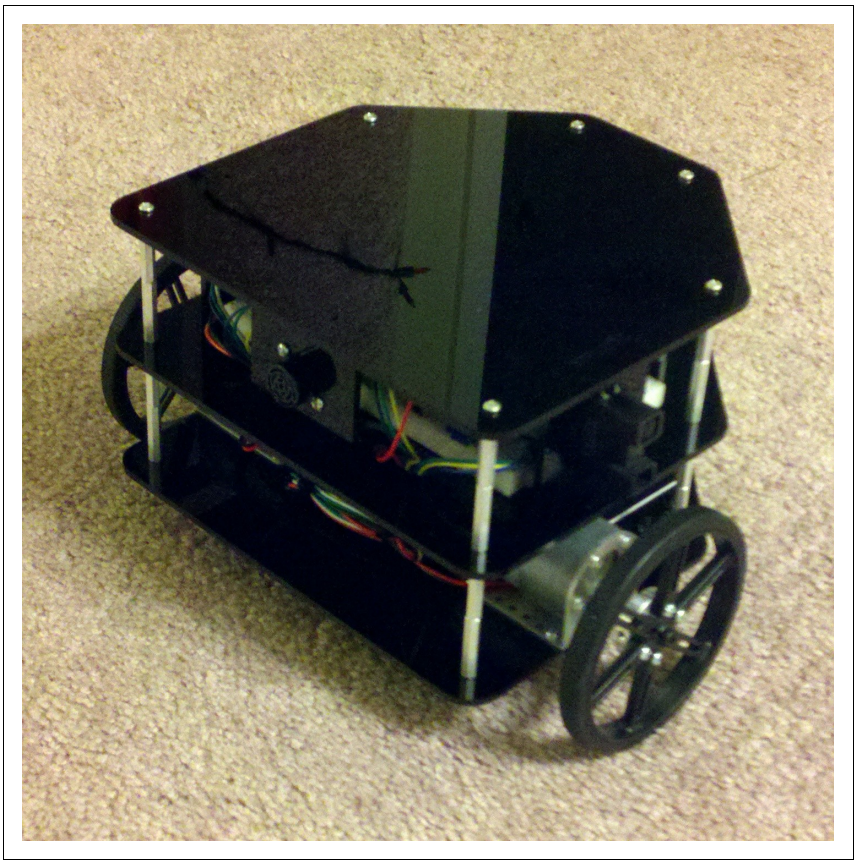
ECE791/792 First Semester Progress Report
Project Team: Justin Woodman
ECE Faculty Advisor: F. C. Hludik
Current Date: December 14, 2011
Project Completion Date: May, 2012

Autonomous Room Mapping Robot System



General Problem Definition

There are many instances where someone may want to know the precise layout of a room or other indoor space, but does not have access to the blueprints. In some cases a person could enter the room and gather this data manually, but this is a tedious process and it requires the room to be well lit, safe, and have a large enough entrance. If there was a device which would eliminate all of these difficulties it could be of great use to emergency personnel, contractors, law enforcement, and military.

The goal of this project is to design and implement an affordable system which will allow its user to quickly and easily obtain a map of any indoor space. This will be done by designing a semi-autonomous robot which can communicate with the user via a wireless interface and computer software. The robot itself will utilize a variety of distance sensors to gather information about the room relative to its position. The robot will be able to localize itself relative to its starting point by using rotary encoders on each of its motors. These encoders will give the robot's microcontroller a measure of how far each wheel has turned, which can be translated into distance traveled.

The computer software will communicate with the robot via a wireless serial link allowing the user to issue certain commands to the robot, which it will carry out. The robot will be able to send its findings back to the user's computer, where the software will plot the data as a map of the room. The robot should also be able to send technical data about its motors, battery, and other hardware back to the user's computer, so that the software can display the information in a meaningful way for the user.

Specific Design Objectives

This project is comprised of two primary parts, the first part being the robot itself. The robot is affordable, yet robust and capable. Three different sensor types are being employed by the robot in order to accurately report data about the room. The first type of sensor is a magnetic rotary quadrature encoder attached to the output shaft of each motor, before the gearbox, with 1856 ticks per wheel rotation. This allows the speed, direction, and position of each motor to be determined, for the robot's microcontroller to determine its relative change in position. Because no information is available about the room at the start of each “mission”, the point at which the mission starts is considered the origin (0, 0). The robot then finds its displacement relative to this position as it moves.

