# Computer Engineering Senior Design Projects II
# EECC 657
# Final Proposal

## EX-Bot
## Map it once and use it forever.

Submitted By: Timothy C. LaForest

_____

Eric Shields

_____

Merve Evran

_____

Submitted To: Dr. Roy Czernikowski
Date Updated: January 17, 2006

## Table of Contents



## Index of Figures



## Index of Tables

## *Introduction*

The objective of this project is to create a robot that will be capable, with the assistance of a PC, of mapping an area, approximately 15 feet by 15 feet, then navigating the mapped area. The system will have a java-based user interface with a display of the mapped area and some controls and will communicate via Bluetooth v1.1 compliant modules. The robot will make use of ultrasonic sensors, a magnetic compass, and infrared phototransistors for mapping and navigation. The robot will use the $I^2C$ standard for communication with the sensors and Bluetooth module (except for the phototransistors).

## *Overall System*

The complete system consists of three components: The robot, a laptop serving as the user interface, and the communications between them. A component diagram of the system is shown below in figure 1.

The overall system design is as follows: Both components have a Bluetooth wireless module. Using Bluetooth v1.1 2.4GHz protocol, a series of signals will be transmitted, in the form of specifically assigned segments of the payload of each packet, to and from the robot.

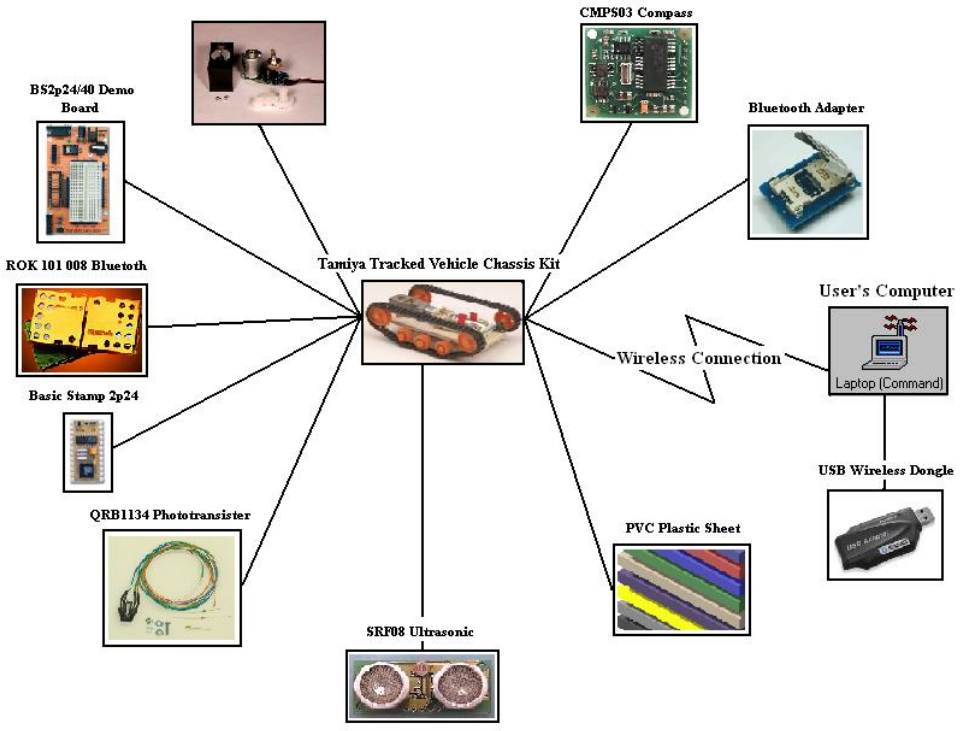


**Figure 1: System Overview Component Diagram**

Some of the operating environment will be created specifically for the demo. These will include 4 outer walls, to eliminate the issues related to outer boundaries, and objects of various shapes and sizes to act as obstacles.

The next three sections detail each of the major pieces of the system, which are stated above.

