The Worksite Leveler Robot: A Comprehensive Report

Team Members:
Jeff Laut
Ronald Poveda
Victor Wassef

CONTENTS

Abstract……………………………………………………………………3
Introduction/Purpose…………………………………………………………4
Bill of Materials………………….……………………………………….…..6
Design Analysis
Mechanical Design…………………………………………..………....7
Electrical/Circuitry Design……………………………….................11
Early Problems/Solutions……………………………………………..21
Prototype Cost Analysis/Mass Production……………………………..23
Conclusions/Improvements/Suggestions…………………………………24
References……………………………………………………………………26

Abstract

In this report, a Mechatronics project named the Worksite Leveler Robot will be introduced, which will be referred to as a type of Autonomous Construction Vehicle (ACV). This robot was designed to operate on an uneven, rugged construction site, take readings of the different elevations (peaks and valleys) of the desired site and place it on a contour map. This will help facilitate construction site preparation without manpower and human risk of injury. The project required the implementation of many different software programs and hardware, particularly the Basic Stamp 2 (BS2) and corresponding software created and developed by Parallax, Inc. [1]. This particular robot was created and is suited for highly technological construction projects so that manpower, injuries, and fatalities will not be a concern during construction site development. This project required the use of several different commonly-used hardware and software engineering programs in its design and construction, all of which will be mentioned and illustrated in this report.

Introduction/Purpose

Microcontrollers are prevalent in many of the small, commonly-used devices that people interact with everyday. Devices ranging from cell phones and computers to household items such as toasters, microwaves, and ovens contain microcontrollers that control their respective operations. Most of these tasks operate autonomously, that is, they do not require human intervention or input while performing their respective tasks. An example of this would be a smart sweeper/vacuum robot, which would vacuum all the floors of a household while avoiding household obstacles and structures, simply by turning it on.

Figure 1: iRobot's [2] "Roomba" autonomous house vacuum

The specific microcontroller of interest of this project is the Basic Stamp 2 (BS2), a line of microcontrollers created and developed by Parallax, Inc. Due to its simple, easy to use microcontroller platform, and readily accessible memory types (ROM for interpreter firmware, EEPROM for user programs, several bytes of RAM, etc.), this specific microcontroller was selected. The BS2 is connected onto a circuit board named the Board of Education, which enables the user to create a desired circuit and enable the circuitry to measure digital and analog signals and conditions as well as carry out designed tasks, much like a Data Acquisition and Control (DAC) board. A program must first be created using the corresponding program editor software written in Parallax Basic, or PBasic, which is a language similar to Basic computing language. Once the program is created on a PC, the program must then be downloaded onto the Basic Stamp via USB. Given that the BS2 can be reprogrammed again and again, the use of a microcontroller on any device is quite flexible when it comes to performing automated tasks.

Many automated systems, as previously stated, are used in today's society. The purpose of this particular project is autonomous work in a rugged environment, where the concern is use of manpower and safety on a construction site. Lately, there has been much concern over construction workers being injured or fatally wounded in construction site accidents. In an effort to try and un-man these relatively unsafe sites, autonomous robots can be used to carry out certain tasks that puts a robot in the line of danger, instead of a man. Many robots have already been conceived around autonomy in the construction site workplace, such as concrete distribution robots, as shown below.

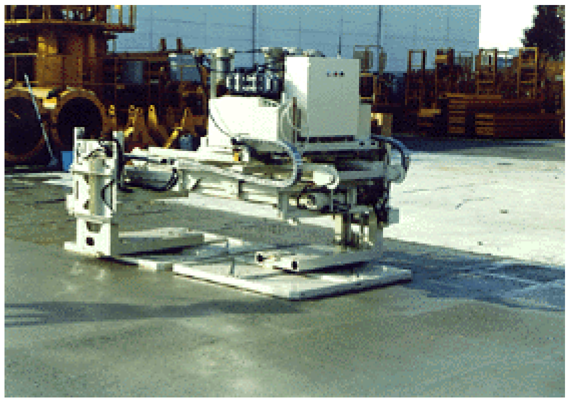
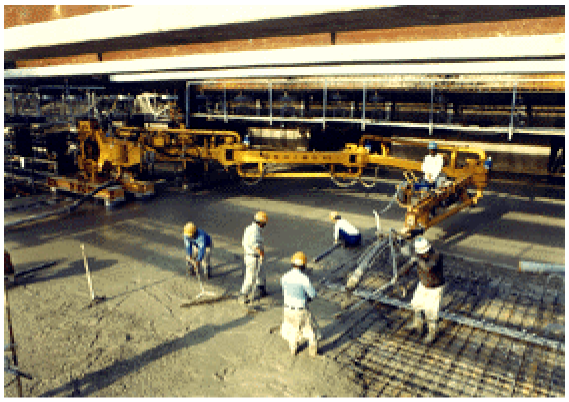



Figure 2: (From left to right, respectively) Concrete Surface Treatment Robot and Concrete Horizontal Distributor [3]

This project was conceived around the idea of an autonomous construction vehicle (ACV), modeled after a typical bulldozer/excavator type of construction vehicle. The robot is designed and programmed to map out the contours of the site in an effort to level the worksite. Through the use of analog and digital sensors and DC servomotors, the robot will first record the different elevations of the site and develop a contour map as it moves around the site.

The Worksite Leveler Robot is intended to be designed for construction companies who are technologically centered and looking for construction safety improvements, as well as construction work efficiency with little or no manpower required.

Bill of Materials

1.) Parallax Circuitry Materials
    a. Basic Stamp 2 Microcontroller Chip - $50.00
    b. Board of Education - $70.00
    c. Basic Analog and Digital Parts Kit - $30.00
        I.) Resistors/Potentiometers (ranging from 100Ω to 100kΩ)
        II.) Capacitors (insulator and electrolytic, ranging from 0.1μF to 3.3mF)
        III.) 8-bit A/D converter chips
        IV.) 555 timer chip
        V.) 3" Jumper wires/leads (6)
2.) Parallax Tank Tread Kit - $35.00
    a. Sprockets/gears
    b. Rubber treads
    c. Screws/Nuts/Bolts/Extra Hardware attachments
    d. Metal panel attachments (2)
3.) Sheet Metal Chassis - $5.00
4.) Tower structure/Coil and Spool Chassis Manufacturing - $5.00
5.) Miscellaneous - $43.00
    a. 1 Breadboard - $5.00
    b. 2 A/D converters - $6.00
    c. Extra Jumper Wires/Leads - $2.00
    d. Telephone wire - $6.00
    e. Rubber Bands (large) - $1.00
    f. Glue Gun (w. Glue) - $10.00
    g. 2 single turn potentiometer (100kΩ) - $9.00
    h. 10-turn potentiometer (100kΩ) - $13.00
    i. EEPROM chip - $1.00

Total Materials Cost: $248.00

Design Analysis

Mechanical design

For the project, a moving robot similar to a construction vehicle was intended to be deigned to run on rough terrain. The robot's initial design considerations were as follows:

1.) The robot should have a durable metal chassis to protect against the elements, terrain, rough movement, etc.

2.) The robot should be moved around on wheels/treads instead of appendages or limbs, much like a typical construction vehicle to grip the surface. The wheels/treads should have one motor to each side of the robot for more power and control when moving around in dirt, sand, etc.

3.) The electronics of the robot (Board of Education, BS2) must be protected within the chassis.

Looking at these considerations, the robot was designed using a 3D computer aided design (CAD) program called Solidworks® 2007, developed by Solidworks Corporation [4]. This program was used to create initial designs of mechanical components, put the components together, and observe how the components would move. From there, the action range and effectiveness of the components of the design could be observed as if it was actually built in front of us.

A large chassis would help support the Board of Education, Basic Stamp, extra breadboards and/or any additional structures that need to be placed on the robot. However, it would be difficult for the robot to detect smaller and subtle changes in terrain elevation if the robot chassis was too large, so the robot chassis was initially designed to have more length than width to its structure. An initial chassis design drawn using Solidworks is shown in Figure 3 below. As seen below, the DC servomotors were created to scale in Solidworks to obtain an idea of how each motor would fit onto the chassis.

