2025

Project Portfolio

Gavin Money
gcmoney@crimson.ua.edu

Alabama Astrobotics – Student Design Team

Alabama Astrobotics is a University of Alabama student design team that competes in NASA's Lunabotics Challenge. The Challenge encapsulates designing an autonomous rover that navigates through rocks and craters to reach a mining zone, where it then repeatedly collects simulated lunar regolith and deposits it as berms in a construction zone. I first started contributing significantly to the software team in the Fall of 2023 when I was given the role of actuation sub-lead. This entailed integrating both new motors and new motor controllers into the codebase and being responsible for all firmware and software relating to actuating systems, including constructing and deconstructing CAN messages. I was effectively a liaison between the software and electrical subsystems, working together with the electrical members to coordinate integration efforts. I oversaw 2 projects during this time. The 1st project required tuning the PID values in the firmware of the motor controllers since they were all set to factory default values. The 2nd project was in response to an incident where our pub/sub was shut down during operations, but motors continued actuating; the solution was to refactor the motor controller parent class by adding a watchdog timer that, if communications time out, publishes stop actuation messages on the CAN buses.

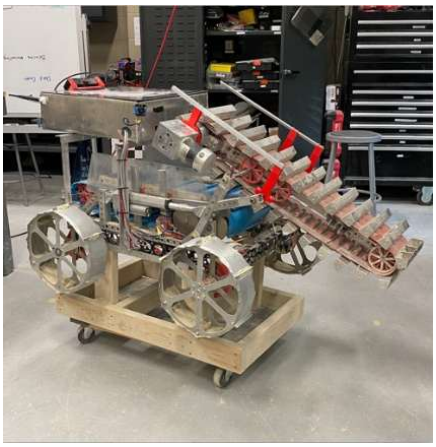
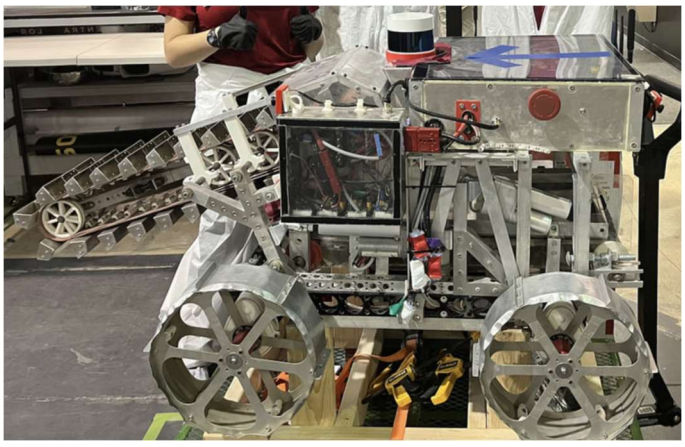
In the Spring, I collaborated with 2 other software members to rewrite the finite state machine (FSM) that defines our high-level autonomy. One concern the electrical and mechanical subsystems shared was that, because the motor that supported our bucket-ladder mining system was so strong, we could destroy the bucket-ladder system and/or the motor itself by becoming stuck on deep gravel when mining. I supervised the implementation of our solution to this problem, which utilizes an exponential moving average (EMA) to monitor amperage draw of the digging motor; if the EMA surpasses a set threshold, then stall protocols in the FSM kick in, such as temporarily reversing both the robot and the direction of the digging motor to become unstuck. During this time, I also supervised a project in which the procedural setup of our pub/sub and CAN buses were automated to simplify piloting the rover.

Weeks from competition, it was discovered that our localization system was inaccurate when facing the AprilTag dead-on. To counter this, while others attempted to debug AprilTag code, I wrote an entirely new version of the FSM that relied on timing as opposed to localization; this is the code we ran at competition, part of which I wrote at competition. In May of 2024, we competed against 50 other schools at UCF and Kennedy Space Center. During this time, we were faced with an onslaught of communication issues and were forced to scrap our communication hardware entirely. We stayed up for multiple days configuring and testing different over-the-counter solutions from the local hardware store. We settled on a router/USB-antenna setup that required installing custom open-source drivers to enable Linux compatibility. When we weren't working on communications, I was working on tuning FSM operations. For 9 days, the software team was averaging 1-2 hours of sleep per night. Despite all this, we overcame as a team and won the entire competition, placing 1st overall, 1st in autonomy, and 2nd in mining.

The following competition cycle, 2024-2025, I was made the software team co-lead. Throughout the year, my co-lead and I held weekly meetings in which we mentored and managed our 43 software members. These meetings involved onboarding new members, giving lectures on our autonomy system, assigning projects, checking in on the progress of projects, and holding work sessions to offer programing help. Our first order of business was building a working localization system. We made the decision to move away from AprilTags and toward LiDAR because past Astrobotics teams have succeeded with LiDAR. The specific LiDAR model we implemented was only sold for 1 year, and the parent company was subsequently acquired, so there was no existing documentation, support, or C++ driver to be found anywhere online. We worked around this by finding a custom, open-source sensor driver for a similar

sensor and modifying it to support the transforms required to construct accurate point clouds. We demoed working localization and autonomous navigation in late November of 2024.

