the wheels (as a function of kV) without needed a gear ratio.

4.3.10.2.3 KV

KV is a measurement of torque that relates to the number of windings within the number, generally speaking the lower the KV the higher the torque, the lower the RPM the motor can produce. To reduce simplicity SigSent is intended to have the wheel directly attached to the motor in a 1:1 gearing ratio, so a motor that has a low KV (and thus high torque) is necessary. KV also relates to RPM per volt under no load. Since SigSent is intended to be able to run at 15mph in wheeled mode, we set a very liberal safety factor of finding the RPM at 25 MPH to account for when the vehicle is under load, with our wheels at a 2" diameter the motor would need to spin at roughly 4000 RPM, with the motor running at worst case 12V and best case 16.8V, we would need a KV rating of around between 300-400.

4.3.10.2.4 Max Current

The maximum current that the motor can handle is important to consider to ensure that the motor will not stall when attempting to move the vehicle, and for calculating the current rating required for the ESC, and the C rating for the battery along with general infrastructure requirements such as wire gauge to supply the power and trace widths through a PCB.

4.3.10.2.5 Weight

Since the motors are at the ends of the legs, the motor weight must be a minimum to reduce the amount of torque required for the servos to push and pull while in walking mode or in suspension during driving.

4.3.10.3 Scores

To determine the best motor to be used for SigSent, the max current of each one positively increased a motor's score while the cost and weight negatively affected its score.

4.3.10.3.1 Formula

In order to quantifiably determine the relevance of one motor over another a simple formula was devised after analyzing the available specifications found in the documentation for each of the motors. Since kV and voltage must be those values, they more filter out incompatible motors rather than effect the score of any motor over another.

Equation 5: Score for Motors

$$Score = \frac{Max\ Current}{Cost * Weight}$$

4.3.10.3.1.1 Comparison and Score Results

Table 9: Specification Comparison of Motors

Description	Weight	Kv	Max current	Price	Score
AX-4114C 330KV	180	330	28	16.6	9.37
4114-320KV Turnigy Multistar	217	320	30	25.8	5.37
Turnigy Aerodrive SK3 - 4250-410KV	415	410	55	36.8	3.6
Turnigy Aerodrive SK3 - 4250-350KV	423	350	53	36.9	3.39
Quanum MT Series 4012 400KV	266	400	16	20	3.01

4.3.10.3.2 Selection Rationale

Based off of the overall score and the low mass of the motor and low cost, we have chosen to use the AX-4114C 330KV. This motor will give us the necessary torque required to move the vehicle at the speeds we need it to with some additional overhead while being very cost effective.

4.3.11 Electronic Speed Controller (ESC)

SigSent utilizes ESCs to drive its motorized wheels at precisely controlled speeds. Differential drive of the wheels enables steering of the unit and is accomplished through setting the various wheels to rotate at dissimilar speeds.

4.3.11.1 ESC Requirements

- Each speed controller needed to be able to supply a constant output of 28 amps to enable the full power range of the motor it's driving.
- Needs to be able to handle a 4s lipoly voltage input.

4.3.11.2 ESCs Under Consideration

- Turnigy MultiStar 32bit 30A Race Spec ESC 2~4S Naked
- Turnigy K-Force 30A Brushless ESC
- Hobby King 30A ESC 3A UBEC

Table 10: Comparison of ESCs Under Consideration

Part #	MultiStar	K-Force	Hobby King
# of Cells	2 – 4	2 – 6	2 – 4
Size	28 x 14 x 5 mm	59 x 24 x 7mm	54 x 26 x 11mm
Weight	9 g	38 g	32 g

4.3.11.3 ESC Selection

The Turnigy MultiStar 30A Race Spec ESC iwas the most appropriate choice for SigSent's speed controllers due to its small size and low weight. The size and weight advantage stems from the ESC's exclusion of a battery elimination circuit, which is unnecessary with a 5V power supply already present on SigSent. Additionally, the ESC can be purchased without any connectors installed, allowing easier adaptation for use in SigSent.

4.3.12 Fuel Gauge

The fuel gauge is used to determine the remaining battery charge available for the robot so that the user operating the unit will know its limits on distance to travel and available remaining surveillance times.

4.3.12.1 Fuel Gauge Requirements

- Communicate with the microcomputer via I2C.
- Accurately measure a 4s LiPoly battery.

4.3.12.2 Gauges under consideration

- TI BQ34Z100-G1
- LT LTC2943
- Maxim Integrated MAX17205

Table 11: Comparison of Gauges Under Consideration

Part #	BQ34Z100-G1	LTC2943	MAX17205
Communication Protocol	I^{2}C, HDQ	I^{2}C, SMBus	I^{2}C
Quiescent Current	145 μA	80 μA	25 μA
Voltage Input	3 – 65 V	3.6 – 20 V	4.2 – 20 V
Packaging	14-Pin TSSOP	8-lead DFN	14 TDFN-EP or 15 WLP

4.3.12.3 Fuel Gauge Selection

The Maxim Integrated MAX17205 was initially selected as the most appropriate fuel gauge for SigSent due to its low quiescent current and ease of use. It boasts the lowest supply current while active, and unlike the BQ34Z100-G1, requires no additional voltage regulators.

During the design of the fuel gauge PCB, the LTC2943 was instead incorporated due to prior experience and comfort which enabled a quicker design time.

4.3.13 Battery

SigSent's mobile operation is enabled by a high-capacity battery. It needs to have a large enough capacity to meet the operating time requirements specified. Below are different battery types, estimated loads from our whole system, and specific batteries we considered for use. One battery was chosen after objectively comparing them all together.

4.3.13.1 Battery Chemistries Under Consideration

- Nickel-Metal Hydride (NiMH)
- Lithium-Ion
- Lithium Polymer (Lipo)

Table 12: Comparison of Battery Chemistries [46]

Chemistry	NiMH	Lithium Ion	Lipo
Nominal Cell Voltage	1.25 V	3.6 V	3.6 V
Gravimetric Energy Density	60 – 120	110 – 160	100-130
Discharge Rate	0.5 C	1 C	1 C
Cycle Life	300 – 500	500 – 1000	300 – 500
Charging Rate	0.5 C	0.5 C	0.5 C

Lithium based batteries are most appropriate for SigSent due to their significantly higher energy density.

4.3.13.2 Estimated Electrical Loads

Table 13: Estimated Electrical Loads

Part	Typical Current Draw (A)	Max Current Draw	Typical Operating Voltage (V)	Typical Power Draw (W)	Qty	Duty Cycle (% of hour)	Typical Hourly Energy (WH)
Servo	0.5	1.5	12	6	18	25	27
Motor	14	28	14.8	207.2	4	25	207.2
Microcomputer	1.5	2.5	5	12.5	1	100	12.5
Microcontroller	0.2	0.5	5	1	1	100	1
IMU	0.004		3.3	0.02	1	100	0.02
GPS	0.068		3.3	0.2	1	100	0.2
Speaker/Amplifier	1	1.6	5	5	1	5	0.25
Lidar	0.7	1	12	8.4	1	100	10
Light Source	0.9		14.8	10	1	100	10

4.3.13.3 Battery Requirements

- Must be able to supply 100 A of current continuously to support the highest power mode.
- Must be able to fit inside SigSent's abdomen.
- Must have a relatively high energy density and specific energy.
- Must be a single pack battery solution to maximize energy density and minimize power system complexity.

4.3.13.4 *Batteries Under Consideration*

- MultiStar 912700006-0
- L&E Battery LND3S956
- Turnigy 9171001348-0

Table 14: Battery Comparison

Part #	**912700006-0**	**LND3S956**	**9171001348-0**
Capacity	10000 mAh	9500 mAh	6400 mAh
Nominal Voltage	14.8 V	11.1 V	11.1 V
Dimensions	160 x 65 x 36mm	155.0 x 44.5 x 41.0 mm	135 x 45 x 42 mm
Weight	804g	588g	485g
Discharge Rate	10 C / 100 A	65 C / 617.5 A	30 C / 192 A
Energy Capacity	148 W	105 W	71 W
Charging Rate	1 C	1 C	2 C

4.3.13.5 *Battery Selection*

The MultiStar High Capacity 10000mAh 4S Lipo pack (912700006-0) is the most appropriate battery for SigSent due to its impressive specific energy, energy density, and relatively low cost. Although its discharge rate is significantly lower than that of the LND3S956 and 9171001348-0, the MultiStar can supply sufficient current at an ample margin, and at a higher voltage.

4.3.14 Audio Amplifier

The audio amplifier was intended to boost the signal of the microcomputer's audio output and directly power the unit's speakers.

4.3.14.1 *Amplifier Requirements*

- Needs to integrate with SigSent's microcontroller. If discrete audio outputs are not included in the microcontroller, functionality can be achieved with a USB Audio Adapter: https://www.adafruit.com/product/1475
- Needs to be relatively power-efficient, using a Class-D amplifier.
- Needs to provide 8 watts output to a 4 Ohm impedance load.
- Needs to output sound with minimal harmonic distortion to ensure comprehension of vocal commands provided through the speaker.

4.3.14.2 *Amplifiers Under Consideration*

- TI TAS5411-Q1
- Rohm Semiconductor BD28412MUV
- Maxim Integrated MAX9736B

Table 15: Comparison of Amplifiers Under Consideration

Part #	TAS5411-Q1	BD28412MUV	MAX9736B
Power Output	8 W @ 4 Ω	2x 8 W @ 8 Ω	12 W @ 4 Ω

Voltage Supply	4.5 – 18 V	4.5 – 13 V	8 – 28 V
Quiescent Current	16 mA	32	45 mA
I/O	I^2C	Boolean	Boolean

4.3.14.3 Amplifier Selection

The TAS5411-Q1 was determined to be the most appropriate amplifier for SigSent with its minimal quiescent current draw and design-friendly I^2C control architecture.

During the design stages of SigSent, an off the shelf USB speaker was chosen for use instead of a component speaker. This eliminated the need for an audio amplifier.

4.3.15 Speaker

A speaker is used to relay commands from SigSent's operator to individuals that the unit encounters. Additionally, the speaker can be used to play a siren to deter trespassers or animals.

4.3.15.1 Speaker Requirements

- Suitable for environmental exposure. Water resistant, 0º C - 50º C in high humidity.
- Responsive across the voiceband (300 hz - 3.4 khz).
- Relatively small so that it may be mounted on top of SigSent and directed in its field of view.
- High efficiency so that the need for power amplification is minimized.

4.3.15.2 Speakers Under Consideration

- PUI Inc.'s AS07104PO-WR-R
- PUI Inc.'s AS07708PS-2-WR-R
- PUI Inc.'s AS06608PS-WR-R

Table 16 Comparison of Speakers Under Consideration

Part #	**AS07104PO-WR-R**	**AS07708PS-2-WR-R**	**AS06608PS-WR-R**
Frequency Response	100Hz~20kHz	250Hz~10kHz	230Hz~12kHz
Dimensions	2.795" L x 1.614" W x 0.984" H	3.032" L x 3.032" W x 1.063" H	2.610" L x 2.610" W x 1.142" H
Sensitivity	86.00 dBa @ 1W / 1 m ~= 92.00 dBa @ 1W / 0.5m	90.00 dBa @ 1W / 0.5 m	95.00 dBa @ 1W / 0.5 m
Rated Power	3 W @ 4 Ohm	4 W @ 8 Ohm	4 W @ 8 Ohm
Environmental Envelope	-20º C ~ 60º C, Water resistant	-40º C ~ 85º C, Water resistant	-20º C ~ 50º C, Water resistant

4.3.15.3 Speaker Selection

The AS07708PS-2-WR-R is the most appropriate speaker for SigSent due to its sufficient frequency response, suitable form factor, sensitivity, and rated power. It's considerable environmental testing provides the most confidence for its long-term reliability during operation outdoors.

Equation 6: Estimated SPL at 10 meters from the unit.

$$SPL_{@10m} = 90 + 10(\log 4) - 20\left(\log\frac{0.5}{10}\right) \approx 70dB$$

With an appropriately matched amplifier powering the speaker at its rated wattage, the speaker should output sound at 70 dB, well above the required 60 dB.

For the prototype build completed, we swapped out the desired AS07708PS-2-WR-R with an actual USB speaker to reduce the number of PCBs we needed to create and to minimize potential error. The USHONK USB Mini Speaker was used in its place as it had seamless integration into the Raspberry Pi's workflow in Ubuntu MATE.

4.3.16 Microphone

A microphone is used for communicating vocal responses through the SigSent robot. The human operator can speak into their microphone at their base station which will then be projected from the robot's attached speaker. Audio can also be transmitted from the robot to the base station by a microphone mounted on the vehicle so that the operator can communicate with agents nearby the robot.

4.3.16.1 Microphone Requirements

- Needs to integrate with SigSent's microcomputer.
- Needs to minimize background noise in order to hear a subject's speech outdoors.
- Needs to survive outdoor environmental use.
- Needs to be relatively small.

4.3.16.2 Microphones Under Consideration

- MicW iShotgun
- PlayStation Eye
- Rode VideoMic Me

Table 17 Microphone Comparison

Part #	**iShotgun**	**PlayStation Eye**	**VideoMic Me**
Frequency Response	100Hz~18kHz	?	100 Hz – 20 kHz
Dimensions	136 l x 8 d mm	80 x 56 x 65	38 x 21 x 80 mm
Sensitivity	-42 dB	?	-33 dB
Cost	$200	$7	$60

4.3.16.3 Microphone Selection

The PlayStation Eye's incorporated microphone array is the most appropriate device for SigSent based on testimony of users utilizing the device for similar project goals. The device is appropriately sized, easily integrated through its USB connection, and designed for speech recognition.

4.3.17 Lighting System

The lighting system is used to provide adequate lighting for the SigSent's vision-based operations at night. It was mounted on the robot to light up the area around the robot. The computer vision modules rely on visibility, and the human operating the unit will need enough light to get a proper image from the robot to view the area it is surveilling.

4.3.17.1 Requirements

- Suitable for exposure to the expected area of operation, meaning at least IP67 certified with an operating temperatures of 0º C - 50º C.
- DC powered.
- Efficient.
- Light weight.
- Appreciably bright to light the area in front of SigSent for acceptable color vision from the camera

4.3.17.2 Light Sources Under Consideration

- superbrightleds.com AUX-6W-RE120
- superbrightleds.com AUX-20W-Dx
- superbrightleds.com WL-17W-RE60

Table 18 Comparison of Light Sources Under Consideration

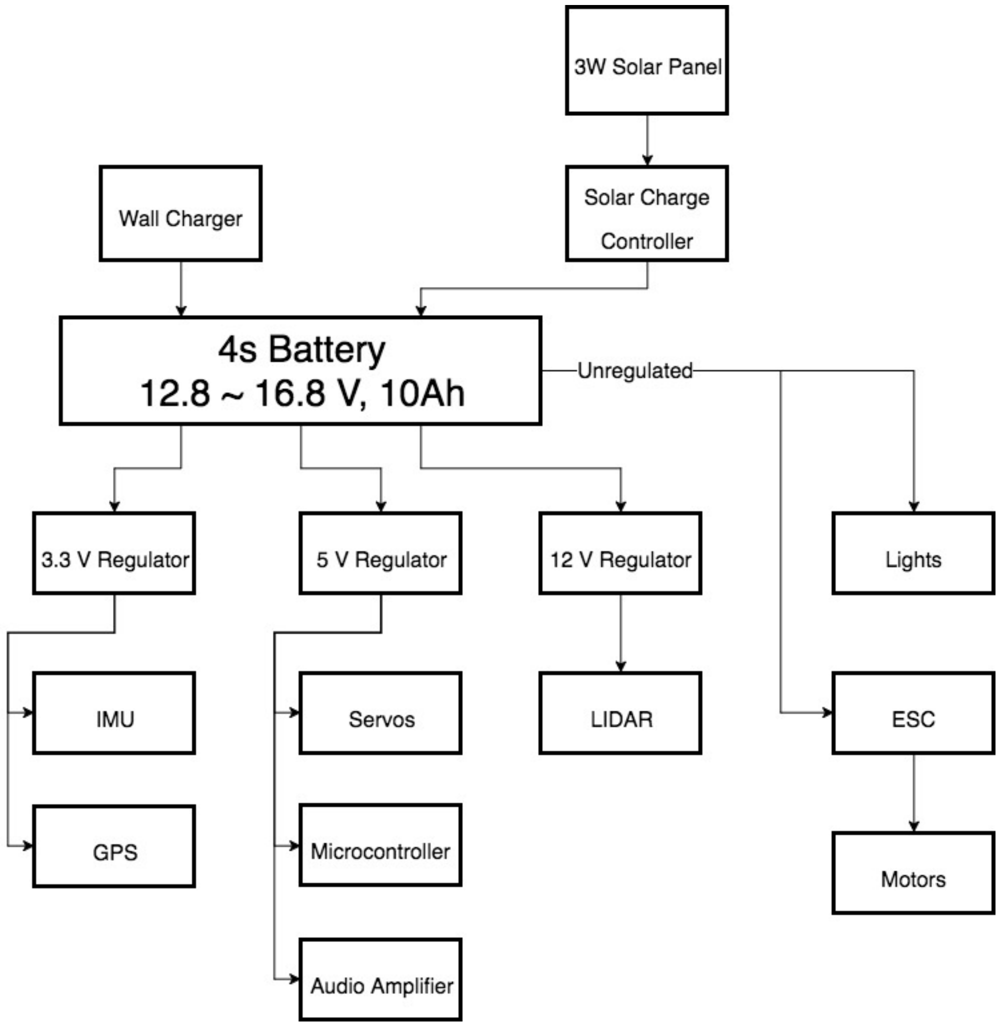
Part #	**AUX-6W-RE120**	**AUX-20W-Dx**	**WL-17W-RE60**
Dimensions	3.67" L x 0.92" W x 2.05" H	3.95" L x 2.1" W x 2.39" H	6.3" L x 2.2" W x 1.78" H
Weight	0.27 kg	0.43 kg	0.41 kg
Brightness (L)	725	1800	1300
Beam Angle	120º	60º	60º
Power (W)	6	20	13
Environmental Envelope	IP67	IP68	IP67, -40º C ~ 56º C

4.3.17.3 Selection

For a light source mounted on the front facing surface of SigSent's abdomen, the WL-17W-RE60 light bar is most appropriate, with relatively good power efficiency, an appropriate beam angle which isn't dispersing the light too much and sufficient environmental protection.

4.3.18 Power System

Figure 21: Power Flow Diagram



4.3.18.1 Solar Panels

A solar panel was explored as a potential candidate to provide an extended period of life for the SigSent robot while it undergoes outside surveillance during the day. The solar panel would need to take weight and power into account to provide any significant benefit to the SigSent's operation.

4.3.18.1.1 Potential Options

- DFRobot FIT0333
- Seeed Technology Co. 3W Solar Panel 138*160

4.3.18.1.2 Requirements

- Surface area smaller than the robot's abdomen, where it'll be mounted. If As many units as possible will be arranged to fit on the surface.
- Suitable for area of operations temperature range.
- Relatively efficient.

4.3.18.1.3 Comparison Table

Table 19 Solar Panel Comparison

Part #	**FIT0333**	**3W Solar Panel 138*160**
Environmental Envelope	-40º C ~ 80º C, “Performance: corrosion, moisture”	"Robust sealing for out door applications”
Dimensions	6.500" L x 1.496" W x 0.020" H	5.43” L x 6.3” W x 0.060” H
Power Density	60 W / m^2	135 W / m^2

4.3.18.1.4 Selection

The 3W Solar Panel would be the more suitable option for our purposes because of its significantly higher power density, which will enable it to provide more energy to SigSent throughout the day.

In order to reduce design complexity, cost, and weight, the solar charging system was not implemented.

4.3.18.2 Solar Charger

If a solar panel were included in SigSent’s final design, a solar charger would have been necessary to handle the charge produced by the solar panel, and distribute the current produced between the battery and spontaneous load.

4.3.18.2.1 Requirements

- Needs to be able to charge a 4 cell LiPo battery.
- Needs to be able to handle at least 3 W of solar power.

4.3.18.3 3.3 V Regulator

- Needs to provide at least 100 mA of current.
- Needs to have mild ripple.
- Needs to accept at least 16.8 V input.

4.3.18.4 5 V Regulator

- Needs to provide at least 30 A of current.
- Needs to have at most mild ripple.
- Needs to accept at least 16.8 V input.
- Needs to be highly-efficient to minimize heat dissipation

4.3.18.5 12 V Regulator

- Needs to provide at least 20 A of current.
- Needs to have at most mild ripple.
- Needs to accept at least 16.8 V input.

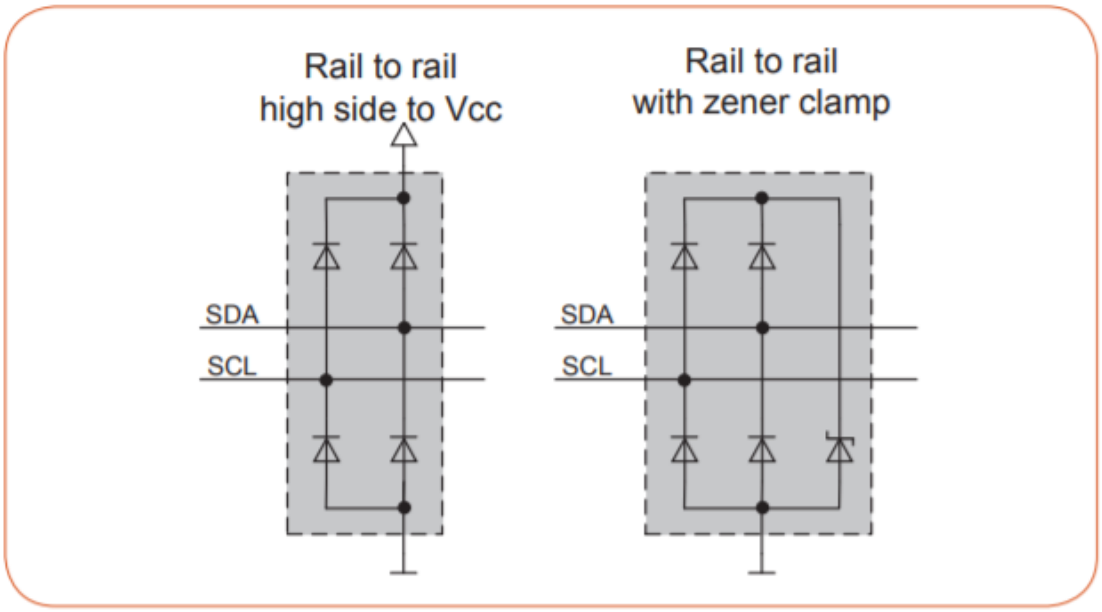
4.3.18.6 Light Switching Transistor

- Needs to be able to switch at least 2 A at 16.8 V or more.

4.3.19 Signals Protection System

A signals protection system is wanted, especially on any signals going into or coming out of the Raspberry Pi since the GPIO pins are very sensitive to transients and overvoltage's. Since this platform is a prototype where lots of manual interactions will be going on, chance of an ESD event or a miss-wiring are high which means that the protection PCB will be significantly useful in safeguarding our project.

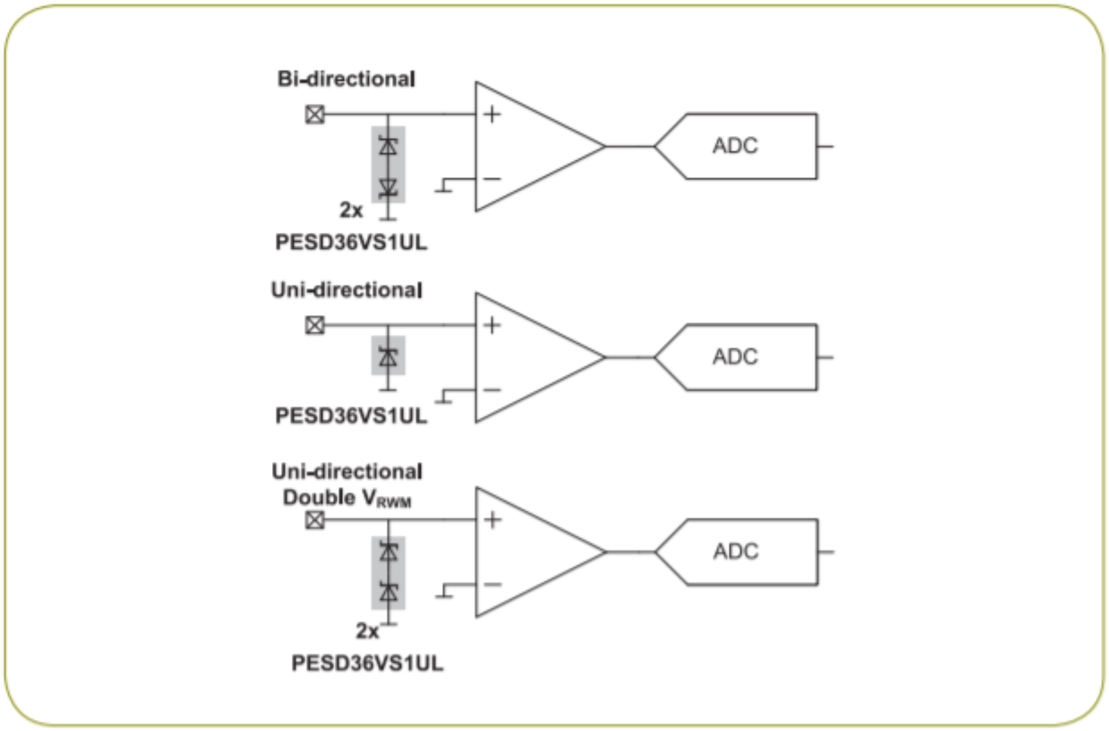
To protect SigSent from ESD events and transients a very common and well-respected system is to use Zener clamp diodes in a way specified by NXP for i2c:



I²C protection two rail-to-rail configurations that minimize capacitive line loading

Figure 22: Example of i2c ESD Protection [47]

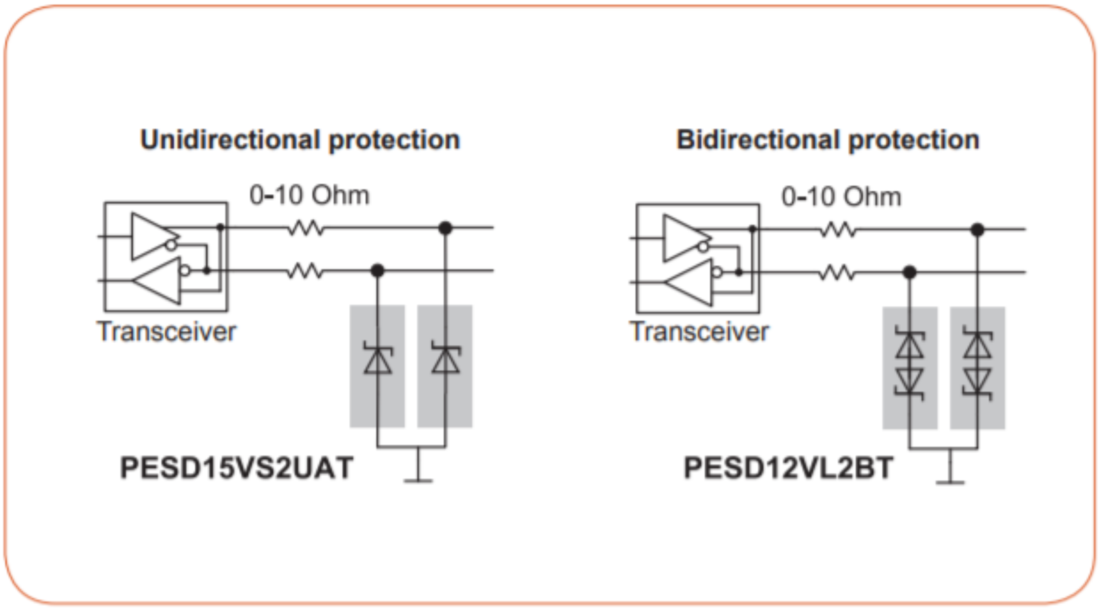
To protect general GPIO Ports from ESD and overvoltage events is to use normal Zener diodes in a pair for bi-directional communication:



Analog I/O protection using the PESD36VS1UL with single or multiple devices

Figure 23: Example of GPIO ESD Protection [47]

To Protect our TTL Serial buses, NXP Recommends similar Zener Clamping Diodes explained in the below diagram:



Standard serial port, RS-485, or RS-232 protection can be either unidirectional or bidirectional

Figure 24: Example of TTL Serial ESD Protection [47]

And Finally to protect our USB Lines from ESD Events, NXP Recommends the following array of Zenner clamping diodes.