## *Robots*

The robot will use the following as a base: The Tamiya Tracked Vehicle Chassis with the Low Cost H-Bridge provides the motor and drive. There will be two tiers, constructed of colored PVC plastic sheets, one for the motor housing and one for the components. On the second tier will sit the BS2p24/40 Demo Board from Parallax, Inc. This board was chosen for the fact that it has an embedded $I^2C$ controller and Parallax, Inc provides PBasic, an easy to use programming language.

Connected to the development section of the board will be the CPMS03 Magnetic Compass, Bluetooth Prototyping Adapter with Ericsson Bluetooth MCM Point-to-Point module, and QRB1134 Photo-transister. The compass and Bluetooth module will be connected to the $I^2C$ bus. The compass will determine the direction that the robot is facing. The photo-transisters, one on each tread, will allow the user's computer to determine the distance travelled by sending a signal to the BS2 everytime a tread hole passes. The Bluetooth module will allow communication with the user's computer.

In addition to what the Controlled robots have, the Mapper robot will have 2 SFR08 High Performance Ultrasonic Range Finders on an additional tier. Each one will sit atop a "turret," a servo motor, which will rotate so as to take measurements at different angles. These "turrets" will be placed such that they fall in the center of the two boxes formed by splitting the robot into two equal segments, lengthwise. The two sensors will always be facing in opposite directions when taking measurements. These range finders are controlled through the $I^2C$ bus on the control board.

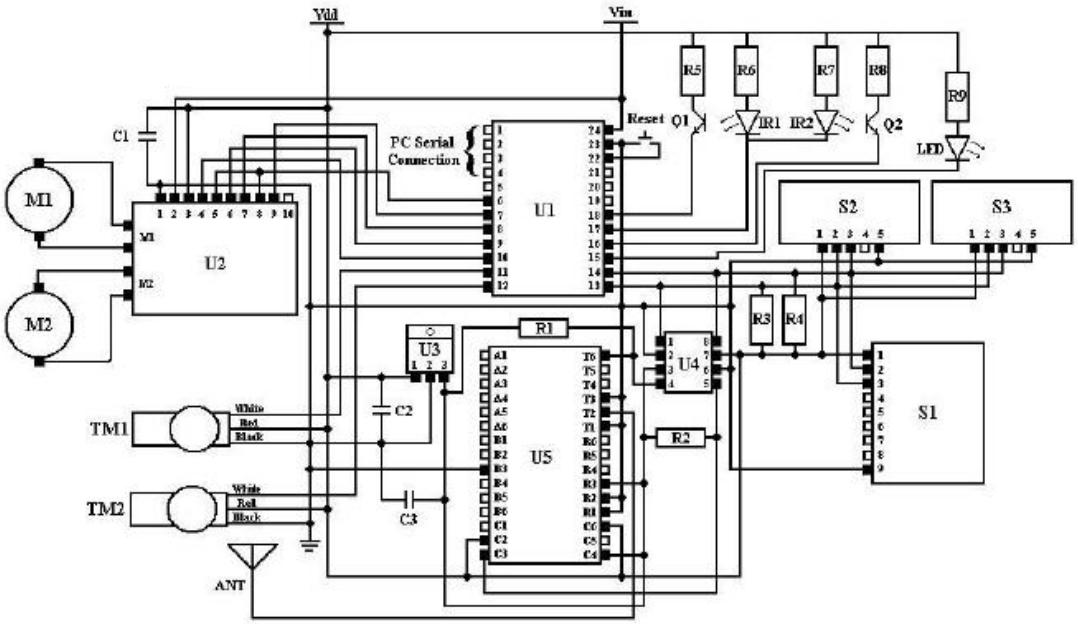


**Figure 2: Robot Schematic**

Additional components are needed to perform power conversions for the various components. The details of robot system are contained in the schematic for the robot, Figure 2 above, and the component list, Figure 3 below.

