Christopher P. Heagney
1 August, 2005

University of Florida

Department of Electrical and Computer Engineering

EEL 5666 - Intelligent Machine Design Laboratory

TAs: William Dubel & Steven Pickles

Instructors: Dr. A. A. Arroyo & Dr. E. M Schwartz

Final Written Report

Table of Contents:

Abstract

RoboShaQ is an autonomous, land based robot platform capable of shooting a free throw basket with the same skill as the famous pro basketball player Shaquille O'Neal. It will move up to and detect the free throw line, measure distance to the basket, calculate desired trajectory, and fire a nerf ball into a basket. This will be accomplished using infrared photo reflectors to detect the free throw line and line up to it. RoboShaQ will then use infrared distance measuring electronics to determine the distance to the basket. Taking into account the distance from the goal, RoboShaQ will then initiate a launch propelling a nerf ball through the air and into the basket.

Introduction

The idea for this project is to design and build a quick and simple land based robot. One that is capable of simple tasks such as obstacle avoidance and line detection. It will also be equipped with a powerful motor capable of catapulting a nerf ball through the air into a basket. Using the high powered Atmel Atmega128 microcontroller as the 'brains' and sensitive IR photo reflectors, IR range finders, bump switches, and iMEMS sensors (Integrated Micro Electro-Mechanical Systems), the RoboShaQ will be more than well equipped to accomplish the stated goal. Taking into account the failure of my previous project, the more sophisticated Autopilot, time is of the essence with this new design. Having only six days to design and build a new robot was quite the challenge, but very rewarding once it was finally completed.

Integrated System

RoboShaQ was created using a Mavric-IIB Atmega128 Board (Figure 1) as the brains, and a host of sensors, including IR Range finder (Figure 2) for forward looking obstacle avoidance and distance measurement, IR Reflectors (Figure 3) for line detection, Bump Switches (Figure 4) for obstacle avoidance, ADXL320 Solid State Accelerometer (Figure 5) to measure tilt, an ADXRS401 Solid State Gyro (Figure 6) to measure rate of change of yaw, and an RC Hummer (Figure 7) as the body.

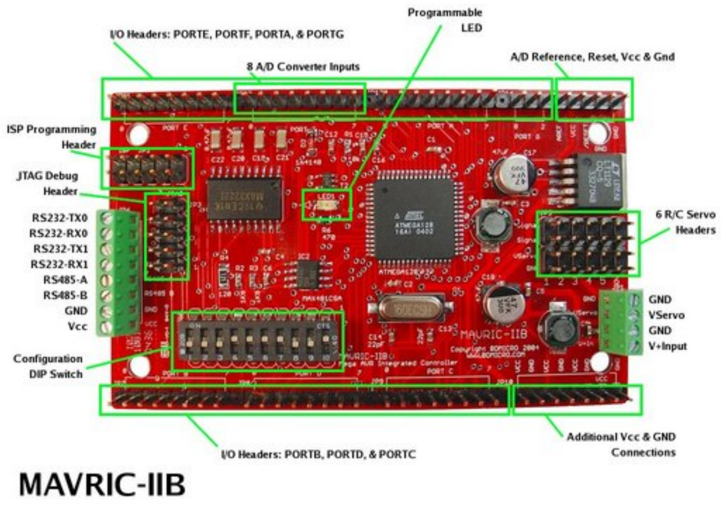

Figure 1.

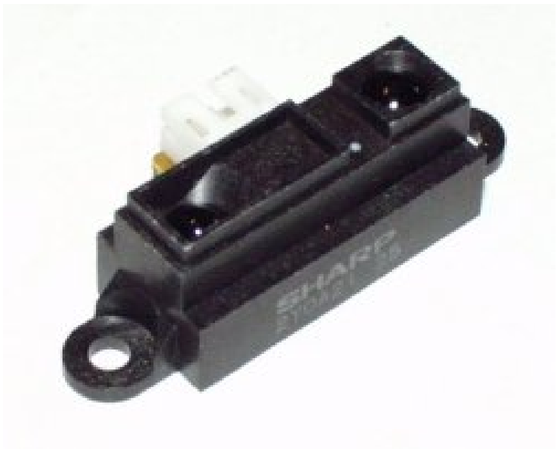

Figure 2.

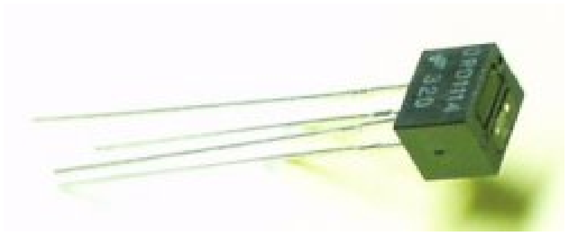

Figure 3.

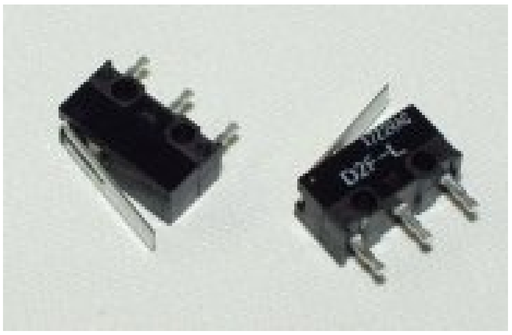

Figure 4.

