SNOW SHOVELING ROBOT

Report and Documentation

Date Submitted: 12-19-06
ME3483 Mechatronics

Group Members	Emails
Ariel Avezbadalov	(arielavezbadalov@yahoo.com)
Roy Pastor	(rpasto02@gmail.com)
Samir Mohammed	(Samirsmohammed@yahoo.com)
Travis Francis	(dogfighter22@hotmail.com)

Table of Contents

1. Theory

The general purpose of “The Snow Shoveling Robot” is to shovel and remove snow from the sidewalk with the utmost efficiency and minimum human intervention and labor. The removable of snow from both the roads and the sidewalk is a very hazardous operation. The manual removal of snow involves hours of exposure to extremely harsh weather and surface conditions; and extreme strain and stress on an individual’s body. It has been proven that these extreme cold and windy conditions coupled with the stress and strain (on the human body) produce by the lifting and pushing of a shovel, can lead to an increase in blood pressure and heart related illnesses, back pain and muscle strain and it can also promote blood clotting. Some studies have also shown that as many as 1200 heart related deaths occur yearly during and after major snowstorms. By reducing the amount of time human beings are exposed to those unforgiving freezing conditions, the number deaths and illnesses caused by snowstorms will be reduced.

The Snow Shoveling Robot will receive the length of the pathway it has to shovel from the user (through a couple of buttons); the user will place it at the end of the pathway; and then give it the begin signal. The robot will contain a “C” shape shovel in its front. The shovel will be angle by about forty degrees to the right, which will allow the robot to push the snow off the sidewalk into the streets and/ someone’s lawn. The robot will be mounted on treads (instead of wheels) because over the last century, treads have proven to be much more reliable on “grainy” surfaces that flow. The robot will feature a salt dispenser at its rear that will release rock salt along the freshly shoveled path in order to provide shoe traction and preventing snow/ice buildup. One of the most important features of this robot is IR sensor mounted on its front that allows it to detect obstacles in its path.

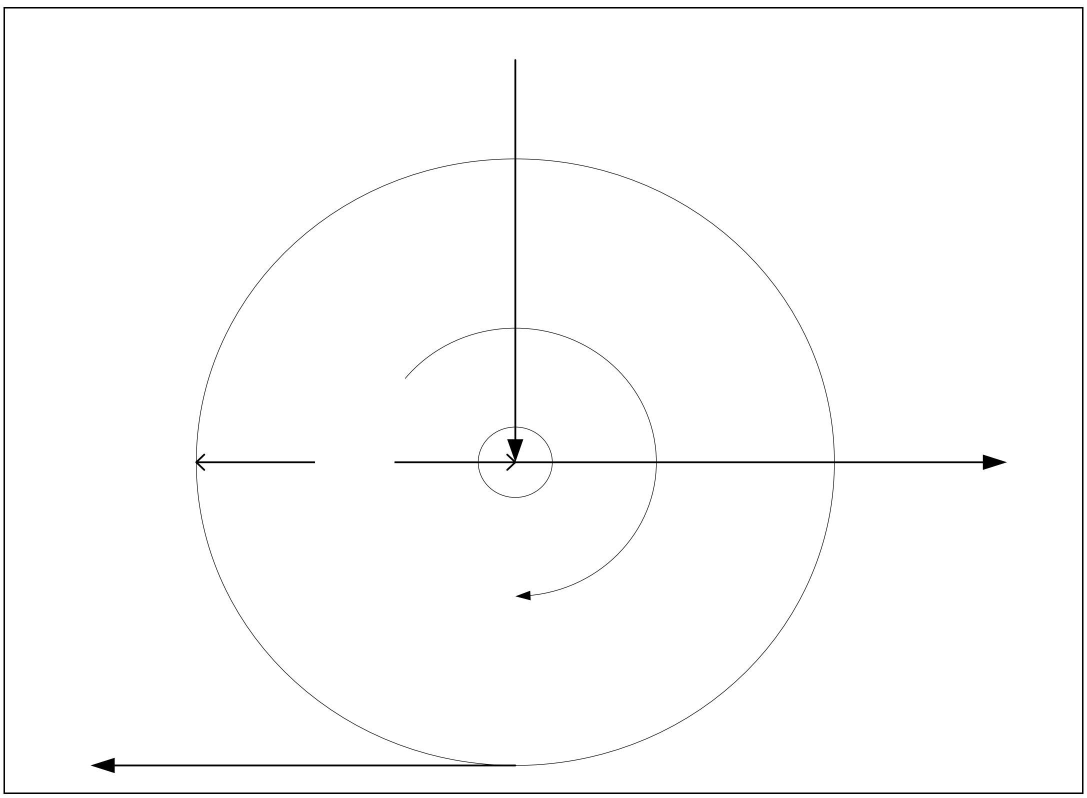
2. Mathematical Background

The A2D converter used by the robot is an 8 bit converter, and its positive input voltage is between 0 to 2.5 volts, therefore its resolution is;

$$2^8 = 256$$

$$\frac{2.5V}{256} = 9.765625mV$$

The total weight of the robot (with a full load of salt) is about 3.5lbs and the total maximum torque produce by the two motors (in its current configuration) is 2.6in-lbs, therefore the maximum force the robot can exerts on the snow (with a full load of salt) is;



