Command and Control Subsystem for Regolith Mining Robot

Semester 2 Project Plan

Team Members - Software Engineering:

- Pablo Canseco - Software/Communications Lead

Team Members - Other Majors:

- Leyane Mohammed — Project Manager
- Domenick Albanese — Systems
- Ronald-Dean Allado — Software
- Kyle Rieder — Software / Consultant
- Mark Thames — Communications / Electrical
- Khalphani Green — Electrical Lead
- Adrian McHargh — Electrical
- Spencer Lower — Electrical
- Nathaniel Voris — Structures Lead
- Zoher Kothari — Structures
- John Breen — Structures
- Jacob Netzley — Structures
- Abigail Stevens — Structures
- Ashle Thompson — Structures

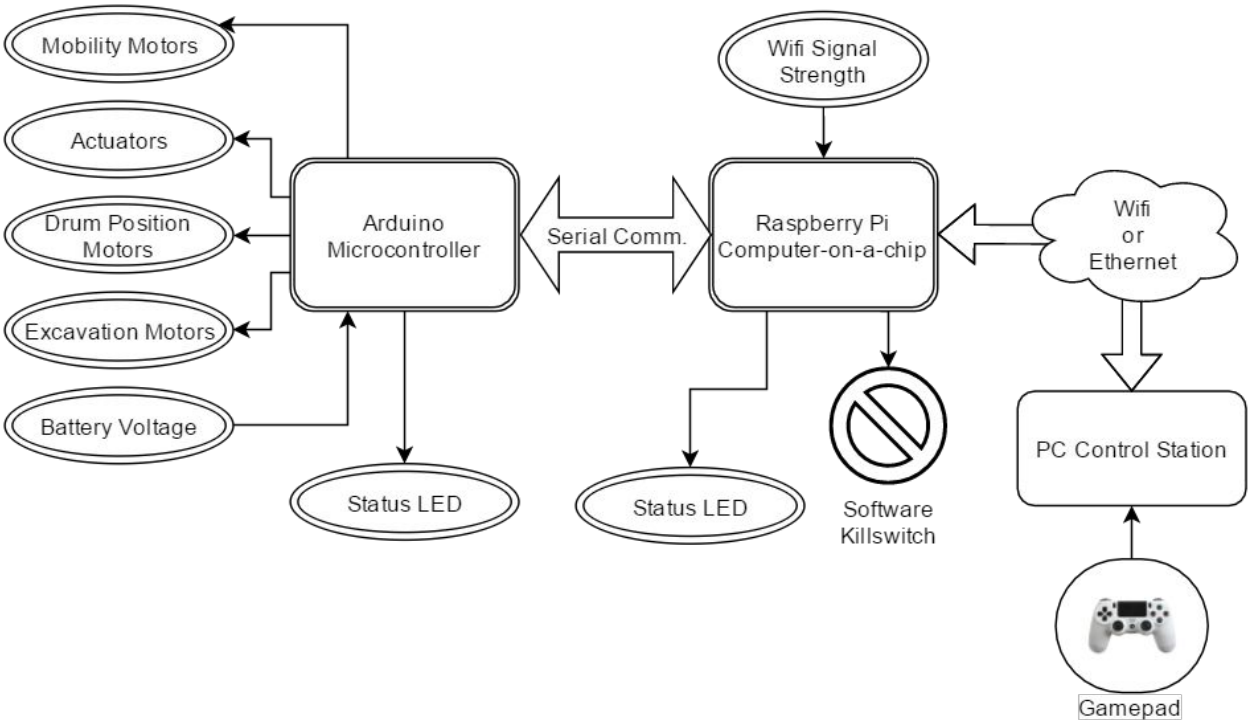
Faculty Sponsor / Customer: _____

Dr. Keith Gallagher

Project Goal / Motivation:

The project aims to provide robust command-and-control capabilities to FIT's entry into NASA's Robotic Mining Competition (NASA RMC), Project ARES. Project ARES will feature a regolith (lunar/martian soil simulant) mining apparatus, a transport mechanism, a communications subsystem, and a data processing / decision making subsystem. Once robust and reliable control of the robot's electrical and mechanical components is achieved, autonomous control options will be explored in order to give the school a competitive edge in the NASA RMC this coming May, 2016.

Approach:



The figure above is a functional diagram for the subsystem of the robot. The project aims to develop the software of the subsystem and allow each component to communicate with the others. The hardware will be composed of three main parts, the Arduino, the Raspberry Pi, and the PC Control Station. The Arduino and Raspberry Pi will be on-board within the robot, and the PC Control Station will be separate and controlled by a human.

The PC Control Station is what the human operator will use to control the robot and be able to see telemetry data from the robot itself. The majority of the input will come from the ergonomic gamepad, and will leave the Control Station via Ethernet or Wifi to the robot. Conversely, data coming back from the robot will use the same connection so it can be received at the Control Station.