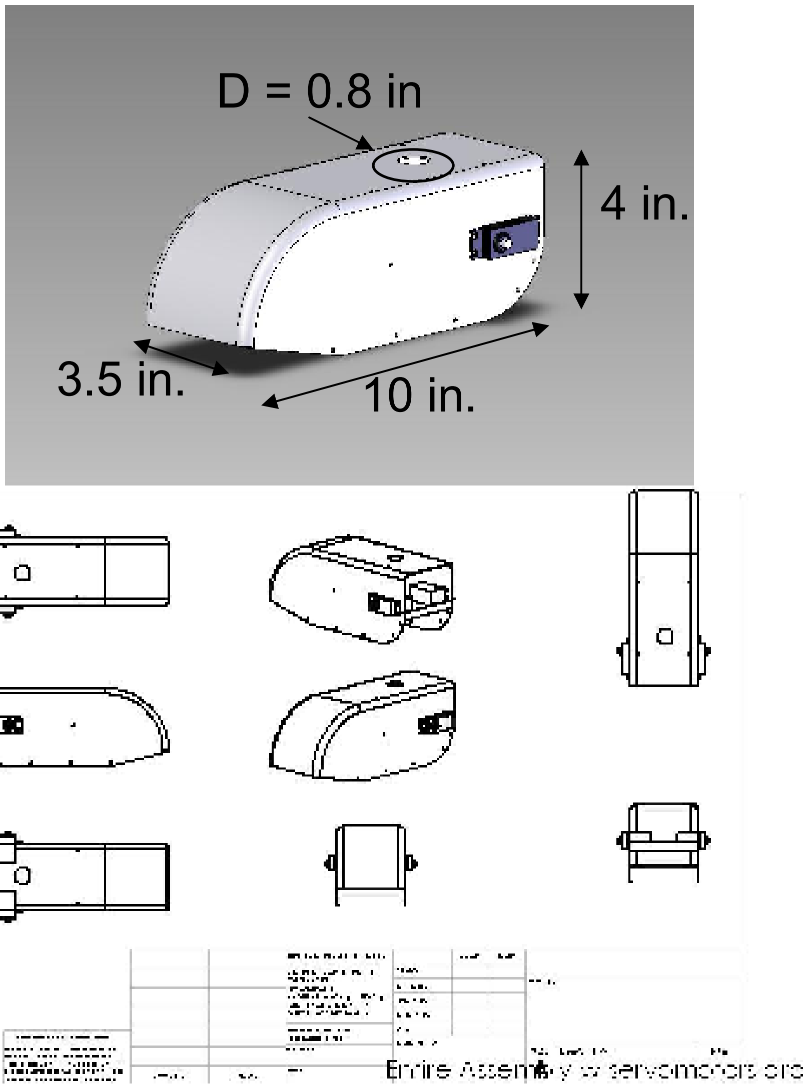


Figure 3: Initial Chassis Design using Solidworks® software (with drawings)

Two DC servomotors were placed in the back of the chassis because a tank tread gear train was planned to be constructed for each side of the robot. Since each servomotor occupied a considerable amount of space inside of the chassis, careful position of the Board of Education would have been crucial to this design.

The tread gears for the robot are composed of six spur/sprocket gears placed on each side of the robot, which were all connected by rubber treads on each side. One of the gears on each side was attached directly to the servomotor. The position of the servomotor on the sides was significant. If the motor on each side was placed too low on the body of the chassis, there may be the possibility of unintended gear/motor shifting due to intense vibrations felt if the motors are close to the ground. Therefore, treads are a recommended choice for this robot, due to the flexibility of motor placement.

The final design that was conceived involved geometric changes of the chassis design. It was decided that the robot needed to cover more ground in order to run on terrain properly and take accurate elevation/distance measurements. Therefore, the chassis was created to be wider and slightly shorter, as shown below with a 7.5 in.×2.1 in.×5 in. chassis.

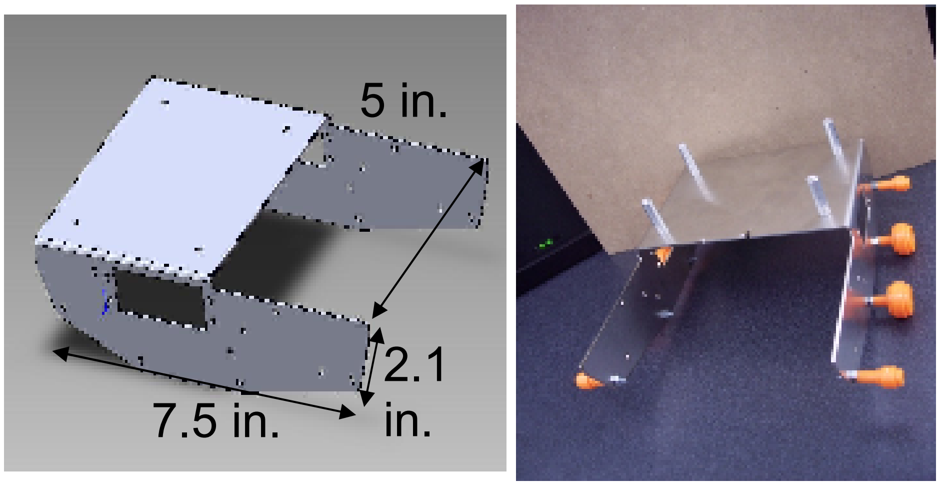


Figure 4: Final Chassis Solidworks Design and actual chassis with gears

A special structure was designed for the robot using Solidworks, which is referred to as a locator tower. This structure was created and used to detect angle measurements and convert them into different distances and elevations of the construction site. The circuitry details will be covered later in the report under the section labeled "Electrical/Software Design". The mechanical design of the robot, however, will be discussed. The locator tower has 2 rotator spools run on bearings. Each spool contains a single turn potentiometer, and is positioned in the horizontal and in the vertical plane. The tower is normally placed on a designated fixed area, but it is attached to a coil-and-spool structure on the chassis via a stripped telephone wire, which will be discussed later in the "Electrical/Software Design" section. The locator tower was also designed using Solidworks, and it is shown in Figure 5 below. The most current design contains all of the necessary wiring, as well as a soldered telephone jack attached to the vertical spool.

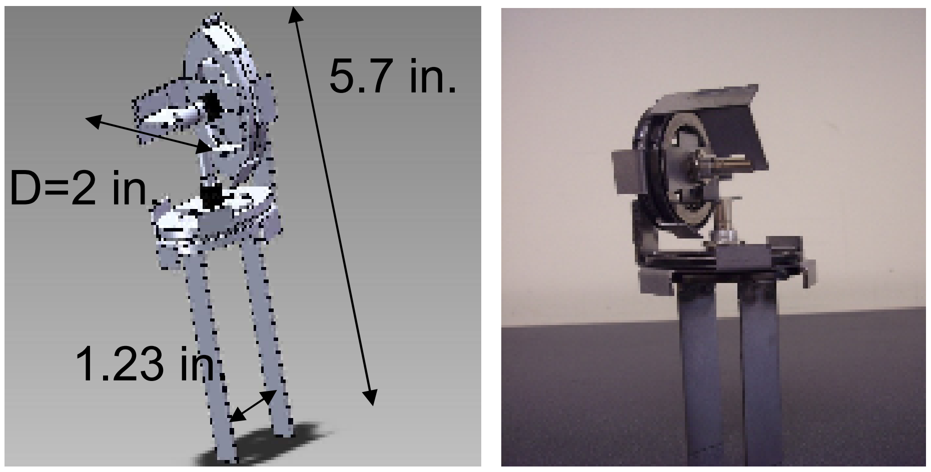


Figure 5: Locator Tower Solidworks design and actual tower

The coil-and-spool structure was also designed using Solidworks, so that constant tension can be maintained throughout the wire as it is extended. A spring from a 16 ft. tape measure was built in to the design, as well as a 10-turn potentiometer to measure the distance. It is shown in Figure 6 below.