$W = 3.5lbs$

$$\tau = 13in \cdot lbs \cdot \frac{20}{100} = 2.6in \cdot lb$$

W

$$\mu = \frac{f}{r} = \frac{.015''}{1.5''} = 0.01$$

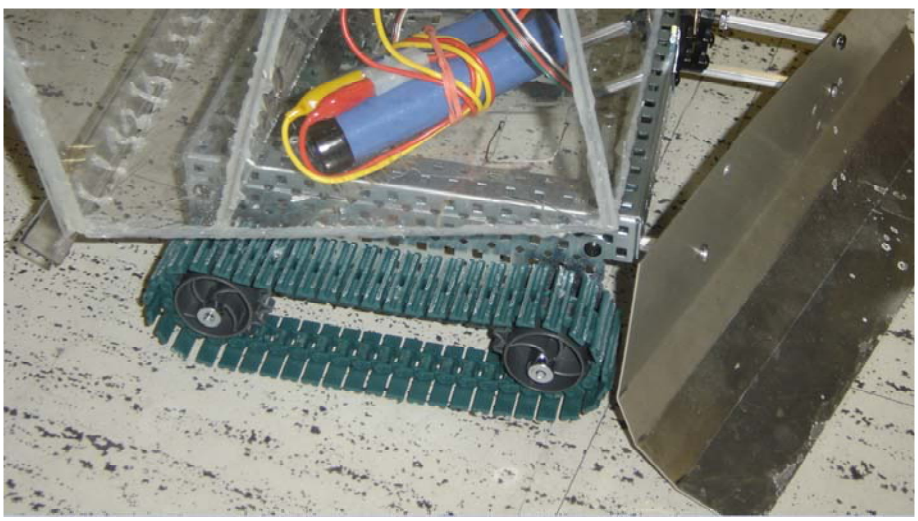
Since $\sum F_y = 0$

Then $R = \tau \times r - \mu W$

$$R = 3.9lbs - 0.035lbs = 3.865lbs$$

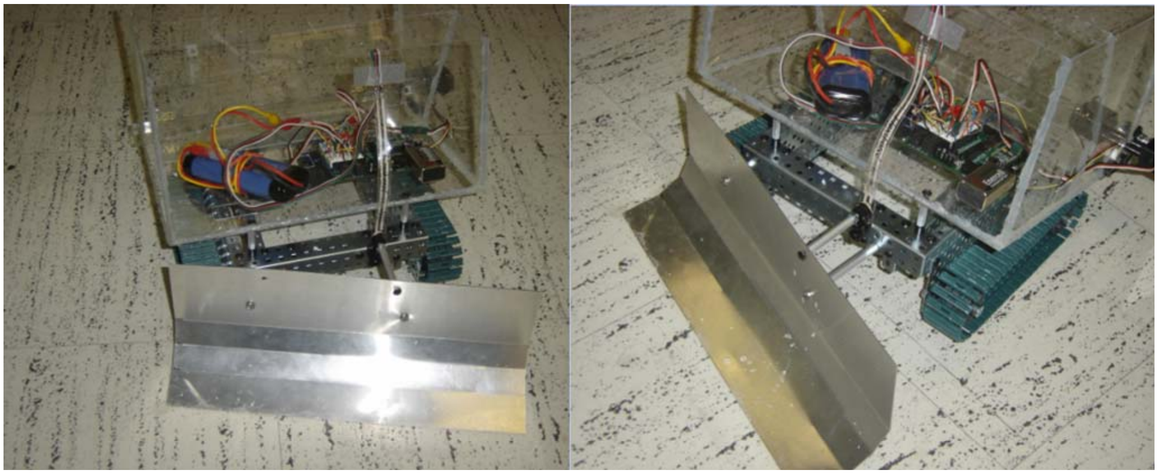
3. Mechanical Designs:

The mechanical design of the Snow Shoveling Robot must ensure the robot's capability of performing the main task of shoveling snow. This includes the robot's ability to withstand a certain force exerted by a specified thickness of compressed snow in addition to having the ability to smoothly penetrate through a snow-packed sidewalk using treads/wheels with high coefficients of friction. The design must be equipped with a structural piece of equipment capable of storing and proficiently dispersing salt as well. Other main tools on the robot include the use of a well constrained angled shovel made of a highly dense material in order to scrape and push aside the snow and the use of powerful servo motors to drive the device with a well exerted force. Digital photos of the robot along with their detailed descriptions are provided below:



Picture-01: Side view image of the Snow Shoveling Robot

As shown in the side view photo of the robot above, the choice of treads over the selection of wheels is quite obvious. Unlike ordinary wheels, treads provide the traction required especially in the circumstances the robot is expected to encounter. Since the robot is shoveling snow, it is expected to overcome a material that has a very low frictional factor. Unlike wheels, treads have a large contact area with the ground and maintain better terrain handling with smaller wheel height. As a result, they are less likely to slip as opposed to wheels. These features of the tread make it more suitable for the robot to use especially in conditions of snow, which involves a very moist and slippery environment. The treads used were treads made of plastic material.

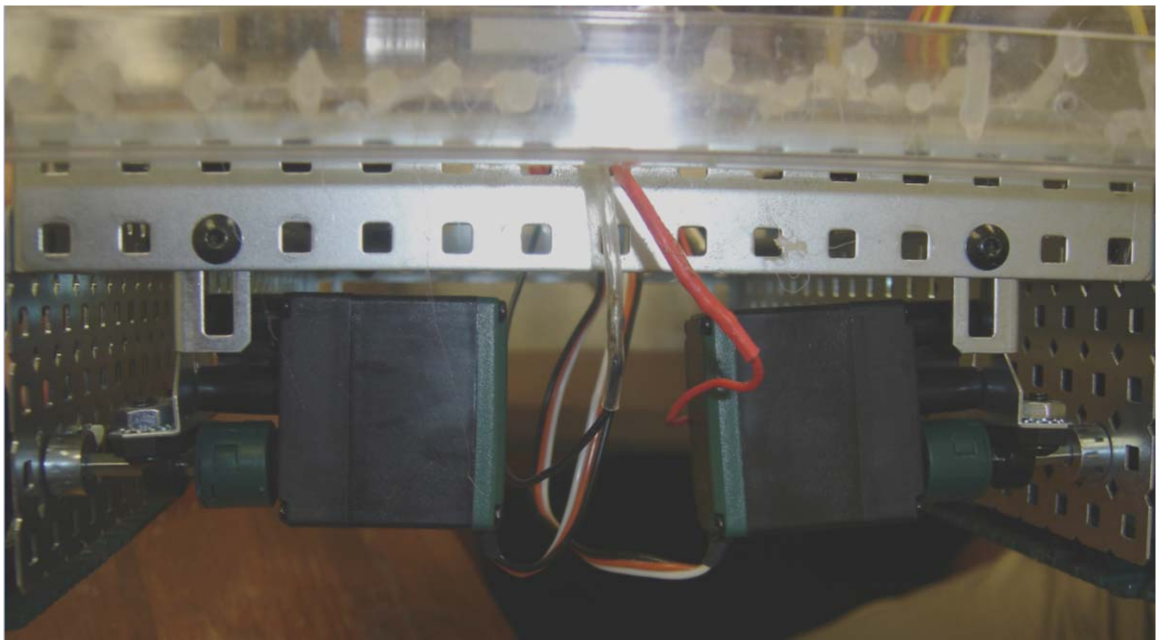


Picture-02&03: Front and Side view images of shovel

As shown above, the Snow Shoveling Robot comes equipped with a well constrained angled shovel. The shovel is made of sheet metal and is creased at the near center in order to provide enough strength and slant in abrading the snow that is in contact with the ground. The general angled position of the shovel enables the snow to be manually placed to the side of the sidewalk while providing free pavement and a snow-free walkway for the robot and people to pass through. The shovel is also wide enough to prevent any fatal disturbances with the running treads.

Picture-04&5: Salt Dispenser

Built on top of the base infrastructure of the Snow Shoveling Robot is the salt container. This container can store up to 2-3 pounds of salt. At the rear bottom of the container are open slots approximately .5 inches apart. The open slots along with a mechanical constrained servo controlled slider allow a specified rate of salt to disperse to the ground. This allows the container to empty at a constant pace rather than the salt falling out haphazardly and the user having to refill the container in short periods of time. The mechanically controlled slider is time controlled via the PBasic coding and functions as an opening and closing mechanism for the slots at a constant pace.



Picture-06: Unconstrained Servo Motors

The unconstrained servo motors play an extremely important role in the functioning of the robot. Unlike the servo that is attached to the slider and regulates the salt dispersal, the servo motors attached to the treads are unconstrained in the sense that they are not limited by a stopper. In other words, they are able to rotate a full 360 degrees. These servos provide enough speed and torque necessary for the robot to operate in snow conditions.

Important to note is the temperature conditions that the robot will encounter. There would be no purpose behind the device itself if the fundamental electrical and mechanical building blocks of the system were not fully functional under the extreme chilly conditions of the atmosphere that the robot is expected to work under. Fortunately the parts selected are able to withstand these extreme conditions and are fully functional up to temperatures [below freezing.] of -10 degrees Celsius.

