Java-Based Wireless Robot

New Mexico
Supercomputing Challenge

Final Report
April 7, 2010

Team 81 - Manzano High School

Team Members
Phillip Atencio
Dustin Chavez
Nathan Hassler
Nick Ratzer
David Young

Teacher Sponsor / Project Mentor
Mr. Stephen Schum

Table of Contents

Executive Summary

Introduction

Research and Science Background

Project Overview

Results

Conclusions

References and Acknowledgements

Appendix:
- **pBasic 2.5 Computer Code**
- **Java Computer Code**

Executive Summary:

The purpose of our project was to design, assemble and program two small robots -- a pBasic-stamp Boe-Bot and a Javelin-stamp Java-Bot to utilize LED’s, “whiskers”, photo resistors, and IR pairs to perform various maneuvers and environmental detection tasks, such as detecting and monitoring light, objects, and distances to objects, etc. . Then we intended to add a small solar panel to make the Java-Bot solar-powered and add wireless communication with the Java-Bot.

A real-world robot executes essential tasks in hard to reach or dangerous environments. Robots look for life in burning buildings, they detect dangerous gases and chemicals in hazardous environments or spills, they monitor radiation, they study far away planets and moons (Examples: Mars Explorer Rovers) and, in general, they can access and provide sensory input for a variety of physical parameters in various remote environments. Once the robots are assembled and functional, scientists and technicians can communicate programs of instructions to the robots and retrieve data via wireless technology. Furthermore to reduce the high costs and time demands of rechargeable batteries, solar panels can be added to provide electrical power to the robot.

During fall 2009, in our Manzano High School class, Pre-Engineering Electronics, we learned and solved problems for the basic physics topics of velocity, force, work, power, electric charge, electric fields, DC voltage, DC current, resistance, Ohms Law, DC series and parallel circuits, and the power formula (P = I V). Then we spent the past five months assembling and programming a mini-robot with Basic computer programs to make the robot do a variety of maneuver on a flat surface and to detect motion and light.. In doing so, we learned some Basic programming techniques with variables, counter, for loops, etc.

From January through April, we assembled and programmed a similar Java-based robot to maneuver, plus to perform various maneuvers and to detect light, shadows, and objects. Next we will also add the wireless component to communicate programs to the robot and to return data from the robot. Our Java programs will also output retrieved data to the display. Furthermore we will add small solar panels with sufficient area to power the robot, hopefully during both natural sunlight and interior lighting conditions

Introduction

Over the past several months, our group has worked on programming and assembling a pBasic-based mini-robot and a Javelin-based mini robot that will complete various complex tasks that a real world robot would normally carry out. Our basic knowledge of programming began with the pBASIC code and evovled to Javelin (Java-based) code. We wrote programs and assembled the circuit board and robot hardware to perform various maneuvers and environmental detection tasks, such as detecting and monitoring light, objects, and distances to objects. Then we modified the programs and hardware to allow the Boe-Bot and Java-Bot to successfully navigate a complex maze. This took much of the prior programming and wiring knowledge to achieve an escape from the maze.

Research and Science Background

Our endeavor began last fall with a review of the foundations of physics – motion, forces, work, energy, power, static electricity and DC electricity. Upon completion of our Basic stamp and Javelin stamp robots, we spent much time learning about circuit diagrams and DC current.

