Date: 12/10/02
Student Name: Don McMann
TA: Uriel Rodriguez
Jason Plew
Instructor: A. A Arroyo

University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory

B.E.E.R.-Bot

Beverage Equipped Entertainment Robot

Final Report

Table of Contents

Abstract

The Beverage Equipped Entertainment Robot (B.E.E.R.-Bot) was designed to serve beverages at social events and to entertain at the same time. When B.E.E.R.-Bot finds a person, it will offer them a beverage using voice recognition. After the beverage has been dispensed, B.E.E.R.-Bot will wander the room looking for another person to give a beverage to. All the while making quips related to its behaviors and environments.

Executive Summary

B.E.E.R.-Bot is an autonomous robot designed to navigate a room and distribute beverages. To move around the room, B.E.E.R.-Bot utilizes two Crown Victoria power window motors for its drive train. These motors run at approximately seven percent of maximum speed in order to maintain reliable control and stability.

In order to interact with its' environment, B.E.E.R.-Bot uses 3 infrared sensors and 3 bump switches for obstacle avoidance, a pyroelectric sensor to detect the presence of humans, a voice playback integrated circuit and a voice recognition system for communication during beverage dispensing. B.E.E.R.-Bot uses a combination of the I.R. sensors and the pyrosensor to locate and approach a person.

A five-liter mini keg system is used to contain and transport the beverage. The tapping system uses a twelve-gram CO_2 cartridge to pressurize the keg for dispensing.

The 16F877 PIC microcontroller was used to control B.E.E.R.-Bot and all of its' behaviors. A Panasonic 12 volt, 7.2-amp/hour battery is used to power B.E.E.R.-Bot.

Introduction

In order to decrease the time spent waiting in line for beverages at large social functions. B.E.E.R.-Bot was designed to wander around at these events and dispense beverages to people. It randomly seeks out individuals and offers them a beverage. Upon hearing a "yes" response, B.E.E.R.-BOT will dispense the beverage and then go back to searching for other individuals to deliver its payload to.

This paper will describe the design, testing and procedures used to implement this project.

Integrated Systems

B.E.E.R.-Bot's integrated system is controlled by the PIC16F877 microprocessor mounted on the PicProto64 board for system integration. The PicProto64 is a bare circuit board that had to be put together.

Its' platform consists of a ¾" birch round platform to support the weight of the keg. It uses a differential drive system with two automobile replacement power window motors controlled by separate D100-B25 motor drivers. The motor drivers are directionally controlled via logic lines and speed controlled with a pulse width modulation from the microprocessor. The motor drivers are powered directly from the battery to limit current drawn by the microprocessor.

The infrared sensors, used for obstacle avoidance, are connected to the processor via analog ports. As well as the pyroelectric sensor, which is used for people detection together with the I.R. sensors. The bump switches, also used for obstacle avoidance are each connected to separate digital inputs.

The voice recognition system is self-contained and notifies the microprocessor when a key word has been recognized, and leaves all decisions and behaviors based on a positive recognition up to the microprocessor. The message playback system is controlled by the microprocessor and messages can be selected based on the preprogrammed behaviors of the robot.

These systems are used jointly to meet the objective of the robot, which is to find people and give them a beverage in a fun manner while avoiding obstacles.

The following flow chart demonstrates B.E.E.R.-Bots' integrated electrical control system:

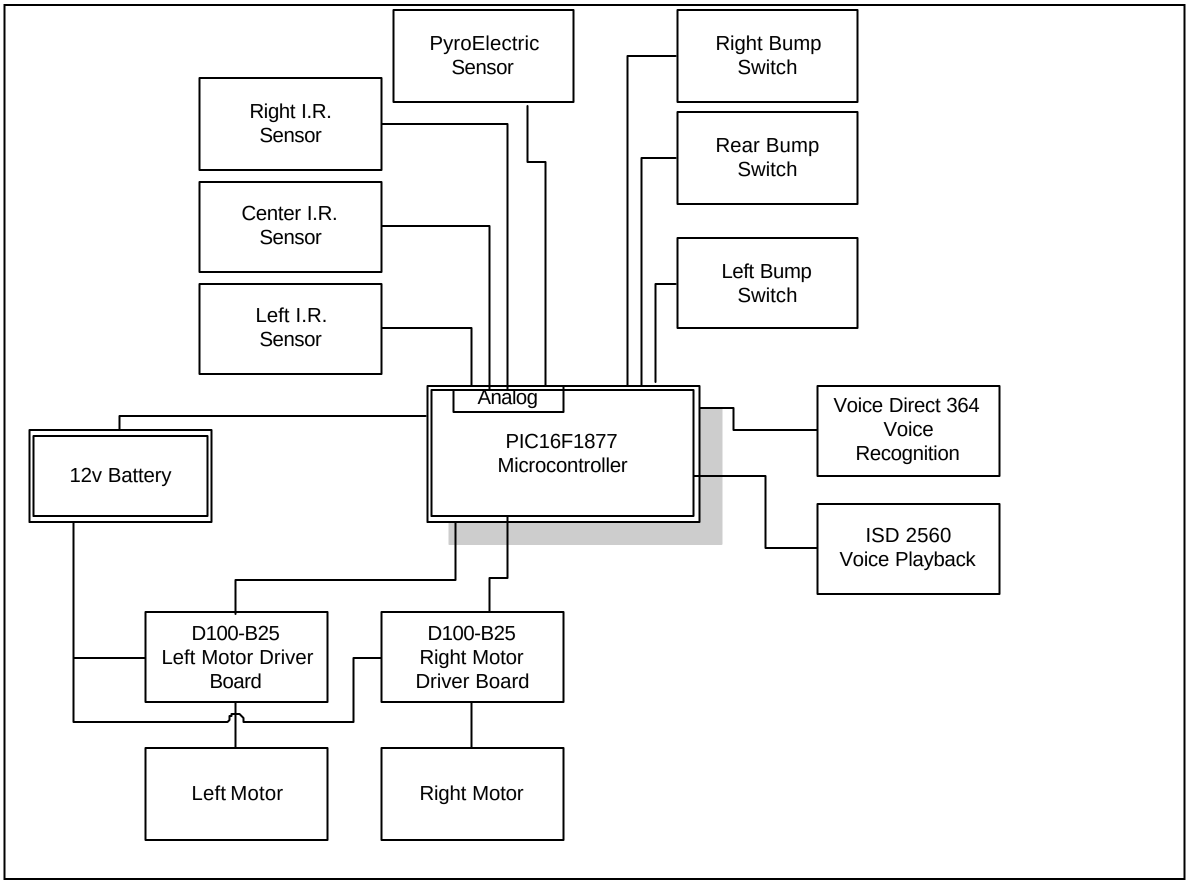


Figure 1 Integrated System

Mobile Platform

The mobile platform used was based on the Rug Warrior design. The platform itself is a precut 12” diameter, ¾” thick piece of birch. It incorporates a smaller 6 ½” platform that the keg sets in and a battery holder that was designed using AutoCAD and cut out on the T-Tech machine. Figure 2 is the AutoCAD design.

