

# Initial Report for Lunar-cy

Brad Mouring  
Lunar-cy  
EEL 5666  
Intelligent Machine Design Lab  
Dr. Arroyo  
Dr. Schwartz  
Adam Barnett  
Kevin Claycomb

## Table of Contents

Abstract.....	3
Executive Summary.....	3
Introduction.....	3
Integrated System.....	3
Mobile Platform.....	4
Actuation.....	5
Sensors.....	6
Behaviors.....	6
Experimental Layout and Results.....	7
Conclusion.....	7
Documentation.....	8
Appendix A.....	8
Sample Mavric II “Blinkenlights” Code.....	8

## Abstract

The main thrust of this robot is to explore possible approaches to the 2008 IEEE SCon hardware competition robot. This robot must simulate a Lunar Mining game in which various “ores” must be collected and deposited in the “home base.” This will require a solid platform (needed for traversing the rough terrain) as well as some form of item identification. Identification is the area I wish to explore by attempting to locate the blocks before starting and plotting a tree (or perhaps a “skippable” list) of best routes (as the environment will likely change as the other robot will be collecting at the same time).

## Executive Summary

This section is not included in this initial report.

## Introduction

The problems presented by this project range from the standard issues (such as obstacle avoidance) to more complex (such as discerning a block of the same color as the background it sits in). To overcome the issues presented by the problem, various sensors will be exploited to gain knowledge of the surrounding environment and, thereafter, modify the behavior based on the observed environment.

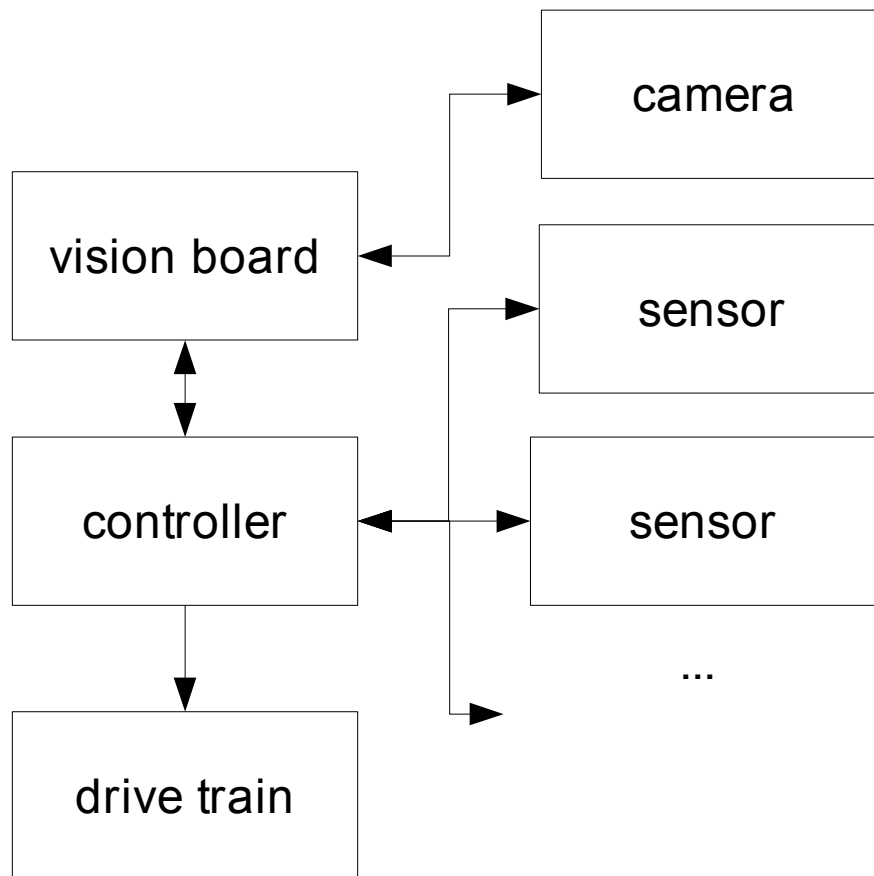
The reason for approaching this problem personally is the interest in advancing my personal knowledge of computer vision and basic artificial intelligence as well as mechanical skills and the tools that accompany them. It would have been quite easy to simply work towards personal strengths alone while planning the robot, however it is much more rewarding to expand new skill sets rather than fall back on existing ones.

The scope of such a project, as alluded to above, is many “locate items, find an efficient method to collect them” problems. The framework for identifying the goals will need to be changed, however the remainder of the system can remain relatively untouched and perform the desired new task.

