# MAE 576 Final Project- Interfacing Two Basic Stamps
A Compact Disk Sorter with a Central Database

Humbly Submitted by:
Matt Szymanski
Nicholas Gill

On:
May Fifteenth, In the year of our Lord two thousand and three.

Table of Contents

## Abstract

Micro-controllers can be used to control specific tasks. When many tasks are involved, the total system needs the ability to communicate between micro-controllers. This assignment is to interface a mobile robot (BOEbot) with a base station. There are several methods of communication: wired, infrared (IR), and radio frequency (RF). This project utilizes the BOEbot as a CD sorter that runs along a track. The base station sends the specific CD for the BOEbot to retrieve and reports the current location of the BOEbot on a LCD. The CD sorter links the micro-controllers via wired serial communication and IR serial communication. The hardware used and a price for the individual parts is listed. The individual tasks to build this system show how the CD sorter can be replicated. The software and circuit diagrams explain the program flow. The discussion provides and analysis of the design and future recommendations.

## Introduction

Robots can simplify tasks that require sporadic human intervention. One of the options for a CD burn is to compile several songs from different CDs. This can be a tedious process, because the data has to be extracted from each CD individually and the user has to be present to switch the CD. Another option is to burn several copies of the same CD. This project is a representation of devise that would remove the CD from the computer tray and exchange the CD with the next CD needed. During the exchanging process, the BOEbot reports the status back to the base station. Using relatively simple tactile switch techniques, the robot can accurately sense, and more importantly, interact with the environment. This same setup can be used in a biology laboratory and take digital pictures, add drugs to samples, and report statistical information such as temperature at the command of the main base.

The project can be divided into five smaller projects: construction of the physical structure, communication of the BOEbot and the Base Station, interfacing the LCD screen, construction of the CD grabber, and movement of the BOEbot on the track. Each task has hardware involved and except for the construction of the structure, all the tasks had software involved. During the design phase, the tolerances of the individual subsystems must be considered. Once all the tasks are completed, the project can be assembled and integrated into one main unit.

## Hardware Used

Below is a table of the hardware used, quantity and approximate cost.

| # | Component | Price | Quantity | total |
|---|---|---|---|---|
| 1a | StampWorks Full kit, Basic Stamp II | 349.99 | 1 | 349.99 |
| | 10 k resisters | | 3 | |
| | Wire | | 3' | |
| 1b | Hitachi LCD screen | | 1 | |
| 1c | Servo Motor, unmodified | | 1 | |
| 2 | Firestick II IR kit | 40 | 1 | 40 |
| 3 | Cable for Serial Communication | 5 | 1 | 5 |
| 4 | BOEbot Kit | 229.99 | 1 | 229.99 |
| 5 | 8" x 8" sheet of Styrene (for CD trays) | 1 | 8 | 8 |
| 6 | 24" x 10" x 1" Blue Ren | 40 | 2 | 80 |
| 7 | 1/2" sheet of particle board | 10 | 1 | 10 |
| 8 | Rainbow switch- PS 12-f03 | .50 | 2 | 1 |
| 9 | Misc. Fasteners | 5 | 1 | 5 |
| | Grand Total | = | | 728.98 |

1. StampWorks full kit, Basic Stamp II

The full kit contains more parts than needed for the final project, but this ensures any application change can be done readily. For example, resister values can be changed easily. There is plenty of wire, the proper tools, a multi-meter, AC adapter, stepper motor, servo-motor, 7 segment display, LCD, etc.[1]

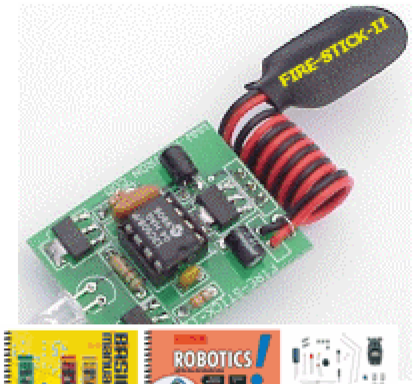
A cheaper version for this project would include only the items listed in the first item of this category. A breadboard, such as the Board of Education would also be needed.

A. The basic stamp II is programmed in PBasic. It can handle up to 600 line of instruction. There are 16 I/0 ports. The execution speed is 4000 lines/sec. It has 26 bytes of RAM and 2k bytes of EEPROM.

B. The Hitachi LCD screen, pictured above connected to the StampWorks, 2 two lines by 16 characters. It interfaces with the basic stamp though the HD44780 controller. The controller has CGRAM space as external memory.

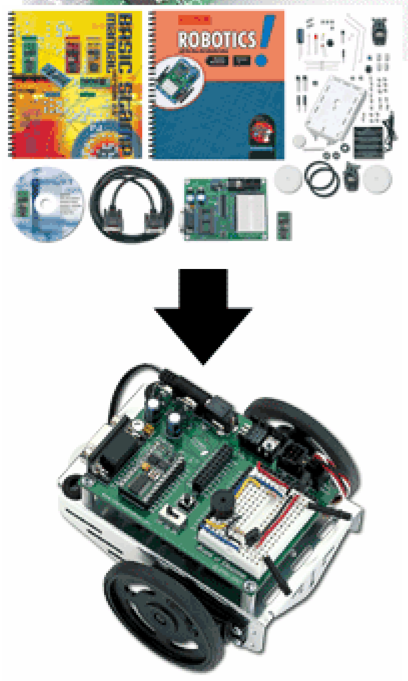
C. The Parallax Standard Servo (900-00005) is not modified and therefore it has two distinct stroke limits.