The second type of sensor is an ultrasonic rangefinder mounted to the front of the robot. Ultrasonic rangefinders can give a fairly accurate measure of distance, but they typically have a large beam width which means that the distance reported is the closest thing to the sensor that falls within its beam. Therefore the rangefinder is not intended to gather any data about the room, but simply to allow the robot to determine if it is about to hit something. An ultrasonic rangefinder with a beam width slightly wider than the robot's chassis has been selected.

The third and final type of sensors are infrared rangefinders. Like the ultrasonic rangefinder, infrared rangefinders give a measure of distance. However, their beam width is almost infinitely narrow which means that they report the distance to a single point on the object they're aimed at. Two of these are being used, one on each side of the robot. As the robot moves both of the infrared rangefinders will be capturing distance continuously and sending this data, along with the robot's position, back to the user's computer.

The second primary part of this project is the computer software. This software's primary purpose is to plot the sensor data for the user. Since the robot reports its position and the distance reported by its infrared rangefinders, the map of the room will be plotted pixel-by-pixel. To make this useful, the software also has to allow the user to control the robot. Ideally the user should be able to click an area on the map, and the robot should try and go to that location reporting distance data along the way while also avoiding collision autonomously. This allows the user to be in control of gathering the data they feel is necessary, without having to deal with a difficult and cumbersome joystick type controller.

Design Process

The design process for this project thus far has been entirely focused on the robot hardware, circuitry, and its microcontroller software. Before the proposal for this project, a chassis was designed using Google SketchUp. The chassis is made up of three sheets of laser acrylic, manufactured by Ponoko, and separated by 2” standoffs. This three tiered design allows the motors and batteries to be placed on a separate layer from the microcontroller and its peripherals, helping reduce electrical noise. It also allows the sensors to be elevated enough on the second level so that they see over the wheels.

The sensors are placed so that their beams are collinear with the robot's center of rotation, which will make some of the sensor math simpler. The ultrasonic rangefinder is placed directly on the front of the robot, allowing it to find the distance to the nearest object in its path of motion. Two infrared rangefinders are used, each placed perpendicular to the robot's path of motion on the left and right side. This design allows the robot to continuously find the distances from the walls of the room as it moves forward. These distance readings will be sent continuously back to the user's computer along with the robot's current position and angle of rotation.

The current microcontroller and sensor circuitry for the robot is constructed on a breadboard which is adhered to the second level of the chassis allowing changes to be made. After the circuitry is satisfactory, a printed circuit board will be manufactured.

The microcontroller software is being written in Spin, a proprietary programming language designed exclusively for the Parallax Propeller. Each time a change is made in software, it is tested on the actual robot. The robot is kept propped up when testing in case of a coding error which could be damaging to the robot if it were allowed to move freely. Once a change in code has been tested on the propped up robot and it can be seen that the change will be safe, the robot it allowed to move freely which shows that the measurements it reports are accurate and that it moves correctly.

This “alter and evaluate” style of coding design will likely continue all the way through the completion of the computer software. Once the project is “feature complete”, meaning that all functionality of the project is implemented, the remaining time will be used to tweak the code in order to get best usability, accuracy and reliability.

Implementation and Testing Plan

The implementation of this project has been broken into two parts: the hardware (robot) and the software. Because of the fact that the robot's on-board software (microcontroller) must communicate with the computer software, these two implementation tasks will need to overlap quite a bit.

The robot itself is powered by eight AA (1.5v) batteries, which supply the robot's electronics with 12 volts. This allows the robot to accept rechargeable and non-rechargeable batteries. The robot's motors are two 12v Pololu gear motors each with a magnetic rotary quadrature encoder. An H-bridge motor driver IC manufactured by Freescale Semiconductor, the MC33926, is used to allow the robot's microcontroller to safely interface with the motors.

The robot's microcontroller is an 8-core, 80MHz Parallax Propeller. Due to the multi-core nature of the Propeller, entire processor cores can be dedicated to monitoring sensors, ensuring uninterrupted results. Voltage regulators are used to bring the robot's 12 volt power source down to the 3.3 volts that the Propeller and all additional components require.