**ANT:** 50Ω Bluetooth Antenna

**C1:** 5V 1.0F Polorized Capacitor
**C2:** 0.47 µF Capacitor
**C3:** 10V 33µF Polorized Capacitor

**IR1:** QED223 Plastic Infrared LED 880nm (on Left Tread)
**IR2:** QED223 Plastic Infrared LED 880nm (on Right Tread)

**LED:** Test Indicator LED (Red)

**M1:** FA-130 DC Motor (Connected to Left Tread)
**M2:** FA-130 DC Motor (Connected to Right Tread)

**Q1:** QSD124 Plastic Silicon Infrared Phototransistor 880nm (on Left Tread)
**Q2:** QSD124 Plastic Silicon Infrared Phototransistor 880nm (on Right Tread)

**R1:** 1.2KΩ Resistor
**R2:** 1.2KΩ Resistor
**R3:** 4.7KΩ Resistor
**R4:** 4.7KΩ Resistor
**R5:** 1KΩ Resistor
**R6:** 2KΩ Resistor
**R7:** 2KΩ Resistor
**R8:** 1KΩ Resistor
**R9:** 2KΩ Resistor

**S1:** Devantech CMPS03 Magnetic Compass Module (Centered on Robot)
**S2:** Devantech SRF08 Ultrasonic Range Finder (Mounted Facing Front)
**S3:** Devantech SRF08 Ultrasonic Range Finder (Mounted Facing Rear)

**TM1:** Turret Motor – GWServo S03N STD (Mounted in Front)
**TM2:** Turret Motor – GWServo S03N STD (Mounted in Rear)

**U1:** Basic Stamp Microprocessor (BS2p24)
**U2:** Low Cost Dual 1.1A H-Bridge Module
**U3:** LM3940 - 1A Low Dropout Regulator for 5V to 3.3V Conversion
**U4:** MAX3372E/MAX3373E - ±15kV ESD-Protected, 1µA, 16Mbps, Dual Low-Voltage Level Translator
**U5:** ROK 101 008 Bluetooth Module

**Vdd:** Regulated +5.0V
**Vin:** +8.4V Battery

**Figure 3: Robot Component List**

## *Wireless Communications*

The "glue" that holds this project together is the wireless communications that occur constantly between the robot and the user interface. These communications will use a specially-designed packet structure, shown in Figure 2 below.


**Figure 4: Communication Packet Setup**

The state bits represent what the robot is currently doing. The possible states are listed in the following table:

| | |
|---|---|
| 000 | Mapper Mode - Stopped |
| 001 | Mapper Mode - Started |
| 010 | Controlled Mode - Stopped |
| 011 | Controlled Mode - Started |
| 100 | Scanning |
| 101 | Travelling |
| 110 | Unused |
| 111 | Error |

**Table 1: Robot States**

When the packet is being sent from the robot to the user interface, all sections will contain valid data and the state data will represent the current state of the robot. When the packet is being sent from the user interface to the robot, the Ultrasonic Data section will not contain valid data (should be all 0's), and the state data will represent the new state that the robot will switch to. The robot will decide whether to read the other sections based on the new state (i.e. the robot will read the compass and distance data if told to change to state 011).

States 100 and 101 will not be used by the user interface except as a check that the robot is doing what it should. State 110 is not used.

State 111 denotes that an error occurred. Specific errors are not defined at this time.

This state-based control greatly refines and reduces the needed communication. The user interface could either send out the individual packets telling the robot to stop then switch from controlled to mapper or vise versa, or the user interface could simply send mapper-stopped as the new state and let the robot switch over on its own.

Acknowledgement packets will be sent in the form of the current state, with all other values in the packet 0, unless it is deemed that other information could be useful in the acknowledgement.

The compass value will be an integer from 0 to 3599, representing 0 to 359.9 degrees.

The user's computer will be running a Java program and using a plug-and-play USB Bluetooth Dongle, so no restrictions are anticipated on what computer it could be set up on. Each robot will have a single Bluetooth module operating in the 2.4GHz frequency range. The protocol stantard to be used for this project is Bluetooth v1.1, which is the highest supported by all the chosen wireless modules.

## *User Interface*

Figure 3 below shows the general look that the user interface will have. This window is resizable, so that the user can change the size of the map.

