LOW-LEVEL AI VISION ROBOT: FINAL REPORT

DANIEL CHAN • TOMMY LEUNG
DECEMBER 15, 2003

ABSTRACT

This project implements a rudimentary robotic vision algorithm using two infrared distance sensors and Motorola's PIC 18F452 processor. Using the Lego Mindstorms robot as a portable platform to house all electronic components, the team mounted the sensors to the front and back of the robot and controlled the entire contraption with the PIC. This setup gives the robot a 360-degree field-of-view, along a two-dimensional plane about six inches above the floor. To demonstrate this, the robot has instructions to constantly maintain a half-meter circle of space around itself, running away from anything that comes too close. Though there are blind spots, due to the robot being slightly wider than the sensor's field of view, overall, the vision and escape algorithm turned out successful.

INTRODUCTION

Artificial intelligence vision, ranging from the ability to distinguish whether an object is present or not to mapping out entire rooms in three dimensions, remains one of the largest areas of robotics research. The ability to create self-oriented robots capable of navigating unknown areas has always proved a challenge, because of problems related to knowing and controlling exactly "how much" or "how far" a robot moves or turns.

In order to create the groundwork for what will hopefully become a more elaborate research project in the area of AI vision, this team implemented a low-level vision system on a Lego Mindstorms robot. To demonstrate the vision algorithm, the team created an object-avoiding robot controlled by Motorola's PIC 18F452 microprocessor. Interfaced with two Sharp GP2D12 infrared distance sensors, a Xilinx XCS10 FPGA, and mechanical parts from Lego, the robot keeps a 0.5 ± 0.1m radius of space around itself at all times. If it cannot find an escape route within a preset time, it will flash a set of onboard LEDs.

SUBSYSTEMS

LEGOS

The robot's mechanical parts are completely based off of Legos. Using two 9V Lego motors, the team experimented with several designs to come up with the best balance between weight, stability, and speed. The final design resembles that of a tank. Power from the motor is directly transferred to the treads via a chain, so that no power is lost to gears. Although the treads are slightly tight and tend to pull the axles inward too strongly, overall, the final mechanical design works.

The team encountered several weight and speed issues. Several times, the robot couldn't move under its own weight when solely powered by batteries. Several other times, the robot could move backward and forward, but not turn. Once the robot started moving as expected, vibrations would knock the batteries or the protoboard off. To resolve these problems, the team adopted a minimalist design with a wide platform and a low profile, and later physically tied down all electronic components to the robot.

IR BROADCAST AND DETECTION SYSTEM

The IR sensors continuously report the distance of the object closest to it. Although the sensor runs off 5V, its output only

swings from 0.2 to 2.4V, and the output voltage exponentially drops as distance increases. These factors led the team to try to increase the resolution of the distance measurement by changing the A/D unit's reference voltages with a voltage divider. However, it turns out that whenever an input channel to the A/D unit saturates, the entire unit begins to misbehave. The team first encountered this problem while using a potentiometer to simulate sensor readings. At the lowest and highest potentiometer settings, the A/D unit began to unreliably report voltage levels from all A/D channels. To remedy this, ground and Vdd were set as the reference voltages, and in a series of informal tests throughout the rest of the project, it turned out that initial concerns about a lack of resolution were unfounded.

EDGE DETECTION

A previous object-avoiding robot met its inevitable doom when it flew off a set of stairs as it tried to escape from people chasing it on both sides. To prevent this from happen to this robot, the team gave it ground sensors to help it detect steep declines. The sensors are tied to a high-priority interrupt on the PIC, which immediately suspends all motor activity as soon as the interrupt is tripped.

