COMMAND-AND-CONTROL SUBSYSTEM FOR REGOLITH MINING ROBOT

Design

Fall 2015
Pablo Canseco
pcanseco2011@my.fit.edu
Design Document

Purpose

Project ARES is the name of the robot for FIT's entry into the NASA Robotics Mining Competition taking place May 2016. ARES is a robot designed to mine as much lunar soil simulant as possible in a ten-minute run. The C2C subsystem will be responsible for ensuring ARES is operational, responsive, and is able to achieve the goals of the competition, yielding FIT as many points as possible.

Scope

The C2C Subsystem will be composed of the software running on the onboard computer of ARES, the Control Station software running on a personal computer, and the firmware of all other electromechanical components of the robot. The C2C subsystem will ensure that sensor data is available to the Control Station as well as ensure that the Control Station can communicate and control the robot during its operation. The C2C subsytem will collect data from ARES, send it over the communications link to the Control Station, and accept commands from the Control Station to control ARES.

Definitions, acronyms, and abbreviations

- **ARES** – The robot FIT will use for NASA's 2016 Robotics Mining Competition
- **C2C** – Command-and-Control, the subsystem this document describes
- **CS** – Control Station, where a human operator will interact with the robot
- **Regolith** – Lunar soil simulant to be mined by ARES
- **RMC** – NASA's Robotics Mining Competition

Participants

Pablo Canseco	Software / Communications Lead
Leyane Mohammed	Project Manager
Domenick Albanese	Software
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

General Overview

We plan to develop a complete solution for remotely controlling a robot using standard software practices recommended by other software developers using similar platforms. We intend to break up the system into two parts, a control station and the ARES onboard software. Linking them will be a TCP/IP connection directly between the two systems. The robot will act as a server and the control station will act as a client. During manual operation, there will be two connections to the server, one exclusively for robot data from ARES to the control station, and one for motor and hardware control. Autonomous operation will drop the connection for motor and hardware control such that ARES robot / operational data is the only thing going through the connection.

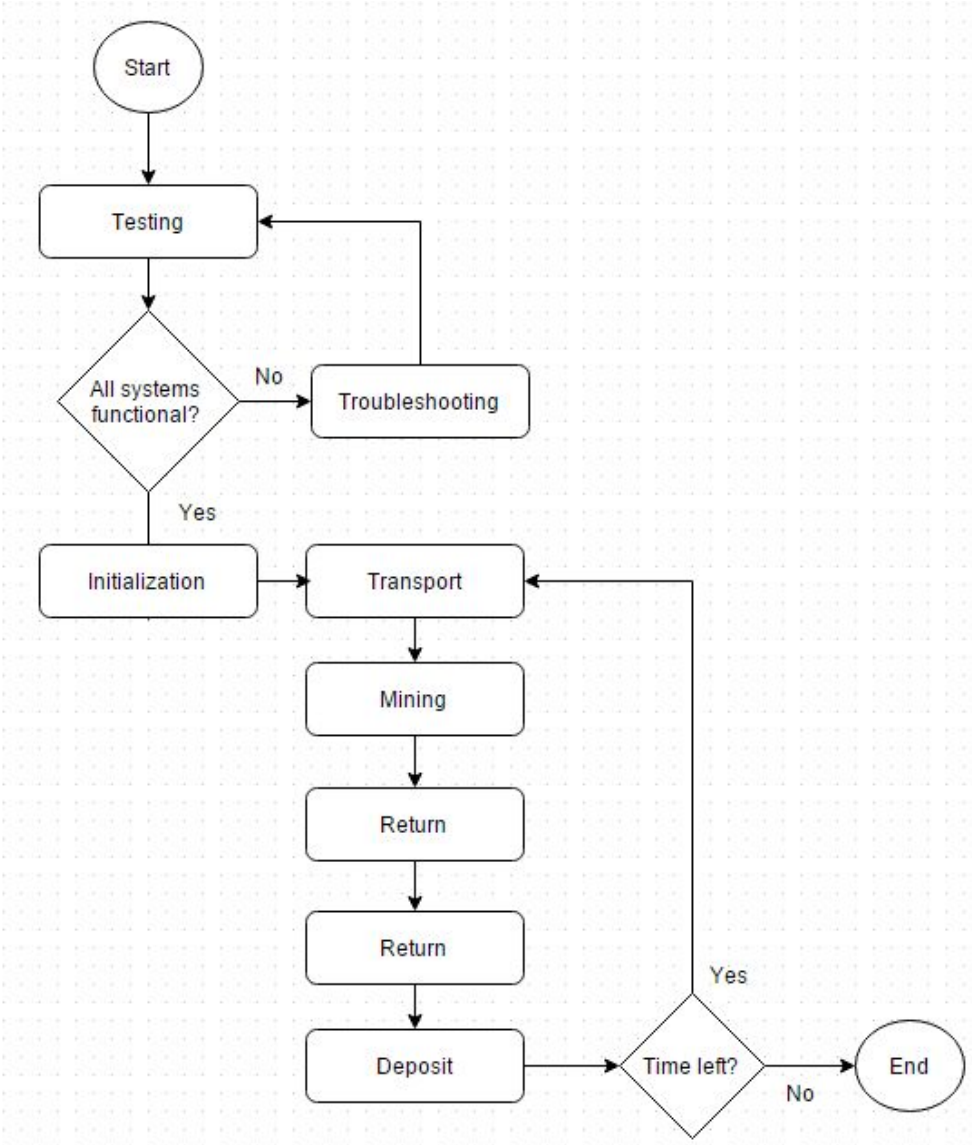
On the robot, software running on a Raspberry Pi will control an attached Arduino. The Arduino will control the various motors and collect data from the various sensors. The Arduino will send the data back to the Raspberry Pi for processing and ultimately to the control station. The aim is to make the system responsive by using input smoothing and a low-level TCP (or UDP if TCP yields too much latency) connection. The robot, under manual control will move away from it’s starting location in the arena, move towards a mining zone, mine as much regolith as the mining mechanism can hold, move back towards the regolith collection bin, and dump it all in.