In the following Spring, I managed 6 projects. These included: debugging IMU yaw drift, which revealed our IMU had a broken magnetometer and needed to be replaced; building a grid visualization tool that allowed us to visualize our obstacle detection, path-planning, and path-following real-time; developing an algorithm that utilizes random-sampling and region-growing to find obstacle-free areas in the mining zone that are optimal for mining; writing a backup localization alternative to our EKF that, instead of fusing sensor data, switches between sensors that are available based on priority (ex: trust LiDAR over everything else, but if LiDAR drops out, switch to wheel odometry to estimate linear distance travelled, and use the IMU for yaw estimation); tuning wheel odometry to account for new gear ratios and validating our dynamic slip coefficient calculations; writing scripts to port onboard camera-feeds to laptops over SSH, configuring compression, resolution, FPS, and other parameters that affect bandwidth usage. Throughout the Spring, our access to hardware for testing was extremely limited, which led to debugging autonomy at competition. We were able to achieve consistent autonomous operations in the Lunar Regolith Simulator (LRS) in our lab, but when we competed at UCF and Kennedy Space Center in May, we were unable to translate our fully autonomous capabilities to the new arenas. At UCF, we ran into issues with RANSAC: there was a massive rock ledge that raised the ground plane so that everything in front of the rover appeared to be a crater. At Kennedy, we experienced a localization error of about 20 cm during our navigation, causing the rover to become stuck on a rock and requiring teleoperation intervention, temporarily halting autonomous operations. Despite these challenges, our team worked together vigorously to adapt and overcome, placing 2nd in autonomy, 3rd in mining, and 3rd overall. Our next steps as a software team are to run as many autonomous iterations as possible in the LRS to discover what other bugs exist in our system.

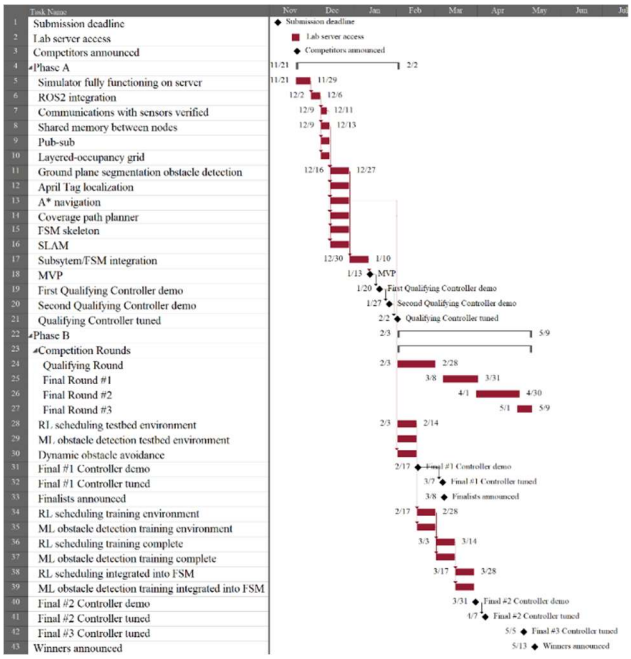



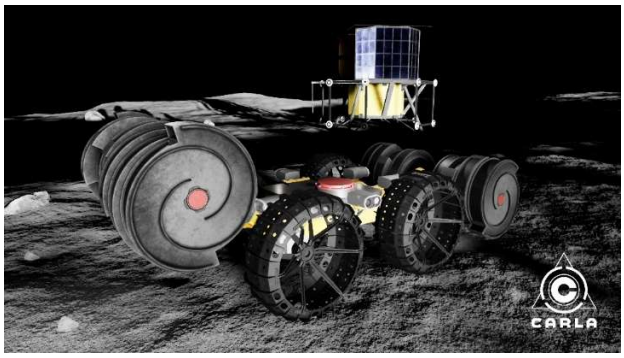
<u>The Lunar Tide – Student Design Team</u>

The Lunar Tide was a student design team that 5 friends and I put together in the Fall of 2024 to compete in the NASA Lunar Autonomy Challenge. The Challenge consisted of programming a simulated digital twin of the ISRU Pilot Excavator (IPEx) rover to autonomously traverse and map the surface of the moon surrounding a lunar landing module. I maintained the role of concept of operations (ConOps) lead, being responsible for the integration of all systems into a cohesive unit. To enter the competition, we were required to submit a project management plan (PMP). In addition to helping author this documentation, I came up with a project roadmap that included all deadlines and deliverables, both

internal and external, and organized them into a Gantt chart. I integrated each submodule into an FSM and then shifted my attention towards aiding my teammates in their efforts.

I worked with multiple simultaneous localization and mapping (SLAM) packages, building them in the simulator and editing python-wrappers as necessary. After much testing, it was determined that, because of the low-light conditions and non-feature-intensive environment of the lunar surface, SLAM could not be feasibly implemented in our given timeline. At this point, I switched my focus to aiding the implementation of AprilTag localization. Our localization lead left the team in February for personal reasons, leaving us without a working AprilTag system. There were no position transforms written out, and once they were completed, we tested for 2 weeks without success. We figured we must have messed up the transformation math somewhere, but after investigating the AprilTag package itself, it turned out we were using a corrupted fork and needed to start from scratch with a whole new package. Unfortunately, due to time constraints, we did not have a working version of our autonomous agent ready to submit by the qualification round deadline. Even though we did not see the results we were hoping for, this project provided valuable lessons in effective project management under time constraints.