This initial report will discuss proposed ideas and work already performed in the overall system construction, the platform design, motion or actuation systems and ideas, sensor selection and use, system behaviors, and will discuss experiments performed in the process of discovering the use and characteristics of the various subsystems.

## Integrated System

The system will consist of a primary drive controller, the drive train itself, the sensors it will use to determine how to proceed, and a vision system that will inform the drive controller where to go. A simple diagram laying out the overall hierarchy of that system is provided below.



This modular approach to separating platform control from vision systems allows for simplified integration between different revisions of the base, allowing for improvement and exploration in terms of testing out experimental base ideas.

From the flowchart, the vision board can communicate to the controller board as to the heading of the next block to acquire, while the controller board, interfaced with the pertinent sensors, can best decide how to get to the location dictated by the vision board while avoiding obstacles.

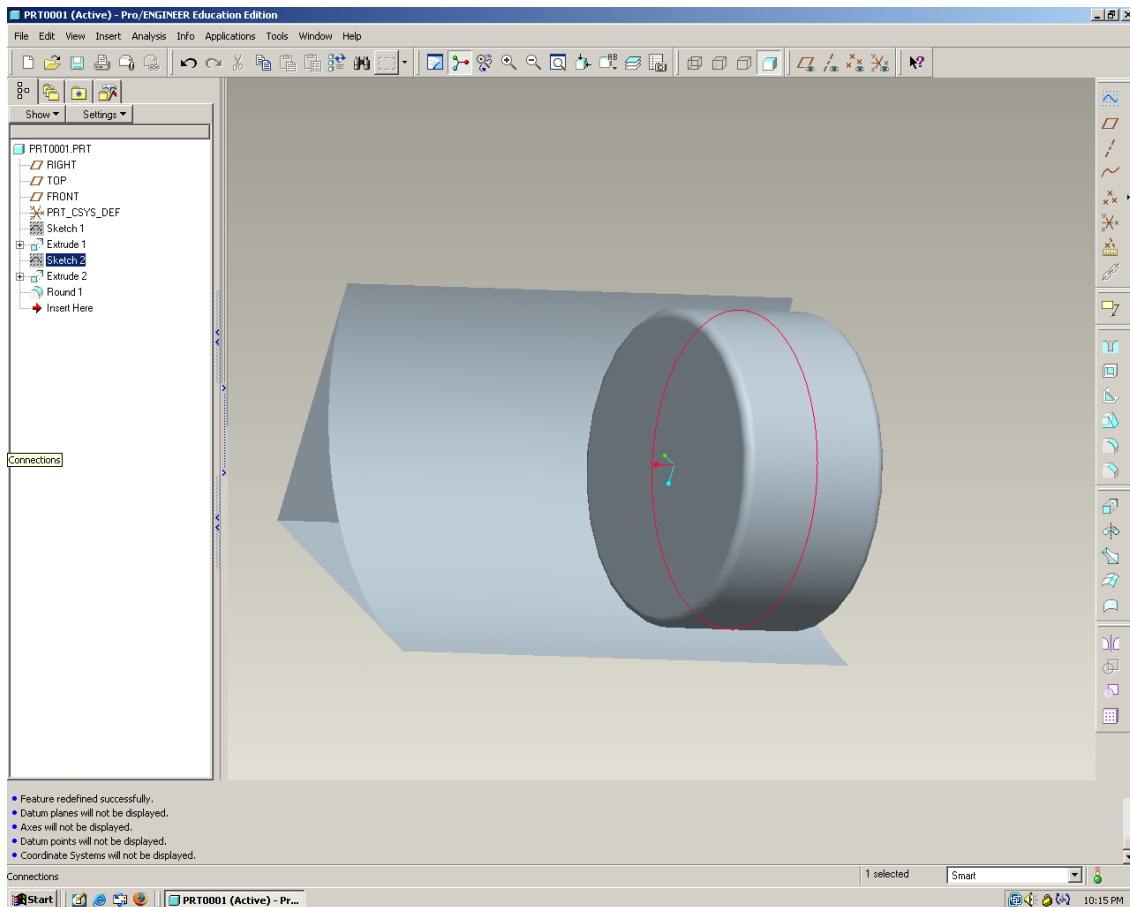
## Mobile Platform

The base for the project must be able to cope with semi-rugged terrain (lava rocks and marble chips adhered to the playing field). Additionally, if design decisions can be made to assist the project as a whole in the picking up of blocks, this would be a desirable feature.