Comprised of two conducting Lego pieces (made conductive by creative wire stripping and attachment), these sensors cause the FPGA to output a rising edge whenever the robot reaches a ledge. To accomplish this, one Lego piece stays connected to a pull-down resistor, while another Lego piece is tied to Vdd. When the robot is about to fall over a one-inch tall (or higher) decline, the Lego rod falls, completing the circuit. The robot has two of these sensors, one in front and one in back. When either sensor trips, the FPGA sends a high priority interrupt to the PIC., which then jumps to a stop routine. To prevent the rod from catching on a Lego piece when it should be falling, the team attached an additional Lego piece to guide the rod's falling motion.

Edge detection presented one of the greatest challenges for this project. The team tried several different sensor designs before finding one with a reasonable weight, minimal impact on movement, and decent reaction time. Failed designs that were also tried include a spring-based sensor that "sprung up" when an edge was detected (too heavy, greatly impacted turning), mechanical push buttons (range of motion was not great enough), and paper-clipped Legos (not enough flexibility in the paper-clips to securely wrap around Lego pieces).

ROBOT MOVEMENT CONTROL

The robot movement control subsystem is responsible for driving the motors and interpreting commands from the robot escape algorithm. It utilizes the L293 Motor Driver, which has two on-board, bi-directional H-bridges. This IC supplies higher voltages and greater current to the motors than the PIC can. Powered with two 9V alkaline batteries, each H-bridge needs three inputs from the PIC in order to spin the motor in both directions: an enable pin indicating whether to run the motor or not, and two pins whose voltage difference decides what direction the motor spins. Because the robot is designed with treads, it can turn in place and move backward or forward using only two motors. All robot movement code is written modularly, so that the robot can be controlled simply with a "human" command, such as "turn_right".

Designing the robot movement control to use a motor driver IC and treads allows the team to eventually migrate to a new robot, perhaps one with larger motors and higher power requirements that will let it transport heavier loads at greater speeds.

POWER SYSTEM

Three 9V batteries power the robot. Two of these batteries connect to the L293 IC, providing each motor with its own power source. The third is connected to a +5V voltage regulator and a 10µf bypass capacitor, supplying a clean source of power to the PIC, FPGA, and IR sensors. A 9V battery replaced four double AA's as the PIC's power source because of the 9V's lighter weight.

The team ran into several battery power issues. The original design relied on two power sources, one to power the motors and one to power the processing system. This design did not sufficiently power the motors; within a minute, the robot slowed to a crawl. To provide additional power, the team also tried using rechargeable batteries. Alkaline batteries have relatively large internal impedances compared to other battery sources, such as NiMH and NiCD rechargeables. This decreases the amount of current that they can output, slowing down the robot. With rechargeables, the opposite situation became the problem—although NiMH and NiCD batteries have lower internal impedances, they are also rated at lower voltages for the same size batteries. This resulted in each motor not having enough power to turn, or to carry the weight of the robot and its accompanying electronics while moving forward.

The motor power system also contains several unnecessary bypass capacitors. The team initially installed them in order to protect current from being slammed back into L293—something that could occur, for example, after a direction change. Four

470 µf capacitors, two for each motor, were placed in series to take care of this problem. However, this "safety feature" only served to discharge the the robot's batteries faster, because the L293 has internal diodes to prevent damage from current spikes.

ROBOT ESCAPE ALGORITHM

The escape algorithm takes distance information from the IR sensor to determine if an object violates the robot's half a meter space bubble. If so, the algorithm also determines if an escape route exists directly in front of or behind the robot, and issues a command to escape in the unobstructed direction. If there are objects on both sides of the robot, it rotates until it finds an open path. A time-based, open loop control dictates how long the robot rotates before it determines that no escape path exists. At this point, LEDs flash, and the robot continues rotating and scanning for new paths.