The two types of rangefinders that are used, ultrasonic and infrared, interface with the microcontroller as well. The ultrasonic rangefinder is a sensor from Maxbotix, chosen for its reliability and ease of use. The infrared rangefinders chosen are GP2Y0A02 sensors from Sharp. Both of these sensors output their distance as an analog value. Unfortunately the parallax propeller microcontroller does not have built in Analog to Digital Converters (ADC). This means that a separate ADC IC, the Microchip MCP3202, is necessary.

The overall system employs a wireless link between the robot and computer software. Because both the computer and microcontroller speak serial, an Xbee pro transceiver is used on both the robot and the PC. These particular transceivers have a range of up to a mile.

The computer software will be written in Python so that it is easy to develop and is cross-platform. Because the computer is much faster and more powerful than the robot's microcontroller, most of the data processing will be done on the computer before being plotted. A simple graphical interface will be provided for the user so that they can focus as little as possible on any technical aspects of the device.

Testing has taken place as the system is being developed. Initially, the circuitry of the robot itself was constructed and verified. It was important to test the motors and ensure that they could be properly controlled from the microcontroller. All of the sensors have been tested and verified, and their interface with the microcontroller was tested and verified as well. Finally for the circuitry, the wireless

link was tested. Its ability to communicate data between the microcontroller and the computer must was verified and was found to work well.

Once the circuitry was complete and tested, it was all be mounted on the chassis to make the robot mobile. At this point, software for the microcontroller has been written which allows the robot to be moved to a specific set of coordinates. Now that this is done, microcontroller code must be added so that the data from the sensors is continuously relayed back to the computer while the robot is moving.

Implementation and testing of the computer software will follow. The important first goal is allowing the user to click a location on the software's map display area and send the robot there. After this is working, the computer must be able to receive and plot the sensor data. Because this is the final important feature to implement, refining the design to make it as usable and reliable as possible will follow.

Evaluation of Progress

The progress made thus far is consistent with the original project timeline. The preliminary experimentation and design was done prior to the proposal, and allowed a quick start of the project design and construction. The written and oral proposals were accepted, as which point work had already begun on some construction.

After deciding that the general model for the robot was to be a three-wheeled differential drive system with two active wheels and one caster, the first major step was the design of the chassis. This involved sourcing parts, analyzing the datasheets, and taking measurements to ensure that everything would fit onto the chassis in a small, affordable and sturdy way. The chassis design chosen was a three-tiered modular-style platform made from laser cut acrylic sheets manufactured by Ponoko. The actual layout for the design is pictured below on the left, while the assembled robot prototype is pictured below on the right:

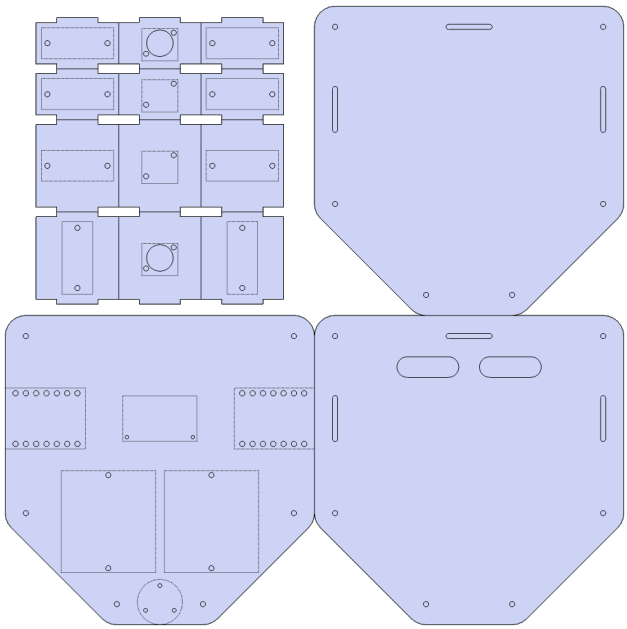


Figure 1: Chassis design file

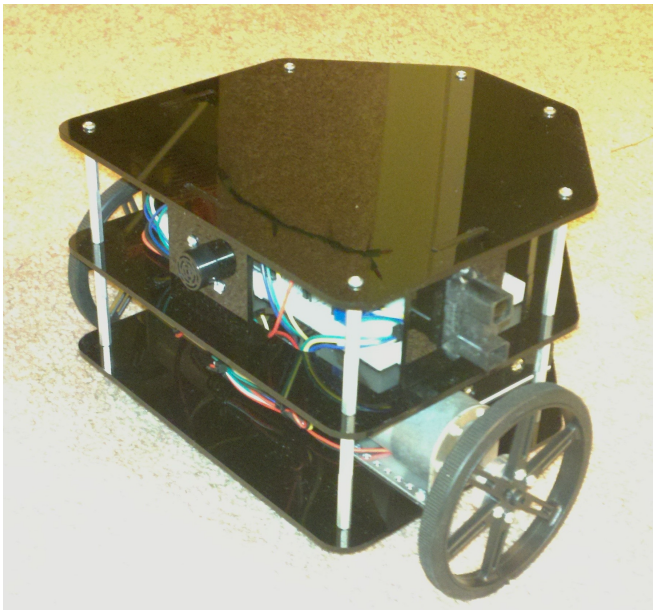


Figure 2: Assembled prototype robot

As can be seen from figure 1 above, the sensors are mounted on small acrylic rectangles with tabs. The tabs fit into slots cut on the second and third levels, which allows the infrared rangefinders on the sides to see over the wheels. The motor driver PCB is mounted directly between the motors so that the motor wires can be kept as short as possible, and so that the signal wires to the driver can be run up to the second level through the two oval shaped holes. A spherical caster is used on the back to ensure

three points of contact with the ground. The batteries are mounted directly between the caster and the motors, so that the center of gravity is equally placed in the center of the three contact points. The second level contains all of the circuitry for the microcontroller, sensors, and wireless serial bridge.

The majority of the electrical design of this project was done after the design and assembly of the chassis. The entire circuit, besides the motor driver, has been assembled on a breadboard affixed to the robot's second chassis level. The control lines for the motor driver (including PWM, H-bridge inputs, and enable lines) are connected to the breadboard via jumper wires. The motor encoder outputs are connected via screw terminals to the breadboard as well. A picture of the robot from the front, showing the connections from the lower level to the second level, can be seen below on the left (figure 3), while a top image of the robot with the third level removed shows the breadboard with all current connections (figure 4). Please note that the connections for the ADC and sensors are not made yet.

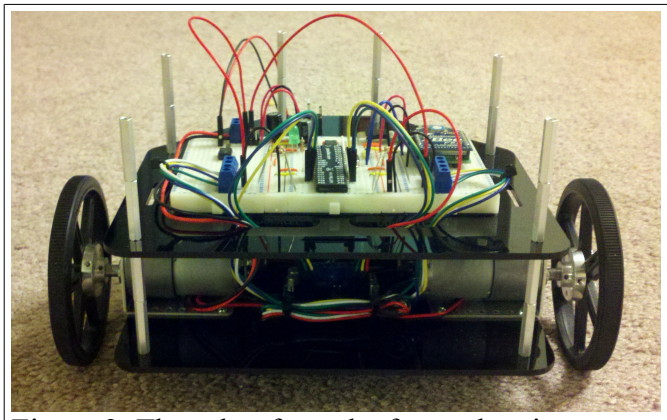


Figure 3: The robot from the front, showing connections from the lower level to the upper level