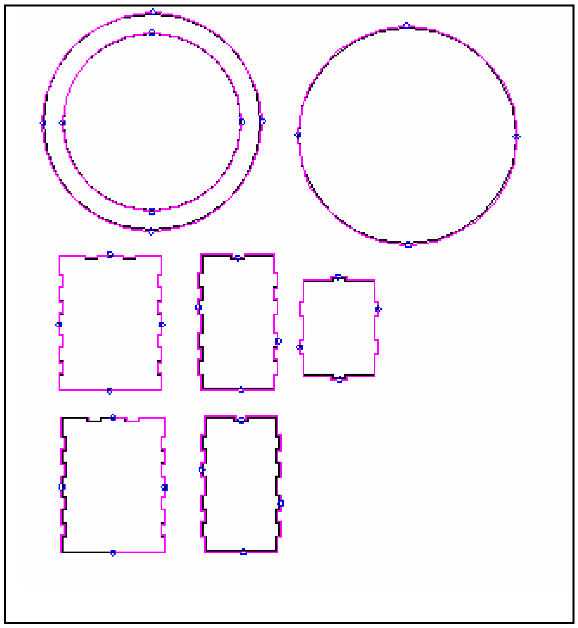


Figure 2 AutoCAD

Originally the platform was designed using a thin 2.0 Ah battery, however this had to be changed to a 7.2 Ah battery that is 3 times the size of the original. This took up considerable space and added a lot of weight to the rear of the platform. This caused the platform to be highly unstable when the weighted keg was not present during testing.

If I were to do it again I would use a 15” diameter platform to compensate for the large battery.

Actuation

B.E.E.R.-Bots' motors have to be strong enough to carry the weight of the full keg, the battery, platform and other parts. For this reason two replacement motors for the Crown Victorian power window motors were chosen. They are controlled by the D100-B25 motor drivers using two control lines for forward and reverse control of the motors. The input line truth table is depicted in Table 1.

Enable	IN1	IN2	Action
H	L	L	Stop
H	L	H	Forward
H	H	L	Reverse
H	H	H	Stop
L	X	X	OFF

Table 1 Input Line Truth Table

Motor speed is controlled with a 1kHz pulse width modulation signal supplied by the microprocessor. Originally a 20 MHz oscillator was used to control the microprocessor but the smallest PWM signal that could be generated was a 1.22 kHz signal. Therefore a 4 MHz oscillator was used to control the microprocessor and generate a 1kHz PWM.

The motor drivers draw power directly from the battery to minimize current draw from the microprocessor board. Figure 3 illustrates the wiring diagram for the motors and drivers.

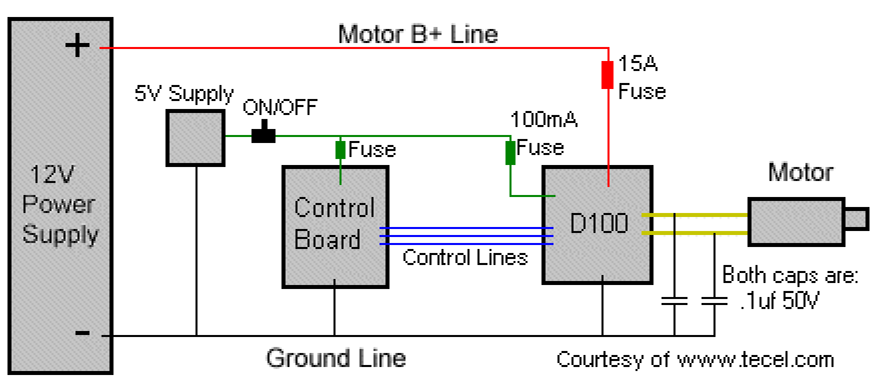


Figure 3 D100 wiring diagram

The 0.1µF capacitors are used for noise suppression, B.E.E.R.-Bot was tested with and without these capacitors and ran fine either way.

It was recommended that the logic lines be inserted before the PWM signal to protect the PAL's on the driver board. To do this a hardware PWM was implemented via the PIC and a relay was placed inline between the microprocessor and the D100 to control activation of the PWM. The relay is controlled by the microprocessor and is delayed 10msec after the logic lines are asserted. The wiring diagram for the relay circuit is shown in figure 4.

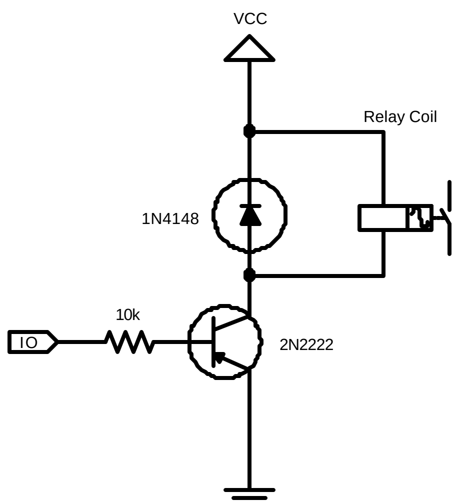


Figure 4 Relay Control Circuit

Sensors

IR Sensors

Three Sharp GP2D12 infrared distance-measuring sensors are used for obstacle avoidance. The GP2D12 documentation reports that it accurately determines the range to a target between 3.9 inches and 31.5 inches and can be used as a proximity detector to detect objects between 0 and 51inches.

I tested each sensor using an 11 x 16 inch cardboard square held perpendicular to and directly in front of each sensor. A maximum distance of 84 inches was used as the initial measuring point. An output was recorded every three inches between 84 and 10 inches, and every inch between 10 and 0 inches. The following graph demonstrates the results of this experiment:

