

# **State University of New York at Buffalo**

*Department of Mechanical and Aerospace Engineering*

MAE 405/505: MECHATRONICS

## **EXPERIMENT #3**

### **Controlling a Wheeled Mobile Robot With an Infrared Remote**

**DATE:** April 18, 2002.

Instructor: Dr. V. Krovi

SUBMITTED BY:

Lee Leng Feng  
Rajankumar Bhatt  
Krishnakumar Ramamoorthy

## TABLE OF CONTENTS:

<b>Topics Covered</b>	<b>Pages</b>
1. Abstract.	1
2. Objectives	1
3. Goals	1
4. Description of the Tasks.	
<i>-Part A. Assembly and testing of the BOE-Bot.</i>	1
<i>-Part B. Interfacing a infrared controller with BOE-Bot.</i>	2
<i>-Part C. Combining Hardware and Software Mechatronic Elements</i>	2
5. Hardware used.	3
6. Laboratory Procedures, Analysis & Problems.	
<i>-Part A. Assembly and Testing of the BOE-Bot.</i>	4
<i>-Part B. Interfacing a infrared controller with BOE-Bot.</i>	11
<i>-Part C. Combining Hardware and Software Mechatronic Elements</i>	12
7. Flow Chart	16
8. Source Code of the Experiment	17
9. Discussion	19
10. Evaluation of the Performance	20
11. Components used in this experiment and its price	21
12. Contribution of each member	21
13. Conclusion	21
14. Reference	22

## **LIST OF FIGURES**

Figure	Description	Page
1	A Boe-Bot	3
2	SONY RM-V201 remote	3
3	The pulse train used in the pre-modified servomotor	5
4	Plot for calibration of Right wheel	7
5	Plot for Calibration of Right Wheel in Linear region	8
6	Plot for calibration of Left wheel	9
7	Plot for Calibration of Right Wheel in Linear region	10
8	Circuit diagram for interfacing the IR Detector	11
9	Flow Chart for the working of the program	16

## **LIST OF TABLES**

Table	Description	Page
1	Calibration data of Right and Left servomotors	6
2	Components use in this lab and its Price	21
3	Contribution of each member	21

**ABSTRACT:**

In this lab, we examine the interfacing of a small semi-autonomous wheeled mobile robot to a microprocessor, focusing especially on development alternative user-interfaces and performance calibration. We built a mobile robot from Boe-Bot Kits that is a product from Parrallaxinc Company. An infrared controller is then interfaced with the mobile robot that we have built to control its motions. We have successfully interfaced the infrared controller with the mobile robot and we can control the period of the motion by increasing the time period of our motion execution. Beside this, using a single command, we successfully control the mobile robot to turn a circle with two-difference radius.

**OBJECTIVES:**

The objectives of this experiment are:

1. Build and calibrate a wheeled mobile robot from the kit components.
2. Interface a 38KHz Infrared (IR) remote sensor to read input signals from an IR TV remote control.
3. Gain experience with implementing alternative and unconventional user-interfaces.

**GOALS:**

The goals of this experiment are:

1. Gain knowledge of interfacing an infrared controller with a microprocessor.
2. Know how to calibrate a servomotor and knows its limitations.
3. Learn how to evaluate the hardware and the limitations of our mobile robot.

**DESCRIPTION OF THE TASKS:**

There are three parts in this experiment:

***Part A. Assembly and testing of the BOE-Bot.***

The first part of this experiment is to assemble the mobile robot from the kits provided. The BOE-Bot kit and the complementary series of exercises discussed in the Robotics Student Workbook provide an expedient method for creating simple wheeled mobile robots that will serve as our experimental testbed. In this experiment, the hardware and software framework was explored so that we can then further extend the capabilities of this BOE-Bot for Part C of this lab and for the final projects. Hence, in this first part, the tasks are to assemble, test and calibrate a differentially driven mobile robot from the BOE-bot kit components.

### ***Part B. Interfacing a infrared controller with BOE-Bot.***

In this part of the experiment, the task is to receive and decode the 38Khz modulated IR signal from a TV remote, using the Basic Stamp and a 38Khz IR receiver, and display the decoded result on an LCD screen. Remotes from different manufacturers may use different coding schema for sending information. A particular remote control was selected and the corresponding scheme was to be found from the Internet.

Setup the Stamp circuitry with the IR receiver module to the mobile robot. Create a program that can recognize the relevant bits from the IR message. Once we have this part working integrate it with the program for motor control that we developed in Part A, the remote control will take on the role of the keypad – i.e. pressing keys on the remote control will control the operation of the motor.


### ***Part C: Giving specific motions to the mobile robot.***

Many alternative interfaces can be created by a combination of hardware and software mechatronic elements. In this experiment, two basic motions was required to implement on the mobile robot, as describe follow:

Mode 1: Steering wheels and joysticks are two of the most common hardware interface devices used for operation of remote-control toys. We, however, would like to use an infrared TV remote coupled with suitable electronics as a means to actively control the operation of the Wheeled Mobile Robot. For example, the Up/Down Left/Right keys on the IR remote could be used to drive the robot forward and backward. Thus, in this interactive operation mode, we would like to be able to drive the robot using the remote control as the active driving interface.


Mode 2: Very often, in order to partially automate the task of driving, it is beneficial to preprogram most of the motion control into a series of parametric primitives/subroutines such as Drive\_in\_a\_circle, Drive\_Straight, Turn\_on\_the\_spot. Thus, in the second mode, we would like to explore this process further using one sample primitive, tracing a circle. Create a generic subroutine/program that can move the mobile robot in a circular trajectory, for a given radius of the circle inputted using the IR remote control.