2. Firestick II IR kit- The irststick is an IR emitter. It accepts serial data though the SERIN command. It will accept a 9-volt battery and regulate the voltage if a regulated voltage is not available.



3. The cable for the communication between the two basic stamps is three 22 gage wires with an outer PVC cover.

4. BOEbot kit - The BOEbot kit is another product produced by Parallax. Again, this part of the cost could be cheaper, but it is much easier in the development of the project to purchase the entire kit. The parts used were the wheels, chassis, Board of Education, battery box, ball wheel, two modified servos, the basic stamp, and misc. hardware.[2] The modified servos will spin a full 360 degrees.



5. The styrene sheets were for the CD trays. They were chosen because they can be easily vacuum-formed.

6. The blue Ren is used because it can easily be machined. It does not split and it resists warpage much better than traditional wood. For long term use, it should be sealed to avoid flaking.

7. The particleboard was used to mount the equipment and keep it in place. This facilitates shipment, and ensures a reliable work environment. Ren was not used for the base, because it was heavy and expensive.

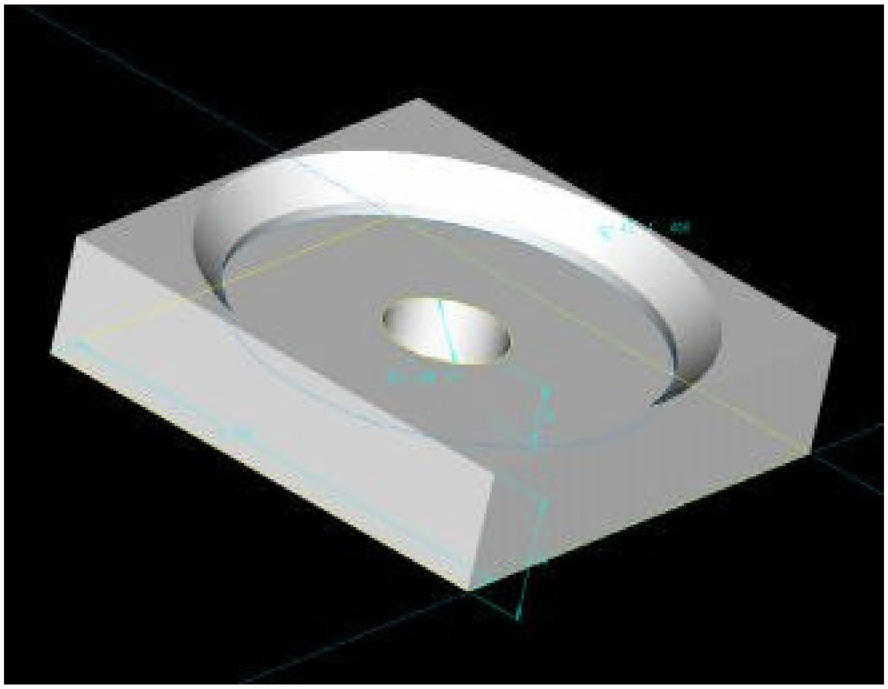
8. Rainbow Switch- PS12-f03- This tactile push switch has .100" throw and activation after .050." It is rounded on both sides of the trigger, and will cam smoothly.

[1] For a complete listing, see: http://www.parallax.com/detail.asp?product_id=27297
[2] For a total listing see: http://www.parallax.com/detail.asp?product_id=28132

## Tasks and Procedures

The project can be divided into five tasks: construction of the physical structure, movement of the BOEbot on the track, construction of the CD grabber, communication of the BOEbot and the Base Station, and interfacing the LCD screen.

1. Construction of the Physical structure:
   - A. The CD trays were formed by vacuum forming sheet styrene over a buck.

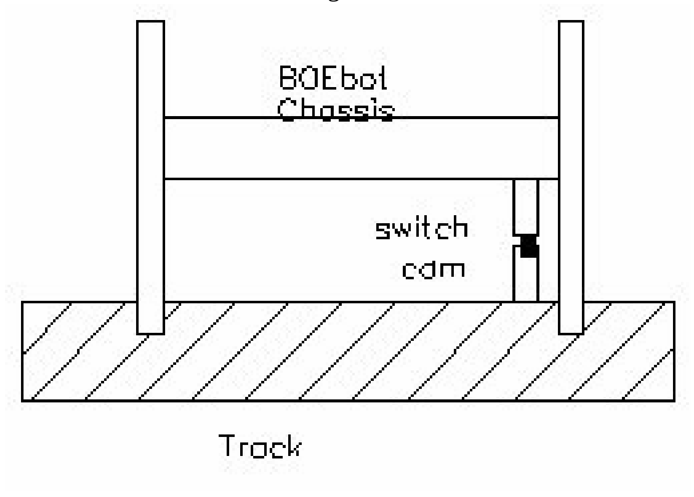


Above is a 3D model of the buck. Note that the buck is always a wall thickness smaller than the finished part. The vacuum-forming process heats a styrene sheet until it is soft and pliable. The sheet is lowered on top of the buck and is sucked onto the buck from a vacuum below. There is some variation in size due to the vacuum-forming process. There is around .100" variation in the CD tray diameter, and the BOEbot must be able to handle this variation.