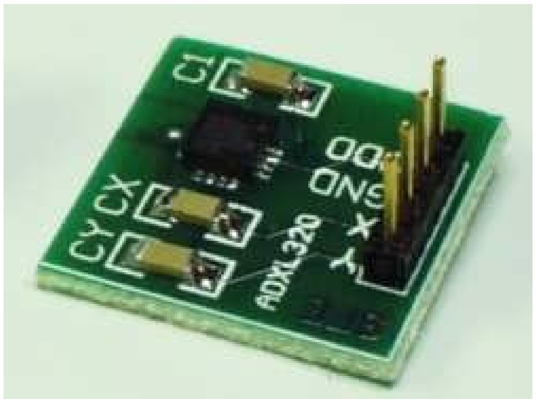

Figure 5.

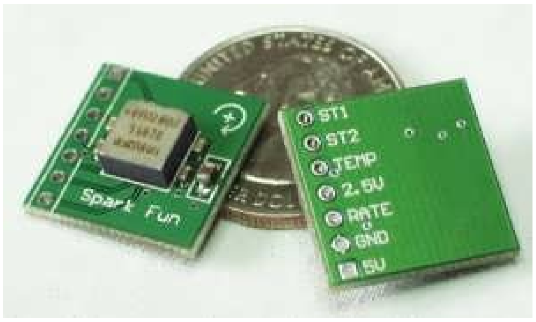

Figure 6.

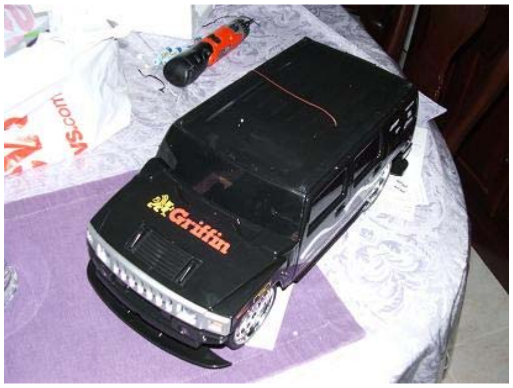

Figure 7.

The IR range finder, accelerometer, and gyro sensors are interfaced through Port F on the Mavric-IIB board via ADC interface. The servos are all run through the available servo channels built into the board. The IR reflectors and bump switches are digital and interfaced through the many digital I/O pins on board.

Mobile Platform

The mobile platform is a pre-constructed plastic RC car from Toys-R-Us. I decided to go with a pre-built RC car simply because of time issues. After the Autopilot project was abandoned, I did not have time to order parts and wait for them to arrive. Using an RC Hummer has the advantage of a ready to roll chassis, motors/controllers, wheels, battery back/charger, and everything I need already in the box. Its only disadvantage is having to hack the electronics to gain control of the motors. The final chassis has full mobility and hides all the wires and electronics inside the case, making for a nice finished presentation.

Actuation

RoboShaQ is powered by two simple RC car motors. Motor control in this particular RC is ultra simplistic, having 3 modes, on, off, and reverse. A lack of pwm hardware requires me to control it through software. This was originally an issue because the RC car traveled so fast at first. But after the catapult contraption was added, the weight was high enough to bog down the motors to a comfortable traveling speed. Uses of simple 9 gram servos (Figure 8) to control motor direction and a powerful electric fan motor (Figure 9) to drive the catapult arm. Servo control is easily done via the onboard servo pins. While catapult motor control is accomplished via a 30V 10A relay. The relay is triggered by a signal from one of the digital I/O pins, then run into a 2N2222 transistor to force the relay into position. Software code is fairly simple also. Servo control, as well as relay control, is shown in my software code attached below.

Figure 8.

Figure 9.

Sensors

RoboShaQ comes equipped with many powerful sensors that supply a plethora of information on all external events. These sensors include a forward looking IR range finder, IR reflectors, bump switches, dual-axis accelerometer, gyroscope, and temperature sensor. These sensors will be broken down in detail below.

IR range finder (Figure 10)