**HARDWARE USED:***1. BOE-Bot mobile robot Kits.*

 <p><i>Figure 1. A BOE Bot.</i></p>	<b>Power Supply</b>	4 AA's for BASIC Stamp and motor control (not included)
	<b>Assembly Tools</b>	Angled cutters, small Phillips screwdriver, and 1/4" wrench.
	<b>Time Required to Build and Program</b>	2.0 hours - Boe-Bot uses the pre-built Board of Education to save time.
	<b>I/O Components</b>	LEDs, speaker, pushbutton, photoresistors, resistors and capacitors, infrared LEDs and receivers, '555 timer.
	<b>Use of BASIC Stamp I/O pins</b>	Components are placed on the breadboard and may be moved around. Flexible use of I/O pins.
	<b>Object Detection</b>	Infrared object detection, object detection with whiskers
	<b>Expandability</b>	Breadboard or through-hole AppMod will fit, but hang off the side of the Boe-Bot. Expandability is possible with the Compass Appmod or Line-Follower Module. (not included)

Note: For more information, [www.parallaxinc.com/downloads/Stamps\\_in\\_Class/rob.pdf](http://www.parallaxinc.com/downloads/Stamps_in_Class/rob.pdf)

*2. SONY universal remote control. Model: RM-V201.*

 <p><i>Figure 2. SONY RM-V201 remote.</i></p>	<ul style="list-style-type: none"> <li>• <b>Operating distance:</b> Approx. 23 feet (7m)</li> <li>• <b>Power requirements Remote:</b> AA x 2 Batteries (not supplied)</li> <li>• <b>Battery life:</b> Approx. 6 months (varies depending on frequency of use)</li> <li>• <b>Weight:</b> Approx. 3.5 oz. (100g), not including batteries</li> <li>• <b>Dimension:</b> 2 1/2" x 7 5/8" x 1 1/4" (61 x 191 x 30mm)</li> </ul>
--	--

## LABORATORY PROCEDURES, ANALYSIS & PROBLEMS.

Each of the procedures, analysis and problem that we faced in different task of this experiment will be explain in the following paragraph.

### *Part A. Assembly and testing of the BOE-Bot.*

#### **Procedures:**

1. Once we have the Kits that are provided, we check the entire component in the Kits to make sure all the parts is equipped.
2. The hardware of the mobile robot is built according to the way shown in the manual. [4]

*Caution: here we need to know that we cannot connect the servo motor straight away to the AC adapter or a 9V battery. This could damage the motor.[4]*

3. After the hardware part of the motor was built, the circuit was again checked and the servomotor was tested.
4. We tested the servomotor by using the available program provided in the Workbook followed with the Kits. [4]. The motor that provided in this Kits is pre-modified version in which we can use the programming language shown in the Workbook to control the servo directly without doing any modification.
5. The theory for controlling the servomotor is discussed in the analysis portion of this section.
6. After all the parts are working properly, we proceed to second part of our experiment.

#### **Analysis:**

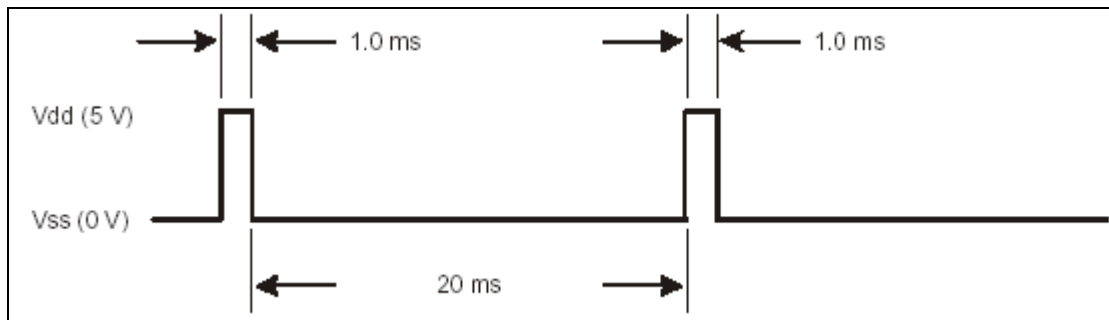
##### *Pre-modified servomotor:*

1. Normal or un-modified hobby servos are very popular for controlling the steering systems in radio-controlled cars, boats, and planes. These servos are designed to control the position of something such as a steering flap on a radio-controlled airplane. Their range of motion is typically 90° or 180°, and they are great for applications where inexpensive, accurate high-torque positioning motion is required. The position of these servos is controlled by an electronic signal called a pulse train. An un-modified hobby servo has built-in mechanical stoppers to prevent it from turning beyond its 90° or 180° range of motion. It also has internal mechanical linkages for position feedback so that the electronic circuit that controls the DC motor inside the servo knows where to turn to in response to a pulse train. This kind of servo motor does not meet our requirement in this experiment.[4]

2. On the other hand, a pre-modified servo does not have the position feedback and mechanical stoppers that found in normal hobby servos. We can send the same electronic signals a pulse train to a pre-modified servo as we normally send to a hobby servo. In a hobby servo, a given pulse train makes it turn to a certain position and stay there. On the other hand, the same pulse train causes a pre-modified servo to turn continuously. [4] The pulse train also sets the pre-modified servo's speed and direction. So, instead of controlling airplane flaps, the pre-modified servos are used as BASIC Stamp controlled motors that make the Boe-Bot's wheels turn.

*Pulse train:*

1. The control signal the BASIC Stamp sends to the servo's control line is called a "pulse train," and an example of one is shown in Figure 3. The BASIC Stamp can be programmed to produce this waveform using any of its I/O pins. First, the BASIC Stamp sets the voltage to 0 V (low) for 20 ms. Then, it sets the voltage to 5 V (high) for 1.0 ms. Then, it starts over with a low output for another 20ms, and a high output for another 1.0 ms, and so on.



*Figure 3. The pulse train used in the pre-modified servomotor.*

2. Pre-modified servo can be pulsed to make its output shaft turn continuously. The pulse widths for pre-modified servos range between 1.0 and 2.0 ms for full speed clockwise and counterclockwise respectively. Gives a pre-modified servo 1.25 ms pulses, it will turn clockwise at roughly half of full speed. Whereas give a pre-modified servo 1.90 ms pulses, the servo will turn at almost full speed counterclockwise. The "center pulse width" is 1.5 ms, and that makes the servo stay still.
3. The program that control the pulse is called PULSOUT *Pin, period*. For example, to give pin 12 on the basic stamp a 1000 period, will shown as:

PULSE 12, 1000

It mean that it send the 5 V signal in pulse time of  $1/1000=1$  ms. This will cause the servo motor to turn clockwise in full speed. Giving a period of 500, will give a pulse time of  $1/500=2$ ms for every 5 V signals send to pin 12. This will cause the servo motor to turn in counterclockwise in full speed. By controlling the period of

the signal sending to the pin, we can control the servo motor speed and also its direction easily.

### *Calibrations:*

As will discuss in the problems section in the following paragraph, we had some problems with the servomotors that we got with the Boe-Bot kit. After replacing them we started our calibration procedure.

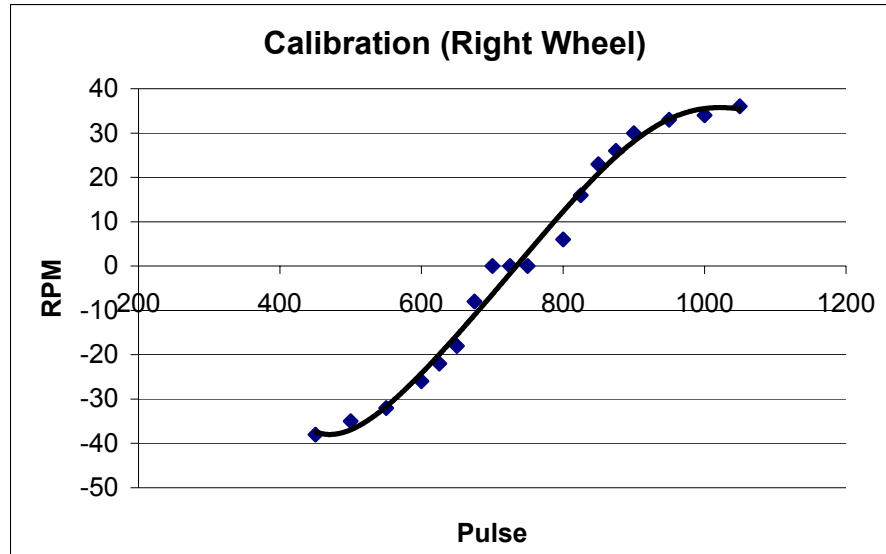
1. First step was to stop the servomotors by giving certain period of the pulse, which would be around 750. With the new servomotors, we could successfully stop the motors from rotating at 750.
2. Then we were to prepare a calibration curve for the motors. We found out that both the motors behave in different manners and thus we had two different calibration curves. For preparing calibration curve, we marked one direction on the wheels. Then we let them rotate by giving a specific period of pulse that varied from 400 to 1100. Manual [4] says that the servomotors saturate at 500 in one direction and at 1000 in the other. We got the following table for the Left and Right servomotors.

<b>Right Servomotor</b>		<b>Left Servomotor</b>	
<b>Pulse</b>	<b>RPM</b>	<b>Pulse</b>	<b>RPM</b>
400	-38	400	39
450	-38	450	36
500	-35	500	36
550	-32	550	34
600	-26	600	28
625	-22	625	25
650	-18	650	20
675	-8	675	13
700	0	700	3
725	0	725	0
750	0	750	0
800	6	800	0
825	16	825	-15
850	23	850	-22
875	26	875	-26
900	30	900	-28
950	33	950	-34
1000	34	1000	-38
1050	36	1050	-38
1100	36	1100	-

*Table1: Calibration data of Right and Left servomotors.*

*Calibration for right wheel:*

After Calibration, we tried to fit a polynomial curve using Excel spreadsheet program to the data points and we result was shown in Figure 4.



*Figure 4. Plot for calibration of Right wheel*

The equation that fit the trendline in Figure 4 was given as follow:

$$y = 1E-14x^6 - 6E-11x^5 + 1E-07x^4 - 0.0001x^3 + 0.0658x^2 - 19.732x + 2387.7$$

where y is RPM and x is the Pulse given in the command.

From Figure 4, it is clear that the curve is very much linear in some region. For our calibration purpose, we would like to use the relation in this region so that to make the calibration much straight forward. Those data points are marked with a dark background in the calibration table. For this reason, we tried to fit a linear curve to those data points and the result was shown in Figure 5.

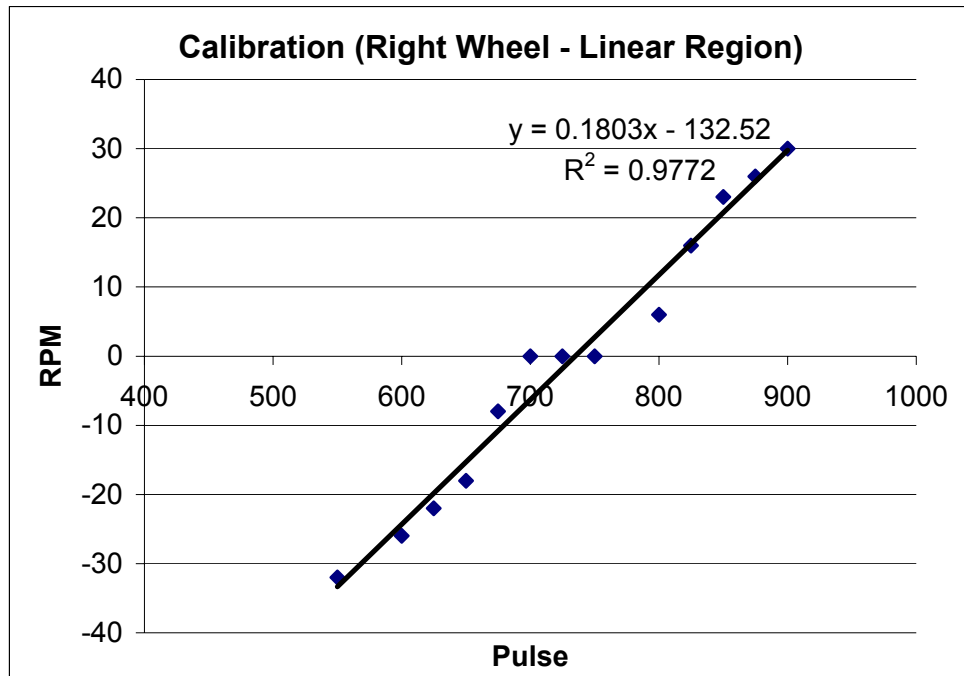


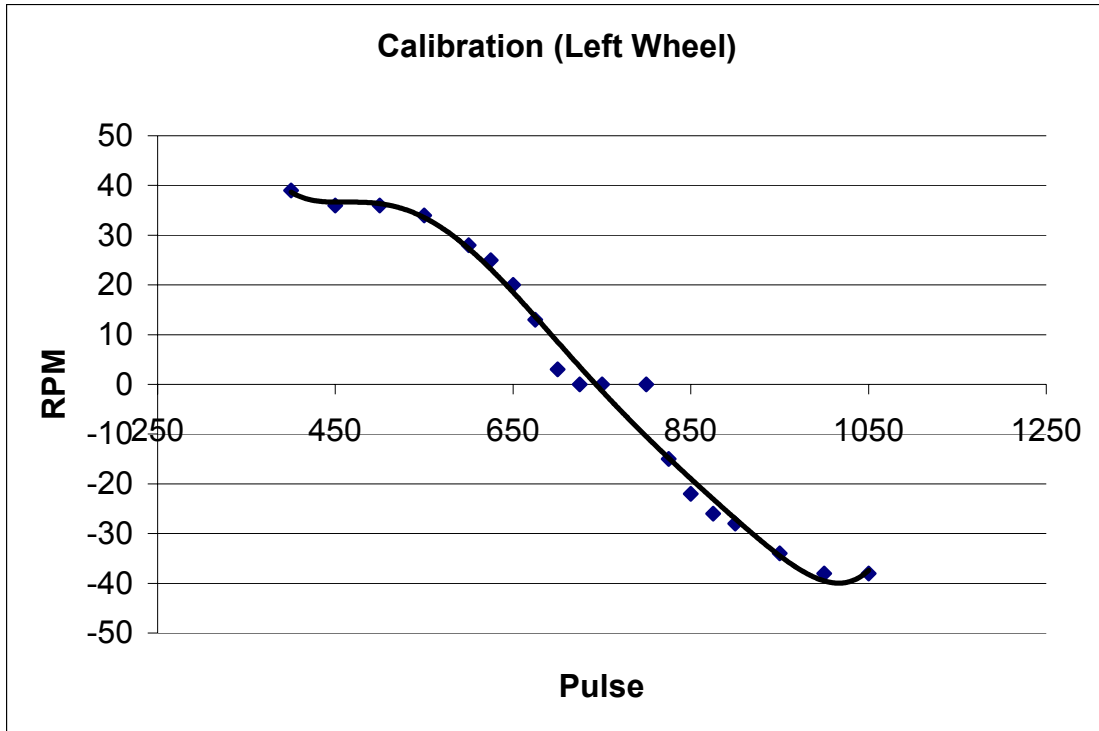
Figure 5. Plot for Calibration of Right Wheel in Linear region

From the  $R^2$  value from given by the *trendline*, we can know that 97.72% of the data fits to the linear equation. Using a linear fit relation, the equation of the line is given by:

$$y = 0.1803x - 132.52$$

*Calibration for left wheel:*

The calibration curve for the left wheel was also drawn and was shown in Figure 6.



*Figure 6. Plot for calibration of Left wheel*

The equation that fit the curve is a polynomial curve as describe in following equation:

$$y = 3E-14x^6 - 1E-10x^5 + 3E-07x^4 - 0.0002x^3 + 0.1256x^2 - 33.186x + 3588.5$$

From Figure 6, we can also see the there is a portion of the curve can be describe by linear equation, these data points are those shaded in Table 1. We plot the data in this region as in Figure 7 and find out the linear relation of the rotation of the wheels and the pulse given to the servo motor, this is done in Figure 7.

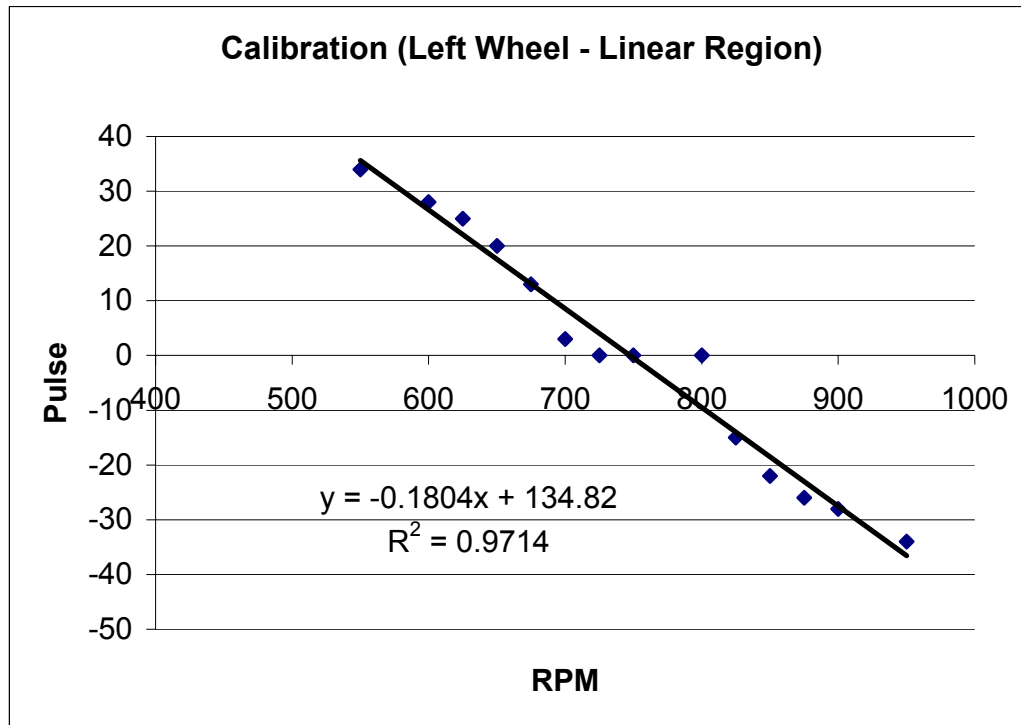


Figure 7. Plot for Calibration of Right Wheel in Linear region

The equation of the linear curve was shown in Figure 7, from the  $R^2$  value of the linear fit, we can see that 97.14% of the data fits that linear curve. The equation of the linear fit is give by:

$$y = -0.1804x + 134.82$$

Again, the y value is the pulse that we should give to the servo motor and x is the desired rotation speed that we want.

After having the calibration data for both the wheel, we can now use the relation for implementing different motion to the mobile robot by given different rotational speed to two different wheels. This will proceed in the following task.

**Problems:**

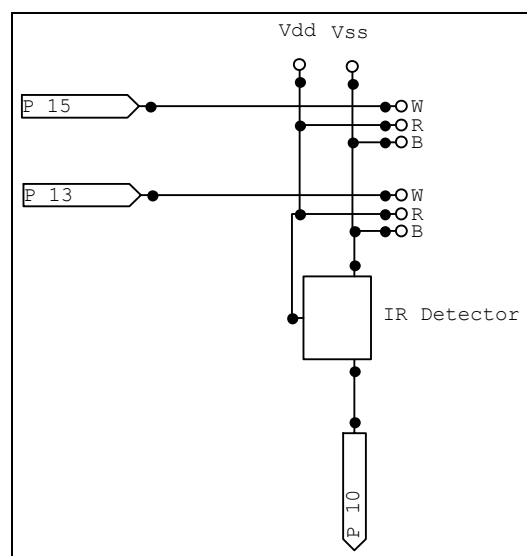
1. In this section, we have faced a problem with our servomotor. We found that the servomotor that provided by the Kits does not give the desired turning direction when we gave the specific pulse period to the pin. Beside this, it does not stop when the period of the pulse is give by 750.
2. We have discovered this problem and we try to solve it by changing the potentiometer inside the servo. This is done by adjusting the two terminal of the potentiometer in the servomotor such that the are having the same resistance

value, as suggested by one of the group who also faced the same problem with their servomotor.

3. However, after we have adjusted the potentiometer in the servo motor, the problem still remain and we finally make a conclusion that the servo motor that we have in our kits is broken. We then get a new motor to continue our experiment but we have spent a lot of time in figuring out the problem.

### ***Part B. Interfacing an infrared controller with BOE-Bot.***

#### **Circuit:**



*Figure 8: Circuit diagram for interfacing the IR Detector*

#### **Procedures:**

1. We first connect the circuit and the infrared receiver on the board, we use the pin 10 to take the input from the remote control.
2. Then we have the data sheet of the Sharp IS1U60 infrared receiver and the code that was provided by Dr. Krovi that received the signal and processed it such that the Basic stamp can understand the signal. The analysis of the sony remote and the code is given in analysis part.
3. Using the debug command in the program, we can test the remote control by pressing a button on the remote control and a value will shows in the debug window, by doing this, we can know that our circuit in instead connected correctly and working properly.
4. A specific task is then give to each different input from the remote control. This part will be shown in the programming part of this section.

**Analysis:**

The Infrared remote that we used for operating our Boe-Bot kit is SONY RM-V201. Sony remotes use pulse coding scheme. [6] The light waves are modulated by the emitter at a frequency of 40 KHz. This may have been done in order to cut out other sources of IR such as electric lamps, etc. The remote sends the signal every 25 milliseconds if the button is pressed. It sends a header, which is followed by 12 bits of data. The header is 4T long where T is 550 microseconds. Then, the remote sends a stream of data that is 12 bits long. Zero is represented by a pulse with length T followed by space of length T while One is represented by a pulse with length 2T followed by space of length T.

These 12 bits of data contain 7 bits of data followed by 5 bits of address. The 7 bits of data contain the information about which button was pressed and the 5 bits of address contain the information about on what device to use that data. The various devices could be TV, VCR, Tuner etc.

**Problems:**

As the code was provided, we really didn't have any problems with getting the Infrared receiver working. But we were first misguided by the manual[5] which shows a circuit with a Capacitor and resistor. The capacitor and resistor with those values were not provided with the kit. But when we saw one of such receivers working, which was not having a capacitor and a resistor, we neglected the circuit in the manual and connected it directly and made it working.

***Part C: Combining Hardware and Software Mechatronic Elements***

In this part we interface the hardware elements using software. What we try to do is to interface a given remote to the BOE-Bot and perform custom actions like moving forward, rotating in circle, etc.

**Procedure:**

1. The BOE-Bot was checked for assembly and all the mechanical parts were put intact.
2. Sample program with different inputs for the servos were run and the servos were checked for proper functioning.
3. The principal concept of different rates of rotation of the servos caused different movements was used.
4. The Mode 1 consists of interfacing the mechanical device to control the BOE-Bot. In this lab it was an IR remote control. The Remote control has a specified code for each button pressed that was available from the web.
5. An IR receiver was mounted on the BOE-Bot. The receiver receives the signal from the remote and sends it to the Microprocessor after decoding.
6. The decoding of the IR signal received is done based on the code of the Remote control.
7. The program for decoding eliminates the header and sends the information received to the microprocessor.

8. The Mode2 was made of the following actions are performed:
  - a. Moving Straight Forward  
In this motion the 2 servos rotate in same direction with same velocity
  - b. Moving Backward  
In this motion the 2 servos rotate in the opposite wise to that of the moving forward mode
  - c. Turn Right  
In this motion one wheel is fixed with no velocity and the other wheel rotates.
  - d. Turn Left  
In this motion the wheel fixed is the other one than the one in spin right
  - e. Spin Right  
In this motion the wheel rotate in the opposite direction at the same rate that causes spin effect on its own axis.
  - f. Spin Left  
The direction of the rotation of the wheels are opposite in sense to that performed in the spin right that causes spin in the right direction.
  - g. Drive in Circle  
In this motion the difference in velocity of the 2 wheels was calculated by analysis and was calibrated to drive the BOE-Bot in circle.
9. Program is made such that each button on the remote control commands a motion specified in the Mode 2.
10. The controls are assigned in the program to increase and decrease the step size.
11. The step size can be selected only for the linear and turning operations, but for the circle the step size is fixed to make a complete circle.

### **Analysis:**

#### Linear Motion:

##### *Moving forward and backward*

The wheels of the BOE-Bot are mounted on the servos. The servos are fixed on the BOE-Bot in such a way that they face away from eachother. This setup causes opposite rotation of the wheels when same input is given to both the servos. Thus for causing linear motion, one wheel is made to rotate in clockwise and the other in anti-clockwise direction with the same rpm.

The input of same rpm is another issue that was resolved by calibration. The rotation in rev/sec corresponding to a particular pulse width was noted down for each wheel. The calibration chart was made out of the observation and which gave the

calibration equation. Using the calibration equation for a particular rpm, pulse width required was found which was input to the BOE-Bot servo.

### Rotational Motion:

#### *Turning right and left, Spin right and left*

The principle of difference in rotation of the wheels cause rotation was used. For turning right and left one wheel was stopped while the other wheel rotated forward which caused the BOE-Bot to rotate on the stationary wheels axis that caused rotation. For the turn in the other direction the other wheel is made stationary and the spin is on the forward direction that caused turn.

For spin on it own axis, both the wheels were turned at same velocity, one in forward and the other in backward direction. When the wheels spin in the direction opposite to the previous mode, the spin is in the other direction.

Based on which direction we look at the BOE-Bot, the left and right wheel are denoted. For turning left the left wheel is made stationary and vice versa. And the higher the velocity the faster it turns. The degree of turn can be varied by the option provided by the remote.

#### *Moving in Circle*

Moving in a circle follows the basic principle of the difference in rotation of the wheel. The wheels are denoted as inner wheel, nearer to the center of the circle, and the outer wheel, away from the center of the circle. The program is made for the circle with radius 5 and 7 inches. The ratio of the velocity of the wheels is calculated and the calculated factor is multiplied to get the velocity of the other wheel provided velocity of one wheel.

Calculating the ratio of rotation:

The velocity here is represented in rpm

Let	N1	=	the inner wheel velocity
	N2	=	the outer wheel velocity
	R1	=	radius of the circle traced by the inner wheel
	r	=	radius of the wheel
	d	=	distance between wheels
	R2	=	radius of the circle traced by the outer wheel = (R1+d)
	P1	=	perimeter of the inner circle
	P2	=	perimeter of the outer circle
	T1	=	number of turns required to make a circle by inner wheel
	T2	=	number of turns required to make a circle by outer wheel

We have to find N2, provided N1 and R1.

For the BOE-Bot to be in circle the time of rotation must be equal for both the wheel.

Thus time taken for the inner wheel to cover P1= time taken for the outer wheel to cover P2

Thus,

$$P1/N1=P2/N2$$

representing P1 and P2 in terms of R1 and r, we get

$$T1 = R1/r$$

$$T2 = (R1+d)/r$$

and

$$P1 = T1*2*\pi*R1$$

$$P2 = T2*2*\pi*R2$$

We get,

$$N1/N2 = (R1+d)/R1$$

This ratio is used to obtain the N2 given N1.

After obtaining the N1 and N2 the calibration equation is used to convert them to the pulse width and applied to the wheel appropriately.

Perfect circle wasn't obtained, as there were errors in the output of the servos and the mounting of the wheel. This caused an oscillating circle traced by the BOE-Bot.

The above mentioned was an Engineering approach to solving the problem. But when we had problems with that approach, and we figured out that, in fact, all groups had problem such as this, we adopted a trial and error approach. We fixed the rpm of one of the wheels and kept changing the rpm of other till we got the required radius. We then hard coded the circle code and at this point of time we learnt that all that we did to calibrate the servos and arrive at a linear fit, was in vein.

## FLOW-CHART:

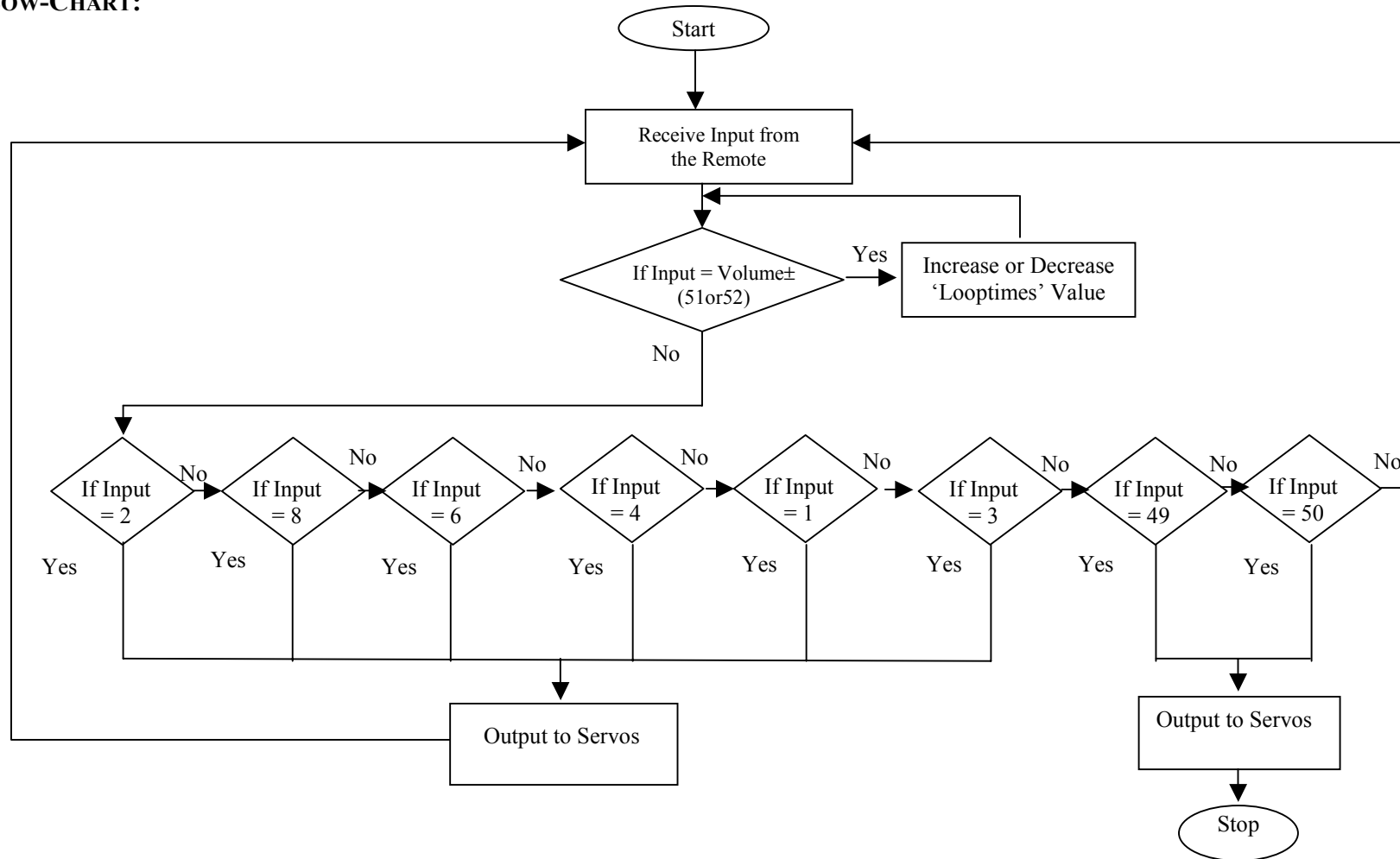


Figure 9: Flow Chart for the working of the program

## SOURCE CODE:

```

=====
'{$STAMP BS2}
'File name: Lab3groupE.bs2
'This is a program used to control the Boe Bot. This is the source
'code for controlling the Boe Bot to go different directions and also
'turning a circle of fixed radius using the Sony remote.
'Copyright by: Group E. SUNY at Buffalo
'Members: Rajankumar, Krishnakumar, Leng Feng
'Date: 11 April 2002.
=====

'-----Call the subroutine to use reset switch-----
'
'         as and stop of the mobile robot
'-----
gosub boot_subroutine

'-----Define Constants-----
LeftWheel   CON    13           'Pin 13 used to control left wheel
RightWheel  CON    15           'Pin 15 used to control right wheel
forLeft CON    858           'Forward motion of Left wheel
forRight   CON    622           'Forward motion of Right wheel
backLeft   CON    612           'Backward motion of Left wheel
backRight  CON    820           'Backward motion of Right wheel
stopall CON    750           'Stops the servo motor
ir         CON    10           'Pin 10 used to get data from remote

'-----Define Variable-----
p1         var    word          'Store the first pulse
p2         var    word          'Store the second pulse
p3         var    word          'Store the third pulse
p4         var    word          'Store the fourth pulse
p5         var    word          'Store the fifth pulse
p6         var    word          'Store the sixth pulse
Remote var    word          'Store the whole decimal number
looptimes var    byte         'Variable for increasing pseudo-speed
counter var    byte         'Counter for circle
counter1   var    byte         'Counter for circlce

'-----Initialize-----
LOW RightWheel 'Initialize left wheel
LOW LeftWheel  'Initialize right wheel
looptimes=10   'Initialize pusedo-speed
Remote=0       'Initialize Remote

'-----Main Control Loop-----
Main:
    GOSUB Pollir 'Wait for command
    Goto Main    'Start again

'-----Subroutines-----
'
' Below are subroutines that control the movement of the Boe Bot
' Each subroutines use to control the wheels rotations of the Boe Bot
'
'-----
turnLeft: 'take a left turn
for counter = 0 to looptimes
    pulsout 15,stopall
    pulsout 13,forLeft
    pause 20
next
goto Main

turnRight: 'take a right turn
for counter = 0 to looptimes
    pulsout 15,forRight
    pulsout 13,stopall
    pause 20
next
goto Main

spinLeft: 'spin left about own axis
for counter = 0 to looptimes
    pulsout 15,backLeft
    pulsout 13,forRight
    pause 20

```



```

end                                     ' main routine.
                                          ' Stops program execution and places
                                          ' BASIC Stamp 2 in low power mode.

Pollir:
pulsin ir,0,p1                          'Wait for pulse
if (p1 < 1000) then Main                 'Check is pulse is great then 1000
pause 25

pulsin ir,0,p1                          'Read the head
pulsin ir,0,p1                          'Destroy the head store first pulse
pulsin ir,0,p2                          'Read second pulse
pulsin ir,0,p3                          'Read third pulse
pulsin ir,0,p4                          'Read fourth pulse
pulsin ir,0,p5                          'Read fifth pulse
pulsin ir,0,p6                          'Read sixth pulse and forget about
the rest.

' *****
' * This subroutine will convert the pulse from the remote and convert *
' * it to a decimal number and then call the subroutine according to *
' * the signal read in but the 38k IR receiver. *
' *****

Remote = 1 + (p1/450)                   'Convert the first pulse to binary and add 1
Remote = 2*(p2/450) + Remote            'Con P2 to bin then shift left 1 and add to remote
Remote = 4*(p3/450) + Remote            'Con P3 to bin then shift left 2 and add to remote
Remote = 8*(p4/450) + Remote            'Con P4 to bin then shift left 3 and add to remote
Remote = 16*(p5/450) + Remote           'Con P5 to bin then shift left 4 and add to remote
Remote = 32*(p6/450) + Remote           'Con P6 to bin then shift left 5 and add to remote

'debug ? Remote

    if Remote = 2 then forward           'if 2 on the remote
    if Remote = 8 then backward          'if 8 on the remote
    if Remote = 6 then spinRight         'if 6 on the remote
    if Remote = 4 then spinLeft          'if 4 on the remote
    if Remote = 1 then turnRight         'if 1 on the remote
    if Remote = 3 then turnLeft          'if 3 on the remote
    if Remote = 51 then incrloop         'if VOL+ on the remote
    if Remote = 52 then decrloop         'if VOL- on the remote
    if Remote = 49 then circle1          'if CH+ on the remote
    if Remote = 50 then circle2          'if CH- on the remote
return                                   'go to the calling subroutine

```

## DISCUSSIONS:

In this experiment, we face some problems; most of the problems come from the hardware that we are using. The problem happened in our servo motor have taken a lot of our time while doing this experiment. We spent quite an amount of time to figure out how to adjust the servomotor. We get some idea from other group to modify the servo motor by adjusting its potentiometer. However, this does not work for our group. Finally, a new servomotor is given to us and everything works well. In controlling the mobile robot to turn in a desired radius of circle, we also face some problem. We try to control the mobile robot so that it can move in a perfect circle; that is, it will go back to its original position after a circle is completed. However, the mobile robot does not work out as we desired. It turns in circle but does not go back to its initial position, we try to solve this problem in software and try to calibrate it by try an error but it still did not go backs to its initial position. From an engineering approach, we try to find out the problem from our calibration curve and we found out that if we want to turn a circle, we need to fix one rotation speed of one wheel and calibrated the rotation speed of another wheel. We found that the rotation speed of the second will fall in a region that is not in the linear portion of our calibration curve. As a result, we have to do it by try an error to find out the pulse period that gives the desired rotation speed of the wheel. We successful accomplished

this part and figure out the speed required for the wheel for a turning a circle of radius 5 inch and 7 inch.

While tracing the path followed by the mobile robot, we attached a pen on the mobile robot and let it draw its paths when it is moving, we finally found that the paths that it follows does not shows any tendency towards a certain direction but in fact the error is randomly occurs. This suggests that this error does not come from the control part of the system itself but come from the hardware. Dr Krovi had suggested that this error might come from the slipping effect of the front wheel that we cannot control. we thus made a conclusion that this is the limitation of the mobile robot of just using two wheels. However, to reach to this conclusion, we have spend some amount of time that we could have use to figure out other interesting part of this experiment.

#### **EVALUATION OF THE PERFORMANCE:**

We have successfully complete the required task of this experiment. We interface the SONY RM-V201 universal remote control with our mobile robot and control is motions. We have implement 6 different basic motions to the mobile robot, which are forward, backward, turn left using one wheel, turn right using one wheel, turn left by cooperative of both wheels, and turn right by cooperative of both wheels. This six basic motion are control by the number button on the SONY RM-V201 universal remote control. These different motion can be use to accomplish different motion for different surface condition and achieved the desired location for moving the mobile robot. We also successfully control the mobile robot to turn in a circle of radius of 5 inch and 7 inch. We initially wants to control the radius of the mobile robot by getting input from the user using the remote control, however, because of the calibration problems that we face in controlling the two wheels and the slipping problem cause by the front circle wheel, we do not have sufficient time to accomplish our initial plan. Also because of this problem, we did not have time to accomplish the challenging part of this experiment. That is to control the mobile robot to move to a specific X and Y locations. However, our mobile robot have a advantage over other group such that we can control the loop time for each turning of the wheel such that the “chucking” effect of the mobile robot can be reduced, with the lost of larger time pulse between each command that we can gave to the mobile robot. This part is done by writing a command loop such that when pressing a button, we can increase the loop time in each subroutine.

**COMPONENTS USE IN THIS LAB AND ITS PRICE:**

	<b>Components</b>	<b>Price</b>	<b>Source</b>
1	BOE bot Kits	\$186.15	<a href="http://www.parrallaxinc.com">www.parrallaxinc.com</a>
2	SONY RM-V201	\$9.90	<a href="http://www.sel.sony.com">www.sel.sony.com</a>
3	Computers	Free	System laboratory, ARM lab
4	IS1U60 IR receiver	\$3.25	<a href="http://www.hvwtech.com/sensors.htm">www.hvwtech.com/sensors.htm</a>
5	4 AA size batteries	Free	Provided by Group B.

**CONTRIBUTION OF EACH MEMBER:**

1	Lee Leng Feng	Task 1 & 3, Programming, report writing.
2	Rajankumar Bhatt	Task 1 & 2 & 3, programming, report writing, assembly.
3	Krishnakumar A Ramamoorthy	Task 1 & 2, programming, report writing.
Calibration and testing was done by all as a group		

**CONCLUSION:**

With the cooperation of each group member; we successfully complete the give task in this experiment. We have successfully given a minimum of 8 motions to the mobile robot, with also implemented a control button for the changing the loop time interval for each motion. From this experiment, we have learned the way to interface a remote control to control the motion of the mobile robot. With this basic idea, we can further implement this idea that we could built such a mobile robot that is operate on the Mars and control its motion from the earth. This of course, required more reliable data transmission mode.

Beside this, in this experiment, we have explored the limitation of this mobile robot, especially in the hardware of the kits provided. We have learned that every hardware has its limitations. Although the theory that we formulated may works correctly in an ideal world, but in reality, the limitation of the hardware is also an important issue that we need to explore and find out the limitation of each component that we are using. This is also an important aspect of this course. By doing this experiment, it provide us some important idea and also some exploration in controlling the mobile robot, these experience will greatly be helpful in doing our final project.

**REFERENCES:**

- [1]. Sony Universal remote control. RM-V201.  
<http://www.sel.sony.com/SEL/consumer/ss5/home/accessories/universalremote/rm-v201.shtml>.
- [2]. Boe Bot Kits.  
[http://www.parallaxinc.com/html\\_files/products/StampsIC/boe-bot\\_brief.asp](http://www.parallaxinc.com/html_files/products/StampsIC/boe-bot_brief.asp)
- [3]. Boe bot Kits reference.  
[http://www.stampsinclass.com/html\\_files/sic\\_curr/curriculum\\_robo.asp](http://www.stampsinclass.com/html_files/sic_curr/curriculum_robo.asp)
- [4]. Robotics! Student workbook. Version 1.5. Parallaxinc educational materials. Or  
<http://www.stampsinclass.com/downloads/Robo/rob.pdf>
- [5]. Sharp IS1U60 Data Sheet.  
<http://www.hvwtech.com/dnload/datasheets/is1u60.pdf>
- [6]. IR code for Sony remote  
<http://cgl.bu.edu/GC/shammi/ir/>