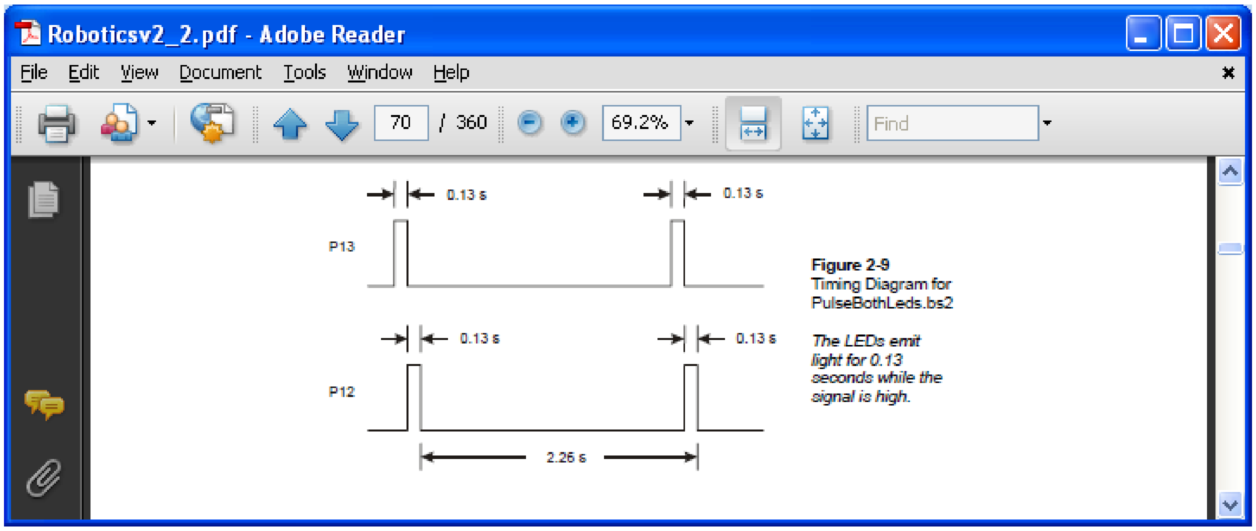
Roboticsv2_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
70 / 360
69.2%
Find
0.13 s
0.13 s
P13
0.13 s
0.13 s
P12
2.26 s

Figure 2-9
Timing Diagram for PulseBothLeds.bs2

The LEDs emit light for 0.13 seconds while the signal is high.

Figure 1. LED Pulse Timing Diagram (Parallax, Inc. “Robotics2_2.pdf”)

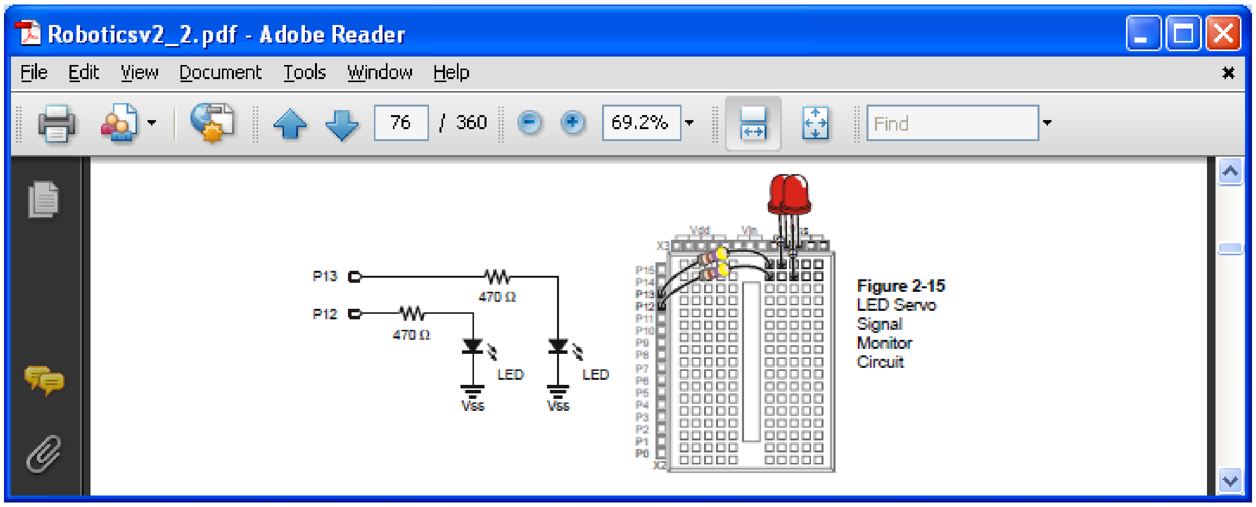

Figure 2-15
LED Servo Signal Monitor Circuit

Figure 2. LED Circuit Schematic (Parallax, Inc. “Robotics2_2.pdf”)