The dimensions of the robot are shown below:

Tread to tread	11”
Treads Front-back	8.25”
Shovel Length	13.125”
Shovel Height	4.625”
Container	11.4375”x9.25”x6.25”
Total height (ground-top)	10.5”

4. Electronic Circuits

Obviously, the use of pure mechanical elements would not suffice the system to run appropriately. Integrated within the Snow Shoveling Robot are various electrical components designed for additional functionalities. The combinations of these mechanical and electrical subsystems help establish an integrated network system in which the communications of sensors are established. Playing the role as the “decision maker” is the BS2 Board of Education microcontroller from Parallax. The user interface includes two push buttons that will intake the desired distance of the user as well as initiate the start of the robot. All the sensors and motor circuits have been wired into the Basic Stamp Microcontroller. Schematics of the circuits wired along with their brief description are provided below:

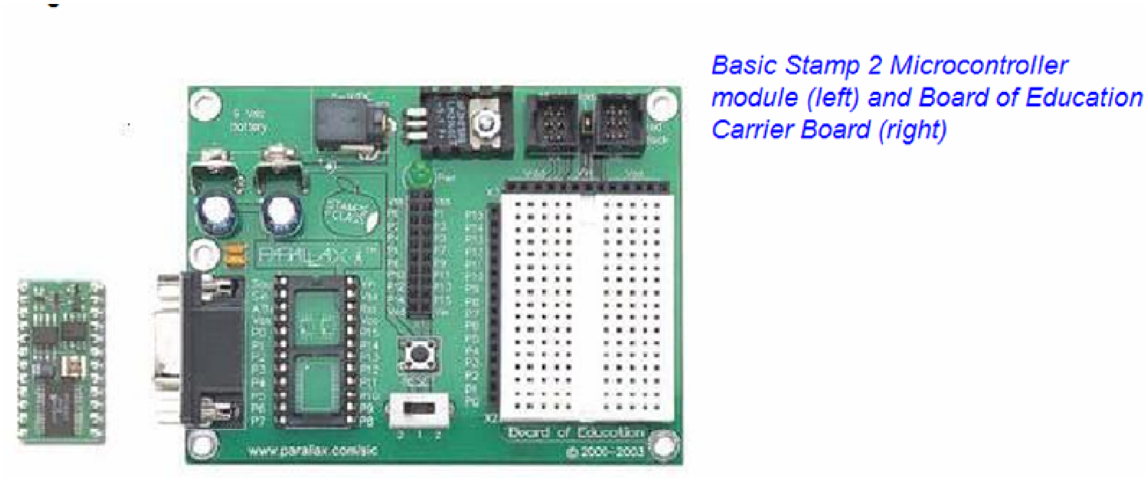
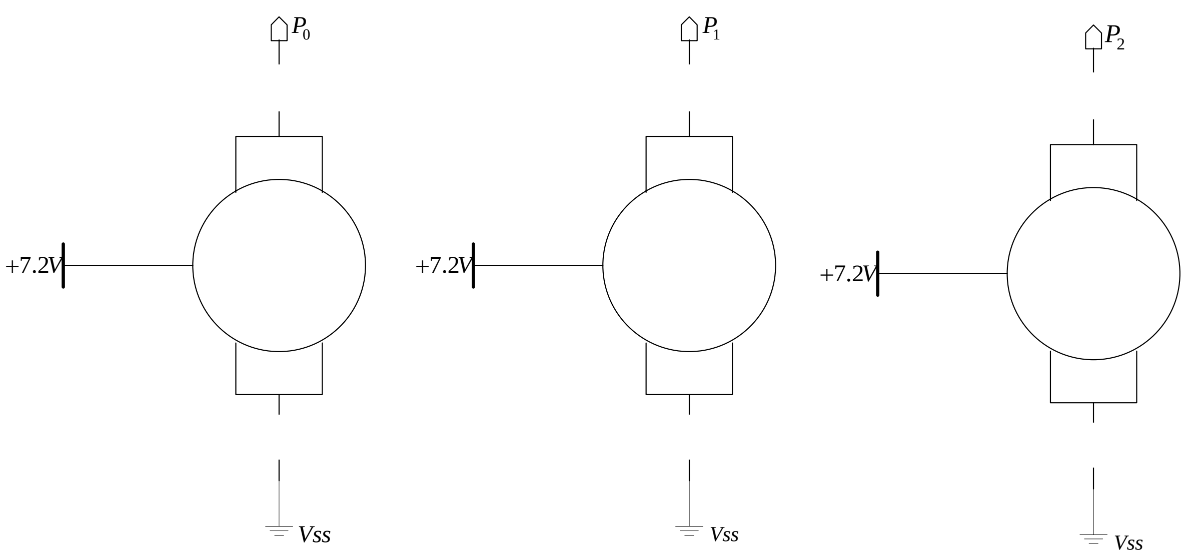


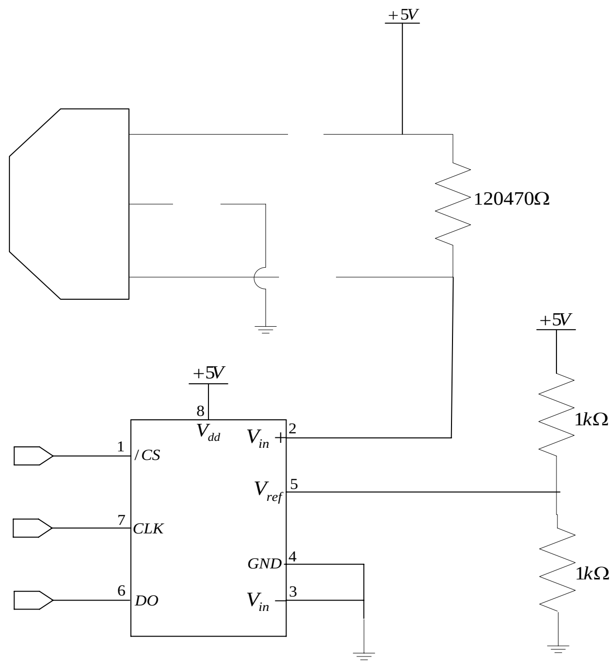
Photo-01: Basic Stamp 2 Microcontroller from Parallax

This Basic Stamp microcontroller and Board of Education Carrier Board, shown in the image above, serves the purpose of reading input devices from the sensors, processing the information, and controlling the output devices such as the motors and the piezo speaker. Some of the sensors the BS2 chip will be reading off include an infrared sensor as well as a photoresistor. The Basic Stamp microcontroller is essentially the “decision maker” of the robot since the code being executed is within the microcontroller’s EEPROM.



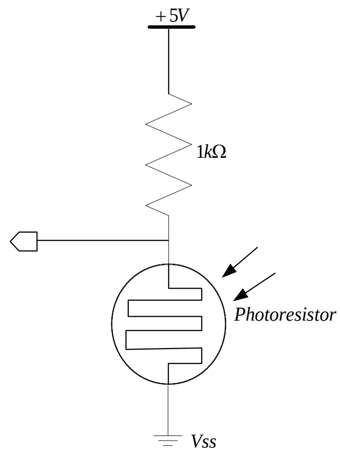
Schematic-01: Unconstrained Servos (left and right), constrained Servo (center)

Shown above are the circuit schematics of the servo motors. As mentioned previously, the unconstrained servos run the robot while the constrained servo is responsible for handling the rate of salt dispersion. The unconstrained servo motors are controlled via pins 0 and 2 while the constrained servo s controlled via pin 1. All the servos are powered up through an external 7.2 Volt Nickel Cadmium battery source. The motors that drive the robot are directly utilized without the use of any gears and run at approximately 1.2 feet per second without any load.