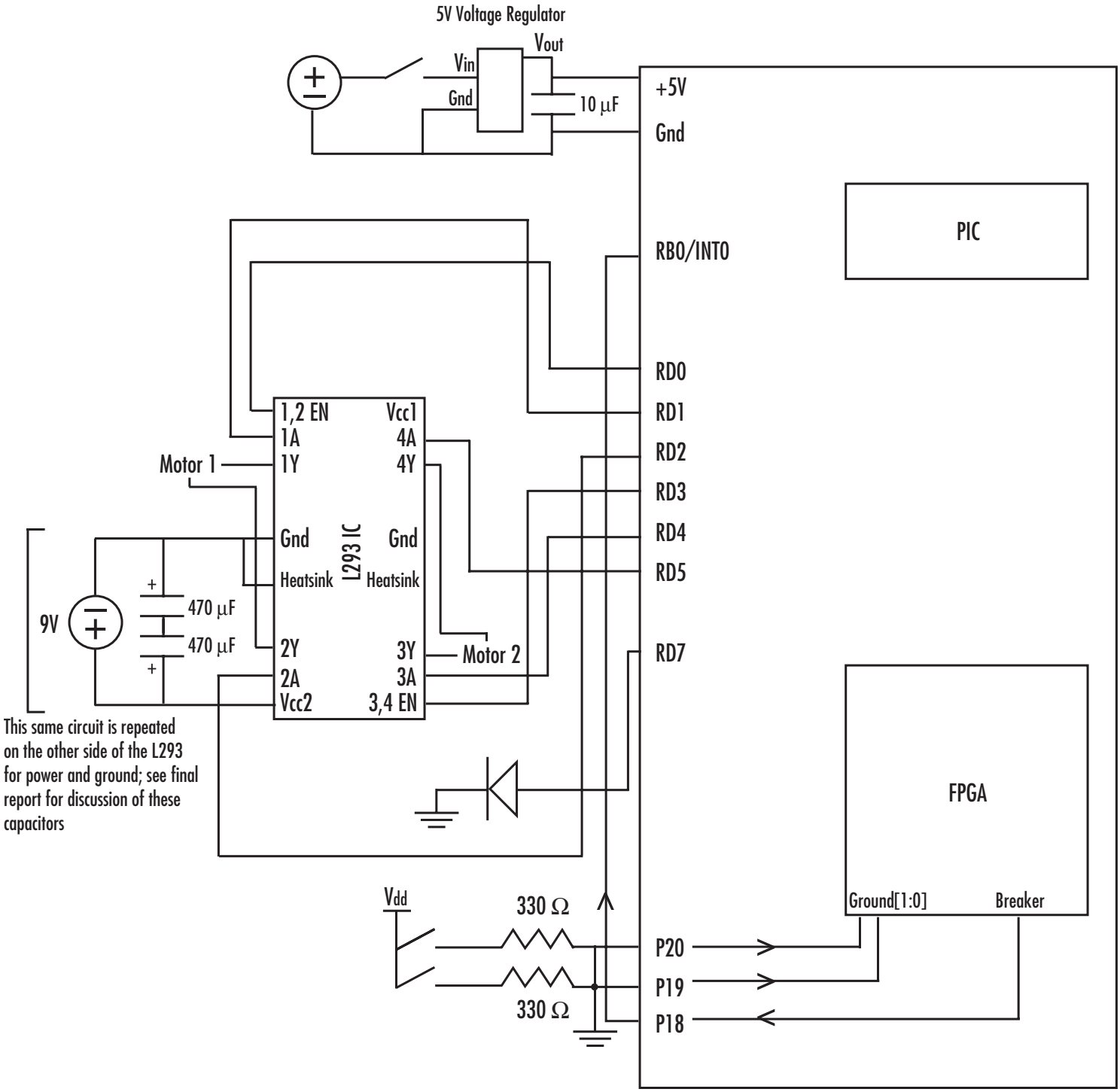
The largest problem with the robot escape algorithm was convincing the robot that it had, in fact, turned long enough to establish that no feasible escape path exists. Initial attempts to use TMR1 and compare it without the help of interrupts resulted in a simulation that worked fine, but the code failed when stepped through at full speed. Though no reason was ever found for this, the team simply rewrote the code to take advantage of CCP1's interrupt abilities.