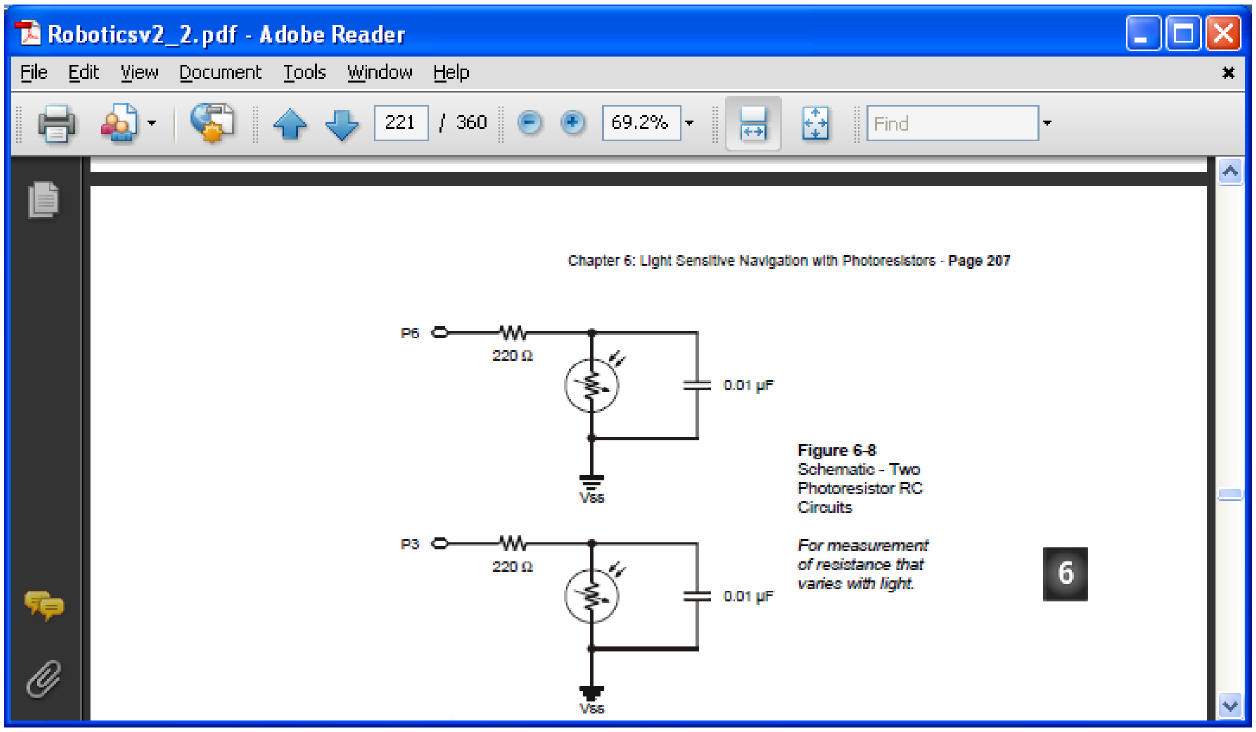
Roboticsv2_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
221 / 360
69.2%
Find
Chapter 6: Light Sensitive Navigation with Photoresistors · Page 207
P6
220 Ω
0.01 µF
Vss
P3
220 Ω
0.01 µF
Vss
6

Figure 6-8
Schematic - Two Photoresistor RC Circuits

For measurement of resistance that varies with light.