Schematic-02: Infrared circuited sensors

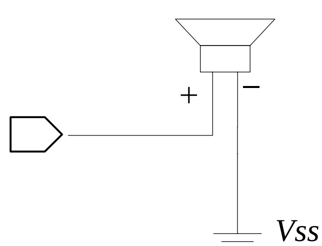
The schematics for the infrared sensors are shown above. The IR sensor located at the forefront of the robot is responsible for detecting people or obstacles that may interfere with the path of the moving robot. This sensor should be able to detect an obstacle or person from about 16 inches away. Once an object is detected, it will transfer this signal to the Basic Stamp which will then command the servo motors to stop running, signal the piezo speaker to send a beep, and flash an overhead LED to signify to the user that an object has been detected.



P5

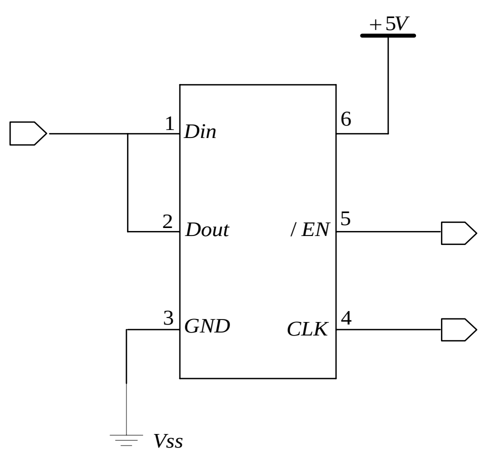
Schematic-03: Photoresistor circuit

A photoresistor is an electrical element that varies its resistance through varying light. In other words, the value of its resistance is a function of the amount of light present on it. The resistance verses temperature curve is a linearly decreasing slope which indicates that the value of the resistance decreases as the amount of light increases. This resistor is hooked up in the back of the robot under the salt dispenser. This sensor plays the role of determining whether the salt dispenser is carrying a sufficient amount of salt or whether the container needs refilling. Because the sensor is placed underneath the salt dispenser, the salt in the container blocks the light from falling onto the resistor. However, as the amount of salt decreases, the amount of light that penetrates and is exposed to the photoresistor increases. If a sufficient amount of light gets exposed to the photoresistor, a voltage signal is sent to the Basic Stamp which then signals the piezo speaker to go off and an LED to flash, indicating to the user that the container needs to be refilled.



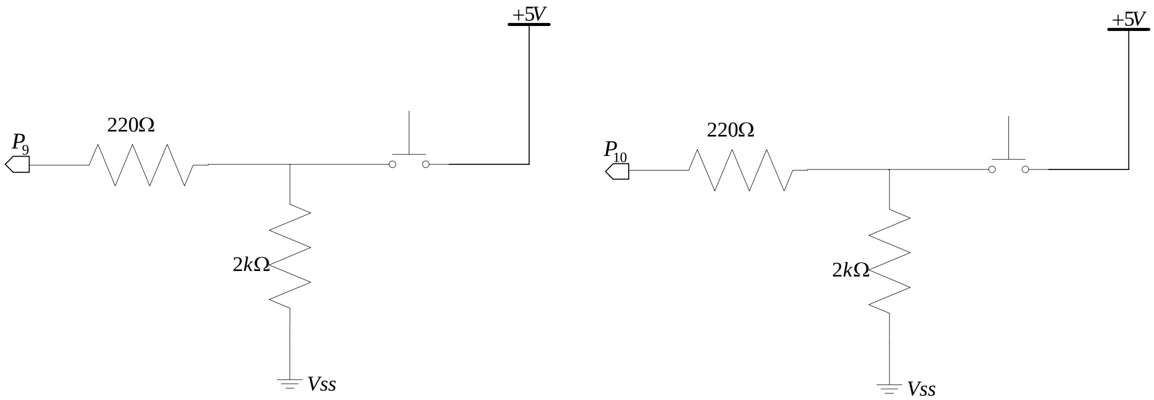
Schematic-04: Piezo Speaker

Shown above is the piezo speaker built into the system. A piezo speaker was selected in order to establish a communication mechanism with the user. Since the robot is autonomously designed for users to remain indoors while the robot functions outdoors, a signal needs to be interfaced with the system in order to indicate to the user the status or any needs of the robot. Programmed into the Snow Shoveling Robot are two output frequencies released from the speaker to communicate with the user. Each of the sounds emitted indicates one of two things to the user. In the case where the sound being emitted from the speaker is at a frequency of 5000 and 5050, the robot is indicating to the user that an object has been detected by the IR sensor and the robot has stopped. On the other hand, a sound emitted with a frequency of 4000 & 4005, indicates that the salt container is empty and needs to be refilled. Both sounds are very distinct and combine with the flashing of an LED to establish communication with the user.



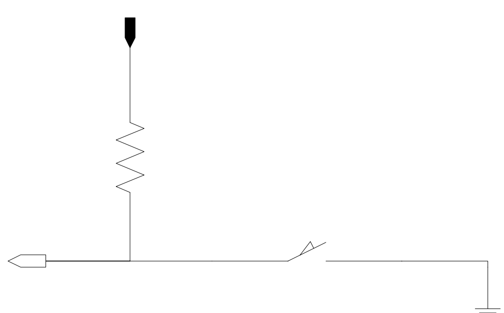
Schematic-05: Compass circuit

Assisting the robot on moving in a straight path is a compass. This sensor helps maintain the robot’s linear path since it keeps track of the direction and orientation of the robot. Knowing that the compass can determine the angle change of the device, the Snow Shoveling Robot can be programmed to maintain, for example, a ± 5 degrees of freedom. When the angle read by the compass surpasses this ± 5 degrees, the Basic Stamp will then take action depending on whether the value exceeded is too far left or too far right and will counteract the treads in order to stabilize the robot back to ± 5 degrees of the arbitrary straight direction.



Schematic-06: Button Circuits (User Interface)

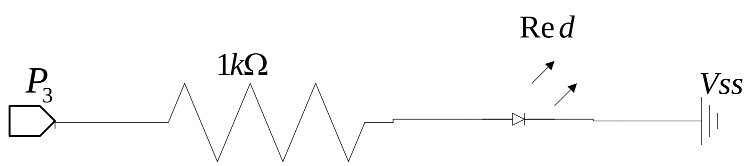
As mentioned previously, the user interface consists of two buttons. Each of the buttons play an important role in terms of their function. One of the buttons is responsible for handling the distance the user wants traveled while the other button is responsible for initiating the robot's movement. The initial button functions as a counter in the sense that the more times the button is pressed, the more the distance gets incremented. This distance gets incremented by 50 centimeters each time. So, for example, if the user wants the robot to travel 250 centimeters, the user would have to push the button five times. The second button would then be pressed to initiate the program and commence the robot's movement.



Schematic-07: Limit Switch

The limit switch is located near the right tread of the robot. With the help of two attached hex screws glued equidistant to one another on the plastic tread, the switch will be able to keep track of the distance the robot is expected to cover. Every revolution performed by the treads, which includes the limit switch being triggered twice, will increment the counter by one. This limit switch was deliberately placed near the right tread since the left tread was programmed to manage both forward and reverse directions. In order for the robot to turn, for example, at least one of the treads would be required to move in both directions. In addition to this, with the use of a limit switch, the right tread would have to be restricted from moving in the reverse direction otherwise the switch would get damaged. This also enables the microcontroller to calibrate the proper distance the robot has traveled without the inclusion of any error into the signal.