<u>Camgian - Artificial Intelligence/Machine Learning Co-op</u>

Camgian contracts with the Department of Defense to create counter-UAS solutions using artificial intelligence (AI) and machine learning (ML). I've completed 4 work terms with Camgian, having worked on simulations, backend software, a reinforcement learning testbed, and a computer vision testbed. In 2023, I began my 1st work-term, working exclusively with AFSIM simulations, collecting, interpreting, and presenting data. In my 2nd term, I worked with backend software, optimizing algorithms using multi-threading and multi-processing.

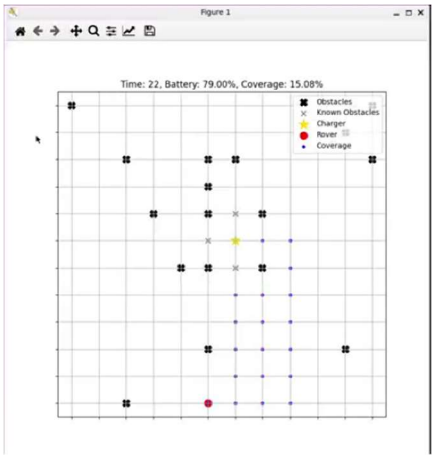
In my 3rd term, I worked with a reinforcement learning testbed, focusing specifically on reward engineering enhancement. To finish off the term, I was tasked with creating a mock flight controller that mimicked different movements of airborne objects. In my final term, I travelled to 4 field tests and

helped transport, set up, and break down hardware; I then visualized data on my local machine for senior developers to interpret. Following a field test, I would run collected data through an analytics system and compile the data in a document. I also worked on a classification project where another developer and I began researching different classification algorithms. After conducting this preliminary research, we chose 4 model architectures to compare more closely. I performed literature reviews on 3 of the 4 models, documenting traits such as differences in macro/micro architectures and training methods. I then built a testbed to train different versions of these models and generate training and validation metrics. Finally, I generated metrics for all these algorithms and compiled them into a 185-page document along with the literature review findings, testbed model configurations, and my own personal recommendation as to which algorithm the company should move forward with. I then presented our findings in the sprint retro meeting.

BOREALIS – Forthcoming Conference Publication

BOREALIS is a "Battery-Oriented Resource Efficiency Algorithm for Lunar Intelligent Systems" that will be presented in our forthcoming SciTech Forum 2026 conference publication, *Soft Actor-Critic Learning of Finite State Machine Transition Policies for Lunar Rovers.* This paper evaluates different reinforcement learning policies in the context of making FSM state transition decisions. The inspiration for this paper originally came from The Lunar Tide project, where we competed in NASA's Lunar Autonomy Challenge as aforementioned. In that challenge, the rover must occasionally return to the lunar landing module to recharge its battery and radiate heat.

The simplest approach to charging is to set a battery percentage threshold that determines when you return to the lander to charge (i.e. go back when you have 30% battery life remaining). Our team believed that scheduling the decision to return to the lander could be optimized with reinforcement learning by making it not only a function of battery percentage, but also of rover position, area mapped, etc. Although we did not get a chance to implement it in the Lunar Autonomy Challenge, 2 teammates and I decided to pursue the research anyways. I created the proof-of-concept, building a testbed and training models that were comparable to the simple approach, and visualizing the rover decisions on a rudimentary grid. Since then, we've continued developing the testbed and comparing different reinforcement learning policies, such as how algorithms with continuous versus discrete action spaces compete.



Embedded and Robotics Systems Lab – Undergraduate Research Assistant

The Embedded and Robotics Systems Lab specializes in computing for embedded applications; embedded software development; computer vision; real-time computation; autonomous navigation including localization, path-planning, obstacle detection and avoidance; machine learning; robotics simulation; FPGA applications; space-based robotics.

I began working with a PhD candidate, Leo, in the Spring of 2025, to apply AI strategies to perform object recognition in unstructured environments for unmanned ground vehicles operating in off-road environments. Currently, Leo has trained models on labeled temporal point-cloud data that depicts trails in wooded environments. His algorithm is promising compared to benchmarks against other models, but the training data needs to be improved.

When I joined the lab, Leo's point-cloud visualization script had to be rerun for every single point-cloud it visualized. It was also running on an outdated OS and only worked on the lab computer – all other computers faced dependency issues. I solved the dependency problem by writing a script that went through and logged every single incompatible library, all of which I removed. I then built up from there, installing libraries until the visualization script worked before putting together an environment.yaml file to streamline future setup. I then added functionality to the script, allowing point-clouds to be scrolled through continuously. Finally, I added the ability to save and delete point-clouds from a working list of point-cloud files, which I then used to sort through and filter out bad point-cloud data.