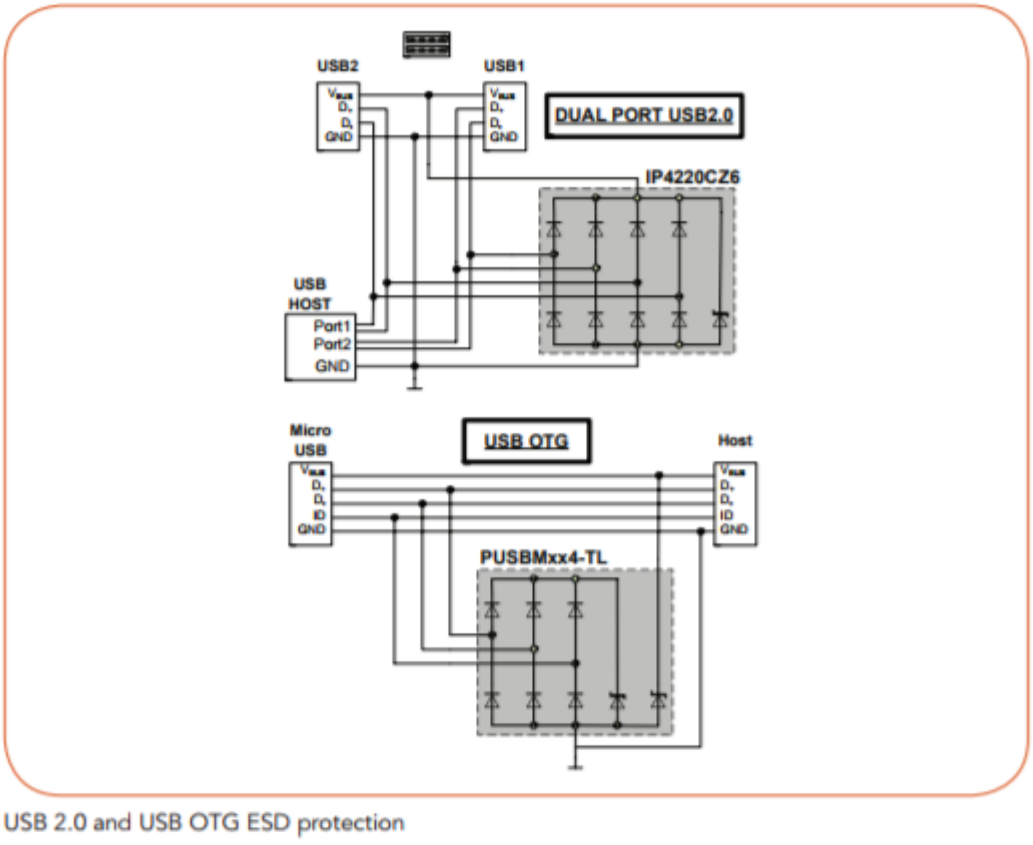


Figure 25: Example of USB ESD Protection [47]

Beyond ESD Protection it is important to protect the power lines of our devices to prevent things such as miss-wiring the boards or plugging in a power source in the wrong direction. A common a method to efficiently prevent reverse polarity events is to use a PMOS connection like described in the figure below created by Texas Instruments:

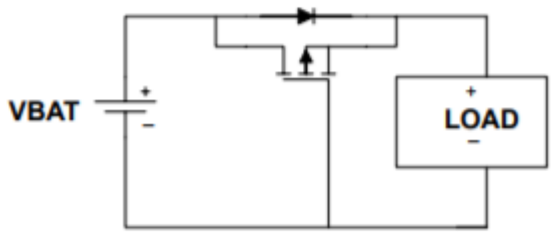


Figure 26: PMOS FET in Power Path for Reverse Circuit Protection [48]

To Prevent an overcurrent situation, fuses will strategically and minimally be placed on all power buses to prevent an overcurrent event either damaging the battery, destroying sub systems, or causing an electrical fire.

4.3.20 Base Station

The base station computer will encompass the hardware necessary to communicate with the SigSent unit. Any item necessary in the remote operation of the robot by the user will be listed below with its relevant requirements and final selections.

4.3.20.1 Laptop

The laptop is where the GUI program will be run that communicates with the SigSent robot. The program is not computationally expensive. It allows for a connection to the robot over Wi-Fi, a video feed from the robot, remote operation through a joystick, and debugging over the air. The actual computation and calculations performed for the operation of the robot's autonomous intelligent systems and control systems are all being done on the actual robot's microcomputer, not the base station laptop.

4.3.20.1.1 Requirements

- Bright enough screen to be easily visible outdoors so that the operator can stay close to SigSent if necessary.
- Man portable so that a user can easily bring it with them to the area of operation.
- Can reliably run ROS Kinetic Kame and rqt, the software used to communicate instructions to SigSent.
- Can connect to the same Wi-Fi network as SigSent.
- Enough battery life on one charge to last as long as SigSent does on one charge, so that SigSent isn't left stranded because its controller is unable to communicate instructions to it.
- Contains a USB port compatible with the joystick.
- Outputs headset audio and receives microphone input.

4.3.20.1.2 Selection

A Lenovo ThinkPad laptop on hand will be utilized as the base station. It meets the requirements listed above and will reduce the cost of the unit for our development.

4.3.20.2 Headset

A headset is necessary for listening and communicating through the SigSent robot. The base station allows for the user to listen to audio being received by the robot's speaker. The microphone on the headset will be used as an audio input into the base station GUI program that will be outputted from the SigSent speakers so that vocal responses can be projected from the robot.

4.3.20.2.1 Requirements

- Ergonomic and comfortable to wear outdoors for an extended period of time.
- Contains a microphone so that the operator may communicate vocal commands over SigSent's speaker.

4.3.20.2.2 Selection

Basic iPhone headphones were used on a Macbook Pro running Ubuntu 16.04 in a VM to communicate with SigSent over ROS audio_common packages that were responsible for relaying audio between the two machines.

4.3.20.3 Joystick

A joystick is used to facilitate the remote operation procedures through the base station GUI program. The joystick brings an intuitive method of moving the robot by the human operator. By simply tilting the joystick, the robot will move accordingly, as controlled by its control system and AI intelligent system.

4.3.20.3.1 Requirements

- Ergonomic so the robot can operate by the same user daily without strain from repeated use.
- Intuitive operation and translation of control inputs into XY motion so that operators can begin using SigSent with minimal training.
- Interfaces with the base station via USB cable.

4.3.20.3.2 Selection

A joystick on hand, the Logitech Extreme 3D Pro Joystick will be used. It contains more than enough usability for our project with multiple axes of rotation (pitch, yaw, roll).

4.3.20.4 Router

In order to communicate with the SigSent unit at distances greater than possible with an ad-hoc network from the basestation, a wireless router can be connected to the base station laptop. A router on hand, a Linksys WRT54GC, was initially utilized to minimize project cost. To maximize range and throughput of the router, it should utilize its fastest available protocol, 802.11g, and a channel on the 2.4 GHz band. Power for the router will be provided through a USB cable from the base station's laptop. With this configuration, bandwidth should be sufficient to support streaming of video, audio, and diagnostics, estimated to be approximately 5 Mb/s [7]. The router will utilize WPA2 Personal (AES) security to ensure confidentiality and integrity of communication between the base station laptop and SigSent unit.

The router was swapped out with the Tenda AC1200 near the end of the SigSent development period. The Tenda featured longer range and the ability to easily bridge its connection from the UCF Guest network. This allowed SigSent to have network access anywhere on campus that the UCF Guest network was available. One of the machines on the network simply had to authenticate the Guest settings and each machine connected to the bridged network would have a connection from there on out. This bridging capability was not initially accounted for as we intended on wiring the router to a nearby ethernet port, but it came in handy upon discovering the functionality a few days prior to finishing the project.

4.3.21 SigSent's Sensors and Non-Mechanical Parts

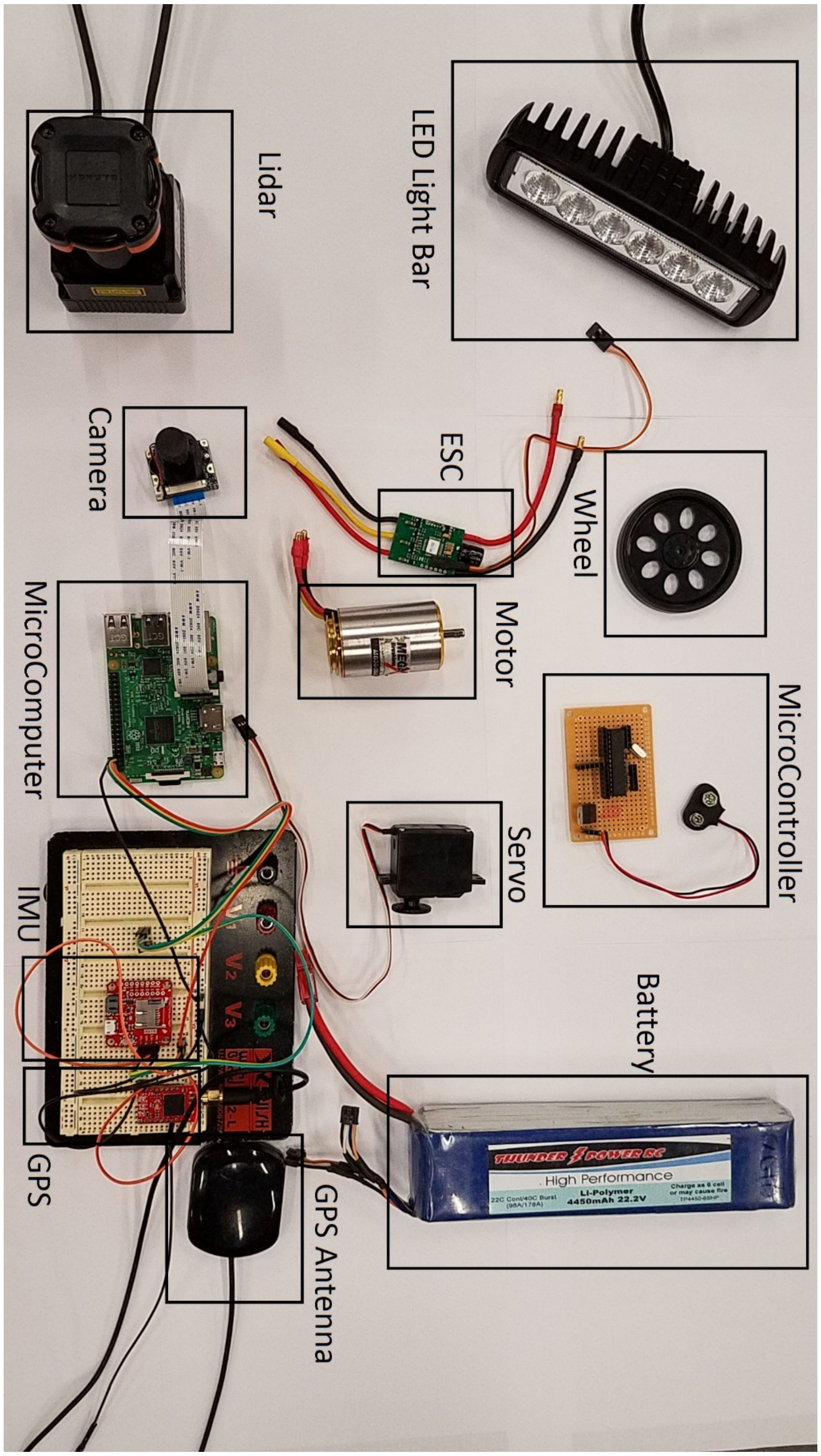


Figure 27: List of Parts with Annotations

5 DESIGN

5.1 DESIGN SUMMARY

After researching the potential parts for consideration in section 4, each sensor and individual component was objectively scored/compared such that the most fitting option under our design constraints was chosen. These modules will be used in the design of SigSent. Below are high level overviews on the hardware and software in block diagrams that denote each module and its division of labor among each team member according to their individual specifications. The hardware schematics are designed and discussed below in their respective PCB sections. The software design decisions are discussed and followed by UML class diagrams and use case diagrams to scaffold out the individual, complex modules for SigSent's code.

5.2 HARDWARE DESIGN

Because of the immense amount of components involved in SigSent's operation, the hardware design includes schematics on the integration of each sensor and its supporting hardware in the robot. The design of each schematic follows our design constraints and standards previously mentioned, and the schematics include each relevant component that was selected in our hardware research.

5.2.1 High Level Hardware Block Diagram

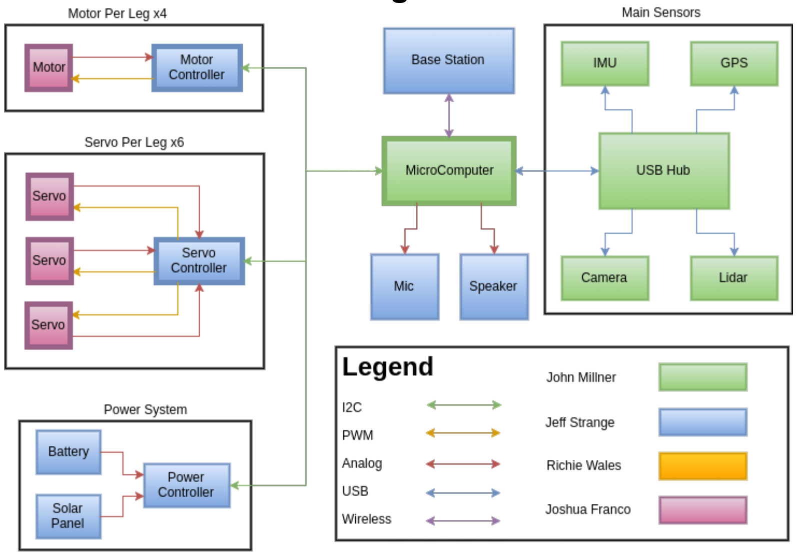


Figure 28: High Level Hardware Block Diagram

5.2.2 Hardware Design Overview

Hardware Design is split into five main section: wheels, legs, power source, main sensors, and base station.

The Wheels and Legs sections closely mirror each other, in the motor section, we have four high torque brushless motors each connected to an Electronic Speed Controller (ESC) which inputs a PWM and outputs the proper phasing required to drive the DC brushless motor, and with the servo we input a PWM signal and the servo motor then rotates to a predefined angle attached to that PWM value. There is also a current sensor monitoring the amount of current going to the motor or servo motor to detect whether or not the motor or servo motor is completing its desired task. Both the PWM signal and the current sensor output are converted into an I2C Signal which passes through protection circuitry and goes to the microcomputer which will input and output data to and from the motors and servo motors.

In the power source section, we have the battery which sends data through a Fuel Gauge which then outputs the status of the battery (voltage, current, coulombs consumed) through the I2C interface, passing through I2C protection circuitry and then to the microcomputer which uses the battery data to send alerts or modify its path to be more energy efficient focused.

For the main sensors section, we have a USB hub connected to the microcomputer which connects most of the sensors together such as the camera, lidar, the wireless network, IMU, and GPS.

The base station section is separate from the robot and acts as a control point for the robot's supervisor to access, control, or receive alerts from the robot - the base station stays in near continuous communication with the robot through a wireless connection.

5.2.3 Pi Hat PCB

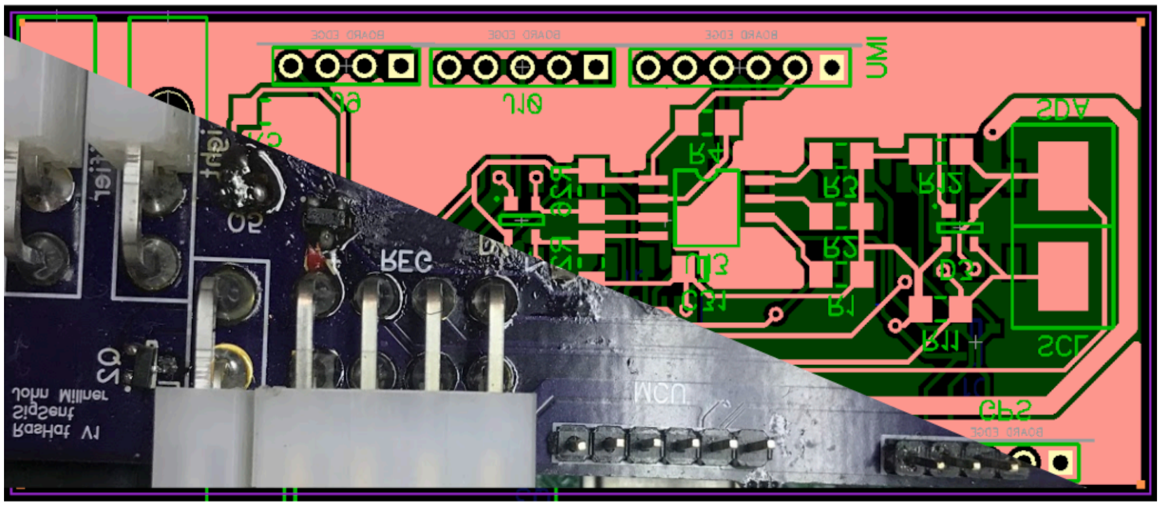


Figure 29 Pi Hat Diptrace layout and physical board

The Pi Hat as shown in Figure 29 is a custom circuit board that directly connects onto the Raspberry Pi's main stack connector. The purpose of the board is to create an easy to use "unifying" board where all sensors and devices are connected that need to interact with the Raspberry Pi. Inside the board contains an EEPROM that identifies the board connected to the Pi as an official PiHat [49] as defined in the Raspberry Pi Documentation. There is reverse voltage protection on all input power lines to ensure that devices downstream of the hat are protected against a mis-pinned cable or bad power supply. There are low capacitance ESD protections on all signal lines going to the pi to prevent any ESD events, which could be common on a prototype project such as this. The device also has several test points designed to work well with oscilloscope probes – specifically on the I2C nets to help diagnose any communication troubles. The decision was also made to expose an extra 8 GPIO pins from the Pi for future proofing or backups in the event the scope or understanding of the project changed once the board had been made. Partly because of the design of this board, there we're no sensors or other devices lost from ESD events, and even after a slight change which required an extra GPIO pin, this board was still able to be easily and effectively used.

5.2.4 MCU PCB

The MCU PCB allows for direct communication to the transportation layer from the sensor layer. This board features an ATmega2560, headers to connect to the Raspberry PI over SPI, for ICSP to program the MCU via another Arduino Mega, and to connect to the motor ESCs. The PCB features all of the additional hardware needed for the ATmega chip, like the 16MHz oscillator crystal. A reset button is also implemented for ease of use.

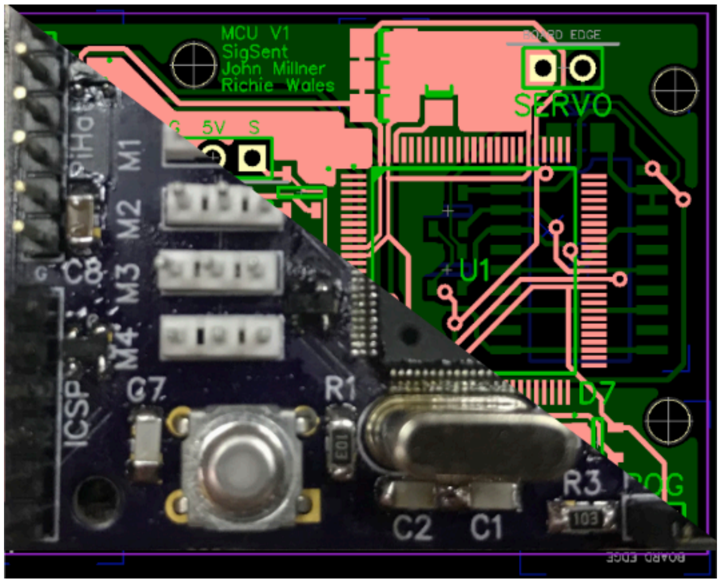


Figure 30 MCU Diptrace layout and physical board

5.2.5 Servo Regulator PCB

The Servo Regulator PCB shown below provides a stable 12V output for the 3 servomotors of a corresponding leg. Each of SigSent's six leg necessitated a separate regulator board. The boards also passed the serial command line from the microcontroller through to the servomotors of the leg. Input power is provided by the battery, and reverse voltage protection is provided by an N-channel MOSFET, the AO4411, and overcurrent protection is provided by a 7.5A fuse.

The regulator is a buck converter incorporating TI's LM25116 controller. Two MOSFETs, the n-channel IRF8714TRPBF and the n-channel CSD18542KTT were used to switch output current.

The regulator design was developed through the use of TI's WEBENCH software.

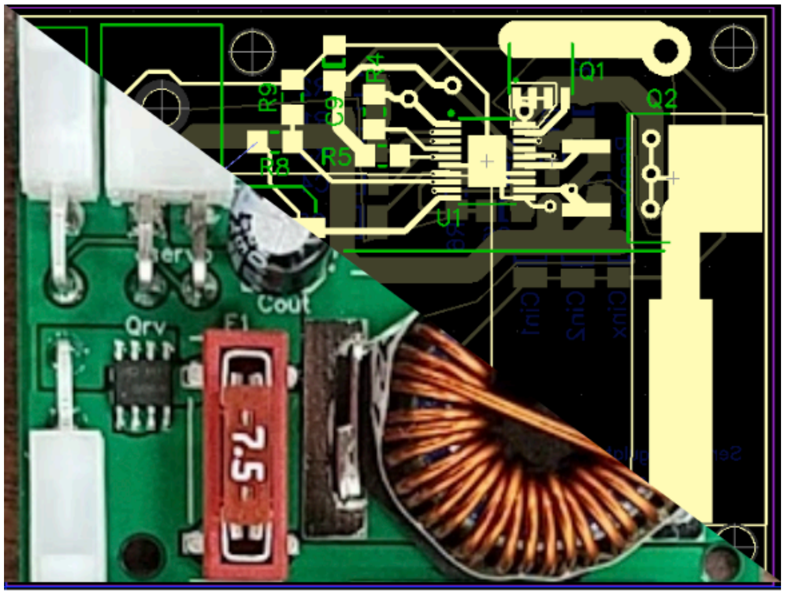


Figure 31 Servo regulator Diptrace layout and physical board

5.2.6 IMU Module PCB

The IMU module PCB covers the connection between the IMU and the microcomputer such that its values can be read by the intelligent systems software modules to aid in the terrain classification based on the vehicle's orientation and angular acceleration (movement due to rough terrain).

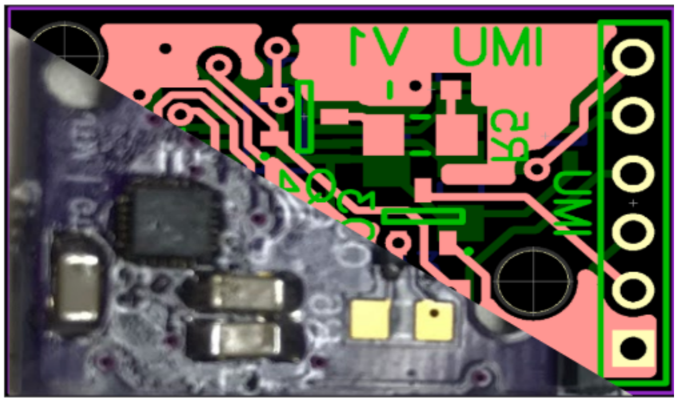


Figure 32 IMU Diptrace layout and physical board

5.2.7 Battery Fuel Gauge PCB

An LTC2943 fuel gauge IC is used to monitor SigSent’s battery, measuring its voltage, calculating its current draw, and estimating its remaining charge. Electrical measurements are made across a 300$\mu\Omega$ shunt resistor. Measurements are communicated to the Raspberry Pi through I^{2}C, which can be toggled by an included TCA9617B level-translator.

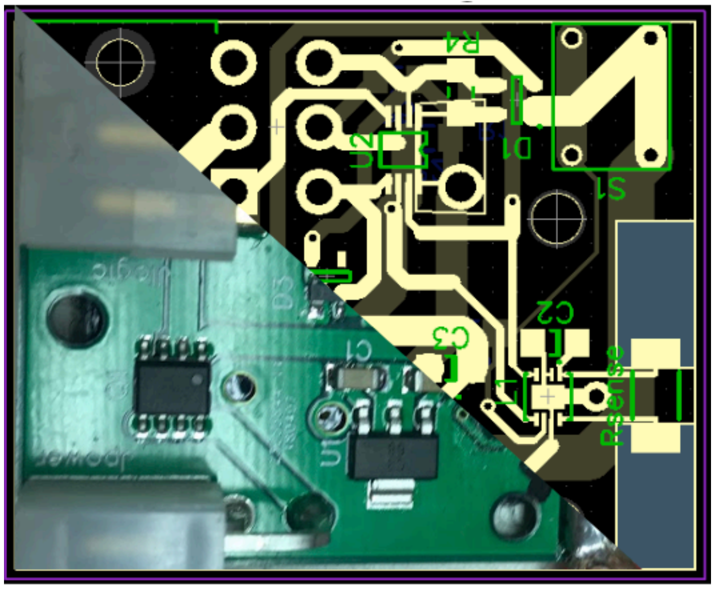


Figure 33 Fuel gauge Diptrace layout and physical board

5.3 Modular Layout

SigSent’s hardware layout was done modularly such that each layer would have its own role in operation and could be easily maintained. Additionally, the top sensor layer is easily removable, secured by Velcro, allowing the user to move the brains of the robot to any unit that they would like. The power distribution layer sits below the sensor layer and is responsible for connecting the regulator boards and the devices to the bus bars adjacent to the battery. The locomotion layer holds the hardware that moves SigSent. The MCU board sits here where it is then connected to the servo daisy chain as well as the four ESCs.

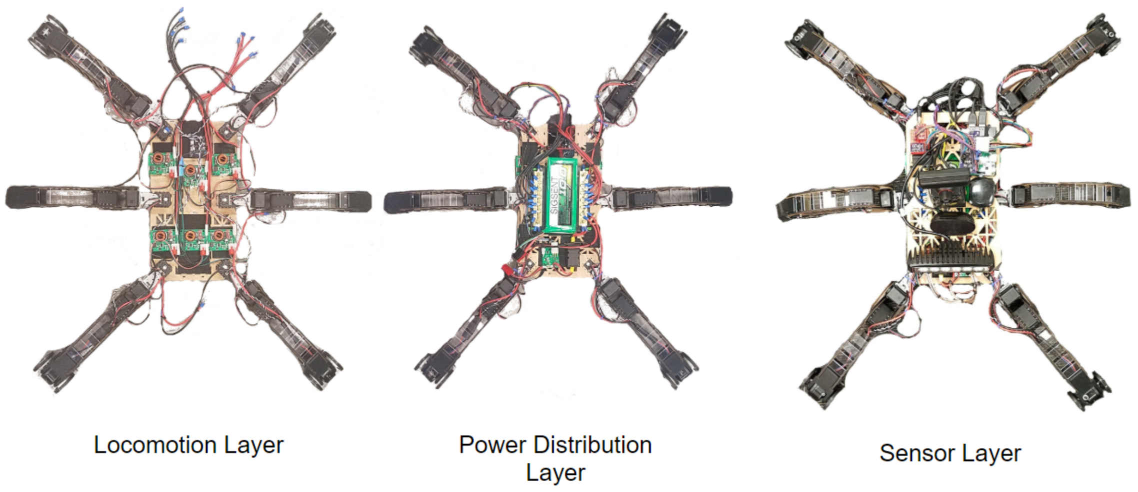


Figure 34 Modular design layout

5.4 Software Design

SigSent's software will be organized modularly for each discrete system. The high-level planning of the software is discussed with accompanying block diagrams, and UML for class diagrams and use case diagrams. The software is designed with the highest usability and maintainability in mind. Design principles and architectures are compared and selected based on their strengths for SigSent.