Figure 3. Photoresistor RC Circuits Schematic (Parallax, Inc. "Robotics2_2.pdf")

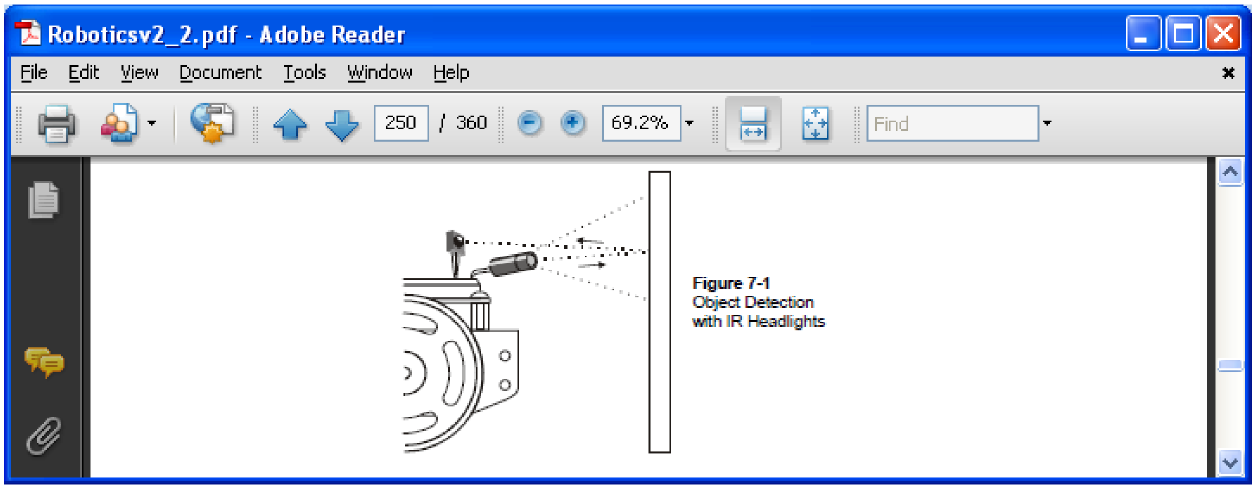
Roboticsv2_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
250 / 360
69.2%
Find

Figure 7-1
Object Detection with IR Headlights

Figure 4. IRED Source IR Detector Diagram (Parallax, Inc. "Robotics2_2.pdf")

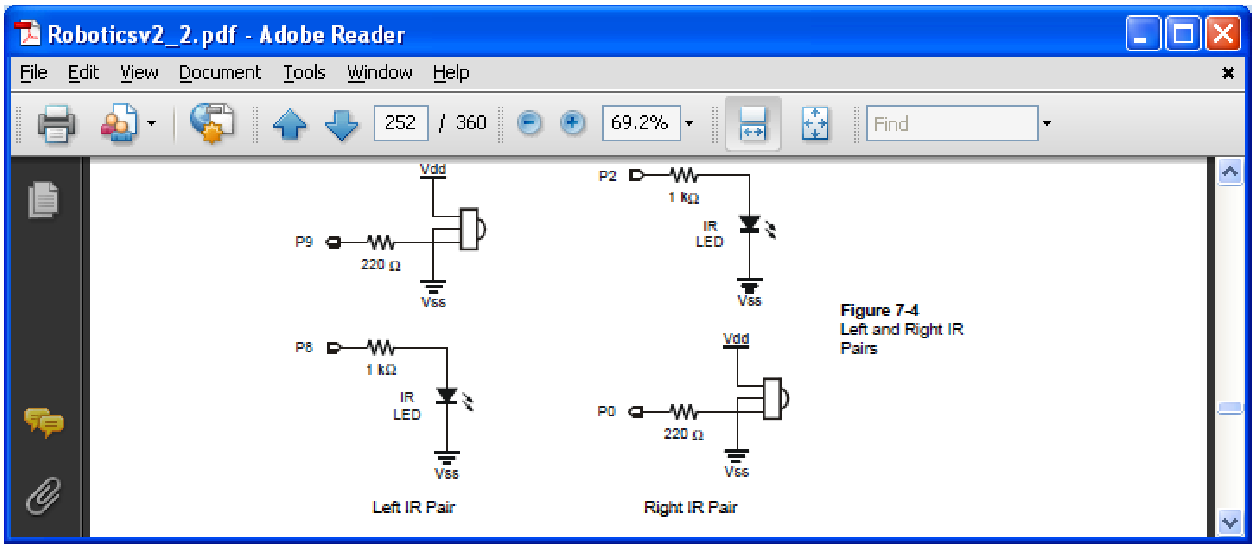
Roboticsv2_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
252 / 360
69.2%
Find
Vdd
P9
220 Ω
Vss
P2
1 kΩ
IR LED
Vss
P8
1 kΩ
IR LED
Vss
Vdd
P0
220 Ω
Vss
Left IR Pair
Right IR Pair
Figure 7-4 Left and Right IR Pairs

