DESIGN REPORT

A.L.I.C.E. ROBOT

Nicholas Stocks
(stoc2874@vandals.uidaho.edu)
Brandon Bitseff
(bits2983@vandals.uidaho.edu)
Donald Bellevue
(bell7827@vandals.uidaho.edu)
Alexius Pinkham
(pink2149@vandals.uidaho.edu)
Daniel Schneider
(schn5072@vandals.uidaho.edu)

Design Report

Rabbotix

5/7/2015

Doctor Janet Rachlow and Courtney Conway,

The following is a summation of the Senior Capstone project you sponsored to build a device that would map and explore subterranean animal burrows. This report is intended to provide details on our design process, as well as to provide a foundation for any further development of the prototype. Herein you will find a statement of the problems addressed in the final design, a timeline of the project's milestones, descriptions of the final design, an explanation of concepts considered but not included and recommendations for work to be done in the future.

Regards,

Donald Bellevue, Nick Stokes, Brandon Bitseff, Alex Pinkham, Daniel Schnieder

-Team Rabbotix

Contents

Executive Summary

Our capstone project was to design a mechanism to take video, map, and measure the volume of subterranean animal burrows; pygmy rabbit, and burrowing owl specifically. This is a robot that can be controlled remotely by a laptop. It features two cameras to allow the user visual feedback for navigation in both forward and reverse. The forward camera is also equipped with a digital video recorder, which can save a high resolution video to a micro SD card. The compact system fits comfortably down a 3.5" diameter hole, allowing it to investigate the small tunnels of pygmy rabbit burrows. Our design includes an array of range finders, an accelerometer, and a pair of motor encoders. These devices, are capable of collecting data which can be used to generate a three dimensional map of the tunnels the robot navigates. This map also provides the volume of the burrow. This data will be vital to fulfilling our client's needs. These abilities could also be utilized to explore other small spaces, and places unsafe for human presence.

Background

The purpose of the project taken on by the team Rabbotix is to create a tool to aid in the research of pygmy rabbits and burrowing owls in Southern Idaho. Both of these animal species are currently on the endangered species list, and their numbers continue to decline. As their habitat is destroyed to make room for an ever expanding agricultural industry, it becomes increasingly important to study both species and their impacts on the environment and their respective ecosystems. By studying these animals and their burrows more closely, we can begin to understand to what effect removing them from the environment may cause.

Problem Definition:

The ultimate goal of our project is to create a remote control vehicle that can enter a pygmy rabbit, or other subterranean animal, burrow. Once inside the burrow live video, area measurements, and distance measurements will be collected and can then be used to generate a 3D map of the burrow. Software to control the device from a laptop will also be provided

The team analyzed data provided to us by our client to determine our constraints. The major constraint is that the device has to fit inside of the average burrow with a diameter of 4 inches. Given the need to be able to navigate within the confined space we determined that a mechanism diameter of 3 inches would be optimal.

Using the data provided by Dr. Janet Rachlow, on the distance between burrow entrances we decided our tether for communications should be 50 feet long and the run time of the device should be at least 1 hour to provide ample time for recording data from burrows.

Project Plan:

The main task at the start of our project was project learning. The team looked into burrow dimensions, computers, sensors, drive trains, communications, force requirements, and power requirements to make selections and designs that would meet our desired specifications. Once we were far enough along with our project learning we moved on to mechanical design. Purchased part selections were finalized and solid models were created for the fabricated parts. With the parts purchased and built we then assembled the device and went on to start testing. We created a Gantt chart to help us schedule our project. (Shown in table 1) Team member's responsibilities are as follows:

Alex: provides design decision documentation, and indicators of performance to further enhance system design. He works alongside Nick and Daniel in performing mechanical design analyses and manages the part-file database for CAD related materials. Alex is also charged with graphic design for team logos and merchandise.

Brandon: works on the computer controls expected in our system. He is required to make final selections in electrical components to ensure compatibility in hardware and software. Brandon supplies firmware used in data collection, motion control and video stream support.

Dan: aids Nick and Alex in mechanical analysis, and works with Brandon to select and integrate hardware. He is also responsible for machine shop activity and a large portion of prototype fabrication, as well as mechanically critical hardware considerations.

Don: is tasked with ensuring team productivity lay in line with the client's desires and demands. He does so by relaying information back and forth with the client, and making it available for the rest of the team to use. He is also charged with maintaining the budget and ordering parts.

Nick: works alongside Alex and Daniel to supply the team with quality mechanical design analyses used to influence design decision making. Nick is also responsible for keeping time estimates for each member of the group, and monitoring progress of the project with respect to milestones.

Concepts Considered

Initial designs concepts began with the research of similar exploration robots. Notable concepts found were the pyramid exploring robots used to navigate small crevasses and holes inside the pyramids. Nearly all used multi track systems, up to six, that adjusted on either side so that the tracks could expand against the top and bottom of the passages to gain traction within the passages. The Upuaut – 2, Pyramid Rover, Tomb Trekker and Chaos all carried this design. One notable exception to this design was the Djedi. The Djedi was a four wheeled device that inched along a bit like a caterpillar, being able to extend and shorten while pulling a small trailer with various measuring equipment.

The multi-track design was a highly favorable choice to go with as most the robots with this design were specifically for navigating small holes. Advantages with this design were that traction would not be an issue. The multi tracks system provides plenty of traction along with the ability to use both the top and bottom of the hole if even more traction was needed. However, design flaws with these designs when applied to our application of a rabbit burrow ended up outweighing the benefits.

The design flaws for a multi-track design started with the size. The robots used to explore the pyramids were typically about 2-3 feet long and close to a foot in height. This allowed the designs to carry the complex multi-track system. All used a tether of some sort for controlling the robots and they were only really meant to travel in a straight line but did have the ability to turn, again the length of the robots really wouldn't let them turn in tight situations. Lastly the costs of building a robot with this design would have been very expensive and fell far above our given budget. The combination of all these design flaws led us to use another design choice.

The design ideas the team came up with were adaptations of current styles of robots or vehicles. Some of the ideas proposed were the tank, skid steer, hinge, and car designs. Important considerations had to be made on size, maneuverability, traction and costs. (As shown in table 3)

The hinge style design proposed would have been a four wheeled device that used two main bodies that pivoted at a central point. The idea was that this would give easier maneuverability around tight corners. Traction would be on the lower end of what we demanded and costs were estimated to be higher than what we had.

The tank and skid steer are practically the same idea. Only difference in what we defined a skid steer is that the tracks did not encompass the entire body of robot. Advantages towards our application were highest with this design. Having two tracks satisfied our

concerns with traction and the additional considerations of adding more weight would further increase this factor. The ability to turn on a single axis fulfilled our concerns with tight corners within the burrow.

Concept Selection

I. Drive Train

The final selection for our drive train was two full body-length treads located on the lower half of the robot, and powered by two gear motors. The goal of our design was to have a very low center of gravity and enough power to maneuver in the expected terrain both forewords and backwards. We had considered multiple options before deciding upon the current drive train design.

One alternative we considered was using wheels instead of treads. An advantage of wheels, over treads, is accessibility and ease of use. We assumed that most users of the robot would have at least some experience driving wheeled vehicles, so this design would be familiar to many people. One major problem with a wheeled design is its wide turning radius when compared to treads utilizing a skid-steer system. We decided that a non-zero turning radius would compromise our ability to maneuver in the tight confines of the rabbit burrows. We also determined that the increased traction of treads would provide more accurate distance measurements than wheels by providing minimal slip. Given the pros and cons we determined that a tread system would be the best option.

After the team had decided that treads were the better design option, the location of the treads on the vehicle was the next design hurdle. In the earliest stages of development, one of the team's goals was to build a robot that could continue operation even should the vehicle tip over. The thought was that if the vehicle flipped over in the burrow that the user could invert their controls and continue driving. However, this would mean that the treads would have to reach up over the top of the vehicle, which would require the body of the robot to be much smaller to compensate. The team studied the chances of the robot flipping over completely, and concluded that it was much more difficult than initially thought. These two facts lead the team to prioritize lowering the center of gravity on the robot and choosing a smaller tread design.

We also considered using a worm gear to power the vehicle's drive train. The thought was that with the limited space in the vehicle, the worm gear would take up less space than a traditional motor and gear box. The worm gear would also give us plenty of power to move

the vehicle up and down the entrances. However, the worm gear design has a large disadvantage when moving in reverse, and being more complex. The group concluded that the ability to move backwards was necessary in the burrow environment because the vehicle may not always have the ability to turn around. Thus, the gear motor and gearbox design was chosen, and saving space in the chassis for the larger design was given a higher priority.

II. Range Finders

Designing the vehicle to find accurate volume measurements was an important goal for the project. The final design uses five infrared range finders to measure the vehicle's distance from the burrow walls, and two wheel encoders to measure the distance the vehicle has traveled. This allows the user to later calculate the volume of the burrow system.

At first the team considered using ground penetrating sonar, but the high gravel content of the burrow sites made this a very difficult tool to use.

Another idea the team considered was to use mechanical "whiskers" to measure the volume. The whisker design used five metal fins attached to the front of the robot to physically touch the walls of the burrow. The deflection of the fins would be measured by potentiometers, and the robot would be able to tell how far away from each of the walls it was at any time. The first problem with this design is that the burrows vary in size from about three inches in diameter to about eighteen inches in diameter. Whiskers would be unable to measure the diameter of the tunnel at every location, and would require more space for the system due to the number of potentiometers the design would require. The second problem with the whiskers is that if the robot needed to travel backwards through the burrow, the whiskers would dig into the walls and stop the vehicle. The Infrared design is essentially whisker design, which has none of the drawbacks but all of the benefits.

III. Control Tether

From the very beginning, the team struggled with the problem of communicating with the robot. At first it seemed that wireless communication would be preferred. However, the soil of the burrow environment would not let a strong enough wireless signal penetrate deep enough into the ground to effectively control the robot and receive video information. The solution was to find a cable that was small enough that it could be pulled behind the

robot, but robust enough that the signal would not get lost while traveling to or from the vehicle. The team considered using fiber optic cable for our data transmission, however the computer would require extensive programing in order to send or receive information over fiber optics. This lead the team to what is now the final design. The control tether selected is a fifty-foot twin axial cable measuring 2mm in diameter. This cable is durable enough to survive being towed behind the vehicle, but also light enough that the vehicle would not need extra power in order to drag it.

IV. Power Supply

The final design challenge for the project was the vehicle's power supply. The final design reserved almost half of the interior space of the vehicle for an on-board battery. The battery has a one hour lifespan and outputs enough power to run all of the computing equipment and motors on the vehicle. Before choosing the battery, the team considered using a power cord to power the robot. The power cord design would give the robot an essentially endless run time, but it would also be very heavy and require larger motors in order to be pulled through the tunnel. With motor space already being given a very high priority, the team compared the size requirements of two larger motors verses using a battery pack. In the end, the battery pack ended up taking less internal space, and was thus chosen to be included in the final design.

System Architecture

Our final design should be able to record a series of data points to an onboard memory storage device for later processing. This data will correlate to the three dimensional mapping of burrow systems. The design should also be able to stream live video feed to a remote client, where an operator can manually control the exploration of the system. The design should provide high enough video quality for tasks such as counting eggs.

This design involves five major components: an array of infrared range finders, forward and reverse cameras, a pair of motor encoders, the communications tether, and the microcontroller. The purpose of the range finders is to record the distance from the sensor to the wall of the burrow. These data points are stored on a thumb drive, logging the potentially non-uniform diameter of the tunnel for use in the volume, and three dimensional map evaluation. The pair of motor encoders provide the values for the distance traveled, which are also stored on a thumb drive, to be used for the remainder of the volume computation, and contribute to the three dimensional mapping. The

microcontroller is used to control the motor controllers, as well as manage and transmit the desired data to memory, and a remote client. Forward and reverse cameras supply the visual information desired by our clients, streaming a video to the remote client, and saving a higher resolution video on board. A forward and rear camera are required because the length of our robot reduces the odds of successfully turning around inside the burrows. The final key component is our communications tether. The tether directly transmits the video stream to a control system, as well as transferring data between the control system and the microcontroller. The block diagram is show in figure (1).

I. Microcontroller

The microcontroller, an Olinuxino IMX233 Nano, is the main element tying together the components of our device. It is able to receive data from the sensors and the control system while sending data or commands to the motor controllers, video switch, and control system. The communication to and from the remote client is sent via one of the microcontrollers Universal Asynchronous Receiver/Transmitter (UART). The main control program then takes the commands received and executes a command or moves the motors accordingly. (Example code shown in appendix) The commands for the motor controller are sent through the microcontrollers other UART. Once the features are coded the microcontroller will also be able to receive data from the motor encoders through general purpose input/output pins as well as from the Infrared sensors and Accelerometer via an Inter-Integrated Circuit (I^2C) Bus. Also the microcontroller will be able to switch which camera output is sent to the tether through the video switch using Pulse Position Modulation (PPM) signals.

Future Work

The future of this project is very coding intensive. We were unable to mount the infrared range finders, LEDs, accelerometer, or implement code for a video switch and encoder distance. In order to achieve the end goal of the project these tasks need to be completed. We suggest a group primarily consisting of electrical engineers and computer scientists to consolidate the many breakout boards, finish and improve coding, generate a streamlined user interface. Some suggestions for GUI improvements are the ability to switch the active camera, initiate and terminate the video recording remotely, an artificial horizon to provide better spatial awareness, and a real time map of explored areas. We also suggest generating a method for three dimensional mapping, and volume computation for non-technical operators.