5.4.1 High Level Software Block Diagram

The software block diagram in Figure 35: High Level Software Block Diagram displays the modular design of the software. The division of labor is also visible by the colors denoted in each block.

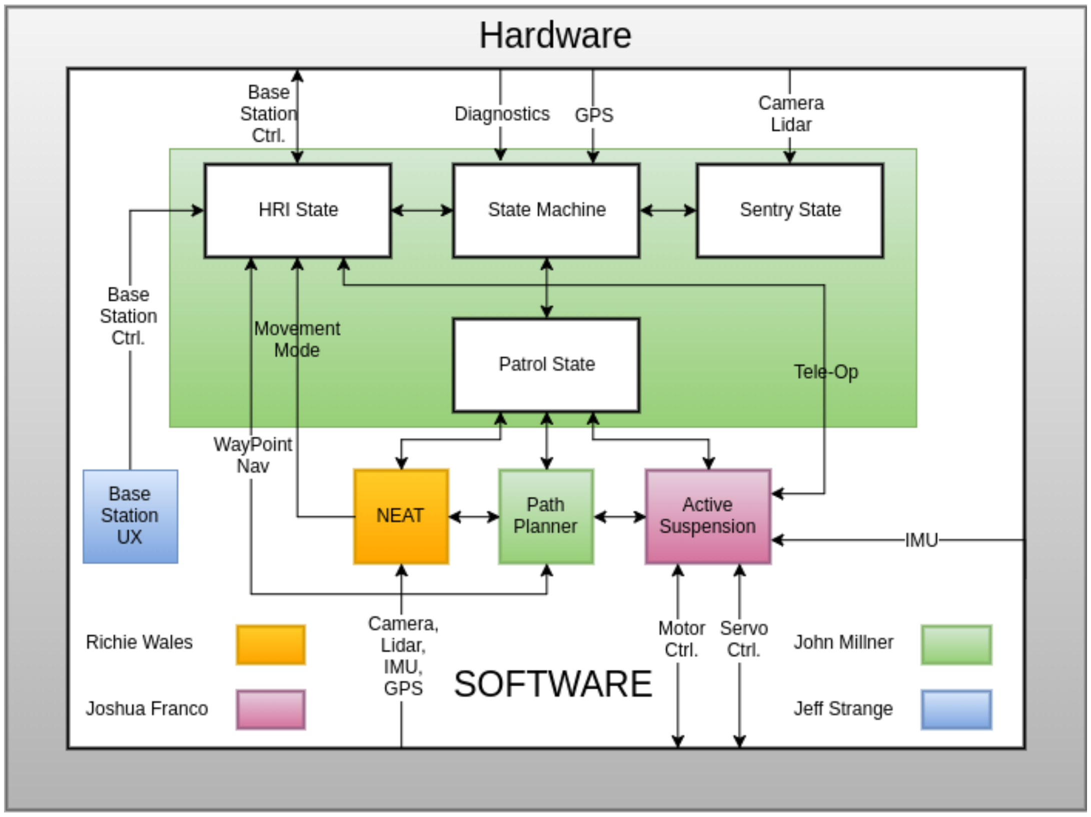


Figure 35: High Level Software Block Diagram

5.4.2 Software Design Overview

The software encompasses the code for the autonomous performances on the robot. The design processes for development are discussed below. Each development procedure was chosen for the most streamlined development process.

5.4.2.1 Design Methodologies

Our team used the Scrum framework for agile software development. It is used commonly in small teams of rapid development cycles. While developing, daily 15-minute meetings are conducted where each developer discusses their progress from the day prior as well as what tasks they plan to tackle on that current day. Every two weeks, a sprint planning meeting is conducted so that a new set of tasks can be allotted for the two-week development cycle. These sprints are exactly as they sound, a fast, brief time span, where each developer takes a task from the remaining list of those allotted and tries to complete them all before the end of the two-week period. A Scrum Master, who acts as the manager of sorts, keeps the group on track and focuses on improving the team's velocity over the development time. At the sprint planning meeting, a retrospective is performed on the previous two weeks where the Scrum Master will help guide the team in a discussion on the main events of the sprint. If the velocity of the team was low, the team must figure out why that occurred and what can be changed to improve that in the future. The sprint retrospective can then be demonstrated to the product owner or stakeholder if necessary, however for our time, that will not be needed. Although there will only even be a couple programmers working on a part of the project at once, having a design methodology outlined will keep the team on track and will also expose everyone to a real software engineering environment and show how useful agile methodologies are in the workplace [50].

5.4.2.2 Technology

The intelligent system technologies used in SigSent's software backbone will consist of the artificial intelligence software suite provided by NEAT's neuroevolution on an artificial neural network, as well as the terrain classification used to define the nearby landscape that the robot is navigating across. OpenCV will be used as well in the stack to provide detection of movement of intruders to alert the user monitoring the SigSent robot. ROS will be used to manage the control signals to and from the robot's sensors and motors. The High-Level Software Block Diagram in 0 demonstrates how each technology is associated with the other significant modules to our project. There are some extensions viewable in the block diagram that are not essential to the project at this time. The Path Planner module will be used for future work. It relies on the success of the control systems and NEAT AI modules and can only be integrated when they are fully functional.

5.4.2.3 Architecture

The software architecture follows a singleton design pattern for each intelligent system. The NEAT system is managed by a single overarching class that will do the evolutionary computation work as well as communicate with the ROS modules. In Singleton's, if a reference does not exist, it is created and returned to the user. If a single static reference already has been created, then a new one is not created; The single reference is returned instead. By following the singleton design pattern, we ensure that only one single reference to the ANN will be used everywhere in the code. Singletons are necessary when the programmer must enforce this idea. Design patterns like this can strictly followed by forcing it upon the code rather than on the developers and users. The machine learning system that handles the terrain classification will also feature a singleton for the same advantages.

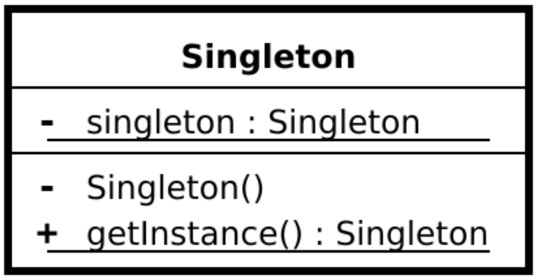


Figure 36 Singleton example. Public Domain, https://commons.wikimedia.org/w/index.php?curid=1484985

The publish-subscribe (also informally called pub-sub) model is used heavily as well as it pertains to ROS. In ROS, there are nodes that publish topics that others can subscribe to. In this case, data from SigSent's sensors and intelligent systems can be easily passed between fragments of code by simply "subscribing" to the whichever information is needed by that node/class. The publish-subscribe design pattern is very scalable. In the case that new, more intelligent systems are provided, or additional data inputs are necessary, new nodes in the dependency/usage graph can be added. There can also be additional complexity in how the data is used and filtered by the subscribers. If many nodes need to strip the sensor data to a new, parsed format, a new node can be added that performs that filtering and then publishes this new data for others to take advantage of.

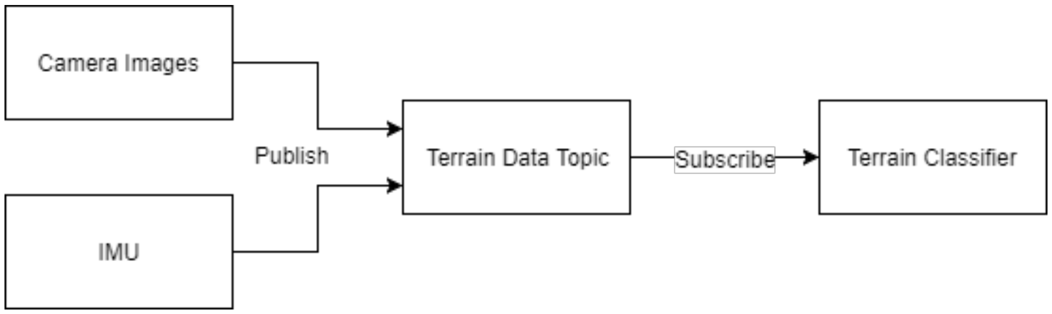


Figure 37 Sample diagram representing basic Pub-Sub

5.4.2.4 Class Diagrams

Class diagrams help visualize a lower level of the program's structure (while still at some high level of abstraction). In the planning stages of software development, it is important to plan out what classes will be necessary and how they will be managed. Inheritance and OOP concepts like polymorphism can be discussed in this stage such that the program can be developed in the most readable/usable manner. Below are the two intelligent systems modules with plans for their possible methods and instance variables.

5.4.2.4.1 Terrain Classifier Class Diagram

The terrain classification system was intended to be much simpler than the neuroevolution performed in the mobility mechanism decision process. A simple Artificial Neural Network (ANN) is created and trained against a test set of data (sample or artificially created data mimicking that from the LIDAR unit and camera) with outputs for what type of terrain those inputs should be classified as. By running a backpropagation algorithm, described in a former research section, the ANN has its weights updated such that training set of data passed to the train_classifier method is accurately classified. Backpropagation updates the ANN's weights by calling the update_weight method in the ANN's data structure class. After training the classifier, the classify_terrain method should be able to precisely decide what terrain is being imaged by the inputted LIDAR and camera data. These inputs are passed to the ANN housed within this class where the output is given by the compute_output method. The main classifier class must take the int output from this method and choose one of the enum values for TerrainType, whichever one corresponds to the ANN's output.

For SigSent, this approach was not taken in its prototype build, however its layout is displayed below.

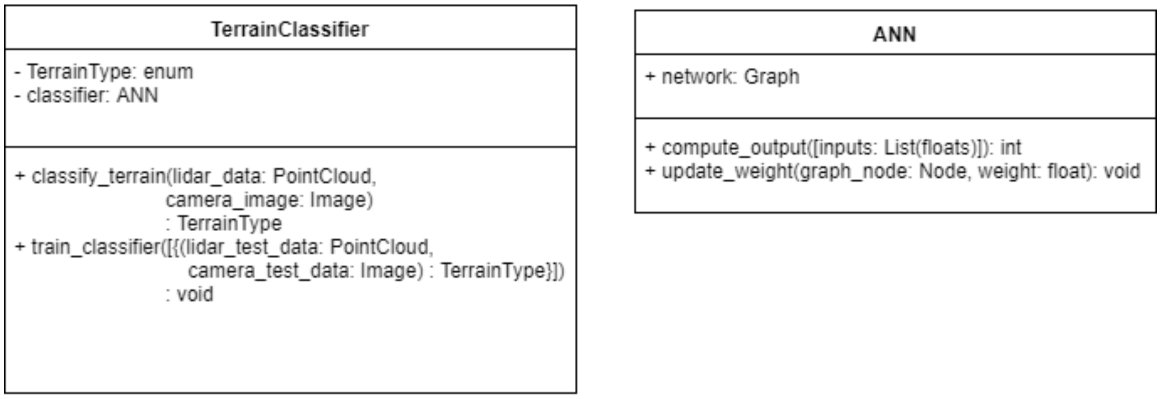

TerrainClassifier
- TerrainType: enum
- classifier: ANN
+ classify_terrain(lidar_data: PointCloud, camera_image: Image) : TerrainType
+ train_classifier([{(lidar_test_data: PointCloud, camera_test_data: Image) : TerrainType}]) : void
ANN
+ network: Graph
+ compute_output([inputs: List(floats)]): int
+ update_weight(graph_node: Node, weight: float): void

Figure 38 TerrainClassifier Class Diagram

5.4.2.4.2 NEATManager Class Diagram

The NEATManager class is used to communicate between the ROS backbone and with the multi-terrain NEAT classifier. There are three main ways to run the manager, as a training session with a provided list of sample data (like when being run OFF of the robot, in a simulation), with live data streamed from ROS containing information about what is happening at discrete ticks in real time, or as a normal run where one-time step's input is sent and run through the ANN, with a single output being returned to enable or disable to multi-terrain setting. The ROS nodes responsible for this knowledge transfer will carry sensor data and mobility classifications to and from the NEATManager.

There are also useful methods for outputting data on the current feedforward ANN that exists in the manager. The ANN can be outputted as an svg image, displaying the nodes and weighted connections in a graph. The average speciation of the last training session can be shown as well. This will display how significantly different network topologies propagated throughout the search so that we can decide if the search space was properly explored to find the current best network. The statistics of the latest training session can be viewed as well. The `output_stats` method can be used to display this information, regarding best fitness values discovered, at what iteration they were found in, and training rates for the improved learning that took place over time. This can be helpful in seeing if the training was effective in improving over time, if suboptimal (local maxima) solutions were found and escaped from, and depending on the concavity of the fitness scores, whether our number of generations in training the NEAT ANN was long enough. If the fitness was still continuously improving, training time can be added such that we do not finish running the session until some horizontal asymptote is found and the fitness has leveled off. Positive concavity would show that the fitness is increasing more over time and that there is still a lot of hill climbing for the intelligent system to complete to find the best-fit ANN topology and weight combination.

Due to the items outlined above in 4.2.6.3, the NEAT code discussed was not implemented, but is still referenced here for future work.

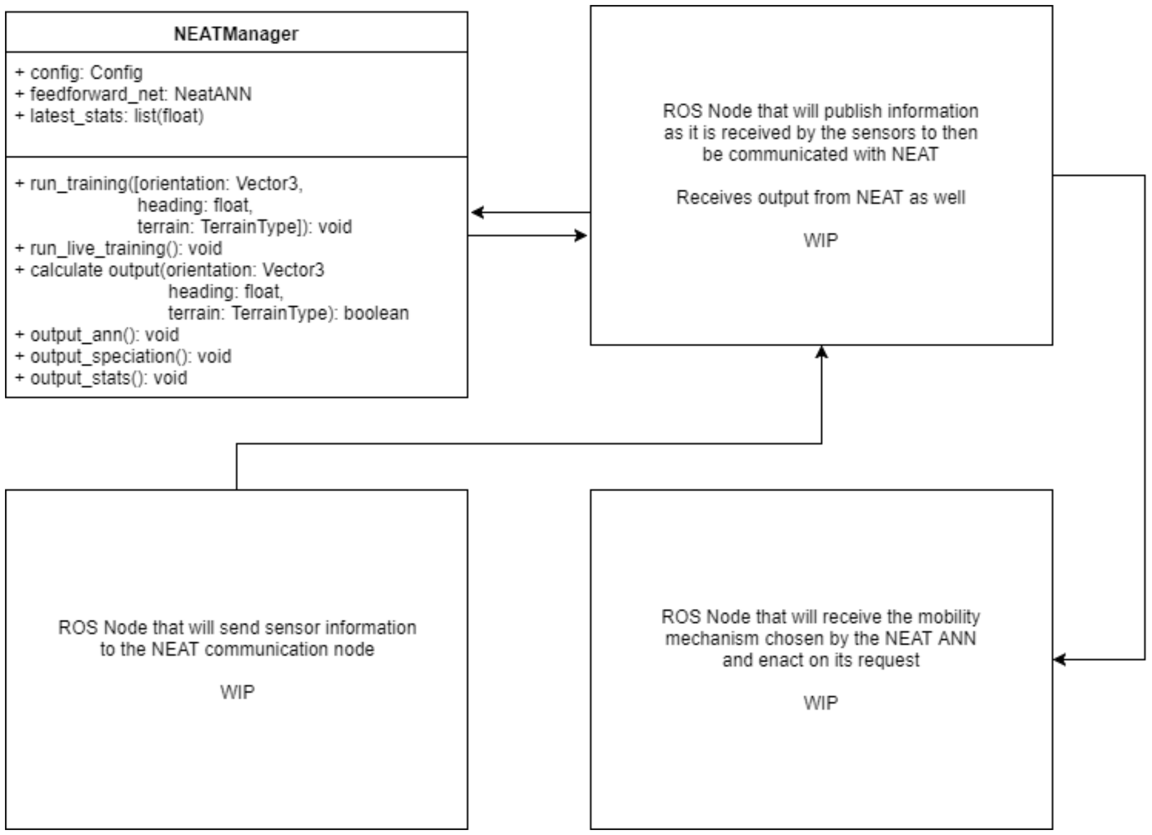

NEATManager
+ config: Config
+ feedforward_net: NeatANN
+ latest_stats: list(float)
+ run_training([orientation: Vector3, heading: float, terrain: TerrainType]): void
+ run_live_training(): void
+ calculate output(orientation: Vector3 heading: float, terrain: TerrainType): boolean
+ output_ann(): void
+ output_speciation(): void
+ output_stats(): void
ROS Node that will publish information as it is received by the sensors to then be communicated with NEAT
Receives output from NEAT as well
WIP
ROS Node that will send sensor information to the NEAT communication node
WIP
ROS Node that will receive the mobility mechanism chosen by the NEAT ANN and enact on its request
WIP

Figure 39 NEAT Class Diagram

5.4.2.5 Use Case Diagrams

UML Use Case diagrams are a high-level representation on who will interact with the software (known as actors) and which features they will be able to access and enable within the program's bounds. The interactions between the human user, base station controller, and the SigSent robot are noted below. In most UML use case diagrams, relationships for "include" and "extend" are shown in the actor associations. Since our actors have many single actions with little association necessary to denote, our use case diagram showing each module's abilities and program responsibilities does not include this extraneous relationship [51].

The user's actions are displayed as what steps they can take through the GUI program to interact with the base station, and through that, the SigSent robot. Their first action necessary to work with the robot is to launch the base station program (assuming that the base station is turned on and fully operational through testing). The user can speak into their microphone and listen from their speaker with the headset peripheral to use the audio communication capabilities. The user can also use the GUI to view the camera feed from the robot. This is the user's main source of sentry abilities. They will be able to do surveillance remotely with the robot in this manner.

The base station program is responsible for being the effective middle man medium through which the user communicates with the robot. The base station is connected to the robot through Wi-Fi from which it gains the necessary statistics and diagnostics from the robot's sensors. These diagnostics are shown visually in the GUI mockup in the prototyping section 0. The diagnostics include information on remaining battery life, IMU values, terrain classification by the terrain classifier class shown above in its class diagram in section 5.4.2.4.1. The base station will transfer the joystick inputs coming from the joystick to the robot to be used in moving the unit according to the direction of the joystick.

Finally, the SigSent robot has its own level of interaction with its features. Each action it is associated with is related with its high-level feature list described from the requirements and objectives. The SigSent robot will send audio using its local microphone mounted on the robot to the base station. The audio will be sent over the Wi-Fi connection it has established with the base station. The robot will output audio from the user's microphone on the base station headset via the robot's speakers. The robot will be moved through the TeleOp feature whenever the user is using their joystick and enabled the TeleOp mode, rather than the Sentry mode, in the base station's GUI program. Only when these conditionals are made true will the robot be moveable from the base station's joystick commands. The robot's sensor information is sent to the base station in real-time for active debugging and for ease of use. Without information like the battery life, the user would not be able to effectively operate the robot without worrying about the robot becoming stranded or failing while at work in a mission critical situation. The SigSent robot will be sending raw values for its sensors to the base station to lower the computational load on the robot's microcomputer and microcontroller operating environment. The base station GUI program will handle any necessary sanitization of inputs and beautifying the outputs to be as readable as possible. Finally, the SigSent unit will be changing its mobility mechanism dependent on the NEAT artificial intelligence module in the intelligent systems section. This is the main, sophisticated operation of the robot featured in SigSent.

While these actions associated with each operating actor seems minimal, the infrastructure supporting each miniscule action contains significant overhead in discrete calculations and CPU computation, and code supporting it all. The class diagrams outline the software's high-level plan as "code" while the use case diagrams note what each system's level of interaction is and their possible actions they may take.

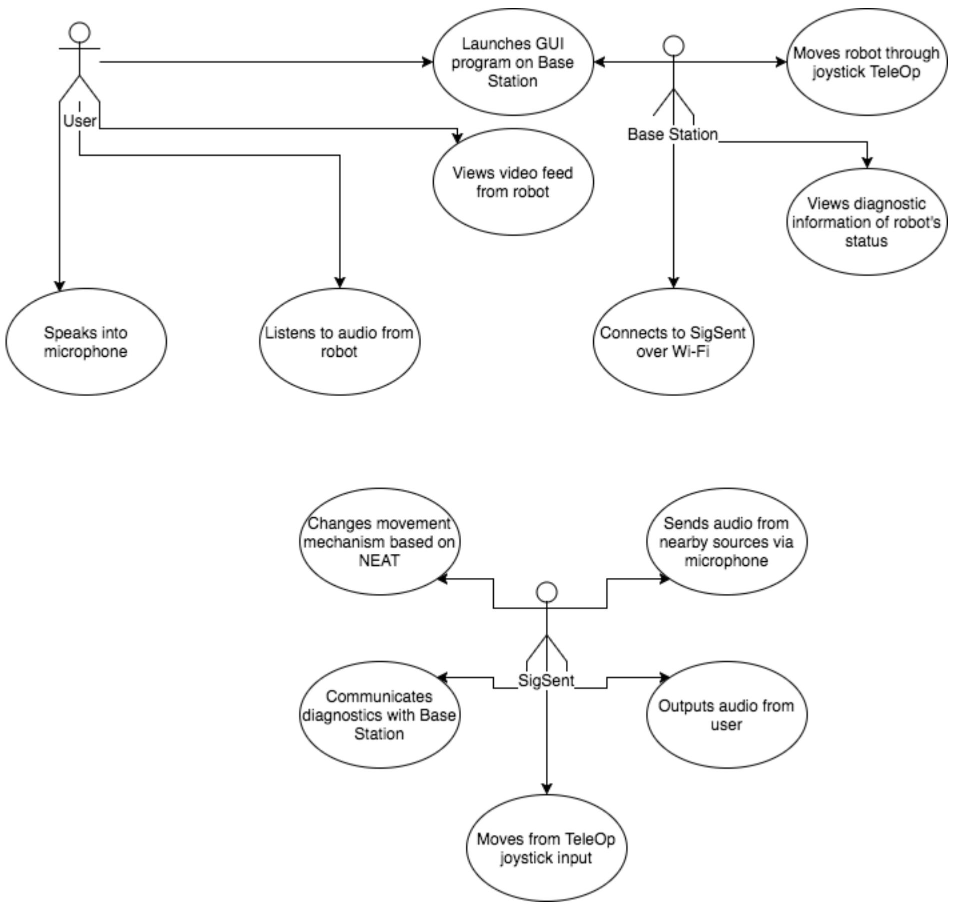

User
Launches GUI program on Base Station
Base Station
Moves robot through joystick TeleOp
Views video feed from robot
Views diagnostic information of robot's status
Speaks into microphone
Listens to audio from robot
Connects to SigSent over Wi-Fi
Changes movement mechanism based on NEAT
Sends audio from nearby sources via microphone
SigSent
Communicates diagnostics with Base Station
Outputs audio from user
Moves from TeleOp joystick input

Figure 40 UML Use Case diagram on User, Base Station, SigSent interaction

5.4.3 State Machine

SigSent has three main states: Sentry State, Patrol (Walk Path) State, and Interface State. There are also be several other meta states such as Alert, and Status.

In Sentry State, SigSent stays motionless in a designated spot and stands watch while processing camera and lidar data looking for anomalies and unknown behavior. Upon detection of any unknown behavior the state will exit and change to the alert state.

In the Patrol (Walk Path) State, SigSent walks along a preprogrammed path (a set of GPS waypoints previously programmed) and sends those goals to the path planner which determines the ideal path for SigSent to take depending on priority between time, energy, and risk to robot. Once the path planner determines the ideal path it will send a vector to the Active Suspension Program which will then calculate the ideal values to send to its motors and suspension system based on various sensor data and the output of a NEAT ANN trained to determine which terrain mode the robot should use. Upon detection of any unknown behavior the state will exit and change to the alert state.

In Interface State, SigSent can send status and telemetry to the base station containing information such as battery percentage, live streaming the camera on SigSent, and viewing the current state of the NEAT ANN. In Interface State the robot can also be programmed with a new GPS waypoint path, teleoperated, or have the operator's voice transferred to output on the robot's speaker.

the Alert State is only entered when either the sentry or patrol state detect a human presence. When this happens, the robot will send an "ALERT" signal to the base station where the base station will be alerted to the unknown behavior.

The Status state is only entered when the robot has been placed in a situation that the robot deems needs human intervention but is not an alert. Things such as low battery, stuck in terrain, or otherwise unknown or diagnostic mode. In this state GPS beacons are sent to the base station to better help humans find the robot.

5.4.4 Base Station

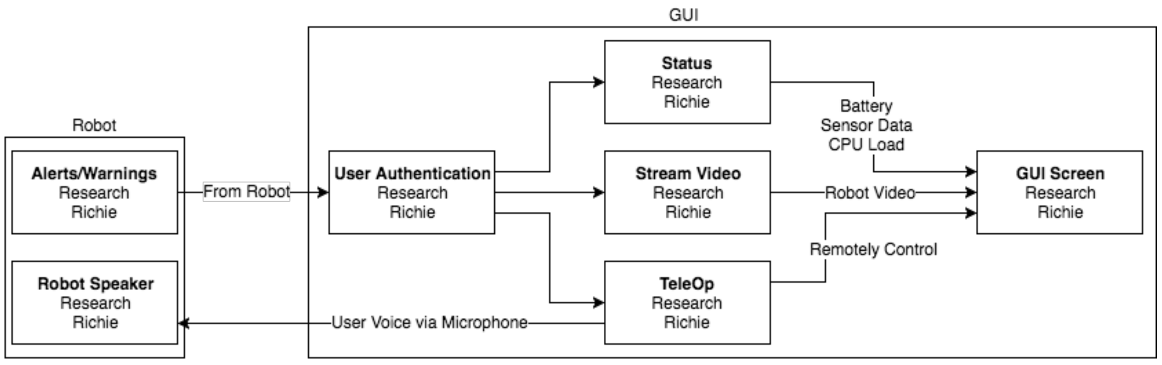


Figure 41: Base Station GUI Diagram

A base station is used to communicate with and remotely operate the robot. Alerts and warnings from the robot encompassing low battery levels and motion detection will always appear on the main screen of the GUI. The GUI is accessed by a single administrator log-in manually created when setting up the system to prevent prying eyes. The user can selectively view the status of the vehicle (including battery levels, raw sensor data, and the current CPU load), watch a streamed video feed from the robot's camera, and remotely control the robot with a Logitech Flight Stick (Extreme 3D Pro Joystick). The TeleOp control also allows for the user to speak into a microphone at the base station that will then project the audio from the robot's attached speaker.

5.4.5 SPI Communication

The Raspberry Pi communicates with the MCU over SPI. A header byte message is sent that details what commands will follow. The header is necessary so that the MCU knows what bitmasks to use to parse the bytes.

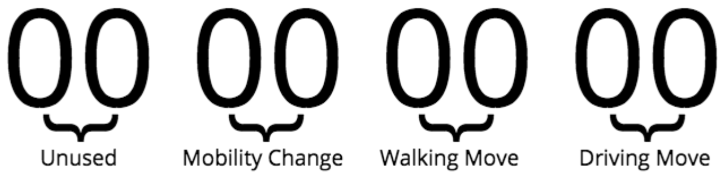


Figure 42 Bitmask header example

As shown in Figure 42 Bitmask header example, the byte will have two of its eight bits set to a value of 1 to signify what kind of message will be following the header. Two of the most significant bits were left unused for future proofing our design. This allows us to add another possible command in the future without changing our existing message structure while also keeping a hamming distance of four among the entire code set of the three different message types.