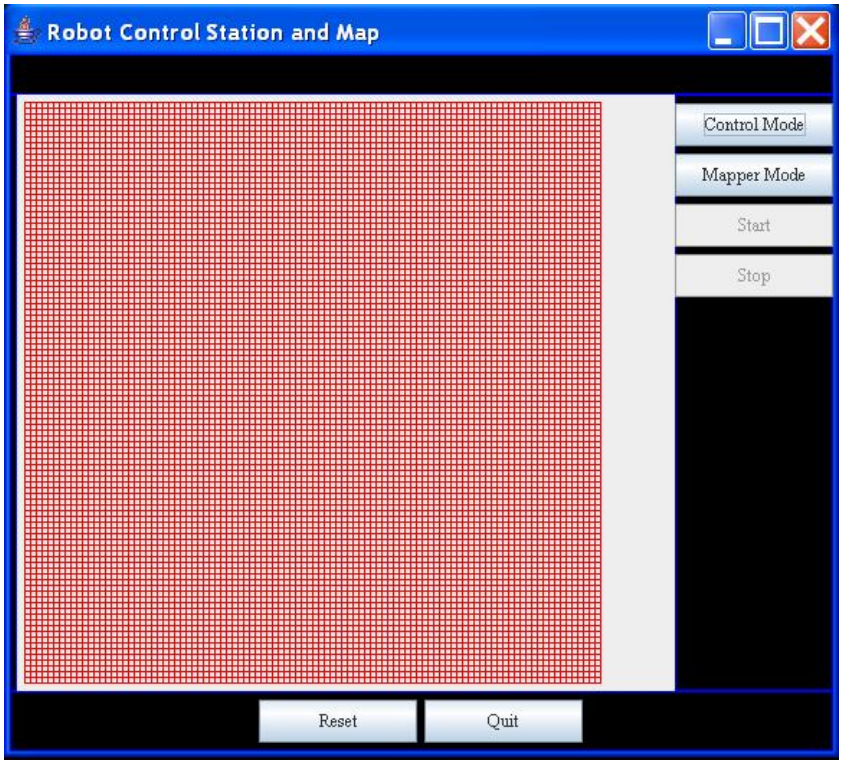


**Figure 5: User Interface**

In this view, the user can control the general actions of the mapping robot. There are four basic functions, plus the ability to change from Mapper mode to Controlled mode, or vise versa. Of the basic functions, two deal with the robot directly, one deals with the interface alone, and one deals with both. In the grid used, each box represents a 5 inch by 5 inch square.

Pressing the start button will cause the robot to either start mapping from the current position or continue idling until a destination is provided by the user.

Pressing the stop button will cause the robot to cease travel after the current instruction completes or to cease scanning after the current scan completes, depending on its current state.

Pressing the Reset button will clear all of the mapped area from the coordinate grid and replace it with black boxes. The coordinates of the robot, however, will NOT be reset. This way the mapping can begin anew without replacing the robot at the start and the new map will directly correspond with the old one. This will also act as if the stop button was pressed.

Pressing the Quit button exits the program. All coordinate data is lost. The robot will act as if the stop button were pressed.

The Mapper Mode and Controlled Mode buttons are used to change the current mode of the robot, to Mapping and Controlled, respectively. In Mapper mode, the robot will follow the mapping algorithm, listed below in the System Functionality and Algorithms section. In Controlled Mode, the robot will wait until it is given a destination. This is done by clicking a location of the map.

The destination must be in a mapped area. The destination cannot be in an object. (objects and unmapped areas will be represented by black areas.) The destination cannot be outside of any predefined boundaries. (The boundary feature may or may not be implemented in the final product.)

Additionally, both for speed and to prevent deadlocks, the user will have the option to click on the map during the mapping process. Doing this will act the same as pressing stop, setting the mode to Controlled, pressing start, clicking on a destination, clicking stop once it arrives, setting the mode to Mapper, and finally clicking start again. Once the robot reaches the last segment of its journey to the destination, a pop-up window will appear asking for the orientation of the robot. The user should input a value from 0 to 359, with 0 meaning to face west, 90 meaning north, 180 meaning east, and 270 meaning south. No change in which buttons are grayed out will occur during this process.

Once the user selects a location for the robot to travel to, in either mode, the software will calculate the best path for that robot to travel to reach the destination, avoiding all objects. For a description of the algorithm used to determine the shortest path, see the System Functionality and Algorithms section.