Figure 5. IRED Source IR Detector Diagram (Parallax, Inc. “Robotics2_2.pdf”)

Project Overview:

After we became well versed with comprehending circuit diagrams, we developed the pBASIC program for the Boe-Bot and assembled a pBASIC-stamp Boe-Bot to complete various tasks such as detecting light, objects and distance to objects using LED’s, IRED’s and IR detectors and “whiskers”. Then we wrote a pBasic 2.5 program (See Appendix.) to allow the Boe-Bot to successfully navigate a complex maze with multiple turns and corners as well as descending a 45 degree angle, and various obstacles. (See Figures 1 and 2 below.) Other Boe-Bot programs included photo resistors for light detection, use of “whiskers” for object detection, an LED on/off switch, a pair of left and right IRED sources and IR detectors for distance and object detection, and a “shadow bot” program where one shadow Boe-Bot was able to follow a leading Boe-Bot. (We found on You-Tube a video of 28 shadow-bots following one lead-bot!!)

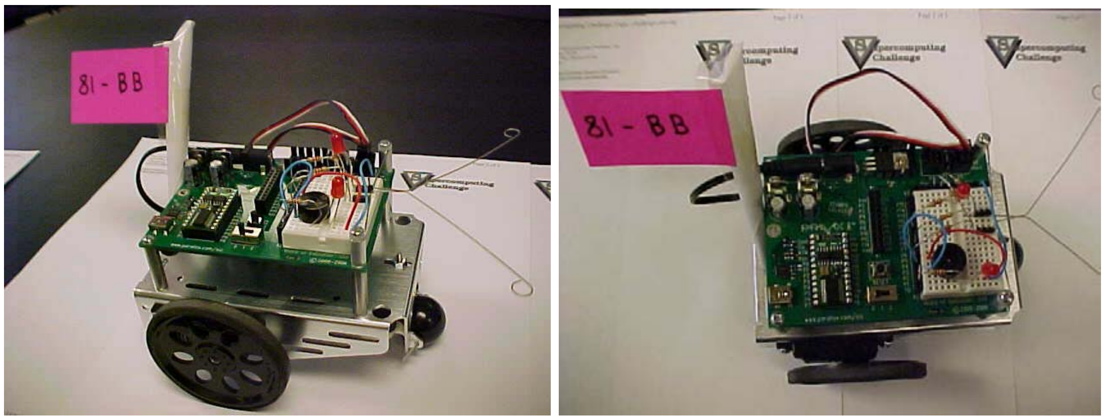

Fig 6. Team 81 Boe-Bot Assembled with Whiskers for Object Detection

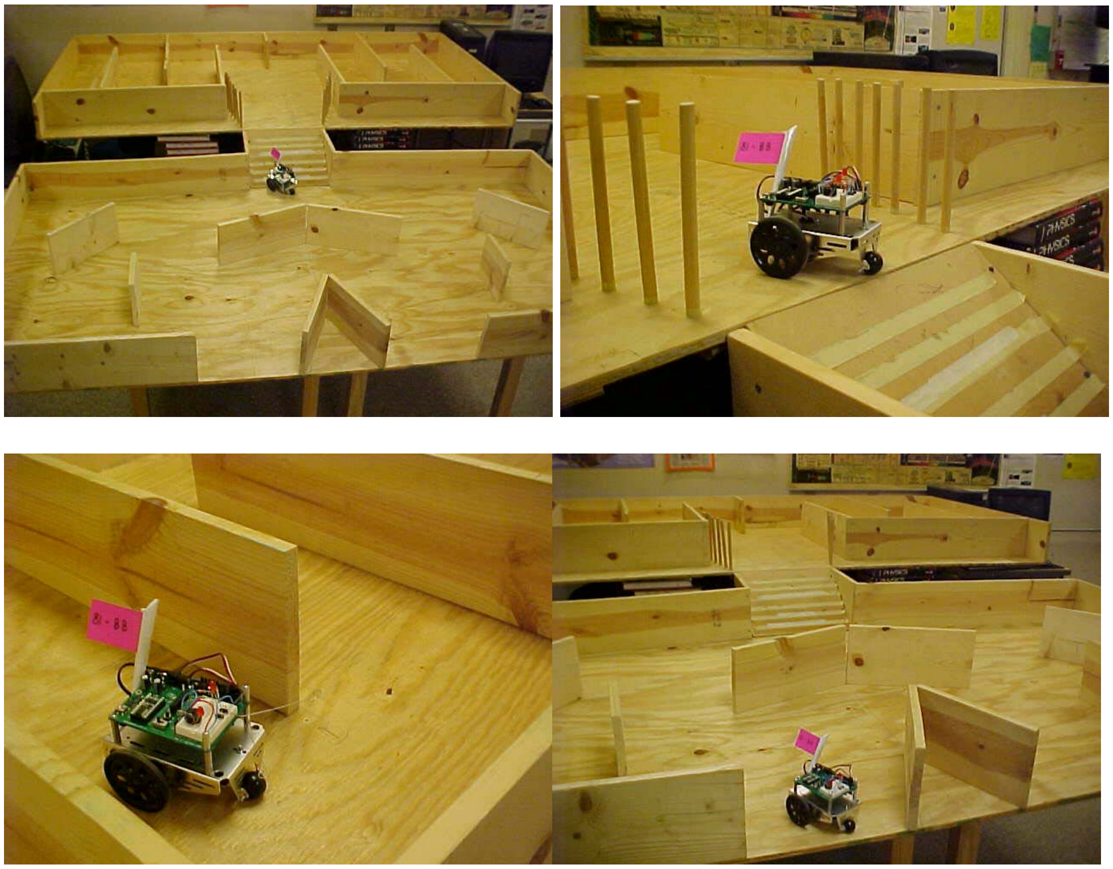

Fig 7. Team 81 Boe-Bot Attempting to Get Out of the Maze

For the past three months, we have been learning Javelin (a Java-based code) from a Javelin manual for a, Java-Bot, which has a Javelin stamp located on the circuit board. And we have been assembling the circuit components on the Java-Bot for the LED, IRED and IR detector, photo resistors, and “whiskers”. Within the next few weeks, we expect to have the Java-Bot accomplish the same tasks as those stated above for the Boe-Bot.   This includes programming the Java-Bot to successfully navigate our complex maze.