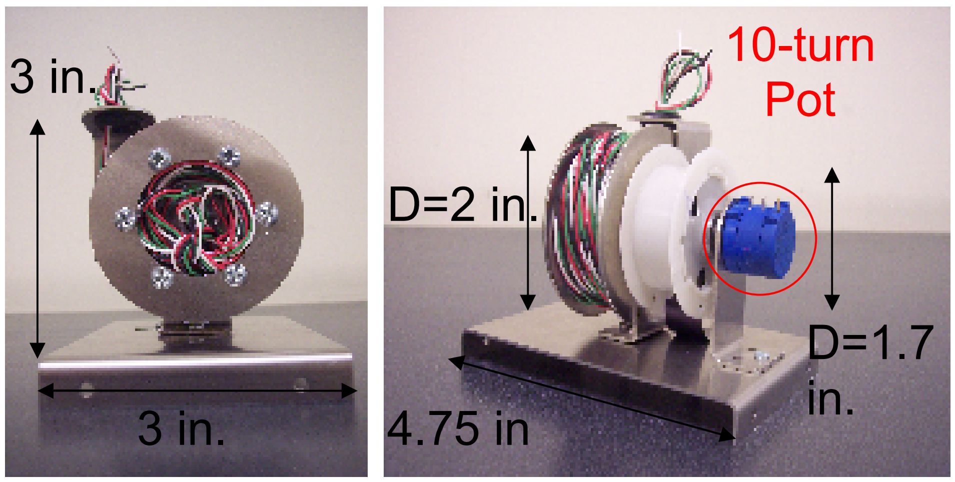


Figure 6: Coil-and-Spool structure

Using all of these structures, the robot is able to measure elevation and distance along a specified area. The completed robot is shown in Figure 7 below.

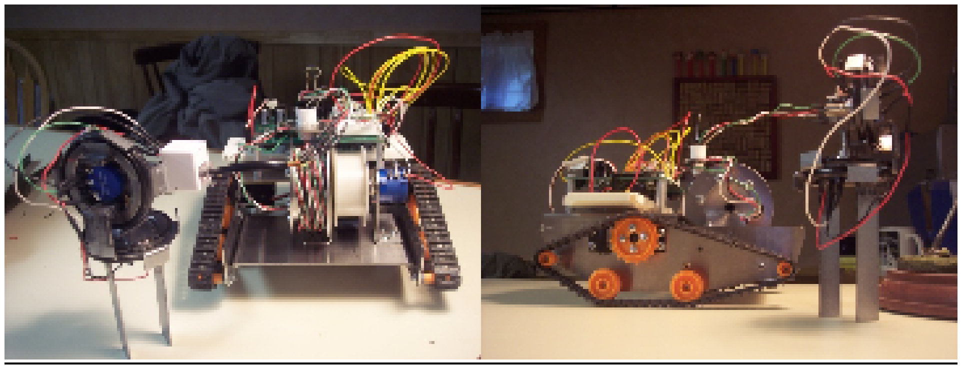


Figure 7: Completed Leveler Robot with tower

Electrical/Software Design

Several electrical components were considered and used in the circuit design. On the robot, resistors, potentiometers, A/D converters and 555 timers were incorporated into the robot's circuitry. For the leveler robot, the main concern was accurate navigation. The process of autonomous navigation of this robot would mostly be dictated by how we can design and develop an appropriate and efficient circuit control system that would move the robot and accurately detect position and elevation on all 3 coordinate axes as the robot traverses through the site. The circuit system would also require digital sensors, special electrical components, and appropriate P-Basic programming in order to differentiate between higher and lower elevations along the site.

For the motor portion of the circuitry, it was decided to control the motors without using direct BS2 programming. Instead, an independent circuit would be built for each motor, which would enable the user to simply place a high or low voltage (Vcc) to each circuit to move the servomotors at a high power. This was done by creating each circuit using a 555 timer in astable mode, which is known as an astable pulse train generator. It includes 2 resistors, a ceramic capacitor, a 555 timer, and a DC servomotor directly connected onto the breadboard. The circuit would send pulses to each servomotor with certain pulse widths that would transmit a high amount of power. The diagrams for the 555 timer and for each circuit are shown below.

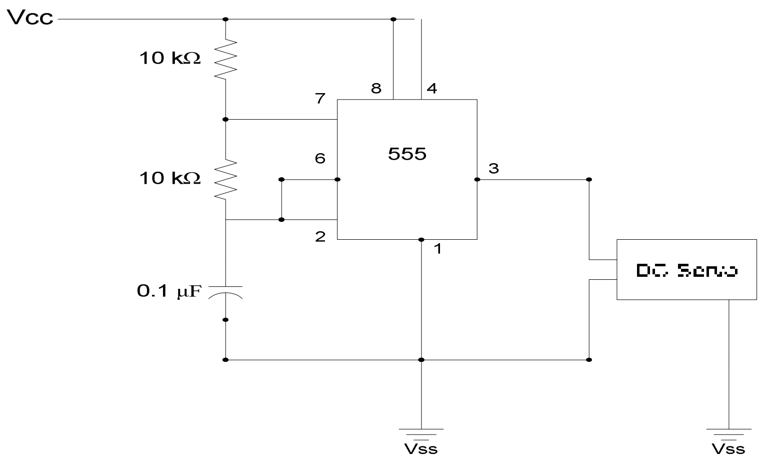


Figure 8: Right Motor Astable Circuit

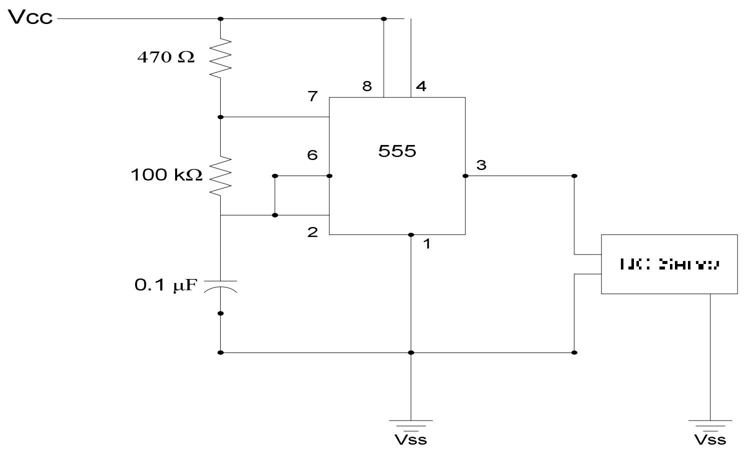


Figure 9: Left Motor Astable Circuit

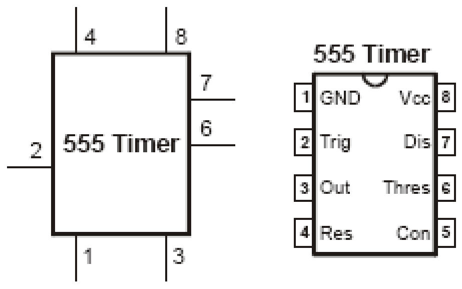


Figure 10: 555 timer diagram

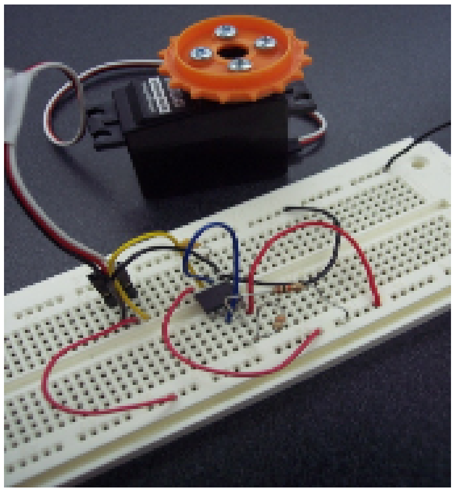


Figure 11: Actual Circuit for one motor

The circuits in Figures 8 and 9 look nearly identical. However, in order for the motors to turn in opposite directions so that the robot can move forward or backward, the 2 resistors connected to Pins 2, 6, and 7 were changed. Each motor was previously calibrated using PBasic software by turning the set potentiometers in each motor to a certain pulse width so that the motors would stop, which was calculated using the PBasic code as follows.

```
DO
PULSOUT 13, 750
PAUSE 20
LOOP
```

Pulse width: $750 \times 2\mu s = 0.0015$ seconds

The pulse width sent to one motor must be above 0.0015 seconds, and the other motor must receive a pulse width lower than 0.0015 seconds so that both motors can move forward or backward when the circuit input reaches a high. The pulse width was calculated for each circuit using the values of the resistors and the capacitor, as shown below.