On startup, the interface will default to Mapper mode and stopped. When Mapper mode is active, the Mapper Mode button will be grayed out and the Controlled Mode button active. They will switch when Controlled mode is active. The same holds for stop and start. When the robot is doing something, the start button will be grayed out and the stop button active, and vise versa when the robot is stopped. Reset and Quit will always be avtive.

## *Testing Strategy*

The testing for this project will be long and difficult. First and foremost, each of the robot's individual components and the wireless interface software must be individually tested to ensure functionality.

The user interface software will be constructed piece by piece and tested for functionality (e.g. buttons send correct signals to network, information is displayed accurately, etc.). It will also be tested by creating a series of fake packets and ensuring that the correct map details are created.

The sensor network will be tested individually for the robot to ensure functional sensors and accurate reading of data from the sensors by the robots. The following is a Built-In Self-Test (BIST) for the robot that will be run upon startup.

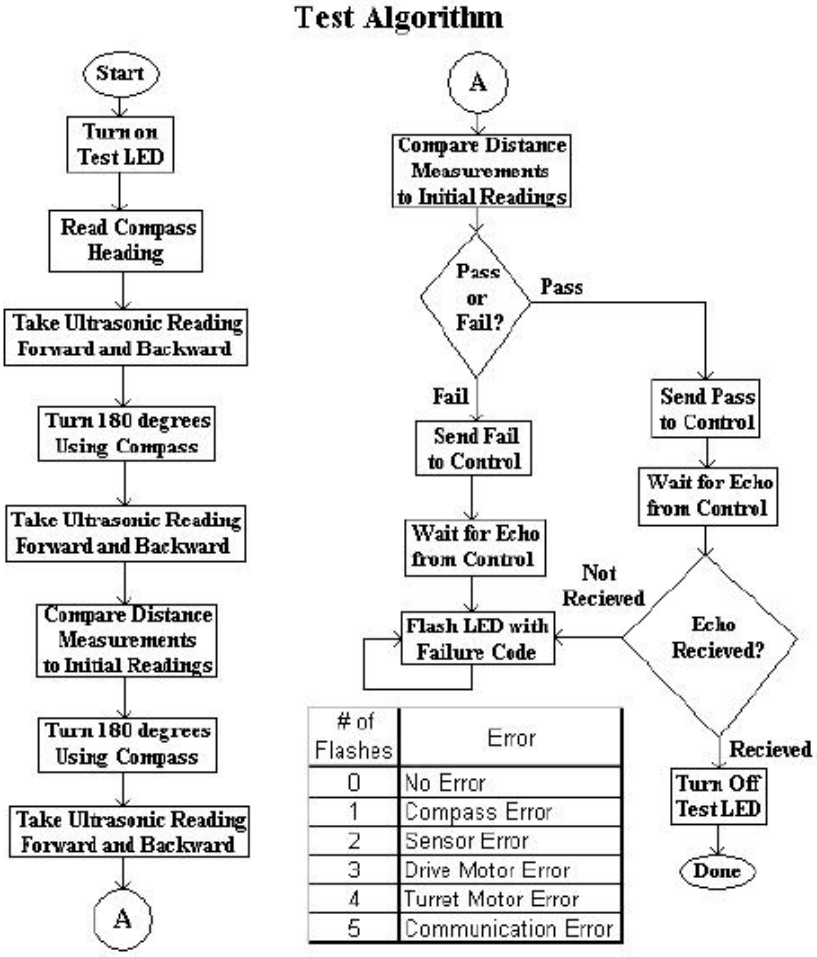


**Figure 6: Robot BIST**

The ability of the robots to make appropriate turns on the carpet must be tested. If it is not plausible, an alternative surface must be created.

Finally the mapping and path finding algorithms must be tested for cases such as open area, impossible to reach destinations, destination out of range, and various types of obstacles.