The hamming distance should be maximized to preserve functionality in the case of errors in the data transmission. Although errors are not anticipated in a wired SPI network as we are using, errors should be minimized to avoid damaging the robot or nearby pedestrians if an incorrect command is parsed and run. With a hamming distance of four, an error of three bits can be detected and avoided. A one bit error could be corrected, though correction was not necessary for our application [52].

The messages that follow use a similar convention, using set bitmasks to define some command while also maximizing the hamming distance among the set. The tables below outline what each command's message sequence contained.

Table 20 Walking SPI messages

Walk Header (Byte one)	00001100
Forward (Byte two)	00000000
Left (Byte two)	00001111
Right (Byte two)	11110000

Table 21 Driving SPI messages

Drive Header (Byte one)	00000011
Forward (Byte two)	00000000
Left (Byte two)	00001111
Right (Byte two)	11110000
Backward (Byte two)	11111111
Speed (Byte three)	00000000 -> 11111111 (Range)

Table 22 Mobility change SPI messages

Change Mode Header (Byte one)	00110000
Enter Driving Mode (Byte two)	00000000
Enter Walking Mode (Byte two)	11111111

5.4.6 NEAT

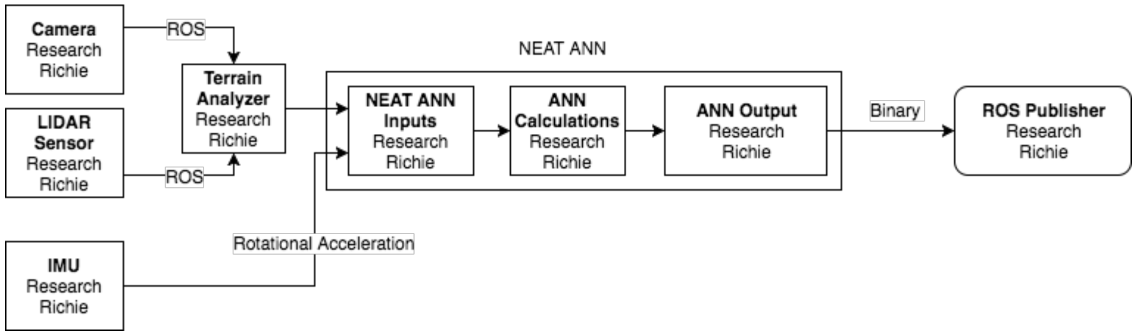

Camera Research Richie
ROS
LIDAR Sensor Research Richie
ROS
Terrain Analyzer Research Richie
IMU Research Richie
Rotational Acceleration
NEAT ANN
NEAT ANN Inputs Research Richie
ANN Calculations Research Richie
ANN Output Research Richie
Binary
ROS Publisher Research Richie

Figure 43: NEAT ANN Diagram

Using the NeuroEvolution of Augmenting Topologies (NEAT) library, the robot could learn to alternate between mobility types based on the environment it is traversing. Throughout the learning phase, NEAT will create Artificial Neural Networks (ANNs) that a Genetic Algorithm (GA) will use and score based on their performances. Each ANN is used in a test environment where sensor values are passed as inputs into the network to receive some desired output values. A camera and LIDAR will be used to identify what kind of terrain the robot is moving over, which is then sent as an input into the network. The IMU will pass its rotational acceleration values as a second input. The output is a binary value of what type of mobility mechanism to engage. The best ANNs are used to create a new population of networks, using popular genetic operators from biology, including crossover and mutation. Crossover occurs more frequently, moving values from performant networks to create successful offspring. Mutation continues to add diversity to the population so that NEAT properly explores the domain's search space.

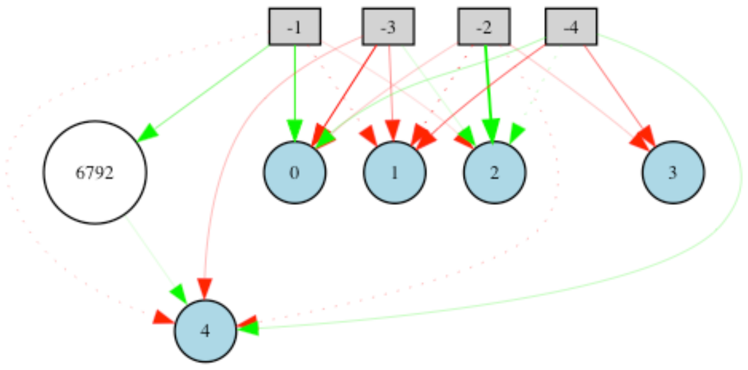


Figure 44: Example Generated Neural Network

In the example above, the gray squares are the input nodes and the blue circles are the output nodes. This minimal example was performed in a command-line environment using a 2D grid as the environment, where a robot is an object on the map (with a location designated by its *x* and *y* coordinate on the grid) and has access via "sensors" to its four neighboring cells in each compass direction in a non-toroidal map. The four neighboring cells are inputs into the ANN. Four of the five outputs correspond to the future direction the robot will be headed in, where the node with the highest output value decides what direction the robot takes. The final output node chooses what mobility type the robot will enter prior to moving. If the robot attempts to move onto a cell labeled as being a "rocky" environment, it must have the proper mobility mechanism engaged before it can actually move onto the new cell. A fitness score is assigned based on how far the robot travels, as well as how many unique cells it visits. Based on this fitness, this ANN can be compared to the performances of the other ANNs in the GA's population to decide on what network topologies will continue to proliferate and what search directions should be pruned.

As shown in the graph to the right Figure 45: Fitness of the Example Network, the average fitness is steadily growing while the best seen fitness value makes jumps whenever a new, well-performing ANN is discovered. A better ANN results in a better "brain" controlling the robot. Higher fitness values can be achieved by modifying the NEAT parameters to be more optimal for this specific use case. The generation limit should be increased until the fitness levels off at an acceptable value. Additional time should also be considered to account for the GA struggling to break away from suboptimal extrema.

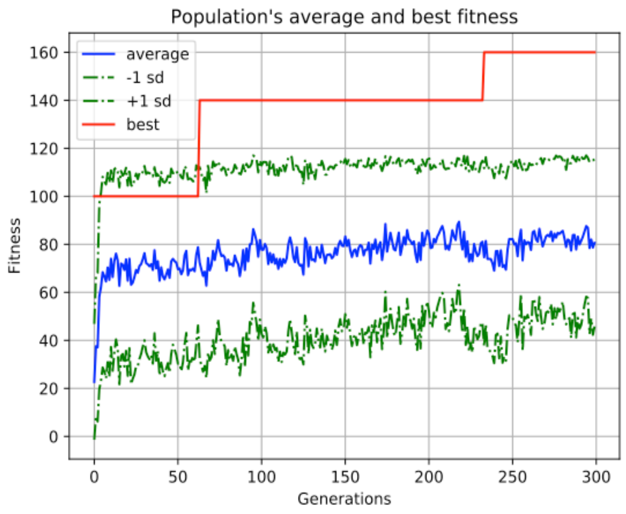


Figure 45: Fitness of the Example Network

5.4.7 Kinematics of Movement

In order for the robot to know how to walk, an algorithm is needed so the robot can figure out how to move its legs. The robot can't just flop its legs around in a random fashion as this would not guarantee a stable movement for the main body of the robot. It also can't have hard coded movements for its legs as this would not allow for a robust and adaptive mode of mobility, and would spell disaster as the robot would hit rough terrain with heights and obstacles unknown and inevitably fall over. The way to solve this is with kinematics of movement/rotation. With techniques in this field, a model and algorithm for an individual leg, then for the chassis with all the legs can be created. This would be converted into code for the on-board computer to handle the desired movements. This algorithm will then be tested and validated in Gazebo until it is tuned for the Ideal movement scheme for the robot.

These models and algorithms of the kinematic system of the robot will be created using two concepts called forward kinematics and inverse kinematics.

5.4.7.1 General Set-Up

Our robot's system is comprised of six legs, each with three degrees of freedom. This ends with a total of 18 degrees of freedom for our system. To start the calculations, we need to simplify the system down to its non-redundant/irreducible state. That is one leg with three joints or three degrees of freedom as shown in the figure below: Figure 46: Diagram of a single leg of SigSent Robot demonstrating three jointed members.

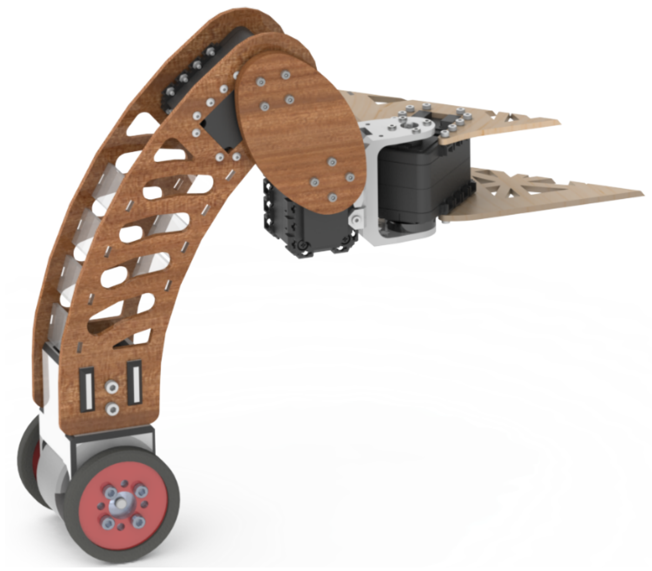


Figure 46: Diagram of a single leg of SigSent Robot demonstrating three jointed members.

To simplify it even further we can look at the outer two joints of the leg excluding the joint connecting directly to the chassis, which makes for a problem with only two degrees of freedom as shown in the figure below: Figure 47: Kinematic diagram of two degree of freedom linkage system [53]. The paper reference for the figures below were modeling a robotic arm but the it still applies to our robot leg if you think of the first joint as the origin instead of the base of the system.

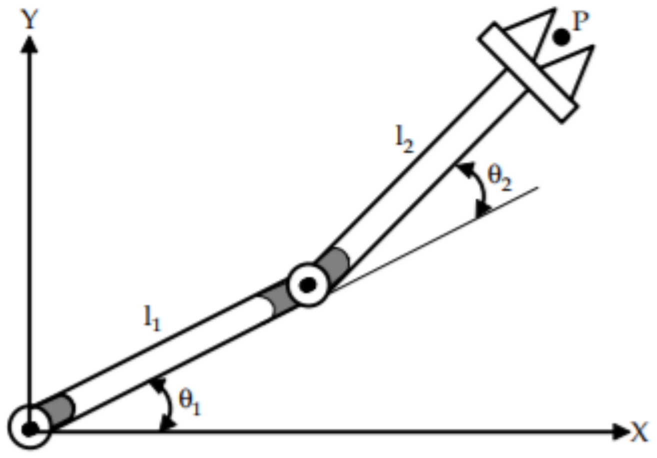


Figure 47: Kinematic diagram of two degree of freedom linkage system

This system can now be reduced down to a representation linkage system with linkage members as lines and joints as connecting nodes. From there the simple linkage dimensions can be derived from simple geometry and trigonometry in an XY cartesian coordinate system or even a radial/circular coordinate system. From this we are able to visually see and represent the end point or the robot's end effector for the linkage system as point, P. We also label all measurements such as the lengths of the linkages, l, and angles of the of the linkages with respect to our frame of reference and relating the subsequent dimensions of all linkage's end effects that cause the position of the final end effector. This is shown in the figure below: Figure 48: Trigonometric kinematic diagram of two degree of freedom linkage system [53].

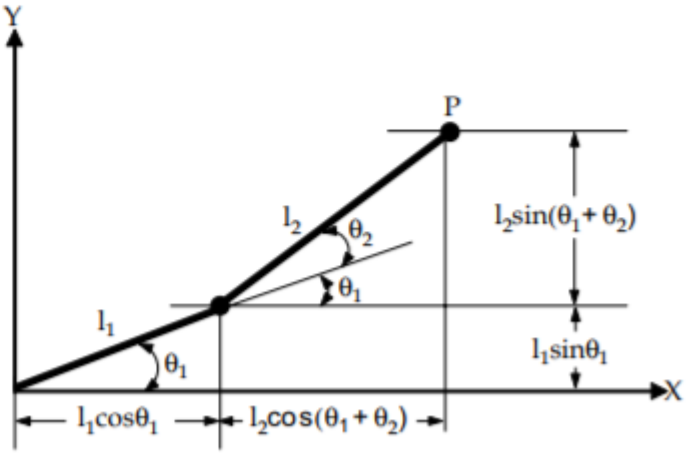


Figure 48: Trigonometric kinematic diagram of two degree of freedom linkage system

Our reasoning for this simplification is that the joint that we are excluding only deals with rotation in a purely z-axis motion, while the other two joints dictate the extension of our legs and where they will fall on the ground. Thus, making it that these two joints are responsible for the contact with the ground to ensure a firm foot hold and in the end balance of the overall system, the excluded joint ends up contributing more to the directional movement of the system. While these assumptions previously stated for the simplification do hold true in the case presented, they will not always hold true when moving the robot over rough and rugged terrain, but for the fundamental derivations we will be looking at the simplified two degree of freedom model to start. This model allows us to solve for our scenario via trigonometry.

A common method that is used represent and solve for these linkage measurements is the Denavit-Hartenberg parameters. "These are four parameters that are associated with a particular convention for attaching reference frames to linkages of a spatial kinematic chain. [54]" The four parameters that define the Denavit-Hartenberg model are the link length (a_i), link angle(α_i), link offset(d_i), and joint angle(θ_i). These four parameters relate the next link to the current link with their respective frames of reference. This is shown in the below figure: Figure 49: Representational kinematic diagram for Forward Kinematic Denavit-Hartenberg parameters definition [55].

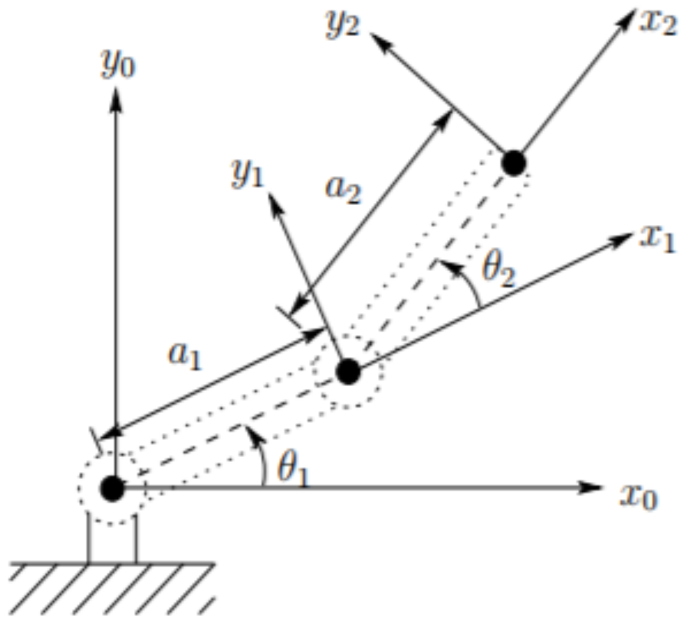


Figure 49: Representational kinematic diagram for Forward Kinematic Denavit-Hartenberg parameters definition

These parameters can be simplified for our scenario down to the joint angle and the link length. This only holds true since the axis of rotation for the joints are parallel for this scenario.

Both Forward and Inverse kinematics solutions can be solved for using either a trigonometry or Denavit-Hartenberg parameters.

5.4.7.2 Forward/Direct Kinematics

One of the techniques used that is to create the control algorithms in each of the legs and joints of SigSent is forward or direct kinematics. This implementation and derivation will be discussed below for a simplified version of the leg (two-linkage system) and then the true arm (three-linkage system) for algorithm simplification in certain case and scenarios for the rough terrain movement.

5.4.7.2.1.1Forward Kinematics Two-Linkage Implementation

The concept of forward kinematics is very similar to most problems solved in math courses and the one people will be most familiar with. Forward kinematics is a solution to the model of a system given all the inputs of that system. So given the system from the general setup section, if we wanted to know where the end effector of our leg would land then we would simply need to give the system some inputs for the length of the linkages and angles of rotation provided by the servo motors once they are calibrated and the final coordinate of the end effector for the system can be solved for as the desired next position. From this desired position in space, we can solve for the resulting heights and widths of the linkages and determine from the boundary condition discussed before to verify if this desired position is possible given our environment and current position or if our joints will end up reaching a limiting angle of rotation.

This the forward kinematics algorithm for movement is simple and straightforward but if the system doesn’t have quite a few sensors available on them a lot of the resulting effects on the system's stability of the desired next position will remain unknown unless tested. Additional if the wrong inputs are chosen and calculated to cause an issue with the system then a whole new set of inputs will have to be selected and the end effector recalculated. This has the potential to pull a lot of resources from the on-board computer and could end up delaying other system responses, if the operations take too long to solve for a viable solution. This makes higher degrees of freedom models too costly to control.

This method will be a good for and implemented for solving the simple and quick movements over a relatively smooth or flat surface but would not do so well for us under rough terrain that requires precise placement of footing for the stability of the system.

The derivations for the two-arm linkage system starts with the derivation of the Denavit-Hartenberg parameters. Shown above in the general set-up section.

Table 23: List of Forward Kinematic Denavit-Hartenberg parameters

Link	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	a_1	0
2	θ_2	0	a_2	0

From this the matrices that define the system can be created. With the shorthand notation of c_i & s_i being equal to cos θ_i & sin θ_i respectively, $\theta_1 + \theta_2$ by θ_{12}, and cos($\theta_1 + \theta_2$) as $c_{12,}$ we obtain the matrices shown below [55].

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 7: Foward Kinematic Denavit-Hartenberg Matrices

This then leads to the derivation of the T matrices which yields:

$$T_1^0 = A_1$$

$$T_2^0 = A_1A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 8: Derivation of the T Matrices

From these matrices the position of the end-effector in relation to the base frame is given by the first two elements in the last column of T_2^0. This is shown below [55].

$$x = a_1c_1 + a_2c_{12}$$

$$y = a_1s_1 + a_2s_{12}$$

Equation 9: Position of the End-Effector in relation to Base Frame

5.4.7.2.2 Forward Kinematics Three-Linkage Implementation

Similarly, to the derivation above in the inverse kinematic for the two-linkage system, we can solve for the three-linkage system of the whole leg. For this we will use the Denavit-Hartenberg parameters again this time with the values for alpha and d not equal to zero since the axis orientation from the first joint to the second aren't parallel. Thus, ending up with this general form of a transformation matrix from linkage I to linkage i-1 [56].

$$T_2^0 = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\alpha_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i cos\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 10: General Transformation matrix for the forward kinematics three-linkage system

The resulting coordinates for the end effector can be solved for which results in the equations below. [56].

$$x = cos\theta_1(L_1 + L_2 cos\theta_2 + L_3 \cos(\theta_2 - \theta_3))$$
$$y = sin\theta_1(L_1 + L_2 cos\theta_2 + L_3 \cos(\theta_2 - \theta_3))$$
$$z = d_1 + L_2 sin\theta_2 + L_3 \sin(\theta_2 - \theta_3)$$

Equation 11: Position equations solutions from the Forward kinematics implementation

From this derivation the position equations (x, y, z) for the end effector has been solved for. This allows us to put in values for the linkage lengths and angles of the servo motors to get a resulting position. These equations will be a step toward the gait generation method.

5.4.7.3 Inverse Kinematics

One of the other techniques that is used to create the control algorithms in each of the legs and joints of SigSent is inverse kinematics. This implementation and derivation will be discussed below for a simplified version of the leg (two-linkage system) and then the true arm (three-linkage system) for algorithm simplification in certain case and scenarios for the rough terrain movement.

5.4.7.3.1.1 Inverse Kinematics Two-Linkage Implementation

The concept of inverse kinematics is the opposite of forward kinematics. We need to solve for the inputs/angles of the joints of the system that would result in a desired end effector position. While it is more versatile and popular in a lot of applications, can be quite difficult to solve and may not even have a solution or a unique one either. The solution can be solved analytically but some cases require a numerical solution as the equations might not be directly solvable. This make boundary conditions, initial inputs, and the very dimensions/structure of the linkage system is very important to include to the derivation and solution.

While it is costly to compute via this method it can deal with higher degrees of freedom and give precise desired movement once finally solved. This method will be a good for and implemented for solving the complex movement needed for traversing over rough uneven terrain. A simple distance sensor is needed to find approximate distance from the robot chassis to the ground and then the equations can be solved to get the desired position of the robot's legs that would keep stability in the system.

The derivations for the two-arm linkage system starts with the derivation of the Denavit-Hartenberg parameters to solve for the known general solution of the system. This equation relates the base frame of reference to the frame of reference of the end effector. This transformation matrix can be described by the multiplication of the reference frames of every joint down to the end effector in series [53].

$$_{end\ effector}^{base}T = {}_{1}^{0}T * {}_{2}^{1}T * \dots * {}_{n}^{n-1}T$$

Equation 12: Transformation Matrix

The base transformation matrix can also be defined as the matrix representing the rotation elements of the system as well as the position of the end effector. Shown below [53].

$$_{end\ effector}^{base}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 13: Transformation Matrix based on Rotation Elements

Definition of the Denavit-Hartenberg parameters are shown below in the figure and table below: Figure 50: Representational kinematic diagram for Inverse Kinematic Denavit-Hartenberg parameters and Table 24: List of Inverse Kinematic Denavit-Hartenberg parameters [53].

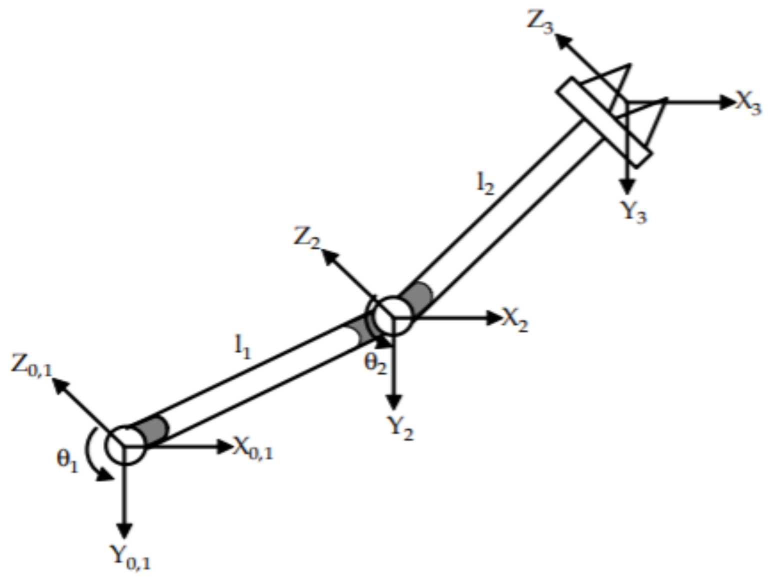


Figure 50: Representational kinematic diagram for Inverse Kinematic Denavit-Hartenberg parameters

Table 24: List of Inverse Kinematic Denavit-Hartenberg parameters

Link	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	0	0
2	θ_2	0	l_1	0
3	0	0	l_2	0

From this, the linkage transformation matrices that define the system can be created. With the shorthand notation of $c\theta_i$ & $s\theta_i$ being equal to cos θ_i & sin θ_i respectively, we obtain the matrices shown below [53].

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_1 & -s\theta_2 & 0 & l_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}_{3}^{2}T = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 14: Transformation Matrices for each Joint

From a general solution described above and simple matrix manipulation we can obtain the equation shown below [53].

$$ {}_{1}^{0}T^{-1} * {}_{3}^{0}T = {}_{2}^{1}T * {}_{3}^{2}T $$

Equation 15: General Solution of Transformation Matrices of each Joint

After the multiplication of the defined matrices we obtain the partial solved equation of [53].:

$$ \begin{bmatrix} . & . & . & c\theta_1 p_x + s\theta_1 p_y \\ . & . & . & -s\theta_1 p_x + c\theta_1 p_y \\ . & . & . & p_z \\ . & . & . & 1 \end{bmatrix} = \begin{bmatrix} . & . & . & l_2 c\theta_2 + l_1 \\ . & . & . & l_2 s\theta_2 \\ . & . & . & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 16: Partially Solved Transformation Matrix

This will result in solutions for θ_1 & θ_2. Starting with θ_2 we can square both matrices and set the elements [1,4] and [2,4] of both matrices equal to each other. After some simple algebraic manipulation, a solution for θ_2 arises shown below [53].

$$ c^2\theta_1 p_x^2 + s^2\theta_1 p_y^2 + 2p_x p_y c\theta_1 s\theta_1 = l_2^2 c^2\theta_2 + 2l_1 l_2 c\theta_2 + l_1^2 $$
$$ s^2\theta_1 p_x^2 + c^2\theta_1 p_y^2 - 2p_x p_y c\theta_1 s\theta_1 = l_2^2 s^2\theta_2 $$

$$ p_x^2(c^2\theta_1 + s^2\theta_1) + p_y^2(s^2\theta_1 + c^2\theta_1) = l_2^2(c^2\theta_2 + s^2\theta_2) + 2l_1 l_2 c\theta_2 + l_1^2 $$
$$ p_x^2 + p_y^2 = l_2^2 + 2l_1 l_2 c\theta_2 + l_1^2 $$
$$ c\theta_2 = \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2} $$

Equation 17: Algebraic Manipulation of Transformation Matrices

For θ₂ we can derive a solution from Table 2 in Reference [53]. This leads to a solution of:

$$\theta_2 = Atan2\left(\mp\sqrt{1-\left[\frac{p_x^2+p_y^2-l_1^2-l_2^2}{2l_1l_2}\right]^2}, \frac{p_x^2+p_y^2-l_1^2-l_2^2}{2l_1l_2} \right)$$

Equation 18: Solution for θ_2

For θ₁ we can use elements [2,4] of both matrices and set them equal to each other [53].

$$c\theta_1 p_x + s\theta_1 p_y = l_2 c\theta_2 + l_1$$

Equation 19: Partial Solution for θ_1

We can then derive a solution from Table 2 in Reference [53]. This leads to a solution of:

$$\theta_1 = Atan2(p_y, p_x) \mp Atan2(\sqrt{p_y^2 + p_x - (l_2c\theta_2 + l_1)^2}, l_2c\theta_2 + l_1$$

Equation 20: Solution for θ_1

With the two joint angles solved for we have a completed solution for the inverse kinematic algorithm for our legs. As stated above we the inverse kinematic solution will need boundary/limiting condition to avoid unwanted behavior in the leg movements as this derivation leads to multiple solutions for our desired positions.

5.4.7.3.1.2Inverse Kinematics Three-Linkage Implementation

Similarly, to the derivation above in the inverse kinematic for the two-linkage system, we can solve for the three-linkage system of the whole leg. For this we will use the Denavit-Hartenberg parameters again this time with the values for alpha and d not equal to zero since the axis orientation from the first joint to the second aren't parallel. Through basic trigonometry some angle values can be solved given the figure shown below [56].