General Pulse Width Equation:

$$T_P = 0.69 \times C \times (R_A + R_B)$$

For Right Motor:

$$T_P = 0.69 \times 0.1\mu F \times (20k\Omega) = 0.00138\ seconds$$

For Left Motor:

$$T_P = 0.69 \times 0.1\mu F \times (110470k\Omega) = 0.00693\ seconds$$

The vales of the resistors were determined through trial and error by using a potentiometer for each circuit and adjusting it for its full range to determine the best fit resistor for each motor to exert maximum power. High value resistors were tested (≥10KΩ), as well as small capacitances (≤10μF).

In order to measure distance and elevation, the locator tower was used. The structure and design of the locator tower is such that it moves with angular displacement along the vertical and horizontal planes so that angles can be obtained in those respective directions. Using those measured angles, distances and elevations can be obtained in the x, y, and z directions. A transmission wire, made using stripped phone cable, is connected to a retractable coil-and-spool structure placed on the robot itself. When the wire is stretched out, it will be under constant tension, which is needed for accurate measurements. The change in resistance can dictate the change in position. In case of possible extra resistance from the wire due to its length, the resistance of each wire (which was determined to be a copper wire) was calculated for precautionary reasons. The resistivity of the copper wire is $1.67 \times 10^{-6}\Omega \cdot cm$, the length of each wire is 215 cm, and the diameter was found to be 0.0635 cm.

$$R = \frac{\rho L}{A} = \frac{(1.67 \times 10^{-6}\Omega \cdot cm)(215\,cm)}{0.0635\,cm^2} = 0.113\Omega$$

As can be seen above, the resistivity in the wire is practically negligible compared to the resistances that are used in the circuit, as will be shown later.

This type of circuitry required the use of A/D converters in order to transfer the analog data (distance traveled) into digital data through the use of the locator tower. A PBasic code was needed to convert from spherical (angles) to Cartesian coordinate measurements. Therefore, an ADC chip and corresponding circuit was used to measure the angles in the horizontal and vertical direction, as shown in Figure 12 below.

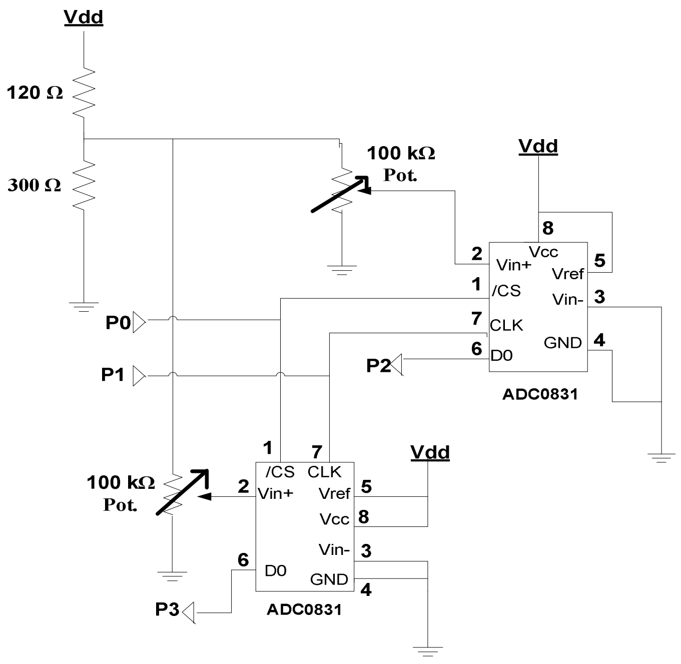


Figure 12: ADC Circuit for locator tower

The 2 ADC's are connected through Pins 1 and 7, which are the chip select and the clock pins. The potentiometers are each connected to Pin 2 of each ADC. For the circuit, the voltage division formula was used to determine which resistors to use to send the proper voltage to each of the 100 kΩ potentiometers. However, this method was not successful, since the values that appeared on the DEBUG screen were not the correct angle values for the spool positions of the locator tower. Therefore, the appropriate resistor values were determined empirically to calibrate the locator tower spools. A similar circuit was constructed for the coil-and-spool structure, using a 10-turn potentiometer of the same resistance.

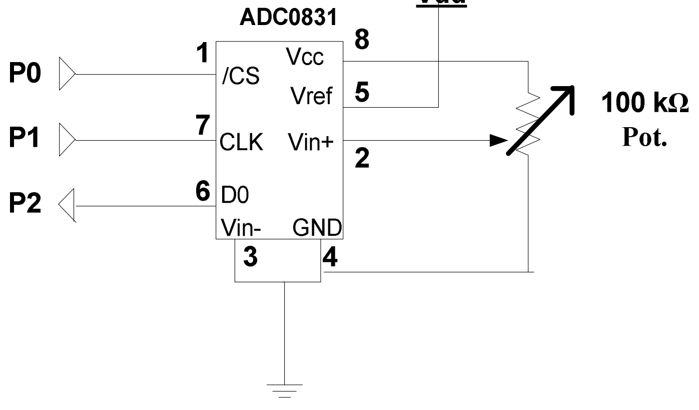


Figure 13: ADC Circuit for coil-and-spool structure

The PBasic software records the angles measured in Binary radians (Brads). Angles on a circle are not represented as a full 360 degree circle on PBasic. They are represented in Brads with an 8-bit range from 0 to 255 as shown below. The distance from the center of the circle is measured to be 127 units. This information was useful in constructing an appropriate conversion program for angles measured.