The circuit diagram of the red LED is shown below”



Schematic-08: LED circuit

5. Bill-of-Materials

Shown below is a table illustrating the materials incorporated into the design in addition to the pocket cost of each of the items. Important to note though, is that the table below is not the cost of the prototype but the bill of materials in terms of the cost that came out of the group's personal pocket. Not all the items however needed to be paid for as some of the items were provided from various sources. The sources of the unpaid items are also shown below.

Items	Cost	Provided by:
BS2 [Board of Education]	$100.00	Parallax
Vex Motors	----	Provided by group member with Vex kit
Infrared Sensor	$12.50	Parallax
Photo Resistor	----	Provided by Robotics club
Compass	$30.00	Parallax
Plexiglas	$10.00	Sid's Hardware
Chassis	----	Provided by group member with Vex kit
Tank Treads	----	Provided by group member with Vex kit
Sheet Metal	$6.00	Sid's Hardware
Miscellaneous	----	Provided by Parallax & Vex Kit by grouop member
Total	**$128.50**	

Table-01: Bill of Materials

6. Prototype Cost

The Snow Shoveling Robot was designed through various electrical, mechanical, and material components. However, it was the electrical components such as the Basic Stamp Board of Education Microcontroller and sensors that significantly make up most of the cost. Other factors that make up a good amount of the cost are the chassis and tank treads that were provided from Vex Robotics Design System. The ranges of items incorporated in the miscellaneous cost include the bolts and nuts, glue, and circuit wires. A table of the bill of materials is shown below:

Part	Quantity	Price Each	Total
BS2 [Board of Education]	1	100	100
Vex Motors	3	20	60
Infrared Sensor	1	12.5	12.5
Photo Resistor	2	1.95	3.9
Compass	1	29.95	29.95
Plexiglas	-	-	10
Chassis	-	-	31
Tank Treads	2	15	30
Sheet Metal	-	-	6
Miscellaneous	-	-	30
	Total Cost		**313.35**

Table-02: Prototype Cost

7. Cost Analysis for Mass production

Most of the mass production cost was calculated using the manufacturing websites that the parts were ordered from. These manufacturers include Parallax and Vex Robotics Design System. The mass production was calculated with a bulk quantity of 100 items. The BS2 Board of Education microcontroller from Parallax was easily found via the website with 100 Board of Education microcontrollers costing about $76 each. Other sensors within that manufacturer's site included the photoresistors and compass. The cost of the vex motors were found on the Vex Robotics Design System site. However, for most of the items, the cost for a bulk quantity of 100 items was not shown. As a result, the quantities given on the sites were analyzed and when calculated for a bulk of 100 items turned out to be approximately 25% less from the original price. Therefore, the items that lacked a bulk cost on the websites were calculated via a 25% discount for mass production. The table below includes the unit cost of a Snow Shoveling Robot manufactured for mass production in addition to the total cost of the purchase of the total 100 bulk quantity.

Part	Quantity	Price Each	Mass Production
BS2 [Board of Education]	100	75.96	7596.2
Vex Motors	100	45	4500
Infrared Sensor	100	9.4	940
Photo Resistor	100	1.46	146
Compass	100	19.2	1920
Plexiglas	100	7.50	750
Chassis	100	23.25	2325
Tank Treads	100	22.5	2250
Sheet Metal	100	4.50	450
Miscellaneous	100	23.50	$2,350.00
	Total	**173.52**	**23227.2**

Table-03: Mass Production Cost

8. Advantages/Disadvantage

Advantages

- Requires no human labor (in the actual removal of the snow)
- Performs task with minimum supervision
- Create a walkway path for pedestrians
- Provides shoe traction through rock salt
- Prevents snow/ice buildup

Disadvantages

- Restricted to user input
- Straight and flat pavement
- Stops when there is an obstacle in its path
- Stops when it is out of salts
- The user must go out in the cold to rest it when it stops.

9. PBasic Code

9.1 Commands

For the Snow Shoveling Robot project, Basic Stamp 2 was programmed using the PBasic programming language. PBasic is a language that was developed by Parallax for their Basic microcontrollers. Through the PBasic language, the microcontroller can perform many tasks, through the use of the commands that were provided by Parallax. The commands and descriptions that were relevant to this project will be discussed below:

DEBUG: debug OutputData

The debug command was used for communication between the Basic stamp and the PC via a serial cable. This command was used to display text or numbers through the PC in order for the group to maintain control and flow of the program in addition to obtaining an understanding of the significance of the command's functionality. The user can display a string of text to the PC by placing the strings in double quotation marks, for example, "Hello World". The user can also display the data in binary, hexadecimal, or decimal notation. This is accomplished by specifying prefixes such as bin, hex, or dec with the debug command.

DO…LOOP: do…loop

This command is a repeating loop that executes the program between the DO and LOOP infintely.

DO…LOOP UNTIL: do…loop until (Condition)

The DO…LOOP UNTIL command is similar to a DO…LOOP where it executes a repeating loop. However, the loop will not end until it meets the condition specified in the parenthesis.

IF…THEN: if Condition then Address

This command is a decision maker in basic stamp. It checks to see if a certain condition has been met. If the condition has been met, then it executes the specified address label. The condition set for the IF statement may be simple or complex. This condition can be used with a comparison operator, such as <, >, <=, >=, or = and conditional logic operators AND, NOT, OR, or NOR.

FOR…NEXT: for Counter = StartValue TO EndValue {StepValue}… next

This command is a repeating loop that executes a program between the FOR and NEXT command a limited number of times. This repetition is determined by the StartValue, EndValue, and the value of StepValue. The Counter will be a variable that can hold a certain amount of value specified by the user. This variable is usually a byte or a word that plays the role of a counter. The StartValue is a variable/constant/expression (0-65535) that specifies the initial value of the counter. The EndValue is a variable/constant/expression (0-65535) that signifies the end value of the counter. Once the EndValue has been exceeded, the FOR…NEXT loop stops and executes the instructions after NEXT. The StepValue is an optional variable/constant/expression (0-65535) by which the counter either increases or decreases by as the FOR…NEXT loop is running. A minus sign in front of the StepValue signifies a negative step in the counter. However, making the StartValue larger than the EndValue in Basic stamp also means that a negative step will occur in the counter. This command is generally used with a servo motor in order to rotate the motor to a specified location.

FREQOUT: freqout Pin, Period, Freq1 {,Freq2}

The FREQOUT commands can generate one or two sine waves using pulse width modulation. The Pin parameter is a variable/constant/expression (0-15) that specifies the input or output pin that will be used in Basic Stamp. In this case the pin is an output. The Period parameter is a variable/constant/expression (0-65535) that specifies the amount of time the tone(s) will be generated. Freq1 is a variable/constant/expression (0-32767) that specifies the frequency of the first tone. Freq2 is optional and similar to Freq1. If Freq2 is used then the frequency of the two will be mixed together on the specified pin. This can be used to play certain tones on a speaker or flash an LED.