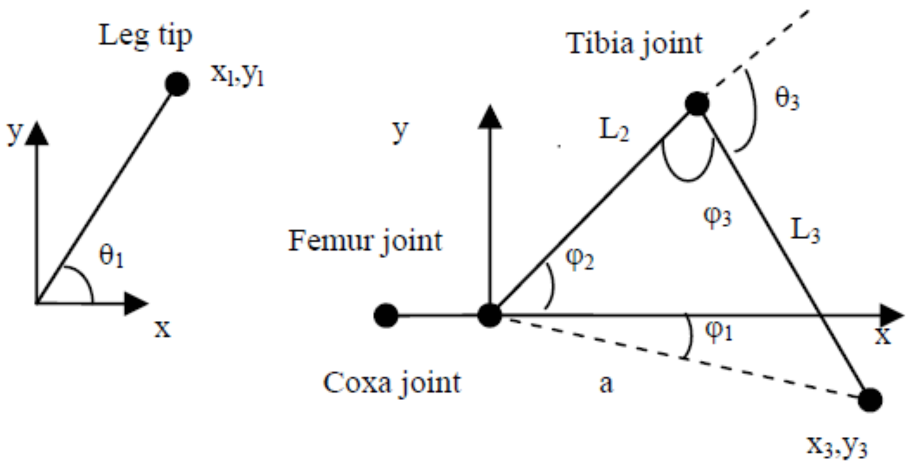


Figure 51: 2D planar view of the joints of SigSent's leg [56].

$$\theta_1 = atan2(y_1, x_1)$$

Equation 21: Solution for θ_1

With use of a transformation matrix to convert the end effector coordinates to the coxa frame. A solution for the remaining angles was derived. [56]

$$\varphi_1 = atan2(y_3, x_3)$$

$$\theta_2 = \varphi_2 = acos\left(\frac{L_2^2 + a^2 - L_3^2}{2L_2a}\right) + \ atan2(y_3, x_3)$$

Where $a = \sqrt{x_3^2 + y_3^2}$ from law of cosines

$$\varphi_3 = \ acos\left(\frac{L_2^2 + L_3^2 - a^2}{2L_2L_3}\right)$$

$$\theta_3 = \pi - \varphi_3$$

Equation 22: Angle equations solutions from the Inverse kinematics implementation

From this derivation the angle equations for the joints of the leg have been solved for. This allows us to put in a desired coordinate for the leg to move to and the required servo angles to achieve that coordinate can be solved for. These equations are will be a step toward the gait generation method, just as the forward kinematics solution would be.

5.4.7.4 Gait Generation

One of the concepts that is used to create the control algorithms of movement for each of the legs and joints of SigSent is Gait Generation. The different possible techniques that will be used in an effort to generate the different gaits will be discussed below.

5.4.7.4.1 What is Gait?

In reference to SigSent, gait refers to the style or manner that the hexapod will walk, this inherently refers to the line of motion in 3D space the legs make for the hexapod to move in any direction as shown in Figure 52:Gait path diagram below [57]. Gait usually is designed to emulate insectoids already in nature to attempt to maintain efficiency and natural looking movements. They are usually modeled after sinusoidal wave forms to attempt to match a natural arching curve in the stride forward. There are multiple methods to solve and model for gait, those are the attempts made to find the best method and solution to a gait generator for SigSent.

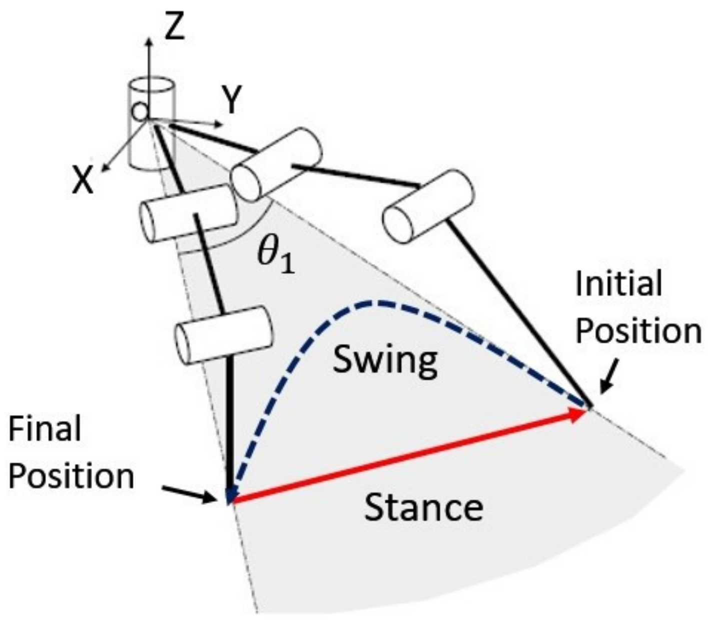


Figure 52:Gait path diagram [57]

5.4.7.4.2 Gait Generation via Kinematics modeling

This method involves modeling the exact path the legs will take to perform its gait. This is usually done path planning with a polynomial function of sorts to create a known value that the leg is desired to reach. Then the path is broken up into important key frames that the legs must reach. The values of the key frames are then put through the forward or inverse kinematic equations to get the subsequent values needed for the servo motors to reach the desired values.

In order to generate the commands needed to reach the key frames that are defined the kinematics modeling of forward and inverse kinematics must be already done as to find the coordinate locations of the desired configuration or joint angles must be definable or solvable. This method involves a more brute force technique in order to generate the desired gaits of the hexapod's legs. While this is the case, this method still does provide appropriate solutions for the gaits.

5.4.7.4.3 Gait Generation via Neural Networks

This method of gait generation follows the same principle that is defined in the sections 4.2.6.2 about reinforcement learning & 5.4.6 about NEAT. A random set of gaits will be generated and populate the system. The gaits will then be tested in simulation and given fitness values based on the performance of the different gaits. From the results, based on known outputs or targets the weights for the Neural Networks are recalculated in either a forward or back propagation fashion and the new gaits are tested on the system for a new set of outputs.

5.4.7.4.4 Gait Generation via Genetic Algorithm

Once the kinematic models of the system are derived and defined, the set-up for the Genetic algorithm can take way. Gait, for this method, can be defined as a sequence of consequent steps where every following step is a derivative of the state of legs from the previous step. The state of each leg can be defined in terms of three angles from the three joints [58]. These three joints are put into a vector/matrix format to represent each leg and its current state. Each step of the system can then be modeled as the six states of the legs or 18 angles values in total. A gait can then be seen as the number of states/steps that it takes to complete the motion and repeat all over again. So, gait is a vector holding N number of steps. Each step holds 18 state values that define the position of the servo motor at that consequent step. The genetic algorithm is used in an attempt to find the optimal values for the states to have a walking mechanism that follows the modeled used for the ideal gait.

The genetic algorithm starts with a random generation and population of gaits. The gait's performance is then tested and simulated, then based on its performance from criteria we determine we can assign it a fitness value. This is done until the most optimal gaits for this generation are produced. Then "offspring"/children for the next generation of gaits are produced from those optimal gaits of the previous generation/parents in an attempt to find a stable optimal generation better than the previous generation. This process of testing, simulation, and reproducing is repeated until the most optimal gaits for our design is produced. This process model can be seen in the Figure 53: Genetic Algorithm Model below.

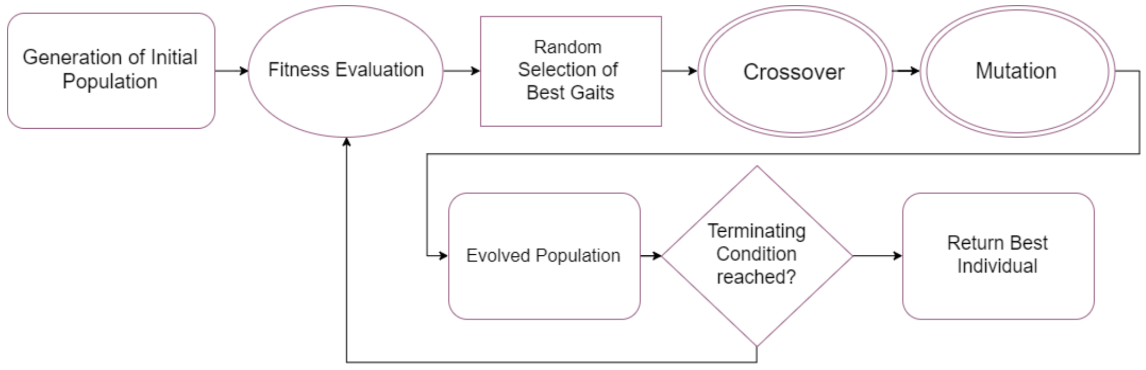

Generation of Initial Population
Fitness Evaluation
Random Selection of Best Gaits
Crossover
Mutation
Evolved Population
Terminating Condition reached?
Return Best Individual

Figure 53: Genetic Algorithm Model [58]

This method will be modeled following the paper: "Adaptive Gait Generation for Hexapod Robot using Genetic Algorithm" [58], that was published in the IEEE international conference paper on Power Electronics, Intelligent Control and Energy Systems.

5.4.7.4.5 Gait Generation via fuzzy algorithms

Once the kinematic models of the system are derived and defined, the set-up for the fuzzy algorithm can take place. Fuzzy logic is method that closely emulates the thinking process of natural human brains. It is also a way of mapping the input states to the output states of the defined system. This is done with computers by creating in-between values to the 0 and 1 or false and true paradigm that is implemented with systems. So, in the case of SigSent, instead of exact coordinates/joint angles generated or desired by the system, states of distance defined as very near (VN), near (N), far (F), and very far (VF) are created. Similarly, five different fuzzy values are assigned for both angle and deviation, namely left (LT), ahead left (AL), ahead (A), ahead right (AR), and right (R) [59]. This process of fuzzy logic definitions in turn is creating regions of position for the hexapod to follow. This creates a scenario where the hexapod robot can "understand" if it is accomplishing the goal of, for example, moving forward.

This "fuzzy algorithm" is then created with the foundation of fuzzy logic implemented on the different states of the legs on the hexapod robot and genetic population and modification of generations from Genetic algorithms. The model for this algorithm can be see below in Figure 54: Model of GA-Fuzzy Algorithm. From this figure it can be seen that the input is made up of states of the environment that are known and states of the robot that are known as well. This gets processed through the Fuzzy Logic Controller and produces an output. These outputs are then put through a genetic algorithm to process crossover and mutate to create the next generation of gaits and paths. The system is ran again until the tuning from the genetic algorithm has converged on an optimal gait and path

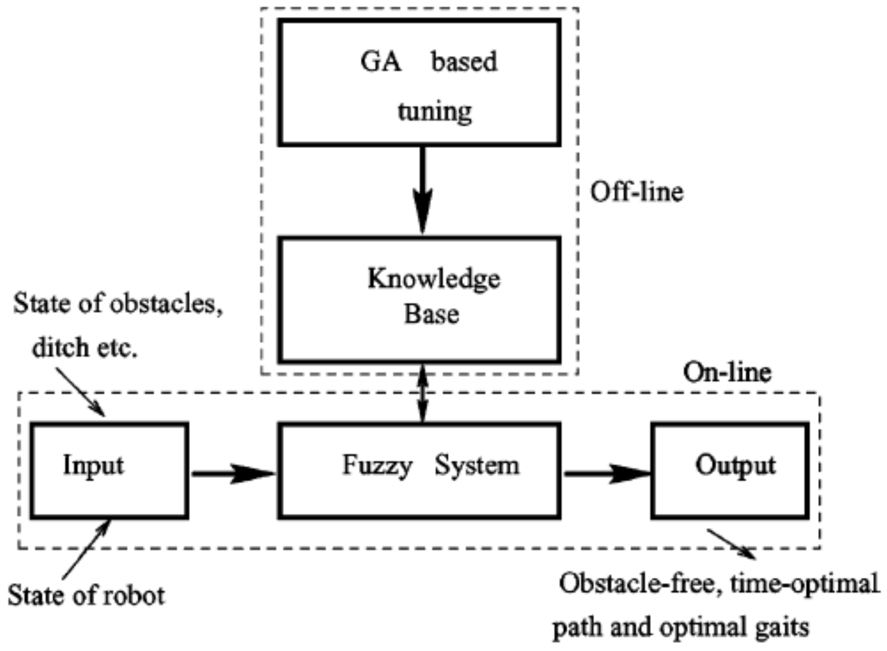


Figure 54: Model of GA-Fuzzy Algorithm [59]

This method, in all, creates optimal path and gaits generated by using fuzzy logic controllers and generic algorithms to find optimized fuzzy logic controllers that are then used to maneuver and manipulate the hexapod robot in test-case scenarios [59].

This method will be modeled following the paper: “Optimal path and gait generations simultaneously of a six-legged robot using a GA-fuzzy approach” [59], that was published in the Elsevier journal on Rob0tics and Autonomous Systems [59].

5.4.7.5 Open-Source Kinematic Tools

For basic simulation and testing there is a forum of community of creators around the Trossen Robotics that have some test programs and open source calculation tables to help with certain types of hexapods and crawling robots. MatLab’s community based forum also has quite a few tools available to the public from Mathworks themselves and from community creators that are sharing their projects with everyone else. The fundamental understanding of kinematics is necessary for navigation of these tools that are available to the community. These tools though only help with simple derivation of some kinematic characteristics for the robot.

For final, simulation, testing, and verification Gazebo should be used. This would put the algorithm through a test of various conditions to verify if the algorithm produced is sufficient to control the robot’s movements.

5.5 Mechanical Design

The mechanical design of SigSent is based on the hexapod robot design found common in many commercially available multi-terrain robots today with the addition of motorized wheels to four of its six legs. This allows SigSent to use its hexapod movement method across rough terrain or move through smoother surfaces efficiently in a traditional wheeled configuration.

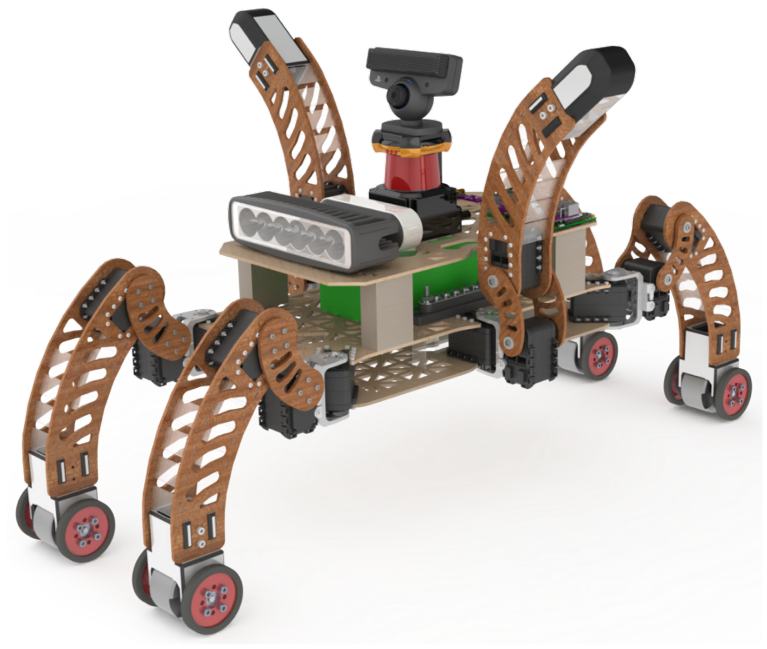


Figure 55: Rendering of SigSent in Wheeled Mode

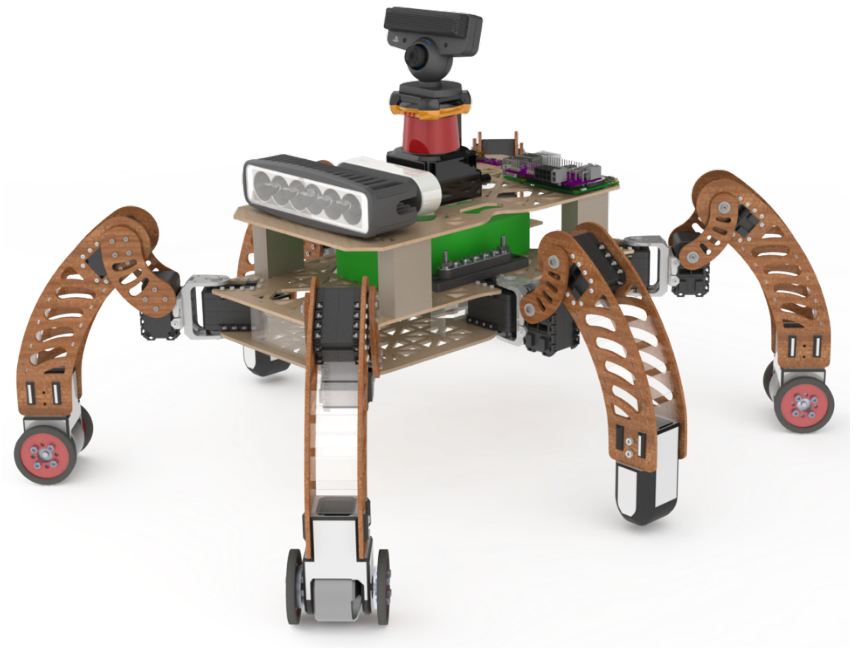


Figure 56: Rendering of SigSent in Terrain Mode

6 PROTOTYPING

6.1 SCHEMATIC

Schematics are produced for each module's PCB layout to plan out the overall design of the circuit.

6.2 PRINTED CIRCUIT BOARDS

SigSent features several printed circuit boards to accomplish the level functionality of the robot including system protection, signals transfer, and power transfer as well as a platform for some sensors and actors. Below are the designed printed circuit boards including figures and some rational of the design for that specific board.

6.2.1 PCB Design Considerations

While designing the PCB boards, special consideration was made to keep decoupling capacitors and charge pump capacitors as close to the IC's they were operating on as possible to reduce impedance.

6.2.2 Breadboard Test

Before proceeding with the final PCB design, a test was done with the components on a breadboard to verify that the microcomputer was working correctly with the sensors that it will be interfacing with in the PCB. The results from testing the sensors for valid output are shown in sections 7.1.5 and 7.1.6.

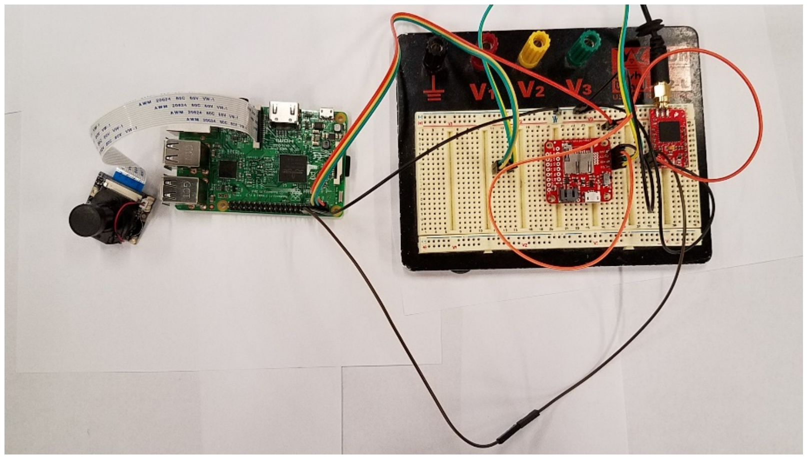


Figure 57 Breadboard test with Raspberry Pi microcomputer connected to GPS and IMU sensors

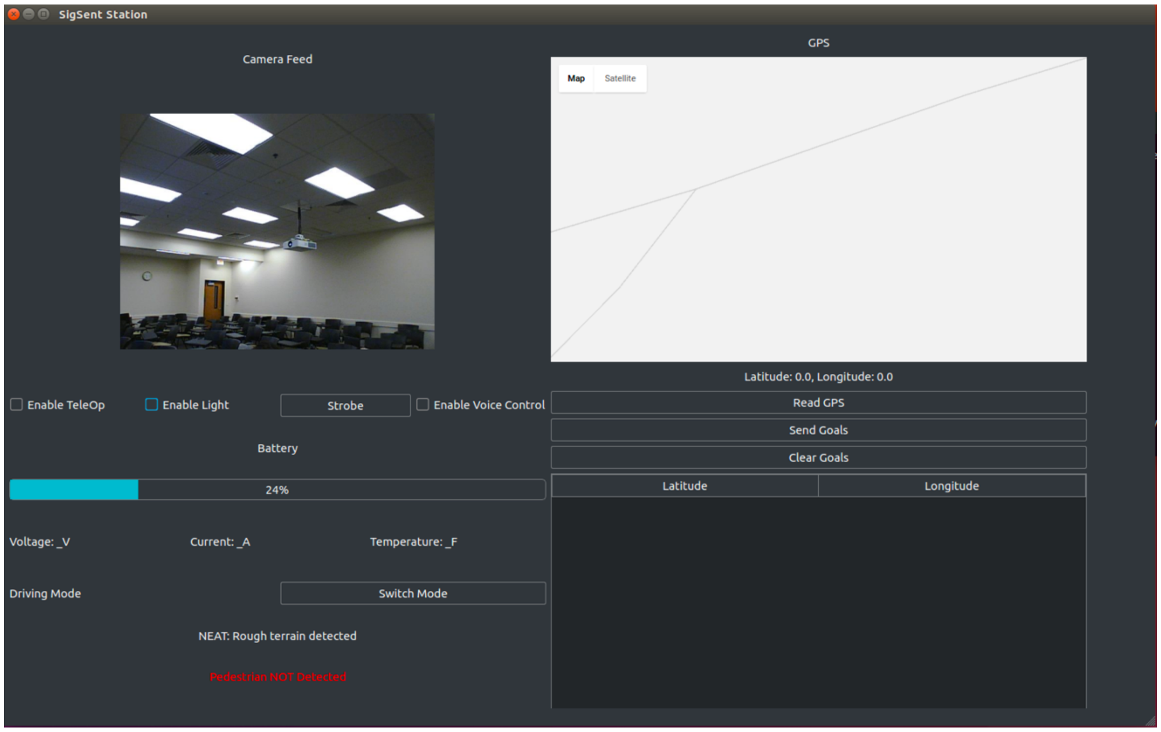
6.2.3 PCB Designs

Each PCB is specified and shown in 5.2 with its Diptrace layout and physical assembly boards.

6.2.4 PCB Fabrication and Assembly

OSH park and JLCPCB were used to fabricate the PCBs. To assemble the boards, the boards will be soldered on location within the Robotics Club at UCF laboratory. Techniques to create proper solder bonds will use a combination of hand soldering and hot-air reflow soldering depending on the specific package being soldered.

6.3 GUI



Latitude: 0.0, Longitude: 0.0

Voltage: _V Current: _A Temperature: _F

NEAT: Rough terrain detected

Pedestrian NOT Detected

Figure 58 Basestation GUI

The graphical user interface (GUI) features a live video feed, streaming from the SigSent robot. The GPS location is featured on an interactive map applet. There are buttons that the operator will use to interact with the robot. A radio button section is used to switch between the sentry and TeleOp movement modes of the robot. The battery level of the robot is shown as well to give a better idea on the estimated time remaining for operation. The GUI was developed with an educational/open-source Qt license with Python.

The top-left image view is a camera feed coming from the SigSent robot. Underneath the video feed are radio buttons for changing the operating mode of the robot. Sentry mode and TeleOp mode are the two main operations for SigSent. In Sentry mode, the robot will remain in a single position as an active video input. In TeleOp mode, the operator will be controlling the robot with their joystick that is plugged into the base station.

The map view on the top right will be using the GPS signal from the SigSent to give a fairly accurate location of the robot. A marker is placed to display the robot among the map's visual representation. Since the battery level is related to its travel, the estimated remaining battery life is displayed in a filled bar below the map. This information comes from the robot's "fuel gauge" sensors. Based on the remaining battery life and the location of the robot, the user will know how much farther they can travel without stranding their unit.

To aid in debugging the intelligent systems, a table is displayed underneath both visuals that display the terrain classification, IMU values, and the final NEAT output from the artificial neural network. Seeing the SigSent sensor values and intelligent systems output will ensure that the system is functioning correctly. Without a visual aid for this, it is hard to be sure that the system is being used at all, especially if the mobility mechanism is not switching (either due to a hardware failure or due to the ANN simply not outputting a new value as its been trained).

The GUI program will be built with a Python module, PyInstaller, that will allow it to be compiled into a native executable for Windows, macOS, and Linux operating systems. This means that changing base stations to communicate with the robot will be a painless process if the desired computer is not on-hand. The base station's computational requirements are not very heavy, so any computer should be usable for this process.

6.4 Prototype Expectations

The prototypes for the hardware and software components were made to test components and to demo component and code functionality. The prototypes were used to ensure that the hardware and software is functioning so that the real integration testing on the physical system could be put into place. Below are issues that could occur with the prototyping and some consistent errors that were experienced.

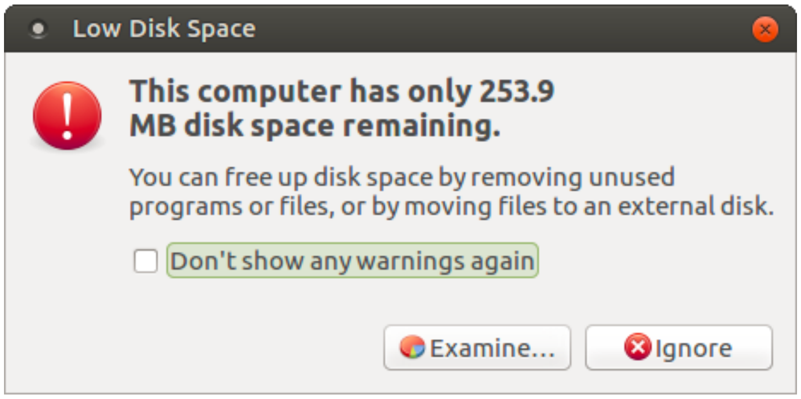
6.4.1 Potential Hardware Issues

There are several potential issues to worry about as far as the hardware is concerned. There are the obvious faults such as poor design of the circuit boards, designing boards that introduce impedance or create unacceptable EMI, the same thing holds for improper wiring. Improper Soldering could create intermittent problems that could be hard to diagnose – it will be very important to properly inspect each solder joint and ensure to check solder joints again if intermittent issues appear during testing and application. Other issues that might arise are overheating issues as the leg subassemblies each consume a considerable amount of power and could generate enough heat to potentially damage some control components in Florida's hot weather, especially since the robot operates outside. Beyond design issues for hardware there is the human to consider, we as imperfect being can cause significant problems even in a well-designed system (that does not account for human interaction). Proper case to ESD harden sensitive circuitry such as the GPIO pins on the Raspberry Pi, and to properly and visibly label the connectors that go into SigSent will be very important. Beyond things that will be interacted in normal operation, proper labeling and color code standards will need to be enforced uniformly throughout the construction of the vehicle to prevent human errors such as miss wiring a connector or plugging a connector into the wrong port. With proper design and training these issues should be minimized but will still always be an ever-present risk.

6.4.2 Potential Software Issues

Potential software issues included the obvious bugs that could be present in the code leading to faults. Running unit tests attempts to capture these errors before running everything in production. Every sensor and communication device needs to be set to use a constant address space or IP address identifier such that the code can be statically coded and work every time.

A common error that was occurring in Ubuntu MATE with the Raspberry Pi was a disk space error as the microcomputer was running low on available memory. Initially, an 8GB SD card was used in the Pi. After running into numerous disk space issues, it was upgraded to a 64GB card. All of the necessary libraries and frameworks being used (ROS, OpenCV, NEAT, etc.) required several gigabytes of storage each. Since most of the space needed to run SigSent is allocated before runtime, there should not be a high risk of reaching a disk space error in operation.



Low Disk Space

This computer has only 253.9 MB disk space remaining.

You can free up disk space by removing unused programs or files, or by moving files to an external disk.

Figure 59 Ubuntu MATE low disk space error

A potential issue in the intelligent systems module pertains to how well it is trained. If the robot is trained with a terrain type and is then placed in an environment radically different from its training data, it will not perform optimally. In some situations, the operating location can change quickly without warning, especially in some critical application where there is no time to retrain the robot (e.g. military applications). In this case, the robot would not operate correctly and could possibly cause harm to the people relying on the successful function of the robot. To avoid this issue as best as possible, the robot should be trained in a variety of locations. Since Florida, the host location of SigSent, is not varied in terrain types, small demoable environments should be recreated for small-scale training sessions. These should be sufficient for our operations, however for something more significant, extensive testing and training would need to be performed in real locations that closely resemble those of the places the robot will be actively used in.