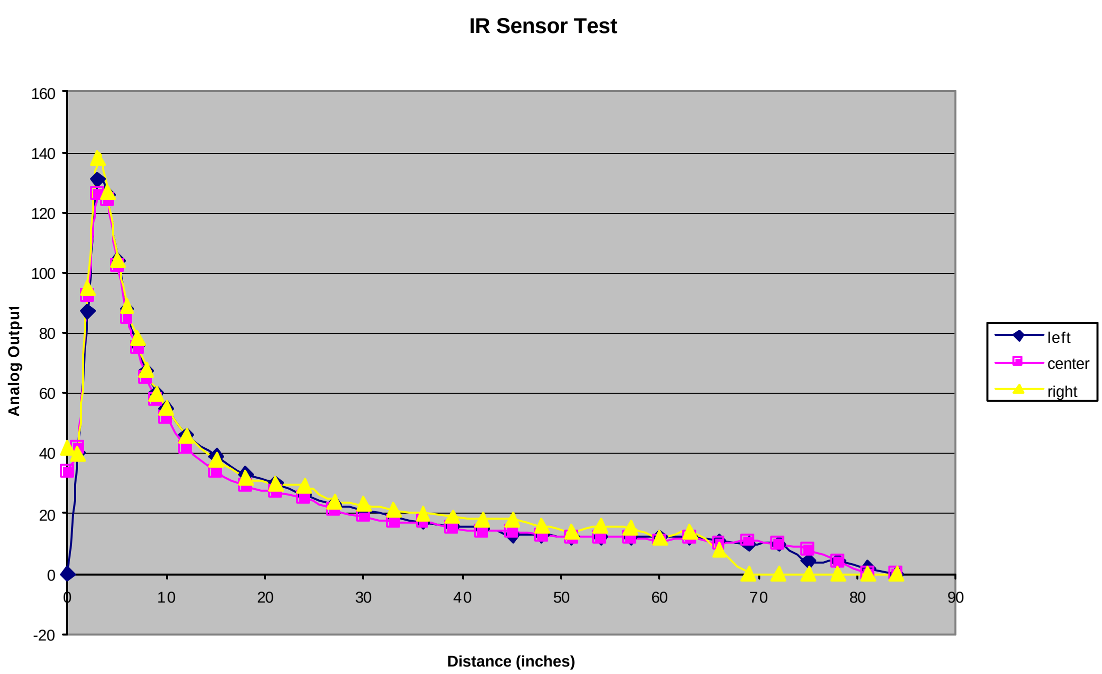


Figure 5 I.R. Sensors

As can be seen from the graph, all three sensors are very similar and begin a noticeable up trend at 30 inches that peaks at 3 inches. The distance of an object within this range can be seen fairly accurately. Anything between 30 inches and 60 inches can be detected, but an accurate distance cannot be established.

B.E.E.R.-Bot incorporates two automobile replacement power window motors as a drive system. When initial obstacle avoidance testing began, a PWM was used to run the motors at 80% of their maximum output. This was too fast as B.E.E.R.-Bot could detect an obstacle and turn without hitting the obstacle or spilling its payload. After running several trials a PWM of 7% was chosen as this allowed B.E.E.R.-Bot to see an obstacle and execute an avoidance maneuver smoothly.

B.E.E.R.-Bot has an IR sensor mounted on the front center of the platform to detect obstacles directly in front of it. In this case it stops, reverses directions, turns randomly to the left or right for a random amount of time and continues on.

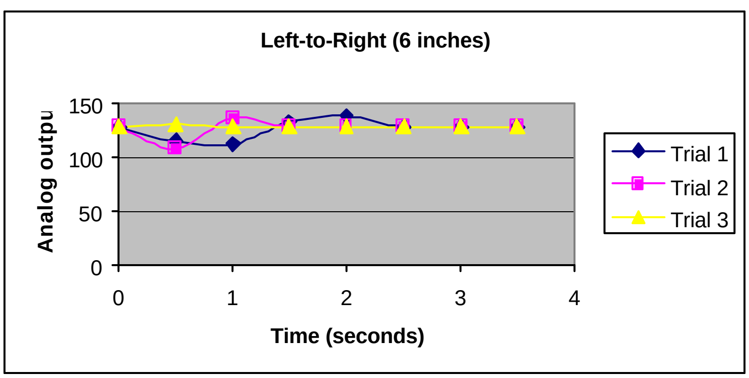
B.E.E.R.-Bot's drive wheels protrude beyond the platform slightly at approximately the 50-degree marks, measured from the center of the platform, to the left and right. To compensate for this fact, the left and right IR sensors were mounted at the 45-degree marks to guard the wheels. This left an unexpected to gap in the sensing areas between the center and left and right sensors. To correct this error, the left and right sensors were each moved inward 5 degrees toward center.

Bump sensors in the form of whiskers were placed around the platform, 2 in the front and one in the rear, to stop the robot if the IR sensors fail to detect an obstacle. These were one of the last systems put on the robot as platform space was a major concern during development of B.E.E.R.-Bot.

Pyroelectric Sensor Package

B.E.E.R.-Bot incorporates an Eltec 442-3 dual-element lithium tantalite pyroelectric sensor. The lithium tantalite crystal is doped with an electrode on opposite sides. When infrared energy between the range of 8 and 14 micrometers hits the substrate, heat is generated which displaces electrons and creates a charge between the two electrodes. The voltage difference between the two elements is amplified and creates a change in the nominal output voltage of the sensor. The output rises when motion is detected in one direction and falls when motion is detected in the opposite direction. Thus the pyroelectric sensor is ideal for detecting the movement of humans, as the wavelength of maximum energy radiated by humans is about 10 micrometers.

I tested the pyroelectric sensor by walking across its field of view at ½, 3, 5 and 8-foot intervals at a moderate walking pace in the left-to-right and right-to left directions for each interval. The nominal value when no movement is in the field of view of the sensor is 2.5 volts on the output pin, which corresponds to a reading of 128 on the analog output of the microprocessor. The LCD is set to refresh every 0.5 seconds and two people recorded results as another walked across the field of view. The first result on each graph is the nominal value of 128, the next result is the first non-nominal value seen. The following graphs show the results of these tests:



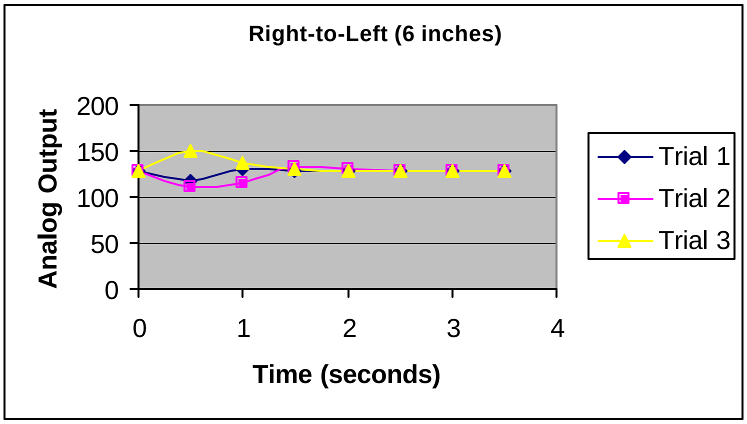


Figure 6 Pyro at 6”

As can be seen from these graphs, there are no distinct differences in the outputs between the left-to-right and right-to-left motions at six inches from the sensor. Therefore the presence of a person can be detected, however people following capabilities will not be possible at this distance.