Conclusions

We intend to put a solar panel on the Java Bot so that we can run it without using batteries. Plus we plan to install wireless communication hardware on the Java-Bot. We will write Java programs that will be communicated to the Java-stamp on the robot circuit board wirelessly. And the program will allow the robot to wirelessly communicate back to the computer the sensory output/data acquired by the various sensors placed on the robot, while being powered by an aftermarket solar panel. Our greatest hope is to become well versed with Java, and able to send our robot programs, wirelessly, while it is running. When we are able to successfully communicate with our robot wirelessly, we will attempt to complete complex tasks that can be easily applied to “real life” applications which would be too dangerous for any human being to do. Such applications include, but are not limited to: detecting smoke, body temperature, and light.

Acknowledgements:

Our team would like to acknowledge the NM Supercomputing Challenge Program for 2009-2010 and Mr. Stephen Schum for teaching the Pre-Engineering Electronics program here at Manzano High School and for helping us problem solve throughout the duration of the project. We would also like to give a special thanks to the MHS Administration for supporting the Pre-Engineering Electronics class.

References:

Zitzewitz, et al (2005), Physics: Principles and Problems, Glencoe, Chap 1-11; 20-24

Lindsay, Andy,(2003-2004), Robotics with the Boe-Bot, Parallax, Inc.

Lindsay, Andy,(2002-2006) , Javelin Stamp User's Manual, Parallax, Inc.

Lindsay, Andy, (1999-2008), Basic Analog and Digital, Parallax, Inc.