7 TESTING

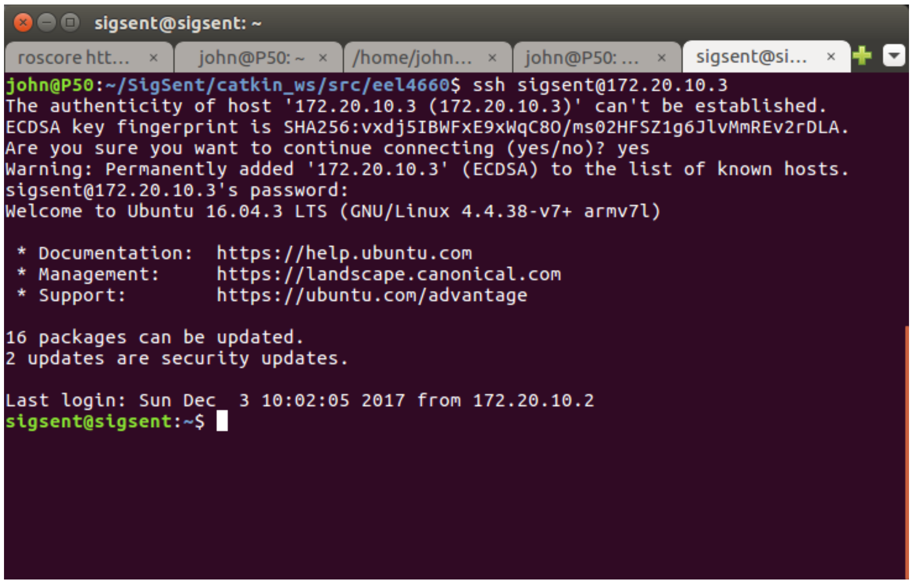
7.1 HARDWARE TESTING

The hardware components that were selected and purchased for use were tested according to their procedures listed below. Images corroborating the successful tests' results accompany the procedures' outlines.

7.1.1 Raspberry Pi 3 Microcomputer Testing

The Raspberry Pi 3 will be tested for being operational. It will be assumed to be working as intended if boot-up and an SSH connection has been established. Its complete functionality will be decided by the correct operation and communication between its sensors and miscellaneous connected components.

To further test the Raspberry Pi, we conducted a test of SSH using a local network and what will be our base computer. This is a simple test and allows for us to work remotely on SigSent without having to directly plug into the vehicle. This also allows for the vehicle to roam wirelessly while we monitor the vehicle through SSH without becoming tangled or unplugged as the robot moves around.



```
john@P50:~/SigSent/catkin_ws/src/eel4660$ ssh sigsent@172.20.10.3
The authenticity of host '172.20.10.3 (172.20.10.3)' can't be established.
ECDSA key fingerprint is SHA256:vxdj5IBWFxE9xWqC80/ms02HFSZ1g6JlvMmREv2rDLA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.20.10.3' (ECDSA) to the list of known hosts.
sigsent@172.20.10.3's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.38-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

16 packages can be updated.
2 updates are security updates.

Last login: Sun Dec  3 10:02:05 2017 from 172.20.10.2
sigsent@sigsent:~$ 
```

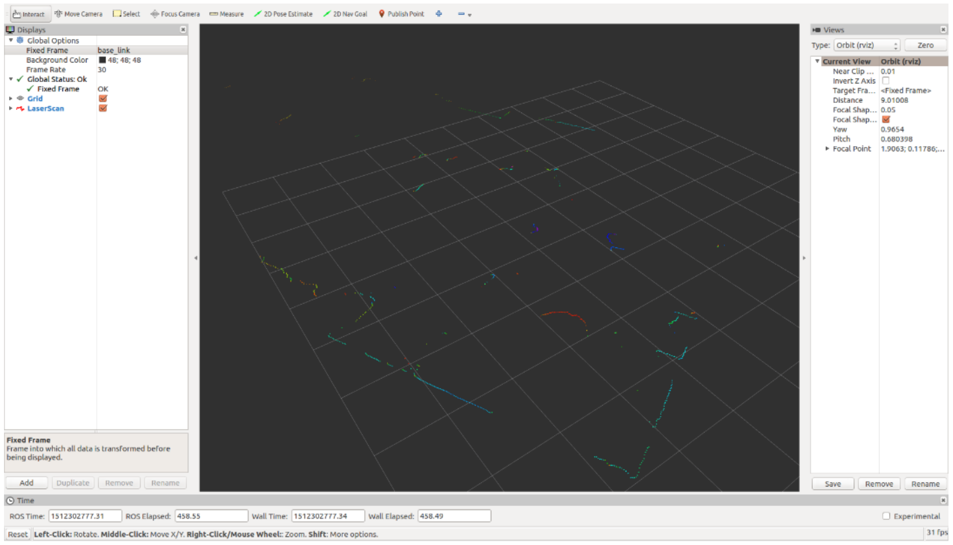
Figure 60: Screenshot of successful login of SSH over Wifi from a base station computer to SigSent's microcomputer

7.1.2 Microcontroller Testing

The microcontroller was assumed to be operating correctly based on its working functionality in integration with the robot's control systems. Signals will be sent to each joint's respective servo to move in 10-degree angle increments. If each motor is correctly accessed and moved with the necessary measured precision, the microcontroller was marked as testing successfully.

7.1.3 Lidar Testing

To test the lidar, which using serial communication through the ACM0 protocol we used a ROS node called URG_NODE which is specifically designed to interact with the Hokuyo brand of lidar sensors including our UTM-30LX. URG_NODE reads in the sensor data and reformats it into the standardized ROS message named laser_scan in the topic /scan. From there we both visualize and confirm the relative accuracy of the Lidar data in RVIZ compared to what we are seeing and we can view the raw data by echo'ing the rostopic /scan in the Command Line Interface (CLI).



Global Options
Fixed Frame base_link
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid ✓
LaserScan ✓

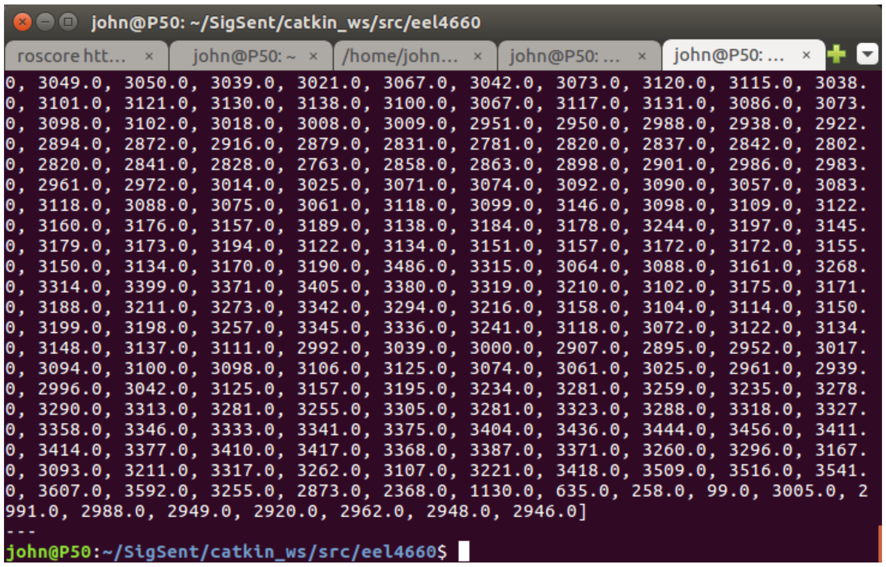
Fixed Frame
Frame into which all data is transformed before being displayed.

Type: Orbit (rviz) Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis ☐
Target Fra... <Fixed Frame>
Distance 9.01008
Focal Shap... 0.05
Focal Shap... ✓
Yaw 0.9654
Pitch 0.680398
Focal Point 1.9063; 0.11786;...

ROS Time: 1512302777.31 ROS Elapsed: 458.55 Wall Time: 1512302777.34 Wall Elapsed: 458.49 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options.

Figure 61: Visualization of Lidar Data in RVIZ



```
0, 3049.0, 3050.0, 3039.0, 3021.0, 3067.0, 3042.0, 3073.0, 3120.0, 3115.0, 3038.
0, 3101.0, 3121.0, 3130.0, 3138.0, 3100.0, 3067.0, 3117.0, 3131.0, 3086.0, 3073.
0, 3098.0, 3102.0, 3018.0, 3008.0, 3009.0, 2951.0, 2950.0, 2988.0, 2938.0, 2922.
0, 2894.0, 2872.0, 2916.0, 2879.0, 2831.0, 2781.0, 2820.0, 2837.0, 2842.0, 2802.
0, 2820.0, 2841.0, 2828.0, 2763.0, 2858.0, 2863.0, 2898.0, 2901.0, 2986.0, 2983.
0, 2961.0, 2972.0, 3014.0, 3025.0, 3071.0, 3074.0, 3092.0, 3090.0, 3057.0, 3083.
0, 3118.0, 3088.0, 3075.0, 3061.0, 3118.0, 3099.0, 3146.0, 3098.0, 3109.0, 3122.
0, 3160.0, 3176.0, 3157.0, 3189.0, 3138.0, 3184.0, 3178.0, 3244.0, 3197.0, 3145.
0, 3179.0, 3173.0, 3194.0, 3122.0, 3134.0, 3151.0, 3157.0, 3172.0, 3172.0, 3155.
0, 3150.0, 3134.0, 3170.0, 3190.0, 3486.0, 3315.0, 3064.0, 3088.0, 3161.0, 3268.
0, 3314.0, 3399.0, 3371.0, 3405.0, 3380.0, 3319.0, 3210.0, 3102.0, 3175.0, 3171.
0, 3188.0, 3211.0, 3273.0, 3342.0, 3294.0, 3216.0, 3158.0, 3104.0, 3114.0, 3150.
0, 3199.0, 3198.0, 3257.0, 3345.0, 3336.0, 3241.0, 3118.0, 3072.0, 3122.0, 3134.
0, 3148.0, 3137.0, 3111.0, 2992.0, 3039.0, 3000.0, 2907.0, 2895.0, 2952.0, 3017.
0, 3094.0, 3100.0, 3098.0, 3106.0, 3125.0, 3074.0, 3061.0, 3025.0, 2961.0, 2939.
0, 2996.0, 3042.0, 3125.0, 3157.0, 3195.0, 3234.0, 3281.0, 3259.0, 3235.0, 3278.
0, 3290.0, 3313.0, 3281.0, 3255.0, 3305.0, 3281.0, 3323.0, 3288.0, 3318.0, 3327.
0, 3358.0, 3346.0, 3333.0, 3341.0, 3375.0, 3404.0, 3436.0, 3444.0, 3456.0, 3411.
0, 3414.0, 3377.0, 3410.0, 3417.0, 3368.0, 3387.0, 3371.0, 3260.0, 3296.0, 3167.
0, 3093.0, 3211.0, 3317.0, 3262.0, 3107.0, 3221.0, 3418.0, 3509.0, 3516.0, 3541.
0, 3607.0, 3592.0, 3255.0, 2873.0, 2368.0, 1130.0, 635.0, 258.0, 99.0, 3005.0, 2
991.0, 2988.0, 2949.0, 2920.0, 2962.0, 2948.0, 2946.0]
---
john@P50:~/SigSent/catkin_ws/src/eel4660$ 
```

Figure 62: Raw Lidar data echo'd from the ROS topic /scan

7.1.4 Camera Testing

The camera was connected to the Raspberry Pi and a basic image was captured indoors in a low-light environment. Even with suboptimal conditions for lighting, the image was still captured at an acceptable quality. SigSent will be operating outside. As such, daytime sunlight will provide more than enough lighting for basic sentry operations and surveillance capture. During its nightly patrols, the powerful LED light bar mounted on the robot will illuminate the area in front of the camera's field of view. Below, in Figure 63 Camera test indoors in low-light, clear colors, contrast, and sufficient optical quality are seen. Highlights are overblown in the ceiling lights, however this is normal for a low-light scenario to compensate for the poor illumination throughout the laboratory room. The camera adjusts its exposure settings as such to provide an average normal exposure that is viewable in bright and dimly lit environments. The tests provided showcase that the camera will be acceptable for the minimal computer vision computations we will be doing for terrain classification and basic human/anomaly detection.

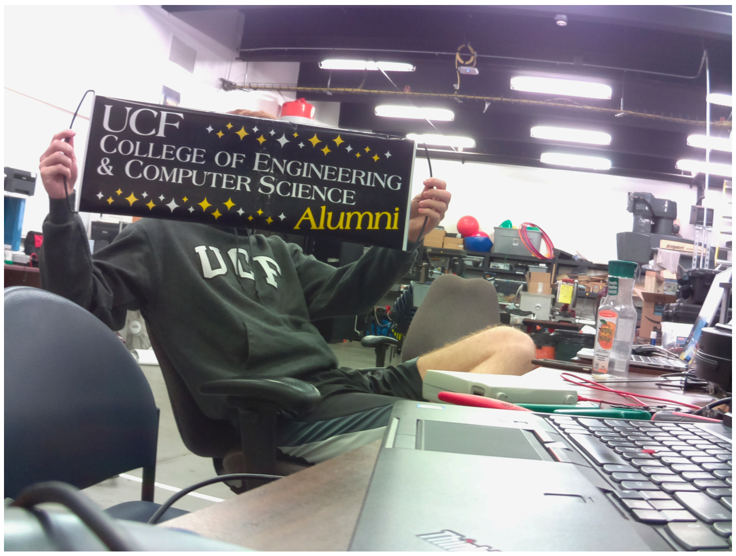


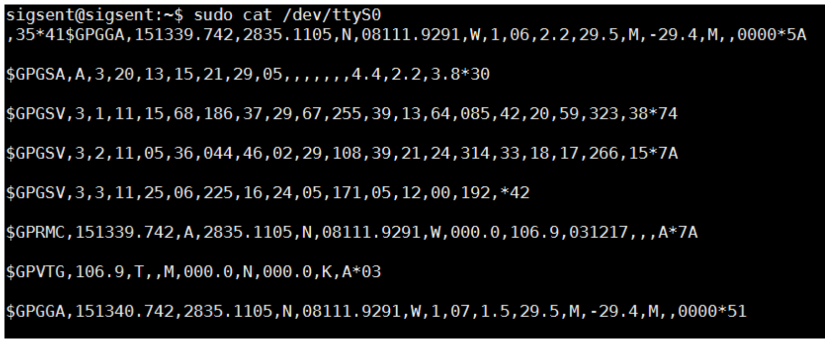
Figure 63 Camera test indoors in low-light

7.1.5 IMU Testing

IMU Testing for the MPU-9250 was conducted by hooking up the sensor and reading its values through serial in Linux and confirming that they changed logically as we moved and rotated the sensor around. Once we have the robot fully assembled more in-depth tests can be made, however to simply confirm that the sensor is outputting data in the expected fashion this test was proved successful.

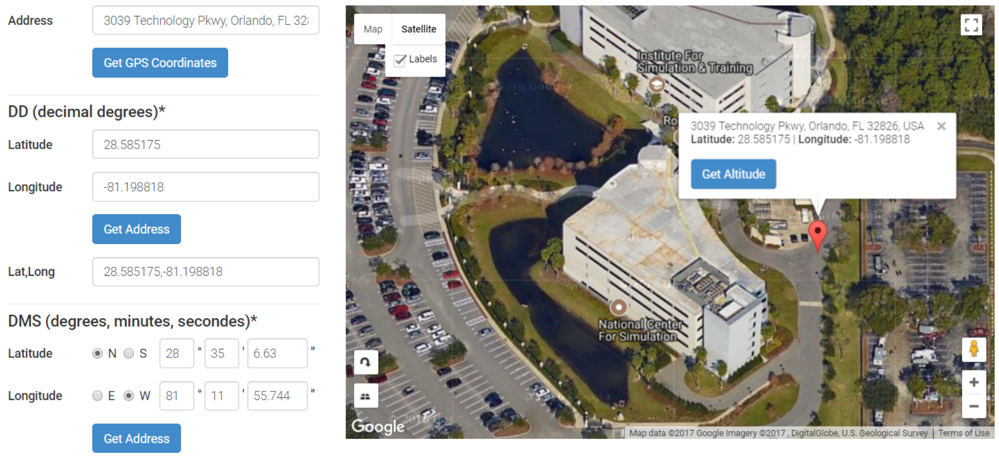
7.1.6 GPS Testing

GPS Testing for the Venus 638FLPL was conducted at UCF's Partnership II and III campus. The testing process was to bring the GPS unit outside hooked up the Raspberry Pi 3 through a TTL serial feed. The GPS outputs a constant stream of NMEA GPS data which we then parsed into to standard GPS coordinates, which we then fed into Google Maps to confirm that the GPS points were accurate. We found that the GPS points were very accurate outside and confirmed its location on Google Maps to well within the specified 2.5 meters of stated accuracy. We also found that indoors the GPS did not perform well and often was not able to contact enough satellites to output a valid GPS point. Since the vehicle will be outside this is considered a non-issue, but may present problems during presentations and demos if conducted indoors.



```
sigsent@sigsent:~$ sudo cat /dev/ttyS0
,35*41$GPGGA,151339.742,2835.1105,N,08111.9291,W,1,06,2.2,29.5,M,-29.4,M,,0000*5A

$GPGSA,A,3,20,13,15,21,29,05,,,,,,,4.4,2.2,3.8*30

$GPGSV,3,1,11,15,68,186,37,29,67,255,39,13,64,085,42,20,59,323,38*74

$GPGSV,3,2,11,05,36,044,46,02,29,108,39,21,24,314,33,18,17,266,15*7A

$GPGSV,3,3,11,25,06,225,16,24,05,171,05,12,00,192,*42

$GPRMC,151339.742,A,2835.1105,N,08111.9291,W,000.0,106.9,031217,,,A*7A

$GPVTG,106.9,T,,M,000.0,N,000.0,K,A*03

$GPGGA,151340.742,2835.1105,N,08111.9291,W,1,07,1.5,29.5,M,-29.4,M,,0000*51
```

Figure 64: Testing output of NMEA GPS data from GPS Unit



Address: 3039 Technology Pkwy, Orlando, FL 32

Get GPS Coordinates

DD (decimal degrees)*

Latitude: 28.585175

Longitude: -81.198818

Get Address

Lat,Long: 28.585175,-81.198818

DMS (degrees, minutes, secondes)*

Latitude: [✓ N] S 28 ° 35 ' 6.63 "

Longitude: E [✓ W] 81 ° 11 ' 55.744 "

Get Address

3039 Technology Pkwy, Orlando, FL 32826, USA
Latitude: 28.585175 | **Longitude:** -81.198818

Map
Satellite
Labels
Institute For Simulation & Training
Get Altitude
National Center For Simulation
Google
Map data ©2017 Google Imagery ©2017 , DigitalGlobe, U.S. Geological Survey
Terms of Use

Figure 65: Confirming accuracy of GPS data by placing GPS coordinates into Google Maps [60]

7.1.7 Servo motor Testing

The servo motor characteristics of stall torque, stall current and weight will be measured and tested. To test the stall torque characteristic of the servo motor a known weight on a cantilever at a set(adjustable) and known distance will be put on the servo motor. The servo motor will then be commanded to rotate with increasing distances applied to the weight up to create increasing torques on the servo motor. This will be done until the known stall torque has been reached or a smaller stall torque has been reached. The stall current will be measured and tested during the testing of the stall torque by actively measuring the current applied to the servo motor at the time of testing. A fuse, or emergency stop will be implemented to ensure no damage to the servo motor when stall has been reached. The weight of the servo motor will also be measured to ensure the proper product within design has been obtained.

7.1.8 Motor and ESC Testing

A tachometer will be used to measure the RPM of the motor to determine if it meets the specifications required by the robot and as notated in its datasheet. The motors will be run using their standard operating voltage. If the motors reach the necessary speed, the testing will be determined as successful.

7.1.9 Battery Testing

The battery's capacity will be tested by measuring the time it takes for the battery to "die" while under a predefined load. The battery will be charged until the voltage across its terminals reads 16.8 V with a multimeter. A stopwatch will engage when a 10 A constant current dummy load is first attached to the battery's terminals and will stop when the voltage across the terminals measures 13.6 V with a multimeter. To be considered minimally successful, the time must be greater than or equal to 1 hour.

7.1.10 Speaker and Amplifier Testing

Test tones at various frequencies within the voiceband will be played from the speaker system at maximum volume, and will be measured with a sound level meter from 10 meters away. At minimum, the test will be considered successful if the meter reads a level greater than 60 dB for a 4 kHz test tone. Human hearing is typically most sensitive near this frequency [61], which makes it suitable for use in a siren.

The utility of the speaker system would be further validated with sound level measurements significantly greater than 60 dB across the voiceband. Additionally, reliable comprehension of vocal instructions 10 meters from the speaker system would demonstrate full success of the audio system design.

7.1.11 Microphone Testing

To test the microphone being used on the robot for basic operational situations, it will be connected to the Raspberry Pi microcomputer and debugged in Ubuntu Mate for proper feedback. If the operating system's audio manager cannot receive input from the microphone, it will need to be debugged. If the microphone interfaces correctly, it will be successfully tested. Its specifications for recording were known through the hardware research and further testing on these parameters is unnecessary if the operation of the device and its integration goes smoothly.

7.1.12 Lighting System Testing

The lighting system will be tested for sufficient illumination and operating times. A light meter is used to capture how much light is being output by the system. The lighting system must provide the illumination specified by the requirements in section 4.3.17.1. The lighting system will be run for two hours to verify that it is still operational after a long duration. The operational time is meant to test if the lighting system can reliably operate for the entire duration of the robot's battery life. Two hours is much higher than the upper-bounds given in the requirements section: 4.3.13.3. If it lasts two hours without any issues, it will be decided to be sufficient.

7.1.13 Power System Testing

All components need sufficient power to turn on and operate. The voltage and current at each node for a component will be tested such that each component is receiving the correct amount for operation. Incorrectly delivering the wrong values could damage our components. After testing that everything is under the correct operating constraints, the components will be tested for correct functionality.

7.1.14 Signal Protection System Testing

To test the Protection System we will purposefully introduce ESD and transients events into signals lines while measuring with an oscilloscope and ensure that the various Zener diode clamps properly clamp those events.

To test reverse circuit protection, we will reverse the power input to SigSent with all vulnerable devices removed and check that the reverse current does not pass through the protection method, this will be checked via a multimeter.

To test overcurrent protection, a fuse will be hooked up to a battery with high power low ohm resistors to create an overcurrent event with the fuse inline, the circuit will be closed creating the overcurrent event and the event timed to ensure that the fuse opens at the expected time.

7.1.15 Base Station Testing

The base station testing encompassed testing that the computer is working as well as the GUI program that communicates with the robot. The computer was booted up and verified that it is functional. The base station's GUI program was launched to verify that it is able to correctly communicate with the SigSent robot. With the robot able to be controlled from the base station, the testing was marked as successful.

The peripherals for the base station will need to be tested in conjunction with the laptop as well. The joystick will be tested in the Windows Joysticks tool, where inputs can be shown as being received by the operating system. If the joystick has been correctly installed to work with the OS, then it will be tested in the base station's GUI program for correctly controlling the SigSent robot in TeleOp mode. The headset used for listening and vocally communicating through the robot will be tested as well. The default Windows playback/recording devices menu will be used to ensure that the operating system is receiving input and producing output through the headset device. The GUI program will then be run as well to test if the headset is receiving audio from SigSent and also outputting audio from the microphone through the robot's speaker.

7.2 Software Testing

The software developed for SigSent must be rigorously tested in modular unit tests, in a simulated environment, as well as in the actual physical environment. The procedures to facilitate the testing are notated below.

7.2.1 Software Testing Overview

Testing is a necessary and important part of any software development lifecycle. Before code can pushed into a production environment, especially in a mission critical application, testing can be done to prevent errors from making it into a release. When working with physical hardware, where real harm can be done due to software mistakes, testing is very important. We will be employing a variety of techniques and frameworks discussed below to ensure that human faults do not lead to significant failures, or unexpected results. The unit tests will be done using Python's included `unit test` library. Tests can be written to verify that every function returns the exact values that it is expected to. Edge cases can be easily tested in this method by calling each individual function with these obscure situations to check that they are functioning correctly. Integration tests are then run that combine modules of the software environment together to verify that the many software components still function as expected when working together. Integration testing will ensure that as new features are added, the overall quality of the SigSent's performance is constant.

7.2.2 Simulated Testing

Gazebo can be used as a software solution for testing the ROS control systems code, the movement and gait generation algorithms, as well as the basic neuroevolution artificial intelligence work. The artificial intelligence can be run in a simulated environment; however, it must be used with caution as the data gathered in this way will not be accurate, or at least cannot be depended on. The simulation testing is to ensure that after each individual function and module has been testing in isolation and integration, that the robot as a whole works as intended. The artificial intelligence can only be tested for validation in the full simulated Gazebo environment and in the physical testing process.

In Gazebo, a sample environment consisting of varying terrains for smooth and rough surfaces will be used to test SigSent in each of its possible operating conditions. Testing in a simulated environment, while not completely accurate in modeling true physical performance, allows for quick iterations in different locales. Obstacles can be added and removed as necessary to provide SigSent with a comprehensive setting for testing. As testing is done, the software can be modified as needed. The NEAT learning platform will be under the most scrutiny throughout the testing, as the training and neural network parameters are modified to produce the best possible results.

7.2.3 Physical Testing

After completing a successful iteration in each testing environment listed above, the software was tested on the physical robot. Because of the physical limitations of the simulation not being achieved, the software was modified for usage in tests with the actual SigSent robot. Parameters were modified in the code pertaining to the movement to have the best performance. Changes were quickly made and re-uploaded to the robot to test in the physical environment until an acceptable performance is achieved. The base station GUI was the most helpful throughout the physical testing process as it provides real-time updates on the robot's sensor values and how it pertains to the intelligent systems' artificial intelligence software modules.

7.3 TESTING PLATFORM

In order for our team to work in parallel while the mechanical system and control systems are being developed we are using the Clear Path Robotics Turtlebot 2 supplied by the Robotics Club at UCF to test the software and hardware that does not rely on our movement systems in a physical model. The Turtlebot is a simplistic framework of a robot that has a controllable base based off the Kobuki/Roomba platform and is designed specifically to be ideal for testing robotics with many mounting points and an easy interface to control with.

On the Turtlebot we tested our lidar, IMU, GPS, and camera as well as our power and protection systems as a complete system on the Turtlebot using the state machine with GPS waypoint navigation and human detection. Once we had SigSent's chassis and control system ready, we migrated all the systems being tested on the Turtlebot over to SigSent's chassis for further testing.

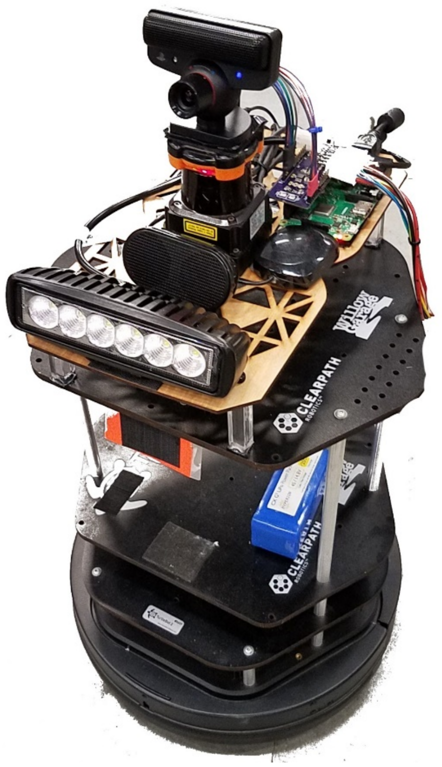


Figure 66: Picture of Turtlebot equipped with several of our sensors in anticipation of testing

7.4 Finished Prototype

The finished, working prototype is detailed below. SigSent went through three rounds of renders before we arrived at the finalized build.