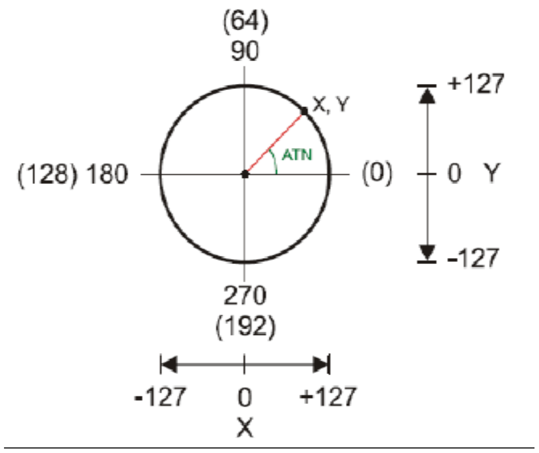
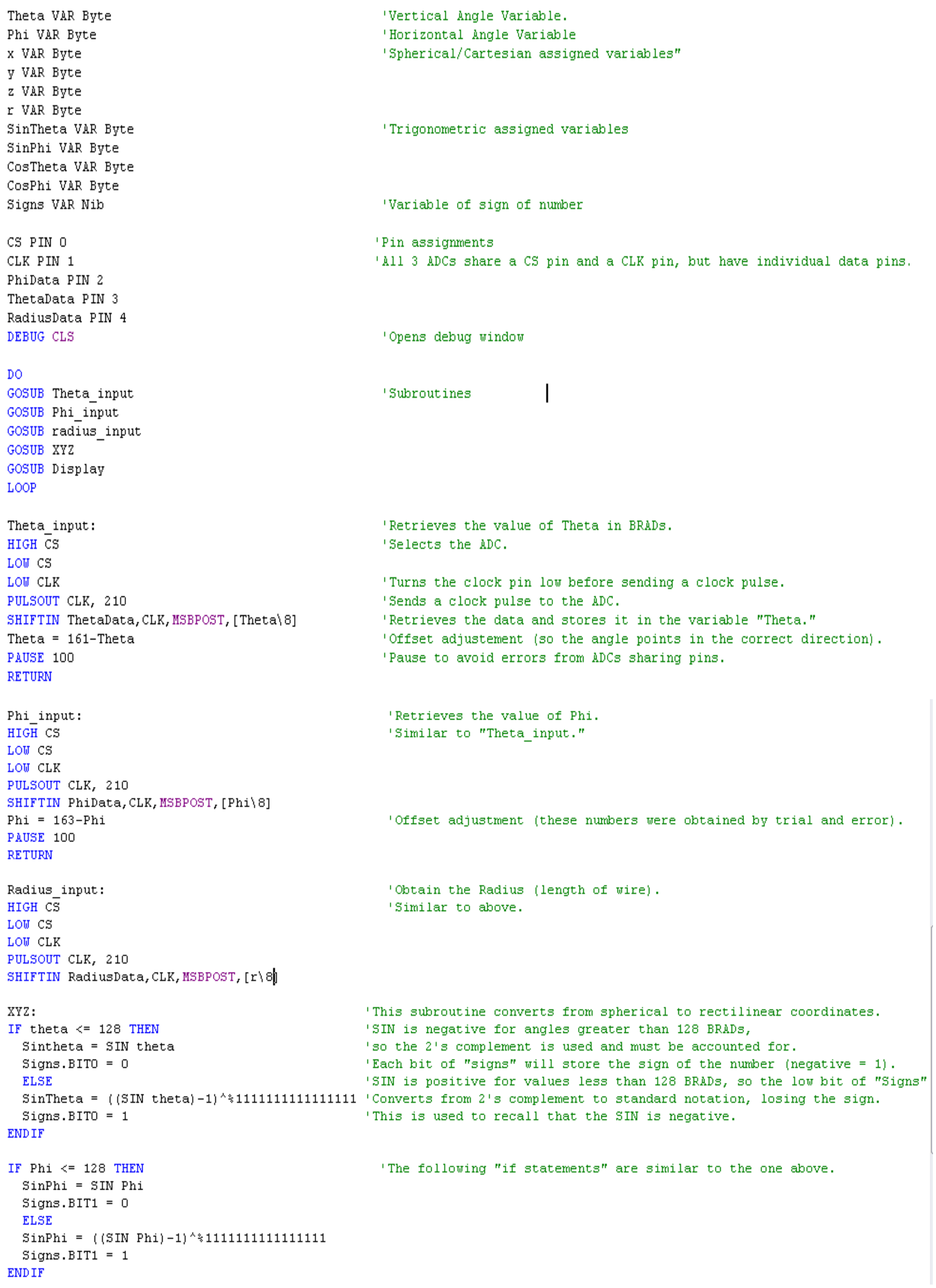
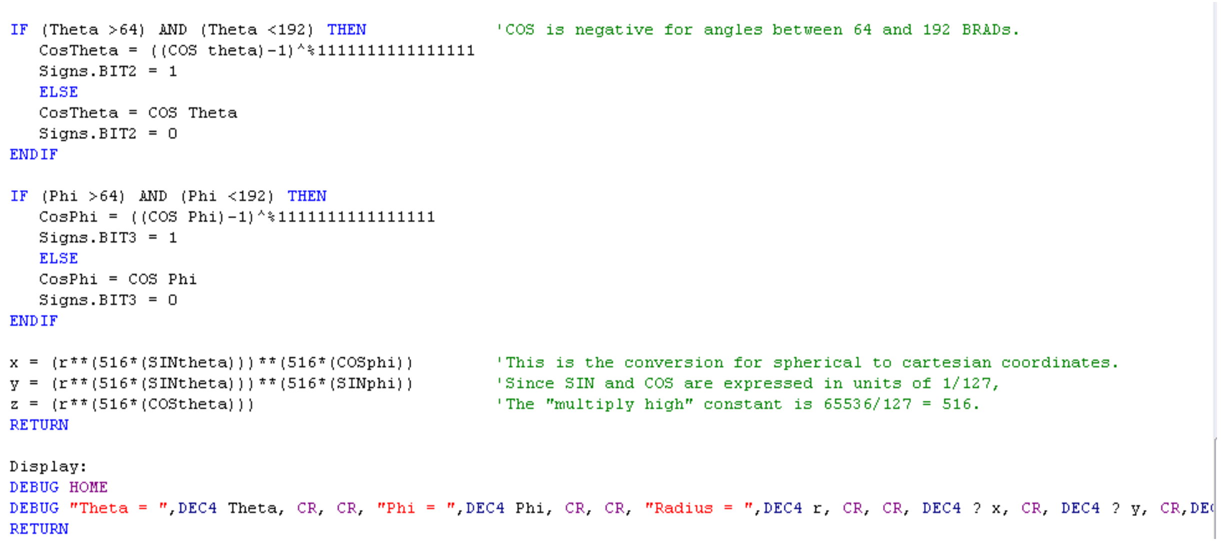


Figure 14: Circle with angles in Degrees and Bit Radians (*Courtesy of ME5643: Mechatronics Lectures by Prof. Vikram Kapila, PhD.*

Taking Figure 14 into consideration, the commands SIN and COS in PBasic were used to convert the angles measured into lengths and elevations. First, the values obtained by the SIN and COS operators were multiplied by 1/127 (see Figure 14 above) to convert into distance units that were manageable for calculation in PBasic. The values were then multiplied by 65536 in order to obtain length and elevation values. The PBasic code is shown below. The vertical angle is referred to as “theta”, and the horizontal angle will be referred to as “phi”.



```
Theta VAR Byte                                  'Vertical Angle Variable.
Phi VAR Byte                                    'Horizontal Angle Variable
x VAR Byte                                      'Spherical/Cartesian assigned variables"
y VAR Byte
z VAR Byte
r VAR Byte
SinTheta VAR Byte                               'Trigonometric assigned variables
SinPhi VAR Byte
CosTheta VAR Byte
CosPhi VAR Byte
Signs VAR Nib                                   'Variable of sign of number

CS PIN 0                                       'Pin assignments
CLK PIN 1                                      'All 3 ADCs share a CS pin and a CLK pin, but have individual data pins.
PhiData PIN 2
ThetaData PIN 3
RadiusData PIN 4
DEBUG CLS                                       'Opens debug window

DO
GOSUB Theta_input                               'Subroutines
GOSUB Phi_input
GOSUB radius_input
GOSUB XYZ
GOSUB Display
LOOP

Theta_input:                                    'Retrieves the value of Theta in BRADs.
HIGH CS                                         'Selects the ADC.
LOW CS
LOW CLK                                         'Turns the clock pin low before sending a clock pulse.
PULSOUT CLK, 210                                'Sends a clock pulse to the ADC.
SHIFTIN ThetaData,CLK,MSBPOST,[Theta\8]         'Retrieves the data and stores it in the variable "Theta."
Theta = 161-Theta                               'Offset adjustement (so the angle points in the correct direction).
PAUSE 100                                       'Pause to avoid errors from ADCs sharing pins.
RETURN

Phi_input:                                       'Retrieves the value of Phi.
HIGH CS                                          'Similar to "Theta_input."
LOW CS
LOW CLK
PULSOUT CLK, 210
SHIFTIN PhiData,CLK,MSBPOST,[Phi\8]
Phi = 163-Phi                                    'Offset adjustment (these numbers were obtained by trial and error).
PAUSE 100
RETURN

Radius_input:                                    'Obtain the Radius (length of wire).
HIGH CS                                          'Similar to above.
LOW CS
LOW CLK
PULSOUT CLK, 210
SHIFTIN RadiusData,CLK,MSBPOST,[r\8]

XYZ:                                          'This subroutine converts from spherical to rectilinear coordinates.
IF theta <= 128 THEN                          'SIN is negative for angles greater than 128 BRADs,
  Sintheta = SIN theta                        'so the 2's complement is used and must be accounted for.
  Signs.BIT0 = 0                              'Each bit of "signs" will store the sign of the number (negative = 1).
  ELSE                                        'SIN is positive for values less than 128 BRADs, so the low bit of "Signs"
  SinTheta = ((SIN theta)-1)^%1111111111111111 'Converts from 2's complement to standard notation, losing the sign.
  Signs.BIT0 = 1                              'This is used to recall that the SIN is negative.
ENDIF

IF Phi <= 128 THEN                              'The following "if statements" are similar to the one above.
  SinPhi = SIN Phi
  Signs.BIT1 = 0
  ELSE
  SinPhi = ((SIN Phi)-1)^%1111111111111111
  Signs.BIT1 = 1
ENDIF
```



```
IF (Theta >64) AND (Theta <192) THEN           'COS is negative for angles between 64 and 192 BRADs.
   CosTheta = ((COS theta)-1)^%1111111111111111
   Signs.BIT2 = 1
   ELSE
   CosTheta = COS Theta
   Signs.BIT2 = 0
ENDIF

IF (Phi >64) AND (Phi <192) THEN
   CosPhi = ((COS Phi)-1)^%1111111111111111
   Signs.BIT3 = 1
   ELSE
   CosPhi = COS Phi
   Signs.BIT3 = 0
ENDIF

x = (r**(516*(SINtheta)))**(516*(COSphi))      'This is the conversion for spherical to cartesian coordinates.
y = (r**(516*(SINtheta)))**(516*(SINphi))      'Since SIN and COS are expressed in units of 1/127,
z = (r**(516*(COStheta)))                      'The "multiply high" constant is 65536/127 = 516.
RETURN

Display:
DEBUG HOME
DEBUG "Theta = ",DEC4 Theta, CR, CR, "Phi = ",DEC4 Phi, CR, CR, "Radius = ",DEC4 r, CR, CR, DEC4 ? x, CR, DEC4 ? y, CR,DE
RETURN
```