Parallax, Inc. (2009), "Robotics2_2.pdf" on Boe-Bot Resources CD, Parallax, Inc.

http://marsrover.nasa.gov/home/ NASA Mars Explorer Rover Home Page

Appendix

pBasic Code: Boe-Bot Get Out of Maze

```
' -----[ Team 81 Boe-Bot Get Out of Maze ]--------------------------------------

' {$STAMP BS2}                              ' Stamp directive.

' {$PBASIC 2.5}                             ' PBASIC directive.

' -----[ Variables ]---------------------------------------------------------

pulseCount     VAR     Byte                 ' For...next loop counter.

counter        VAR     Nib                  ' Counts alternate contacts.

old7           VAR     Bit                  ' Stores previous IN7.

old5           VAR     Bit                  ' Stores previous IN5.

' -----[ Initialization ]----------------------------------------------------

FREQOUT 4, 2000, 3000                       ' Signal program start/reset.

counter = 1                                 ' Start alternate corner count.

old7 = 0                                    ' Make up old values.

old5 = 1

' -----[ Main Routine ]------------------------------------------------------

DO

' ---  Detect Consecutive Alternate Corners -----------------------

' See the "How EscapingCorners.bs2 Works" section that follows this program.

  IF (IN7 <> IN5) THEN                      ' One or other is pressed.

    IF (Old7 <> IN7) AND (Old5 <> IN5) THEN  ' Different from previous.

      counter = counter + 1                 ' Alternate whisker count + 1.

      old7 = IN7                            ' Record this whisker press
      old5 = IN5                      ' for next comparison.
      IF (counter > 4) THEN                ' If alternate whisker count = 4,
        counter = 1                   ' reset whisker counter
        GOSUB Back_Up                      ' and execute a U-turn.
        GOSUB Turn_Left
        GOSUB Turn_Left
      ENDIF                           ' ENDIF counter > 4.
    ELSE                              ' ELSE (old7=IN7) or (old5=IN5),
      counter = 1                     ' not alternate, reset counter.
    ENDIF                             ' ENDIF (old7<>IN7) and
                                    ' (old5<>IN5).
  ENDIF                               ' ENDIF (IN7<>IN5).
' ---  Same navigation routine from RoamingWithWhiskers.bs2 ------------------
  IF (IN5  = 0) AND (IN7 = 0) THEN          ' Both whiskers detect obstacle
    GOSUB Back_Up                          ' Back up & U-turn (left twice)
    GOSUB Turn_Left
    GOSUB Turn_Left
  ELSEIF (IN5  = 0) THEN                   ' Left whisker contacts
    GOSUB Back_Up                          ' Back up & turn right
    GOSUB Turn_Right
  ELSEIF (IN7  = 0) THEN                   ' Right whisker contacts
    GOSUB Back_Up                          ' Back up & turn left
    GOSUB Turn_Left
  ELSE                                ' Both whiskers 1, no contacts
    GOSUB Forward_Pulse                   ' Apply a forward pulse
  ENDIF                               ' and check again
LOOP
' -----[ Subroutines ]-----------------------------------------------------
Forward_Pulse:                           ' Send a single forward pulse.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN
Turn_Left:                               ' Left turn, about 90-degrees.
  FOR pulseCount = 0 TO 9
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 9                ' Right turn, about 90-degrees.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT

  RETURN

Back_Up:                               ' Back up.

  FOR pulseCount = 0 TO 19

    PULSOUT 13, 650

    PULSOUT 12, 850

    PAUSE 20

  NEXT

  RETURN

                        ' Send a single forward pulse.

  PULSOUT 13,850

  PULSOUT 12,650

  PAUSE 20

  RETURN

                          ' Left turn, about 90-degrees.

  FOR pulseCount = 0 TO 6

    PULSOUT 13, 650

    PULSOUT 12, 650

    PAUSE 20

  NEXT

  RETURN

  FOR pulseCount = 0 TO 6          ' Right turn, about 90-degrees.

    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
                                    ' Back up.
  FOR pulseCount = 0 TO 6
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```