The terrain the platform must deal with demands a rugged, well-constructed base if it is to withstand a round in competition, let alone tens or hundreds of rounds of testing. In order to facilitate this demand, a planned, mechanically sound design will be implemented in a CAD program. Currently, a tank-drive, four-wheel drive base is the planned design. This will provide a simple, rugged, powerful base on which to work from.

In order to facilitate the easy retrieval of the 2" cube blocks, a hybrid scooper/launcher is being considered. This would allow the platform to pick up blocks quickly without needing to stop, all the while adhering to the competition rule stating that no part of the robot may extend

from the robot while it is in motion. A rough prototype is pictured below.



In order to accommodate the rough terrain, the scooper lip will be slightly less than an inch off of the ground and recessed behind the frontmost wheels. As this puts the lip lower than the (theoretical) center of mass of the block, and with the aid of the rolling assembly (which will have a surface with a high coefficient of friction to “grab” the block and will be located ), this should launch the block into the hopper, waiting to be delivered back to home base. If need be, I will research lowering this mechanism into place when approaching a block to ensure successful pickup.

## Actuation

Actuation is required to “bring the robot to life,” so to speak. The actuation must move parts of the robot in order to accomplish the goals specified by the solution to the problem to be solved.

As a result of the planning of this robot, relatively little is needed in the way of actuation. The wheel system will be gearhead brushless motors to ensure they can traverse the terrain (and that very terrain makes encoded motors essentially useless for dead-reckoning) powered through an isolated power system. I am currently researching the motor, wheel, and driver combination(s) that will best meet my needs.

As noted in the **Platform** section, I may use a servo system to lower the scoop/grabber

assembly if the static model isn't performing well. Limiting complexity is a cornerstone of many successful IEEE SECon winners, including Kevin Claycomb's entry two years ago. A potential risk of lowering the assembly is catching the lip of the assembly on a rock on the playing surface, and as such an additional sensor may be required to detect this situation and take the appropriate action (more in subsequent sections).

Additional actuation is likely to be needed to move the camera to "scan" the environment. The needs of this functionality will be explored when the test playing field is completed and some test images are examined.

The roller to be used in the "grabber" will simply be a brushless DC motor that is enabled or disabled, most likely via a relay. This system is simple enough that to attempt to add complexity here would serve no purpose other than to soak up time and money.

## Sensors

Sensors are the robot's window into the real, analog world (unless you believe we are all in the Matrix, in which case it's a window into a simulation of the real, analog world). The controllers in the robot will gather data from the sensors, review the data, and modify its behavior based on that information. This simply means that the sensors play a crucial role in any project of this scope.

In order to detect and avoid collisions, my project will utilize two to three ultrasonic range finders. In order to assist in detection of a wall or other robot in lieu of a block, a two-plane model is going to be used. This will help inform the controller on how best to react to the perceived object.

In the hopper mechanism, an IR sensor will be used to detect the successful acquisition of a block. This simple system will ensure that the robot does not leave until the block has been "grabbed" by the otherwise feedback-less scoop/grabber system.

Bump sensors will likely be implemented as a collision detection system (hopefully it won't be needed in the final design). This system would be considered a failsafe system as it normally shouldn't be required but is rather in-place to avoid catastrophic results (possible disqualification) in the situation when it is needed and isn't implemented.

The most "visible" sensor (pardon the pun) is the camera. This, when used in conjunction with detection code detailed later, will target the blocks. This complexity is added in the hopes that, with a successful implementation, a best-effort run can be executed to better the chances of winning the SECon competition.

Time permitting, I'd like to add "end of the world" sensors (downward angled IR or sonar) to help protect my investment in time and money from an untimely death by plummeting off of a table.

## Behaviors

The behaviors ingrained in the code running on the robot serve to analyze the data received from the sensors and adjust the operation of the robot to fit the perceived situation. It is this facet that gives robots an anthropomorphic appearance. The goal with the behaviors is to create as failsafe a behavior scheme for all aspects of the robot as possible to ensure predictable responses in unpredictable environments.

The base will, without input from the camera, operate in a constantly moving obstacle avoidance mode. This behavior will examine the data from the sonars. When it detects a block (as evidenced by the lower sonar plane pinging close and the upper plane failing to ping or pinging far), the robot will attempt to collect the block. If a wall or another robot is detected (both planes detect a close object), the robot will avoid the object. To prevent looping behavior, the avoidance direction will be pseudo-randomly picked by either a pseudo-random number generation system or as a portion of a truly random system (A/D reading from a sensor modulo 2, for example).