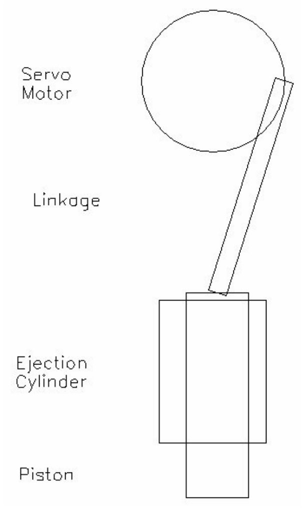
   - B. The track grooves were done in Blue Ren on a mill. Two sections were used because the length limitation of the mill. A ball bit was used for the circular groove for the rear wheel.
   - C. A ribbed box was made for the StampWorks board. This ensures the IR will be consistent.
   - D. All parts were fastened to ½" particleboard.
2. Movement of the BOEbot on the track.

A. The BOEbot was assembled according to the directions provided with the kit. It was decided to avoid modifications to the BOEbot because it was on loan.

B. The BOEbot travels along the track in the rails. As the BOEbot travels along the rails, cams activate the switch. A button counting sub- routine counts the number of times the button is pressed. The position of the BOEbot is known by counting the activation of the button. There is a home position that a second switch (not shown) uses to detect the location of the CD drawer. The BOEbot always returns to home to avoid stack-up of errors.



3. The CD grabber utilizes a servo motor and piston cylinder. There are three positions for the CD. The Servo is at its maximum stroke, a rubber tip on the bottom of the piton is pushed into the hole in the center of the CD. This position is held for two seconds. The chamfered tip of the rubber slowly slides into the CD hole. The CD is lifted to clear the height of the CD trays. The final position ejects the CD by stripping the CD off the rubber as the tip and is pulled into the ejection cylinder. The length of the cylinder is longer compared to the piston's diameter. This will ensure there will be a minimal amount of binding between the piston and the cylinder. The stroke of the piston can be lengthened by modifying the rotation of the servomotor, and by changing the radius on the effort arm on the servomotor. Changing the radius was a gross adjustment, while the stroke of the servo was a fine adjustment accomplished in software.



4. Communication of the BOEbot and the base station. There are two forms of communication used in this project: IR and wired.

A. Infra Red communication was accomplished using the Firestick II kit. Serial Information from the base station was input into the Firestick and it transferred the data to the collector on the BOEbot.

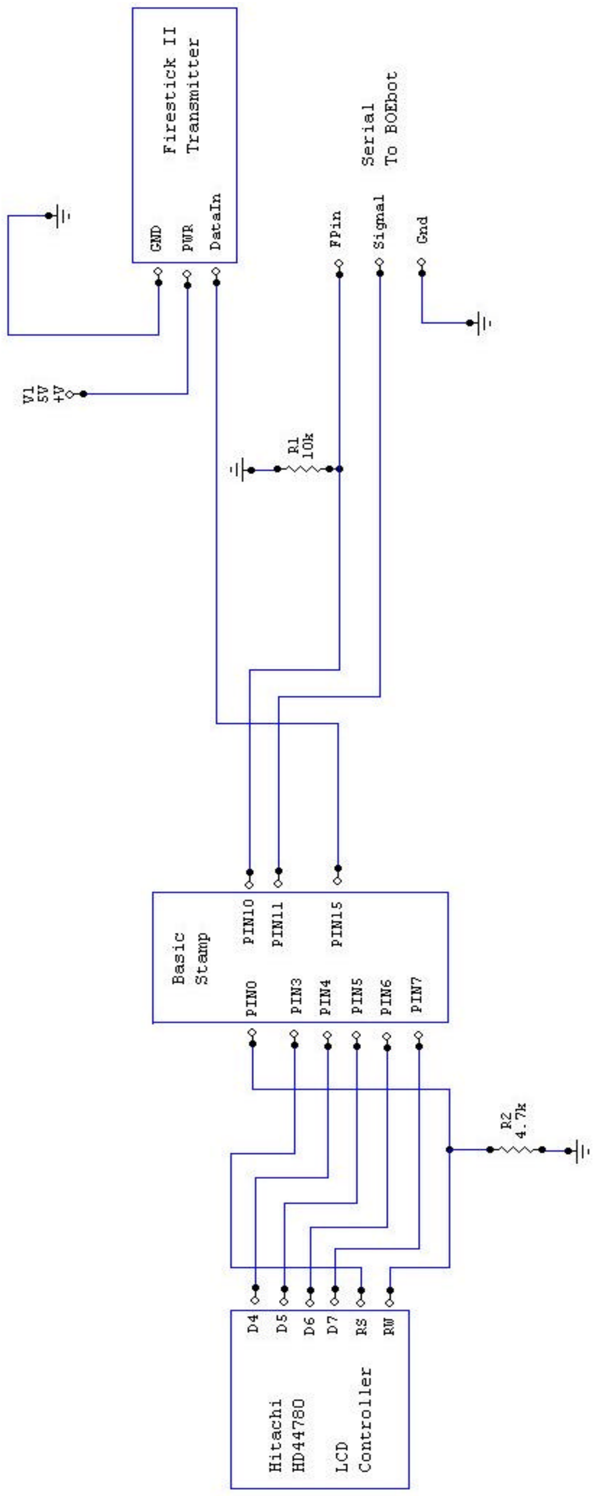
To be sure a line of sight was always present, communication was only done while the BOEbot was at the home position. To debug the transmission, a program was loaded into a Palm Pilot that detects IR. This helped point the problem of unsuccessful transmissions to either the emitter or collector. To ensure the serial information received was received properly, a wait command was used in SERIN. It waited for the letter "A" before accepting transmission.