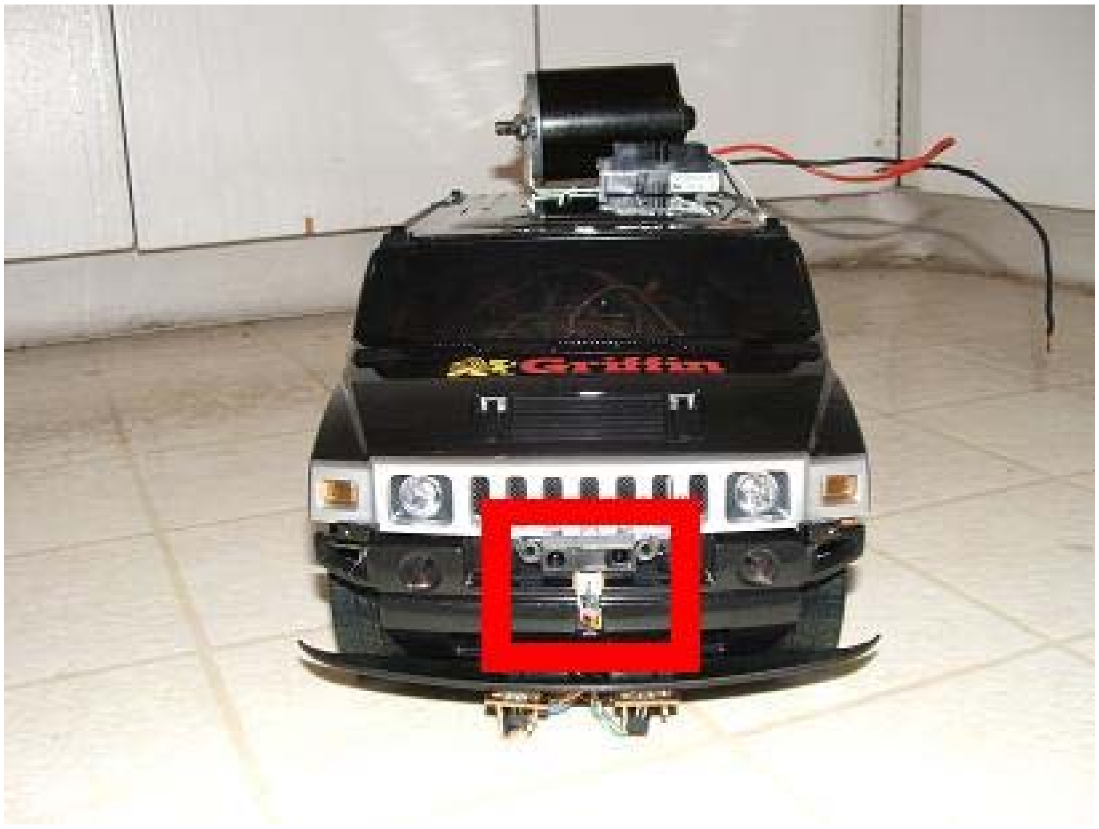

Figure 10.

The first priority of the forward looking IR range finder is obstacle avoidance. If an object gets too close to the front of the car, it will detect an eminent collision and take evasive action before any contact is made. It also functions to measure the distance to the basket. This is useful in judging how hard the catapult needs to launch the ball. Interface with the IR range finder is accomplished via the built-in ADC port, Port F. The ADC pin returns a value in software between 0-1024. The closer the object is, the higher the number gets. This allows for me to easily set the desired sensor threshold, in software, of a certain action.

IR reflectors (Figure 11)

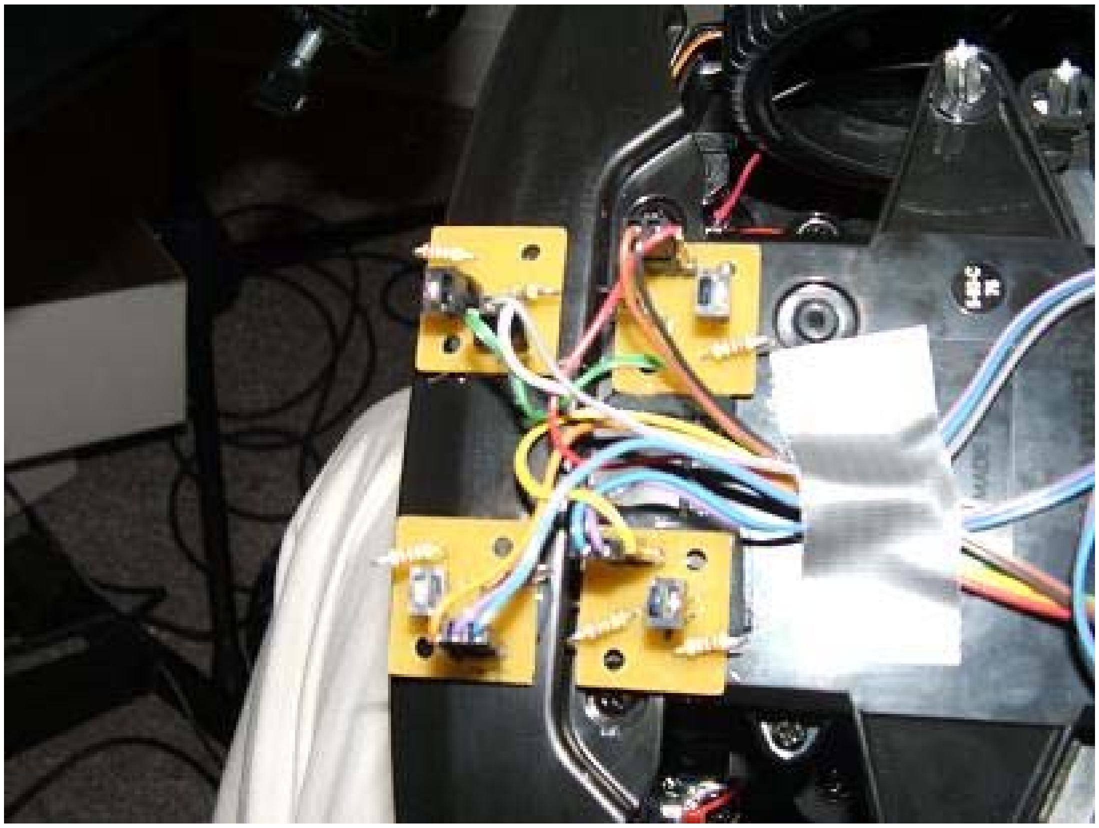

Figure 11.