RESULTS

The robot works as expected, with battery power being the largest obstacle. The robot scans a 360 degree field-of-view as expected, flashes LEDs when no escape path is present, and runs away in the prescribed manner when an escape path is available. As the batteries run down, the robot significantly slows down, and turning presents a far greater challenge for the robot, even if it is able to crawl forward or backward. In addition, the robot's vision is extremely two-dimensional on a plane parallel to the ground, and both its IR sensors and ground sensors have minor blind spots at the robot's corners.

In testing, the robot responded immediately to stimuli obstructing its sensors. It rarely ran into problems with its blind spots, as it generally was capable of detecting gradually approaching objects (as opposed to foots placed directly in front of the robot) and competently escaped them. On a fresh set of alkalines, the robot lasted for about five to seven minutes. On the power supply, the robot always moved faster than on batteries, drawing about 400mA on average.

APPENDIX A: BREADBOARD SCHEMATIC



APPENDIX B: ASSEMBLY CODE

```
; AI.ASM
;
; WRITTEN BY:    TOMMY LEUNG (TLEUNG@HMC.EDU) AND DAN CHAN (DCHAN@HMC.EDU)
; LAST MODIFIED: DECEMBER 9, 2003

; THIS PROGRAM TAKES TWO ANALOG DISTANCE INPUTS FROM RA0 AND RA1,
; DETERMINES THE CLOSEST OBJECT, AND ISSUES COMMANDS FOR THE ROBOT
; TO MOVE AWAY.

; USE THE 18F452 PIC MICROPROCESSOR
      LIST P=18F452
      INCLUDE "P18F452.INC"

; ALLOCATE VARIABLES
LEFT_MOTOR        EQU 0X00
RIGHT_MOTOR       EQU 0X01
ROTATE_LIMIT      EQU 0X02
ROTATE_COUNT      EQU 0X03
TEMP              EQU 0X04
FRONT_SENSOR      EQU 0X05
BACK_SENSOR       EQU 0X06
DISTANCE          EQU 0X07
TIME              EQU 0X08

      ORG 0
      BRA START

      ; ALL INTERRUPTS HAVE THE SAME PRIORITY
      ORG 0X0008
      BRA INTERRUPT

      ORG 0X0018
      BRA INTERRUPT

START

      ; DISTANCE TO MAINTAIN FROM ANY OBJECT
      MOVLW 0X20
      MOVWF DISTANCE

      ; OPEN-LOOP CONTROL FOR THE NUMBER OF ROTATIONS
      ; TO PERFORM BEFORE DECIDING THAT NO ESCAPE PATH
      ; IS POSSIBLE
      MOVLW 0XFF
      MOVWF CCPR1H

      MOVLW 0X02
      MOVWF ROTATE_LIMIT

      CLRF ROTATE_COUNT
```

```
; ---INTERRUPT SETUP

      ; CLEAR THE A/D AND CCP1 INTERRUPTS
      CLRF PIR1

      ; DISABLE PRIORITIZED INTERRUPTS
      ; RCON = 00011111
      MOVLW 0X1F
      MOVWF RCON

      ; SET INTCON TO ENABLE ALL INTERRUPTS,
      ; ALLOW EXTERNAL INTERRUPTS TO STOP THE PROGRAM,
      ; AND CLEAR THE EXTERNAL INTERRUPT FLAG BIT
      ; INTCON = 11010000
