HONORS UNDERGRADUATE RESEARCH: AUTONOMOUS ROBOT FOR REMOTE DETECTION OF UXO

By

Joshua Galloway

A prototype autonomous robot controlled via IEEE 802.11b/g for the purpose of aiding in the removal of remnant landmines and unexploded ordinance submitted in partial fulfillment of the requirement of the University Honors Program at Southern Polytechnic State University and the Bachelor of Science degree in the Electrical and Computer Engineering Technology Department.

Southern Polytechnic State University
4 December 2008

Approved by:

_____	_____
Faculty Advisor:	Department Chair:
_____	_____
Honors Com. Rep.:	Honors Director:

NOTICE TO BORROWERS

In presenting this thesis or capstone paper as a partial fulfillment of the requirement of the University Honors Program of Southern Polytechnic State University, I agree that the college library shall make it available for inspection and circulation in accordance with regulations governing materials of this type. It is understood that any copying from this document must be done in accordance with proper citations and that any potential use of this thesis or capstone paper for financial gain will not be allowed without written permission of its author.

The author of this thesis is:

Joshua Galloway
1955 Bells Ferry Rd. Apt# 4231
Marietta, GA 30066

The director of this thesis or capstone paper is:

Daren Wilcox
Electrical and Computer Engineering
Technology Department
Southern Polytechnic State University
Marietta, Georgia 30060

Borrowers of this thesis or capstone paper not regularly enrolled as students at Southern Polytechnic State University are required to indicate acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis or capstone paper for the use of their patrons are required to see that the borrower records here the information requested.

Name of Borrower	Address	Date

ACKNOWLEDGMENTS

Special thanks to the Electrical and Computer Engineering Technology Department Heads Association of the American Society of Engineering Education and its Chair Ronald Land for their financial support of this research through the endowment of a $1000 mini-grant.

Also, I would like to recognize the following people for their contributions, guidance, and support in helping me to complete this thesis and my degree at Southern Polytechnic State University:

Professor Charles Bachman, Professor Daren Wilcox, Professor Charles Duvall, Dr. Tom Fallon, Dr. Nancy Reichert, My Mother Andrea Redd, My Father David Galloway, Grandma and Grandpa Galloway, Grandma and Grandpa Andrus, My Sister Abby Galloway, My Aunt Deborah Galloway, My Late Uncle Frank Andrus, Ian and Christensen Family, Jennifer Ramsay, and finally, God.

ABSTRACT

The Electrical and Computer Engineering Technology (ECET) Honors student developed a prototype for an autonomous landmine and unexploded ordinance (UXO) seeking robot. The system provided functionality including locating metallic landmines and UXO within a defined area/environment, recording the location of said landmines and UXO's, and storing the data off unit via an IEEE 802.11b/g connection to a Windows or Linux-based personal computer. The project afforded the student the opportunity to explore the field of robotics with the added benefit of providing experience with mechanical systems, printed circuit board (PCB) layout/production, and network programming in Java. Application of the prototype and corresponding research may lend themselves to de-mining the more than 100 landmine/unexploded ordinance affected countries in the world particularly in desert terrain (US Department of State Fact Sheet, 2 July 2003).

Honors Undergraduate Research:
Autonomous Robot for Remote Detection of UXO

PREPARED BY
Joshua Galloway

In partial fulfillment of the requirements for
ECET 4904

ELECTRICAL AND COMPUTER
ENGINEERING TECHNOLOGY
DEPARTMENT

SOUTHERN POLYTECHNIC STATE
UNIVERSITY

TABLE OF CONTENTS

1. INTRODUCTION

The United Nations estimates that 2,000 people are killed or maimed by mine explosions each month, and for every mine cleared, twenty more mines are laid. Of the people affected, a vast majority are civilians[1]. Most landmines and unexploded ordinance (UXO) are removed by trained personnel wearing ballistic armor probing the ground with a baton, which puts the personnel at an unsafe distance from the explosive. This task is made less dangerous through the application of robotics and other technologies. However, these technologies are of considerable expense, and since most of the landmines and UXO are in the Global South, expense is a great concern[2]. The goal of the research preformed by the student was to arrive at an inexpensive prototype that could autonomously sweep a field and record the location of any metal detected. Because of diverse locations in which landmines have been deployed, a single terrain was selected based on the ubiquity of landmine and UXO deployment in each affected environment. The robot was designed around this selection.

2. DEFINING THE PROBLEM

2.1 Types of Landmines

In order to define a detection system, a study of the types of landmines deployed was necessary. What follows is a description of the most preponderant types of landmines.

Figure 1: Anti-Tank Mine[3]

- Anti-Tank Mine—A landmine designed to be buried and to produce a large upward explosion capable of destroying a large vehicle. These mines are usually

[1] United Nations, "Landmine Factsheet" [Online Document], 1997 Sept., [cited 2008 Nov. 22], Available: http://www.un.org/cyberschoolbus/banmines/facts.asp
[2] United Nations, "Countries Affected by Landmines", [Online Document], 1997 Nov., [cited 2008 Nov. 22], Available: http://www.un.org/cyberschoolbus/banmines/resources/affected.asp
[3] How Stuff Works, "How Landmines Work" [Online Document], [cited 2008 Nov. 22], Available: http://science.howstuffworks.com/landmine.htm/printable

designed to trigger at a pressure higher than a person would provide so that the mine is less likely to be triggered by something other than a vehicle[4].

Figure 2: Blast Mine[5]

- Blast Mine—A landmine designed to be shallowly buried and to produce an upward explosion for the purpose of wounding the victim and is triggered by pressure provided from above[6].

Figure 3: Fragmentation Mine[7]

- Fragmentation Mine—A landmine designed not to be buried that shoots metal fragments at the victim and is usually triggered by a trip-line. This mine is typically lethal within 50 meters and dangerous within 100 meters[8].

[4,6,8,10,12,13] Robert Keeley, "Understanding Landmines and Mine Action" [Online Document], 2003 Sept., [cited 2008 Nov. 22], Available: http://www.minesactioncanada.org/techdocuments/UnderstandingLandmines_MineAction.pdf

[5] Andy Smith, "AP Blast Mine Factsheet R2M1/2 and Variants" [Online Document], 2002, [cited 2008 Nov. 22], Available: http://www.ddasonline.com/R2M2factsheet.htm

[7] Wikipedia, "Anti-Personnel Mine", [Online Document], 2008 Nov., [cited 2008 Nov. 22] Available: http://en.wikipedia.org/wiki/Anti-personnel_mine

Figure 4: Bounding Mine[9]

- Bounding Mine—A fragmentation mine that is designed to be buried with several detector spikes extending from the earth. Upon disruption of a spike, the fragmentation mine is fired around two feet into the air, after which, the fragments are deployed in a horizontal fashion. These mines may also be setup to be triggered by a trip-line[10].

Figure 5: Unexploded Ordinance[11]

- Unexploded Ordinance (UXO)—Bombs, grenades, bullets, shells... that did not explode during deployment. These weapons may be triggered by almost any disturbance, or not at all[12].

2.2 Affected Areas

Landmines can be found in many countries throughout the world. Most of these countries are the underdeveloped or developing states of the Global South. Table 1 shows the top eight countries affect by remnant landmines and UXO's.

[9] Wikipedia, "PROM-1", [Online Document], 2008 April, [cited 2008 Nov. 22], Available: http://en.wikipedia.org/wiki/PROM-1
[11] Luke Powell, "Demining (Afghanistan)",[Online Document], [cited 2008 Nov. 22], Available: http://www.un.org.pk/latest-dev/afg-gallery2/index.htm

Table 1: Countries Affected by Landmines[2]

County	Estimated Number of Landmines
Egypt	23,000,000
Iran	16,000,000
Angola	15,000,000
Afghanistan	10,000,000
China	10,000,000
Iraq	10,000,000
Cambodia	6,000,000
Vietnam	3,500,000

3. AVAILABLE SOLUTIONS

3.1 Prodders and Manual Removal

Figure 6: Prodder[14]

The most widely used method of landmine removal is by use of a simple prodder in tandem with a metal detector. This method is very slow and exceedingly dangerous. A person is dressed in protective gear and sweeps the field with a metal detector. When a metal object is found, the de-mining personnel use a prodder to feel for hard objects that may be a landmine. If an object that may be a landmine is discovered, personnel carefully dig the device out, and it is removed for destruction. While this method is

[14] University of Western Australia, "Demining Research", [Online Document], 2000 Jan., [cited 2008 Nov. 22], Available: http://www.mech.uwa.edu.au/jpt/demining/tech/prodder/harc99.html

dangerous and time consuming, it is still the most inexpensive and reliable means of landmine removal[13].

3.2 Mechanical Methods

Figure 7: Mechanical De-mining Device[16]

Mechanical methods of de-mining are usually aimed at actuating the landmine. Machines use flailing chains, large spiked rollers, and other similar means to detonate the mines. The problems associated with this solution are cost and effectiveness. Machines stout enough to withstand repeated explosions from both anti-personnel and anti-tank mines are extremely expensive, and then, they can be only 80% effective in some cases[15].

3.3 Robotic Methods

Due to the high level of danger involved in the prodding method and the high cost of the mechanical methods, a robotic solution that could utilize the skills of the de-mining personnel at a safe distance would be highly beneficial. However, teleoperated robots, the likes of which are used by bomb squads and police forces, are still too expensive for most humanitarian de-mining teams to acquire[17].

[15,16,17,21] DeTeC, “Demining Robots, Anti-Personnel Mine Removal”, [Online Document], 1997 Nov., [cited 2008 Nov. 22], Available: http://diwww.epfl.ch/lami/detec/rodemine.html

4. SYSTEM LEVEL DESIGN

The robot, in order to contend with the problems of cost and effectiveness in available solutions, should be a system that is inexpensive and performs a portion of the hand prodding process. With the problem of cost in mind, it was decided that to produce a robot that would be able to locomote in all the varied terrains that are affected by landmines and UXO's was not viable. Therefore, the robot was designed for application in a specific region to keep cost at a minimum. Also, any data the robot acquired should be stored off-unit due to the possibility of destruction of the robot.

4.1 Locomotion

4.1.1 Terrain Selection. In order to select a locomotion system for the robot, it is necessary to decide on the terrain the robot must negotiate. The environment that the robot is tailored for should be one of the most landmine-affected environments. Using the countries from Table 1, the types of terrain present in each country was tabulated.

Table 2: Type of Terrain in Affected Countries

Country	Type of Terrain			
	Sand	Grassy Field	Rice Paddy	Rocky/Rough
Egypt	X			
Iran	X			X
Angola		X		X
Afghanistan	X	X		X
China	X	X	X	X
Iraq	X			
Cambodia		X	X	X
Vietnam		X	X	X

Taking the information from Table 2 and weighting each "X" in an ad hoc method, a numerical score for each type of terrain was derived. Each country's weight-constant, C_N, was arrived at by dividing its total number of landmines by the country with the largest total landmines.

C_N = *(Estimated # of landmines in the country)/23,000,000*

The score for each terrain was derived summing the total weights of the countries with and "X" in a particular terrain's column.

$Score = \sum C_N * X_N$

Here is an example calculation.

$Score_{Rice\ Paddy} = C_{China} + C_{Cambodia} + C_{Vietnam} = 0.435 + 0.261 + 0.152 = 0.848$

Table 3: Terrain Scores

	Sand	**Grassy Field**	**Rice Paddy**	**Rocky/Rough**
Adjusted Score	3.000	1.935	0.848	2.630

As a result of the tabulation in Table 3, sandy terrain should be the medium in which the robot will have the greatest impact. It was noted that Egypt is geographically in keeping with this choice, and it also has the most total remnant landmines of any country. As a result, the types of landmines in Egypt were researched with the aim of discovering by what means the robot could detect these landmines and UXO. It was found that the Western Desert of Egypt, the state's most affected region, is populated with mines from the World War II era. In addition, the majority of the types of mines deployed have sufficient amounts of ferrous metal in them as to be detected by a simple metal detector[18].

4.1.2 Locomotion System Selection. Three types of systems were considered: a two-wheeled system with a support, a four-wheeled system, and a pedal system.

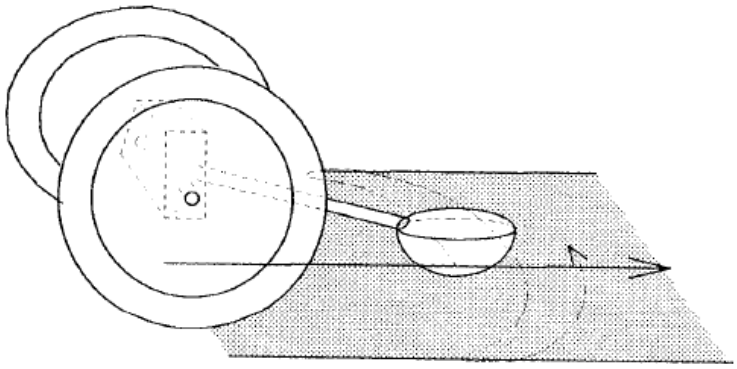


Figure 8: Two-Wheeled System with Support[19]

The two-wheeled system has the advantages of being simple in design, very maneuverable, and inexpensive. The system was found wanting in its possible applications because the support may become stuck in soft sand and certainly would not be applicable to even moderately rough terrain.

[18] United Nations Mine Action Service, "Mine Action Assessment Mission Report", [Online Document], 2000, [cited 2008 Nov. 22], Available: http://www.mineaction.org/downloads/Egypt_Assessm_Report.PDF
[19] Nicoud, J.D., Habib, M.K.,"The Pemex-B Autonomous Demining Robot: Perception and Navigation Strategies", [Online Document], 1995 Aug., [cited 2008 Nov. 22], Available: http://ieeexplore.ieee.org/iel3/3952/11432/00525830.pdf?isnumber=11432&prod=CNF&arnumber=525830&arSt=419&ared=424+vol.1&arAuthor=Nicoud%2C+J.D.%3B+Habib%2C+M.K.

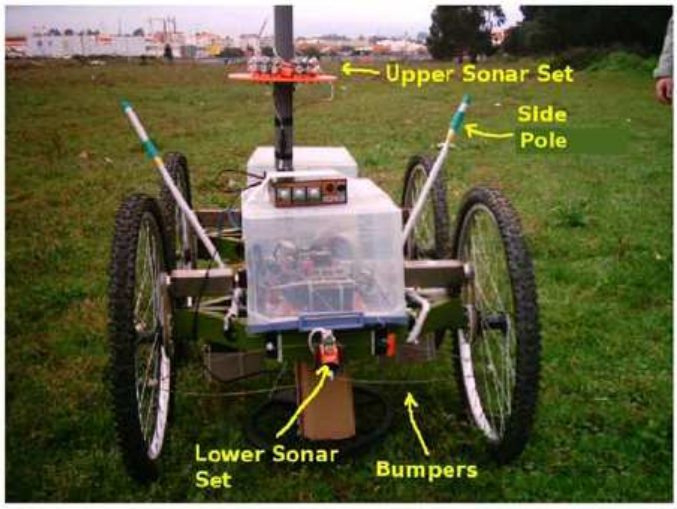


Figure 9: Four-Wheeled System[20]

The four-wheeled system is a very solid system, and unlike the two-wheeled system, it would not be hindered by small pot-holes or slightly rough terrain. These attributes will make the system easily portable to other environments. The only striking disadvantage is that the system's turning radius may be comparatively large, which could hinder maneuverability.

Figure 10: Pedal System[21]

The pedal system is exceedingly maneuverable and can negotiate irregular terrain well. However, this comes with a large increase in complexity and, therefore, expense. The system will be very difficult to implement and may result in a slow-moving system. The additional motors needed to create the dexterous movements necessary for any selected gate will result in a relatively large power consumption.

The systems were all rated numerically from one to five (one being poor, and five being excellent) for the expected system mobility and performance in each of the four types of terrain.

---

[20] Santana, Barata, Correia, "Sustainable Robots for Humanitarian Demining", [Online Document], 2007, [cited 2008 Nov. 22], Available: http://www.ars-journal.com/International-Journal-of-Advanced-Robotic-Systems/Volume-4/ISSN-1729-8806-4205.pdf

Table 4: Locomotion Systems Rating

Type of System	Types of Terrain			
	Sand	Grassy Field	Rice Paddy	Rocky/Rough
Two-Wheeled with Support	3	4	1	1
Four-Wheeled	4	4	3	3
Pedal	2	3	4	4

From the Table 4 and the types of landmines and UXO's found in the selected territory of the Western Desert of Egypt, it is clear that the system should be of the four-wheeled type. The four-wheeled system is easily implemented, maneuverable enough for a desert environment, and low in power consumption.

4.1.3 Locomotion Implementation. To rapidly and inexpensively create a prototype robot, the student used the chassis, motors, and H-bridge of a radio controlled (RC) car. A New Bright 1:8 Radio Control Full Function 4 Door Jeep Wrangler Unlimited was selected as the full implementation of the locomotion system based on size and cost ($60). In a developing country, such a system would be easily obtained and inexpensive. The decorative parts were removed and the car's electronics were tested in order to find the front and rear H-bridge's control lines. A mechanical drawing of the stripped RC car is given in "Appendix A."

4.2 Control System

4.2.1 Control System Selection

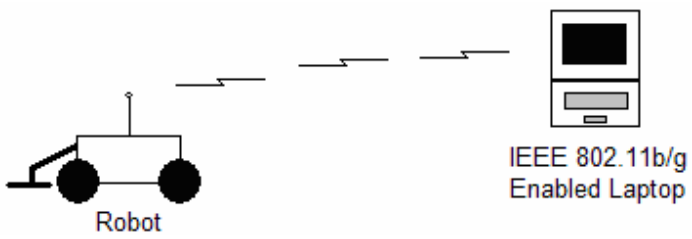


Figure 11: Robot Control Scheme

Because of the possibility that the robot may be destroyed, it was decided to keep as much of the system expense off-unit. The scheme settled upon was to control the robot via IEEE 802.11b/g wireless local area network standards to link a small microcontroller to a personal computer. This allows the system to be composed of readily available networking technology and puts only an inexpensive microcontroller and wireless router on the unit. Also, destruction of the robot during a sweep will not result in a loss of previously acquired data, as it will be stored in the laptop. A block diagram of the proposed system is shown in Figure 12.