The vision board will work to find the blocks in the initial scan phase at the start of the competition, create a tree of best-path choices (as at any given time a block in the current best-path may be absconded by the other robot) based on metrics such as perceived distance and block value, and inform the controller board the heading it should take. While moving, the vision system will be tracking the current target block, updating the controller board with new heading information as necessary. Upon successful acquisition of the target block, the vision board, knowing its location based on the initial scan, will direct the controller board in the heading of the next block in its current best-path. If the vision board is unable to detect the next target on the board, the vision board will take the next branch on its best-path tree and, again knowing its location from the initial scan, give the controller board another heading to attempt to resume on the best remaining path. This will continue until either the hopper is full (an indeterminate number of blocks with the current rules as there are conflicting values in two separate locations in the text) or there are no more detected targets. This is the initial, pie-in-the-sky idea of the workings of the vision system, however with a properly positioned camera, good detection techniques and a few-position-in-frame tricks to approximate location of a block (as the blocks must reside on a virtual, or not-painted, grid that overlays the board, positioning a detected block on a virtual map of the field is not difficult as the problem has a reasonable scope).

## Experimental Layout and Results

Experiments are performed to better understand the characteristics and capabilities of various sensors and systems in the robot's design. Thus, they play a vital role in determining just how successful a theoretical application translates to real-world application.

Currently the only experimenting I have done is working to understand the internals of the ATmega128 microcontroller (which also served to ensure I had a working environment with which to program and debug my board). As such, I have simple test code that can blink the programmable LED on the BD Micro MavricII board.

Additionally, from previous work, I have code that runs on the Intel XScale board that will be used for the vision code and can grab video data from the network camera I will use and can parse out individual frames with which to process. This code must be refactored and cleaned for readability and efficiency, but it provides an invaluable working base.

## Conclusion

The actual work accomplished thus far consists of modeling in preparation of creating a solid, machined base, initial familiarization with the microcontroller that will be serving to drive the base, a framework for gathering data from a camera and parsing it into usable bitmap images,

and ordering many parts.

It is difficult at this point to see limitations of my work, aside from the rules provided by the competition. One limitation would be the level of image processing possible on an embedded XScale system that is without a floating-point unit. I imagine other limitations above those imposed by the competition will become apparent as work progresses. The same inability to forecast exceptional areas or areas that could be improved prevent discussion of those aspects of my project at this point.

Future work involves finalizing the base design, machining the base components, building the base, interfacing the controller board with the various sensors, continued work on a filtering/detection system that can locate blocks on like-colored backgrounds, implementation of a communication protocol to permit the vision board and the control board to communicate, and the inevitable work involved in overcoming an unforeseen obstacle including but not limited to changes in the rules of the competition, part failure, and the ever-imposing timetable.

## Documentation

None yet

## Appendix A

### Sample Mavric II "Blinkenlights" Code

```
#include <avr/interrupt.h>
#include <avr/io.h>
#include <avr/signal.h>
#include <inttypes.h>

/* make sure to test this value each time, don't optimize away! */
volatile uint16_t m_cnt;

/*****
 * msleep() - sleep for n milliseconds
 *****/
void msleep(uint16_t n)
{
    TCNT0 = 0;
    m_cnt = 0;
    while (m_cnt != n); /* wait for timer to update to desired period */
}

/*****
 * millisecond ISR
 *****/
SIGNAL(SIG_OUTPUT_COMPARE0)
{
    m_cnt++;
}
```

```
/******  
 * set T0 to interrupt every millisecond  
*****/  
void init_timer(void)  
{  
    /*  
     * set up the t0 output compare interrupt and set the t0  
     * OC reg for a 1 millisecond count to trip the ISR  
     */  
    TIFR  |= _BV(OCIE0); /* enable OC for T0 */  
    TCCR0 = _BV(WGM01)|_BV(CS02)|_BV(CS00); /* CTC, prescale = 128 */  
    TCNT0 = 0; /* clear count */  
    TIMSK |= _BV(OCIE0); /* enable OC ISR */  
    OCR0  = 125; /* 1 ms */  
}  
  
/******  
 * main routine  
*****/  
int main(void)  
{  
    /* Setup T0 ISR for 1 ms */  
    init_timer();  
  
    /* Enable interrupts */  
    sei();  
  
    /* enable PORTB 1 as an output */  
    DDRB = 0x01;  
  
    /* Loop forever */  
    while (1) {  
        ms_sleep(500); /* period = 1 second, so 1/2 a second */  
        PORTB ^= 0x01; /* XOR B0 to toggle LED */  
    }  
}
```