GOSUB: gosub Address

The GOSUB command stores the address of the next instruction after it and then goes to the point program specified by the Address. GOSUB is also known as the "go to a subroutine". This command executes the code at the beginning of the specified address, and then when it encounters a RETURN command, it will "go to the instruction that follows the most recent GOSUB."[1] This command is useful when the same piece of code will be executed several times from multiple locations.

HIGH: high Pin

This command sets the specified pin to 1 (a +5 volt level), and makes the pin an output. The Pin is a variable/constant/expression (0-15) that specifies which input/output pin will be set to high and be made as an output.

LOW: low Pin

---

[1] Parallax, *Basic Stamp Programming Manual Version 2.0c*. http://www.parallax.com/dl/docs/prod/stamps/basicstampman2_0.pdf

This command sets the specified pin to 0 (a 0 volt level), and makes the pin an output. The Pin is a variable/constant/expression (0-15) that specifies which input/output pin will be set to low and be made as an output.

PAUSE: pause Periods

This command delays the execution of the next instruction for the specified amount of milliseconds. The Period is a variable/constant/expression (0-65535) that specifies the duration of the pause. For each unit of Period will be equivalent to one millisecond.

PULSOUT: pulsout Pin, Period

This command will send out a pulse at the specified Pin with a width equal to the value of the Period. The Pin is a variable/constant/expression (0-15) that specifies the input/output pin being used, and makes it an output. The Period determines the duration of the pulse. The unit of each Period is two microsecond. This is generally used with a servomotor to set its position at a specified angle.

RETURN: return

This command sends the program back to the address immediately following the most recent GOSUB in the program. If no GOSUB has been used, then the RETURN command will go to the first executable line of the program.

SHIFTIN: shiftin Dpin, Cpin, Mode, [Variable {\Bits} {, Variable {\Bits}…}]

This command is used to provide an easy method of acquiring data from synchronous serial devices. Dpin is a variable/constant/expression (0-15) that specifies the input/output pin that will be connected to the synchronous serial device's data output.

This will also change the mode of the pin to an input. Cpin is a variable/constant/expression (0-15) that specifies the input/output pin connected to synchronous serial device's clock input. This makes the mode of the pin to an output. Mode is a variable/constant/expression (0-3), which are predefined symbols in the PBasic language that tells the SHIFTIN the order the bits will be arranged in relationship with the clock pulses to valid data. The different modes can be seen in the table below. The Variable parameter is a variable where the incoming data will be stored. Bits is an optional variable/constant/expression (1-16) that specifies the number of bits that will be inputted by SHIFTIN. If no Bits have been specified, then the default number of bits, which is 8 bits, will be used.

Symbol	Value	Meaning
MSBPRE	0	Data is msb-first; sample bits before clock pulse
LSBPRE	1	Data is lsb-first; sample bits before clock pulse
MSBPOST	2	Data is msb-first; sample bits after clock pulse
LSBPOST	3	Data is lsb-first; sample bits after clock pulse

(Msb is most-significant bit; the highest or leftmost bit of a nibble, byte, or word. Lsb is the least-significant bit; the lowest or rightmost bit of a nibble, byte, or word.)

In a synchronous serial the timing data bits is specified in relationship to clock pulses. A synchronous serial is essentially a shift-register. The SHIFTIN command generally works by first making the clock pin output a low and then causing the data pin to switch to input mode. Afterwards it will read the data pin and generate a clock pulse (PRE mode) or generate a clock pulse and then reads the data pin (POST mode). This will continue for as many data bits required.

SHIFTOUT: shiftout Dpin, Cpin, Mode, [Variable {\Bits} {, Variable {\Bits}…}]

The SHIFTOUT commands works in the same manner as the SHIFTIN command, which used to acquire data from synchronous serial devices. However, this command shifts the data out rather than in. SHIFTIN and SHIFTOUT are generally used with ADCs or parallax compass modules.

9.2 Basic Code Outline

The main goal of the project is to create a snow shoveling robot that is autonomous and user friendly. In order to accomplish this, various sensors, actuators, a Basic Stamp microcontroller, and construction parts were used. The first step is to get user input regarding the distance the robot is going to travel in the forward direction and in the reverse direction. This is done by using two push buttons and a speaker. Each time the first button is pressed, a sound is made to indicate that the robot will travel an additional 50cm. Once the user is satisfied with the number of centimeters it will travel, the user will press the second button for confirmation. The same procedure is done to determine how far the robot will travel in the reverse direction. Once the number of centimeters has been determined the robot will open the salt dispenser and start moving forward. In order, to maintain a straight path, a compass module was utilized, so that if the robot goes off course, the proper action may be taken to return it to the straight path. If the robot deviated to far left, the right motor is turned off while the left motor runs, causing the robot to turn a few degrees toward the forward direction. The opposite is done if the robot deviates too far right.

As the robot moves forward, three sensors are checked, namely the IR sensor, the photo-resistor, and the limit switch. The limit switch determines the number of centimeters the robot has traveled, and it is also use to trigger the robot to keep moving forward. The infrared sensor is used for obstacle detection such that if an object is within 16” of the robot, the robot will stop. The speaker will sound an alarm at a dual frequency of 5000Hz and 5050Hz while an LED flashes on and off. To deactivate the alarm, the obstacle should be moved out of the way, or the robot should be moved, and then pressing the second button resets the state of the robot. The photo-resistor used to determine the salt level in the robot’s salt container is located at the bottom of the container. If the level of the salt is too low, i.e. enough light shines on the bottom of the container, the robot will stop and an alarm with a dual frequency of 4000Hz and 4005 Hz, while an LED flashes on and off. The robot waits for the user to refill the salt dispenser, and push the second button to restart the state of the robot. This process will keep repeating until the robot finishes shoveling the snow in the forward and reverse direction. Once completed, the program will end and close the salt container, and a new set of instructions can be executed. For further details of the code please refer to Appendix A of the report.

10. Justification of Selected Sensors/ICs/Mechanisms

One Servo; - This was used to open and close the bay at the rear of the robot that dispenses the salt. A servo was chosen for this task because the servo could be rotated to specific angular locations (completely opening and closing the salt bay)

One Piezo Speaker; - One use of the speaker is to alert the user that either the salt level is low or there is an object in the robot's path (it does this through a series of beeps). The speaker was used for such a task because the user may not have his/her eyes on the robot when one (or both) of the two pre-mentioned problem occur. So as long as the user is within the sound range of the speaker, they will be informed of the predicament of the robot. Another used of the speaker is to indicator how many feet the user wants the robot to move (the speaker beeps every time the user pushes the button).

One LED; - The Light Emitting Diode is used in conjunction with the Piezo to indicate that the robot is either out of salt or there is an obstacle in its path. So when the user hears the speaker and looks out a window or steps outside, they will be able to tell exactly where the robot is.

One IR Sensor; - The Infrared Sensor (from Acroname Easier Robotics) is mounted on the front of the robot and it is used to detect whether or not there is an obstacle in the robots path. This IR sensor was chosen because it operates between -10°C to +60°C and its range (1.5” to 12”) was ideal for conditions the robot will be operating in.

One A2D IC Chip; - The A2D converter was used to convert the analog output of the IR sensor to a discrete digital output. Although it was possible to incorporate the analog output into the code; the digital output made it possible to use more precise definition of what constitutes an obstacle in front of the robot at a certain distance.