The array of IR reflectors/detectors are used in line detection. They are mounted underneath the front bumper on the RoboShaQ. IR reflectors vary their output as a result of the amount of IR reflected back off of an object. Because white reflects more than black and because closer objects reflect more than farther objects, this is an easy way to accomplish line detection. The array is interfaced through digital I/O pins. Values returned are either 0 or 1. Thus allowing software to easily determine if the front of the care is on a line or not.

Bump Switches (Figure 12)

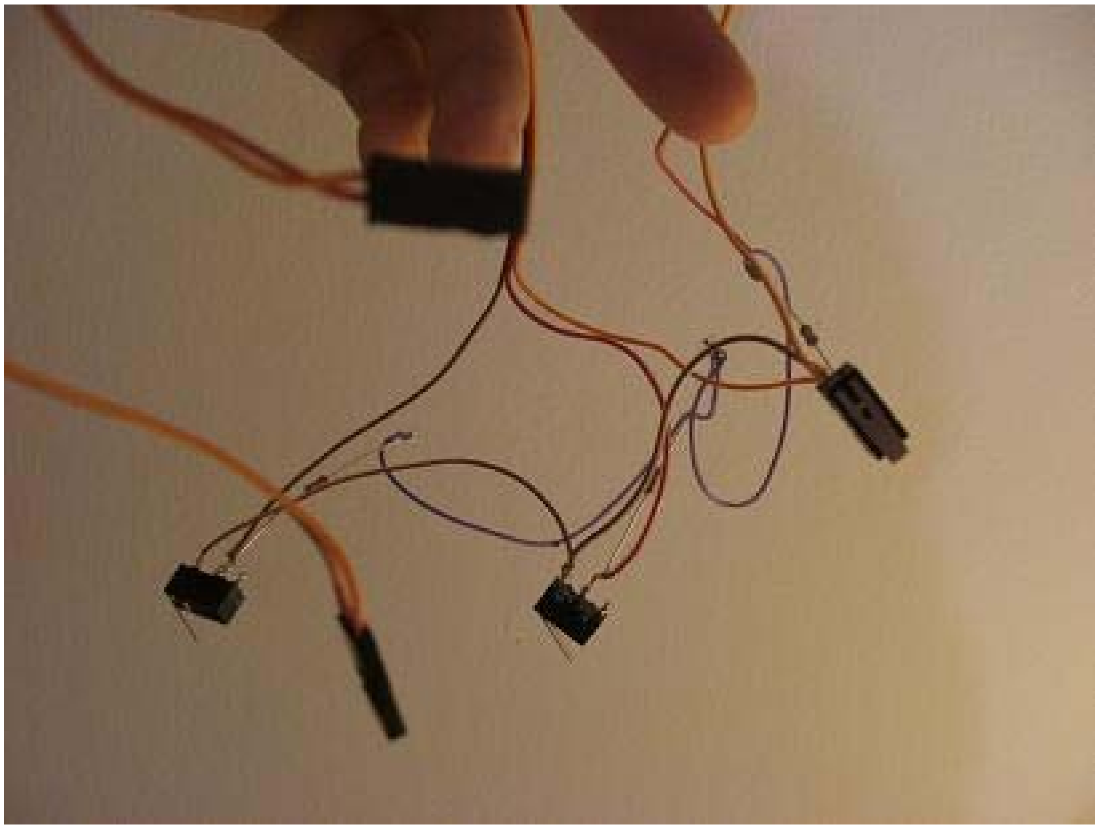

Figure 12.

Bump Switches are incredibly simple. If the button is pressed, it shorts a pullup resistor connected to the digital pin, driving it from its normal state of +5V down to ground. Thus, allowing me to check if it has been pressed very simply by polling a digital I/O pin. If the button has been pressed, then there has been a collision and the car must now backup and avoid the obstacle. This is the most basic form of obstacle avoidance.

iMEMS dual-axis accelerometer (Figure 13)

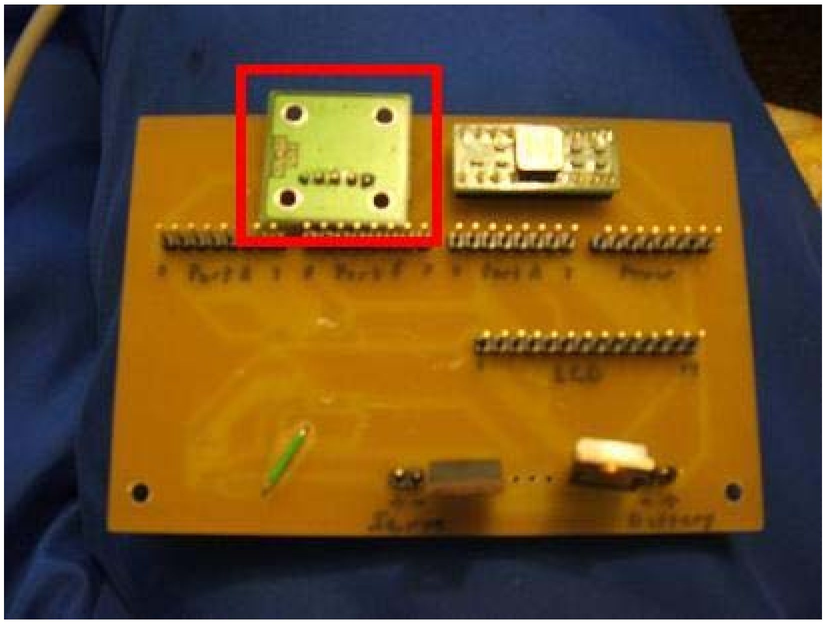

Figure 13.