      MOVLW 0XD0
      MOVWF INTCON

      MOVLW 0X04
      MOVWF PIE1

      ; SET UP TMR1 (10110000)
      MOVLW 0XB0
      MOVWF T1CON
      CLRF TMR1H
      CLRF TMR1L

      ; ---INITIALIZE CCP1
      MOVLW 0X08
      MOVWF CCP1CON
      ; ---END OF CCP1 INITIALIZATION

      ; DISABLE EXTERNAL INTERRUPTS 1 & 2
      CLRF INTCON3

      ; ---INITIALIZE THE A/D CONVERTER
      MOVLW 0X01
      MOVWF ADCON0

      MOVLW 0X0D
      MOVWF ADCON1
      ; ---END A/D CONVERTER INITIALIZATION

; ---END INTERRUPT SETUP

; ---BEGIN I/O PORT SETUP
      SETF TRISA
      SETF TRISB
      CLRF TRISD
      CLRF PORTD
; ---END I/O PORT SETUP
```

```
; CONTINUOUSLY REPORT DISTANCE INFORMATION
AD_LOOP

      ; SELECT SENSOR 0 (ATTACHED TO AD CHANNEL 0) TO SAMPLE
      MOVLW 0X01
      MOVWF ADCON0

      ; INITIATE AN A/D CONVERSION
      BSF ADCON0, 2

      CALL WAIT
      MOVFF ADRESH, FRONT_SENSOR

      ; SELECT SENSOR 1 (ATTACHED TO AD CHANNEL 1) TO SAMPLE
      MOVLW 0X09
      MOVWF ADCON0

      ; INITIATE AN A/D CONVERSION
      BSF ADCON0, 2

      CALL WAIT
      MOVFF ADRESH, BACK_SENSOR

      BRA ESCAPE

; POLL THE A/D UNIT TO MAKE SURE THAT THE CONVERSION IS DONE
WAIT
      BTFSC ADCON0, 2
      BRA WAIT
      RETURN

ESCAPE
      ; MOVE THE ALLOWABLE SPACE INTO WREG
      MOVF DISTANCE, WREG

      ; CHECK FRONT_SENSOR < DISTANCE
      CPFSLT FRONT_SENSOR

      ; IF FRONT_SENSOR > DISTANCE, CHECK FOR AN ESCAPE ROUTE
      BRA CHECK_ESCAPE_BACK

      ; OTHERWISE, IF FRONT_SENSOR < DISTANCE, CHECK THE BACK SENSOR
      BRA CHECK_BACK

CHECK_BACK
      ; CHECK BACK_SENSOR < DISTANCE
      CPFSLT BACK_SENSOR

      ; IF BACK_SENSOR > DISTANCE, CHECK FOR AN ESCAPE ROUTE
      BRA CHECK_ESCAPE_FRONT

      ; OTHERWISE, IF BACK_SENSOR < DISTANCE, CONTINUE SCANNING
```

```
      BRA ROTATE

; CONTINUE SCANNING
ROTATE

      ; CHECK IF TIMER1 IS ON
      BTFSS T1CON, 0

      ; IF NOT, TURN IT ON
      BSF T1CON, 0

      CALL TURN_RIGHT
      CALL TIMER
      BRA AD_LOOP

CHECK_ESCAPE_FRONT
      CPFSLT FRONT_SENSOR

      ; IF FRONT_SENSOR > DISTANCE, IT IS NOT SAFE TO RETREAT FORWARD
      BRA ROTATE

      ; OTHERWISE, RETREAT FORWARD
      CALL GO_FORWARD
      CALL TIMER
      BRA AD_LOOP

CHECK_ESCAPE_BACK

      CPFSLT BACK_SENSOR
      ; IF BACK_SENSOR > DISTANCE, IT IS NOT SAFE TO RETREAT BACKWARD
      BRA ROTATE

      ; OTHERWISE, RETREAT BACKWARD
      CALL GO_BACKWARD
      CALL TIMER
      BRA AD_LOOP


; ---MOTOR CODE
STOP_MOTOR
      CLRF PORTD
      RETURN

GO_FORWARD

      MOVLW B'00101011'
      MOVWF PORTD

      ; TURN OFF THE "NO ESCAPE ROUTE" STATUS