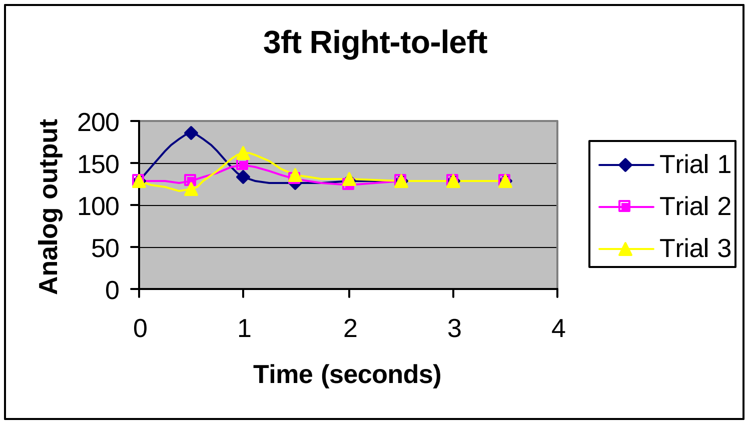
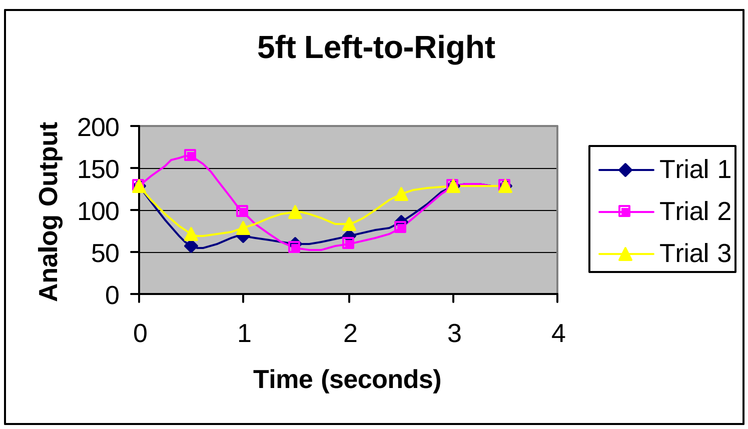


Figure 7 Pyro at 3'

In the preceding graphs there is a marked difference between the separate directions. When moving in the left-to-right direction there is a strong down trend in the output before it levels back to the nominal value, while the right-to-left movement demonstrates a strong up trend in the output and drops back to the nominal value of 128. People following skills and detection can be implemented fairly easily at this distance.



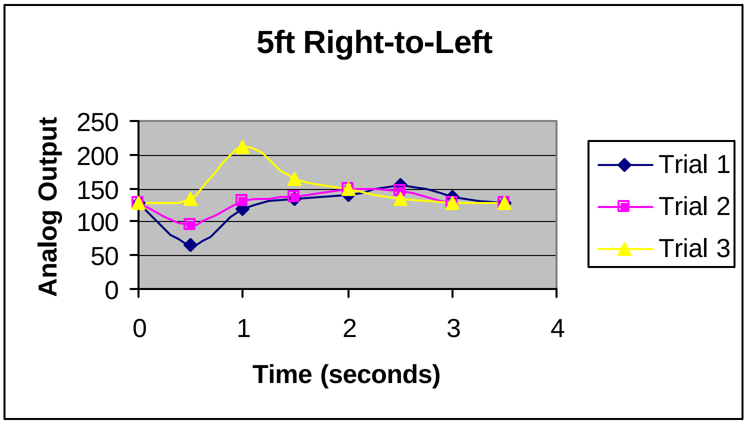
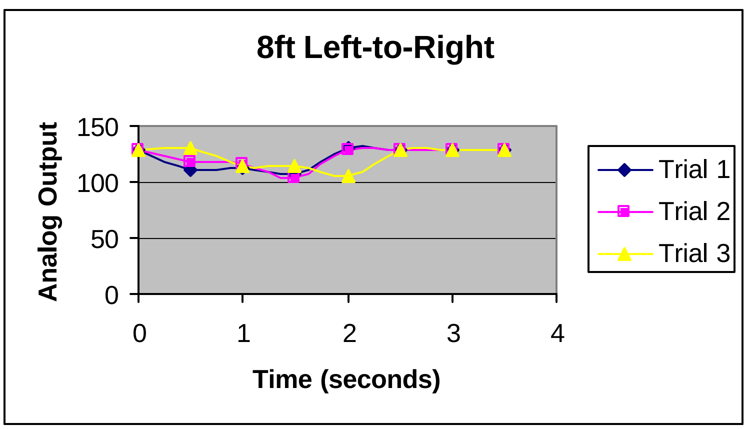


Figure 8 Pyro at 5'

At a distance of 5 feet, there is still noticeable difference between the separate directions, however they are not as tolerant as before. The left-to-right graph drops below the nominal 128 and the right-to-left rises above the nominal value. In both graphs there is at least one anomaly where the output goes in the opposite direction before going in the expected direction. Human detection is obviously possible at the distance however determining the direction of the individual may not be as accurate as before.



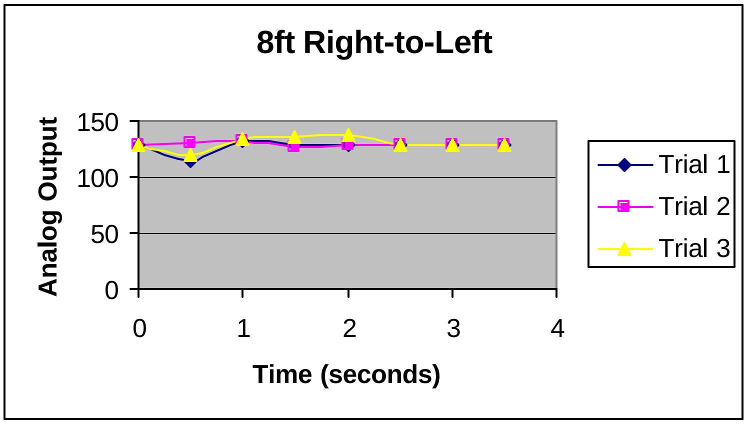


Figure 9 Pyro at 8 ft

At 8 feet, the difference in data is too obscure to determine accurately the direction of motion. Human detection is the only reliable attribute at this distance.

Trying to detect the presence of an individual can be done reliably at any distance between 6inches and 8 feet. However direction of motion should be done between the approximate distances of 3 to 5 feet. This may be accomplished my setting minimum and maximum detection values to create a range of tolerances that can be used to determine whether direction can be accurately detected.

The problem I found when trying to do people following was that the pyro-sensor has a large window to detect motion in and it has no depth perception. To compensate for these faults I columnated the sensor with a 1 ¼'" card stock tube. This shrunk the window size of the sensor and made the system more accurate when locating people. I then used the pyro-sensor together with the I.R. sensors to locate a person, go towards that person and verify that it has indeed found a person.