The Integrated Micro Electro-Mechanical Systems dual-axis accelerometer onboard RoboShaQ has no purpose for the robots current function. It is simply a left over from the original Autopilot project I began this semester. The sensor, originally intended to measure changes in pitch and roll of an aircraft, now still measures pitch and roll…however irrelevant that is to a ground based robot. Although, useless, it is still functioning and returning data to the boards ADC port.

iMEMS gyroscope (Figure 14)

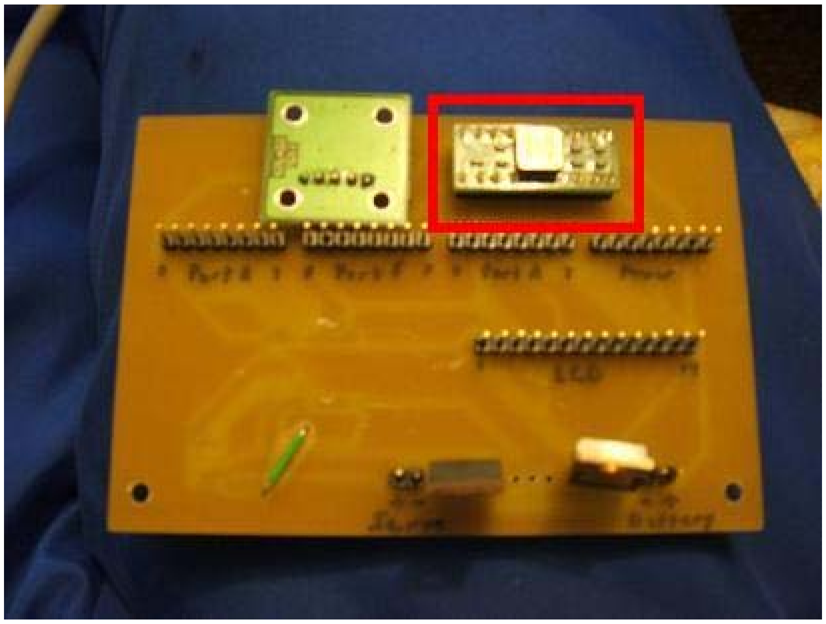

Figure 14.

Very similar to the accelerometer, the gyroscope on the RoboShaQ is also irrelevant. It returns data of rate of change in the yaw axis on the car via the ADC port. This data, although not used for the task of shooting a basket, it available and functioning without any problems.

<u>Temperature Sensor</u> (Figure 14)

Finally, RoboShaQ comes equipped with a temperature sensor left over from the original Autopilot project. Originally intended to keep data on internal fuselage temperatures and control overheating issues, it now just measures room temperature around the RC car. This sensor also interfaces through the ADC port on the Atmega128.

Behaviors

RoboShaQ is a simple, barebones robot with only basic behaviors. Three behaviors define this robot. First, obstacle avoidance, is accomplished using two types of sensors. A forward looking IR range detector detects if an object is getting too close and calls for evasion before any contact takes place. The other sensors used are bump switches places on the corners to detect contact and adjust actions accordingly. Second, line detection, is accomplished by using IR reflectors to measure light reflected back from an object, in this case, a line on the floor. It returns a digital on/off signal if the line on the ground reflects enough IR light back or not. Thus, allowing for simple detection of lines or objects. Finally, shooting the basket is accomplished by sending a digital +5V signal out to a relay that is connected to a higher voltage battery pack. That pack then drives a powerful motor which catapults the ball into the air. The +5V signal is driven to the desired turn on voltage via a 2N2222 transistor in line with the relay coil.

Conclusion

This project has taught me a lot about how to build a robot and how not to build a robot. This class forces you to go back and master the basics. With little room for error, you find yourself checking and rechecking all of the basics to find any mistakes you may have made before you hook up power to your precious robot. That I think is the greatest benefit of this hands-on design experience. Coming in to this class, my only experience with microcontrollers was EEL4744C. Coming out of this class, I now know how to research parts, read pinouts, design entire board schematics, plan board layouts, send out boards to be custom made, interface a variety of components, and do much much more than I previously knew possible. After beginning this semester with high hopes of flight, I have grown a much healthier respect for the difficulty involved with the fundamentals of flight. I feel however, that I have conquered the complications of stability and control and may pursue my dreams of autonomous flight at a later time with better components. When it came down to crunch time, however, I am proud that I could put together an entire robot in only the final week. Although basic, RoboShaQ introduced me to new electronics that I would have otherwise not understood or tried to learn. Some say that with every failure there is a success looming just around the corner. It appears that even though my initial goal or autonomous flight remains unrealized, the lessons learned from this experience will greatly shape my future as an engineer and as a person.

You can check on my webpage http://plaza.ufl.edu/buffstud for pictures and video of the project in action.