Figure 15: PBasic Code for Locator Tower/Coil-and-Spool Circuitry

Data was to be collected as the robot runs around the site. Using the mechanisms mentioned above, the robot detects its position and elevation on the terrain on all 3 axes. However, the difficulty in recording this data using the BS2 is that its processing is limited. It is only able to execute one command at a time or compute data from one pin at a time. The software that interfaces with it is also not able to directly convert data files into the form that is needed to create a contour map. Therefore, an EEPROM chip was used to store all of the measurement data that is recorded and needed to create a contour map. The essential specifications for this chip are as follows. Also, for this task, powerful math and graphing software was the best choice, such as Matlab®, created by The MathWorks, Inc [5]. In this case, the BS2 and PBasic Editor Software must be interfaced with Microsoft Visual Basic Express 2008®[6] in order to create a debug screen that is capable of having the data be converted into an M-file, which is the file extension used in Matlab. Once the data is placed into Matlab, a detailed contour map of the construction site can be created.

The specifications of the EEPROM chip are as follows:

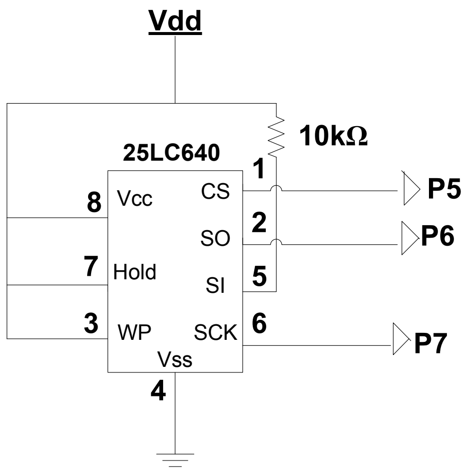
EEPROM (25LC640)

1.) 8,192 data registers

2.) Each register is capable of storing 1 byte of data

3.) Electrostatic Discharge Protection: 4000 Volts

The circuit and pin labels are specified in Figure 13 below.



Vcc = Power Voltage

Hold = allows for program flow control (if program is shut down, it starts where left off)