The Raspberry Pi is a full computer inside a small circuit board the size of a credit card. It features an ARM quad-core processor, 1GB of RAM, and USB as well as Ethernet connectivity. It's running a modified version of Linux and will interact with the Control Station via a TCP server application written in Python, executed on startup. Commands received from the Control Station will be parsed here to determine the next course of action. This is also where data will be collected from the robot and sent back to the Control Station.

The Arduino is a microcontroller capable of Serial communication and driving driving digital signals as well as reading analog signals. The Arduino will have a wired Serial transmission link to the Raspberry Pi. This will mean the Raspberry Pi will send the Arduino commands, and the Arduino will report back telemetry data to the Raspberry Pi for eventually sending back to the Control Station.

Technical Challenges:

- **Arduino Environment**- The Arduino will be what drives the electromechanical components of the robot. Many motors, actuators, the battery, switches, and LEDs will be connected to the Arduino and the software has to be able to interact with all of them.
- **Raspberry Pi and Linux-** This will be the brains of the robot. We need to research using Linux to automatically boot up the server application and come into a ready state without user intervention for faster testing and troubleshooting.
- **TCP/IP communications-** The groundwork already exists for TCP/IP client-server communications between the Raspberry Pi and Control Station, but research needs to be done on how to abstract this so that communication can work identically on a wired or wireless connection.
- **Autonomous robot operation-** We might have enough time to do pre-programmed autonomous runs but we have little knowledge on autonomous robot operations.
- **GUI Development-** User-friendliness, performance, and logical GUI structures need to be researched, as we're not very experienced in GUI development.

Design Diagram: Please refer to the figure at the top of the Approach section for an overview on how the system works.

Progress Summary:

Module	Completion %	To do
Control Station GUI	80%	Map rest of gamepad buttons to robot commands. Variable TCP port.
Raspberry Pi Server	50%	Run on startup, prevent crash on disconnect, accept new connections after disconnect, transmit data to Control Station.
Arduino Firmware	95%	Address any unexpected hardware interaction quirks as they come up.

Milestone 4 (Feb 22)**:**

- Raspberry Pi: Prevent crash on disconnect
- Raspberry Pi: Accept new connections after disconnect
- Raspberry Pi: Transmit data to Control Station
- Control Station GUI: Implement variable TCP port
- Control Station GUI: Ongoing testing of Raspberry Pi server functions
- Aid in assembly of subsystem-related hardware on a mock robot hardware prototype

Milestone 5 (Mar 21)**:**

- Test / verify mock robot prototype mobility, excavator positioning, and actuators
- Test / verify mock robot prototype excavation
- Test / verify mock robot prototype software killswitch and status LEDs
- Test / verify mock robot prototype voltage monitoring
- Fix bugs found during the testing and verification of the above functions
- Aid in assembly, test, and verification of the final robot
- Create poster for Senior Design Showcase

Milestone 6 (Apr 18)**:**

- Test and verify all subsystem functionality on the final robot
- Fix bugs found during testing and verification
- Benchmark the robot to test duration and performance, to create approximated measurements on the GUI based on data from the robot.
- Write user manual
- Create finalized demo video

Task Matrix (Milestone 4):

Task	Pablo
Raspberry Pi: Prevent crash on disconnect	100%
Raspberry Pi: Accept new connections after disconnect	100%
Raspberry Pi: Transmit data to Control Station	100%
Control Station GUI: Implement variable TCP port	100%
Control Station GUI: Ongoing testing of Raspberry Pi server functions	100%
Aid in assembly of subsystem-related hardware on a mock robot hardware prototype	50%

Task Description (Milestone 4):

- Raspberry Pi: Prevent crash on disconnect
    - Currently, if the GUI disconnects from the robot (Raspberry Pi), the server application on the R.Pi crashes. The objective is to make it recover gracefully and await a new connection for further instructions.
- Raspberry Pi: Accept new connections after disconnect
    - This is tied somewhat to the above. A solution was previously tried to prevent the crashing issue but it didn't revert back to a state where it would accept new connections, so a different approach needs to be attempted such that it won't crash, and accept new connections after a GUI disconnection.
- Raspberry Pi: Transmit data to Control Station
    - Additional hardware is required. The Arduino transmits signals at 5V and this is too much for the 3.3V tolerated by the R.Pi. The required hardware has been acquired and needs to be tested.
- Control Station GUI: Implement variable TCP port
    - Currently this is hard-coded but should be configurable by the user
- Control Station GUI: Ongoing testing of Raspberry Pi server functions
    - Ensure no new additions or bugfixes break existing functionality.
- Aid in assembly of subsystem-related hardware on a mock robot hardware prototype
    - A prototype is in the works to function mainly as an electrical, software, and comunications test bed so the subsystems can have hardware to move around programmatically and remotely. Assembly and configuration help will be provide for this effort.

"I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones."

Signature: _____ **Date: _____**