Documentation

The following is a list of resources used:

1. Mavric-IIB Board Manual http://www.bdmicro.com/images/mavric-iib.pdf
2. ADXL320 Data Sheet http://www.analog.com/UploadedFiles/Data_Sheets/84033828ADXL320_0.pdf
3. ADXRS401 Data Sheet http://www.analog.com/UploadedFiles/Data_Sheets/279107307ADXRS401_0.pdf
4. Lassen IQ Data Sheet http://www.sparkfun.com/datasheets/GPS/Lassen%20iQ_Reference%20Manual.pdf
5. Fairchild 74LVX4245 Data Sheet http://www.fairchildsemi.com/ds/74/74LVX4245.pdf
6. Eagle 4.14 Manual ftp://ftp.cadsoft.de/pub/program/4.14/manual-eng.pdf

Appendices

This code is written in Bascom, a BASIC language compiler for AVR chipsets. I have tried to make it as easy as possible to read by italicizing comments. Enjoy ☺

```
'*******************************************************************************
'*  Christopher P. Heagney                                                     *
'*  Intelligent Machine Design Laboratory                                      *
'*  Summer 2005                                                                *
'*  University of Florida                                                      *
'*  College of Electrical and Computer Engineering                             *
'*                                                                             *
'*  Robot Name: RoboShaQ                                                       *
'*  Robot Function: To detect a free throw line, line up to it, measure        *
'*                to the basket, and launch a ball into the basket.            *
'*******************************************************************************




'*******************************************************************************
'*  Hardware connections                                                       *
'*******************************************************************************
'Pinb.4 is connected to left IR reflector
'Pinb.5 is connected to center IR reflector
'Pinb.6 is connected to right IR reflector
'Pind.5 is connected to left front bump switch
'Pind.6 is connected to center front bump switch
'Pind.7 is connected to right front bump switch
'ADC Pin0 is connected to forward IR range finder
'ADC Pin1 is connected to x-axis accelerometer
'ADC Pin2 is connected to y-axis accelerometer
'ADC Pin3 is connected to gyroscope
'ADC Pin4 is connected to temperature sensor
'Pind.4 is connected to catapult relay

'IR range finder:  Orange(GND)   Yellow(VCC)
'Bumper switch array:  Orange(GND)   Red(VCC)
'Servo Power from board:  Grey(GND)   White(VCC)
```

```
'Servos Power:  Grey(GND)   Purple(VCC)
'IR Reflectors:  Purple(GND)   Blue(VCC)  x2
'                Black(GND)   Brown(VCC)
'                Grey(GND)   White(VCC)




'*******************************************************************************
'*  Configuations                                                              *
'*******************************************************************************
'Config chipset and crystal frequency
$crystal = 14745600
$regfile = "m128def.dat"

'Turns on onboard LED to show power is on
Config Portb = &B00000001
Portb = 255                                   'set portb.0 as high

'Config LCD
Config Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5 , Db6 = Porta.6 , Db7 = Porta.7 , E = Porta.3 , Rs =
Porta.2
Config Lcd = 20 * 4                            'configure lcd screen size

'Config ADC
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc

'Config Servos
Config Servos = 4 , Servo1 = Portc.0 , Servo2 = Portc.1 , Servo3 = Portc.2 , Servo4 = Portc.3 , Reload = 20
Config Portc = Output

'Config PortD as digital input for bump switches and IR reflectors
Config Portd = &B00010000
Enable Interrupts




'*******************************************************************************
'*  Dimension Variables                                                        *
'*******************************************************************************
Dim Bootcount As Byte                         'variable used in bootup sequence
Dim Distance As Word                          'variable to hold IR distance value
Dim Screenrefresh As Byte                      'used to temporarily refresh the screen
Dim Irscancomplete As Byte                      'used in line detection
Dim Turnwhichway As Byte                         'variable to store which direction to turn
Dim Whichsensordetectedimpact As Byte                 'which sensor detected impact?
Dim Linedetect As Byte                        'which sensors detected a line?
Dim Motoraction As Byte                        'what do you want the motors to do?
Dim Servo1stop As Byte                        'Servo Configuration Values
Dim Servo1forward As Byte
Dim Servo1backward As Byte
Dim Servo2center As Byte
Dim Servo2left As Byte
Dim Servo2right As Byte
Dim Servo1 As Byte                              'used in bootup servo configuration
Dim Servo2 As Byte
Dim Shot As Byte                              'have we already shot the ammo?



'*****************************************************************************
'*  Declare Subroutines                                                      *
'*****************************************************************************
Declare Sub Avoidimpact(whichsensordetectedimpact As Byte)
Declare Sub Trackline(linedetect As Byte)
Declare Sub Gomotor(motoraction As Byte)
Declare Sub Shootbasket



'*****************************************************************************
'*  System bootup                                                            *