WP = Write Protection

CS = Chip Select

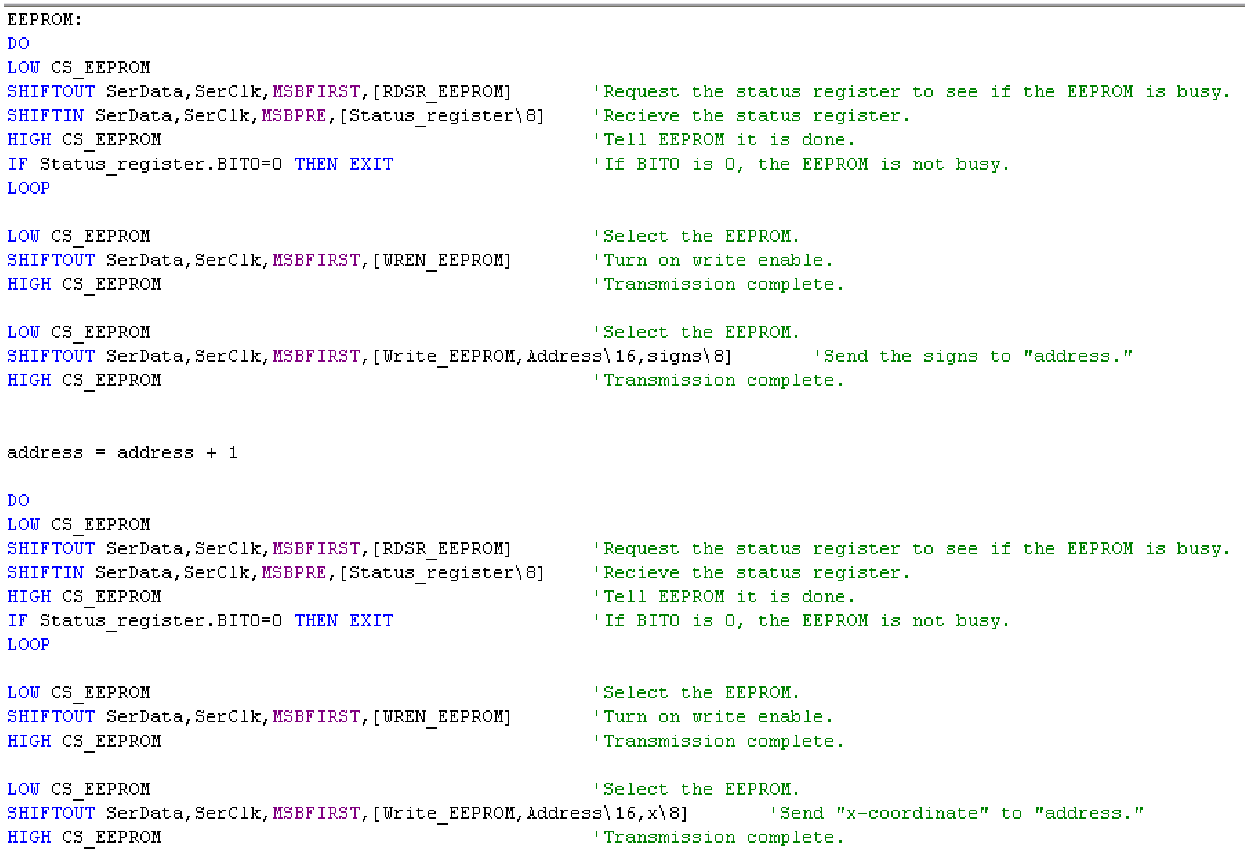
S0 = Serial Output

SI = Serial Input

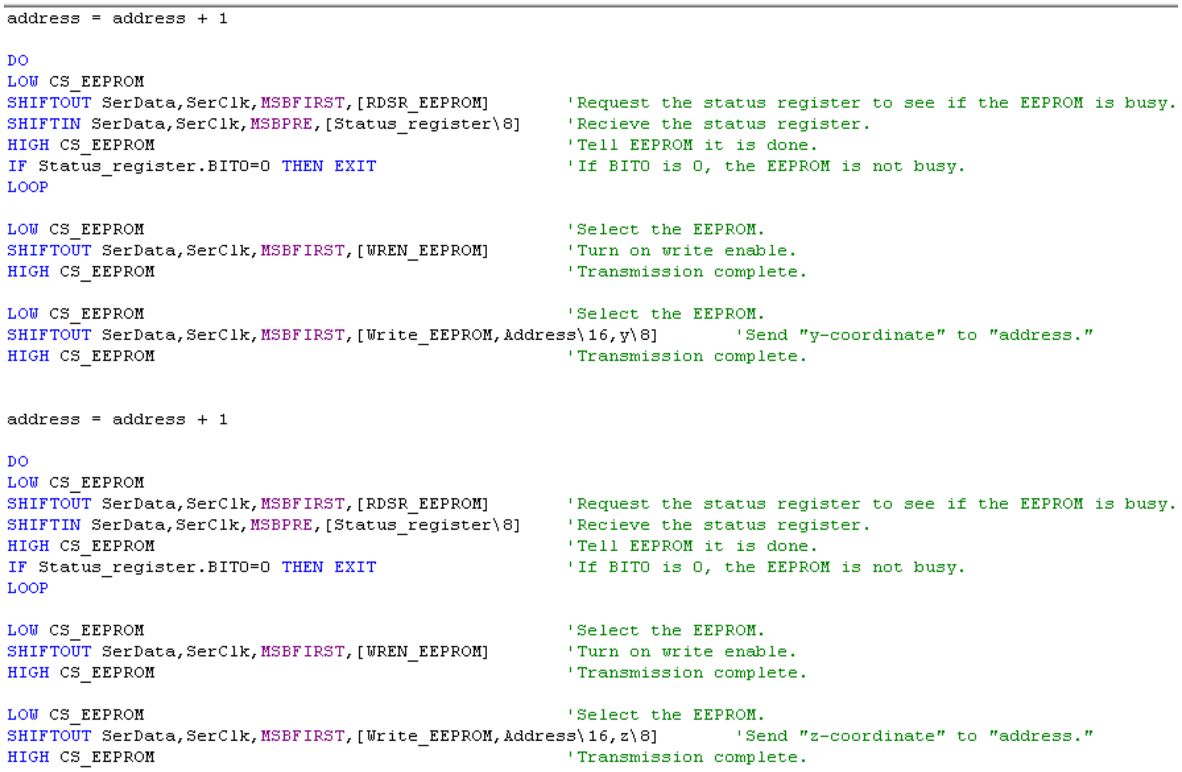
SCK = Clock

Figure 16: EEPROM circuit

For chip functionality, pin P0 was set at a low voltage. A 10kΩ resistor was recommended so that the serial output pin can be brought to a zero signal. PBasic code was written for the EEPROM circuit so that it could store the required distance and elevation data needed for the contour map.



```
EEPROM:
DO
LOW CS_EEPROM
SHIFTOUT SerData,SerClk,MSBFIRST,[RDSR_EEPROM]       'Request the status register to see if the EEPROM is busy.
SHIFTIN SerData,SerClk,MSBPRE,[Status_register\8]    'Recieve the status register.
HIGH CS_EEPROM                                       'Tell EEPROM it is done.
IF Status_register.BIT0=0 THEN EXIT                  'If BIT0 is 0, the EEPROM is not busy.
LOOP

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[WREN_EEPROM]       'Turn on write enable.
HIGH CS_EEPROM                                       'Transmission complete.

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[Write_EEPROM,Address\16,signs\8]       'Send the signs to "address."
HIGH CS_EEPROM                                       'Transmission complete.


address = address + 1

DO
LOW CS_EEPROM
SHIFTOUT SerData,SerClk,MSBFIRST,[RDSR_EEPROM]       'Request the status register to see if the EEPROM is busy.
SHIFTIN SerData,SerClk,MSBPRE,[Status_register\8]    'Recieve the status register.
HIGH CS_EEPROM                                       'Tell EEPROM it is done.
IF Status_register.BIT0=0 THEN EXIT                  'If BIT0 is 0, the EEPROM is not busy.
LOOP

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[WREN_EEPROM]       'Turn on write enable.
HIGH CS_EEPROM                                       'Transmission complete.

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[Write_EEPROM,Address\16,x\8]       'Send "x-coordinate" to "address."
HIGH CS_EEPROM                                       'Transmission complete.
```



```
address = address + 1

DO
LOW CS_EEPROM
SHIFTOUT SerData,SerClk,MSBFIRST,[RDSR_EEPROM]       'Request the status register to see if the EEPROM is busy.
SHIFTIN SerData,SerClk,MSBPRE,[Status_register\8]    'Recieve the status register.
HIGH CS_EEPROM                                       'Tell EEPROM it is done.
IF Status_register.BIT0=0 THEN EXIT                  'If BIT0 is 0, the EEPROM is not busy.
LOOP

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[WREN_EEPROM]       'Turn on write enable.
HIGH CS_EEPROM                                       'Transmission complete.

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[Write_EEPROM,Address\16,y\8]       'Send "y-coordinate" to "address."
HIGH CS_EEPROM                                       'Transmission complete.


address = address + 1

DO
LOW CS_EEPROM
SHIFTOUT SerData,SerClk,MSBFIRST,[RDSR_EEPROM]       'Request the status register to see if the EEPROM is busy.
SHIFTIN SerData,SerClk,MSBPRE,[Status_register\8]    'Recieve the status register.
HIGH CS_EEPROM                                       'Tell EEPROM it is done.
IF Status_register.BIT0=0 THEN EXIT                  'If BIT0 is 0, the EEPROM is not busy.
LOOP

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[WREN_EEPROM]       'Turn on write enable.
HIGH CS_EEPROM                                       'Transmission complete.

LOW CS_EEPROM                                        'Select the EEPROM.
SHIFTOUT SerData,SerClk,MSBFIRST,[Write_EEPROM,Address\16,z\8]       'Send "z-coordinate" to "address."
HIGH CS_EEPROM                                       'Transmission complete.
```

```
address = address + 1
RETURN
```

Figure 17: PBasic Code For EEPROM chip

Once the codes from Figures 15 and 17 were placed together, the final piece of code is the logic control of the motors as the robot travels across the construction site. The robot is to travel around the site, going from end to end with a path similar to Figure 18 below.

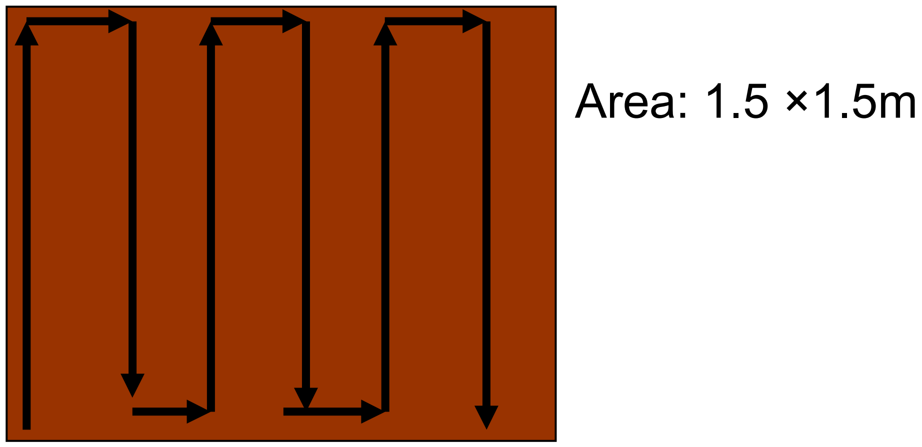


Figure 18: Robot path along designated area

However, due to time constraints and program transition difficulties, the robot was unable to be tested for area movement, even though motor control programming would have been simple to combine with the programming shown above.

Early Problems/Solutions

Some early-design problems were apparent during the construction of the robot, as well as the programming and circuitry. Certain possible sources of error had to be accounted for and minimized in the mechanical and electrical design.

A mechanical problem that reoccurred was the use of the coil-and-spool structure. As mentioned before, the wire connected from the locator tower to the coil-and-spool structure needed to be under constant tension in order to measure distances and elevations accurately. However, too much tension in the wire may hinder the movement of the robot, which was the reoccurring problem. The solutions to this problem were to first use lubricant in each and every moving joint in the coil-and-spool structure. Once the lubricant was added, the coil inside of the structure needed to be replaced with a smaller coil to produce less tension on the robot. Different tape measure coils were tested, but the current coil used in the design is the coil from 16 ft measuring tape, which turned out to be strong enough to keep tension on the wire and not hinder the movement of the robot along the terrain.