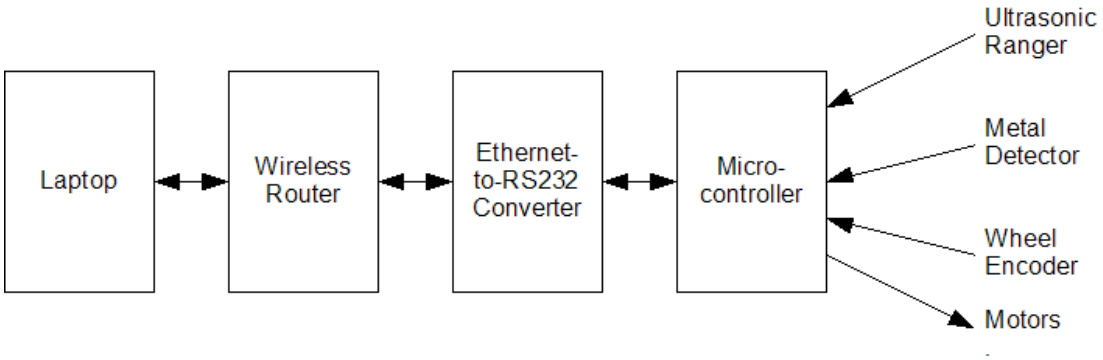


Figure 12: Control Block Diagram

4.2.2 Control System Implementation. The various parts of the control system were implemented as follows.

- Laptop—This was implemented with an Acer Aspire 1640 IEEE 802.11b/g enabled laptop computer with Windows XP and Ubuntu 8.10 installed in a dual-boot setup. However, any computer that can support an accessory to provide the necessary IEEE 802.11 compliance, and load Java's runtime environment will be sufficient. The target system would be on Asus Eee PC or similar low cost laptop. Cost: $200-$250.

- Wireless Router—This was implemented with a Linksys WRT54GT IEEE 802.11b/g router. Its firmware was upgraded to the newest version available (version 4.30.12). Cost: $60.

- Ethernet-to-RS232 Converter—A Maxport ethernet-to-RS232 converter available from Comfiletech was used to realize this portion of the control scheme. Again, the firmware was upgraded to the newest available version upon receipt (version xpt03_6602). Cost: $75.

- Microcontroller—The PIC18F4520 was the microcontroller selected for the robot's input-output (IO) functions and to interface the Maxport. It was selected because it provided many of the functions necessary to interface the robot's sensors, which allowed for a smaller control-logic footprint on the unit. Cost: $4.

4.3 Sensors

4.3.1 Sensor Selection. To keep the cost and power consumption of the unit low, the number of sensors was kept to a minimum. The robot needed to detect the ferrous metal present in the targeted, World War II-era landmines and UXO. An inexpensive solution such as a metal detector will serve this purpose. Also, the robot needed a way to detect objects in its path, so an ultrasonic ranger was necessary. In addition to the ultrasonic ranger, a web-camera was added to allow the unit to be teleoperated and possibly take pictures of the offending areas during an autonomous sweep. Finally, the robot must be able to map its exact location relative to an arbitrarily selected point. This point was

selected as the starting point of the sweep, and odometry was selected as the means of mapping its location.

4.3.2 Sensor Implementation. The sensors were implemented as follows.

- Metal Detector—A Slinky Treasure Tracker Metal Detector was eventually settled upon based on cost. The unit was probed until a suitable control signal was found, and then, it was interfaced to the microcontroller. Cost: $30.

- Ultrasonic Ranger—MaxBotix's LV-MaxSonar-EZ4 was used to implement this sensor. The ranger was reputed to be able to detect objects from 6 inches out to 254 inches. The beam width is 2 feet, and the refresh rate is 20 Hz. The selected interface to this device is analog. Cost: $30.

- Web-Camera—The web-camera selected was the Trendnet TV-IP100 RJ45 Internet Camera Server. This particular camera was the lowest-priced unit from a reliable manufacturer. Cost: $80.

- Odometry—A CTS 288 Series 4-Bit Gray Code Rotary Encoder was decided upon to implement the robot's odometry system. It was coupled to one of the rear-drive wheels via a 2.5 inch diameter hobby airplane tire. Cost: $10.

4.4 Power Distribution

4.4.1 Power System Selection. In order to keep costs low, the decision was made to utilize the RC car's provided 9.6 V nickel cadmium (NiCd) battery for the control and sensor systems with the exception of the metal detector, which was powered via a 9 V PP3 battery. The RC car's motors and H-bridges were powered by way of a separate 9.6 V NiCd battery in order to isolate the control logic from the noise created by the motors. It was necessary to generate several voltages from nominal 9.6 V batteries in order to power the system's varied devices.

4.4.2 Power System Implementation. The power distribution system was implemented as follows:

- Control Logic Battery—This was the 9.6 V battery that was provided with the New Bright 1:8 Radio Control Full Function 4 Door Jeep Wrangler Unlimited used for the locomotion system.

- Motor Battery—A 9.6 V Black & Decker Model No. GC9601SB cordless drill was disassembled for its NiCd battery and charging system. This was, again, done to curb cost, as the drill, with its charger included, was far less expensive than a traditional NiCd battery and charger. Cost: $30.

- Metal Detector—This was powered via a PP3 9 V battery.

- Microcontroller and Remaining Sensors—This portion of the robot was powered through the control-logic battery and a LM7805 linear 5 V, 1 A regulator. This portion of the system drew a total of 250 mA at 5 V.

- Linksys Router—This was powered by way of the 9.6 V control-logic battery and a LM2577-12 step-up switching regulator. The router drew 500 mA at 12 V.

- Trendnet Web-Camera—The camera drew 2.5 A at 5 V, so it was necessary for the device to be put on its own 5 V regulator powered through the control-logic battery. A 5 V, 3 A, linear, LM323 regulator was used to provide the requisite voltage and current.

- Drive Motors and H-Bridge—These devices were all powered by way of the motor battery; no regulation was needed.

4.5 Network Configuration

The network was setup for static IP addresses as follows:

Table 5: Network Configuration

Network Node	IP Address
Linksys Router	192.168.10.1
Maxport Ethernet to RS232 Converter	192.168.10.15
Trendnet Web-camera	192.168.10.30
Acer Aspire 1640 Laptop PC	192.168.10.101

No security or encryption was implemented, and all the default passwords and usernames were left intact. They are given in Table 6.

Table 6: Administrative Usernames and Passwords

Device	Username	Password
Linksys Router	(leave blank)	admin
Maxport	(leave blank)	(leave blank)
Trendnet Web-camera	admin	admin

4.6 Assembled System

Figure 13: Assembled Robot Top and Side View

The system, displayed in Figure 13, was assembled from the parts, which were outlined in this section. Assembly took seven hours to complete in total. The metal detector and router are held in place by way of screws and custom made supports. The web-camera is also attached to a custom formed support, which was fashioned from a scrap piece of aluminum. The PCB's are held in place by hot glue.

5. HARDWARE SYNTHESIS

Custom electronics were designed to synthesize the systems. The student implemented the design by way of three custom printed circuit boards (PCB's) created by the toner-transfer method.

5.1 Control, Locomotion, and Sensor Synthesis

The PIC18F4520 directly interfaced all the sensors, with the exception of the web-camera. A liquid-quartz display (LCD) was added to the project to allow for ease of debugging, and the PIC18F4520's in-circuit serial programming (ICSP) was utilized along with its in-circuit serial debugging (ICSD) capabilities. Figure 14 is the schematic for the control system's interface to all the sensors and motors.

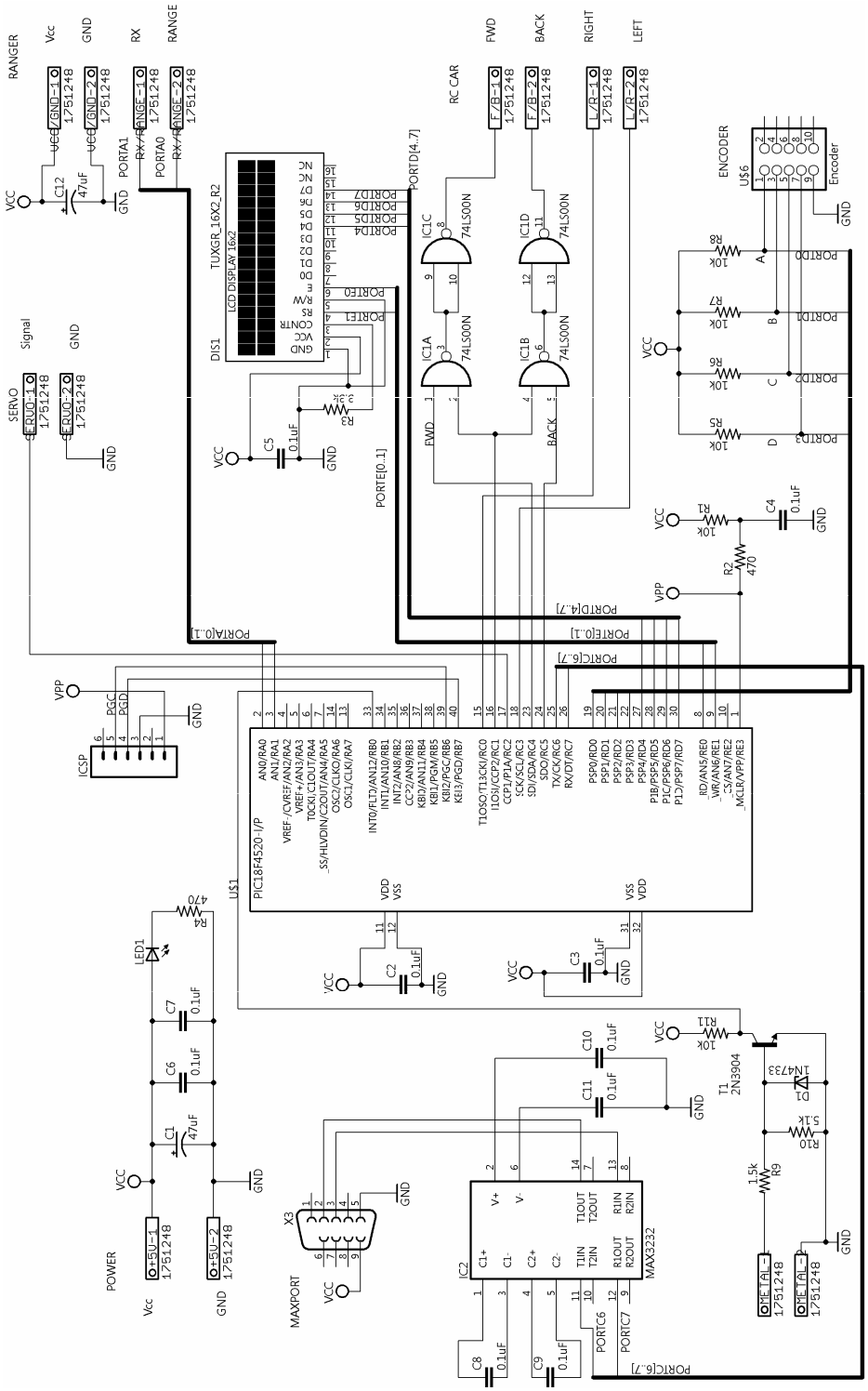


Figure 14: Control, Locomotion, and Sensors Interface Schematic

Each piece of hardware was interfaced as follows:

- Maxport—The levels coming from the Maxport's serial port were standard RS-232 levels. Thus, a level translation integrated circuit (IC) was used to interface with the PIC18F4520's universal asynchronous serial receiver transmitter (UART). The IC used to perform the level translation was an ICL3232—a MAX3232 compatible replacement IC. The interface was one stop bit, 9600 baud, and no flow control.
- LV-EZ4 Ultrasonic Ranger—The interface to this device was chosen to be analog, as the device may be interfaced serially, through pulse-width-modulation (PWM), or an analog voltage. The device's analog output generates (Vcc/512) Volts per inch, and was tied directly to the PIC18F4520's analog-to-digital converter (ADC).
- CTS 288 Series Rotary Encoder—The mechanical rotary encoder was interfaced to the PIC18F4520's Schmidt-triggered inputs on PORT D. Pull-up resistors were used, and consequently, the encoder's signals were all inverted.
- Motors—The front motor was controlled via two pins on the PIC18F4520; no additional parts where required. However, since the rear drive motor was a permanent-magnet dc motor, PWM was required to regulate the drive motor's speed. In order to allow for future expansion of the system to provide a physical marking on suspected landmines, one of the PIC's two PWM channels was reserved for a servo. The remaining PWM channel was multiplexed to the H-bridge through a 74HC00 configured as a simple two-channel multiplexer. Two additional pins from the PIC's where used to generate the required logic for the motor's three states: forward, backward, and stop.
- ICSP and ICSD—The programming and debugging features of the PIC18F4520 were implemented by reserving the two compulsory pins for the program clock and data as described in the manufacture's literature. Also, proper isolation from the programming voltage that was applied to the MCLR/Vpp/RE0 pin during programming was provided by a 470 Ω and 10 kΩ resistive network.
- LCD—The LCD was a Sharp-LM16255, HD44780 controlled, 16x2 character LCD. The device was interfaced in four-bit mode with the data lines on the upper nibble of PORT D, and the control lines on PORT E of the PIC.
- Metal Detector—The metal detector's output upon detection of a ferrous object was measured with a digital oscilloscope. The output waveform was found to be a series of 1.5 kHz square waves that occurred 390 ms for a duration of 390 ms with a peak value of 8.1 V. This was reduced to 5.1 V by a simple zener-diode regulator circuit, and isolated from the PIC18F4520 by a 2N3904 NPN transistor. The regulator was necessary to ensure the 2N3904's V_{EBO} of 6 V was not exceeded. Figures 15 and 16 are the waveforms generated by the metal detector. Note that in Figure 16 the 1.5 kHz is superimposed on the 390 ms high pulse.

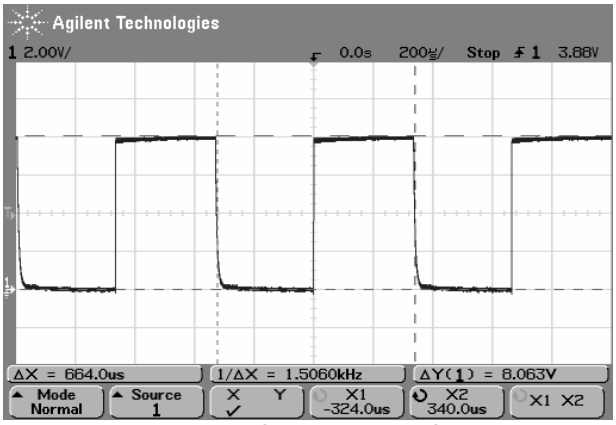


Figure 15: Close up View of Square Wave from Metal Detector

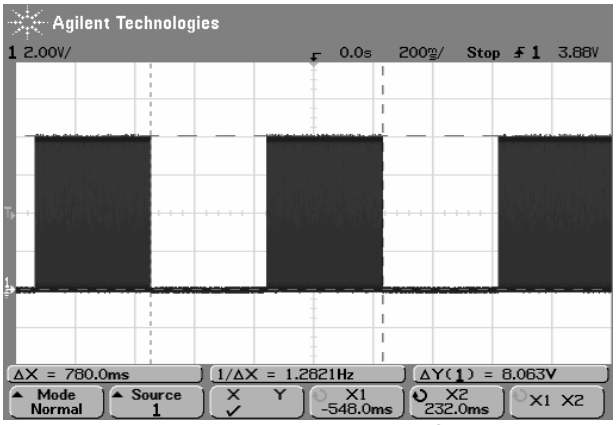


Figure 16: 390ms Pulse with 1.5kHz Waveform Superimposed

5.2 Power System Synthesis

The various regulators needed to generate the required voltages and currents for each set of devices in the control and sensor systems were created on one PCB. The power to the motors, H-bridges, and metal detector were supplied directly from each aforementioned battery without regulation. The grounds of all three supplies were connected. Figure 17 is the schematic for the control and sensor system's power.

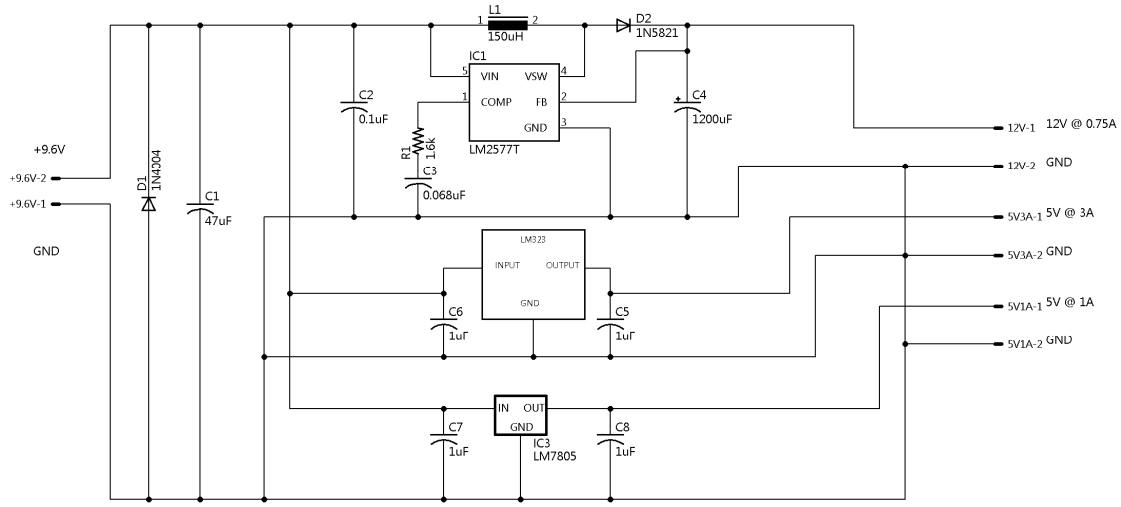


Figure 17: Control and Sensor Systems Power Supply

6. SOFTWARE

The software for the system was broken into two main parts: the PIC18F4520 program and the graphical user interface (GUI)/control software for the laptop. The PIC18F4520 software was written in microchip's proprietary PIC18 series assembler, and the laptop software was written in Java. The use of Java makes the program inherently portable to all major operating systems.

6.1 PIC18F4520 Software

The PIC's program performed three main functions: interfacing the sensors, relaying and receiving data form the ethernet interface with the laptop, and controlling the robot's locomotion.

6.1.1 Interfacing the Sensors. With the exception of the ultrasonic ranger, interfacing the sensors was trivial. The ultrasonic ranger interface was not entirely complicated, but it did require the most attention of all the sensors due to some jitter in the device's output. The analog voltage was converted to a range in inches as follows:
Given that the ranger voltage (V_R) from the data sheet is related to the range as

$V_R = (Vcc/512) / inch * (range\ in\ inches)$,

and, from the general form of a 10-bit analog to digital conversion that has been left justified in the PIC's two 8-bit

$n = [(V_A - V^-ref)/(V^+ref - V^-ref)]*2^{16}$ *noting that* $V_R = V_A$,
where "n" is the number in the PIC's ADC after a conversion,

the range value in inches may be solved for by substitution of V_R for V_A in the second equation (note that V^+ref = 5V and V^-ref = 0).

*Range in inches = n / 2^7 inches = n * 2 / 2^8 inches.*

So, the range in inches may be obtained by shifting the 16-bit number once to the left and dropping the lowest byte. Also, note that the max ranger output is 127 inches, so the carry resulting from the shift operation may also be ignored. Eight ranger measurements were taken in even intervals over approximately 440 ms and then averaged to contend with jitter in the system.

6.1.2 Executing a Move Command. In order to maintain accurate results from the odometry system, slippage of the wheels had to be minimized. This was done by applying an approximately parabolic increase in average voltage to the drive motor unit until the robot begins to move. Figure 18 is the flow diagram showing the method by which the function was performed.

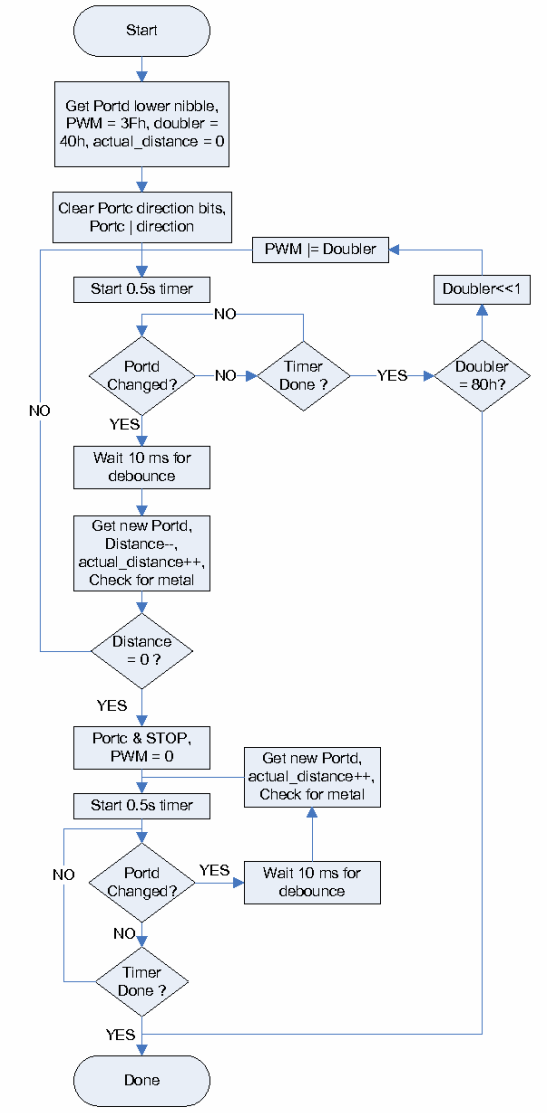


Figure 18: Motor Run Function

6.1.3 Main PIC18 Program. The main program for the PIC18F4520 utilizes a polling procedure to wait for the laptop to send a command. Upon receipt of the command, the system executes it, updates its sensor registers, and sends various status packets and acknowledgements. Figure 19 is the flow diagram.

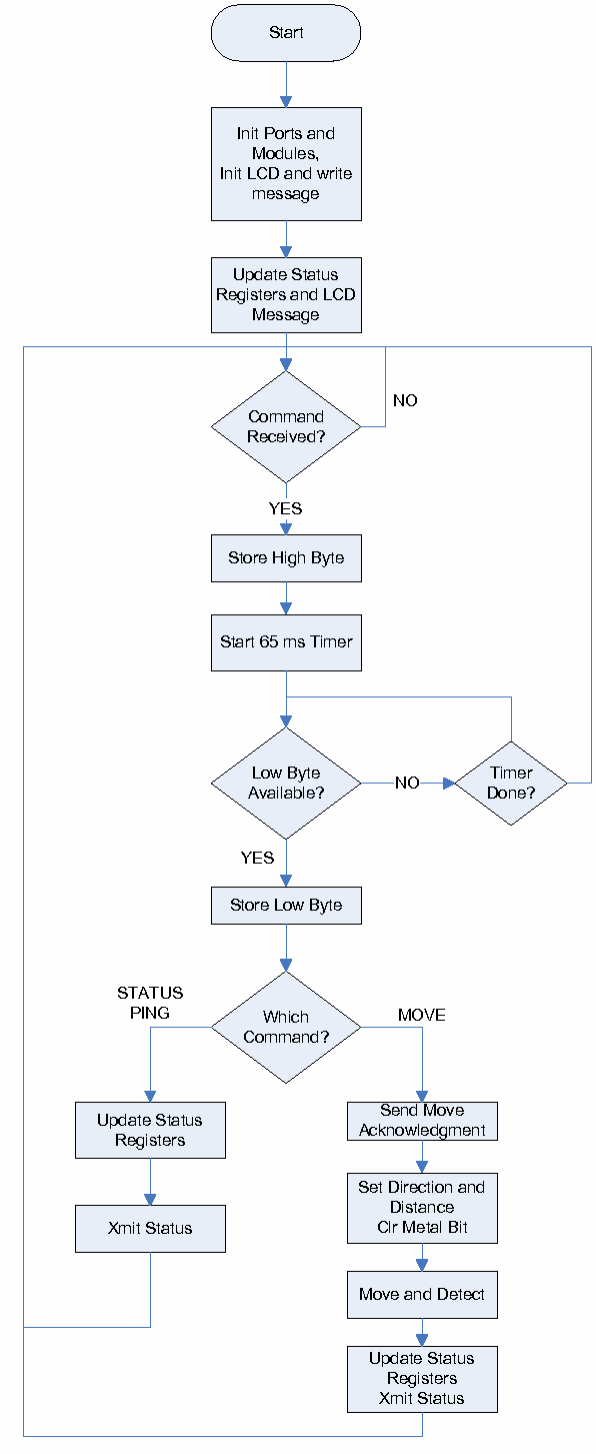


Figure 19: Main PIC Program

6.2 Java Program

The Java program used to control the robot provides a dual functionality. It performs the proposed autonomous sweeping action while storing all metal detection hits in a text file, "autonomous_sweep.txt." Also, it allows the user to manually guide the robot in manual mode. The program was compiled with version 1.4.2_16 of Java's software development kit, and it was tested with Java runtime environment j2re1.4.2_16.

6.2.1 Manual Mode

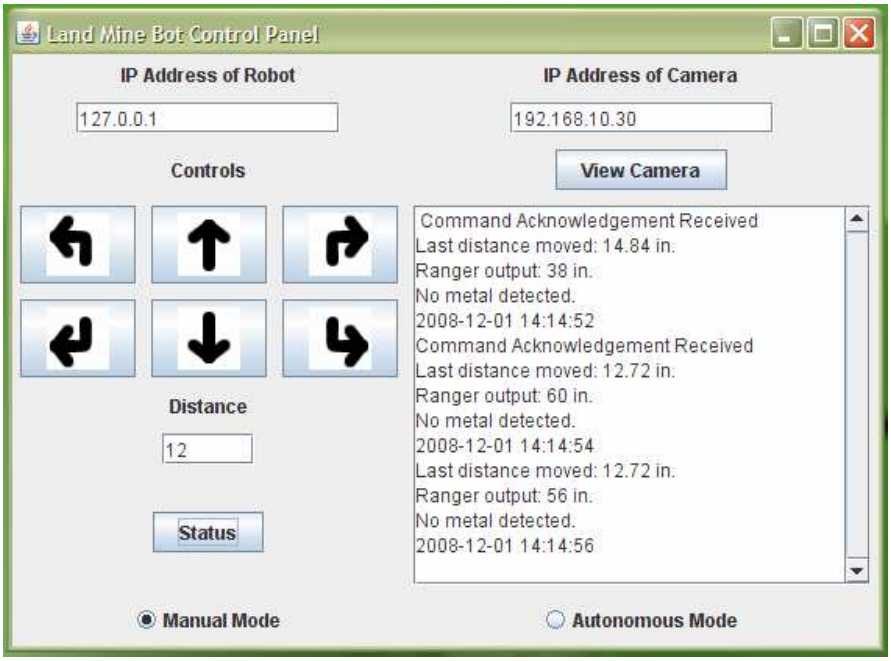


Figure 20: Manual Mode

The controls move the robot in the direction of the arrows a prescribed distance, which is entered in the text box below the controls. The status button performs a status ping to the robot and returns the value of all the status registers in the PIC18F4520. Following each command or status ping, the software updates the status with the details of each command and resulting robot feedback. The view camera button opens the computer's default internet-browser to the IP address in the appropriate box. It also records the time the browser was opened in the status window. All commands and operations in manual mode are time-stamped. Through the functionality provided by the manual mode, the robot may be used as a teleoperated metal detector with visual feedback from the robot's environment.

6.2.2 Autonomous Mode

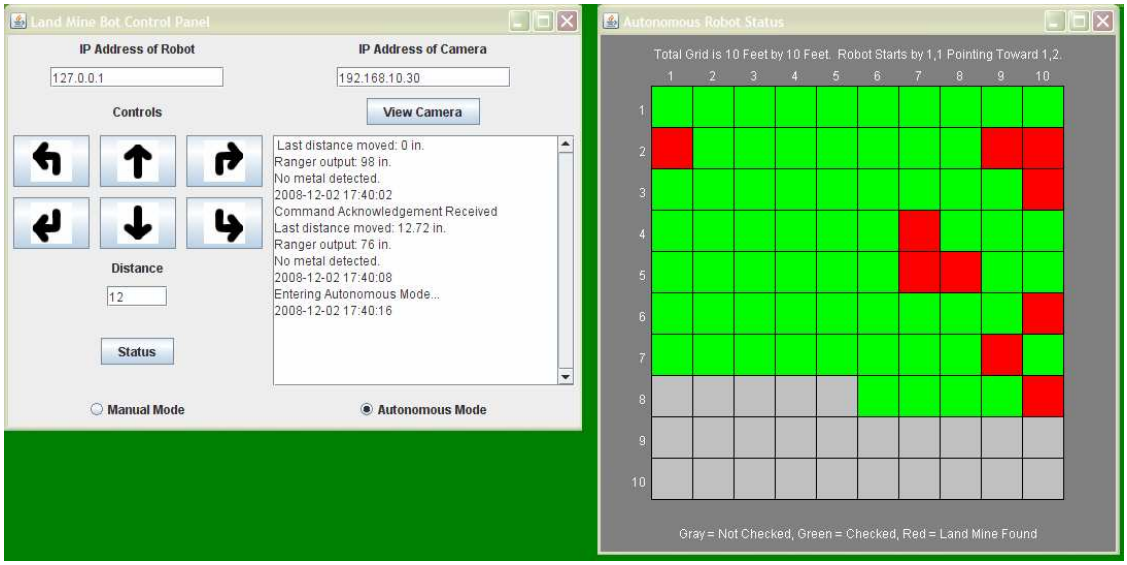


Figure 21: Autonomous Mode

The autonomous mode performs an autonomous sweep across a predefined 10 foot by 10 foot area. The system provides a progress or status map that shows an approximate location of the robot and an approximate location of any ferrous metal located during the sweep. The sweep is executed on its own thread to keep from locking the GUI interfaces. The control panel is present during the sweep, but its buttons are locked to keep from affecting the autonomous sweep. When a ferrous metal is detected, the program records the metal's location in the form of x and y coordinates in a text file called "autonomous_sweep.txt." This allows the system's user to locate any detected metal within the full available accuracy of the system.

7. COMMUNICATIONS PROTOCOL

The system, as a whole, communicates over the IEEE 802.11b/g protocol. However, a custom command set was created to interface the Java and PIC programs. This protocol included a two-byte packet for commands sent to the robot and a three-byte status message as well as a one-byte move-command-received acknowledgement from the PIC to the laptop.

7.1 Communication to the PIC18F4520

The packet received by the PIC18F4520 is composed of two bytes. The high byte (the first byte received) has two pieces of data in it: the command type, and distance to travel. In the case of a status ping, the most significant bit (MSB) of the high byte will be set. The remaining seven bits contain the distance that the robot is being prompted to move if the MSB is not set. The low byte contains the direction in which the robot should move if the MSB is clear. Figure 22 shows the data contained in the packet's payload.

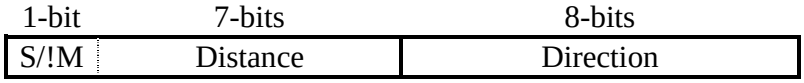


Figure 22: Command Packet

- S/!M (1-bit)—Command Type. 1 = Request status packet, 0 = Move Command
- Distance (7-bits)—The distance the robot is to move.
- Direction (8-bits)—The direction in which the robot is to move. This byte-sized value is the bit configuration that must be inclusively ORed with a cleared PORTC of the PIC18F4520 to produce the requested directional movement. Table 7 is a tabulation of the varied bit configurations and their resulting direction.

Table 7: Directional Commands

Hexadecimal Low Byte Value	Resulting Movement
0x18	Forward Left
0x10	Forward
0x11	Forward Right
0x28	Backward Left
0x20	Backward
0x21	Backward Right

7.2 Communication to the Laptop

The PIC18F4520 responds to a status pin by sending a three-byte packet to the computer, which contains the data from the PIC's freshly updated status registers. In like fashion, a move command terminates by sending data from the PIC's status registers; however, a one-byte move acknowledgement is sent prior to the execution of the move command. The move acknowledgment is 0x0F, and is sent to show that the command was received. The three-byte status packet contains information on how far from the beginning of the last move command execution metal was detected, if there was metal detected during the last move, the ranger output, and the total distance traveled during the last move command. Figure 23 is a visual of the packet's payload.

8-bits	1-bit	7-bits	8-bits
Distance to Metal	M	Range	Distance Traveled

Figure 23: Status Packet

- Distance to Metal (8-bits)—The distance form the beginning of the last executed move command to any detected metal.
- M (1-bit)—Metal found in last move. 1 = yes, 0 = no.
- Range (7-bits)—The output of the ultrasonic ranger.
- Distance traveled (8-bits)—The distance actually traveled during the last executed move command.

8. RESULTS

The robot was tested in manual mode at a nearby baseball field. The system was run through a ten foot by ten foot grid with landmine simulates, in the form of half-inch galvanized washers, scattered randomly inside the grid. Figure 24 shows the grid on which the robot was tested.

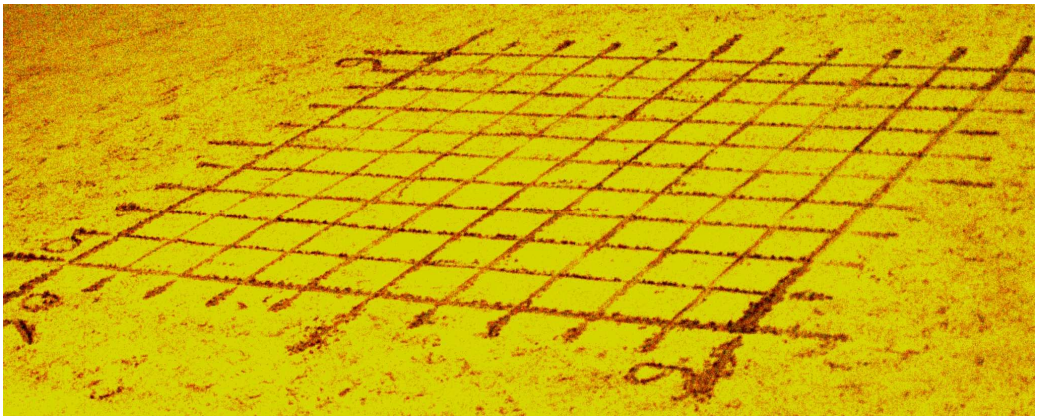


Figure 24: Test Grid

The robot was started at the point marked 0,0 of Figure 24. The sweep was performed by moving in a forward motion from 0,0 to 0,9, repositioning the robot to the next row at square 1,9, and then, sweeping in a backward motion (the robot moving in reverse) to 1,0. This process was repeated until the field had been completely covered. Figure 25 shows a screen-shot of the sweep in progress.

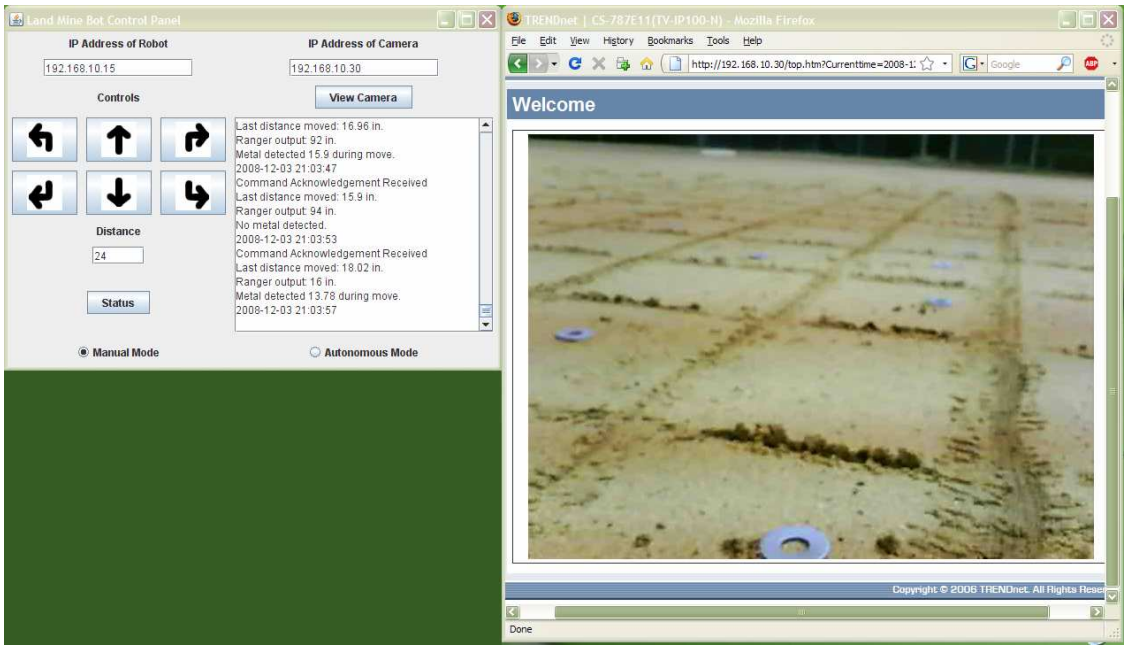


Figure 25: Sweep in Progress

The simulates were laid out as is depicted in Figure 26.

	0	1	2	3	4	5	6	7	8	9
0		X								
1			X						X	
2		X								
3	X			X			X			
4		X				X		X		
5	X					X	X		X	
6			X					X		X
7	X				X			X		
8		X		X				X		X
9	X		X	X	X				X	

Figure 26: Simulate Locations

The robot completed the sweep in ten minutes. This is less the time it took to recharge the battery on the Acer Aspire 1640 series laptop, which ran out when the robot was on square 3,2 of the matrix. The sweep results are shown in Table 8, and the original test

data is given in “Appendix C.” Figure 27 shows the spot as which the robot was when the laptop battery failed. The sweep was continued after recharging the battery, and the robot was not moved in the interim.

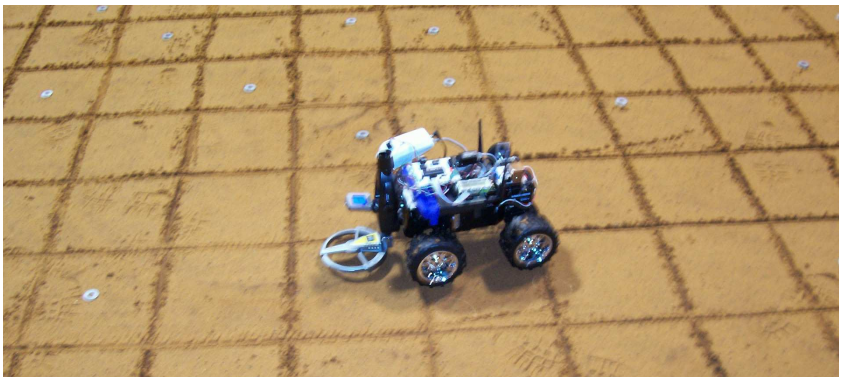


Figure 27: Robot after Laptop Battery Failed

Table 8: Sweep Results

Total Simulates	30
Total Successful Detections	27
Total Missed Detections	3
Total False Detections	0
Probability of Detection	90%

The missed detections were in squares 3,3; 5,6; and 9,3. Both the simulates in 3,3 and 5,6 were not run over by the metal detector. The simulate in 9,3 was run over but the detection was recorded in the same move sequence as the correctly detected simulate in square 9,4. So, the detection was effectively not recorded.

9. CONCLUSION

The robot was tested in manual mode with a high degree of success. The majority of the missed landmine simulates were missed because they did not pass under the robot's metal detector. This could be rectified by increasing the size of the metal detector or by decreasing the width of the rows in the search pattern. No false detections occurred during the test, and the robot never lost communication with the controlling laptop except when the laptop battery failed. This shows that the robot could easily function as a teleoperated metal detector in its current state, and it could be used to magnetically map an affected area from a safe distance.

The network portion of the Java program functioned as was intended. The system was able to communicate bi-directionally over a wireless network, and the robot performed move commands with a relatively high degree of accuracy. The autonomous portion of the Java program did not perform consistently enough in simulation to allow for feasibility of a real-world test. The simulated robot and controlling program intermittently became hung when reaching the end of a search row.

The router and wireless network functioned well under what would have been the intended circumstances. With no interference from surrounding wireless networks, the system functioned without any drop outs for the duration of the control-logic battery's charge. Stray radio frequencies (RF) in the 2.4 GHz range would not be an issue in a remote mine field. However, near the university—where the preliminary testing was performed—stray RF caused the robot to routinely lose connection with the laptop.

Several of the robot's subsystems functioned very well. The step-up regulator constructed to power the 12V router off of a 9.6 V battery worked as did all the regulators. The LCD was properly controlled by the PIC18F4520, and the ultrasonic ranger produced accurate results under most circumstances. The only failing of the ultrasonic ranger was in the student's selection of the model with the narrowest detection beam. A narrow beam is, according to the company's applications engineer, prone to malfunction when an object is in the beam's periphery.

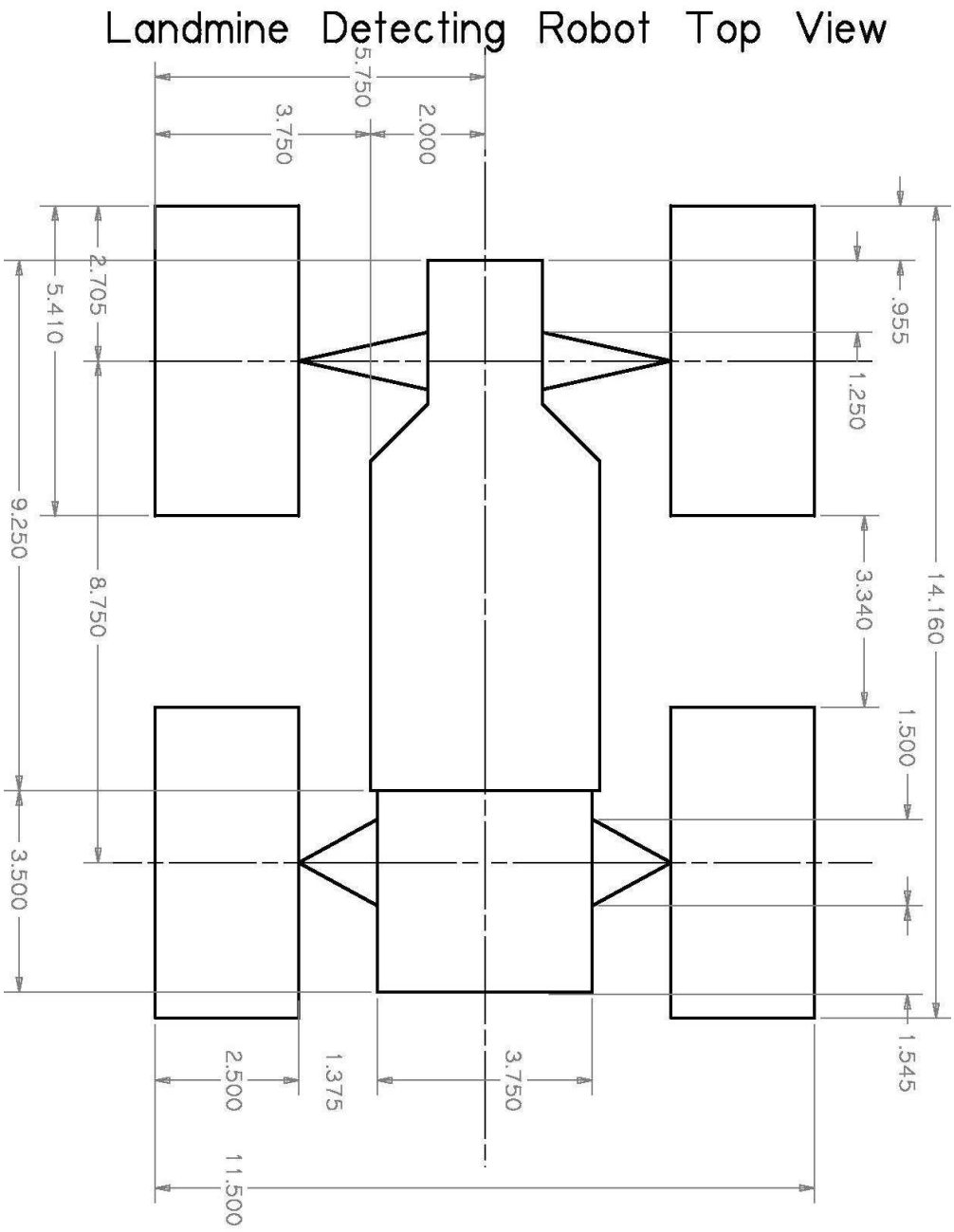
Overall, the system showed it could perform the function of sweeping a mine field under manual control, and given more time, the system could perhaps perform this function autonomously. For $250 and, if necessary, the price of a laptop, the robot could provide a landmine removal team with some idea of what awaits them in a suspected minefield without endangering any personnel. The system provides proof of concept.

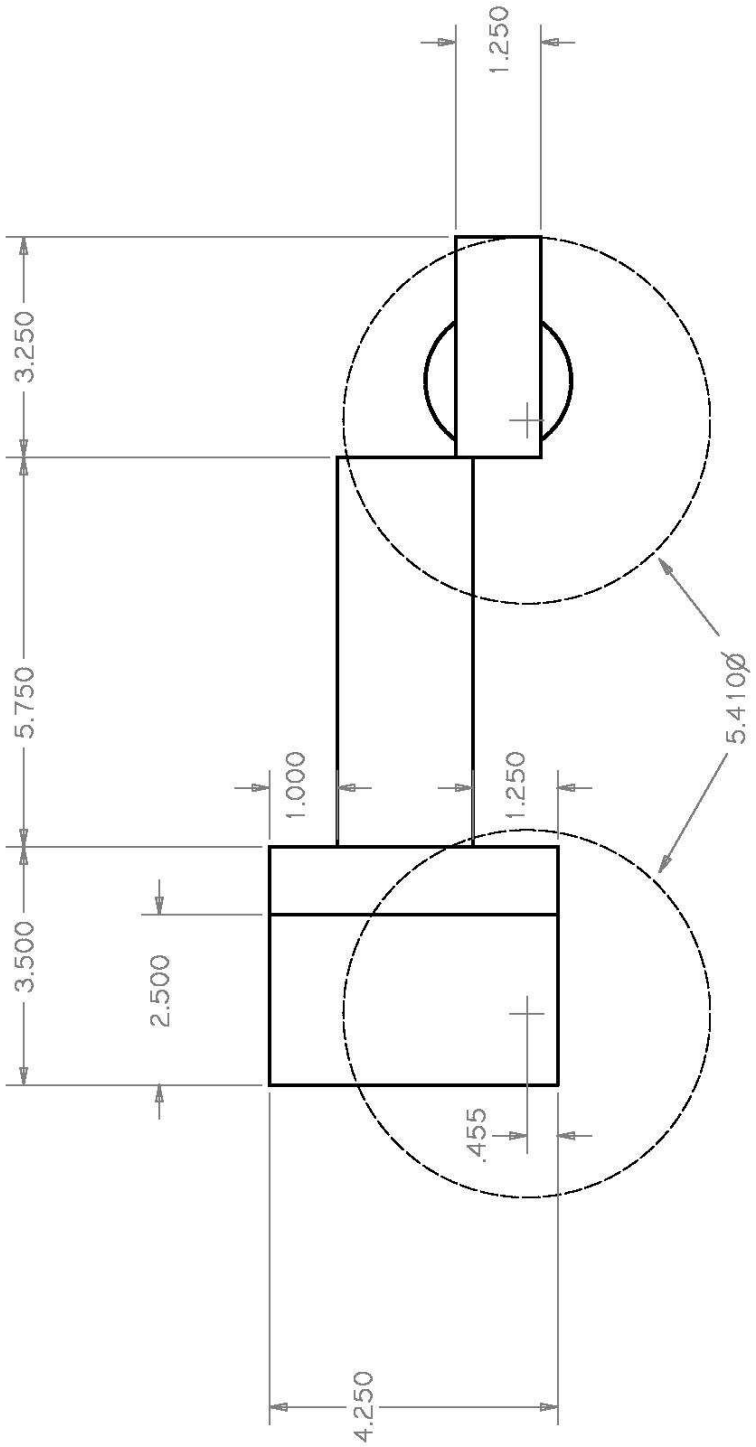
10. FUTURE WORK

Several revisions and additions would increase the system's performance, and they are as follows:

- Replace the LV-EZ4 unit with an LV-EZ2, a unit with a wider beam, to increase the accuracy of the ultrasonic ranger.
- Add a servo and marking system to the front of the robot by way of the reserved lines in the control hardware.
- Replace the control-logic battery with another battery of a higher amp-hour rating to extend the size of the field the robot can sweep.
- Develop autonomous control software.

APPENDIX A: LOCOMOTION SYSTEM MECHANICAL DRAWINGS

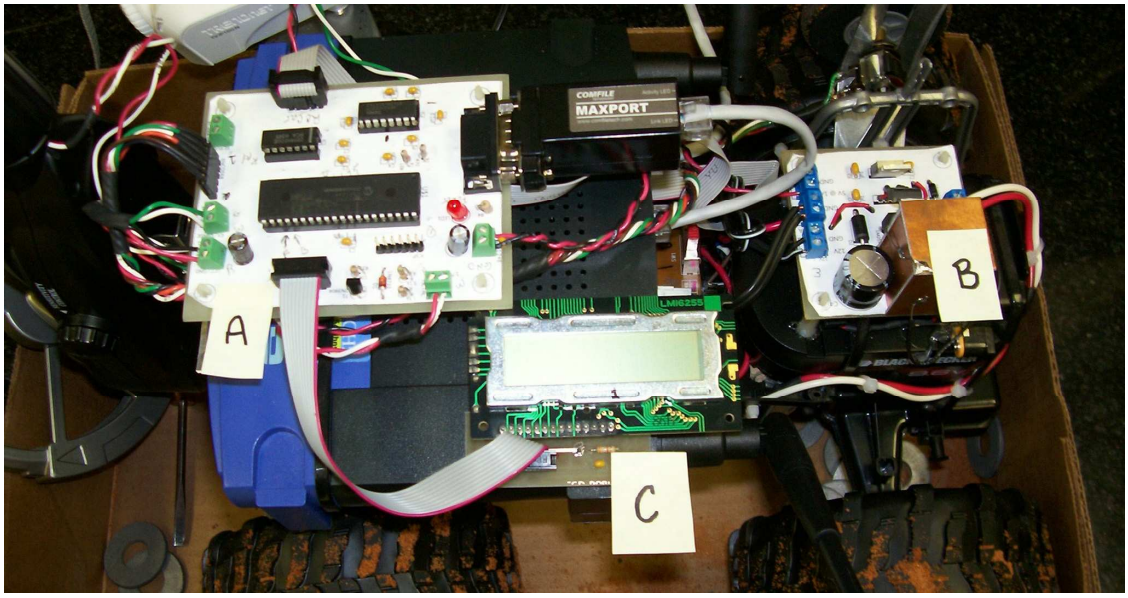




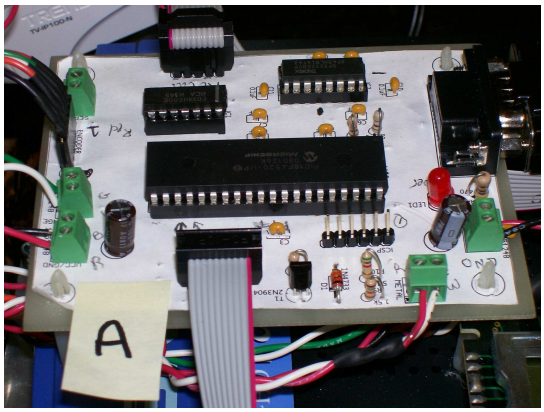
Landmine Detecting Robot Side View

APPENDIX B: CIRCUIT BOARDS

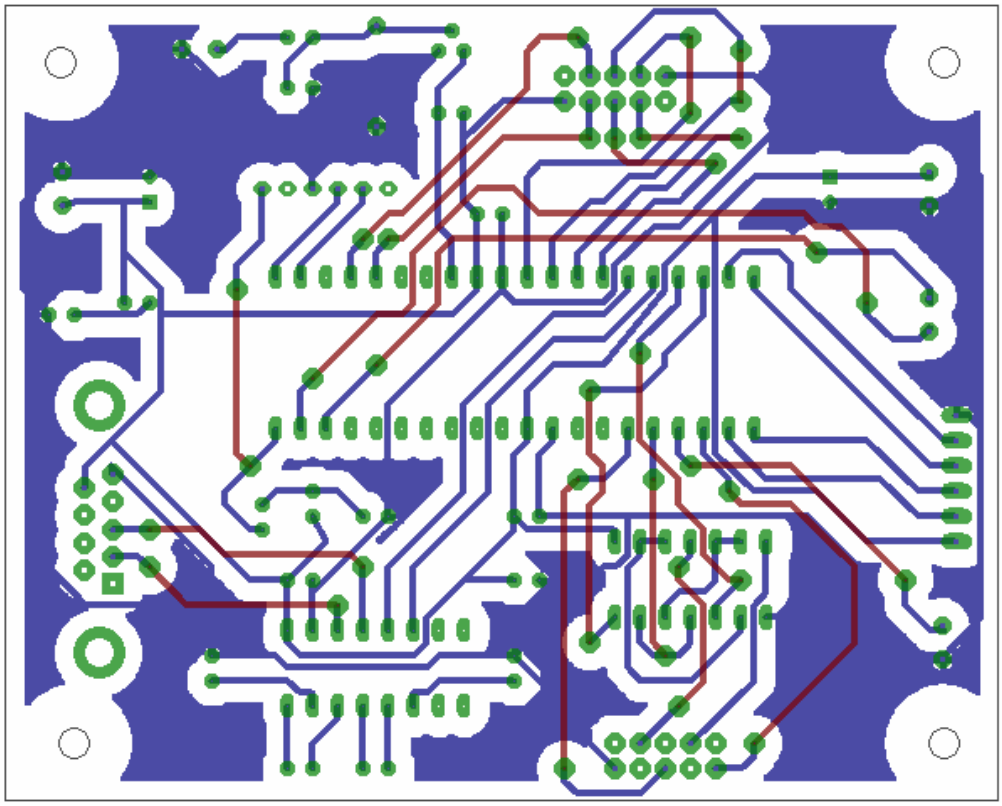
Robot Circuit Boards A, B, and C



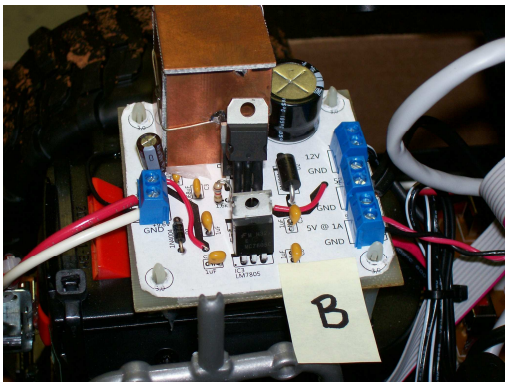
Board A: PIC18F4520 and Sensor Interfaces



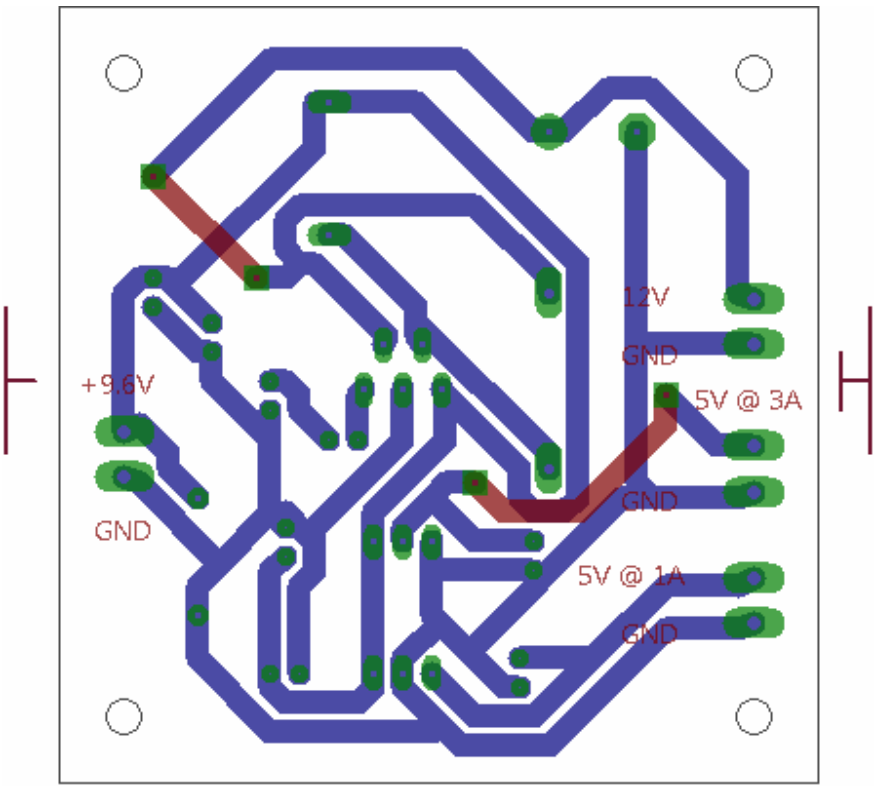
PIC18F4520 and Sensor Interfaces: Top View



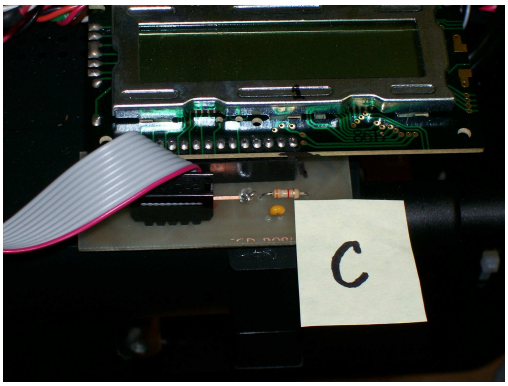
Board B: Power Distribution Board



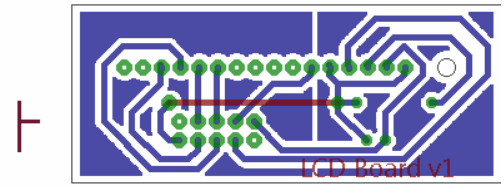
Power Distribution and Regulation Board: Top View



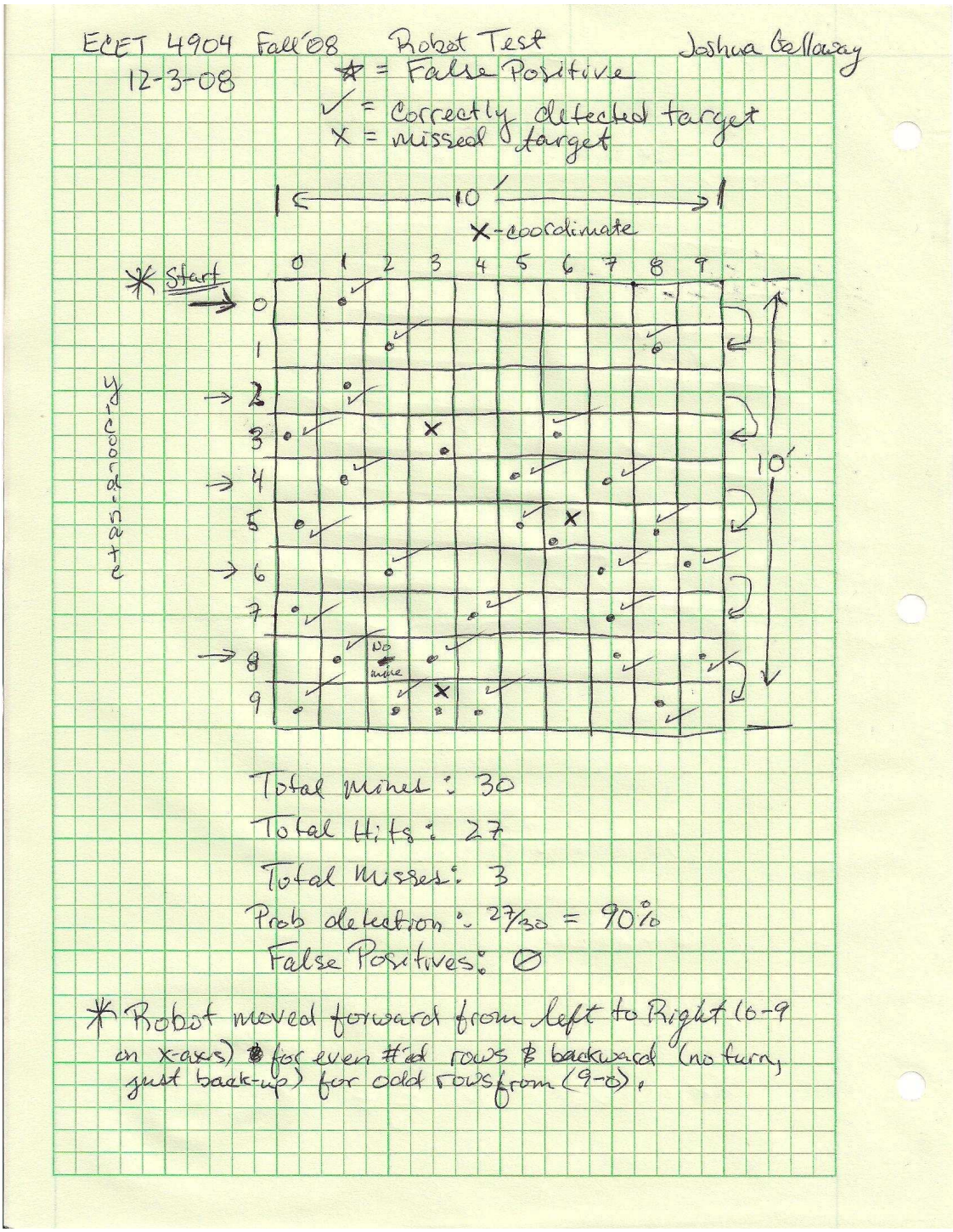
Board C: LCD Peripheral Board



LCD Peripheral Board: Top View



APPENDIX C: ORIGINAL TEST DATA



APPENDIX D: PIC18F4520 CODE

```
;*******************************************************************************
;                                                                              *
;    This program interfaces several devices in order to control a robot       *
;    over wireless 802.11b/g protocol.                                         *
;                                                                              *
;       Interfaced Devices:                                                    *
;           Maxport Ethernet to RS232 Converter (i/o)                          *
;           LV-MaxSonar-EZ4 Ultrasonic Range Finder (i)                        *
;           Sharp 16x2 Character LCD (o)                                       *
;           CTS 288 Series 4-bit Gray Code Encoder (i)                         *
;           RC car H-Bridge (o)                                                *
;           Metal Detector (i)                                                 *
;                                                                              *
;*******************************************************************************
;                                                                              *
;    Filename: landBotBrain.asm                                                *
;    Date: 10-30-08                                                            *
;    File Version: original                                                    *
;    Device: PIC18F4520                                                        *
;    Clock Frequency: 8 MHz                                                    *
;                                                                              *
;    Author: Joshua Galloway                                                   *
;    Company: Southern Polytechnic State University                            *
;                                                                              *
;*******************************************************************************
;                                                                              *
;    Files required:         P18F4520.INC                                      *
;                            18F4520.LKR                                       *
;                                                                              *
;*******************************************************************************

   LIST P=18F4520, F=INHX32 ;directive to define processor and file format
   #include <P18F4520.INC>  ;processor specific variable definitions

;*******************************************************************************
;Configuration bits
;   Oscillator Selection:
    CONFIG  OSC = INTIO67        ;Internal oscillator, I/O on RA6 and RA7
;   Fail-Safe Clock Monitor Enable bit:
   CONFIG  FCMEN = OFF          ;Fail-Safe Clock Monitor disabled
;   Internal/External Oscillator Switchover bit:
    CONFIG IESO = OFF            ;Oscillator Switchover mode disabled
;   Power-up Timer Enable bit:
     CONFIG PWRT = OFF              ;PWRT disabled
;   Brown-out Reset Enable bits:
     CONFIG BOREN = SBORDIS         ;Brown-out Reset enabled in hardware only (SBOREN is
disabled)
;   Brown-out Reset Voltage bits:
     CONFIG BORV = 3                ;Minimum setting
;   Watchdog Timer Enable bit:
     CONFIG WDT = OFF               ;WDT disabled (control is placed on the SWDTEN bit)
;   MCLR Pin Enable bit:
     CONFIG MCLRE = ON              ;MCLR pin enabled; RE3 input pin disabled
;   Low-Power Timer1 Oscillator Enable bit:
     CONFIG LPT1OSC = OFF           ;Timer1 configured for higher power operation
;   PORTB A/D Enable bit:
     CONFIG PBADEN = ON             ;PORTB<4:0>  pins  are  configured  as  analog  input
channels on Reset
;   CCP2 MUX bit:
     CONFIG CCP2MX = PORTC          ;CCP2 input/output is multiplexed with RC1
;   Stack Full/Underflow Reset Enable bit:
     CONFIG STVREN = ON             ;Stack full/underflow will cause Reset
;   Single-Supply ICSP Enable bit:
     CONFIG LVP = OFF               ;Single-Supply ICSP disabled
;   Extended Instruction Set Enable bit:
     CONFIG XINST = OFF             ;Instruction  set  extension  and  Indexed  Addressing
mode disabled (Legacy mode)
;   Background Debugger Enable bit:
```

```
     CONFIG DEBUG = OFF            ;Background debugger disabled, RB6 and RB7 configured
as general purpose I/O pins
;   Code Protection bit Block 0:
     CONFIG CP0 = OFF           ;Block 0 (000800-001FFFh) not code-protected
;   Code Protection bit Block 1:
     CONFIG CP1 = OFF              ;Block 1 (002000-003FFFh) not code-protected
;   Code Protection bit Block 2:
     CONFIG CP2 = OFF              ;Block 2 (004000-005FFFh) not code-protected
;   Code Protection bit Block 3:
     CONFIG CP3 = OFF              ;Block 3 (006000-007FFFh) not code-protected
;   Boot Block Code Protection bit:
     CONFIG CPB = OFF              ;Boot block (000000-0007FFh) not code-protected
;   Data EEPROM Code Protection bit:
     CONFIG CPD = OFF              ;Data EEPROM not code-protected
;   Write Protection bit Block 0:
     CONFIG WRT0 = OFF             ;Block 0 (000800-001FFFh) not write-protected
;   Write Protection bit Block 1:
     CONFIG WRT1 = OFF             ;Block 1 (002000-003FFFh) not write-protected
;   Write Protection bit Block 2:
     CONFIG WRT2 = OFF             ;Block 2 (004000-005FFFh) not write-protected
;   Write Protection bit Block 3:
     CONFIG WRT3 = OFF             ;Block 3 (006000-007FFFh) not write-protected
;   Boot Block Write Protection bit:
     CONFIG WRTB = OFF             ;Boot block (000000-0007FFh) not write-protected
;   Configuration Register Write Protection bit:
     CONFIG WRTC = OFF             ;Configuration registers (300000-3000FFh) not write-
protected
;   Data EEPROM Write Protection bit:
     CONFIG WRTD = OFF             ;Data EEPROM not write-protected
;   Table Read Protection bit Block 0:
     CONFIG EBTR0 = OFF            ;Block 0 (000800-001FFFh) not protected from table
reads executed in other blocks
;   Table Read Protection bit Block 1:
     CONFIG EBTR1 = OFF            ;Block 1 (002000-003FFFh) not protected from table
reads executed in other blocks
;   Table Read Protection bit Block 2:
     CONFIG EBTR2 = OFF            ;Block 2 (004000-005FFFh) not protected from table
reads executed in other blocks
;   Table Read Protection bit Block 3:
     CONFIG EBTR3 = OFF            ;Block 3 (006000-007FFFh) not protected from table
reads executed in other blocks
;   Boot Block Table Read Protection bit:
     CONFIG EBTRB = OFF            ;Boot block (000000-0007FFh) not protected from table
reads executed in other blocks

;*******************************************************************************
;Variable definitions
; These variables are only needed if low priority interrupts are used.
; More variables may be needed to store other special function registers used
; in the interrupt routines.
   CBLOCK 0x00
      WREG_TEMP      ;variable in RAM for context saving
      STATUS_TEMP    ;variable in RAM for context saving
      BSR_TEMP       ;variable in RAM for context saving
      delay          ;amount of milliseconds to delay in delayms subroutine
      lcdReg         ;data to be written to lcd
      distance       ;distance in inches to move for 'move' subroutine
      direction      ;desiered direction to move for 'move' sub
      actual_dist    ;measured distance moved in "move" sub
      doubler        ;used to double the duty cycle of pwm
      encoder        ;temporary storage for move sub
      adHigh         ; registers for mutibyte ad conversion
      adLow
      range          ; current ranger output
      num2Convert    ; binary number to be converted to packed bcd
      tens_ones      ; pack bcd for range to ascii conversion
      hundreds
      cnt1
      tempConv       ; to be able to use cpfgt
```

```
      avgLow         ; for get average range calculator
      avgHigh
      botStatusU     ; distance traveled to last metal detected
      botStatusH     ; high byte for robot status packet
                     ; metal dectected in last dist. traveled(1 bit,...
                     ; ...1 = yes),ranger output(7 bits)
      botStatusL     ; low byte for robot status packet(last dist. traveled)
      commandL       ; place for incomming command low byte
      commandH       ; place for incomming command high byte
   ENDC
;*****************************************************************************
;Constant Definitions
   #DEFINE  RANGE_RX PORTA,RA1   ;RX pin of ultrasonic ranger
   #DEFINE  METAL    PORTB,RB0   ;metal detector output
   #DEFINE  LCD_EN   PORTB,RB3   ;lcd enable pin
   #DEFINE  LCD_RS   PORTB,RB4   ;lcd register select
   #DEFINE  RIGHT_BIT PORTC,RC0  ;turn car right
   #DEFINE  LEFT_BIT PORTC,RC3   ;turn car left
   #DEFINE  FORWARD_BIT PORTC,RC4   ;car forward
   #DEFINE  BACK_BIT PORTC,RC5   ;car backward
   #DEFINE  LEFT     B'00001000' ;postion of left bit on portc
   #DEFINE  RIGHT    B'00000001' ;position of right bit on portc
   #DEFINE  FORWARD  B'00010000' ;position of FWD bit on portc
   #DEFINE  BACK     B'00100000' ;position of BACK bit on portc
   #DEFINE  STOP     B'11000110' ;constant to reset all direction bits
   #DEFINE  SPEED    CCPR2L      ;pmw for car
   #DEFINE  LCD_ON   0x0C        ;turns on just lcd
   #DEFINE  LCD_OFF  0x08        ;turns off lcd
   #DEFINE  LCD_CURSOR  0x0E     ;solid cursor displayed
   #DEFINE  LCD_BLINK 0x0F       ;blinking cursor
   #DEFINE  LINE1    0x80        ;move cursor to first line
   #DEFINE  LINE2    0xC0        ;move cursor to second line
   #DEFINE  LCD_CLR     0x01     ;clear and home lcd
   #DEFINE  MOVE_ACK 0x0F        ;move acknowledgemet

;*****************************************************************************
;Reset vector
; This code will start executing when a reset occurs.

RESET_VECTOR   ORG   0x0000

      goto  Main     ;go to start of main code

;*****************************************************************************
;High priority interrupt vector
; This code will start executing when a high priority interrupt occurs or
; when any interrupt occurs if interrupt priorities are not enabled.

HI_INT_VECTOR  ORG   0x0008
      bra      INT0_ISR          ; service the external interrupt

;*****************************************************************************
; INT0_ISR:  services external interrupt 0, (metal detector)
INT0_ISR:
   movff STATUS,STATUS_TEMP      ; save status reg
   movwf WREG_TEMP               ; save working register
   movff BSR,BSR_TEMP            ; save bsr

   movf     WREG_TEMP,w          ; restore working register
   movff BSR_TEMP,BSR            ; restore bsr
   movff STATUS_TEMP,STATUS      ; restore status
   retfie

; Start of Subroutines and macros

;===================
; Init: initializes internal osc, ports, a/d, pmw, and uart
; input: none
; output: none
```

```
; calls: none
; destroys: w
Init:
      ; init oscillator to 8MHz
      movlw 0x70              ; set osccon to 8MHz
      movwf OSCCON
      ;init analog to digital module
      movlw 0x0E              ;vref=Vdd&Vss, all but AN0 channels off (digital i/o)
      movwf ADCON1
      movlw 0x28              ; left justify result Tacq = 3us, Tconversion = 36us
      movwf ADCON2
      movlw 0x01              ;select ch 0, a/d enabled, idle
      movwf ADCON0
      ;initalize ports
      movlw 0xFD              ;set porta input RA[7:2,0] output RA1
      movwf TRISA
      bsf      RANGE_RX       ;start ranger's calibration cycle (lasts 50ms)
                              ; and continuous conversions
      movlw 0xE7              ;set portb input RB[7:5,2:0], output RB[4:3]
      movwf TRISB
      movlw 0x00              ;initialize outputs to zero
      movwf PORTB
      movlw 0xC0              ;set portc in RC[7:6], output RC[5:0]
      movwf TRISC
      movlw 0x00              ;init all outputs to zero
      movwf PORTC
      clrf  PORTD             ;init all outputs to zero
      movlw 0x0F              ;set portd in RD[3:0], out RD[7:4]
      movwf TRISD
      ;initalize PWM modules
      movlw 0x7F              ;setup ccp2, period 1ms, or f = 1kHz
      movwf PR2
      movlw 0x00              ;init pmw to approx. 0% duty cycle
      movwf CCPR2L
      movlw 0x03              ;set TMR2 prescale to 1:16
      movwf T2CON
      bsf   T2CON,TMR2ON      ; start timer
      movlw 0x3F              ;configure CCP2 to PMW mode
      movwf CCP2CON
      movlw 0x80              ; init to 50% duty cycle
      movwf CCPR1L
      movlw 0x3F              ; configure the modual to PWM mode
      movwf CCP1CON           ; set ccp1 to single output PMW and lsb's B'11'
      ;initalize the euart
      movlw 0x0C              ; set baud rate to 9615 baud (0.16% error)
      movwf SPBRG
      bcf   TXSTA,BRGH
      bcf   BAUDCON,BRG16
      bcf   TXSTA,SYNC        ; set to asynchronous mode
      bsf   RCSTA,SPEN        ; enable serial port
      bsf   TXSTA,TXEN        ; enable transmission
      bsf   RCSTA,CREN        ; enable reception
      return
;=================
; delay_ms: delays operation for number of ms in delay variable
; input: delay
; output: none
; destroys: timer0, w, delay
delay_ms:
   clrf  T0CON                ; stop timer,1:2 prescale,16 bit mode, clk internal osc
   bcf   INTCON,TMR0IF        ; clear interrupt flag for Timer 0
   movlw 0xFC                 ; set timer to 1ms
   movwf TMR0H
   movlw 0x18
   movwf TMR0L
   bsf   T0CON,TMR0ON         ; start timer
   btfss INTCON,TMR0IF        ; stay until overflow
   goto  $-2
   decfsz delay,1             ; exit if done else continue
```

```
   goto  delay_ms
   clrf  T0CON
   bcf   INTCON,TMR0IF
   return
;=================
; delay_s: delays operation for number of seconds in delay variable
; input: delay
; output: none
; destroys: timer0, w, delay
delay_s:
   movlw 0x04                ; stop timer,1:32 prescale,16 bit mode, clk internal osc
   movwf T0CON
   bcf   INTCON,TMR0IF       ; clear interrupt flag for Timer 0
   movlw 0x0B                ; set timer to 1s
   movwf TMR0H
   movlw 0xDC
   movwf TMR0L
   bsf   T0CON,TMR0ON        ; start timer
   btfss INTCON,TMR0IF       ; stay until overflow
   goto  $-2
   decfsz delay,1            ; exit if done else continue
   goto  delay_s
   clrf  T0CON
   bcf   INTCON,TMR0IF
   return

;=================
; xmitStatus: sends the botStatus word
; input: botStatusU, botStatusH, botStatusL
; output: none
; destroys: TXREG,TXIF,w,delay
; calls: delay_ms
xmitStatus:
   movlw d'10'             ; delay 10ms to make sure..
   movwf delay             ; ..maxport can keep up
   call  delay_ms
statusUpper:
   btfss PIR1,TXIF         ; wait for txreg to be empty TXIF = 1..
   goto  statusUpper       ; ..means empty
   movff botStatusU,TXREG  ; send Upper byte
   movlw d'10'             ; delay 10ms to make sure..
   movwf delay             ; ..maxport can keep up
   call  delay_ms
statusHigh
   btfss PIR1,TXIF         ; wait for txreg to be empty TXIF = 1..
   goto  statusHigh        ; ..means empty
   movff botStatusH,TXREG  ; send high byte
   movlw d'10'             ; delay 10ms to make sure..
   movwf delay             ; ..maxport can keep up
   call  delay_ms
statusLow:
   btfss PIR1,TXIF         ; wait for completion as before
   goto  statusLow
   movff botStatusL,TXREG  ; send the low byte
   return

;=================
; wrLCDcom:  Writes a command to the lcd
; input: lcdReg
; output: none
; destroys: w, RD<7:4>, RB<4:3>
; calls: delay_ms
wrLCDcom:
   bcf   LCD_EN            ; clear enable
   bcf   LCD_RS            ; command write
   movlw 0x0F              ; set upper nibble of portd to 0
   andwf PORTD,f           ; ..w/o disturbing lower nibble
   movf  lcdReg,w          ; get lcdReg in w reg
   andlw 0xF0              ; mask the upper nibble
```

```
   iorwf PORTD,f           ; set data on portd
   bsf   LCD_EN            ; toggle the enable pin..
   nop                     ; ..for 1 microsecond
   bcf   LCD_EN
   movlw 0x06              ; wait 6ms for lcd..
   movwf delay             ; ..to complete command
   call  delay_ms
   movlw 0x0F              ; zero upper nibble of PORTD again
   andwf PORTD,f
   swapf lcdReg,w          ; swap nibbles and store in w reg
   andlw 0xF0              ; mask upper nibble (lower nibble of command)
   iorwf PORTD,f           ; set data on portd
   bsf   LCD_EN            ; toggle the enable pin..
   nop                     ; ..for 1 microsecond
   bcf   LCD_EN
   movlw 0x06              ; wait 6ms for lcd..
   movwf delay             ; ..to complete command
   call  delay_ms
   return
;=================
; wrLCDdata: writes data to lcd (characters) found in lcdReg
; input: lcdReg
; output: none
; destroys: w, RD<7:4>, RB<4:3>
; calls: delay_ms
wrLCDdata:
   bcf   LCD_EN            ; clear enable
   bsf   LCD_RS            ; data write rs = 1
   movlw 0x0F              ; set upper nibble of portd to 0
   andwf PORTD,f           ; ..w/o disturbing lower nibble
   movf  lcdReg,w          ; get lcdReg in w reg
   andlw 0xF0              ; mask the upper nibble
   iorwf PORTD,f           ; set data on portd
   bsf   LCD_EN            ; toggle enable pin
   nop
   bcf   LCD_EN            ; clear enable
   movlw 0x01              ; wait for lcd to be ready
   movwf delay
   call  delay_ms
   movlw 0x0F              ; zero upper nibble of PORTD again
   andwf PORTD,f
   swapf lcdReg,w          ; swap nibbles and store in w reg
   andlw 0xF0              ; mask upper nibble (lower nibble of data)
   iorwf PORTD,f           ; set data on portd
   bsf   LCD_EN            ; toggle enable pin
   nop
   bcf   LCD_EN            ; clear enable
   movlw 0x01              ; wait for lcd to be ready
   movwf delay
   call  delay_ms
   return
;=================
; move:         increases the pwm duty cycle until
;           the robot is moving,
;           while monitoring the number of changes
;           in the encoder to move a set distance
; inputs: distance, direction
; output: actual_dist
; destroys: w, CPPRxL (aka SPEED), actual_dist, distance, doubler, enconder, timer0
; calls: delay_ms
move:
   ;init variables
   clrf  botStatusU        ; init to zero, no metal found
   movlw 0x1F              ; init pwm to about 25% duty cycle
   movwf SPEED
   movlw 0x20              ; init doubler to B'00100000'
   movwf doubler
   clrf  actual_dist       ; init actual_dist to zero
   ; get and store portd/encoder value
```

```
   movf  PORTD,w           ; get the value of portd (lower nibble is encoder)
   andlw 0x0F              ; get just the lower nibble
   movwf encoder           ; store in a temporary register
   ;start motors
   movlw STOP              ; clear portc motors
   andwf PORTC,f
   movf  direction,w       ; get direction in w reg to put on portc
   iorwf PORTC,f           ; enable motors
timerStart:
   ;start 1 second timer
   movlw 0x03              ; stop timer,1:16 prescale,16 bit mode, clk internal osc
   movwf T0CON
   bcf   INTCON,TMR0IF     ; clear interrupt flag for Timer 0
   clrf  TMR0H
   clrf  TMR0L             ; set timer for about 0.5 second
   bsf   T0CON,TMR0ON      ; start timer
portdChange?:
   btfsc METAL             ; check for metal
   goto  checkPort
   movlw 0x80              ; set the metal detected bit in
                           ;.. the botStatusH reg
   iorwf botStatusH,f
   movff actual_dist,botStatusU  ; update distance to last metal detected
checkPort:
   movf  PORTD,w           ; check to see if encoder changed value
   andlw 0x0F
   xorwf encoder,w         ; if the same, w is zero and Z bit is set
   btfss STATUS,Z          ; if zero set don't skip to debounce
   goto  debounce          ; yes, it changed
   ; no, it didn't
   ; timer done?
   btfss INTCON,TMR0IF     ; if timer done, don't check for portd change again
   goto  portdChange?      ; no, not done
   ;yes, done
   ; doubler = 80h?
   movf  doubler,w         ; move doubler var to w reg to test msb
   andlw 0x7F              ; if it was b'10000000', w =0 and Z bit set
   btfsc STATUS,Z          ; if not zero, add move power to motor
   goto  endMove           ; yes, it's zero, we stalled the robot
   ; no, not zero
   rlncf doubler,f         ; double duty cycle to pwm channel
   movf  doubler,w         ; get in w reg
   iorwf SPEED,f           ; pwm now doubled
   goto  timerStart
debounce:
   movlw 0x0A              ; wait 10ms for debounce (5ms= max bounce form data sheet)
   movwf delay
   call  delay_ms
   incf  actual_dist,f     ; update the count, we have traveled 1.06"
   movf  PORTD,w           ; get new portd value
   andlw 0x0F              ; mask the upper bits
   movwf encoder           ; store in encoder var
   decfsz   distance,f     ; decrement distance, don't continue to move if zero
   ;distance = 0?
   goto  timerStart        ; no, not zero; continue to move
   ; yes, it's zero
   ; stop robot
   movlw b'11001111'       ; stop drive motor
   andwf PORTC,f
   clrf  direction         ; clear direction register
   clrf  SPEED             ; pwm duty cycle = 0
timerStartAgain:
   movlw 0x03              ; stop timer,1:16 prescale,16 bit mode, clk internal osc
   movwf T0CON
   bcf   INTCON,TMR0IF     ; clear interrupt flag for Timer 0
   clrf  TMR0H
   clrf  TMR0L             ; set timer for about 0.5 second
   bsf   T0CON,TMR0ON      ; start timer
portdChangeAgain?:
```

```
   btfsc METAL             ; check for metal
   goto    checkPortAgain
   movlw 0x80              ; set the metal detected bit in
                           ;.. the botStatusH reg
   iorwf botStatusH,f
   movff actual_dist,botStatusU  ; update distance to last metal detected
checkPortAgain:
   movf  PORTD,w           ; check to see if encoder changed value
   andlw 0x0F
   xorwf encoder,w         ; if the same, w is zero and Z bit is set
   btfss STATUS,Z          ; if zero set don't skip to debounce
   goto  debounceAgain     ; yes, it changed
   ; no, it didn't
   ; timer done?
   btfss INTCON,TMR0IF     ; if timer done, don't check for portd change again
   goto  portdChangeAgain?    ; no, not done
   ;yes, done
   goto  endMove
debounceAgain:
   movlw 0x0A              ; wait 10ms for debounce (5ms= max bounce form data sheet)
   movwf delay
   call  delay_ms
   incf  actual_dist,f     ; update the count, we have traveled 1.06"
   movf  PORTD,w           ; get new portd value
   andlw 0x0F              ; mask the upper bits
   movwf encoder           ; store in encoder var
   goto  timerStartAgain
endMove:
   movlw STOP              ; kill all motors
   andwf PORTC,f
   bcf   STATUS,C          ; clear carry
   rrcf  actual_dist,f     ; compensate for actual dist being doubled
   movlw 0x00
   addwfc actual_dist,f    ; round if necessary
   movlw 0x80              ; comp. for botStatusU x2
   andwf botStatusU,w      ; preserve metal detected bit
   xorwf botStatusU,f      ; clear the metal bit in the botStatusU reg
   bcf   STATUS,C          ; clear the carry
   rrcf  botStatusU,f           ; divide by 2
   iorwf botStatusU,f      ; reset the metal bit to orginal
   movlw 0x00              ; clear w reg to add carry
   addwfc botStatusU,f     ; round if necessary
   return

;=================
; getRange:  Gets the ouput of the ultrasonic ranger, and converts it to inches
; input: none
; output: range
; destroys: w, adLow,adHigh,ADC
getRange:
   bsf   ADCON0,GO         ; start conversion
   btfsc ADCON0,GO         ; when 'GO'= 0 conversion is done
   goto  $-2
   movff ADRESL,adLow      ; copy the a/d result
   movff ADRESH,adHigh
   bcf   STATUS,C          ; multiply by 2 by shifting left, make sure carry is clear
   rlcf  adLow,f           ; rotate low byte
   rlcf  adHigh,f          ; rotate middle
   movff adHigh,range      ; since max range is 254 inches < 2^8...
   return                  ; ..the carry doesn't matter (should be 0).

;==================
; getAvgRange: collects 8 range readings and averages them
; input: none
; output: range
; destroys: w, cnt1, timer0, avgLow, avgHigh, range
; calls: getRange
getAvgRange:
   movlw 0x08              ; init loop var to 8
```

```
   movwf cnt1
   clrf  avgLow            ; init avg to zero
   clrf  avgHigh
get8_sum:
   call  getRange          ; get areading in range var
   movf  range,w           ; get value in w reg
   addwf avgLow,f          ; add to sum
   movlw 0x00              ; zero to add carry to upper reg
   addwfc   avgHigh,f
   movlw D'55'             ; wait 55ms (ranger make conversion every 50ms)
   movwf delay
   call  delay_ms
   decfsz   cnt1,f         ; decrement loop var calc avg if done
   goto  get8_sum
   ; calc average by dividing by 8
   rrcf  avgHigh,f         ; div by 2 ...
   rrcf  avgLow,f
   rrcf  avgHigh,f         ; div by 4 ...
   rrcf  avgLow,f
   rrcf  avgHigh,f         ; div by 8...
   rrcf  avgLow,f          ; average now in avgLow
   movff avgLow,range      ; store in range
   return



;==================
; convBin2BCD:  converts a one byte number from binay to hundreds bcd and
;           tens/ones packed bcd form via double dabble algorithim
; input: num2Convert
; output: hundreds, tens&ones
; destroys: num2Convert,tens&ones,w, STATUS, cnt1
convBin2BCD:
   clrf  hundreds          ; init hundreds to 0
   clrf  tens_ones         ; init tens and ones to 0
   movlw 0x07              ; method must be performed 8 times
   movwf cnt1              ; counter var is init to 8
   bcf   STATUS,C          ; just in case
convLoop:
   rlcf  num2Convert,f     ; rotate original through other two regs..
   rlcf  tens_ones,f       ; ..if any of the target nibbles are > 4..
   rlcf  hundreds,f        ; ..add 3 to them before next shift, do shift 8 times.
   movf  tens_ones,w       ; check for > 4
   andlw 0x0F              ; low nibble first
   movwf tempConv          ; save value
   movlw 0x04              ; compare to 4
   cpfsgt tempConv         ; if nibble <= 4 check next nibble
   goto  tensNibble
   movlw 0x03              ; add three if nibble > 4
   addwf tens_ones,f
   movlw 0x00              ; propogate carry
   addwfc hundreds,f
tensNibble:
   movf  tens_ones,w       ; check upper nibble
   andlw 0xF0              ; get upper nibble alone
   movwf tempConv          ; store
   movlw 0x40              ; to check if >4
   cpfsgt tempConv
   goto  iterDone          ; if nibble <= 4 finish iteration
   movlw 0x30              ; add three if upper nibble > 4
   addwf tens_ones,f
   movlw 0x00              ; propogate the carry
   addwfc hundreds,f       ; done with this nibble
iterDone:
   decfsz cnt1,f           ; check if shfted and check 7 times
   goto  convLoop
   rlcf  num2Convert,f     ; shift one last time to make 8
   rlcf  tens_ones,f       ;
   rlcf  hundreds,f        ;
```

```
   return
;==================
; print3BCD: prints the contents of hundreds, and tens_ones
;           to the lcd at the current cursor location
; input: hundreds, tens_ones
; output: none
; destroys: w, lcdReg
; calls: wrLCDdata
print3BCD:
   movlw '0'               ; get ascii value of number 0 in w
   addwf hundreds,w        ; convert hundreds to ascii store in w
   movwf lcdReg            ; write to lcd
   call  wrLCDdata
   swapf tens_ones,w       ; get tens bcd in low nibble of w
   andlw 0x0F              ; strip it off
   addlw '0'               ; convert to ascii
   movwf lcdReg            ; write it to the lcd
   call  wrLCDdata
   movlw 0x0F              ; get ones in w
   andwf tens_ones,w
   addlw '0'               ; convert to ascii
   movwf lcdReg            ; write to lcd
   call  wrLCDdata
   return

;=================
; initLCD:  Initalizes the LCD
; input: none
; output: none
; destroys: w, RD<7:4>, RB<4:3>, lcdReg
; calls: delay_ms, wrLCDcom
initLCD:
   movlw 0x64              ; wait 100 ms for lcd to power up
   movwf delay
   call  delay_ms
   bcf   LCD_EN            ; make sure enable is clear
   bcf   LCD_RS            ; select command register
   movlw 0x0F              ; set upper nibble of portd to 3
   andwf PORTD,1
   movlw 0x30
   iorwf PORTD,1
   bsf   LCD_EN            ; toggle enable
   nop
   bcf   LCD_EN
   movlw 0x06              ; wait six ms
   movwf delay
   call  delay_ms
   bsf   LCD_EN            ; send 3 again
   nop
   bcf   LCD_EN
   movlw 0x01              ; wait 1 ms
   movwf delay
   call  delay_ms
   bsf   LCD_EN            ; send 3 a third time
   nop
   bcf   LCD_EN
   movlw 0x01              ; wait 1 ms
   movwf delay
   call  delay_ms
   bcf   PORTD,RD4         ; make portd upper nibble 2
   bsf   LCD_EN            ; write to enable 4-bit mode
   nop
   bcf   LCD_EN
   movlw 0x06              ; wait 6 ms
   movwf delay
   call  delay_ms
   movlw 0x28              ; set interface command, 4-bit,2 lines...
   movwf lcdReg
   call  wrLCDcom          ; write the command
```

```
   movlw LCD_OFF           ; turn off lcd
   movwf lcdReg
   call  wrLCDcom
   movlw LCD_CLR           ; clear and home the lcd
   movwf lcdReg
   call  wrLCDcom
   movlw 0x06              ; set cursor move direction right
   movwf lcdReg
   call  wrLCDcom
   movlw LCD_ON            ; turn on lcd
   movwf lcdReg
   call  wrLCDcom
   return
;==================
; updateStatus:  updates all the status registers and lcd
; input: none
; output: botStatusH, botStatusL
; destroys: range, num2Convert, lcdReg, w
; calls:  getAvgRange, convBin2BCD, wrLCDcom,wrLCDdata
updateStatus:
   call  getAvgRange       ; read the ranger
   movlw 0x80              ; clear the range part of the botStatusH byte
   andwf botStatusH,f      ; all cleared, metal detected left alone
   movf  range,w           ; get range in w
   andlw 0x7F              ; just in case there was an error and bit 7 is currupt
   iorwf botStatusH,f      ; updated range is in botStatusH
   movff actual_dist,botStatusL  ; update last distance traveled
   movff range,num2Convert ; update lcd range
   call  convBin2BCD       ; convert number, put in hundreds, tens_ones
   movlw LINE1+7           ; over write previous range
   movwf lcdReg
   call  wrLCDcom          ; cursor now at old data
   call  print3BCD         ; print the numbers
   movff actual_dist,num2Convert ; convert last distance travled to bcd
   call  convBin2BCD       ; converted to packed bcd in hundreds, tens_ones
   movlw LINE2+d'11'       ; overwrite old data
   movwf lcdReg
   call  wrLCDcom          ; cursor now at old distance
   call  print3BCD         ; data overwritten
   return



;====================
;print:  prints a tabulated string to the lcd
;        at the current cursor location
print macro tableAdd
   movlw low tableAdd
   movwf TBLPTRL
   movlw high tableAdd
   movwf TBLPTRH
   movlw upper tableAdd
   movwf TBLPTRU
   tblrd*+                 ; get first value of table in tablat, inc pointer
   movf  TABLAT,0            ; move value ot w reg
   btfsc STATUS,Z          ; if clear, not 0 and not done
   goto  $+10
   movwf lcdReg            ; write the data
   call  wrLCDdata
   goto  $-10
   nop                     ; to align program memory so $+10 works
   endm


;*****************************************************************************
;Start of main program
; The main program code is placed here.

Main:
```

```
;======= initialize the robot ==============
   call  Init              ; initialize the microcontroller's resources
   call  initLCD
;=======  print an initialization message =============
   print init0             ; "   ECET 4904"
   movlw LINE2             ; move cursor to line 2
   movwf lcdReg
   call  wrLCDcom
   print    init1          ; "   Fall 2008"
   movlw 0x02              ; delay 2 seconds for viewing
   movwf delay
   call  delay_s
   movlw LCD_CLR           ; clear and home lcd
   movwf lcdReg
   call  wrLCDcom
   print init2             ; "Advisor:"
   movlw LINE2             ; move cursor to line 2
   movwf lcdReg
   call  wrLCDcom
   print    init3          ; "Prof. Wilcox"
   movlw 0x02              ; delay 2 seconds for viewing
   movwf delay
   call  delay_s
   movlw LCD_CLR           ; clear and home lcd
   movwf lcdReg
   call  wrLCDcom
   print init4             ; "Student:"
   movlw LINE2             ; move cursor to line 2
   movwf lcdReg
   call  wrLCDcom
   print init5             ; "J. Galloway"
   movlw 0x02              ; delay 2 seconds for viewing
   movwf delay
   call  delay_s
   movlw LCD_CLR           ; clear and home lcd
   movwf lcdReg
   call  wrLCDcom
   print lcdMessL1         ; Range: 000"
   movlw LINE2             ; move cursor to line2
   movwf lcdReg
   call  wrLCDcom
   print lcdMessL2         ; Last Dist: 000"

; init status regs to zero
   clrf  actual_dist
   clrf  range
   clrf  direction
   clrf  distance
   clrf  botStatusU        ; clear distance to last metal detection
   clrf  botStatusH        ; clr metal bit and range
   clrf  botStatusL        ; zero last distance traveled
   clrf  commandH          ; zero command high byte
   clrf  commandL          ; zero command low byte
; update all status registers
initialStatus:
   call  updateStatus      ; update all the status registers and lcd
waitForCommand:
   btfss PIR1,RCIF         ; wait for command high byte
   goto  $-2               ; keep waiting
   movff RCREG,commandH    ; store command's high byte
   clrf  T0CON             ; use timer0 as a wdt for byte..
   clrf  TMR0L             ; ..receipt, wait approx 65ms
   clrf  TMR0H
   bcf   PIR1,TMR0IF       ; make sure flag is clear
   bsf   T0CON,TMR0ON      ; start timer
getSecondByte:
   btfss PIR1,RCIF         ; if pir1=1 second byte is here
   goto  checkTMR0         ; else, check to see if timer1 is done
   movff RCREG, commandL   ; store low byte
```

```
   goto  whichCommand      ; data received
checkTMR0:
   btfss PIR1,TMR0IF       ; if overflow, give up on receive
   goto  getSecondByte     ; if not, keep trying
   bcf   T0CON,TMR0ON      ; stop timer
   goto  waitForCommand    ; start over at waiting
whichCommand:
   bcf   T0CON,TMR0ON      ; kill timer0
   rlcf  commandH,w        ; rotate the high command and store in w...
                           ; ..commandH left unaltered
   btfss STATUS,C          ; if carry is set, bit 7 was high and...
                           ; ..the command was a status request
   goto  commandMove       ; if clr, it was a move command
statusRequest:
   call  updateStatus      ; update the status registers
   call  xmitStatus        ; send the status packet
   goto  waitForCommand    ; all done
commandMove:
   ;send ack
   btfss PIR1,TXIF         ; wait for clear tx reg.
   goto  $-2
   movlw MOVE_ACK          ; acknowledge the move command
   movwf TXREG             ; send data
   ; set direction and distance
   movff commandL,direction   ; get direction in direction reg
   movff commandH,distance    ; distance now in distance reg
   ;clear status registers
   clrf  botStatusU
   clrf  botStatusH
   clrf  botStatusL        ; all cleared
   ; get ready to move
   call  move              ; let the bot move
   ; update status and send results
   call  updateStatus      ; update the status registers
   call  xmitStatus        ; send the new status
   goto  waitForCommand


;*******************************************************************************
;Tables
   org 0x7000
init0: db "   ECET 4904",0
init1: db "   Fall 2008",0
init2: db "Advisor:",0
init3: db "Prof. Wilcox",0
init4: db "Student:",0
init5: db "J. Galloway",0
lcdMessL1: db "Range: 000",0x22,0
lcdMessL2: db "Last Dist: 000",0x22,0

;End of program

      END
```

APPENDIX E: JAVA CODE

```
/**
 * RunLandBot: Creates a JFrame to display
 *          the landBotPanel; in essence,
 *             it runs the program.
 *
 * Author: Joshua Galloway
 * Date: 11/13/08
 * Modifications: Original Code
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class RunLandBot
{
   /** runs the landbot gui */
   public static void main(String[] args)
   {
      LandBotPanel l = new LandBotPanel();
      JFrame f = new JFrame();
      f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
      f.setTitle("Land Mine Bot Control Panel");
      f.add(l);
      f.pack();
      f.setLocation(0,0);
      f.setVisible(true);
   }
}
/**
 * LandBotPanel: Creates a JPanel and action listener/handler by which
 *               the robot is controlled.
 *
 * Author: Joshua Galloway
 * Date: 11/13/08
 * Modifications: Original Code
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.io.*;
import java.lang.String;
import java.util.Calendar;
import java.text.SimpleDateFormat;


public class LandBotPanel extends JPanel implements ActionListener
{
   public static JFrame autoBot;
   public static BotMap botMap;
   private static final int port = 9876;
   public static JButton status,fLeft,fwd,fRight,bLeft,bkw,bRight,camera;
   public static JTextField ipBotText,ipCamText;
   public static JTextArea connStat;
   public static JScrollPane scp;
   public static JRadioButton auto,manual;

   public static JTextField dist;
   //public static JComboBox dist;
   public static JLabel ipBotLabel,ipCamLabel,distLabel,controlLabel;

   //constructor
   public LandBotPanel()
   {
      // initialize all the gui elements
      this.initElements();
```

```
   // add everything to the panel
   this.populate();
}

// method implemented from ActionListener
public void actionPerformed(ActionEvent ae)
{
   Object source = ae.getSource();
   if(ae.getActionCommand()=="auto")
   {
      updateStatusWindow("Entering Autonomous Mode...");
      Sweep as = new Sweep(ipBotText.getText(),port,this);
      as.setPriority(Thread.NORM_PRIORITY);
      as.start();
   }
   else if(ae.getActionCommand()=="manual")
   {
      updateStatusWindow("Entering Manual Mode...");
      autoBot.setVisible(false);
   }
   else if (camera == source)
   {
         String str = "http://"+ ipCamText.getText() +"/";
         BrowserLaunch.openURL(str);
         str = "Opened " + str;
         updateStatusWindow(str);
   }
   else if ((status == source)&(manual.isSelected()))
   {
      try
      {
         BotStatusPing bsp =
               new BotStatusPing(ipBotText.getText(),port);
         updateStatusWindow(bsp.getFormattedStatus());
      }
      catch(Exception e)
      {
         updateStatusWindow("Status ping failed.." +
                     "\n" + e.getMessage());
      }
   }
   else
   {
      if(auto.isSelected())
         return;
      // otherwise it must be a directional button
      this.directButtonPressed(source);
   }
}

   // handle a dictional button being pressed
   public void directButtonPressed(Object o)
   {
      int direction;          // var to set direction
      if (fLeft == o)
      {
         direction = BotMoveCommand.FORWARD_LEFT;
      }
      else if (fwd == o)
      {
         direction = BotMoveCommand.FORWARD;
      }
      else if (fRight == o)
      {
         direction = BotMoveCommand.FORWARD_RIGHT;
      }
      else if (bLeft == o)
      {
         direction = BotMoveCommand.BACK_LEFT;
```

```
      }
      else if (bkw == o)
      {
         direction = BotMoveCommand.BACK;
      }
      else  // bRight == o must be true
      {
         direction = BotMoveCommand.BACK_RIGHT;
      }

      // send command
      try
      {
         BotMoveCommand bmc = new BotMoveCommand(ipBotText.getText(),
               port,direction,getIncrement());
         updateStatusWindow(bmc.getFormattedStatus());
      }
      catch(Exception e)
      {
         updateStatusWindow("Move command failed" +
                     "\n"+ e.getMessage());
      }

      return;
}


//  add all the elements to the panel
public void populate()
{

   // setup panel layout
   this.setLayout(new GridBagLayout());

   // add all the elements

   // label for text box with robot ip address
   this.add(ipBotLabel, MyGUIMethods.getConstraints(
      0,0,3,1,GridBagConstraints.CENTER));

   // text box with robot ip address
   this.add(ipBotText, MyGUIMethods.getConstraints(
      0,1,3,1,GridBagConstraints.CENTER));

   // label for the manual controls
   this.add(controlLabel, MyGUIMethods.getConstraints(
      0,2,3,1,GridBagConstraints.CENTER));

   // Forward Left manual button
   this.add(fLeft, MyGUIMethods.getConstraints(
      0,3,1,1,GridBagConstraints.CENTER));

   // Forward button
   this.add(fwd, MyGUIMethods.getConstraints(
      1,3,1,1,GridBagConstraints.CENTER));

   // Forward Right button
   this.add(fRight, MyGUIMethods.getConstraints(
      2,3,1,1,GridBagConstraints.CENTER));

   // Back Left button
   this.add(bLeft, MyGUIMethods.getConstraints(
      0,4,1,1,GridBagConstraints.CENTER));

   // Backward button
   this.add(bkw, MyGUIMethods.getConstraints(
      1,4,1,1,GridBagConstraints.CENTER));

   // Backward right button
```

```
   this.add(bRight, MyGUIMethods.getConstraints(
      2,4,1,1,GridBagConstraints.CENTER));

   // Label the distance selector combo-box
   this.add(distLabel, MyGUIMethods.getConstraints(
      0,5,3,1,GridBagConstraints.CENTER));

   // Distant selector combo-box
   this.add(dist, MyGUIMethods.getConstraints(
      0,6,3,1,GridBagConstraints.NORTH));

   // Label for ip camera text box
   this.add(ipCamLabel, MyGUIMethods.getConstraints(
      3,0,3,1,GridBagConstraints.CENTER));

   // ip camera address text box
   this.add(ipCamText, MyGUIMethods.getConstraints(
      3,1,3,1,GridBagConstraints.CENTER));

   // button to open the camera in a browser  3231
   this.add(camera, MyGUIMethods.getConstraints(
      3,2,3,1,GridBagConstraints.CENTER));

   // status window scrolling display   3334
   this.add(scp, MyGUIMethods.getConstraints(
      3,3,3,5,GridBagConstraints.CENTER));

   // manual radio button  0731
   this.add(manual, MyGUIMethods.getConstraints(
      0,8,3,1,GridBagConstraints.CENTER));

   // autonomous radio button   3731
   this.add(auto, MyGUIMethods.getConstraints(
      3,8,3,1,GridBagConstraints.CENTER));

   // status ping button  2821
   this.add(status, MyGUIMethods.getConstraints(
      0,7,3,1,GridBagConstraints.CENTER));


   return;

}

/**initialize and setup all the gui elements*/
public void initElements()
{
   //set control buttons
   fLeft = new JButton(new ImageIcon("arrows/fLeftArrow.gif"));
   fLeft.addActionListener(this);
   fwd = new JButton(new ImageIcon("arrows/fwdArrow.gif"));
   fwd.addActionListener(this);
   fRight = new JButton(new ImageIcon("arrows/fRightArrow.gif"));
   fRight.addActionListener(this);
   bLeft = new JButton(new ImageIcon("arrows/bLeftArrow.gif"));
   bLeft.addActionListener(this);
   bkw = new JButton(new ImageIcon("arrows/bkwArrow.gif"));
   bkw.addActionListener(this);
   bRight = new JButton(new ImageIcon("arrows/bRightArrow.gif"));
   bRight.addActionListener(this);

   // camera button
   camera = new JButton("View Camera");
   camera.addActionListener(this);

   // status button
   status = new JButton("Status");
   status.addActionListener(this);
```

```
      // setup combo box
      //String[] s= {"6 inches","1 foot"};
      //dist = new JComboBox(s);
      dist = new JTextField("12",5);

      //Text Fields
      ipBotText = new JTextField("192.168.10.15",15);
      ipCamText = new JTextField("192.168.10.30",15);
      connStat = new JTextArea(" ",15,25);
      connStat.setLineWrap(true);
      connStat.setEditable(false);
      scp = new JScrollPane(connStat,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                        JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);

      //create labels
      ipBotLabel = new JLabel("IP Address of Robot");
      ipCamLabel = new JLabel("IP Address of Camera");
      distLabel = new JLabel("Distance");
      controlLabel = new JLabel("Controls");

      // setup radio buttons
      auto = new JRadioButton("Autonomous Mode");
      auto.setActionCommand("auto");
      auto.addActionListener(this);
      manual = new JRadioButton("Manual Mode");
      manual.setActionCommand("manual");
      manual.addActionListener(this);
      manual.setSelected(true);
      ButtonGroup bg = new ButtonGroup();
      bg.add(auto);
      bg.add(manual);
      // all done
      return;
   }


   /** Figures out the increment the robot should move in manual mode */
   public static int getIncrement()
   {

      try
      {
         return Integer.parseInt(dist.getText());
      }
      catch(Exception e){ return 0;}

   }

   /**writes a message to the status window with a time stamp*/
   public static void updateStatusWindow(String s)
   {
      Calendar cal = Calendar.getInstance();
       SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
      connStat.append(s + "\n" + sdf.format(cal.getTime()) + "\n");
      connStat.setCaretPosition(connStat.getText().length());
      return;
   }

}

/**
 * RunLandBot: Creates a JFrame to display
 *          the landBotPanel; in essence,
 *             it runs the program.
 *
 * Author: Joshua Galloway
 * Date: 11/13/08
 * Modifications: Original Code
 */
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class RunLandBot
{
   /** runs the landbot gui */
   public static void main(String[] args)
   {
      LandBotPanel l = new LandBotPanel();
      JFrame f = new JFrame();
      f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
      f.setTitle("Land Mine Bot Control Panel");
      f.add(l);
      f.pack();
      f.setLocation(0,0);
      f.setVisible(true);
   }
}

/**
 * Sweep: Performs an autonoumus sweep of a 10x10 foot area
 * Author: Joshua Galloway
 * Date: 11/30/08
 * Modifications: Original Code
 */
import java.net.*;
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.text.NumberFormat;
import java.lang.String;
import java.util.Calendar;
import java.text.SimpleDateFormat;
public class Sweep extends Thread
{
   private static final double Y_DISTANCE = 120.0;   //total distance of field
                                    // to be swept in inches
   private static final double X_DISTANCE = 120.0;
   private static final int INCREMENT = 12;         //incremtent to move in inches
   public String ip;
   public String status;
   public int port;
   public BotMap bm;

   /**  Creates a new autonomous sweep on a separate thread
    *   ad is IP address of bot and p is port number.
    *   Also, creates progress map and text record "autonomous_sweep.txt".
    */
   public Sweep(String ad, int p,JPanel jp)
   {
      JFrame autoBot = new JFrame("Autonomous Robot Status");
      autoBot.setDefaultCloseOperation(autoBot.EXIT_ON_CLOSE);
      bm = new BotMap();
      autoBot.add(bm);
      autoBot.pack();
      // returns Toolkit object
      Toolkit tk = Toolkit.getDefaultToolkit();
      // returns dimentions of the user's screen
      Dimension d = tk.getScreenSize();
      autoBot.setLocation(jp.getWidth()+20,0);
      autoBot.setVisible(true);
      ip = ad;
      port = p;
   }

   public void run()
   {
```

```
 double x,y;    // x and y coordinates of robot
 int x_square,y_square;   // x and y  for the botmap matrix


 // create calendar for time stamps
 Calendar cal = Calendar.getInstance();
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

 NumberFormat nf = NumberFormat.getNumberInstance();
nf.setMaximumFractionDigits(2);

try
{
   File record  = new File("autonomous_sweep.txt");
   PrintWriter pw = new PrintWriter(
                new BufferedWriter(
                new FileWriter(record)));
   pw.println(sdf.format(cal.getTime()) + " New Autonomous Sweep Begun.");
   pw.println("Time of Detection \t\t" +
            "X, Y coordinates (inches)");

   // start running the sweep
   boolean forward = true;
   x_square = 0;
   y_square = 0;
   x = 0;
   y = 0;
   for( ; y < Y_DISTANCE; )
   {

      for(;;)
      {
         try
         {
            BotMoveCommand bmc;
            if(forward)
            {
               // should be moving forward
               int i = (int)(INCREMENT - x % INCREMENT);
               bmc = new BotMoveCommand(ip,port,
                        BotMoveCommand.FORWARD,i);
               if(bmc.metalDetected())
               {
                  // metal detected
                  pw.println(cal.getTime() + "\t" +
                           nf.format(x+bmc.getDist_metal()) +
                           "\t" + nf.format(y));
                  x_square = (int)((x+bmc.getDist_metal())/INCREMENT);
                  if(bm.getGridColor(y_square,x_square).equals(BotMap.INIT_COLOR))
                     bm.setGridColor(y_square,x_square,BotMap.FOUND_COLOR);
                  x += bmc.getDist_traveled();
               }
               else
               {
                  // no metal detected
                  x_square = (int)((x+bmc.getDist_metal())/INCREMENT);
                  bm.setGridColor(y_square,x_square,BotMap.VISITED_COLOR);
                  x += bmc.getDist_traveled();
               }
            }
            else
            {
               //should be moving backward
               int i = (x-INCREMENT > 0)?
                        (int)(INCREMENT - x%INCREMENT):
                        (int)(x%INCREMENT+0.5);
               if((i==0)&(x_square==9))
                  i = INCREMENT;
```

```
                    bmc = new BotMoveCommand(ip,port,BotMoveCommand.BACK,i);
                    if(bmc.metalDetected())
                    {
                       // metal detected
                       pw.println(cal.getTime() + "\t" +
                                nf.format(x-bmc.getDist_metal()) +
                                "\t" + nf.format(y));
                       x_square = (int)((x-bmc.getDist_metal())/INCREMENT);
                       if(bm.getGridColor(y_square,x_square).equals(BotMap.INIT_COLOR))
                          bm.setGridColor(y_square,x_square,BotMap.FOUND_COLOR);
                       x -= bmc.getDist_traveled();
                    }
                    else
                    {
                       // no metal detected
                       x_square = (int)((x-bmc.getDist_metal())/INCREMENT);
                       if(bm.getGridColor(y_square,x_square).equals(BotMap.INIT_COLOR))
                          bm.setGridColor(y_square,x_square,BotMap.VISITED_COLOR);
                       x -= bmc.getDist_traveled();
                    }
                 }
                 bm.repaint();
                 System.out.println("x = " + nf.format(x));
              }catch(Exception e){System.out.println("Exception in Sweep Row Loop");}
              if(forward & (x < X_DISTANCE))
                 continue;
              if(!forward & (x > 0.0))
                 continue;
              break;
           }

           if(forward)
              nextRowFWD();
           else
              nextRowBACK();
           y_square++;
           y += INCREMENT;
           forward = !forward;
        }
        pw.println(cal.getTime() +"  Sweep Completed");
        pw.close();
        for(int i = 0; i<10; i++)
        {
           for(int k = 0; k<10; k++)
           {
              if(bm.getGridColor(i,k).equals(BotMap.INIT_COLOR))
                 bm.setGridColor(i,k,BotMap.VISITED_COLOR);
           }
        }
        bm.repaint();

     }catch(Exception e){System.out.print(e.getMessage());}

     return;
  }


  /** positions the robot at the end of the next row*/
  public static void nextRowFWD()
  {
     //stub
     return;
  }
  /** positions the robot at the beginning of the next row */
  public static void nextRowBACK()
  {
     //stub
     return;
  }
```

```
}
/**
 *   BotMap:  This class draws a 2D map of a 10' x 10' grid
 *            that the robot will sweep.  It allows for updating
 *            through the setGridColor method.
 *
 * Author: Joshua Galloway
 * Date: 11/16/08
 * Modifications: Original Code
 */
import java.awt.*;

public class BotMap extends Canvas
{
   // elements and variables
   private static final String TITLE = "Total Grid is 10 Feet by 10 Feet." +
                              "  Robot Starts by 1,1 Pointing Toward" +
                              " 1,2.";
   private static final String LEGEND =
         "Gray = Not Checked, Green = Checked, Red = Land Mine Found";
   private static final int xSize = 500;
   private static final int ySize = 500;
   private static final int xBorder = 50;
   private static final int yBorder = 50;
   private static final int titleOffset = 10;
   public static final int numCols = 10;
   public static final int numRows = 10;
   public static final Color INIT_COLOR = Color.lightGray;
    public static final Color VISITED_COLOR = Color.green;
    public static final Color FOUND_COLOR = Color.red;
    public Color[][] status = new Color[numRows][numCols];

   /**
    * Constructs the object and initializes all the squares to INIT_COLOR
    */
   public BotMap()
   {
      //init all squares to gray
      for(int row=0;row<numRows;row++)
      {
         for(int col=0;col<numCols;col++)
         {
            status[row][col] = INIT_COLOR;
         }
      }
      setSize(xSize,ySize);
      setBackground(Color.gray);
   }
   /** Paints the map with coordinates*/
   public void paint(Graphics g)
   {
      int xStart = xBorder-1;
      int yStart = yBorder-1;
      int xInc = (int)((xSize-2*xBorder)/numRows);
      int yInc = (int)((ySize-2*yBorder)/numCols);
      for(int row=0;row<numRows;row++)
      {
         for(int col=0;col<numCols;col++)
         {
            g.setColor(status[row][col]);
            g.fillRect(xStart,yStart,xInc,yInc);
            xStart += xInc;
         }
         xStart = xBorder-1;
         yStart += yInc;
      }
      g.setColor(Color.black);

      // draw a grid over top
```

```
      xStart = xBorder-1;
      yStart = yBorder-1;
      for(int row=0;row<numRows+1;row++)
      {

         g.drawLine(xStart,yStart,xStart + xSize-2*xBorder,yStart);
         yStart += yInc;
      }
      // draw a grid over top
      xStart = xBorder-1;
      yStart = yBorder-1;
      for(int col=0;col<numCols+1;col++)
      {

         g.drawLine(xStart,yStart,xStart,yStart + ySize-2*yBorder);
         xStart += xInc;
      }

      // label the thing
      g.setColor(Color.white);
      FontMetrics fm = g.getFontMetrics();
      g.drawString(TITLE,(int)(xSize-fm.stringWidth(TITLE))/2,
                  titleOffset+fm.getMaxAscent());
      g.drawString(LEGEND,(int)(xSize-fm.stringWidth(LEGEND))/2,
                  ySize-titleOffset-fm.getMaxDescent());

      //print coordinates
      xStart = xBorder-1;
      yStart = yBorder-1;
      for(int row=0;row<numRows;row++)
      {
         String x = Integer.toString(row+1);
         g.drawString(x,xStart-fm.stringWidth(x)-5,
                   (int)(yStart+(yInc+fm.getHeight())/2));
         yStart += yInc;
      }
      xStart = xBorder-1;
      yStart = yBorder-1;
      for(int col=0;col<numCols;col++)
      {
         String y = Integer.toString(col+1);
         g.drawString(y,(int)(xStart+(xInc-fm.stringWidth(y))/2),
                  yStart-5);
         xStart += xInc;
      }

   }

   /**
    * Allow the color of the grid squares to be changed
    * int row in matrix, int column in matrix, Color new
    * color of square
    */
   public void setGridColor(int row, int col, Color c)
   {
      status[row][col]=c;
      return;
   }
   /**
    * Allow the color of the grid squares to be accessed
    * int row in matrix, int column in matrix
    *
    */
   public Color getGridColor(int row, int col)
   {
      return status[row][col];
   }
}
/*
```

```
 * MyGUIMethods: Contains reusable methods
 *               for creating gui apps in
 *               java.
 *
 * Author: Joshua Galloway
 * Date: 7/22/2006
 * Modifications: Original Code
 */
import java.awt.*;
import javax.swing.*;
public class MyGUIMethods
{
   /** method to get constraints object for grid bag layout */
   public static GridBagConstraints getConstraints(int gridx, int gridy,
         int gridWidth, int gridHeight, int align)
   {
      GridBagConstraints c = new GridBagConstraints();
      c.gridx = gridx;
      c.gridy = gridy;
      c.gridwidth = gridWidth;
      c.gridheight = gridHeight;
      c.anchor = align;
      c.insets = new Insets(5,5,5,5);
      return c;
   }

   /** Centers a window in any user's screen */
   public static void centerWindow(Window w)
   {
      // returns Toolkit object
      Toolkit tk = Toolkit.getDefaultToolkit();
      // returns dimentions of the user's screen
      Dimension d = tk.getScreenSize();
      w.setLocation((d.width-w.getWidth())/2,
         (d.height-w.getHeight())/2);
   }


}
/**
  * Timer: Creates a timer of a specified length in ms.  It countsdown to
  *        the specified value and then dies.
  *
  * Author: Joshua Galloway
  * Date: 11/16/08
  * Modifications: Original Code
  */
import java.net.*;
public class Timer extends Thread
{
   private int msRate = 100;      // timer resolution  in ms
   private int msLength;         // timer length in ms
   private int msElapsed;        // timer's current value in ms

   /**
     * Creates a timer of a specified length
     */
   public Timer ( int length )
   {
      msLength = length;
      msElapsed = 0;
   }


   /** Resets the timer back to zero */
   public synchronized void reset()
   {
      msElapsed = 0;
   }
```

```
   /** Starts the timer */
   public void run()
   {
      // start timer
      while(true)
      {
         // Put the timer to sleep for resolution duration
         try
         {
            Thread.sleep(msRate);
         }
         catch (InterruptedException ignore)
         {}

         // Use 'synchronized' to prevent conflicts
         synchronized ( this )
         {
            // Increment time remaining
            msElapsed += msRate;

            // Check to see if the time has been exceeded
            if (msElapsed > msLength)
               break;
         }

      }
   }

}
/**
 * LandSimServer: Uses java's random number generator to
 *                simulate the landmine sweeping robot
 * Author: Joshua Galloway
 * Date: 11/27/08
 * Modifications: Original Code
 */
 import java.net.Socket;
import java.net.ServerSocket;
import java.io.IOException;
import java.io.*;
import java.nio.*;
import java.lang.*;
import java.util.*;
import java.text.SimpleDateFormat;

public class LandBotSimServer {
   private static final int port = 9876;
   private static final int max_conn = 1;
   private static final byte moveAck = (byte)0x0F;

   public static void main (String[] args) {
      try {
            ServerSocket ssock = new ServerSocket (port, max_conn);
            int   lastDistance=0;
            int distToMetal=0;
            int distance = 0;
            int range=0;
            boolean metal = false;
            byte[] command = new byte[2];
            byte[] output = new byte[3];

            while (true) {
               Socket sock = ssock.accept(); // accept() blocks
               DataInputStream dis = new DataInputStream(sock.getInputStream());
               DataOutputStream dos = new DataOutputStream(sock.getOutputStream());

               // read command
```

```
                dis.readFully(command);
                System.out.println("Received");
                for(int i = 0; i < command.length; i++)
                {
                   System.out.println(Integer.toString(command[i] & 0xff,
                                     16).toUpperCase());
                }
                System.out.println(now());
                if((int)command[0] < 0)
                {
                   //status ping
                   Random r = new Random();
                   range = r.nextInt(122)+6;
                }
                else
                {
                   dos.write(moveAck);
                   Random r = new Random();
                   try{Thread.sleep(r.nextInt(5000));}
                   catch(InterruptedException ie){}
                   distance = (int)command[0];
                   lastDistance = distance + r.nextInt(3);
                   range = r.nextInt(122)+6;
                   double d = r.nextDouble();
                   metal = ((d) < 0.1);
                   if(metal)
                      distToMetal = lastDistance - r.nextInt(lastDistance+1);

                }
                output[0] = (byte)distToMetal;
                output[1] = metal?(byte)(range*-1):(byte)range;
                output[2] = (byte)lastDistance;
                System.out.println("Sending");
                for(int i = 0; i < output.length; i++)
                   {
                      System.out.println(Integer.toString(output[i] & 0xff,
                                        16).toUpperCase());
                   }
                System.out.println(now());
                dos.write(output,0,output.length);
                sock.close();

             }
          }
       catch (IOException e) {
          System.err.println(e);
       }
    }

       public static String now() {
          Calendar cal = Calendar.getInstance();
          SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
          return sdf.format(cal.getTime());

        }
}
```