Appendix

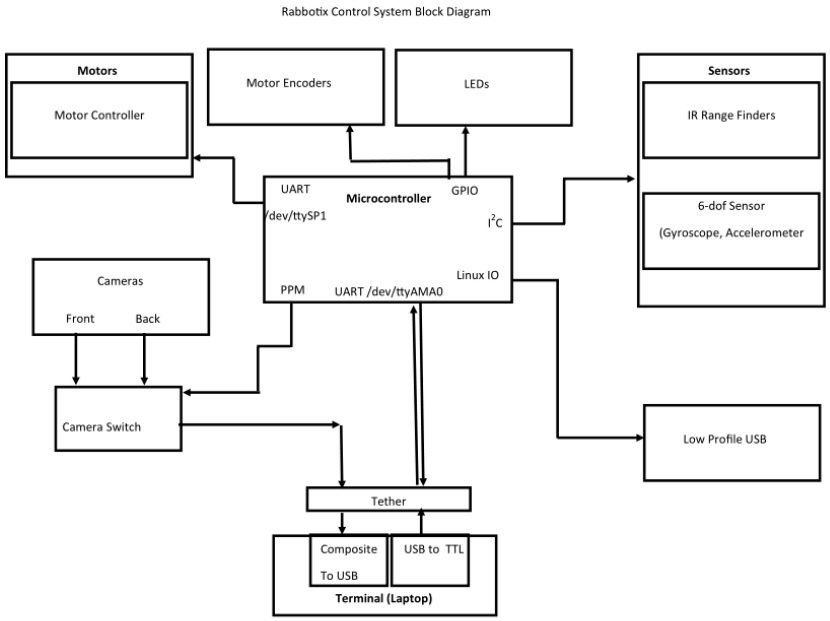

Figure 1 Control System Block Diagram

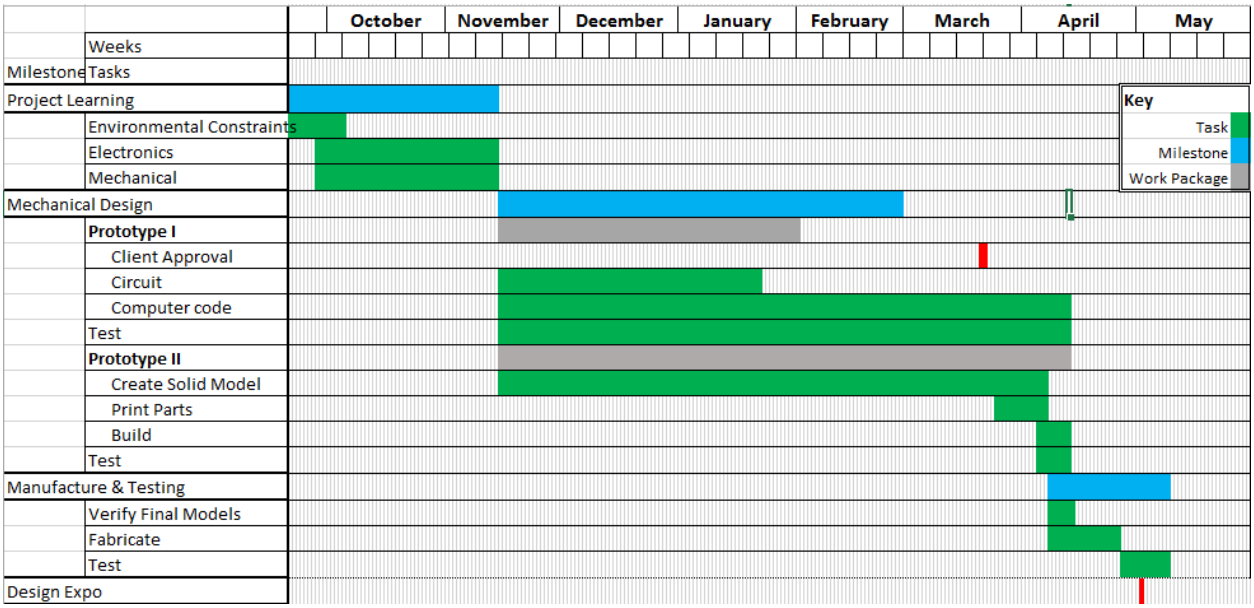

Table 1 Project Timeline

Parameter	Metric	Target Value	Acceptable Value	Actual Value
Device Size	Diameter in inches	3"	3.5"	3”
Distance from User	Feet	50’	10’	50’
Minimum Sensor Distance	Millimeters	0mm	10mm	0mm
Maximum Sensor Distance	Centimeters	20cm	10cm	10cm
Top Speed	Meters per Second	0.25	0.25	0.25
Video Quality	Resolution	1280x720	640x480	1280x720
Run-Time	Hours	1	1	1
On-Board Data Memory	Giga Bytes(GB)	8	4	16
On-Board Video Memory	Giga Bytes(GB)	32	32	32

Table 2 Project Specifications

Ranking of Importance (1-5)	Design Trait	Relative Weight of Trait	Ranking of traits for each design (1-5)

Multiplier	Thing	Weight %	Skid Steer	Hinge	Chaos	Tank	Worm
3.5	Traction	10.9375	3	3	4	4	2
3.5	Simplicity of Use	10.9375	4	3	2	4	1
3	Power Consumption	9.375	4	3	3	4	1
1	Speed	3.125	3	4	3	3	1
4	Durability	12.5	4	3	3	3	1
3	Manufacturability	9.375	4	4	3	4	1
1.5	Bio Impact	4.6875	4	5	4	4	2
4	Cost	12.5	4	3	3	4	1
4	Maneuverability	12.5	4	3	4	4	2
4.5	Cargo Space	14.0625	3	5	2	3	1
32		**Score:**	371.875	350	303.125	370.3125	128.125