Speech Recognition System

The Voice Direct 364 speech recognition kit is utilized to initiate the beverage dispensing process. It is set up for single word continuous listening mode, which means it listens for one key word to be recognized. Some of the major disadvantages I have found using this system is that the microphone must be positioned the same distance from the speakers' mouth during recognition as it was during training. Any differences in the inflection of word during training and recognition will also cause the system not to recognize the word. During testing I had a cold and could not get the unit to recognize anything I said. Background noises will also cause the system not to recognize commands.

The key word recognition seems to be more liberal than the command word. If only one command is required, the command word can be left blank (by not training it) and the output will toggle high when it recognizes the key word. This has helped to have greater recognition accuracy. Figure 10 demonstrates the wiring diagram for continuous listening mode.

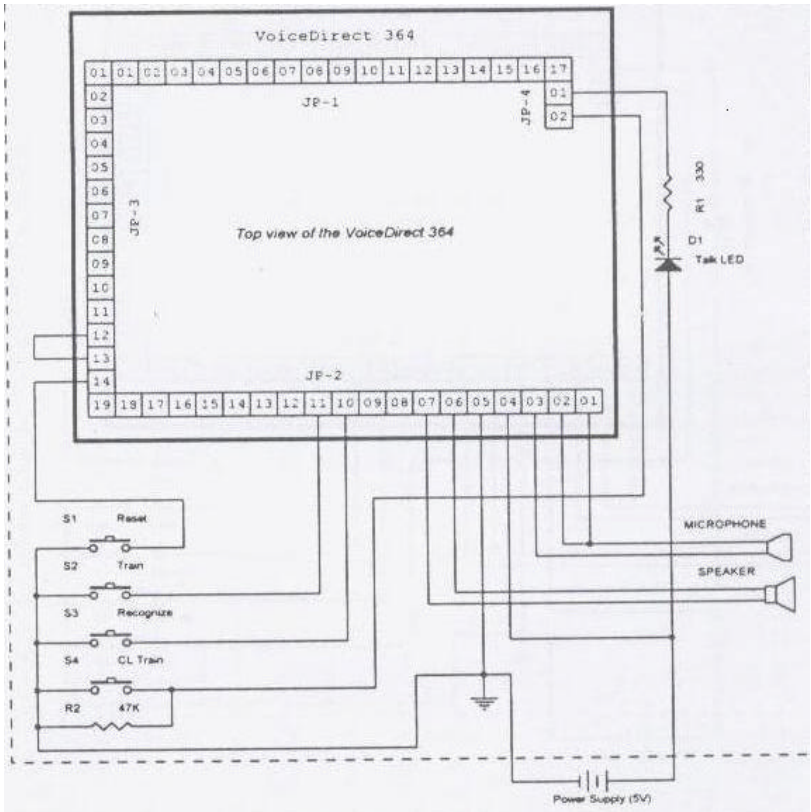


Figure 10 Continuous Listening

Voice Playback

The ISD 2560 was used for voice playback. It can play up to 60 seconds worth of messages sampled at 8kHz. It is capable of message cueing via mode 0 in any order. I use a breadboard setup when recording messages, as this simplifies control of the I.C. Figure 11 demonstrates connections to the ISD2560.

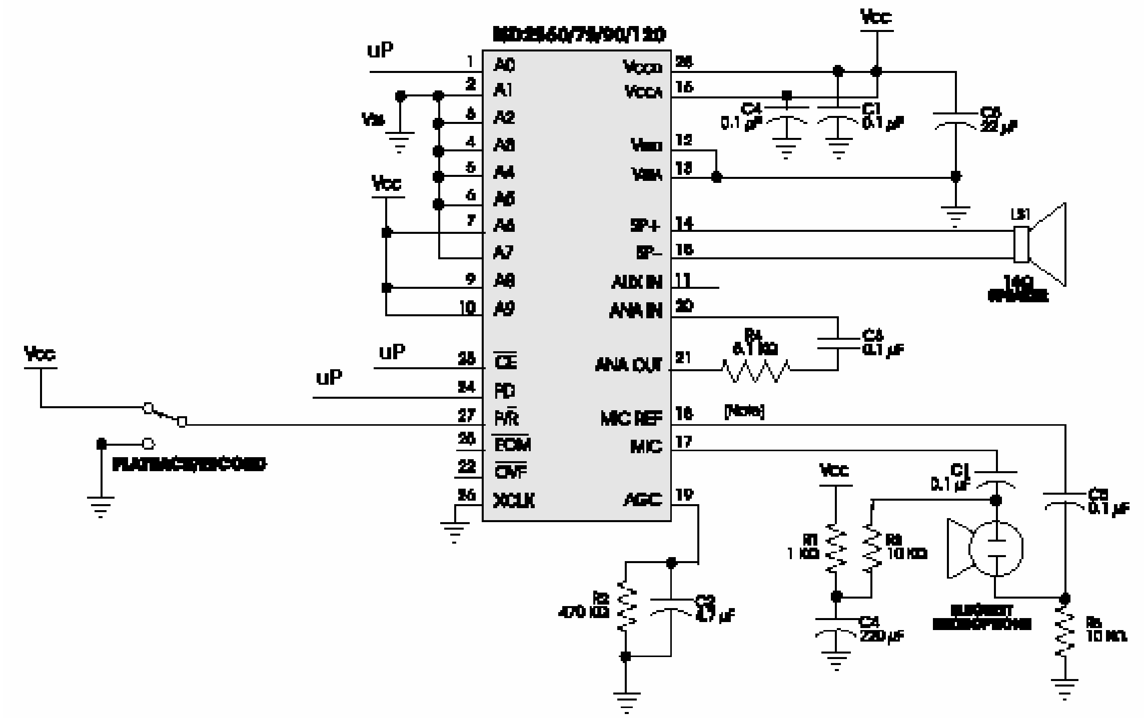


Figure 11 ISD2560

Behaviors

Upon start up, B.E.E.R.-Bot goes into a start up phase where it waits for the pyroelectric sensor to warm up, this takes approximately 20 seconds. Once the warm up phase is complete, one of B.E.E.R.-Bots' motors activates, causing it to go in a circle. This is its' search mode, where it uses the pyroelectric sensor to locate a person. If no one is found within 15 seconds, B.E.E.R.-Bot goes into obstacle avoidance mode for approximately 30 seconds, wandering around randomly. It then goes bask into search mode.