Another possible mechanical problem that arose was the use of the DC motors. As seen in Figures 3 and 4, the circuitry was designed to supply maximum power to the motors. However, the chassis of the robot itself, as well as the structures placed on it were considerably heavy. The robot itself was not expected to have high speed, but it did need as much power as possible to traverse through rough terrain and progress with the tension applied to one of its built-in structures. After testing, it was proven that the power supplied to the motors was adequate enough to move the robot and handle the tension of the transmission wire.

One source of error that may have been incurred by the programming was the use of the Multiply-High operator. This was used in when implementing the SIN and COS function when converting from spherical to Cartesian coordinates. Since PBasic cannot multiply decimal numbers directly, the Multiply-High operator was used. However, PBasic tends to round off any remaining decimals to the nearest whole number once this operator is used, so some of the distance and elevation readings may be off by tenths of a unit measurement.

A possible uncontrollable source of error may be due to wind. If the robot is used in a very windy area, the wind may pull on the wire, which can cause slightly inaccurate measurements. However, since the wire is reasonably heavy, and since there is sufficient tension in the wire, the wind may not have much of an effect on the wire.

Prototype Cost Analysis

Using the list of materials written in the "Bill of Materials" section of this report, a prototype cost analysis can be made using Microsoft Excel®. As can be seen in the tables below, the materials and labor costs are accounted for, assuming 120 hours (8 hours per day for 15 days) of work for the prototype.

COST OF MATERIALS			
Materials		Qty.	Cost ($)
Parallax Circuitry Materials			
	Basic Stamp 2 Microcontroller Chip	1	50
	Board of Education	1	70
	Basic Analog and Digital Parts Kit	1	30
Parallax Tank Tread Kit			35
Sheet Metal Chassis			5
Tower structure/Coil and Spool Chassis Manufacturing			5
Miscellaneous	1 Breadboard	1	5
	2 A/D converters	2	6
	Extra Jumper Wires/Leads	6	2
	Telephone wire	1	6
	2 single turn potentiometer	2	9
	1 10-turn potentiometer	1	13
	Glue Gun	1	10
	EEPROM chip	1	1
	Rubber Bands (large)	4	1
TOTAL MATERIALS COST			248

LABOR			
Time of assembly/programming (hrs)			120
Hourly pay rate ($)			20
Amount of team members			3
TOTAL LABOR COST			7200
TOTAL PROTOTYPE COST			7448

Figure 16: Prototype Cost Table

The total cost, as seen above, is $7448 for each prototype built. Using that cost, an estimate for mass production of the Worksite Leveler Robot can be obtained, as shown below. Since it takes 15 days (8 hours each day) to build each prototype, it will take 1 month to build 2 prototypes. Assumed fixed costs are taken into account in the table below.

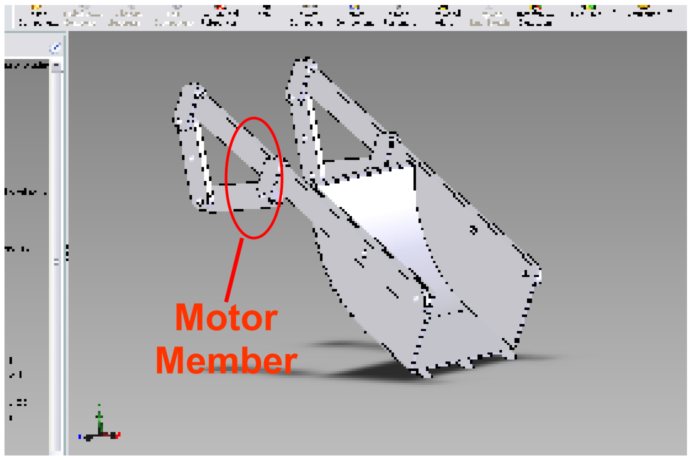
MASS PRODUCTION			
Fixed Factory Cost (Maintenance/Operation) per month			2000
Sheet Metalworking in shop per month			3000
Prototypes made in one month (15 days per prototype)			2
Overhead for each Prototype			2500
PROTOTYPE COST PER UNIT			9948
COST OF TOTAL PRODUCTION PER MONTH			19896

Figure 17: Prototype Cost Table

With a fixed factory cost of $2000 per month as well as the availability of sheet metalworking for $3000, an overhead cost of $2500 will be applied to each prototype. The total prototype cost for each month will be $19,896. If more are made each month, the overhead cost will decrease for each unit.

Conclusions/Improvements/Suggestions

If time had permitted further work on the project, there would have been more tasks that the robot could have been designed and programmed to do. On worksites, if autonomy is focused, there may be several robots, each performing one task each. If each robot were to multitask by performing 2 or more tasks, the amount of robots (as well as possible expenses) can be cut to at least half. In this case, one of the other functions that the worksite leveler robot may have performed was to actually physically level the ground by digging and moving dirt around the site using a shovel attached to the chassis. The design for the shovel component of the robot, as well as Solidworks design drawings are shown below.



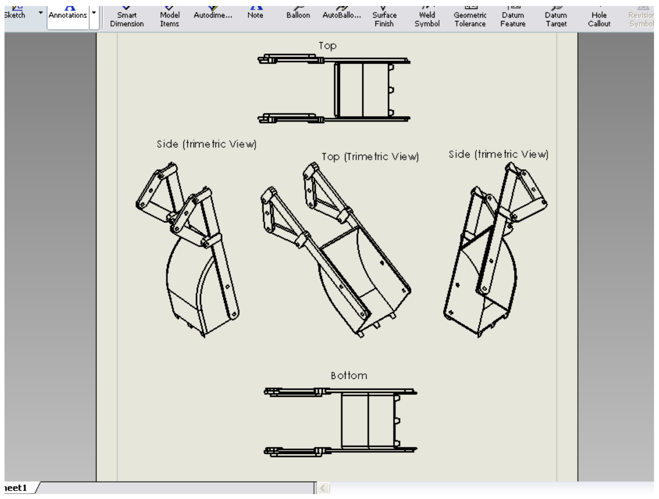


Figure 18: Initial Shovel Design and various orientation drawings using Solidworks® software

The initial shovel structure design would have been constructed as a pair of walking-beam linkages attached to each side of a shovel. One side would be attached to the motor and rotate so that the shovel itself would move up and down in a "digging" motion. It was believed that this would be an efficient design, especially since it required only one DC servomotor to operate.

From a programming standpoint, it was believed that it would be easier to position the shovel inside of the robot chassis instead of typically placing it in the front ot rear of the robot. To move dirt from a certain area with a high elevation designated by the robot's sensory/elevation system, the robot simply would have to place itself over the area and dig in.

Overall, the prototype was able to traverse while measuring the necessary angles in the Cartesian plane, but the robot was not able to be tested for the ability to create a contour map. Due to time constraints and software version difficulty of Visual Basic, the transition of the data from the EEPROM chip to Visual Basic was not possible, so the contour map could not be made. Using this technology, construction companies will be able to tell which areas of the construction site will need to be worked on or be cautious of. This will not only provide ease of site preparation, but it can also inspire robotic autonomous multitasking that can financially benefit companies, as well as lower the risk of injuries and fatalities in a construction site.

References

[1] Online: www.parallax.com, website of Parallax Inc, distributor of Basic Stamp 2 microcontroller.

[2] Online: http://www.irobot.com/, website of iRobot Corporation.

[3] Online Robotic Resource and Links of Autonomous Concrete Construction Robots: http://www.robotmatrix.org/RobotResourcePage.asp?urlBody=http://www.takenaka.co.jp/takenaka_e/techno/53_crobo/53_crobo.htm, Construction Robots from the Architectural Institute of Japan.

[4] Online: http://www.solidworks.com/, website of Solidworks Corp., developer and distributor of Solidworks software.

[5] Online: http://www.mathworks.com/, website of MathWorks Corp. Developer and distributor of Matlab software.

[6] Online: http://msdn.microsoft.com/en-us/vbasic/default.aspx, Microsoft Visual Basic Developer Center

[7] Basic Stamp User Site: Stamp Projects: "Interfacing a 2 wire EEPROM to the Stamp: http://www.geocities.com/SiliconValley/Orchard/6633/projects.html