WorkFlow

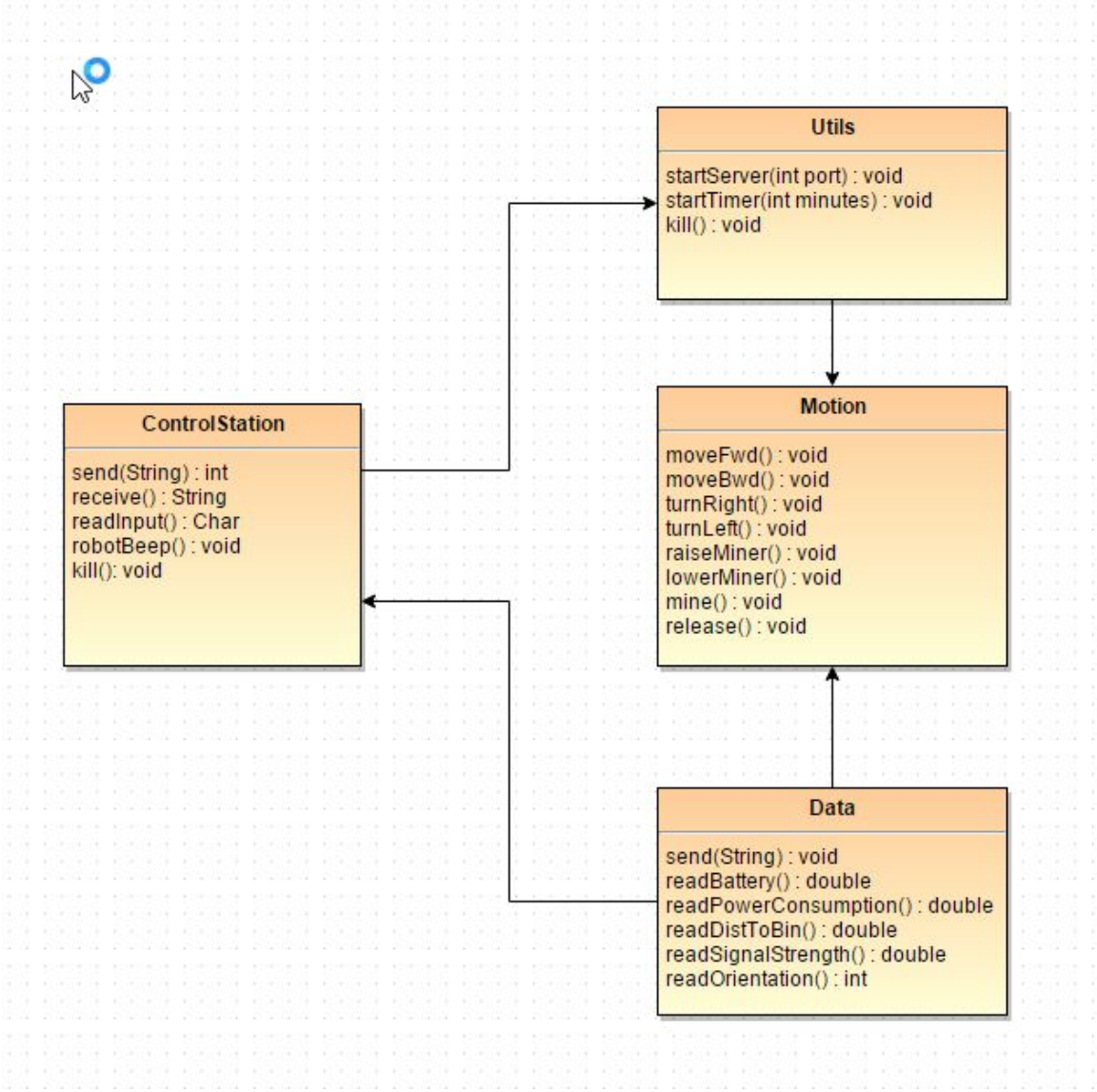
The phases the robot will undergo in the competition will be:

1. Testing
2. Initialization
3. Transport
4. Mining
5. Transport
6. Release

Subsystem Flow Chart



Class Diagram



This is a very high-level overview of the methods and classes that will be implemented in the subsystem. In general, the ControlStation will involve a window with a console that outputs what it reads from user key presses, a few fields where robot data like battery voltage and power consumption will be displayed, and some software buttons for the soft killswitch and to connect to the robot. On the robot side, the software will be entirely text-based with commands being received and data being sent over TCP. A Motion class will control all the motors involved, the Data class will handle reading data and sending it back to the control station, and the Utils class takes care of starting the

server on the robot, will have a timing utility, and will call the killswitch function if instructed to do so by the Control Station.