If someone is detected while in search mode, B.E.E.R.-Bot immediately exits this mode and goes towards the person. It then uses its' I.R sensors to detect the person, as an obstacle. If an obstacle is detected, B.E.E.R.-Bot turns towards the obstacle using the pyro-sensor to verify that it is indeed a person. If a positive verification is established B.E.E.R.-Bot asks if the individual would like a beverage. If the voice recognition system detects a "yes" answer, it then waits while the user pours a beverage from the tap. B.E.E.R.-Bot periodically polls the individual and waits for another "yes" response before leaving and going back into obstacle avoidance mode.

If any of the previous stages come back with a negative response, such as no person found or not getting a positive response from its offer, B.E.E.R.-Bot goes into obstacle avoidance mode and then starts over again. The behavior algorithm is demonstrated in the flow chart of figure 12.

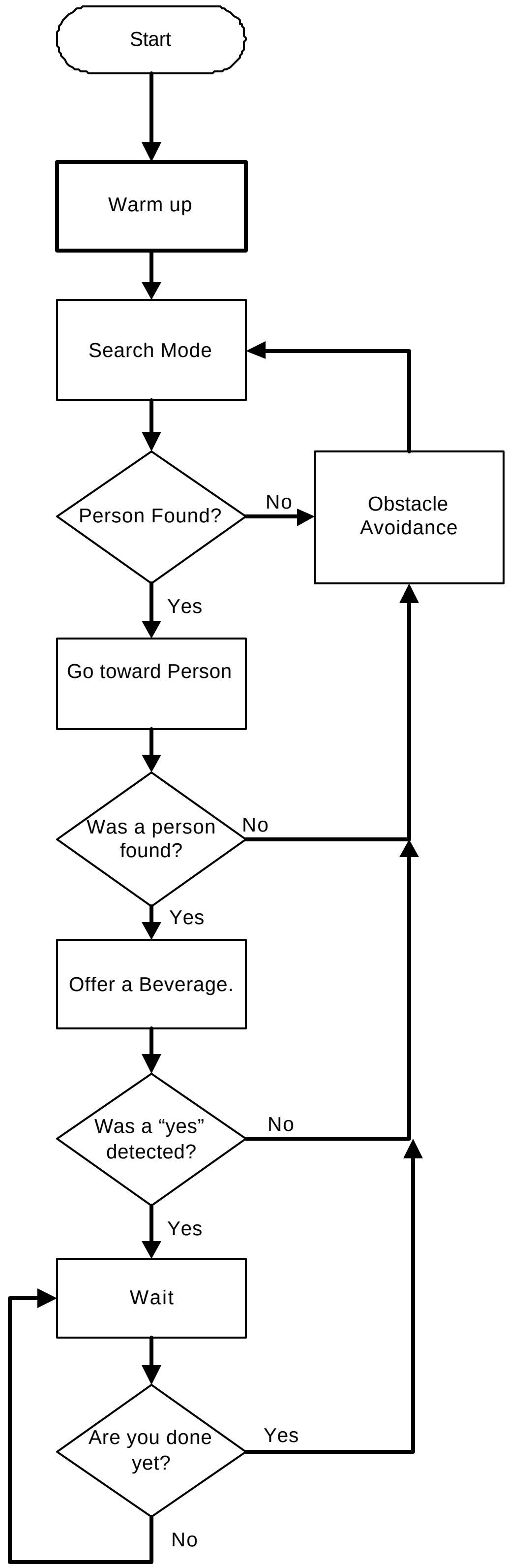


Figure 12 Behavior flow chart

Experimental Layout and Results

I attempted several different ways of following/locating people using the pyroelectric sensor.

The first attempt used a piece of modified code from the “Eltec Pyroelectric Sensor Package”. The subroutine is listed in the appendix as Follow. The code turned the robot left if the pyro-sensor read greater than 134 and turned right if the sensor read less than 122. This code is problematic in the sense that the persons’ relative position changes as the robots direction changes. This procedure caused a very jerky motion that was unacceptable for the objective of B.E.E.R.-Bot.

I then tried turning the robot until the pyro-sensor detected a person. When it detected a person it would stop turning and go forward towards the person. The problem here was that the window size is too large and the robot would stop before it was centered on the person.

I took a small piece of card stock with a thin slit cut out of the middle to reduce the window size. This worked, but proved to be unreliable as the sensor had to be perfectly aligned with slit in the card. Simple movements of the robot would jar the sensor out of alignment.

I then columnated the sensor as described in the sensor section of this report and that seemed to work perfectly.

Conclusion

In the future, B.E.E.R.-Bot will dispense beverages on its own and may incorporate a video camera to better interact with its environment. From listening to others experiences, however, it will not be the CMU cam. I would also like to make the platform bigger and enclose the electronics so it is not exposed to getting wet.

B.E.E.R.-Bot was a challenging and thought-provoking project. I became very familiar with the PIC microcontroller and PIC Basic. Most of the class used the ATMEL microcontroller and from listening to them, it sounds like the ATMEL is a user-friendlier device.

This class was also great for tying the concepts together from most of my previous course work and inspiring confidence in my engineering ability. I found that almost any device I could imagine is already out on the market; I only had to adapt it to my needs.

Documentation

The following were purchased from Jameco Electronics www.jameco.com:

- PIC16F877
- PicProto64
- Voice Direct 364
- ISD 2560 Voice Record/Playback Device
- Assorted electronic connectors, switches and parts

The following were purchased from Acroname www.acroname.com:

- Sharp GP2D12 Detector Package
- Eltec 442-3 Pyroelectric Sensor Package

The following were purchased from www.listermann.com:

- Philtap for Mini-Kegs
- 5 liter Mini-Keg
- CO_2 cartridges

The following were purchased from Lowe's Hardware (352) 376-9900:

- 12" round ¾" birch (platform)
- Small sprinkler head PVC (bump switches)
- Spray paint
- Sliding door guide (bump switch mount)

The motors were purchased form SanteFe Auto Parts (352) 372-2588

The shaft couplers for mounting the wheels were purchased from www.mastercarr.com

Motor drivers were purchased from www.tecel.com

Wheels were purchased at Target (bicycle training wheels)

The Gator hat was purchased at Center Stage Costumes and Magic (352) 374-4334

Thank you to Jeno Nagey for allowing me to use his code for the LCD display.