Table 3 Drive Train Decision Matrix

B:\Downloads\Docs\Capstone\Code\main.c Thursday, May 07, 2015 2:10 PM

```
/************************************************************
* main.c - Rabbotix main control program
* Receives commands from /dev/ttyAMA0 serial and then
* Reacts mainly by sending commands over /dev/ttySP1
* to the motor controller
************************************************************/
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <string.h>
#include <termios.h>
#include <time.h>
#include "uart.h"
#include "motor_controller.h"
#include "gpio-mmap.h"

void set_back_file(void);

int main()
{
char sinput[3];
int i;
int md;
int leds_on = 1;
    gpio_map();
    init_motors();
    gpio_output(0,7);
    stop_motors();
    Serial_Open(&md, "/dev/ttyAMA0", B115200);
    Serial_SendChar(&md, 'R');
    Serial_SendChar(&md, 'e');
    Serial_SendChar(&md, 'a');
    Serial_SendChar(&md, 'd');
    Serial_SendChar(&md, 'y');
    while(1)
    {
        sinput[0] = 0;
        do
        {
        Serial_Read(&md, sinput, 1);//Wait for input
        }while(sinput[0] == 0);
        switch(sinput[0])//React to input
        {
        case 'w'://F1
            move_motors(FORWARD,HALF_SPEED,FORWARD,HALF_SPEED);
            break;
        case 's'://Re1
            move_motors(REVERSE,HALF_SPEED,REVERSE,HALF_SPEED);
            break;
        case 'a'://L1
```

-1-

Example Code pg. 1

B:\Downloads\Docs\Capstone\Code\main.c Thursday, May 07, 2015 2:10 PM

```
            move_motors(REVERSE,HALF_SPEED,FORWARD,HALF_SPEED);
            break;
        case 'd'://R1
            move_motors(FORWARD,HALF_SPEED,REVERSE,HALF_SPEED);
            break;
        case 'W'://F2
            move_motors(FORWARD,FULL_SPEED,FORWARD,FULL_SPEED);
            break;
        case 'S'://Re2
            move_motors(REVERSE,FULL_SPEED,REVERSE,FULL_SPEED);
            break;
        case 'A'://L2
            move_motors(REVERSE,FULL_SPEED,FORWARD,FULL_SPEED);
            break;
        case 'D'://R2
            move_motors(FORWARD,FULL_SPEED,REVERSE,FULL_SPEED);
            break;
        case 'u'://FL1
            move_motors(FORWARD,(HALF_SPEED/2),FORWARD,HALF_SPEED);
            break;
        case 'h'://FR1
            move_motors(FORWARD,HALF_SPEED,FORWARD,(HALF_SPEED/2));
            break;
        case 'l'://ReL1
            move_motors(REVERSE,(HALF_SPEED/2),REVERSE,HALF_SPEED);
            break;
        case 'n'://ReR1
            move_motors(REVERSE,HALF_SPEED,REVERSE,(HALF_SPEED/2));
            break;
        case 'U'://FL2
            move_motors(FORWARD,HALF_SPEED,FORWARD,FULL_SPEED);
            break;
        case 'H'://FR2
            move_motors(FORWARD,FULL_SPEED,FORWARD,HALF_SPEED);
            break;
        case 'L'://ReL2
            move_motors(REVERSE,HALF_SPEED,REVERSE,FULL_SPEED);
            break;
        case 'N'://ReR2
            move_motors(REVERSE,FULL_SPEED,REVERSE,HALF_SPEED);
            break;
        case 'Y'://Stop
            stop_motors();
            break;
        case 'p'://Back Camera
            set_back_file();
            break;
        case 'P'://Back Camera
            leds_on = !leds_on;
            GPIO_WRITE(0,7,leds_on);
            break;
        }
        sinput[0] = 0;
```

Example Code pg. 2