'*****************************************************************************
Servo(1) = 55
Servo(2) = 50
Cls                                   'clear the LCD display
Cursor Off Noblink                           'makes the LCD cursor not blink
Lcd "Christopher Heagney"                       'Prints my name on the first line of the LCD
Lowerline                               'move to the next line
Lcd "System Start in: "
'This loop will countdown 6 seconds before the system starts operating.  This
'  allows time to reprogram the controller, get the robot oriented as desired,
'  make any last second adjustments, display the programs creator, and basically
'  just look cool before everything starts.
For Bootcount = 6 To 1 Step -1                    'start a countdown loop
   Locate 2 , 18                        'place lcd curser at position 2,18
   Lcd Bootcount
   Locate 4 , 17
   Lcd "IM  "
   Waitms 250                           'pause 250ms
   Locate 4 , 17
   Lcd " MD "
   Waitms 250
   Locate 4 , 17
   Lcd "  DL"
   Waitms 250
   Locate 4 , 17
   Lcd "I  L"
   Waitms 250
Next Bootcount

'This next part is simply an initial bootup calibration.  The servos sometimes
'  move and hardcoded values are no longer accurate.  This allows the user to
'  very easily calibrate the servos before every start.  It basically starts
'  the servos at a certain value and asks if the motor is doing the desired
'  result.  If it it, they press a bump switch to let the program know it is
'  good and the program then stores that value.  This is repeated for all six
'  servo settings.  Now the program is ready to go, with all its nicely
'  calibrated values stored in memory.
Cls
Lcd "Servo 1 Diagnostics"
Locate 2 , 1
Lcd "Press Left Front"
Locate 3 , 1
Lcd "Bumper to Config"
Locate 4 , 1
Lcd "Servo 1 Forward:    "
Servo1 = 80
Do
   Servo(1) = Servo1
   Decr Servo1
   Locate 4 , 18
   Lcd Servo1
   Waitms 150
Loop Until Pind.5 = 0
Servo1forward = Servo1
Waitms 150

Locate 4 , 1
Lcd "Servo 1 Stop:       "
Do
   Servo(1) = Servo1
   Decr Servo1
   Locate 4 , 15
   Lcd Servo1
   Waitms 150
Loop Until Pind.5 = 0
Servo1stop = Servo1
Waitms 150

Locate 4 , 1
Lcd "Servo 1 Backward:   "
Do
   Servo(1) = Servo1
   Decr Servo1
   Locate 4 , 19
   Lcd Servo1
   Waitms 150
Loop Until Pind.5 = 0
Servo1backward = Servo1
Servo(1) = Servo1stop
Waitms 300

Locate 4 , 1
Lcd "Servo 2 Left:       "
Servo2 = 70
Do
   Servo(2) = Servo2
   Decr Servo2
   Locate 4 , 15
   Lcd Servo2
   Waitms 150
```

```
Loop Until Pind.5 = 0
Servo2left = Servo2
Waitms 150

Locate 4 , 1
Lcd "Servo 2 Center:     "
Do
   Servo(2) = Servo2
   Decr Servo2
   Locate 4 , 17
   Lcd Servo2
   Waitms 150
Loop Until Pind.5 = 0
Servo2center = Servo2
Waitms 150

Locate 4 , 1
Lcd "Servo 2 Right:      "
Do
   Servo(2) = Servo2
   Decr Servo2
   Locate 4 , 15
   Lcd Servo2
   Waitms 150
Loop Until Pind.5 = 0
Servo2right = Servo2
Waitms 150
Servo(2) = Servo2center
Wait 1


'Ok, the servos have been calibrated and their values stored in memory.  Now
'  press a button to continue to the main program.
Cls
Lcd "Press Left Bumper"
Lowerline
Lcd "To continue.."
Do
Loop Until Pind.5 = 0
Wait 1



'******************************************************************************
'*  Main Program                                                              *
'******************************************************************************
Cls                                       'clear the LCD display
Motoraction = 2                               'set the robot in motion forward at beginning
Screenrefresh = 0                             'counter to periodically refresh the screen
Shot = 0                                    'default start condition is having ammo
Do
'Check to see if front IR is too close to something
Distance = Getadc(0)                             'get IR distance measurement
If Distance > 300 Then                            'If it's less than a threshold ammount, then panic
   Whichsensordetectedimpact = 1                     'tell it impact is coming from the front
   Call Avoidimpact(whichsensordetectedimpact)
```

```
End If


'Check to see if bumper switches have been pressed
If Pind.5 = 0 Then
   Whichsensordetectedimpact = 3                'tell it impact is coming from the left
   Call Avoidimpact(whichsensordetectedimpact)
End If
If Pind.6 = 0 Then
   Whichsensordetectedimpact = 1                'tell it impact is coming from the front
   Call Avoidimpact(whichsensordetectedimpact)
End If
If Pind.7 = 0 Then
   Whichsensordetectedimpact = 2                'tell it impact is coming from the right
   Call Avoidimpact(whichsensordetectedimpact)
End If


'Check IR reflectors for line detection
If Pinb.4 = 0 And Pinb.5 = 0 And Pinb.6 = 0 Then
   Linedetect = 1
   Irscancomplete = 1
End If
If Irscancomplete = 1 Then
   Goto Skipircheck
End If

If Pinb.4 = 0 And Pinb.5 = 0 Then
   Linedetect = 2
End If

If Pinb.5 = 0 And Pinb.6 = 0 Then
   Linedetect = 3
End If

If Pinb.4 = 0 And Pinb.6 = 0 Then
   Linedetect = 4
End If

If Pinb.4 = 0 Then
   Linedetect = 5
End If

If Pinb.5 = 0 Then
   Linedetect = 6
End If

If Pinb.6 = 0 Then
   Linedetect = 7
End If


'Linedetect basically stores which combination of line detectors see the line
'  or not.  Allowing for different behavioral responses.
Skipircheck:
If Linedetect <> 0 Then                      'Did it detect a line?
```

```
   Call Trackline(linedetect)                    'If so, then call trackline
End If
Call Gomotor(motoraction)

'Periodically refresh the screen
Incr Screenrefresh
If Screenrefresh = 255 Then
   Screenrefresh = 0
   Cls
End If
Loop