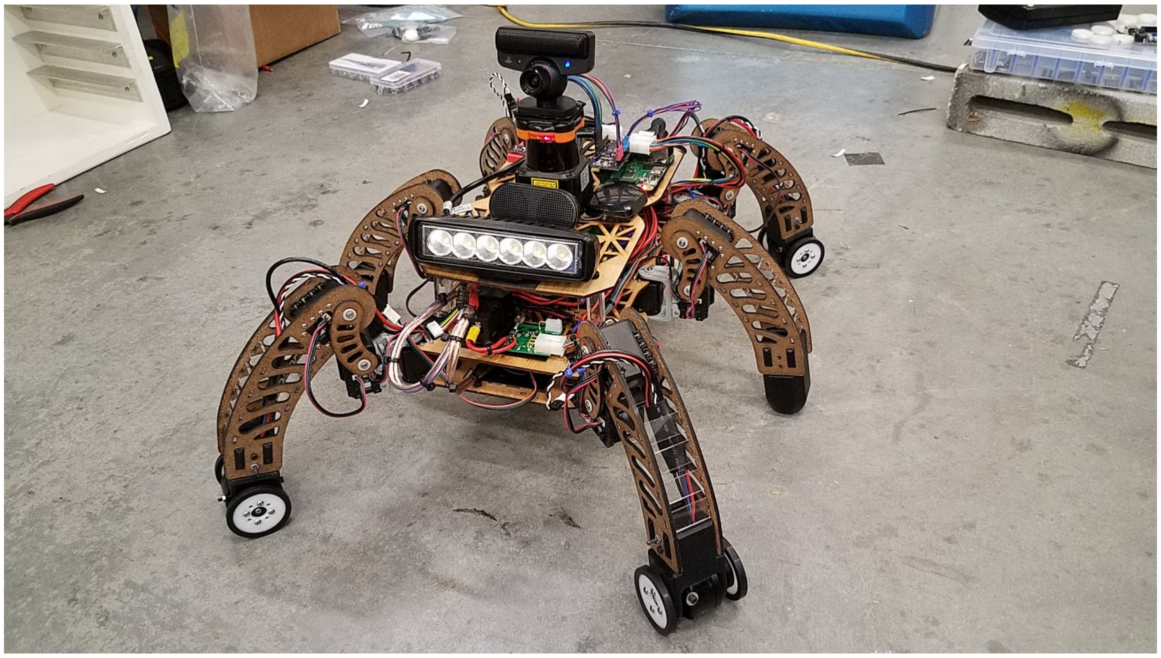


Figure 67 Finalized SigSent Prototype

7.5 Operating SigSent

To operate SigSent, the user needs to only plug in the battery to the nearby Anderson Powerpole connector. After the Raspberry Pi is done booting up (which can be confirmed by pinging the Pi's IP address, which is set to be a static 192.168.1.101), the user can launch the ROS nodes necessary for use. These ROS nodes need to be built beforehand. The user can clone our code's repository, install the necessary dependencies displayed in the repo, and then run catkin_make in the code's workspace directory titled catkin_ws. Source the new bash file, devel/setup.bash, to add the ROS packages to the bash path, and roslaunch the launch file with roslaunch sigsent sigsent.launch. This launch file will launch everything needed to run SigSent locally, except the GUI, and will also SSH into the Raspberry Pi to launch the nodes for the robot's hardware.

Launch the GUI next. To do so, install the Python dependencies by running a simple pip install -r requirements.txt when in the main directory of the code's repository. Change directory into the folder named GUI and run the PyQt script, python gui_test.py. This will launch the graphical application and will automatically connect to the ROS topics necessary if the machines have been setup on the network correctly. The ROS wiki has information to meet any potential issues that may arise when setting up the networking.

8 ADMINISTRATIVE CONTENT

8.1 SOFTWARE TOOLS

In working on SigSent, several software tools were used for communicating between team members, developing the software/hardware implementations for the robot, and also in documenting the project. Each tool was essential in reaching our goals in developing and documenting SigSent.

8.1.1 Communication

Communication between team members is important to provide the best working environment. Being able to quickly get in contact with each other for meetings, to make ad-hoc design decisions, and working remotely was necessary for the tight deadlines we had to meet. The tools below encouraged this communicative process.

8.1.1.1 Discord

To foster better communication, our group used Discord. Using Discord, we were able to easily get updated information on everyone's progress, as well as share useful links or images related to the project's research. Our group had its own "server" that we could connect to, containing voice and text channels to communicate through. We utilized the voice chats during remote meetings when we could not meet physically. The biggest advantage to using Discord was that it featured this powerful communication platform for free. Slack, another free communication tool, does not feature the same rich, low latency voice connection as does Discord.

8.1.2 Development

The development of SigSent for both the software and hardware was achieved with the assistance of high-level software tools. Each tool used is listed below with a brief summary on its working purpose and why it was chosen for use in the SigSent project.

8.1.2.1 MatLab

Used for computation of the Inverse and Forward kinematics of the hexapods movements for each individual leg. As well as used for the gait generation of the hexapod for normal movement patterns. With its multiple different library sets and use of the online community forum, the modeling of the kinematics was made simple and straight forward. Through the ease of matrix manipulations, function definitions, implementation of algorithms and interfacing with multiple other language sets, MatLab remains one of the best multi-paradigm numerical computing environments available to the public. For initial testing, MatLab also offers interfacing with hardware, that allows for testing with hardware out and in of the loop to test efficiency, accuracy, and feasibility in the system. The MatLab software is able to interface with the Raspberry Pi and various microcontrollers as well, which allows for simulation and testing on the overall systems or each sub-system individually.

8.1.2.2 PyCharm

PyCharm features plugins for use with ROS and the Raspberry Pi, making Python development on our platform easy to scaffold out. Although PyCharm is a commercial product made by the company JetBrains, they offer a free account for students, as well as free community editions of some of their IDEs (including PyCharm). The ROS plugin for PyCharm includes support for packages, code execution with roslaunch, node debugging, unit test execution and debugging, and custom message/service creation. The Raspberry Pi support is not via a direct plugin, however PyCharm features exceptional SSH and tunneling support to write code on a local machine and simply execute it over on the remote device, as well as the ability to connect to local database solutions running on the microcomputer. The neuroevolution and ROS modules were written in Python, making PyCharm the obvious choice of IDE for our software development.

8.1.2.3 DipTrace

Used for PCB design, DipTrace is a software suite for creating schematics and PCB design as well as 3D visualization and 3D file exporting completed board with component representation. This software was provided by the Robotics Club at the University of Central Florida [43] and DipTrace [62]. DipTrace has an intuitive and quick UI/UX that allows for rapid development of hardware.

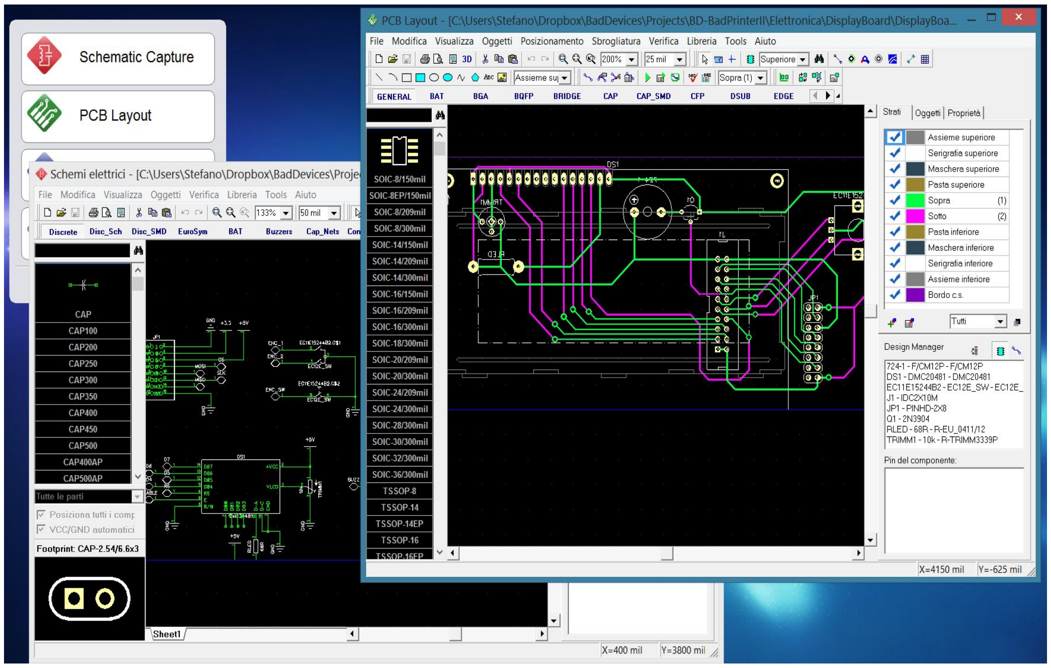


Figure 68: Representation of Diptrace's Schematic Capture and PCB Design applications [63]

8.1.2.4 Git

In the world of version control systems (VCS), Git reigns supreme in popularity. In 2016, *RhodeCode* did a study on VCS interest by reviewing their presence in Google search trends. Git comprised 70% of searches compared to other systems, including (in descending order of search significance) Subversion (SVN), Mercurial, Perforce, and Concurrent Version Systems (CVS) [31]. We used Git as our VCS due to its obvious preference by developers, and previous experience that our group had with using it. Using Git, changes are tracked between the local files and the last committed changes to the repository. A commit essentially is a state in the Git graph that holds the exact version of every file in the repository, backed by a checksum to ensure that nothing differs from exactly what was saved by the user. Branches can be created as well, allowing specific tasks to be implemented in their own state, not interfering with the main master branch. The code added to the other branches can then be cherry picked over into the main branch by their commit hash. Git is also much faster when juxtaposed with performances of similar commands on competing other version control systems. Git is a free and open source software solution, released under the GNU GPL, open source license.

8.1.3 Documentation

The documentation for the project is all saved in in the cloud, but is maintained across multiple websites that provide tools for each type of document.

8.1.3.1 Google Drive/Docs

Google Drive is where every file, excluding the source code and CAD documents, were stored. Google Drive also includes a web-based document editor via Google Docs. Google Docs also includes an editor for excel spreadsheets that we used to document part specifications and comparisons. Google Docs is where the project's documentation was written collaboratively in the cloud. Our shared folder also acted as a remote backup for our documents for higher availability and reliability using a free solution.

8.1.3.2 Draw.IO

Most of the flowcharts and diagrams that we created were done with Draw.IO's editor. They have seamless integration with Google Drive which made synchronization and collaboration easy. Draw.IO includes templates for many popular types of documentation diagrams. For the software class diagrams and other UML documentation figures, we were able to take advantage of Draw.IO's handy pre-built blocks that conform to UML specifications. The class diagrams were essential to our development process to properly plan our architecture prior to actually implementing our ideas. Draw.IO improved our project's scalability by keeping us focused on the big-picture overarching design such that we did not have to waste time refactoring code as the project grew in number of features.

8.1.3.3 Microsoft Word SharePoint Document

To increase the readability of our document, Microsoft Word's shared collaborative environment was utilized for its powerful document editing tools (with automatic table of contents for tables and figures, and automated bibliography generation for citation references). Google Drive is a safe way to save all of our documents in one cloud-based storage solution, however Google Docs does not have the rich editing environment that Microsoft Word offers. When used in conjunction with our other documentation tools, we have created a streamlined, effective workflow for writing.

8.2 Division of Labor

The division of labor between each group member can be more clearly seen in the block diagrams for both the hardware and software components from Figure 28 and Figure 35. The milestones in 8.3 also outline each group member's primary responsibilities and secondary responsibilities with defined deadlines. These deadlines were set-up into phases (Phase 1 – Phase 3) These deadlines were subject to change relative to the progress along the project timeline.

Throughout development, the deadlines were heavily shifted as hardware acquisition and fabrication took longer than anticipated. Most of the main goals were met in the late months of the semester.

8.3 Project Milestones

	Due 11/30/2017	John Millner	Josh Franco	Jeff Strange	Richie Wales
Phase 1	Primary Goal	Mechanical Design & Physical Creation	Control System Legs (Sim)	Power System Design	Simulation Creation
	Backup Goal	Simulation Creation	Power System Design	Mech Design & Creation	Control System Legs (Sim)
	Tasks	Complete Laser Cut Design	Kinematic modeling of movement/legs	Confirm sensor selection	Setup VCS
		Create Laser Cut Model	Kinematic modeling of whole body	Calculate power & energy needs	Familiarize with Gazebo
		Sponsorships/Discounts	Input/feedback data for closed loop	Design schematic	Create basic sim environment
		Order Parts	Movement pattern for different terrain	Find Primary Source Components	Move model SDF from Solidworks
		Wire Management	Familiarize with Gazebo	Find Redundant Sources	Add necessary ROS connections to moving parts

		Complete 3D Printed Design	Get necessary inputs for simulation	Create BOM	Follow ROS turtlesim docs
		Create 3D Printed Model		Determine wire routing	Implement code to move robot
		Have Complete Platform		Order Parts	Test/Debug

	Due 1/31/2017	John Millner	Josh Franco	Jeff Strange	Richie Wales
Phase 2	Primary Goal	ROS Integration	Control System Active Suspension on Robot	Communications	Working on ML on Sim
	Backup Goal	Communications	Working on ML on Sim	Control System Active Suspension on Robot	ROS Sensor Integration
	Tasks	Create/find Packages-nodes-publishers for each sensor	Design active suspension implementations	Design Base Station radio system	Research potential computers/MCUs
		Create state machine for different modes	Choose optimal designs for AS	Design robot radio system	Seek NEAT advising (Dr. Wu/Dr. Stanley)
			Input/Feedback from AS included in control	Implement messaging framework from ROS	Run small-scale NEAT tests without all sensors
			Implement AS in control simulation	Implement basestation command line client	Use sensors from John's implementation
				Test connectivity in various environments	Modify parameters and re-run

	Due 2/28/2017	John Millner	Josh Franco	Jeff Strange	Richie Wales
Phase 3	Primary Goal	ROS Path Planning	Polishing Control Systems	Teleop Control	Neat on the Robot

	Backup Goal	Teleop Control	Neat on the Robot	ROS Path Planning	Polishing Control Systems
	Tasks	Create local path planning for legs	Results from Control Sim	Design base station controller	Run NEAT on robot
		Robot must avoid obstacles	Test on different terrains	Design base station GUI	Seek advising on inevitable failures
		robot must path around obstacles on a global goal	Test movement abilities from Sim	Implement video stream	Re-run training until viable result
		robot must be able to move to a GPS Waypoint	Optimize Control Systems	Test, test, test	Train for longer duration (TBD)
		robot must take goal vectors			Save best resulting ANNs

8.4 BUDGET AND FINANCE

Table 25 Initial Budget

Part Number	Description	Unit Price ($)*	Total Quantity	Total Price ($)*
244000083-0	Motor	18.99	6	113.94
FUTM0043	Servo motor	22.99	18	413.82
595711	Wheel	1.995	6	11.97
57155K383	Bearing	6.42	6	38.52
92775A106	Shaft Set Screw	0.3476	12	4.1712
91292A015	Motor Screws	0.218	24	5.232
92290A474	servo motor horn screws	0.78	72	56.16
98511A300	Wheel Screws	0.841	24	20.184
91292A116	Servo motor Screws	0.0641	72	4.6152

91854A101	Servo motor Nuts	0.1296	72	9.3312
N/A	Custom 3D Leg Prints	80	1	80
N/A	Custom 3D Abdomen Print	65	1	65
3100	Camera	29.99	1	29.99
3055	Microcontroller	35	1	35
Z50003S-25	Battery	25.38	1	25.38
9192000310-0	ESC	10.53	4	42.12
VN-200	IMU / GPS	2600	1	N/A
GF0876	Speaker	5.02	1	5.02
TS4962IQT	Audio Amplifier	0.99	1	0.99
URG-04LX-UG01	Lidar	4800	1	N/A
	PCB Fabrication	89.2	N/A	89.2
	Misc. Replacements	12.61	N/A	12.61
	DigiKey BOM	295.22	N/A	295.22
	Robot Communications	On-hand	1	N/A
	Basestation Communications	On-hand	1	N/A
	Laptop	On-hand	1	N/A
	Servo motor Controller Board	50	1	50
	Power Distribution Board	50	1	50
	Logitech Flight Stick	On-hand	1	N/A
	SD Memory Card	On-hand	1	N/A
			Total Cost:	1458.03

*note discounts and shipping costs not applied

8.5 STRETCH GOALS

In order to expand SigSent’s suitability to additional use cases, providing additional methods for human machine interface would make valuable stretch goals. Adding basic gesture recognition to SigSent’s computer vision system could enable SigSent to be a more valuable partner in a human machine pairing. An operator in the immediate view of a unit could wield quick and intuitive control over the robot. Gesture-based interaction with the unit could prove useful for stealthy operations or could enable interaction with nonverbal individuals.

Similarly, adding in additional voice commands would serve a similar purpose, albeit less stealthy. Voice commands could likely be more verbose, offering greater specificity for an operator to provide commands, using a grammar-based approach.

A mobile app would enable an operator to issue commands to a unit or to receive the audio/visual feed another unit when not in the immediate area. A mobile app could provide similar functionality to a base station in a lighter weight, more mobile package. A mobile app could be designed to operate on both smartphones and tablets to leverage the devices many potential operators would already possess.

Follow the Leader, cost analyses on paths to take, and autonomous investigation are all items that are considerations for future work. The follow the leader strategy has direct implications in the field of patrol and sentry surveillance. If a human security officer could walk their usual route while accompanied by the SigSent robot, the exact pathing could be saved by the robot and then executed for future patrols without the human’s assistance. This would mean that the human operator would not have to use the teleopoeration feature to control the robot’s movements while attempting to do surveillance along a path. Another alternative to the follow the leader strategy would be to define a path on a GPS map that the robot then follows. The GPS technology is still necessary in either approach to maintain the desired path, however the human operator would have to be aware of the obstacles present along the defined path that they draw. While walking with the follow the leader mode enabled, the human operator would be able to steer the robot away from any obstacles that would interfere with the robot’s patrol. Added automation means lower operating costs for a security company and for less man-hours spent maintaining the robot’s functionality

Having a cost analysis on the pathing would mean that the shortest paths could be taken by the robot. Shortest path computations would be effective in rough terrain where taking a shorter path has a significant effect on how much energy is expended by the robot. If the computer vision and terrain classification system is sophisticated enough, then its roughness could be used in the movement costs as well. It could be that a longer path with a smoother terrain is the better travel option to save energy and to keep the robot from entering dangerously risky areas that could prove too difficult to traverse.

Autonomous investigation functionality would mean that the human operator could step away from monitoring the patrol route taken by the robot and allow it to encounter anomalies on its own. Currently, the robot can alert the operator of detected movement so that the human can manually investigate. This still requires a human operator to be vigilant in watching the surveillance from the robot. Automating this feature reduces the chance of human faults. A human could miss something anomalous on the surveillance. The robot could detect movement and alert the user, however the user could be oblivious to the warning. Having the system operate intelligently on its own reduces the human interaction element (where the human element adds to the risk of errors).

9 CONCLUSION

SigSent's novel multi-modal terrain navigation methodology can provide significant gains in energy efficiency of locomotion while maintaining the all-terrain capabilities that a traditional hexapod platform offers.

SigSent offers security professionals additional capability to complement and supplement their traditional organizational structure and roles. A network of SigSent's can multiply the effectiveness of a single security guard and enable quicker response over larger distances.

SigSent relies on a wireless control architecture which incorporates a base station for an operator's use. Although this adds a constraint on the operation of the robot, wireless communication with Wi-Fi is ubiquitous in today's modern society, where the Internet of Things (IoT) has dominated every market.

Currently, Knightscope is the largest company producing autonomous sentry/patrol robots. Their products are aesthetically pleasing and seem to have feature-rich devices. Their systems, while featuring intelligent autonomous bots, do not break the mold in multi-modal terrain traversal. Our project hopes to expand on that aspect. The hexapod design lends itself to the ability to cross over rough terrains without the weakness of wheels. The neuroevolution system creates an ever-changing robot that continuously learns from its environments. SigSent boasts a robustness that not many platforms can offer. A robot that can adapt over time without the need for human intervention lowers operating costs and lowers the risk of obsolescence.

As demonstrated by the design planning and development recorded in this document, SigSent's road to completion is well on its way as an affordable option for security. We believe we have created not only a functional prototype, but also a beautiful, unique design unlike anything currently on the market. We have challenged ourselves as engineers and also as innovators.

APPENDIX A: REFERENCES

[1] "Materials Handling: Heavy Lifting," [Online]. Available: https://www.osha.gov/SLTC/etools/electricalcontractors/materials/heavy.html. [Accessed 21 September 2017].

[2] "The Limits of Human Speed," OSHA, [Online]. Available: https://www.ncsf.org/enew/articles/articles-limitsofhumanspeed.aspx. [Accessed 21 September 2017].

[3] "iPatrol Product Questions," [Online]. Available: https://www.ipatrol.net/faq/. [Accessed 21 September 2017].

[4] "GOV GPS Accuracy," [Online]. Available: http://www.gps.gov/systems/gps/performance/accuracy/. [Accessed 21 September 2017].

[5] "Air 802 FCC Regulations," [Online]. Available: https://www.air802.com/files/FCC-Rules-and-Regulations.pdf. [Accessed 21 September 2017].

[6] "Understanding the FCC Regulations for Low-Power, Non-Licensed Transmitters," [Online]. Available: https://transition.fcc.gov/Bureaus/Engineering_Technology/Documents/bulletins/oet63/oet63rev.pdf. [Accessed 21 September 2017].

[7] "Sound and Vision Bandwidth," [Online]. Available: https://www.soundandvision.com/content/how-much-bandwidth-do-you-need-streaming-video. [Accessed 21 September 2017].

[8] "Security Guards and Gaming Surveillance Officers," 24 October 2017. [Online]. Available: https://www.bls.gov/ooh/protective-service/security-guards.htm. [Accessed 10 November 2017].

[9] M. Mori, K. F. MacDorman and N. Kageki, "The Uncanny Valley [From the Field]," *IEEE Robotics & Automation Magazine,* pp. 98-100, 06 June 2012.

[10] T. Turpin, "Who's Afraid of the Big, Bad Bugs?," Purdue University, 23 February 1990. [Online]. Available: https://www.agriculture.purdue.edu/agcomm/newscolumns/archives/OSL/1990/February/022390OSL.html. [Accessed 11 November 2017].

[11] *Safe Use Of Lasers,* 2014.

[12] NASA, "Requirements for Soldered Electrical and," National Aeronautics and Space Administration, 2012.

[13] IPC, "Generic Standard on Printed Board Design," IPC, Northbrook, IL, 2003.

[14] W. contributors, "IEEE 802.11," Wikipedia, 27 November 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=IEEE_802.11&oldid=812312401.

[15] W. contributors, "I2C," Wikipedia, 3 December 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=I%C2%B2C&oldid=813373339.

[16] *Specification, Universal Serial Bus,* 2000.

[17] The Qt Company, "Licensing," [Online]. Available: https://www1.qt.io/licensing/.

[18] *Telecommunications and information exchange between systems-Local and metropolitan area networks,* 2004.

[19] S. Fluhrer, I. Mantin and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," in *Selected Areas in Cryptography*, Toronto, 2001.

[20] E. Gakstatter, "What Exactly Is GPS NMEA Data?," GPS World, 4 2 2015. [Online]. Available: http://gpsworld.com/what-exactly-is-gps-nmea-data/. [Accessed 20 11 2017].

[21] Knightscope, Inc., [Online]. Available: www.knightscope.com.

[22] E. R. Volpe, "The ATHLETE Rover," Jet Propulsion Laboratory, 3 November 2017. [Online]. Available: https://www-robotics.jpl.nasa.gov/systems/system.cfm?System=11.

[23] Contributers, "Finite-State Machine," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Finite-state_machine. [Accessed 16 November 2017].

[24] Open Source Robotics Foundation, "About ROS," ROS, [Online]. Available: http://www.ros.org/about-ros/. [Accessed 17 11 2017].

[25] B. Gerkey, "What is ROS exactly? Middleware, Framework, Operating System?," ROS, 6 December 2011. [Online]. Available: https://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating-system/#18055. [Accessed 16 November 2017].

[26] Y. Tawil, "An Introduction to Robot Operating System (ROS)," All About Circuits, 26 June 2017. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/. [Accessed 16 Novemeber 2017].

[27] A. Ghatak, "ROS: Can you control the ROS Turtle and make your first ROS System?," MieRobot, 11 May 2017. [Online]. Available: https://www.mierobot.com/single-post/ROS-Turtle. [Accessed 16 Novemeber 2017].

[28] P. Goebel, "Robot Cartography: ROS + SLAM," pi Robot, 26 November 2010. [Online]. Available: http://www.pirobot.org/blog/0015/. [Accessed 16 November 2017].

[29] J. Bohren, "SMACH," ROS, 20 February 2017. [Online]. Available: http://wiki.ros.org/smach. [Accessed 16 November 2017].

[30] J. Bohren, "smach_viewer," Institute for Systems and Robotics, 19 November 2011. [Online]. Available: http://library.isr.ist.utl.pt/docs/roswiki/smach_viewer.html. [Accessed 16 November 2017].

[31] W. contributors, "Supervised learning," 25 September 2017. [Online].

[32] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE transactions on information theory,* pp. 55-63, 1968.

[33] W. contributors, "No free lunch in search and optimization," 9 July 2017. [Online].

[34] D. W. G. Christoph Sommer, "Machine learning in cell biology–teaching computers to recognize phenotypes," *J Cell Sci,* vol. 126, pp. 5529-5539, 2013.

[35] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics,* vol. 6.1, pp. 4-22, 1985.

[36] W. contributors, "Reinforcement Learning," Wikipedia, 3 December 2017. [Online]. Available: https://en.wikipedia.org/wiki/Reinforcement_learning.

[37] G. Cybenko, R. Gray and K. Moizumi, "Q-learning: a tutorial and extensions," *Mathematics of Neural Networks,* pp. 24-33, 1997.

[38] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation,* vol. 10, no. 2, pp. 99-127, 2002.

[39] Gazebo, "Why Gazebo?," Osrf, [Online]. Available: www.gazebosim.org.

[40] P. Goebel, "SV-ROS, Pi Robot win 1st place in IROS 2014 Microsoft Kinect Challenge," Patrick Goebel, 29 September 2014. [Online]. Available: http://robohub.org/sv-ros-pi-robot-win-1st-place-in-iros-2014-microsoft-connect-challenge/. [Accessed 16 November 2017].

[41] W. contributors, "TI MSP430," 15 November 2017. [Online].

[42] Contributers, "Lidar," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Lidar. [Accessed 16 November 2017].

[43] J. Millner, "Robotics Club at UCF," Robotics Club at UCF, [Online]. Available: robotics.ucf.edu. [Accessed 16 November 2017].

[44] Hokuyo, "Distance Data Output/UTM-30LX," Hokuyo, [Online]. Available: https://www.hokuyo-aut.jp/search/single.php?serial=169. [Accessed 16 November 2017].

[45] SuperDroid Robots, "Hokuyo UTM-30LX-EW Scanning Laser Rangefinder," SuperDroid Robots, [Online]. Available: https://www.superdroidrobots.com/shop/item.aspx/hokuyo-utm-30lx-ew-scanning-laser-rangefinder/2252/. [Accessed 2017 November 2017].

[46] "What's the Best Battery?," Battery University, 21 3 2017. [Online]. Available: http://batteryuniversity.com/learn/archive/whats_the_best_battery. [Accessed 22 11 2017].

[47] NXP, "Application guide: ESD protection," 07 2015. [Online]. Available: https://assets.nexperia.com/documents/leaflet/75017664.pdf. [Accessed 3 12 2017].