## *Equipment/Software*

The required components for this project and their cost are listed in the following table.

| Component | # Needed | Cost | Total Cost |
|---|---|---|---|
| Generic USB Dongle | 1 | $ 12.49 | $ 12.49 |
| Parallax BS2p24/40 Starter Kit | 3 | $209.00 | $ 627.00 |
| ROK 101 008 Bluetooth MCM P2P Modules | 3 | $ 1.99 | $ 5.97 |
| Tamiya Tracked Vehicle Chassis Kit | 3 | $ 14.95 | $ 44.85 |
| 12" x 12" Expanded PVC Plastic Sheet | 6 | $ 4.49 | $ 26.94 |
| QRB1134 Phototransister Object Detection Kit | 4 | $ 4.95 | $ 19.80 |
| Devantech SRF08 ultrasonic Range Finder | 6 | $ 59.49 | $ 356.94 |
| Bluetooth Module Prototyping Adapter | 3 | $ 68.40 | $ 205.20 |
| Devantech CMPS03 Digital Compass | 3 | $ 47.00 | $ 141.00 |
| Low Cost H-Bridge CAD | 3 | $ 26.00 | $ 78.00 |
| Computer | 1 | $ - | $ - |
| | | Approximate Project Cost | $ 1,518.19 |

**Table 2: Component Listing With Cost**

This components listing is outdated. A new one will be submitted as soon as it is available. The component listing shown earlier in this proposal is also not complete, it represents only those parts used in the electronics of the robot.

All cost values are subject to change at the will of the companies. All components are subject to upgrading, being made obsolete, etc. at the will of the companies. The listed project cost is not necessarily accurate due to fluctuating currency conversion rates, shipping costs, sales, and bulk purchasing.

The robot will be powered by an 8.4V rechargeable battery.

The software needed for this project includes a Java Runtime Environment (JRE), Parallax free Pbasic compiler, and software to run the USB Dongle. The JRE is a free download and the Parallax compiler and the Dongle software come with the associated products.

## *System Functionality and Algorithms*

Upon starting the demo, the robot will perform a Built-In Self-Test to ensure that the components are functioning as expected. See figure 5 in the Testing Strategy section for details of the BIST. The robot will then wait for instruction from the User Interface. The User Interface will send the command to be a Mapper along with the first command to start. This is because there's no possible location for the user to command the robot to go until at least one mapping scan has completed.

Once the robot receives a start command, it will first take a compass reading, perform a scan. A scan consists of taking a measurement from each ultrasonic sensor, moving the sensors by 10 degrees, in opposite directions from one another, taking another measurement, moving 10 degrees, etc., until each sensor has completed a 180 degree arc. The sensors should be facing opposite directions at all times. The compass and ultrasonic sensor data is sent to the user interface as one packet. If a stop command was issued, this is where it would stop.

Once a scan completes, the robot will move forward by two grid boxes. If a stop command was issued, this is where it would stop. The robot will then ask if mapping has been preformed from this location already. If not, a new scan will commence. If so, the robot will move on two more grid boxes.

If a wall is encountered such that the robot cannot move forward two boxes, it will turn 90 degrees and try again; continuing this process until an opening is discovered or a stop command is received.

If at any time the user clicks on a location on the map, the robot will proceed until reaching a stop point, then follow the same instructions it would if acting as a controlled robot (described later in this section) until that location is reached, then it will rotate to the orientation designated by the user and continue the mapping procedure.

The algorithm just described is given in block diagram format in the following figure:

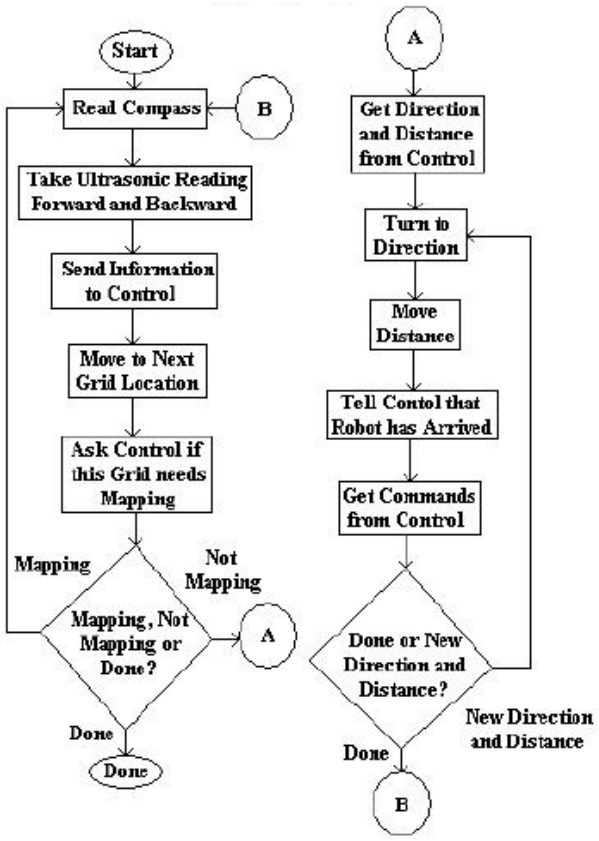


**Figure 7: Mapping Algorithm**

If at any point the robot is switched to Controlled mode, the robot will stop at the first stopping point and wait for instructions. When the instructions have been received, the robot will turn to the designated direction, then move the distance specified. It will then send an acknowledgement to the user interface and wait for another set of instructions. If a stop command is issued, it will stop either after the turn is completed or after the move is completed, dependant on the timing of the command. This algorithm is shown in block format in the following figure:

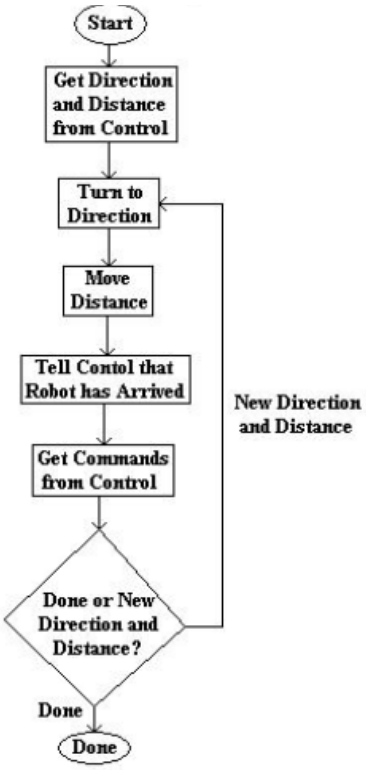


**Figure 8: Controlled Algorithm**

The final and most important algorithm is the path finding algorithm. The purpose of this is to find the best path, or at least a path, from a starting point to a destination point. This algorithm is used both when the robot is in Controlled mode and when it is told to proceed to another location to continue mapping.

1. The path finding algorithm will begin by extending 2 lines from the most appropriate corners of the grid box representing the front of the robot to the most appropriate corners of the box representing the front of the destination of the robot.

2. If either of these lines fall within ½ box of a black area, an algorithm is needed to determine the path around the object.

3. Rotate the bottom line (if rotating clockwise) or the top line (if rotating counter-clockwise) until it is ½ box away from the problem object.

4. Extend a line parallel to the line just created that also passes through the top point of the destination robot position.

5. Using as a reference either a point one box away left one box away right, of ½ box away in both directions, create two parallel lines connecting the top line created in the last step and the bottom line from the original path such that they meet at a 90 degree angle to the bottom original line.

6. Shift the lines created in the last step towards the robot's current position an amount delta. In doing so, keep the distance between the two moving lines equal, at one box width apart. Also keep the lines connected at both ends. The lines will most likely either grow or shrink during this process.

7. Check to see if either of these lines pass within ½ box of a black area. If so, move by another delta. Repeat until the lines no longer touch a black area.

8. Starting from the robot, check each straight line segment to see if it passes through a black area. If it does, assume that the robot's current position is the beginning of the last segment before the problem object and perform the algorithm again.

Notes:

1. Due to the buffer space used and the angles chosen for the connection, (step 5), it is possible that a potential path may be missed.
2. When rotating the lines, if they should not encounter the edge of the problem object before rotation 180 degrees from the line representing the straight-line path, switch to rotating the opposite line in the opposite direction. If the same occurs, assume that no path is possible.

## *Analysis of Possible Difficulties*

One integral problem possibilty is that all of the distance calculations will need to be done in tread holes. This could lead to issues of rounding error and potential of lost information if the robot stops part way between tread holes. The best solution to this at the moment is to simply always use a interger number of holes.

Tread slippage will be a very real problem as well. This will be partially compensated by having a sensor on each tread and comparing the data. Additionally, the expected direction faced will be periodically compared to the reading from the compass. These two methods should remove a significant portion of this error.

Another possible problem is the operation and synchronization of the ultrasonic sensors. This can only be resolved through in depth reading of the manuals for the sensor, combined with a trial and error procedure.

A possibly major problem is the algorithm used to map the area. It will be difficult to ensure that the full area is indeed mapped, especially with particular setups of objects. This will be addressed with an additional user input allowing the robot to be redirected during the mapping procedure.

A definite problem area will be ensuring that the path alrogithm will find a way to get to the destination if one is available. A possible solution will be to switch to an algorithm found online, but additional research into how to implement that algorithm would be needed.

The tread grip on various floor surfaces will be very difficult to compensate for if it becomes a problem when turning. This will attempt to be controlled by specifying what terrain the robot will be traveling on.

## *Deliverables*

The project demo will have the following format:

1. Demonstrate mapping of an area
2. Demonstrate stopping the mapping
3. Demonstrate continuing the mapping
4. Demonstrate sending the robot to a location while mapping
5. Demonstrate switching the robot to Controlled mode
6. Demonstrate sending the Controlled robot on an easy path
7. Demonstrate sending the Controlled robot on an difficult path
8. Demonstrate sending the Controlled robot on an impossible path
9. Demonstrate resetting the map
10. Demonstrate exitting the program

## Time Line

| Item Description | Primarily Responsible Member | Completion Date |
|---|---|---|
| Setup User Interface | ME | 12/6/2005 |
| Finalize Project Algorithms | ALL | 1/12/2006 |
| Conclude Bluetooth Research | ERS | 12/16/2006 |
| Create Website Template | ERS | 12/13/2005 |
| Aquire All Hardware | TCL | 12/16/2005 |
| Implement Buttons/Map Function | ME | 12/16/2005 |
| Test Component Functionality | TCL | 1/12/2006 |
| Implement Mapping Software | ME | 1/13/2006 |
| Construct Robots | TCL | 1/14/2006 |
| Write Avoidance Alrorithm | TCL | 1/13/2006 |
| Rewrite Project Proposal | ERS | 1/20/2006 |
| Implement Packet interface | ERS | 1/12/2006 |
| Test Avoidance Algorithm | TCL | 1/15/2006 |
| Write Path Finding Algorithm | ME | 1/14/2006 |
| Test User Interface Functionality | ME | 1/15/2006 |
| Test Path Finding Algorithm | ME | 1/20/2006 |
| Find New Java Interface Tool | ME | 1/31/2006 |
| Update website | ERS | 1/31/2006 |
| Test Terrain Mapping | ALL | 1/31/2006 |
| System Functionality Testing | ERS | 2/10/2006 |
| Write Final Report | ERS | 2/15/2006 |
| Demo Poster | TCL | 2/17/2006 |
| Team Assesments | ALL | 2/22/2006 |
| Finish Website | ERS | 2/24/2006 |

**Table 3: Project Time Line**

## Conclusion

This project is intended to showcase several things, including our ability to follow standards (Bluetooth v1.1 and $I^2C$), our ability to both build physical systems and create programs, and our desire to create something that can be the basis for bigger and better things.

## Potential Upgrades

The possibilities for upgrading a system such as this are nearly endless. Here are a couple ideas:

1. Add camera(s). The basic functionality given by this project has one major flaw: The inability to determine what those black spots actually are. If cameras were added to the robots, a much more extensive mapping could be done, displaying not only what the black areas represent, but also any patterns and objects not detectable by the ultrasonic sensors on the ground or up in the air.
2. The next Mars Rover. Especially if coupled with the camera(s), a version of this robot could be remotely controlled and explore another planet. Obviously something with a greater range than Bluetooth wireless would have to be used for communication, however.