Appendix A - Program Code

```
DEFINE          ADC_BITS        8       ' Set number of bits in result
DEFINE          ADC_CLOCK       3       ' Set clock source (3=rc)
DEFINE          ADC_SAMPLEUS            50      ' Set sampling time in uS



ADleft  VAR     BYTE
ADCntr VAR BYTE
ADRight         VAR     BYTE
Pyro    VAR BYTE
LCD             VAR     PORTB.0
dutyl   VAR     WORD            ' left Duty cycle value (CCPR1L:CCP1CON<5:4>)
dutyr   VAR     WORD            ' Right Duty cycle value (CCPR1L:CCP1CON<5:4>)
trntime VAR BYTE                ' random time variable'
trndir  VAR BIT
Cntr    VAR     WORD
Lbump   VAR     BIT
Rbump   VAR     BIT
Bbump   VAR     BIT
BckTime         VAR     WORD
IDtime  VAR     WORD
ID              VAR     BIT
RecOut VAR BIT
RecTime VAR WORD
Cruisetime VAR WORD
pour    VAR     BIT
pspin   VAR     WORD



        TRISC.2 = 0                             ' Set PORTC.2 (CCP1) to output
        CCP1CON = %00001100                     ' Set CCP1 to PWM
        T2CON = %00000101               ' Turn on Timer2, Prescale=4

        TRISC.1 = 0                             ' Set PORTC.1 (CCP2) to output
        CCP2CON = %00001100                     ' Set CCP2 to PWM

        PR2 = 249                               ' Set PR2 to get 1KHz out

        dutyr = 30                              ' Set duty cycle
        dutyl = 120

        CCP1CON.4 = dutyr.0             ' Store duty to registers as
        CCP1CON.5 = dutyr.1             ' a 10-bit word
        CCPR1L = DUTYr >> 2

        CCP2CON.4 = dutyl.0             ' Store duty to registers as
        CCP2CON.5 = dutyl.1             ' a 10-bit word
        CCPR2L = DUTYl >> 2

    Output  LCD
    Low     LCD
```

```
        Pause(50)
        SerOut LCD,2,[$FE,$01]
        TRISA = %11111111     ' Set PORTA to all input
        ADCON1 = %00000010 ' Set PORTA analog
        Pause 500              ' Wait .5 second


'**************************'
'****Start Main Program****'
'**************************'
loop2:  ADCIN 3, Pyro
                GoSub  DisPyro
                Pause 50
                While   pyro < 126        'wait til pyro warms up before starting'
                GoTo    loop2
                Wend
loop3:
                GoSub  pfind
                GoSub  getperson
                GoSub  Identify
                IF ID = 1 Then
                        GoSub offer
                Else
                        GoSub  message4
                        GoSub  ObsAvoid
                EndIF
                IF pour = 1 Then
                        GoSub  waitn
                Else
                EndIF
                GoSub  ObsAvoid
                GoTo    loop3
'**************************'
End     '****End Main**********'
'**************************'


'  Subroutine writes left and right IR values to LCD'
DisplayLR:
                SerOut  LCD,2,[$FE,$01] ' Clear Screen and return cursor
                Pause 10
                SerOut  LCD,2,["Left IR: ", #ADleft DIG 2, #ADleft DIG 1, #ADleft DIG 0]
                SerOut  LCD,2,[$FE,$C0] ' Set cursor to next line
                SerOut  LCD,2,["Right IR: ", #ADright DIG 2, #ADright DIG 1, #ADright DIG 0]
        Return

'Subroutine to display Pyro sensor values to LCD'
DisPyro:        ADCIN 3, Pyro            ' Read channel 0 to Pyro val
                        ADCIN 1, ADCntr

                SerOut  LCD,2,[$FE,$01] ' Clear Screen and return cursor
                Pause 10
                SerOut  LCD,2,["Pyro: ", #Pyro DIG 2, #Pyro DIG 1, #Pyro DIG 0]
        Return

'Suroutine to go forward'
```

```
Forward:
                GoSub  RmotorGo
                GoSub  LmotorGo
        Return

'Stop right motor'
RmotorStop:
                Low PORTB.5             'Rrelay off'
                Pause 10
                Low PORTB.3             'In1R low'
                Low PORTB.7             'IN2R low'
                Pause 10
        Return

'Start Right motor'
RmotorGo:
                Low PORTB.3             'In1R low'
                High PORTB.7  'IN2R high'
                Pause 10
                High PORTB.5  'Rrelay on'
                Pause 10
        Return

'left motor stop'
LmotorStop:
                Low PORTB.2             'Lrelay off'
                Pause 10
                Low PORTB.6             'IN1L low'
                Low PORTB.4             'IN2L low'
                Pause 10
        Return

'Start Left motor'
LmotorGo:
                Low PORTB.6             'IN1R low'
                High PORTB.4  'IN2R high'
                Pause 10
                High PORTB.2  'Rrelay on'
                Pause 10
        Return


'Subroutine for Obstacle Avoidance'
ObsAvoid:       Cruisetime = 110
        While Cruisetime > 0
                Cruisetime = Cruisetime - 1
                Pause 100
                        ADCIN 0, ADleft                 ' Read channel 0 to adval
                        ADCIN 1, ADCntr
                        ADCIN 2, ADright
                        Lbump = PORTD.0
                        Rbump = PORTD.1

        IF (ADCntr > 50) OR (Lbump = 1) OR (Rbump = 1)Then
                                Random trntime
                                Low PORTB.5             'Lrelay off'
```

```
                Low PORTB.2             'Rrelay off'
                Pause 500
        'left relay reverse'
                High PORTB.3  'IN1L high'
                Low PORTB.7             'IN2L low'
        'right relay reverse'
                High PORTB.6  'IN1R high'
                Low PORTB.4             'IN2R low'
                High PORTB.5  'Lrelay on'
                High PORTB.2  'Rrelay on'
                Bcktime = 1000
        While Bcktime > 0
                Pause 1
                Bcktime = Bcktime - 1
                Bbump = PORTD.2
                        IF Bbump = 1 Then
                                Bcktime = 0
                        Else
                        EndIF
        Wend

                trndir = trntime.1
                        IF trndir = 1 Then
                                Low PORTB.5             'Lrelay off'
                        Else
                                Low PORTB.2             'Rrelay off'
                        EndIF
                Pause 750
                Pause trntime
                Low PORTB.5             'Lrelay off'
                Low PORTB.2             'Rrelay off'
    EndIF
'Check left IR sensor'
        IF ADleft > 50 Then
                GoSub  RmotorStop
        Else
                GoSub  RmotorGo
        EndIF
'check right IR sensor'
        IF ADright > 50 Then
                GoSub  LmotorStop
                Else
                GoSub  LmotorGo
        EndIF
    Wend
    Return

'Try to find a person'
Pfind:      Pspin = 150


        ADCIN 0, ADleft                 ' Read channel 0 to adval
        ADCIN 1, ADCntr
        ADCIN 2, ADright
        ADCIN 3, Pyro
```

```
                GoSub Lmotorstop
                GoSub RmotorGo

                While (pyro > 118) AND (pyro < 138) AND (pspin > 0)
                        pspin = pspin - 1
                        Pause 100
                        ADCIN 3, Pyro
                        ADCIN 1, ADCntr
                        Lbump = PORTD.0
                        Rbump = PORTD.1



                        IF (ADCntr > 50) OR (Lbump = 1) OR (Rbump = 1)Then
                                pyro = 100
                        Else
                EndIF
                Wend
                GoSub RmotorStop

Return


'**goto Person**
getPerson:
keepgoing:      ADCIN 0, ADLeft                 ' Read channel 0 to adval
                ADCIN 1, ADCntr
                ADCIN 2, ADRight
                Lbump = PORTD.0
                Rbump = PORTD.1

                GoSub LmotorGo
                GoSub RmotorGo
        IF (ADCntr > 50) OR (ADLeft > 50) OR (ADRight >50)  Then
                GoSub RmotorStop
                GoSub Lmotorstop
        Else
                GoTo keepgoing
        EndIF
Return


'**Identify Person**
Identify:

        IF (ADLeft > 50) Then
                        GoSub RmotorGo
                Else
                        GoSub LmotorGo
        EndIF

                IDtime = 500
                While IDtime > 0
                        ADCIN 3, Pyro
                        IDtime = Idtime -1
```

```
                        Pause 2
                        IF (pyro < 118) OR (pyro > 138) Then
                                ID = 1
                                IDtime = 0
                        Else
                                ID = 0
                        EndIF
                Wend
        GoSub RmotorStop
        GoSub Lmotorstop
Return

'Offer a beer
offer:
        GoSub  message1
recout = 0
RecTime = 60000
RecOut = PORTC.7
        While (rectime > 0) AND (recout = 0)
                RecTime = (RecTime  - 1)
                RecOut = PORTC.7
        Wend

        IF RecOut = 1  Then
                GoSub message2
                pour = 1
        Else
                GoSub message3
                pour = 0
        EndIF
Return


waitn:
                recout = 0
                GoSub  message5
                Pause 8000
                GoSub  message7
                Pause 4000
                GoSub  message6
                While recout = 0
                        RecOut = PORTC.7
                Wend
                pour = 0
                recout = 0
Return


showIRnBump:
        SerOut  LCD,2,[$FE,$01] ' Clear Screen and return cursor
                Pause 10
                SerOut  LCD,2,["L: ", #Lbump," C: ", #ADCntr DIG 2, #ADCntr DIG 1, #ADCntr
DIG 0]
                SerOut  LCD,2,[$FE,$C0] ' Set cursor to next line
                SerOut  LCD,2,["R: ", #Rbump, "P: ", #Pyro DIG 2, #Pyro DIG 1, #Pyro DIG 0]
```

```
Return

'play message 1
message1:
        High PORTC.6
        Pause 100

        Low PORTC.6
        High PORTC.5
        Low PORTC.5
        High PORTC.5
        Pause 3000


Return

message2:

        High PORTC.6
        Pause 500
        High PORTC.5
        Pause 100
        Low PORTC.6


        Pause 100
        High PORTC.4
        Pause 500

        Low PORTC.5
        High PORTC.5
        Pause 100
        Low PORTC.4
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 3000


Return


message3:
        High PORTC.6
        Pause 500
        High PORTC.5
        Pause 100
        Low PORTC.6


        Pause 100
        High PORTC.4
        Pause 500

        Low PORTC.5
```

```
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100
        Low PORTC.4
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 3000
Return

message4:
        High PORTC.6
        Pause 500
        High PORTC.5
        Pause 100
        Low PORTC.6


        Pause 100
        High PORTC.4
        Pause 500

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100
        Low PORTC.4
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 3000
Return

message5:
        High PORTC.6
        Pause 500
        High PORTC.5
        Pause 100
        Low PORTC.6


        Pause 100
        High PORTC.4
        Pause 500
```

```
        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100
        Low PORTC.4
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 3000
Return

message6:
        High PORTC.6
        Pause 500
        High PORTC.5
        Pause 100
        Low PORTC.6


        Pause 100
        High PORTC.4
        Pause 500

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100
        Low PORTC.4
```

```
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 3000
Return

message7:
        High PORTC.6
        Pause 500
        High PORTC.5
        Pause 100
        Low PORTC.6


        Pause 100
        High PORTC.4
        Pause 500

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 100
        Low PORTC.4
        Pause 100

        Low PORTC.5
        High PORTC.5
        Pause 3000
Return
```