[48] J. Falin, "Reverse Current/Battery Protection Circuits," June 2003. [Online]. Available: http://www.ti.com/lit/an/slva139/slva139.pdf. [Accessed 03 12 2017].

[49] Raspberry Pi, "ADD-ON BOARDS AND HAT's," [Online]. Available: https://github.com/raspberrypi/hats. [Accessed 27 4 2018].

[50] W. contributors, "Scrum (software development)," Wikipedia, 27 November 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=812385432.

[51] K. Fakhroutdinov, "UML Use Case Diagrams," uml-diagrams, [Online]. Available: https://www.uml-diagrams.org/use-case-diagrams.html.

[52] Hamming, "Error detecting and error correcting codes," *Bell Labs Technical Journal,* pp. 147-160, 1950.

[53] Z. Bingul and S. Kucuk, "Robot Kinematics: Forward and Inverse Kinematics," 2006. [Online]. Available: http://cdn.intechweb.org/pdfs/379.pdf.

[54] "Denavit - Hartenberg parameters," 2017. [Online]. Available: https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters#Kinematics.

[55] Forward Kinematics: The Denavit-Hartenberg Convention, Duke University.

[56] S. Mănoiu-Olaru and M. Niţulescu, "Basic Walking Simulations and Gravitational Stability Analysis for a Hexapod Robot Using Matlab," 2017.

[57] C. S. Gurel, "Hackaday.io," 29 06 2017. [Online]. Available: https://hackaday.io/project/21904-hexapod-modelling-path-planning-and-control/log/62326-3-fundamentals-of-hexapod-robot#header. [Accessed 3 12 2017].

[58] A. Manglik, K. Gupta and S. Bhanoe, "Adaptive Gait Generation for Hexapod Robot using Genetic Algorithm," *IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems,* 2016.

[59] D. K. Pratihar, K. Deb and A. Ghosh, "Optimal path and gait generations simultaneously of a six-legged robot using a GA-fuzzy approach," *Elsevier - Robotics and Autonomous Systems,* no. 41, 2002.

[60] "GPS-Coordinated," [Online]. Available: www.gps-coordinates.net. [Accessed 3 12 2017].

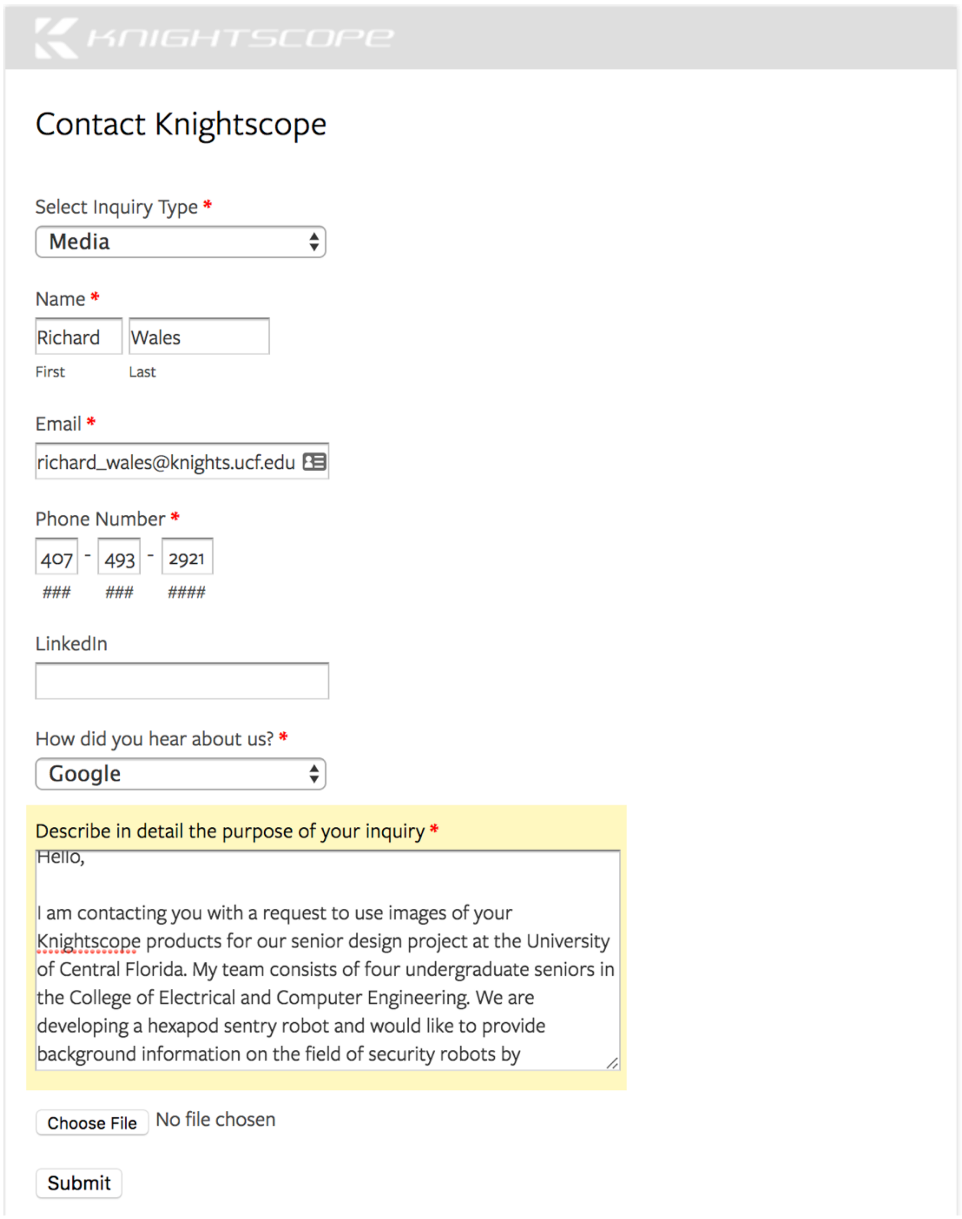
[61] R. Nave, "Frequencies for maximum sensitivity of human hearing," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/maxsens.html. [Accessed 11 November 2017].

[62] Diptrace, "Diptrace," Diptrace, [Online]. Available: https://diptrace.com/. [Accessed 16 Novemeber 2017].

[63] BadDevices, "From Eagle to Diptrace," BadDevices, 5 July 2013. [Online]. Available: https://baddevices.wordpress.com/2013/05/07/from-eagle-to-diptrace/. [Accessed 16 Novemeber 2017].

[64] R. Barraquand, "fOSSa2012," Github, 4 December 2012. [Online]. Available: https://github.com/barraq/fOSSa2012/tree/master/media. [Accessed 16 November 2017].

APPENDIX B: PERMISSIONS


KNIGHTSCOPE

Contact Knightscope

Select Inquiry Type *
Media

Name *
Richard Wales
First Last

Email *
richard_wales@knights.ucf.edu

Phone Number *
407 - 493 - 2921
####

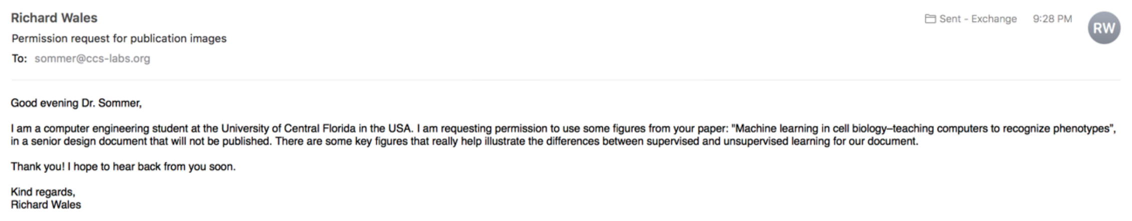
LinkedIn

How did you hear about us? *
Google

Describe in detail the purpose of your inquiry *

Hello,

I am contacting you with a request to use images of your Knightscope products for our senior design project at the University of Central Florida. My team consists of four undergraduate seniors in the College of Electrical and Computer Engineering. We are developing a hexapod sentry robot and would like to provide background information on the field of security robots by



Richard Wales
Permission request for publication images
To: sommer@ccs-labs.org

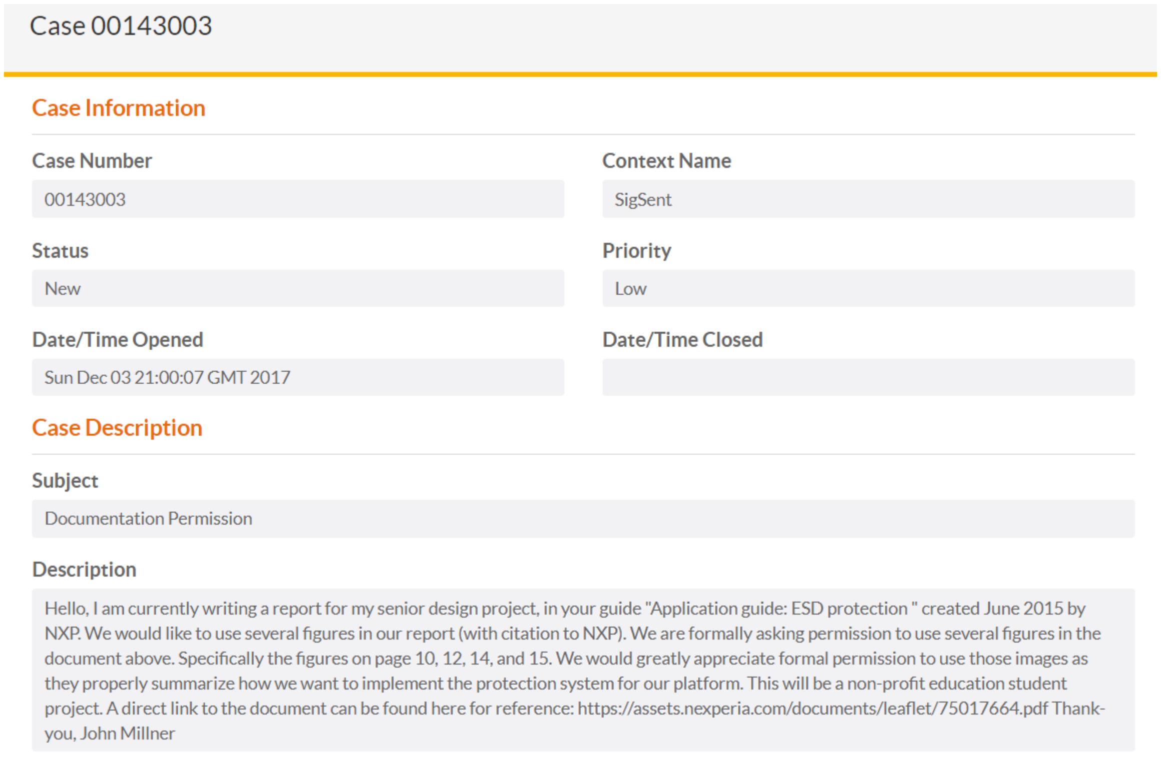
Sent - Exchange 9:28 PM RW

Good evening Dr. Sommer,

I am a computer engineering student at the University of Central Florida in the USA. I am requesting permission to use some figures from your paper: "Machine learning in cell biology–teaching computers to recognize phenotypes", in a senior design document that will not be published. There are some key figures that really help illustrate the differences between supervised and unsupervised learning for our document.

Thank you! I hope to hear back from you soon.

Kind regards,
Richard Wales



Case 00143003

Case Information

Case Number	Context Name
00143003	SigSent
Status	Priority
New	Low
Date/Time Opened	Date/Time Closed
Sun Dec 03 21:00:07 GMT 2017	

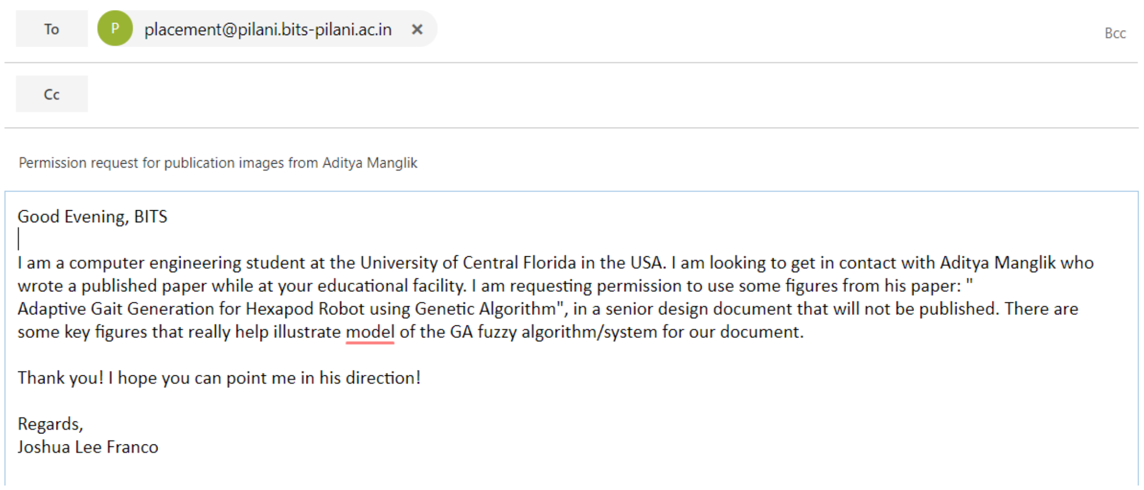
Case Description

Subject

Documentation Permission

Description

Hello, I am currently writing a report for my senior design project, in your guide "Application guide: ESD protection " created June 2015 by NXP. We would like to use several figures in our report (with citation to NXP). We are formally asking permission to use several figures in the document above. Specifically the figures on page 10, 12, 14, and 15. We would greatly appreciate formal permission to use those images as they properly summarize how we want to implement the protection system for our platform. This will be a non-profit education student project. A direct link to the document can be found here for reference: https://assets.nexperia.com/documents/leaflet/75017664.pdf Thank-you, John Millner



To P placement@pilani.bits-pilani.ac.in × Bcc

Cc

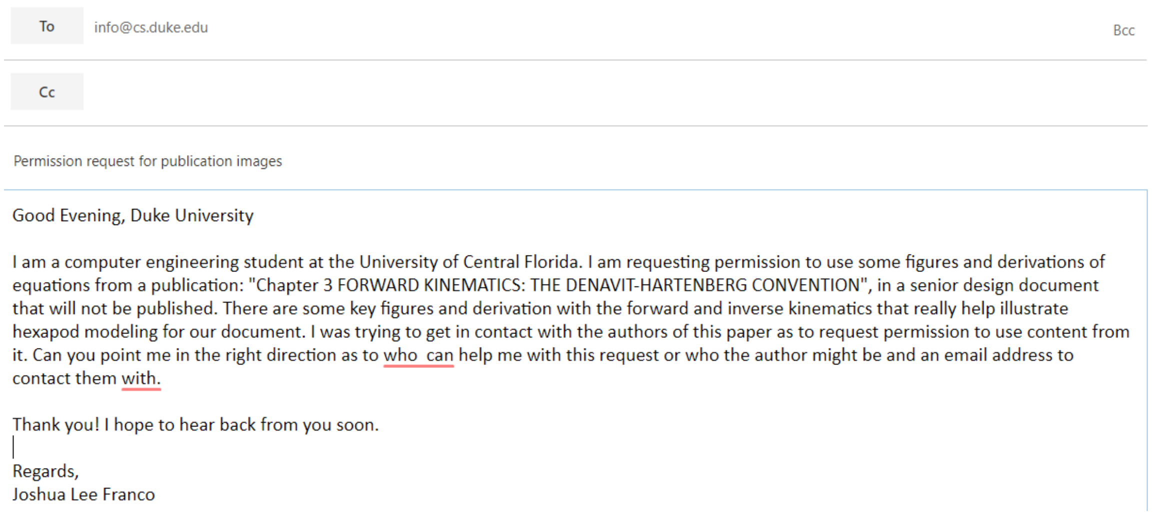
Permission request for publication images from Aditya Manglik

Good Evening, BITS

I am a computer engineering student at the University of Central Florida in the USA. I am looking to get in contact with Aditya Manglik who wrote a published paper while at your educational facility. I am requesting permission to use some figures from his paper: "
Adaptive Gait Generation for Hexapod Robot using Genetic Algorithm", in a senior design document that will not be published. There are some key figures that really help illustrate model of the GA fuzzy algorithm/system for our document.

Thank you! I hope you can point me in his direction!

Regards,
Joshua Lee Franco



To info@cs.duke.edu

Bcc

Cc

Permission request for publication images

Good Evening, Duke University

I am a computer engineering student at the University of Central Florida. I am requesting permission to use some figures and derivations of equations from a publication: "Chapter 3 FORWARD KINEMATICS: THE DENAVIT-HARTENBERG CONVENTION", in a senior design document that will not be published. There are some key figures and derivation with the forward and inverse kinematics that really help illustrate hexapod modeling for our document. I was trying to get in contact with the authors of this paper as to request permission to use content from it. Can you point me in the right direction as to who can help me with this request or who the author might be and an email address to contact them with.

Thank you! I hope to hear back from you soon.

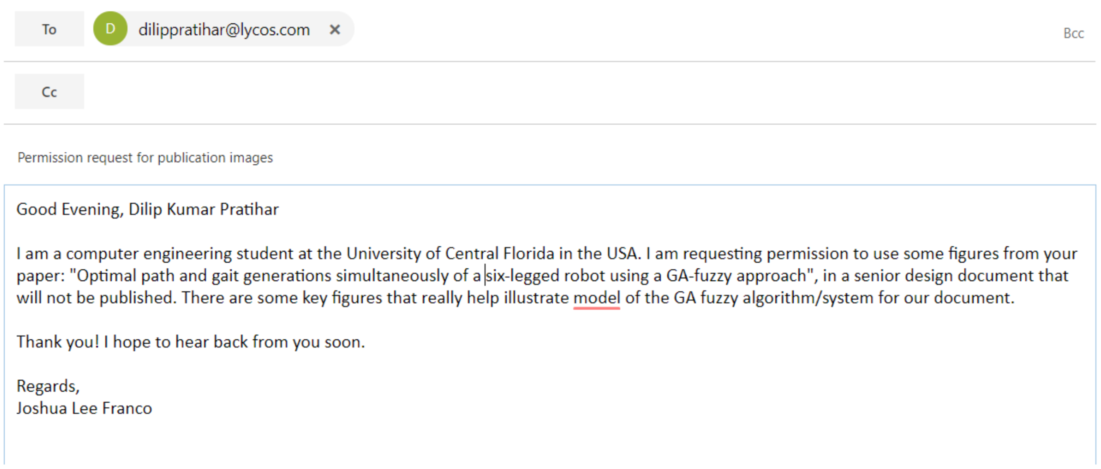
Regards,
Joshua Lee Franco

****

Thank you for contacting the Texas Instruments Semiconductor Customer Support Center. This is an automatic acknowledgement of the receipt of your email. Your inquiry has been assigned Service Request# 1-3754541903 and is currently being worked by our Technical Support Staff.

If additional information becomes available or you have file attachments that you wish to submit, please reply to this email with that information and/or file(s). Please ensure the thread ID at the end of this email is included so your reply is automatically linked to your original request.

> [Email: johnmillner@knights.ucf.edu]
> [Phone: +1 (352) 345-1234]
> [Alt Phone:]
> [Country: USA]
> [Address1: 12413 Golden Knight Cir]
> [Address2:]
> [City: Orlando]
> [State: FL]
> [Postal Code: 34613]
> [Part# or Description:]
> [Queue Name:]
> [Category: University Program]
> [SR Sub-Type: Contests/Promotions]
> [Application:]
> [Support Path:]
> [Problem: Hello, I am currently writing a report for my senior design project, in your guide "Reverse Current/Battery Protection Circuits " created June 2003 by Jeff Falin of TI. We would like to use figure 3 in our report (with citation to TI). We are formally asking permission to use that figure in the document above. We would greatly appreciate formal permission to use that image as they properly summarize how we want to implement the protection system for our platform. This will be a non-profit education student project. A direct link to the document can be found here for reference: http://www.ti.com/lit/an/slva139/slva139.pdf Thank-you, John Millner
>]
> [Automotive App: N]
> [Military Entity: N]
> [CCWSC: 4242307]
> [HJNYYS:41960841]
> [CSVTR:27543900]
> [SR Source: Web]


To D dilippratihar@lycos.com ✕ Bcc

Cc

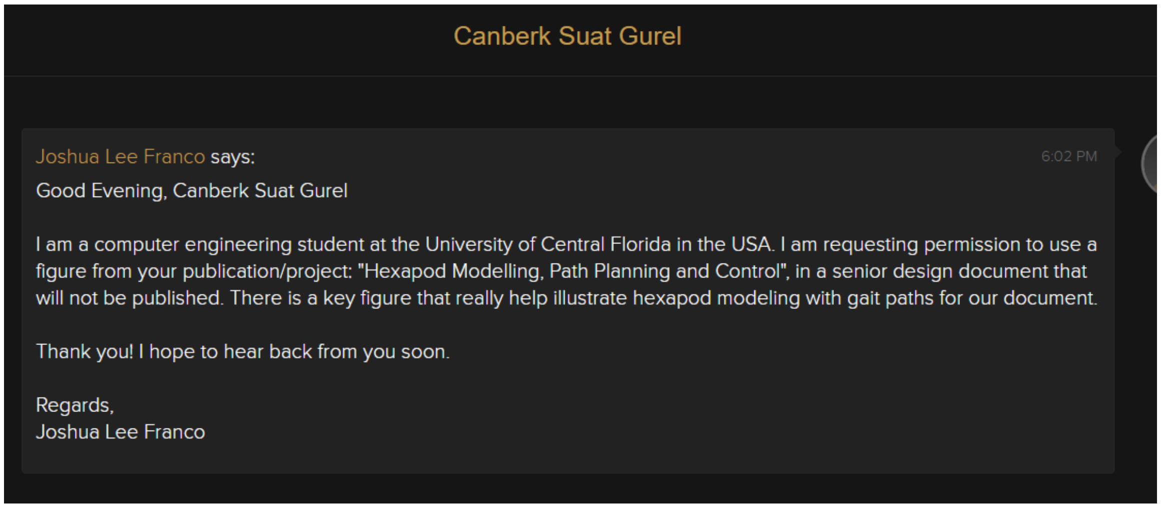
Permission request for publication images

Good Evening, Dilip Kumar Pratihar

I am a computer engineering student at the University of Central Florida in the USA. I am requesting permission to use some figures from your paper: "Optimal path and gait generations simultaneously of a six-legged robot using a GA-fuzzy approach", in a senior design document that will not be published. There are some key figures that really help illustrate model of the GA fuzzy algorithm/system for our document.

Thank you! I hope to hear back from you soon.

Regards,
Joshua Lee Franco


Canberk Suat Gurel

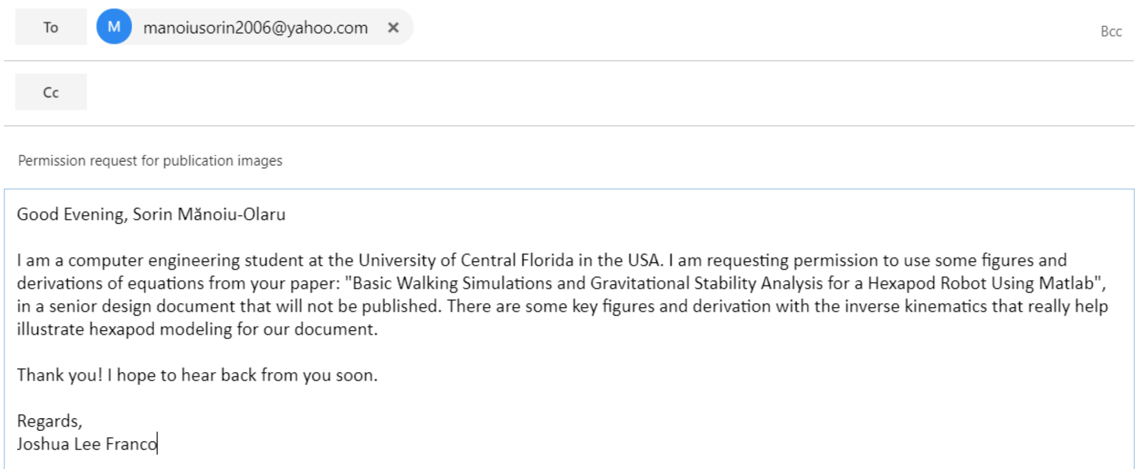
Joshua Lee Franco says: 6:02 PM

Good Evening, Canberk Suat Gurel

I am a computer engineering student at the University of Central Florida in the USA. I am requesting permission to use a figure from your publication/project: "Hexapod Modelling, Path Planning and Control", in a senior design document that will not be published. There is a key figure that really help illustrate hexapod modeling with gait paths for our document.

Thank you! I hope to hear back from you soon.

Regards,
Joshua Lee Franco


To M manoiusorin2006@yahoo.com ✕ Bcc

Cc

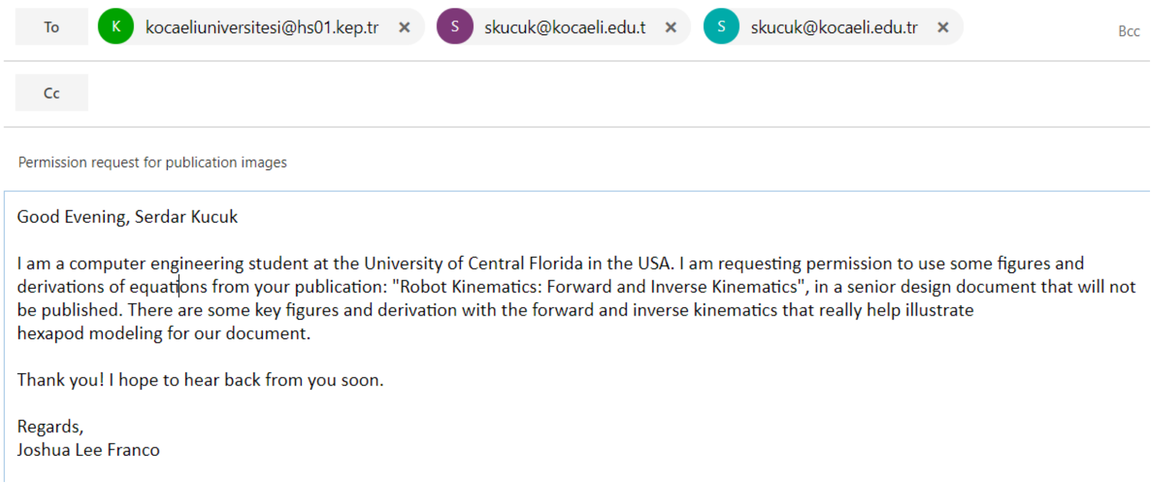
Permission request for publication images

Good Evening, Sorin Mănoiu-Olaru

I am a computer engineering student at the University of Central Florida in the USA. I am requesting permission to use some figures and derivations of equations from your paper: "Basic Walking Simulations and Gravitational Stability Analysis for a Hexapod Robot Using Matlab", in a senior design document that will not be published. There are some key figures and derivation with the inverse kinematics that really help illustrate hexapod modeling for our document.

Thank you! I hope to hear back from you soon.

Regards,
Joshua Lee Franco


To K kocaeliuniversitesi@hs01.kep.tr × S skucuk@kocaeli.edu.t × S skucuk@kocaeli.edu.tr ×

Bcc

Cc

Permission request for publication images

Good Evening, Serdar Kucuk

I am a computer engineering student at the University of Central Florida in the USA. I am requesting permission to use some figures and derivations of equations from your publication: "Robot Kinematics: Forward and Inverse Kinematics", in a senior design document that will not be published. There are some key figures and derivation with the forward and inverse kinematics that really help illustrate hexapod modeling for our document.

Thank you! I hope to hear back from you soon.

Regards,
Joshua Lee Franco