One Photo-resistor; - The Photo-resistor was placed “face-up” under the rear of the robot. The rear section of the robot houses the rock salt that is dispensed after the robot has shoveled an area. The Photo-resistor is incorporated with high and low outputs value that is proportional to the amount of light reaching the photo-resistor. As the level of rock salt decreases, the amount of light reaching the photo-resistor increases, and the voltage passing to the photo-resistor will eventually increase to read a high value in basic stamp.

Two DC Motors; - The robot is mounted on treads and the treads are controlled by two dc motors (from Vex Robotics). Each motor has a stall torque of approximately 6.5 in-lbs. The two motors were chosen because of their high torque output and there are three wires connected to the motors (Black - ground; Orange - (+) power; White - PWM motor control signal), which means that they can be controlled using an external power supply and the BS2 micro-controller without the use of an H-Bridge for speed and direction control.

One Compass; - Small pebbles and fracture along the robot's path (and to a lesser extent, the two motors are not synchronized) all hinders the robot's ability to move along a straight path. To combat this problem, a compass was added to the list of sensors used by the robot. By continuously monitoring the compass the robot can tell whether or not it is still heading in the "right" direction and make the necessary adjustments. The compass is also used to allow the robot to turn around (a 180° turn).

Two Push Buttons; - The push buttons are use to allow the user to input the distance he/she wants the robot to travel. Push buttons were chosen because they are amongst the most efficient and user friendly way to allow humans to interact with machines. The user will start by pressing button 1. Each time he/she presses the button he/she is telling the robot to go one more foot in that direction (the total number of times he/she the button represent the total distance the robot should travel, in a particular direction). After the user is done imputing the length he/she must press button 2 to confirm the input length. The user must then use button 1 again to input the length he/she wants the robot to travel in the opposite direction. Then he/she will press button 2 to confirm the second length and then the robot will by armed and ready for battle with the snow on the sidewalk. The press button 2 will also be used to move the robot forward when the salt level is too low or an obstacle is on the path of the robot.

Limit Switch; - a Limit Switch is interface with the treads to measure the distance the robot has traveled. The treads closes the switch twice (equal distances apart) on every rotation, therefore by measuring the length of the treads (17.5 inches) the robot let how far it has traveled. This system was chosen because it is least effected by the harsh weather conditions within which the robot is designed to operate.

11. References and Credits

http://www.nd.edu/~srdesign/ame470/project3/prva/documents/conceptsheehan.pdf

http://www.vexlabs.com/

http://www.parallax.com

http://www.acronym.com

12. Appendix A

```
' {$STAMP BS2}
' {$PBASIC 2.5}
'          Snow Shoveling Robot PBasic Code
' ==========================================================================
'Declarations
DinDout PIN 8                'P8 transceives to/from Din/Dout
Clk PIN 6                    'P6 sends pulses to HM55B's Clk
En PIN 7                     'P7 controls HM55B's /EN
Reset CON %0000              'Reset command for HM55B
Measure CON %1000            'Start measurement command
Report CON %1100             'Get status/axis values command
Ready CON %1100              '11 -> Done, 00 -> no errors
NegMask CON %1111100000000000    'For 11-bit negative to 16-bits
x VAR Word                   'x-axis data
y VAR Word                   'y-axis data
status VAR Nib               'Status flags
angle VAR Word               'Store angle measurement
Fcount VAR Byte              'Variable to hold forward value entered by user
Bcount VAR Byte              'Variable to hold backward value entered by user
A2DOut VAR Byte              '8-bit variable for A2D output
actualCount VAR Word         'Counter for movement (forward or back)
goBack VAR Bit               'Boolean for determining if robot shoveled back yet
straight VAR Word            'Degree value of forward movement
tempAngle VAR Word           'Temporary variable to find average angle
tempCount VAR Byte           'Temporary variable to count for various one-time loops

'Initializations
goBack=1
tempAngle=0
tempCount=0

'=====Start of Main Program==============================================
GOSUB getUserInput           'Wait for user input
actualCount=0                'Set 'actualCount' counter to zero
PAUSE 3000                   'Pause for 3 seconds so user can place robot in proper orientation

'Finds average of ten angles for a good estimate of the 'straight' angle
FOR tempCount=0 TO 9         'Loop 10 times
  GOSUB getAngle             'Get angle of current orientation
  tempAngle=tempAngle+angle      'Add the current angle value to the average angle variable
NEXT
tempAngle=tempAngle/tempCount    'Calculate the average value of the angle
tempAngle=tempAngle+360          'Add 360 so that when going from 0 degrees to 359 degrees, the
                             '    comparison during main loop will not become negative, causing the
                             '    value of angle to become 65535
straight=tempAngle           'Set 'straight' angle to be to be the calculated average angle

'-----Main Routine-------------------------------------------------------
mainRoutine:
```

```
GOSUB openRear                  'Open the salt dispenser

DO ' Main loop                  'Main loop
GOSUB getAngle                  'Get angle of current orientation
angle=angle+360                 'Add 360 sO that when going from 0 degrees to 359 degrees, the
                          '   COMparison during main loop will not become negative, causing the
                          '   value of angle to become 65535

GOSUB detectObstacle            'Check if an obstacle is in the way and act accordingly

'Move the motors according to the orientation of the robot
IF (angle <= straight+6 AND angle >= straight-6)  THEN       'If robot is within straight range...
  DO                            'Begin loop to run motors
    PULSOUT 2, 1100             'Run right motor forward
    PULSOUT 0, 500              'Run left motor forward
    PAUSE 10                    'Pause for .01s
  LOOP UNTIL (IN4=0)            'Continue loop until the limit switch is hit
  PAUSE 80                      'Pause for .08s to allow limit switch to be released
  actualCount=actualCount+1     'Increment 'actualCounter' counter

ELSEIF (straight+45 >= angle AND angle > straight+6) THEN     'If angle is too far left...
  FOR tempCount=0 TO 5
    PULSOUT 2, 1100             'Output five pulses to the right motor (and no pulses to the left motor)
    PAUSE 20                    'Required settling time for the PULSOUT command
  NEXT

ELSEIF (straight-45 <= angle AND angle < straight-6) THEN      'If angle is too far right...
  FOR tempCount=0 TO 5
    PULSOUT 0, 500              'Output five pulses to the left motor (and no pulses to the right motor)
    PAUSE 20                    'Required settling time for the PULSOUT command
  NEXT
ENDIF

IF (IN5 = 0) THEN               'If the photo-resistor senses light conditions, i.e. salt is low...
  GOSUB salt                    'Go to "low salt mode"
ENDIF
LOOP UNTIL((actualCount=Fcount AND goBack=1) OR (actualCount=Bcount AND goBack=0)) 'Repeat
main loop
                                            'unless the counter has reached its
                                            'end for the appropriate direction.
IF(goBack=0) THEN               'If robot has already shoveled the other side of the sidewalk...
  GOSUB closeRear               'Close the salt compartment
  END                           'End the program
ELSEIF(goBack=1) THEN           'If robot hasn't shoveled the other side yet...
  actualCount=0                 'Reset the 'actualCount' counter
  goBack=0                      'Set the 'goBack' variable to zero to signify that robot is about to turn around
  GOSUB turn                    'Turn around
  GOTO mainRoutine              'Go to main program and shovel the other side of the sidewalk
ENDIF
```

```
'-----Subroutines------------------------------------------------------
getAngle:
'compass module subroutine
HIGH En: LOW En                              'Send reset command to HM55B
SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]
HIGH En: LOW En                              'HM55B start measurement command
SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
status = 0                                   'Clear previous status flags
DO                                           'Status flag checking loop
  HIGH En: LOW En                            'Measurement status command
  SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
  SHIFTIN DinDout,clk,MSBPOST,[Status\4]          'Get Status
LOOP UNTIL status = Ready                    'Exit loop when status is ready
SHIFTIN DinDout,clk,MSBPOST,[x\11,y\11]          'Get x & y axis values
HIGH En                                      'Disable module
IF (y.BIT10 = 1) THEN y = y | NegMask            'Store 11-bits as signed word
IF (x.BIT10 = 1) THEN x = x | NegMask            'Repeat for other axis
angle = x ATN -y                             ' Convert x and y to brads
angle = angle */ 360                         ' Convert brads to degrees
RETURN



getUserInput:
'Waits for user to input how far the robot should shovel forward and then backward.
HIGH 3                        'Turn the red LED on to signify that user input is needed
GOSUB closeRear               'Close the salt compartment

DO                            'Begin loop to get user input for forward distance
  IF(IN9=1) THEN                    'If button 1 is pressed...
    Fcount=Fcount+1                 'Add one to the forward counter
    FREQOUT 11, 300, 4000, 4005          'Play a sound
    PAUSE 100                       'Pause for .1s
  ENDIF
LOOP UNTIL (IN10=1)               'Continue loop until button 2 is pressed
DO
LOOP UNTIL (IN10=0)               'Wait for button 2 to be released

DO                            'Begin loop to get user input for reverse distance
  IF(IN9=1) THEN                    'If button 1 is pressed...
    Bcount=Bcount+1                 'Add one to the backward counter
    FREQOUT 11, 300, 4000, 4005          'Play a sound
    PAUSE 100                       'Pause for .1s
  ENDIF
LOOP UNTIL (IN10=1)               'Continue loop until button 2 is pressed, then stop getting user input
DO
LOOP UNTIL (IN10=0)               'Wait for button 2 to be released

Fcount=Fcount*2                  'Multiply the forward and reverse counters by a conversion factor of 2 to convert
Bcount=Bcount*2                  '   counter value to distance in centimeters.
```

```
LOW 3                               'Turn red LED off to signify that user input is no longer needed
RETURN                                'Return to section that the function was called from



detectObstacle:
'Check if an obstacle is in the way of the robot.
LOW 14                              'Select the A2D
PULSOUT 12,1                            'Send the first “setup clock pulse”
A2DOut = 0                            'Set A2DOut to zero
FOR tempCount = 1 TO 8                'Loop 8 times to get the 8 data bits
  PULSOUT 12,1                        'Send a clock pulse
  A2DOut = A2DOut * 2                   'Shift bits once to the left
  A2DOut = A2DOut + IN13                  'Add A2DOut to the incoming bit
NEXT
HIGH 14                             'De-select the A2D
IF(A2DOut>=100) THEN obstacleDetected'If an obstacle is detected to be approximately 40 centimeters away,
                                '   go to "obstacle detected mode"
RETURN                                'Return to section that the function was called from



obstacleDetected:
'Stops robot, closes salt compartment, and lets user know that an obstacle is present by playing a sound and
flashing an LED.
GOSUB closeRear                         'Close the salt compartment
DO                                  'Begin loop to let user know that the robot encountered an obstacle
  LOW 3                               'Turn off red LED
  FREQOUT 11, 100, 5000, 5050           'Play a sound
  HIGH 3                              'Turn on red LED
  PAUSE 100                             'Pause .1ms
LOOP UNTIL (IN10 = 1)                   'Continue loop until user presses button 2
LOW 3                               'Make sure LED is off
GOSUB openRear                          'Open the salt compartment
RETURN                                'Return to section that the function was called from



salt:
'Stops robot, closes salt compartment, and lets user know that salt level is low by playing a sound and flashing
an LED.
GOSUB closeRear                         'Close the salt compartment
DO                                  'Begin loop to let user know that the salt level is low
  LOW 3                               'Turn off red LED
  FREQOUT 11, 100, 4000, 4005           'Play a sound
  HIGH 3                              'Turn on red LED
  PAUSE 100                             'Pause .1ms
LOOP UNTIL (IN10 = 1)                   'Continue loop until user presses button 2
LOW 3                               'Make sure LED is off
GOSUB openRear                          'Open the salt compartment
RETURN                                'Return to section that the function was called from
```

```
turn:
'Turns the robot 180 degrees and on the opposite side of the sidewalk.
GOSUB closeRear                    'Close the salt compartment
tempCount=0                        'Set temporary counter to zero
DO                                 'Begin loop to turn another 90 degrees
  GOSUB getAngle                   'Get angle of current orientation
  angle=angle+360                  'add 360 so that when going from 0 degrees to 359 degrees, the
                                   '   comparison during main loop will not become negative, causing the
                                   '   value of angle to become 65535

  FOR tempCount=0 TO 20            'Run motors for 20 cycles
    PULSOUT 2, 1100                'Run the right motor forward
    PULSOUT 0, 1100                'Run the left motor in reverse
  NEXT
LOOP UNTIL (angle<straight-90)    'Continue loop until the robot is oriented 90 degrees left of its original
orientation

tempCount=0                        'Set temporary counter to zero
DO                                 'Begin loop to move robot to the left
  DO                               'Begin loop to run motors
    PULSOUT 2, 1100                'Run right motor forward
    PULSOUT 0, 500                 'Run left motor forward
    PAUSE 10                       'Pause for .01s
  LOOP UNTIL (IN4=0)               'Continue loop until the limit switch is pressed
  PAUSE 90                         'Pause for .09s to allow limit switch to be released
  tempCount=tempCount+1            'Increment the temporary counter
LOOP UNTIL (tempCount=2)           'Continue loop until the limit switch is pressed two times

tempCount=0                        'Set temporary counter to zero
DO                                 'Begin loop to turn another 90 degrees
  GOSUB getAngle                   'Get angle of current orientation
  angle=angle+360                  'Add 360 so that when going from 0 degrees to 359 degrees, the
                                   '   comparison during main loop will not become negative, causing the
                                   '   value of angle to become 65535
  FOR tempCount=0 TO 20            'Run motors for 20 cycles
    PULSOUT 2, 1100                'Run the right motor forward
    PULSOUT 0, 1100                'Run the left motor in reverse
  NEXT
LOOP UNTIL (angle<straight-150)   'Continue loop until the robot is oriented 150 degrees left of its original
orientation

straight=straight-180              'Set the new straight value to be 180 degrees left of the original straight value
GOSUB openRear                     'Open the salt compartment
RETURN                             'Return to section that the function was called from


closeRear:
```

```
'Closes the salt compartment
FOR tempCount = 1 TO 20
  PULSOUT 1, 800              'Turn the servo motor counter-clockwise
  PAUSE 20
NEXT
RETURN                      'Return to section that the function was called from


openRear:
'Opens the salt compartment
FOR tempCount = 1 TO 20
  PULSOUT 1, 500              'Turn the servo motor counter-clockwise
  PAUSE 20
NEXT
RETURN                      'Return to section that the function was called from
```