Final Report
EEL5666 4/23/02
Justin Rice

Table of Contents

Abstract

My goal for this project was to build a pair of autonomous mobile robots capable of working together. In order to work together they need to be capable of locating and following the other robot, and have some sort of communication system. To demonstrate their capabilities, the Lemmings play a game of follow the leader. One robot will follow the other until the leader is trapped in a corner or hit by something. They will then switch roles, turn around, and continue moving. Since I had to build two robots, the Lemmings were both built using the same simple design and are modeled after the TJ.

Executive Summary

The Lemmings are a pair of autonomous robots controlled by a PIC microprocessor. I designed the board using a proto-board from Jameco and a PIC16F877. The PIC16F877 is made by Microchip and can be purchased from Jameco for $10. This PIC has 8K of internal flash ROM, 8 A/D input ports and runs at 8 MHz. All of the software is written using PIC BASIC PRO.

Actuation is provided by two Futaba S3003 servos per robot. The object avoidance and the following behaviors rely on three 38 kHz Sharp cans on each robot. A separate LM555 timer is used to generate the 38 kHz modulation. A simple communications system is implemented using a microphone and a LM386 audio amplifier. A bank of three LED's and a magnetic speaker provides feedback.

The Lemmings have identical hardware configurations. The only difference in the software is that one Lemming starts out as a leader, and the other Lemming is a follower. When the leader is hit by something or becomes trapped in a corner it beeps its speaker briefly and then turns around. The follower Lemming will hear the beep on its microphone circuit and also turn around. At this point the robots will both switch modes; the leader will become the follower and the follower becomes the leader and takes off in a new direction.

Introduction

My main goal in this course was to build a pair of autonomous robots that are capable of working together. Most (cheap) robots are only capable of one or two simple behaviors, and I wanted to create two robots that can coordinate their activities. I took EEL4744 last semester, so I was feeling confident about my abilities and decided to use a different chip (the PIC16F877) to control my robots. This decision worked out in the end, but slowed down the start of my project.

The Lemmings employ a combination of analog IR and sound to coordinate their movements and to communicate. I originally intended to use a Sharp GP2W0004YP IrDA transceiver for digital communications between the robots, but was unable to get the device working. The microphone/speaker system is a fallback idea to provide simple communications. The Lemmings are able to move about a room while staying fairly close to one another and the microphone allows them to avoid becoming trapped in corners.

Integrated System

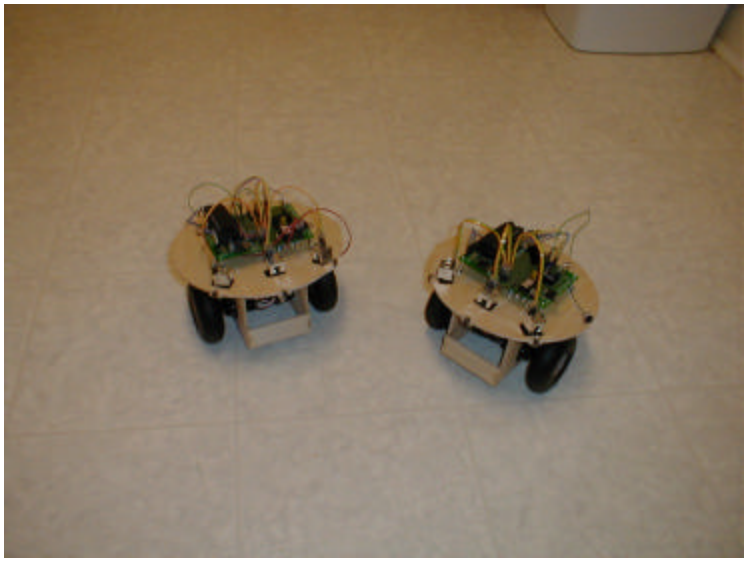
The Lemmings are based on the PIC16F877 microcontroller running on a prototyping board built by me. The PIC acts as the brains and control the robots based on inputs from the various sensors. The PIC contains 356 bytes of RAM and 8K of flash ROM. Two PWM pins allow for easy output waveform generation and built in A/D pins allow for sensor input. Bump sensors determine when an obstacle is hit, and IR is used to avoid objects at a distance. The same IR sensors are also used for following the leader using different software controls. The robots are also able to send simple messages using the sound from a magnetic speaker.

The Lemmings have two basic modes of operation: leader and follower. One robot will start out in each mode. The leader robot will wander around avoiding obstacles and broadcasting its location using IR. The follower robot will try to pick up the IR signal from the lead robot. When the signal is detected it will begin to follow the leader. The Lemmings then move around the room until the leader becomes trapped in a corner or runs into something that it could not detect. The robots then trade jobs after the leader has broadcast an alert using its speaker. The Lemmings then begin to advance in the opposite direction of the obstacle that confused them.

Mobile Platform

The hardware design for each Lemming is identical to allow them to easily change roles. The platform design is based on the TJ; basically a small circle of wood powered by two centrally mounted servos. The platform was designed in AutoCAD and cut out by the T-Tech machine. My circuit board is mounted on the top of the robots, and the sensors are located around the edge of the top surface. The battery pack is located at the rear; underneath the circuit board and directly above the ground to give the robot good balance. The rear of the robot drags on the ground, weighed down by the batteries. The central location of the servos allows the Lemmings to turn in place. Sharp turns are often required by the follower robot to keep pace with the leader. Some of the wiring can be hidden from sight by running it through an access hole cut in the center of the top circle of the robot. The proto board is placed directly above this hole. One problem with my design is that since many of my headers are located near the edges of my board there are some wires that cannot be hidden.

The Lemmings: Figure 1



Actuation

The only actuation required by the Lemmings is simple movement. This can be accomplished simply and fairly cheaply using two wheels driven by hacked servos. I used Futaba S3003 servos bought from www.servocity.com for $10 apiece. They provide 44.4oz-in of torque at 6 V. I soldered a wire to the fourth battery in my eight-pack and ran it to the servo power supply. This created an unregulated 5 V supply. I used the standard hack to cut away the potentiometer limits and make the servos freely rotate.

The PIC has two built in PWM pins, but I just used the PULSOUT command to generate the servo control signals. The PIC BASIC language has a PWM command, but nothing else can be done while this PWM is generated. I set my servos up using the PULSOUT command because that was the best way I saw to handle it at the time. To create the delay between pulse updates, I have a loop that checks my bump sensors and microphone readings for 20ms. Every 20 ms the motor values get updated. This is faster than is usually recommended, so I implemented the smoothing algorithm shown in class.

old_value = ((19*old_value)+desired)/20

I would have preferred to use a constant value much greater than 19, but the division command on the PIC can only handle integers. For constants greater than 20, the results of the division were often truncated and the motors would never reach the desired speeds.

Later I discovered that you could directly modify the settings on one of the internal timer registers and have a PWM output that is generated in the background. This would have been a cleaner solution.

Sensors

The sensors used by the Lemmings gives them basic obstacle detection and avoidance skills. The sensors also allow the Lemmings to follow each other and send simple messages.

Bump Sensors:

These are standard bump sensors on a voltage divider network used to determine when the robot has hit an object. There are three sensors at the front of each Lemming and one at the rear. A somewhat flexible ring of wood around the robot allows a bump to be registered even when the collision does not occur directly on a switch. The bump signal is read into analog pin A3.

Bump Sensor Data:

Bump Sensor	Resistor Value	Reading
Left	47 kΩ	44
Center	20 kΩ	78
Right	10 kΩ	128
Rear	100 kΩ	22

IR Sensors:

The IR sensors are used for two purposes: obstacle avoidance and following. Three hacked Sharp GP1U581Y IR detectors working at 38 kHz are used. The hack for the 38 kHz detectors is the same at the 40 kHz hack available on the IMDL web site. I was able to order these detectors from www.goldmine-elec.com for $1.50 each even though they have not been made in a few years. Figure 2 shows the response of the Sharp cans to obstacles placed a certain distance away.

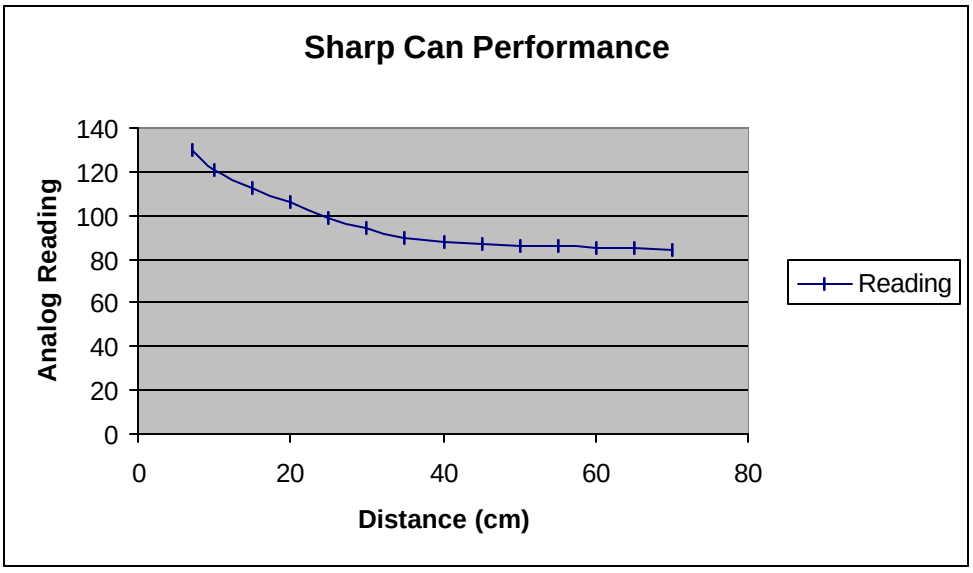


Figure 2

Obstacle avoidance requires that the sensors be located at the front of the robots and point in the direction of motion. However, since the second robot must be able to find the lead Lemming, a larger range of coverage is necessary. The center Sharp can looks directly forward. The left and right cans point about 60 degrees away from the center. The sides facing the center are slightly collimated to create more exclusive zones of detection. Figure 3 shows the final layout of the IR detectors.

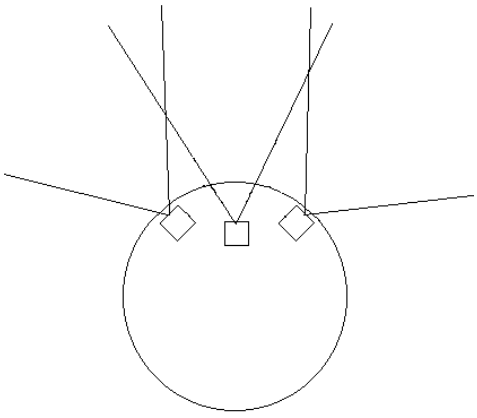


Figure 3

I used analog ports A0, A1, and A2 for the left, center, and right IR devices.

Each detector has an IR LED mounted underneath it t be used for object detection. These LED's are modulated using the output of a LM555 timer to 38 kHz (see Appendix A). The reset pin of the timer is tied to pin B3. When the Lemming is in follower mode, it shuts off the IR so that interfering signals are not generated. A separate IR LED is mounted facing the rear of the Lemming. It is modulated in software using a PWM pin.

This setup would allow the object avoidance and the following systems to be easily moved to different frequencies of IR if new detectors were incorporated. Putting these systems on different frequencies would give the Lemmings better performance, but time and money were limiting factors.

IrDA Communications:

My original goal was to implement a digital IR communication system that would not interfere with any of the other IR systems. The Sharp GP2W0004YP is potentially an excellent device for this purpose. It receives a serial input and outputs the data at speeds from 2.4 kb/s to 115.2 kb/s, modulated at 1.5 Mhz. I briefly had a pair of these devices working, and the performance was great. I was able to transmit data at a range of up to 1.5 meters within a 60-degree cone. There seemed to be no interference caused by fluorescent or incandescent lights. Unfortunately I connected my computer serial link to the same headers at one point and blew my IrDA chip. I spent a great deal of time working with this device, but I was unable to make the system work again.

I was able to get five free sample units by calling the Clearwater Sharp distributor. Marcus Amicci (AmicciM@sharpsec.com) noticed my order, and realized that the units are very tiny and require surface mounting. He realized that this would be a problem for me, so he sent me an additional five units that were pre-mounted on

development boards. All I needed to do was solder on some male headers instead of dealing with surface mount, so I really appreciated the extra effort from Marcus.

I contacted Sharp to see if an employee there could help me with my problems. Everyone I talked to was eager to help, and I quickly found someone who understood the device to help me. I spent a lot of time talking to Robert Stuart (Stuart@sharpsec.com), but I seemed to be doing everything correctly. Robert was able to send me a lot of information and help me better understand how the device operates, but we were unable to determine what I was doing wrong. Even though I was unable to get my chips working, I did notice several problems with their technical documentation that Robert was able to correct for future users.

Sound Detection:

Since I was unable to use the IrDA chips for communications, I needed a quick fix that would allow a simple form of messages. The main problem was that when the lead robot encounters an obstacle, sometimes it needs to turn around or stop. The follower Lemming would just keep on going and ram into the leader. I needed a way to alert the follower that the leader was stuck. I decided that a system of speakers and microphones would be suitable. All of the parts are easily available at Radioshack, which was a big plus given the small amount of time I had remaining after failing to get IrDA to work.

In order to develop a usable signal from the microphone output, it is necessary to use an amplifier. The book “Mobile Robots: Inspiration to Implementation” by Joseph Jones provides a simple amplifier circuit for a microphone using a LM386 audio op-amp (Figure 4). The output of the LM386 amplifier is read on analog pin A4. Using this

circuit, the speakers I am using generate a digital value of 240 at a distance of about ½ of a foot.

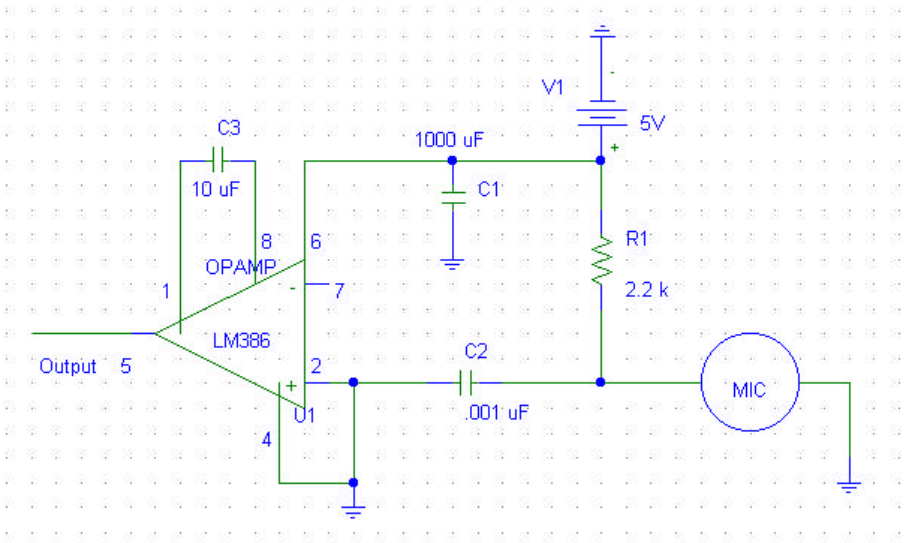


Figure 4

Behaviors

There are three behaviors programmed into each Lemming. The most basic is simple object avoidance. Lemmings are also able to follow a Lemming in lead mode. The last behavior is switching, where the Lemmings change modes based on sensory input.

Leading:

The lead Lemming will wander around a room at random while avoiding obstacles. If the leader gets hit by anything on any bumper, the switch mode will be triggered. The switch mode is also triggered if the leader gets stuck in a corner that causes all of the IR sensors to have very high readings. A red LED is lit while the robot is in leader mode.

Following:

The follower robot will first attempt to find the IR signal coming from the leader robot. A yellow LED is lit when the Lemming is in follow mode. The Lemming spins in a circle scanning the immediate area waiting for a leader to pass by. When some IR signal is detected from the leader, a green LED lights up and the follower Lemming begins to chase after the leader. The green LED remains lit as long as the lead Lemming can be detected. The follower also polls the microphone reading in between servo control pulses. If it detects a loud noise and it has the leader in sight, then it assumes that the leader is in trouble and enters the switching mode.

Switching:

The switching mode can be reached from either the lead mode or the follow mode. In either case the robot immediately sends out a beep alert on its speaker and makes a 180-degree turn in place. After another slight pause, the lead Lemming becomes a follower and the follower becomes a leader.

The purpose of this changeover is to allow the Lemmings to escape from certain environments that may otherwise cause problems. The Lemmings are not incredibly bright, and they may hit something that is too low to the ground to be seen with IR. They may also run into a corner and start panicking if all of the IR eyes see the same wall. In these cases, the following Lemming would normally just plow into the leader and keep pushing until something gave. This would probably be the servos used to power each Lemming since they are not very tolerant of high loads. The switching causes the Lemmings to head off in a new direction and avoid the source of the problem.

Experimental Layout and Results

I encountered two major problems during a test run was that the values I calculated for running my servos did not quite work as intended. The width of the pulse is supposed to be 1 ms for full reverse and 2 ms for full forward. With the pulsout command in PIC BASIC the following format is used:

PULSOUT pin,period

I am using pins B0 and B1 for my servo control. The period is in 5 us increments for a 8 MHz PIC. This means that to get a 1 ms pulse to pin B0 I would use the command: PULSOUT Portb.0, 200

1 ms = 200 * 5us 2 ms = 400 * 5us

In theory 200 would be full reverse and 400 would be full forward. All of my servos seem to be calibrated at the same positions, but none of them react as expected to these inputs. I created a small test program that would hold the right servo at a constant value, but vary the left servo based on the bumper you press. I also held the left servo constant and varied the right one based on bumper inputs.

Using this test program I was able to estimate approximate values for 100% forwards and backwards and for 50% forwards and backwards for the left and right servos.

Left Servo	Right Servo	Result
150	450	Full Reverse
200	450	Full Reverse
225	450	50% back on left wheel
250	450	Nearly 0% on left wheel
275	450	Slightly forward on left wheel
150	375	100% back on right wheel
150	350	100% back on right wheel
150	325	100% back on right wheel
150	300	100% back on right wheel

150	275	100% back on right wheel
150	270	90% back on right wheel
150	265	80% back on right wheel
150	260	50% back on right wheel
150	255	Slightly back on right wheel
150	250	0% on right wheel

From this data I was able to determine the values to use on each servo to move forwards, backwards, turn in place left, turn in place right, and to curve left and right.

Another major problem that I needed to solve was that the microphones are much too sensitive to noise using the default circuit. The 10 uF capacitor from pin 1 to pin 8 of the op-amp increases the gain to 200. This is far too much since the noise generated by the servos is often enough to swing the analog reading to full rail at 5 V.

A R/C circuit between pins 1 and 8 controls the gain on a LM386 op-amp. A resistor placed in series with the capacitor reduces the gain. The minimum gain can be reached with a large resistor or simply by leaving pins 1 and 8 open. This gain is 20, and is not enough amplification to make the speaker signal detectable. The speaker was placed about 2.5 feet from the microphone and the peak voltages were recorded for several resistors.

Resisor	Output
no resistor	4.4 V
100 Ω	4.0 V
220 Ω	3.4 V
270 Ω	3.1 V
no R/C circuit	2.6 V

The 270 Ω resistor decreases the gain enough to make the background noise negligible compared to the speaker signal, but still allow the speaker to generate a distinguishable signal. The modified speaker circuit can be seen in Figure 5.

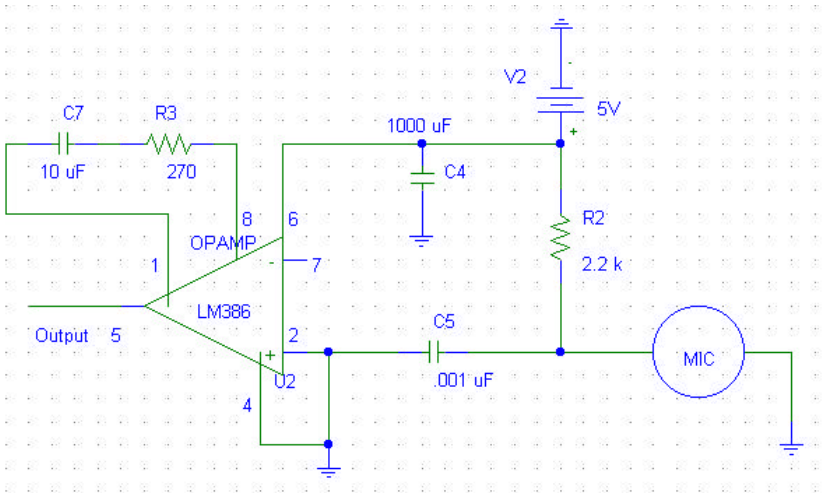


Figure 5

Conclusion

This project has been a great learning experience. I was mostly able to succeed in my goal of building two small robots that can work and play with each other. I was very disappointed that I was unable to make my IrDA communications device work for my project. It would have opened up a lot of new possibilities in the coordination of the Lemmings. The microphone back up plan is adequate for the simple uses I have implemented, but it is far less than what I was hoping to do. I have certainly gained valuable experience in dealing with hardware, software, and deadlines. I was hoping to avoid this part, but I also experienced failure. I was able to somewhat compensate for it, but it was still a dissapointment.

I am quite happy with the following and avoidance systems. The PIC processor proved to be a good choice. I had a slow start with it, but once I familiarized myself with the basic operation I was very happy with it. To me, its biggest selling points are the built in memory and I/O pins, and the Flash ROM. The ability to download a program once and have it run at any time is a very nice feature.

If I were to start from scratch I would keep the PIC processor and the board I designed, and very little else. I would completely scrap my platform. Now that I have some experience in building small robots I would attempt a unique design tailored to my needs. The TJ body shape served its purpose of simple and easy, but now I know that creating your own platform is not very difficult. The IR avoidance and following systems would have worked best if they were running on different frequencies. This would have eliminated the interference problems. I would take another shot at using the Sharp IrDA

chip because it has so much potential and I was able to get it working (briefly). With more time I think that I would be able to incorporate it into my current robots.

Documentation

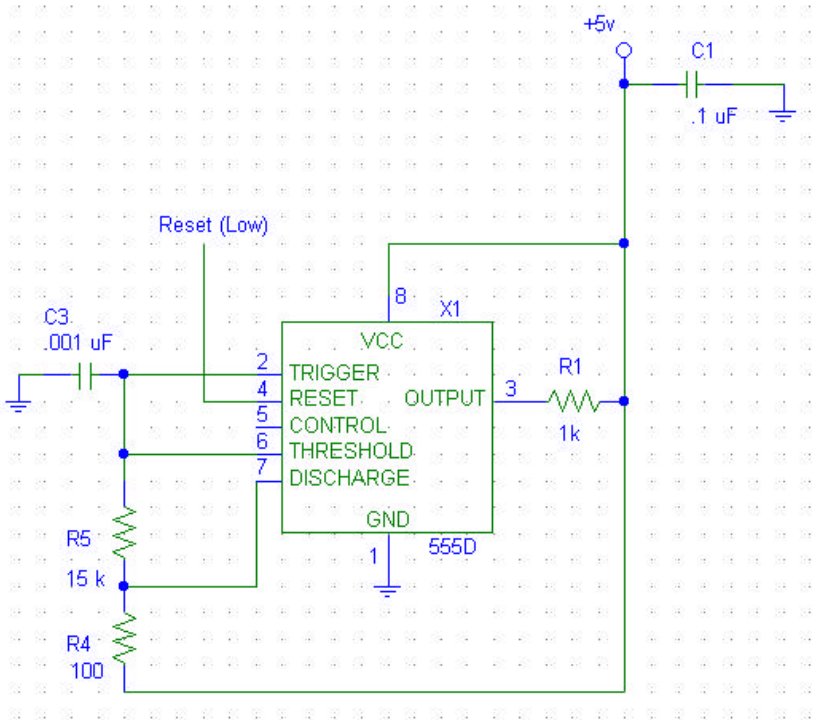
Basic microphone circuit taken from:

"Mobile Robots: Inspiration to Implementation" by Joseph L Jones, Anita M. Flynn, and Bruce A Seiger

Appendices

A) 555 timer circuit diagram

The output is taken at pin 3.



B) Sharp Contacts

Everyone I talked to at Sharp was very friendly and very willing to help.

Marcus Amicci: AmicciM@sharpsec.com

Marcus was able to have the GP2W0004YP development boards shipped to me. They were sent next day FedEx and actually came before the samples I ordered from the Sharp distributor in Clearwater which were ordered at least a week earlier. Marcus also put me in touch with Robert Stuart.

Robert Stuart: Stuart@sharpsec.com

Robert Stuart works on the opto-electronics at Sharp. He was very knowledgeable and helpful while I was trying to get the IrDA devices working. He also worked on the GP1 Sharp cans and was quite amused at the hack we go through to get the distance measurement for our robots.

C) PIC 16F877 Overview

40 pin DIP
External clocks between 4 MHz and 20 MHz supported.
28 I/O pins
    2 PWM
    built in serial Tx and Rx
    8 A/D inputs
8 K of flash memory
368 bytes of RAM

Basic Board Setup Costs

PIC	- $10
Crystal Oscillator	- $2 (at most)
PICProto64	- $17

2 resistors, 2 caps, 5V regulator – basically free

Total Cost per board = $29 + shipping (everything can be found at Jameco)

Since I had an external programmer for the PIC I invested in a $10 ZIF socket for each board. That is the socket type with the little lever so the chip can easily be removed.

The PICProto64 board is a 3”x4” prototyping board. There is room at the top for the PIC and the various components needed to provide power. The rest of the board is plated holes. I soldered groups of male headers and wire wrapped out the connections for all of my sensors. There are two ground bus lines, one on either side, and a power bus along the top. These boards are more expensive than something from Radioshack, but they were quite nice. If I use a PIC again, I will probably just make my own board from an empty Radioshack one, but this was definitely a worthwhile investment for my first time.

D) Leader Code

```
' Justin Rice
' Lemming Leader Program

'*****************************************************************************
' Serial Output Definitions
'*****************************************************************************

' define crystal speed at 8 MHz
DEFINE OSC 8

' define the serial pin to PortC bit 7
DEFINE DEBUG_REG PORTC
DEFINE DEBUG_BIT 6

' Define baud rate for serial debug
DEFINE DEBUG_BAUD 9600

' Define serial debug mode for inverted
DEFINE DEBUG_MODE 1

'*****************************************************************************
' A/D System Definitions

' A/D Clock Selection
'       00 = FOSC/2
'       01 = FOSC/8
'       10 = FOSC/32
'       11 = FRC (clock derived from the internal A/D module RC oscillator)

' A/D Channel Selection
'       000 = channel 0, (RA0/AN0)
'       001 = channel 1, (RA1/AN1)
'       010 = channel 2, (RA2/AN2)
'       011 = channel 3, (RA3/AN3)
'       100 = channel 4, (RA5/AN4)
'       101 = channel 5, (RE0/AN5)
'       110 = channel 6, (RE1/AN6)
'       111 = channel 7, (RE2/AN7)
'*****************************************************************************

' define number of bits in result
DEFINE ADC_BITS 8

' set clock source
DEFINE ADC_CLOCK 0

' sampling time in microseconds
DEFINE ADC_SAMPLEUS 1

' Set PORTA.0 - 4 as inputs
TRISA = %00001111

' Set PORTA to analog
ADCON1 = 2

'*****************************************************************************
' Digital Port Definitions
'*****************************************************************************
' Set PortB outputs
'Left Servo
TRISB.0=0
'Right Servo
TRISB.1=0
'Microphone
TRISB.2=0
'555 timer reset
TRISB.3=0
```

```
'Follower LED
TRISB.4=0
'Leader LED
TRISB.5=0
'Contact LED
TRISB.6=0

'****************************************************************************
' Constants
'****************************************************************************
FORWR100        CON     200
FORWL100        CON     285
BACKR100        CON     285
BACKL100        CON     200

FORWR50         CON     225
FORWL50         CON     260
BACKR50         CON     260
BACKL50         CON     225

'****************************************************************************
' Main Program
'****************************************************************************
IR_Left         VAR BYTE
IR_Right        VAR BYTE
IR_Center       VAR BYTE
IR_Zones        VAR BYTE
bump            VAR BYTE
mic             VAR BYTE
found           VAR BIT
rv              VAR WORD
lv              VAR WORD
rand            VAR WORD

oldr            VAR WORD
oldl            VAR WORD

mic=0

oldr = FORWR100
oldl = FORWL100

' looping variables
i VAR BYTE
j VAR BYTE

' 38 kHz for IR facing rear that robot 2 will follow
  TRISC.2 = 0  ' CCP1 (PortC.2 = Output)
  PR2 = 52     ' Set PWM Period for approximately 38KHz
  CCPR1L = 26  ' Set PWM Duty-Cycle to 50%
  CCP1CON = %00001100  ' Select PWM Mode
  T2CON = %00000100    ' Timer2 = ON + 1:1 prescale

' Turn on IR LED's
  High PORTB.3
  Pause 20


  'Reboot test
  High PORTB.4
  High PORTB.5
  High PORTB.6

' Delay before starting servos
  Pause 1000

  ' Turn on leader LED, turn off follower
  High PORTB.5
  Low PORTB.4
  Low PORTB.6
```

```
lead:

  ' Read in the left, right, and center IR values
  ' ADCIN channel, destination
  ADCIN 0,IR_Left
  ADCIN 2,IR_Right
  ADCIN 1,IR_Center


  ' Determine Reaction

  lv=FORWL100
  rv=FORWR100

  IF (IR_Center < 105) Then
    ' Curve to the left
    IF (IR_Right > 105) Then
      lv = BACKL50
      rv = FORWR100
    EndIF

    ' Curve to the right
    IF (IR_Left > 105) Then
      lv = FORWL100
      rv = BACKR50
     EndIF
  EndIF

  IF (IR_Center > 105) Then
    IF (IR_Center > 120) Then
      IF (IR_Left > IR_Right) Then
        ' Hard Right Turn
          lv = FORWL100
          rv = BACKR100
      Else
        ' Hard Left Turn
        lv = BACKL100
        rv = FORWR100
      EndIF
    Else
      IF ((IR_Left > 120) AND (IR_Right > 120)) Then
        ' Back Up
        High PORTB.2
        Pause 800
        Low PORTB.2
        Pause 200
        GoSub turn180

        ' Turn on IR LED's
        High PORTB.3
        Pause 20

        ' Turn on leader LED, turn off follower
        High PORTB.5
        Low PORTB.4

      Else
        IF (IR_Right > 105) Then
          ' Hard Left Turn
          lv = BACKL100
          rv = FORWR100
        Else
          IF (IR_Left > 105) Then
            ' Hard Right Turn
            lv = FORWL100
            rv = BACKR100
          EndIF
        EndIF
      EndIF
    EndIF
  EndIF
```

```
  ' Motor Control

  oldr=((19*oldr)+rv)/20
  oldl=((19*oldl)+lv)/20

  PulsOut PORTB.0, oldl
  PulsOut PORTB.1, oldr

  i=0
ldelay:

    i=i+1
    ADCIN 3,bump

    IF (bump > 17) Then
      High PORTB.2
      Pause 800
      Low PORTB.2
      GoSub turn180

      ' Turn on IR LED's
      High PORTB.3
      Pause 20

      ' Turn on leader LED, turn off follower
      High PORTB.5
      Low PORTB.4

    EndIF

    IF (i<50) Then
      GoTo ldelay
    EndIF


    GoTo lead           ' Repeat forever
End


turn180:

' Turn off IR LED's
  Low PORTB.3
' Turn on follower LED, turn off leader
  Low PORTB.5
  High PORTB.4

For i=1 to 50
  PulsOut PORTB.0, FORWL100
  PulsOut PORTB.1, BACKR100
  Pause 20
Next i

i=0

Pause 500

oldl = FORWL100
oldr = BACKR100
j=0

follow:

    found=0
    IR_Zones=0

    ' Read in the left, right, and center IR values
    ' ADCIN channel, destination
    ADCIN 0,IR_Left
```

```
ADCIN 2,IR_Right
ADCIN 1,IR_Center

' Determine Reaction

lv=FORWL100
rv=BACKR100

IF (IR_Left > 90) Then
  IR_Zones = IR_Zones | %10000000
EndIF

IF (IR_Center > 90) Then
  IR_Zones = IR_Zones | %01000000
EndIF

IF (IR_Right > 90) Then
  IR_Zones = IR_Zones | %00100000
EndIF

IF ( (IR_Left > 90) AND (IR_Center > 90) AND (IR_Right > 90) ) Then
  IF (IR_Left >= IR_Right) Then
    ' Soft Left
    IR_Zones=%11000000
  EndIF
  IF (IR_Right >= IR_Left) Then
    ' Soft Right
    IR_Zones=%01100000
  EndIF
EndIF

' Hard Left
IF (IR_Zones=%10000000) Then
  lv=BACKL100
  rv=FORWR100
EndIF

' Soft Left
IF (IR_Zones=%11000000) Then
  lv=FORWL50
  rv=FORWR100
EndIF

' Go straight
IF (IR_Zones=%01000000) Then
  lv=FORWL100
  rv=FORWR100
EndIF

' Soft Right
IF (IR_Zones=%01100000) Then
  lv=FORWL100
  rv=FORWR50
EndIF

' Hard Right
IF (IR_Zones=%00100000) Then
  lv=FORWL100
  rv=BACKR100
EndIF

' Motor Control
oldr=((19*oldr)+rv)/20
oldl=((19*oldl)+lv)/20

PulsOut PORTB.0, oldl
PulsOut PORTB.1, oldr

IF ( (IR_Left > 90) OR (IR_Center > 90) AND (IR_Right > 90) ) Then
  found=1
  ' Light up contact LED
```

```
      High PORTB.6
    Else
      ' Turn off contact LED
      Low PORTB.6
    EndIF

fdelay:

    i=i+1
    ADCIN 3,bump

    IF (bump < 49) AND (bump > 39) Then
      ' Go back left
      For i=1 to 50
        PulsOut PORTB.0, BACKL50
        PulsOut PORTB.1, BACKR100
        Pause 20
      Next i
    EndIF

    IF (bump < 133) AND (bump > 123) Then
      ' Go back right
      For i=1 to 50
        PulsOut PORTB.0, BACKL100
        PulsOut PORTB.1, BACKR50
        Pause 20
      Next i
    EndIF

    IF (bump < 85) AND (bump > 75) Then
      ' Go back
      For i=1 to 50
            PulsOut PORTB.0, BACKL50
            PulsOut PORTB.1, BACKR100
            Pause 20
         Next i
    EndIF

    IF (bump < 27) AND (bump > 17) Then
      ' Go forward
      For i=1 to 50
            PulsOut PORTB.0, FORWL100
            PulsOut PORTB.1, FORWR100
            Pause 20
         Next i
    EndIF

    ' Read microphone value
    mic=0
    ADCIN 4,mic

    IF ( (mic>210) AND (found=1) ) Then
        Pause 500

        ' Turn off contact LED
        Low PORTB.6
        found=0

        Return
    EndIF

    IF (i<20) Then
      GoTo fdelay
    EndIF

        GoTo follow           ' Repeat while following

Return
```

E) Follower Code

```
' Justin Rice
' Lemming Follower Program

'****************************************************************************
' Serial Output Definitions
'****************************************************************************

' define crystal speed at 8 MHz
DEFINE OSC 8

' define the serial pin to PortC bit 7
DEFINE DEBUG_REG PORTC
DEFINE DEBUG_BIT 6

' Define baud rate for serial debug
DEFINE DEBUG_BAUD 9600

' Define serial debug mode for inverted
DEFINE DEBUG_MODE 1

'****************************************************************************
' A/D System Definitions

' A/D Clock Selection
'       00 = FOSC/2
'       01 = FOSC/8
'       10 = FOSC/32
'       11 = FRC (clock derived from the internal A/D module RC oscillator)

' A/D Channel Selection
'       000 = channel 0, (RA0/AN0)
'       001 = channel 1, (RA1/AN1)
'       010 = channel 2, (RA2/AN2)
'       011 = channel 3, (RA3/AN3)
'       100 = channel 4, (RA5/AN4)
'       101 = channel 5, (RE0/AN5)
'       110 = channel 6, (RE1/AN6)
'       111 = channel 7, (RE2/AN7)
'****************************************************************************

' define number of bits in result
DEFINE ADC_BITS 8

' set clock source
DEFINE ADC_CLOCK 0

' sampling time in microseconds
DEFINE ADC_SAMPLEUS 1

' Set PORTA.0,1,2 as inputs, PORTA.3-7 as outputs
TRISA = %00001111

' Set PORTA to analog
ADCON1 = 2

'****************************************************************************
' Digital Port Definitions
'****************************************************************************
' Set PortB outputs
'Left Servo
TRISB.0=0
'Right Servo
TRISB.1=0
'Microphone
TRISB.2=0
'555 timer reset
TRISB.3=0
```

```
'Follower LED
TRISB.4=0
'Leader LED
TRISB.5=0
'Contact LED
TRISB.6=0

'****************************************************************************
' Constants
'****************************************************************************
FORWR100        CON     200
FORWL100        CON     285
BACKR100        CON     285
BACKL100        CON     200

FORWR50         CON     225
FORWL50         CON     260
BACKR50         CON     260
BACKL50         CON     225

'****************************************************************************
' Main Program
'****************************************************************************
IR_Left         VAR BYTE
IR_Right        VAR BYTE
IR_Center       VAR BYTE
IR_Zones        VAR BYTE
bump            VAR BYTE
mic                     VAR BYTE
found           VAR BIT
rv              VAR WORD
lv              VAR WORD
rand            VAR WORD

oldr            VAR WORD
oldl            VAR WORD

mic=0

oldr = BACKR100
oldl = FORWL100

' looping variables
i VAR BYTE
j VAR BYTE

' 38 kHz for IR facing rear that robot 2 will follow
TRISC.2 = 0  ' CCP1 (PortC.2 = Output)
PR2 = 52     ' Set PWM Period for approximately 38KHz
CCPR1L = 26  ' Set PWM Duty-Cycle to 50%
CCP1CON = %00001100  ' Select PWM Mode
' Turn on bit 2 to turn on IR
T2CON = %00000100    ' Timer2 = ON + 1:1 prescale

' Turn off IR LED's
  Low PORTB.3
  Pause 20

'Reboot test
  High PORTB.4
  High PORTB.5
  High PORTB.6

' Delay before starting servos
  Pause 1000

' Turn on follower LED, turn off leader
  Low PORTB.5
  High PORTB.4

follow:
```

```
i=0
j=0

found=0
IR_Zones = 0

' Read in the left, right, and center IR values
' ADCIN channel, destination
ADCIN 0,IR_Left
ADCIN 2,IR_Right
ADCIN 1,IR_Center

' Determine Reaction

lv=FORWL100
rv=BACKR100

IF (IR_Left > 90) Then
  IR_Zones = IR_Zones | %10000000
EndIF

IF (IR_Center > 90) Then
  IR_Zones = IR_Zones | %01000000
EndIF

IF (IR_Right > 90) Then
  IR_Zones = IR_Zones | %00100000
EndIF

IF ( (IR_Left > 90) AND (IR_Center > 90) AND (IR_Right > 90) ) Then
  IF (IR_Left >= IR_Right) Then
    ' Soft Left
    IR_Zones=%11000000
  EndIF
  IF (IR_Right >= IR_Left) Then
    ' Soft Right
    IR_Zones=%01100000
  EndIF
EndIF

' Hard Left
IF (IR_Zones=%10000000) Then
  lv=BACKL100
  rv=FORWR100
EndIF

' Soft Left
IF (IR_Zones=%11000000) Then
  lv=FORWL50
  rv=FORWR100
EndIF

' Go straight
IF (IR_Zones=%01000000) Then
  lv=FORWL100
  rv=FORWR100
EndIF

' Soft Right
IF (IR_Zones=%01100000) Then
  lv=FORWL100
  rv=FORWR50
EndIF

' Hard Right
IF (IR_Zones=%00100000) Then
  lv=FORWL100
  rv=BACKR100
EndIF
```

```
    ' Motor Control
    oldr=((19*oldr)+rv)/20
    oldl=((19*oldl)+lv)/20

    PulsOut PORTB.0, oldl
    PulsOut PORTB.1, oldr

    IF ( (IR_Left > 90) OR (IR_Center > 90) OR (IR_Right > 90) ) Then
      found=1
      ' Light up contact LED
      High PORTB.6
    Else
      ' Turn off contact LED
      Low PORTB.6
    EndIF

fdelay:

    i=i+1
    ADCIN 3,bump

    IF (bump < 49) AND (bump > 39) Then
      GoSub backleft
    EndIF

    IF (bump < 133) AND (bump > 123) Then
      GoSub backright
    EndIF

    IF (bump < 85) AND (bump > 75) Then
      GoSub back
    EndIF

    IF (bump < 27) AND (bump > 17) Then
      GoSub backbump
    EndIF

    ' Read microphone value
    ADCIN 4,mic

    IF ( (mic>175) AND (found=1) ) Then
        Pause 300

        ' Turn off contact LED
        found=0
         Low PORTB.6

        GoSub turn180
        ' Turn off IR LED's
         Low PORTB.3
         Pause 20

         ' Turn on follower LED, turn off leader
         Low PORTB.5
         High PORTB.4
    EndIF

    IF (i<20) Then
      GoTo fdelay
    EndIF

    GoTo follow           ' Repeat forever

End



backleft:
For i=1 to 50
  PulsOut PORTB.0, FORWL50
  PulsOut PORTB.1, BACKR100
```

```
  Pause 20
Next i
Return


back:
For i=1 to 50
  PulsOut PORTB.0, BACKL50
  PulsOut PORTB.1, BACKR100
  Pause 20
Next i
Return

backright:
For i=1 to 50
  PulsOut PORTB.0, BACKL100
  PulsOut PORTB.1, FORWR50
  Pause 20
Next i
Return

backbump:
For i=1 to 50
  PulsOut PORTB.0, FORWL100
  PulsOut PORTB.1, FORWR100
  Pause 20
Next i
Return

turn180:
For i=1 to 50
  PulsOut PORTB.0, FORWL100
  PulsOut PORTB.1, BACKR100
  Pause 20
Next i

' Turn on IR LED's
  High PORTB.3
  Pause 20
' Turn on leader LED, turn off follower
  High PORTB.5
  Low PORTB.4

lead:

  ' Read in the left, right, and center IR values
  ' ADCIN channel, destination
  ADCIN 0,IR_Left
  ADCIN 2,IR_Right
  ADCIN 1,IR_Center


  ' Determine Reaction
  lv=FORWL100
  rv=FORWR100

  IF (IR_Center < 105) Then
    ' Curve to the left
    IF (IR_Right > 110) Then
      lv = BACKL50
      rv = FORWR100
    EndIF

    ' Curve to the right
    IF (IR_Left > 110) Then
      lv = FORWL100
      rv = BACKR50
     EndIF
  EndIF

  IF (IR_Center > 105) Then
```

```
    IF (IR_Center > 120) Then
      IF (IR_Left > IR_Right) Then
        ' Hard Right Turn
          lv = FORWL100
          rv = BACKR100
      Else
        ' Hard Left Turn
        lv = BACKL100
        rv = FORWR100
      EndIF
    Else
      IF ((IR_Left > 120) AND (IR_Right > 120)) Then
        High PORTB.2
        Pause 800
        Low PORTB.2
        Pause 200
        Return
      Else
        IF (IR_Right > 105) Then
          ' Hard Left Turn
          lv = BACKL100
          rv = FORWR100
        Else
          IF (IR_Left > 105) Then
            ' Hard Right Turn
            lv = FORWL100
            rv = BACKR100
          EndIF
        EndIF
      EndIF
    EndIF
  EndIF

  ' Motor Control
  oldr=((19*oldr)+rv)/20
  oldl=((19*oldl)+lv)/20

  PulsOut PORTB.0, oldl
  PulsOut PORTB.1, oldr

  i=0
ldelay:

    i=i+1
    ADCIN 3,bump

    IF (bump > 17) Then
      High PORTB.2
      Pause 800
      Low PORTB.2
      Return
    EndIF

    IF (i<50) Then
      GoTo ldelay
    EndIF


GoTo lead           ' Repeat forever

Return
```