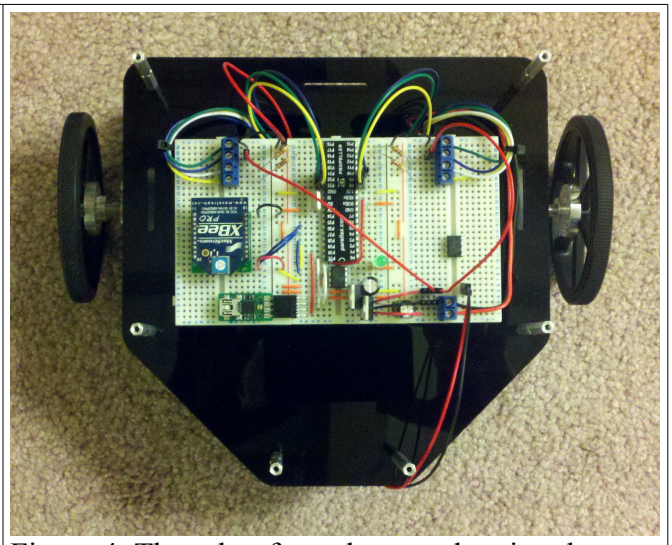


Figure 4: The robot from the top, showing the breadboarded prototype circuit

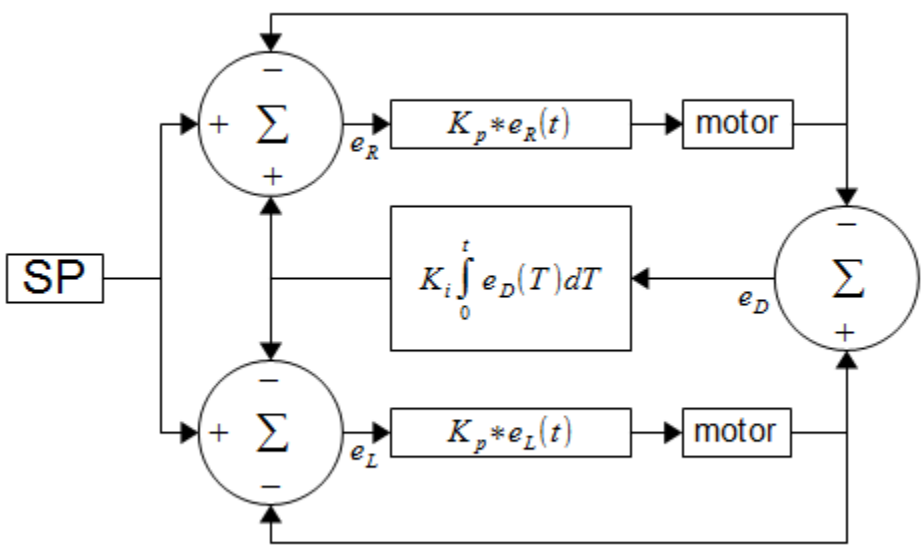
The current microcontroller software accepts three parameters from a serial terminal which has been set up to communicate through the XBee transceiver. The first parameter is the desired angle for the robot to rotate. This can be any angle between -360 degrees and +360 degrees. The software rotates the robot until the actual angle matches the desired angle. The actual angle of the robot is determined using the following equation:

$$\Delta\theta = 2\pi \frac{R_W}{D} \frac{T_1 - T_2}{T_R}$$

Where R_W is the radius of the wheel, D is the distance between wheels, T_1 is the number of ticks recorded from the left encoder, T_2 is the number of ticks from the right encoder, and T_R is the number of ticks in a full rotation of the wheel.

The second parameter accepted at the serial terminal is distance. After the robot rotates, it travels forward this distance. This will allow the computer software to send the robot to any possible location using trigonometry and the Pythagorean theorem given a set of coordinates.

The third parameter accepted is an RPM value. This is the RPM at which the motors will rotate when the robot is moving to its desired location. Maintaining a constant RPM requires a feedback control system. A simple proportional control system with a gain of 1 was decided on to maintain the RPM of each motor. However, this creates a problem when the robot is trying to move in a straight line. Because of the oscillatory nature of proportional control, the two motors were not necessarily rotating at the same speed at all times, causing the robot to swerve back and forth invalidating the angle it had rotated to. The solution for this was to keep track of the difference between motor speeds, and integrate it. This was then summed with the proportional control system error for each motor. This keeps both motors at the same speed, while still allowing the RPM to be set. The diagram below illustrates the system used:



Where SP is the desired RPM and the outputs of the motors are their actual RPM. When Kp is set to 1 and Ki is set to 1/10, the robot moves in an almost perfectly straight line. Any discrepancies at this point could probably be attributed to bumps on the terrain and slippage of the wheels.

The next step will be to connect the sensors to the microcontroller and to start writing the computer software. After those tasks are complete, the process of refining the robot will begin. This will likely last until the system is considered perfect, or there is simply no time left.

Project Timeline

ID	Task Name	Start	Finish	Duration
1	Circuit Construction	9/1/2011	1/31/2012	21w 6d
2	Microcontroller Programming	10/1/2011	2/29/2012	21w 5d
3	Mechanical Design and Manufacture	12/1/2011	2/29/2012	13w
4	Computer Software Programming	11/1/2011	3/31/2012	21w 5d
5	PCB Layout and Manufacture	2/1/2012	3/31/2012	8w 4d
6	System Tweaking and Refining	3/1/2012	4/30/2012	8w 5d