      BCF T1CON, 0
      CLRF TMR1H
      CLRF TMR1L
```

```
      CLRF ROTATE_COUNT
      RETURN

GO_BACKWARD

      MOVLW B'00011101'
      MOVWF PORTD

      ; TURN OFF THE "NO ESCAPE ROUTE" STATUS
      BCF T1CON, 0
      CLRF TMR1H
      CLRF TMR1L
      CLRF ROTATE_COUNT
      RETURN

TURN_RIGHT
      BSF PORTD, 0
      BCF PORTD, 1
      BSF PORTD, 2
      BSF PORTD, 3
      BCF PORTD, 4
      BSF PORTD, 5

      RETURN
; ---END OF MOTOR CODE


; ---TIMER
TIMER
      ; TURN ON THE TIMER AND CLEAR ALL COUNTERS
      MOVLW 0X96
      MOVWF T0CON
      CLRF TMR0H
      CLRF TMR0L
      CLRF TIME

KEEP_COUNTING
      CLRF TMR0H
      CLRF TMR0L
      MOVLW 0X10

ONE_SECOND
      ; TMR0H DOES NOT UPDATE UNLESS TMR0L IS READ
      MOVFF TMR0L, TEMP
      MOVFF TMR0L, TEMP

      MOVFF TMR0H, TEMP
      MOVFF TMR0H, TEMP

      ; SEE IF THE DESIRED AMOUNT OF TIME HAS ELAPSED
      CPFSGT TMR0L
      BRA ONE_SECOND
```

```
      ; IF DESIRED TIME HAS ELAPSED, TURN TIMER OFF AND RETURN
      CLRF T0CON
      RETURN
; ---END TIMER


; ---INTERRUPTS
INTERRUPT

; IF THIS IS NOT A GROUND SENSOR INTERRUPT, CHECK CCP1
      BTFSS INTCON, 1
            BRA ROTATE_INTERRUPT

      BRA STOP

STOP
      CLRF PORTD
      BRA STOP

ROTATE_INTERRUPT
      BTFSS PIR1, 2

            ; IF THIS ISN'T A ROTATE INTERRUPT, RETURN
            BRA CLEAR_INTERRUPT

      INCF ROTATE_COUNT

      MOVF ROTATE_COUNT, WREG

      ; CHECK ROTATE_LIMIT < ROTATE_COUNT
      CPFSLT ROTATE_LIMIT

            ; IF ROTATE_LIMIT > ROTATE_COUNT, DO NOT TURN ON THE NO-ESCAPE LED
            BRA CLEAR_INTERRUPT

      ; CHECK IF DISTANCE < FRONT_SENSOR
      MOVF FRONT_SENSOR, WREG
      CPFSLT DISTANCE

            ; IF DISTANCE > FRONT_SENSOR
            BRA CLEAR_INTERRUPT

      MOVF BACK_SENSOR, WREG
      CPFSLT DISTANCE

            ; IF DISTANCE > BACK_SENSOR
            BRA CLEAR_INTERRUPT

      ; IF THERE IS NO ESCAPE PATH, FLASH AN LED
      MOVF FRONT_SENSOR, WREG
```

```
      BSF PORTD, 7

; RETURN FROM THE INTERRUPT
CLEAR_INTERRUPT
      BCF INTCON, 1
      CLRF PIR1
      RETFIE

; ---END INTERRUPTS

      END
```

APPENDIX C: VERILOG

```
// SIGNAL_CONDUIT.V
//
// WRITTEN BY:          TOMMY LEUNG & DANIEL CHAN
// LAST MODIFIED: NOVEMBER 20, 2003
//
// THIS MODULE ACTS AS A CIRCUIT BREAKER FOR THE ROBOT.  IF ANY OF THE GROUND
// SENSORS ARE TRIPPED, THE FPGA SENDS AN INTERRUPT SIGNAL TO THE PIC.

MODULE SIGNAL_CONDUIT(GROUND1, GROUND2, BREAKER);

    INPUT GROUND1;
    INPUT GROUND2;
       OUTPUT BREAKER;

       ASSIGN BREAKER = GROUND1 & GROUND2;

ENDMODULE
```