End                                         'end the program




'********************************************************************************
'*  Subroutines                                                                  *
'********************************************************************************
Sub Avoidimpact(whichsensordetectedimpact As Byte)
   Motoraction = 1                              'all stop
   Call Gomotor(motoraction)
   Waitms 50
   'If impact on the left, then tell it to turn to the right
   'If impact on the right, then tell it to turn to the left
   'if impact in the center, then randomly pick a direction to go
   Select Case Whichsensordetectedimpact
      Case 1
         Locate 1 , 1
         Lcd "Watchout For Impact!"
         Locate 2 , 1
         Lcd "Attempting Evasion.."
         Turnwhichway = Rnd(1)
      Case 2
         Locate 1 , 1
         Lcd "Impact on the Right!"
         Locate 2 , 1
         Lcd "Attempting Evasion.."
         Turnwhichway = 0
      Case 3
         Locate 1 , 1
         Lcd "Impact on the Left! "
         Locate 2 , 1
         Lcd "Attempting Evasion.."
         Turnwhichway = 1
   End Select
   Motoraction = 3                              'go backwards
   Call Gomotor(motoraction)
   Wait 2
   If Turnwhichway = 0 Then
      Motoraction = 4                           'go left for a while then go straight again
      Call Gomotor(motoraction)
      Wait 1
      Motoraction = 2
      Call Gomotor(motoraction)
   Else
      Motoraction = 5                         'go right for a while then go straight again
      Call Gomotor(motoraction)
      Wait 1
      Motoraction = 2
      Call Gomotor(motoraction)
   End If
End Sub


Sub Trackline(linedetect As Byte)
   Select Case Linedetect
      'If all sensors on, then we've reached the free throw line and stop
      Case 1
         Locate 1 , 1
         Lcd "Lined up, Ready shot"
         Motoraction = 1
         Call Gomotor(motoraction)
         Wait 1
         If Shot = 0 Then                     'have we already shot the ammo?
            Call Shootbasket                  'if not, then shoot it
         Else                              'if we did, then wait for reload
            Cls
            Lcd "Reload then press"
            Locate 2 , 1
            Lcd "Left front bumper.."
            Do
            Loop Until Pind.5 = 0
            Wait 2
            Call Shootbasket                  'once it's reloaded, shoot it!
         End If
      'If left and center are on, then go slowly left
      Case 2
         Locate 1 , 1
         Lcd "Turn Left a little. "
         Servo(2) = Servo2right               'turn wheel right
         Servo(1) = Servo1backward               'then backup
         Waitms 500
         Servo(2) = Servo2center              'then turn wheel straight
         Servo(1) = Servo1forward              'and go forward
      'If right and center are on, then go slowly right
      Case 3
         Locate 1 , 1
         Lcd "Turn Right a little."
         Servo(2) = Servo2left                'turn wheel right
         Servo(1) = Servo1backward               'then backup
         Waitms 500
         Servo(2) = Servo2center              'then turn wheel straight
         Servo(1) = Servo1forward              'and go forward
      'If left and right are on, then somethings screwed up..keep going straight
      Case 4
         Locate 1 , 1
         Lcd "Ummm...Go Straight. "
         Motoraction = 2
         Call Gomotor(motoraction)
      'If only left is on, then break left
```

```
    Case 5
      Locate 1 , 1
      Lcd "Break Left.         "
      Servo(2) = Servo2right                    'turn wheel right
      Servo(1) = Servo1backward                   'then backup
      Wait 1
      Servo(2) = Servo2center                   'then turn wheel straight
      Servo(1) = Servo1forward                   'and go forward
    'If only center is on, then keep going straight
    Case 6
      Locate 1 , 1
      Lcd "Keep Straight ahead."
      Motoraction = 2
      Call Gomotor(motoraction)
    'If only right is on, then break right
    Case 7
      Locate 1 , 1
      Lcd "Break Right.        "
      Servo(2) = Servo2left                     'turn wheel right
      Servo(1) = Servo1backward                   'then backup
      Wait 1
      Servo(2) = Servo2center                   'then turn wheel straight
      Servo(1) = Servo1forward                   'and go forward
  End Select
End Sub


Sub Gomotor(motoraction As Byte)
'Case 1: All stop
'Case 2: Forward
'Case 3: Backward
'Case 4: Left
'Case 5: Right
'Case 6: Left slowly
'Case 7: Right slowly
  Select Case Motoraction
    Case 1
      Servo(2) = Servo2center
      Servo(1) = Servo1stop
    Case 2
      Servo(2) = Servo2center
      Servo(1) = Servo1forward
    Case 3
      Servo(2) = Servo2center
      Servo(1) = Servo1backward
    Case 4
      Servo(2) = Servo2left
      Servo(1) = Servo1forward
    Case 5
      Servo(2) = Servo2right
      Servo(1) = Servo1forward
    Case 6
      Servo(2) = Servo2left
      Servo(1) = Servo1forward
      Waitms 100
      Servo(1) = Servo1stop
```

```
         Waitms 100
         Servo(1) = Servo1forward
      Case 7
         Servo(2) = Servo2right
         Servo(1) = Servo1forward
         Waitms 100
         Servo(1) = Servo1stop
         Waitms 100
         Servo(1) = Servo1forward
   End Select
End Sub


Sub Shootbasket
   Cls
   Lcd "Fire!!            "
   Portd = 255                              'Pulse on catapult relay
   Waitms 250
   Portd = 0
   Shot = 1                                'tell it that we have shot the ammo
End Sub
```