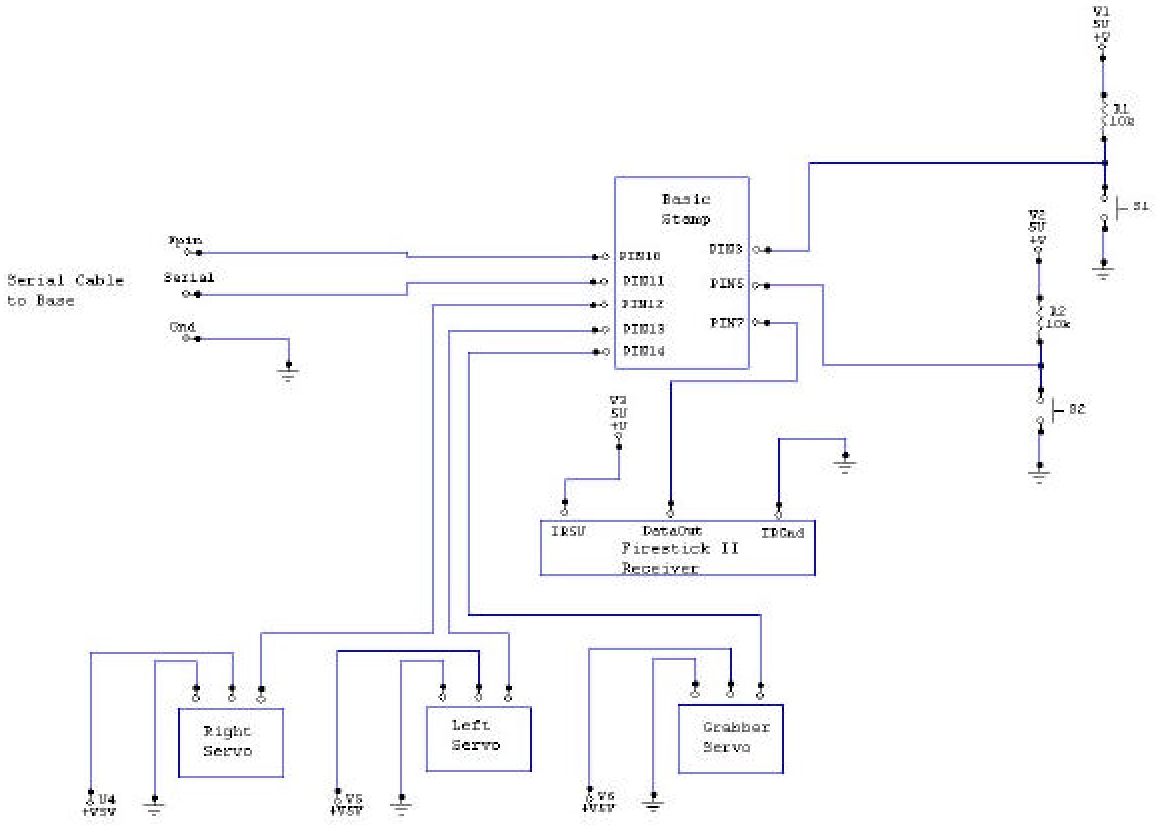
B. Communication from the BOEbot to the Base Station used wired serial communication. In an ideal situation, only one wired would be needed. To ensure the communication is robust three wires are used. One wire transmits the data. The grounds on the two units are connected because the voltage supplies are different, and therefore the grounds can be different. If the grounds vary dramatically, they can cross the 1.5 V threshold and the data will be incorrect. The final wire is for a follower pin. This pin tells the main sender that it is ready for transmission. SERIN and SEROUT commands facilitate the actual data transfer.

5. The LCD screen should be connected to the Basic Stamp according to the manufacture's requirements. There are four data pins needed and two transmission pins needed for this experiment. The LCD must be initialized in software before data can be sent in the program.

## Circuit Diagrams

### Base Circuit





**BOEbot Circuit**

## Software Development

The software for our compact disc loader is based on the idea of dividing tasks between the micro-controller on the BOEbot and the micro-controller on the base. Effective communication and scheduling are key to the software design.

The primary task of the Base is to tell the BOEbot what CD to get, and to run the LCD display to output the system status to the environment. The primary function of the BOEbot is to physically get the CD and know where to put it back and where to get the next one. The software operation of the BOEbot and Base station is largely repetitive and is based on a core set of subroutines which are called in a particular sequence by the main program on each micro-controller. The sequence for both micro-controllers is broken into four main parts: initialization, the main sequence, the sequence for the first CD, and the sequence for the last CD.

The initialization sequence for the BOEbot allows the robot to "know" where it is by finding it's home position. Once at it's home position, the BOEbot signals the bas e to let it know it is ready to execute instructions. The initialization sequence for the Base clears the display, clears the memory for the CD number, and loads the text for the LCD into the EEPROM. It then waits for the message from the BOEbot that it is at home.

The main sequence for each picking up and replacing CDs has the base tell the BOEbot which CD to retrieve, the BOEbot replaces the CD currently in the drive, and then retrieves the CD and reports it's status. Between each operation, the BOEbot returns home to ensure that it is still correctly calibrated and to facilitate communication scheduling. To take care of scheduling and to make sure that either end does not miss a message, the base and the BOEbot only communicate while the BOEbot is at home. The BOEbot only sends it's status while at the home position and only can receive instructions from the base while at the home position.

The sequence for picking up the first CD is unique because the BOEbot doesn't have to put a CD back before picking up the CD requested by the base. In the same way, the sequence for the last CD is unique because the BOEbot must put the last CD back and then report back to the home position and tell the base there are no more CDs to move.

To determine the order in which the CD's are fetched by the BOEbot, the base calls a subroutine which sets the CD number the BOEbot should pickup. For simplicity, the source code used for demonstration purposes simply calls the CDs in number order.

```
CD_orderer:                   'Determine the order to get the CDs
CDorder= CDorder + 1          'Get the CDs in numerical order
```

The code is designed so that the CD_orderer subroutine can be replaced with code used to determine any other order for the CDs. As long as the subroutine sets the value for the variable "Cdorder", the program will continue to run.

The additional subroutines in the software for the base take care of displaying the proper messages on the LCD. The BOEbot communicates it's status by setting a number value to the variable "status." The base then cross references the number value with the appropriate text that should be displayed on the LCD and then calls subroutines to actually display the text on the screen.

For the communication from the Base to the BOEbot, the SEROUT command is used to interface with the Firestick II Infrared Transmitter. The (wait "A") modifer is added to provide flow control and to add "encryption" to ensure the data is received by the correct robot.

<u>Base</u>

```
Send_CD:
SEROUT 15, 17197, ["A", CDorder]          'Send the letter A and then the CD number
```

<u>BOEbot</u>

```
Listen:
SERIN 3,813,[wait("A"),CDnumber]             ' Wait for ASCII letter A and then get data
```

The main function of most of the subroutines in the software for the BOEbot is to accurately control the servomotors. For the modified servomotors that run both wheels, the zero motion point for each servo was found. For forward motion (which is defined as moving away from the home position and towards the base) the desired speed of the motor is subtracted from the 0 motion value for the right motor, and added to the 0 motion value for the Left motor.

```
Move_Forward:
PULSOUT RightMotor, (757 - PulseSpeed)              'Move right motor forward
PULSOUT LeftMotor, (760 + PulseSpeed)               'Move left motor forward
Pause 25
```

```
BUTTON ButtonPin, 0, 255, 10, swData, 1, Increment   'count the bumps and pickup CD
GOTO Move_Forward                                    'repeat until we hit the correct bump
```

For reverse motion, the signs are flipped.

```
BackHome:
PULSOUT RightMotor, (757 + PulseSpeed)               'Move right motor forward
PULSOUT LeftMotor, (760 - PulseSpeed)                'Move left motor forward
Pause 10
BUTTON ButtonHomePin, 0, 255, 10, swData, 1, Drop_CD 'sense when we are home
GOTO BackHome
```

For the unmodified servo motor which controls the CD grabber, the subroutines are set up to define a position for the motor to move to, and then provide an adequate number of pulses for the motor to fully move to that new position.

```
Drop_CD:
FOR PulseCycle = 0 TO 50                             'give servo time to move
PULSOUT GrabberPin, 300                              'set drop position of servo
PAUSE 10
NEXT
Return
```

The only difference between picking up, carrying, and dropping a CD is in the position of the servo motor.

For the communication from the BOEbot to the Base, a wire was used along with serial communication with flow control using the Fpin modifier in the SEROUT command.  This allows us to know when the other processor is ready to send and because a wire is used, "encryption" or some kind of identifying header is not needed.

BOEbot

```
Send_Data:
SEROUT 11\10,16468,[status]           'send the data
```

Base

```
Listen:
SERIN 11\10,16468, [wait("A"), status]          ' Wait for letter A then send value
```

## Discussion

With Mechatronics being the interaction of mechanical systems with electrical systems, the implementation of this project brought to light the interdependency between the two systems. Many situations came up which demonstrated how one system affects the other.

There were many aspects of this project that worked out well while constructing the different elements of the system. Since a CD loader only needs to operate in a linear fashion, it made sense to put he BOEbot into a track. By constraining the robot in such a way, the calibration of the servomotors for the wheels became critical. By changing the values sent to the servomotors by the mircocontroller, we were able to fairly accurately determine the zero motion point for each of the modified servos. We incremented the output until the motor started to turn and then repeated the process for the opposite direction. By taking half the difference in the input values, the zero point from which to add or subtract the speed control value was determined. This method worked very well and produced good, repeatable straight tracking for the BOEbot. The alternative would have been to utilize a live axle driven by one servomotor, but that option was not needed.

Another aspect of the mechanical, electrical interaction the worked well was the use of the cams on the track to activate the switches. The cams could be easily filed and adjusted to correctly position the robot. Once a good speed was determined, the behavior of the robot as it moved over the cams was very reliable.

The execution of the final program also turned out to be much simpler than previous versions that we had developed. We originally had programmed the software so that the different subroutines could call each other without being sequenced in the main program. We soon found that it was difficult to plan the logic and inefficient from a programming standpoint to have the subroutines calling each other. By making the subroutine mostly self sufficient, each subroutine could be called from the main program. Thus, the program logic was laid out in the main program, with the subroutines called to perform their tasks and then return to the main program. This provided good flow and an efficient design.

Along with the systems that worked well, there were also many areas that were troublesome, some of which were removed from the system. One example is we originally tried to implement a more robust error checking algorithm for the communication between the two processors. We had intended on having processor A send a message, processor B, send back the same message, processor A compare the original with the one received, and if they were the same, send the message again to be executed by processor B. It turned out to be too difficult to get the timing correct and we ended up abandoning the idea.

We had also set out to have the BOEbot travel at a higher rate of speed. When stream ling our code and removing unnecessary "debug" commands the speed of the robot increased from that which we were originally testing. As the speed increased, the robots ability to stop at the correct location became much worse. We ended up adding pauses and in some cases leaving in un-needed debug commands in order to get the speed back to where the robot was accurate.

Given the present system, and observing it's function, there are many things that can be done to improve our system and make it much more robust. One feature that would be nice to implement in the future is some sort of subsystem to check if the CD was actually picked up when the grabber cycled down to get the CD. A photo detector or a simple leaf switch on the outer circumference of the CD would work well for this task. If the leaf switch was not activated after the grabber lifted, then the robot could return to home and then try to obtain the correct position again.

It would also be a beneficial addition to add some sort of sensing to determine the open or closed status of an actual computer CD drive. Our setup, created for demonstration purposes, accurately shows how CDs could be placed in a certain position and removed in a specified order. To make the system usable in the real world, it would be nice to have the BOEbot sense an open CD drawer, remove the CD, and replace it with the next one. Furthermore, serial communication between the BOEbot and computer could further enhance the system. By allowing the computer to specify the order in which to load the CDs, and the BOEbot send a command back to the computer telling it to close the open CD drawer the system would be 100% automated.

This system is also far from efficient. If the BOEbot did not have to travel back to the home position between every operation, much time could be saved in the loading and unloading of the CDs. The efficiency could also be improved by having the BOEbot reorder the CD's in the trays, moving the finished CD's to the end of the track, and the CD's not operated on to the beginning of the track, all while the disk drive was busy with it's current disc. This way, the next CD to be loaded would be nearest to the drawer.

Overall, the CD loader system as a whole works well and is a good demonstration of distributed tasking between two micro -controllers. Our system provides a good starting point for the development of a commercially viable product.

## Conclusion

Many times a system will warrant the use the two micro-controllers. The CD sorter utilized two controllers by using one controller to manipulate the CD's and the other controller to display the current condition on the LCD. The communication was done by serial communication though IR and wired connection. Both forms has error checking. The IR used a wait command, and the wired used a flow control wire.

Several items could be explored further. It would be convenient to have a manual mode to control to BOEbot. An eject button that senses when the door of the CD-ROM is open would allow the project to be used for burning multiple CD's. The BOEbot returns home for secure data transmission. Perhaps the IR communication could be eliminated and only wired would be used.

This design could be used in other applications where a main base sends commands to a mobile robot that interacts with only precise discrete positions. One example of this would be a medical laboratory.

## Appendix

```
' ===============================================================================
'
'{$PORT COM2}
'   File...... Final Project-BOEbot.BS2
'   Purpose... Automatic CD Loader- BOEbot
'   Author.... Nick Gill and Matt Szymanski
'   Started... 19 APRIL 2003
'   Updated... 01 MAY 2002
'
'   {$STAMP BS2}
'
' ===============================================================================
' -------------------------------------------------------------------------------
' Automatic CD Loader- BOEbot
' -------------------------------------------------------------------------------
' Constants
' -------------------------------------------------------------------------------

ButtonHomePin         CON     5              'Home button connected t pin 5
ButtonPin             CON      7             'CD button set to pin 7
RightMotor            CON    12              'Right servo connected to pin 12
LeftMotor             CON    13              'Left servo connected to pin 13
GrabberPin            CON    14              'Grabber servo connected to pin 14

' -------------------------------------------------------------------------------
' Variables
' -------------------------------------------------------------------------------

PulseCycle            VAR    word            'Hold value of pulse for servo

CDNumber              VAR    word            'Hold which CD to move

PulseSpeed            VAR    word            'set the servo speed

swData                VAR    byte            'Workspace for Button
ButtonCounter         VAR    word            'count the activations

status                VAR    word            'hold the satus of the BOEbot

' -------------------------------------------------------------------------------
' Initialization
' -------------------------------------------------------------------------------

LOW RightMotor                               'Right motor pin to output-low
LOW LeftMotor                                'Left motor pin to output-low

PulseSpeed = 100                             'Speed to add to stopped value

FindHome:

PULSOUT RightMotor, (757 + PulseSpeed)       'Move right motor forward
PULSOUT LeftMotor, (760 - PulseSpeed)        'Move left motor forward
PAUSE 25
BUTTON ButtonHomePin, 0, 255, 10, swData, 1, Main   ' Debounce and execute button
GOTO FindHome                                'Repeat the process until button is pressed


' -------------------------------------------------------------------------------
' Program Code
' -------------------------------------------------------------------------------

Main:

status=0                     'set the message to- at home

GOSUB Send_Data              'send value of status to base
GOSUB Listen                  'find out which CD to get

status =1                     'set message to- Getting CD
GOSUB Send_Data               'value of status to base

ButtonCounter=0               'reset the button for next run
GOSUB Move_Forward            'physically get the CD

Pickup:                       'marker for everything but the first CD

GOSUB Get_CD                  'Pick up the CD

GOSUB Backhome                'Return to home

status = 2                    'set message to- Playng CD
GOSUB Send_Data               'send value of status to base

GOSUB Listen                  'Get new CD number
status = 3                    'set message to- Return CD

GOSUB Send_Data               'send value of status to base

GOSUB Get_CD                  'pick up the CD

CDnumber = Cdnumber - 1       'we want the previous CD location
ButtonCounter=0               'reset the button for next run
GOSUB Return_CD               'just like move forward, but CD - 1

Dropoff:                      'marker for after Return CD

GOSUB Drop_CD:                'drop returned CD on tray

GOSUB BackHome:               'Return Home

IF CDnumber > 5 THEN CD_End   'If we are done- jump to end
status = 1                    'set message to- Getting CD
GOSUB Send_Data               'send status to base

CDnumber = CDnumber + 1       'restore the "real" CD we want
ButtonCounter=0               'reset the button for next run
GOSUB Move_Forward

CD_End:                       'marker for the last CD

Pause 2000
status = 4                    'set message to All Done
GOSUB Send_Data               'send message

END

' -------------------------------------------------------------------------------
' Subroutines
' -------------------------------------------------------------------------------

BackHome:

PULSOUT RightMotor, (757 + PulseSpeed)              'Move right motor forward
PULSOUT LeftMotor, (760 - PulseSpeed)               'Move left motor forward
Pause 10
BUTTON ButtonHomePin, 0, 255, 10, swData, 1, Drop_CD 'sense when we are home
GOTO BackHome                                       'repeat until button is pressed


Move_Forward:

PULSOUT RightMotor, (757 - PulseSpeed)              'Move right motor forward
PULSOUT LeftMotor, (760 + PulseSpeed)               'Move left motor forward
Pause 25
BUTTON ButtonPin, 0, 255, 10, swData, 1, Increment   'count the bumps and pickup CD
GOTO Move_Forward                                   'repeat until we hit the correct bump
```

```
RETURN


Return_CD:

PULSOUT RightMotor, (757 - PulseSpeed)               'Move right motor forward
PULSOUT LeftMotor, (760 + PulseSpeed)                'Move left motor forward
Pause 25
BUTTON ButtonPin, 0, 255, 10, swData, 1, Increment_Two 'count the bumps and drop CD
GOTO Return_CD

RETURN


Get_CD:

FOR PulseCycle = 0 TO 50                             'give servo time to move
PULSOUT GrabberPin, 1050                             'set down position of servo
PAUSE 10
NEXT

Pause 2000

FOR PulseCycle = 0 TO 50                             'give servo time to move
PULSOUT GrabberPin, 600                              'set up position of servo
PAUSE 10
NEXT

RETURN


Drop_CD:

FOR PulseCycle = 0 TO 50                             'give servo time to move
PULSOUT GrabberPin, 300                              'set drop position of servo
PAUSE 10
NEXT

Return


Increment:
ButtonCounter = ButtonCounter + 1                    'count the buttons
IF (ButtonCounter = CDNumber) THEN Pickup            'stop at the correct CD and get CD
GOTO Move_Forward                                    'move to the next button


Increment_Two:
ButtonCounter = ButtonCounter + 1                    'count the buttons
IF (ButtonCounter = CDNumber) THEN Dropoff           'stop at the correct CD and drop CD
GOTO Return_CD                                       'move to the next button


Listen:
SERIN 3,813,[wait("A"),CDnumber]              ' Wait for ASCII letter A and then get data

RETURN


Send_Data:
SEROUT 11\10,16468,["A",status]               'send letter A and then the data

RETURN
```

```
' =============================================================================
'
'{$PORT COM1}
'   File...... Final Project-Base.BS2
'   Purpose... Automatic CD Loader- Base
'   Author.... Nick Gill and Matt Szymanski
'   Started... 19 APRIL 2003
'   Updated... 01 MAY 2002
'
'   {$STAMP BS2}
'
' =============================================================================
' -----------------------------------------------------------------------------
' Program Description
' -----------------------------------------------------------------------------
' Automatic CD Loader- Base

' -----------------------------------------------------------------------------
' I/O Definitions
' -----------------------------------------------------------------------------

E               CON     0                    ' LCD Enable pin  (1 = enabled)
RS              CON     3                    ' Register Select (1 = char)

LCDbus          VAR     OutB                  ' 4-bit LCD data out

' -----------------------------------------------------------------------------
' Constants
' -----------------------------------------------------------------------------
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM control

' -----------------------------------------------------------------------------
' Variables
' -----------------------------------------------------------------------------
time            VAR      byte                   'variables for the LCD
CDorder         VAR      byte
char            VAR     Byte                    ' character sent to LCD
addr            VAR     Byte                    ' message address
sendingcounter  VAR    Word
programloop     VAR     byte
status          VAR    Word

' -----------------------------------------------------------------------------
' EEPROM Data
' -----------------------------------------------------------------------------
Msg0           DATA   "     AT HOME    ", 0      'set the messages to be displayed
Msg1           DATA   "   GETTING CD   ", 0
Msg2           DATA   "   PLAYING CD   ", 0
Msg3           DATA   "REPLACING OLD CD", 0
Msg4           DATA   "     ALL DONE   ", 0

' -----------------------------------------------------------------------------
' Initialization
' -----------------------------------------------------------------------------

Initialize:
  DirL = %11111101                              ' setup pins for LCD

LCD_Init:                                       'Initialize the LCD
  PAUSE 500                                     'let the LCD settle
  LCDbus = %0011                                '8-bit mode
  PULSOUT E, 1
  PAUSE 5
  PULSOUT E, 1
  PULSOUT E, 1
  LCDbus = %0010                              ' 4-bit mode
  PULSOUT E, 1
  char = %00001100                            ' disp on, crsr off, blink off
  GOSUB LCD_Command
  char = %00000110                            ' inc crsr, no disp shift
  GOSUB LCD_Command


CDorder = 0                                    'reset the CD number to 0

  char = ClrLCD                                ' clear the LCD
  GOSUB LCD_Command

' -------------------------------------------------------------------------------
' Program Code
' -------------------------------------------------------------------------------

Main:

GOSUB CD_orderer              'determine which CD to get

GOSUB Listen                  'Get the status of the BOEbot

GOSUB Get_Message             'Check which message to display

GOSUB Show_Message            'Send the message to LCD

PAUSE 5000                    'wait for the fun of it
GOSUB Send_CD                 'send which CD number to get

GOSUB Listen                  'get the status of the BOEbot

GOSUB Get_Message             'Check which message to display

GOSUB Show_Message            'Send the message to LCD

Loop:                         'Marker for everything but the first CD

GOSUB Listen                  'get the status of the BOEbot

GOSUB Get_Message             'Check which message to display

GOSUB Show_Message            'Send the message to LCD

GOSUB CD_orderer              'determine which CD to get

PAUSE 5000                    'pause while the old CD plays
GOSUB Send_CD                 'send the BOEbot new CD number

GOSUB Listen                  'get the status of the BOEbot

GOSUB Get_Message             'Check which message to display

GOSUB Show_Message            'Send the message to LCD

GOSUB Listen                  'get the status of the BOEbot

IF status > 3 THEN All_done
GOSUB Get_Message             'Check which message to display

GOSUB Show_Message            'Send the message to LCD


GOTO Loop:                    'Do this for every CD except for the last


All_Done:                     'The last CD is special

GOSUB Get_Message             'Check which message to display
GOSUB Show_Message            'Send the message to the LCD

END

' -------------------------------------------------------------------------------
' Subroutines
' -------------------------------------------------------------------------------

CD_orderer:                   'Determine the order to get the CDs
CDorder= CDorder + 1          'Get the CDs in numerical order

Return


Send_CD:
SEROUT 15, 17197, ["A", CDorder]                'Send the letter A and then the CD number

RETURN


LCD_Command:
  LOW RS                                        'enter command mode for LCD

LCD_Write:                                      'write the message to the LCD
  LCDbus = char.HighNib                         'output high nibble
  PULSOUT E, 1                                  'strobe the Enable line
  LCDbus = char.LowNib                          'output low nibble
  PULSOUT E, 1
  HIGH RS                                       'return to character mode
  RETURN


Get_Message:
  char = ClrLCD                                 'clear the LCD
  GOSUB LCD_Command
  LOOKUP status, [Msg0, Msg1, Msg2, Msg3, Msg4], addr 'determine which message to display


Show_Message:
  READ addr,char                                ' read a character from EEPROM
  IF (char = 0) THEN Msg_Done                   ' if 0, message is complete
  GOSUB LCD_Write                               ' write the character
  addr = addr + 1                               ' point to next character
  GOTO Show_Message


Msg_Done:
  RETURN


Listen:
SERIN 11\10,16468, [wait("A"), status]          ' Wait for letter A then send value

RETURN
```