Appendix B – Follow Sub Routine

```
'Subroutine for people following'
Follow:         ADCIN 0, ADleft                 ' Read channel 0 to adval
                    ADCIN 1, ADCntr
                    ADCIN 2, ADright
                    ADCIN 3, Pyro
        IF ADCntr > 50 Then
                        Random trntime
                        Low PORTB.5             'Lrelay off'
                        Low PORTB.2             'Rrelay off'
                        Pause 500
                    'left relay reverse'
                        High PORTB.3  'IN1L high'
                        Low PORTB.7             'IN2L low'
                    'right relay reverse'
                        High PORTB.6  'IN1R high'
                        Low PORTB.4             'IN2R low'
                        High PORTB.5  'Lrelay on'
                        High PORTB.2  'Rrelay on'
                        Pause 750
                        trndir = trntime.1
                            IF trndir = 1 Then
                                Low PORTB.5             'Lrelay off'
                            Else
                                Low PORTB.2             'Rrelay off'
                            EndIF
                        Pause 750
                        Pause trntime
                        Low PORTB.5             'Lrelay off'
                        Low PORTB.2             'Rrelay off'
                    EndIF

                    IF (ADleft > 50) OR (Pyro >134) Then
                        Low PORTB.5             'Lrelay off'
                            Pause 10
                            Low PORTB.3             'In1L low'
                            Low PORTB.7             'IN2L low'
                            Pause 10
                        Else

                        Low PORTB.3             'In1L low'
                            High PORTB.7  'IN2L high'
                            Pause 10
                            High PORTB.5  'Lrelay on'
                            Pause 10


                    EndIF

                    IF (ADright > 50) OR (Pyro < 122) Then
                            Low PORTB.2             'Rrelay off'
                            Pause 10
                            Low PORTB.6             'IN1R low'
                            Low PORTB.4             'IN2R low'
```

```
                Pause 10
            Else
            Low PORTB.6         'IN1R low'
                High PORTB.4    'IN2R high'
                Pause 10
                High PORTB.2    'Rrelay on'
                Pause 10


        EndIF
Return
```

Appendix C – LCD Display

To facilitate sensor testing and troubleshooting, an HD44780 controlled LCD display was connected to the microprocessor. This was accomplished using the EDE702 Serial LCD Interface IC.

The EDE702 provides serial control of the LCD display via a 9600 Baud rate data link from the micro controller. When power is applied, the EDE702 automatically clears the LCD, initializes the cursor and sets it to receive four-bit parallel input from the IC. This frees up an additional 6 lines on the processor and simplifies coding of the LCD initialization.

The following is a modified wiring schematic from the EDE702 on-line documentation, and demonstrates how I connected the LCD to the PIC 16F877: