A Hardware Interfacing And Control Protocol

Using RobotBASIC
And The Propeller Chip

John Blankenship & Samuel Mishal

ISBN-13: 978-1438272849 ISBN-10: 1438272847

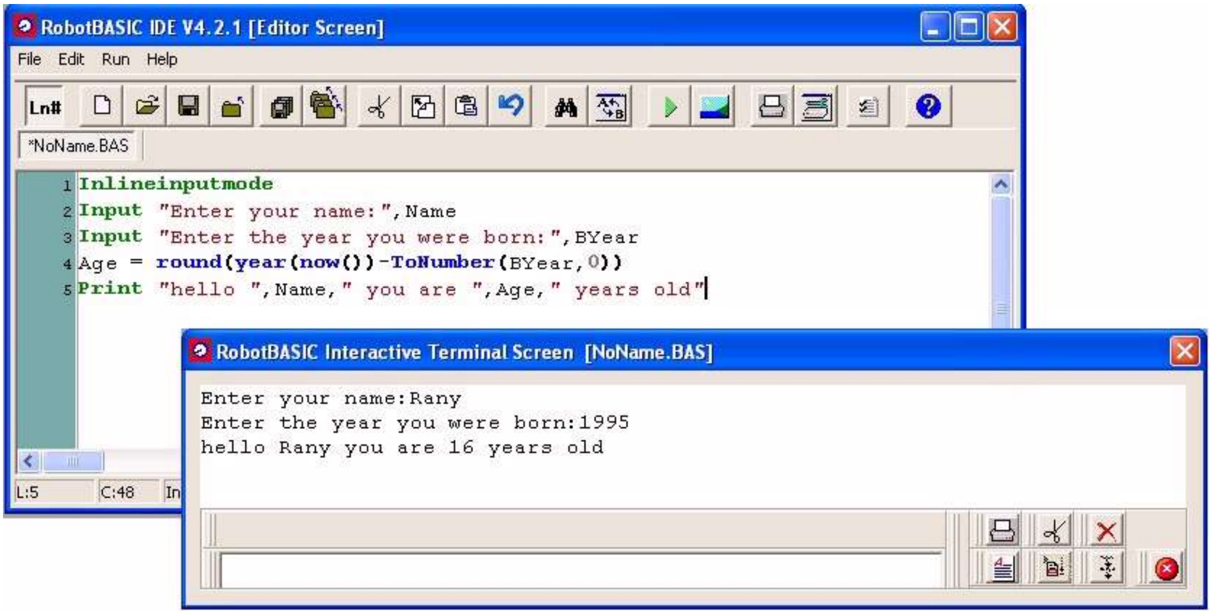
Images of proprietary devices and sensors are reproduced with the permission of the manufacturing companies.

The information and projects in this book are provided for educational purposes without warranty. Although care has been taken in the preparation of this book, neither the authors or publishers shall have any liability to any person or entity with respect to any special, incidental or consequential loss of profit, downtime, goodwill, or damage to or replacement of equipment or property, or damage to persons, health or life, or any costs of recovering, reprogramming, or reproducing any data, caused or alleged to be caused directly or indirectly by the information contained in this book or its associated web site.

The source code for the program listings in this book (and much more) is available to the readers at http://www.RobotBASIC.com

This PDF prvides:

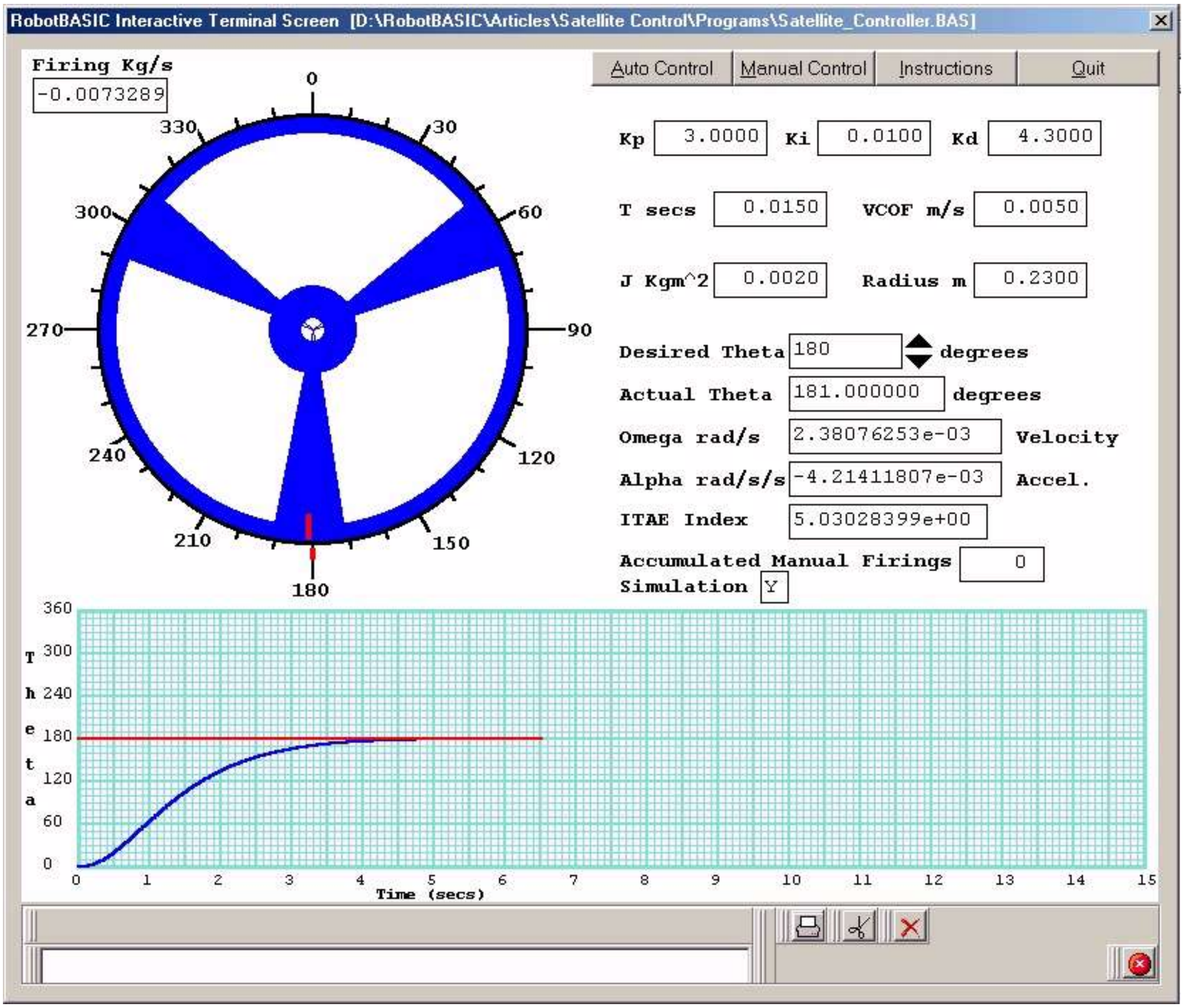
1. **All the figure in the book that are best viewed in color.**
2. **The procedural strategy for adding other hardware to the protocol as in Chapter 8.2.**
3. **All the links in the book that are listide in Appendix A.**
4. **The Schematics and Tables in Appendix B.**


RobotBASIC IDE V4.2.1 [Editor Screen]
File
Edit
Run
Help
Ln#
*NoName.BAS
L:5
C:48
In
RobotBASIC Interactive Terminal Screen [NoName.BAS]

```
Inlineinputmode
Input "Enter your name:",Name
Input "Enter the year you were born:",BYear
Age = round(year(now())-ToNumber(BYear,0))
Print "hello ",Name," you are ",Age," years old"
```

```
Enter your name:Rany
Enter the year you were born:1995
hello Rany you are 16 years old
```

Figure 1.1: A simple user interface program



RobotBASIC Interactive Terminal Screen [D:\RobotBASIC\Articles\Satellite Control\Programs\Satellite_Controller.BAS]

Firing Kg/s -0.0073289

Kp 3.0000 Ki 0.0100 Kd 4.3000

T secs 0.0150 VCOF m/s 0.0050

J Kgm^2 0.0020 Radius m 0.2300

Desired Theta 180 degrees

Actual Theta 181.000000 degrees

Omega rad/s 2.38076253e-03 Velocity

Alpha rad/s/s -4.21411807e-03 Accel.

ITAE Index 5.03028399e+00

Accumulated Manual Firings 0

Simulation Y

Figure 1.2: An example of a GUI program in action

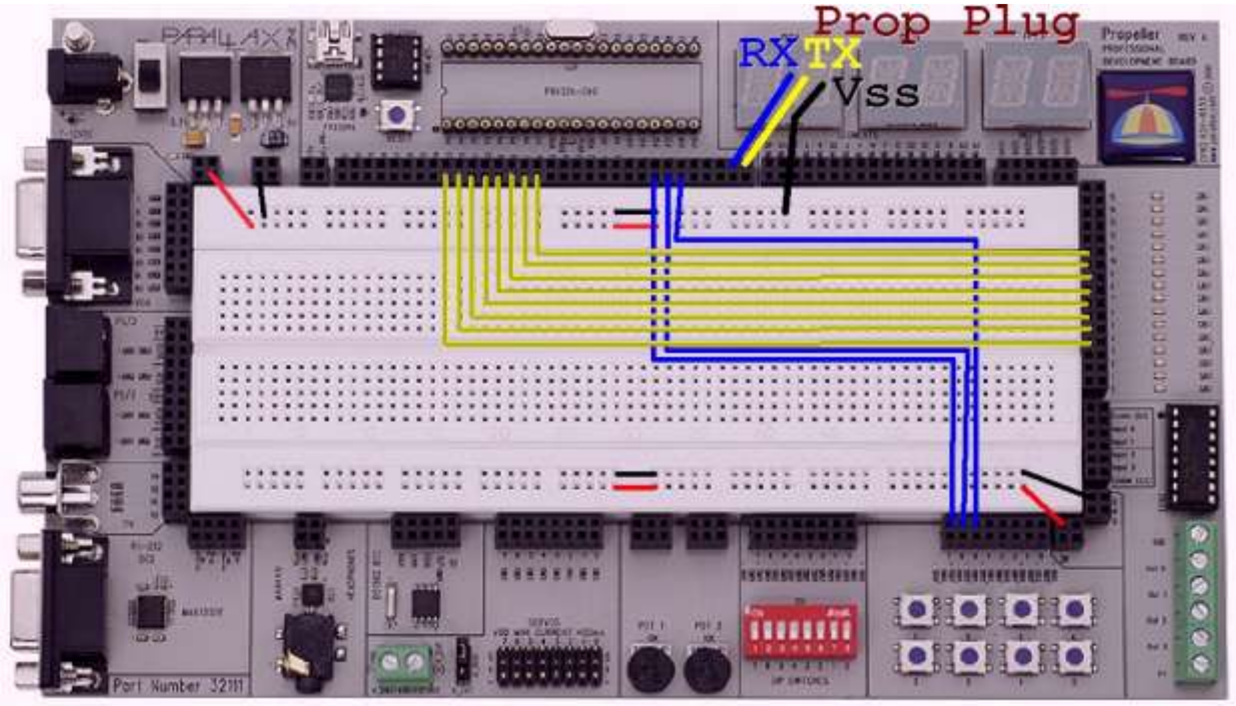


Figure 2.4: How to setup the PPDB for the initial work.

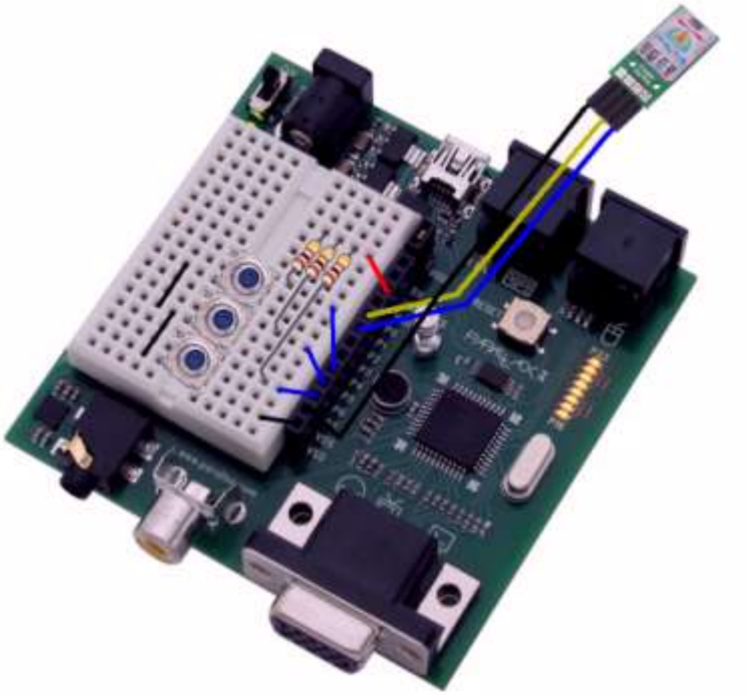
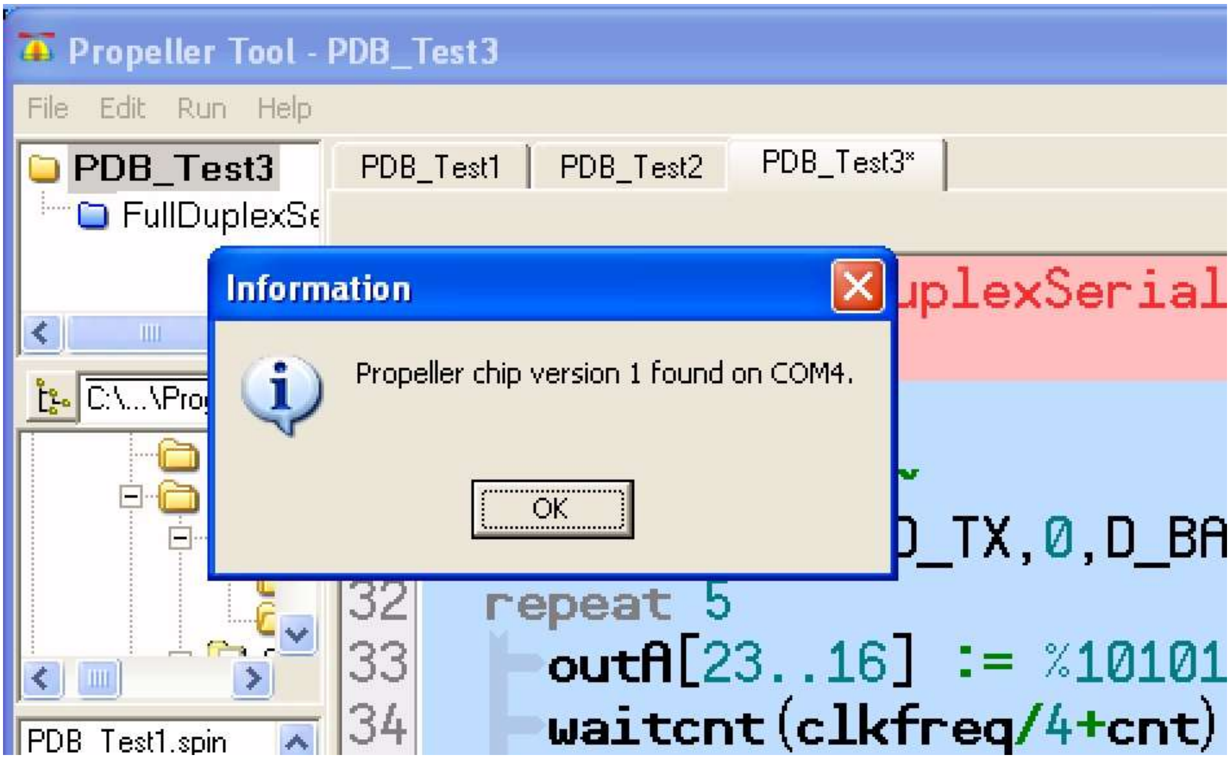


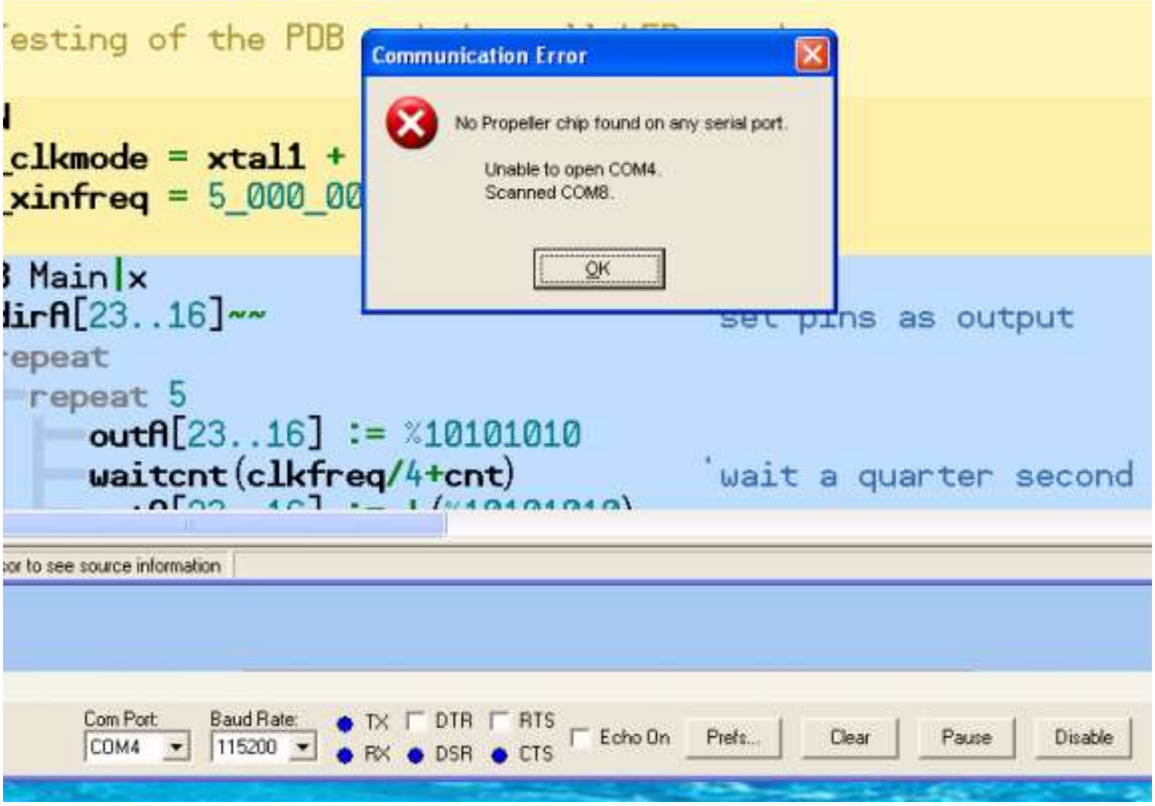
Figure 2.5: How to setup the PDB.



Propeller chip version 1 found on COM4.

```
uplexSerial
D_TX,0,D_BA
repeat 5
  outA[23..16] := %10101
  waitcnt(clkfreq/4+cnt)
```

Figure 2.12: Pressing F7 to determine the port number of the USB port for programming the Propeller.



```
'esting of the PDB

_clkmode = xtal1 +
_xinfreq = 5_000_00

 Main|x
irA[23..16]~~                                   set pins as output
epeat
 repeat 5
   outA[23..16] := %10101010
   waitcnt(clkfreq/4+cnt)                      'wait a quarter second
```

Communication Error

No Propeller chip found on any serial port.

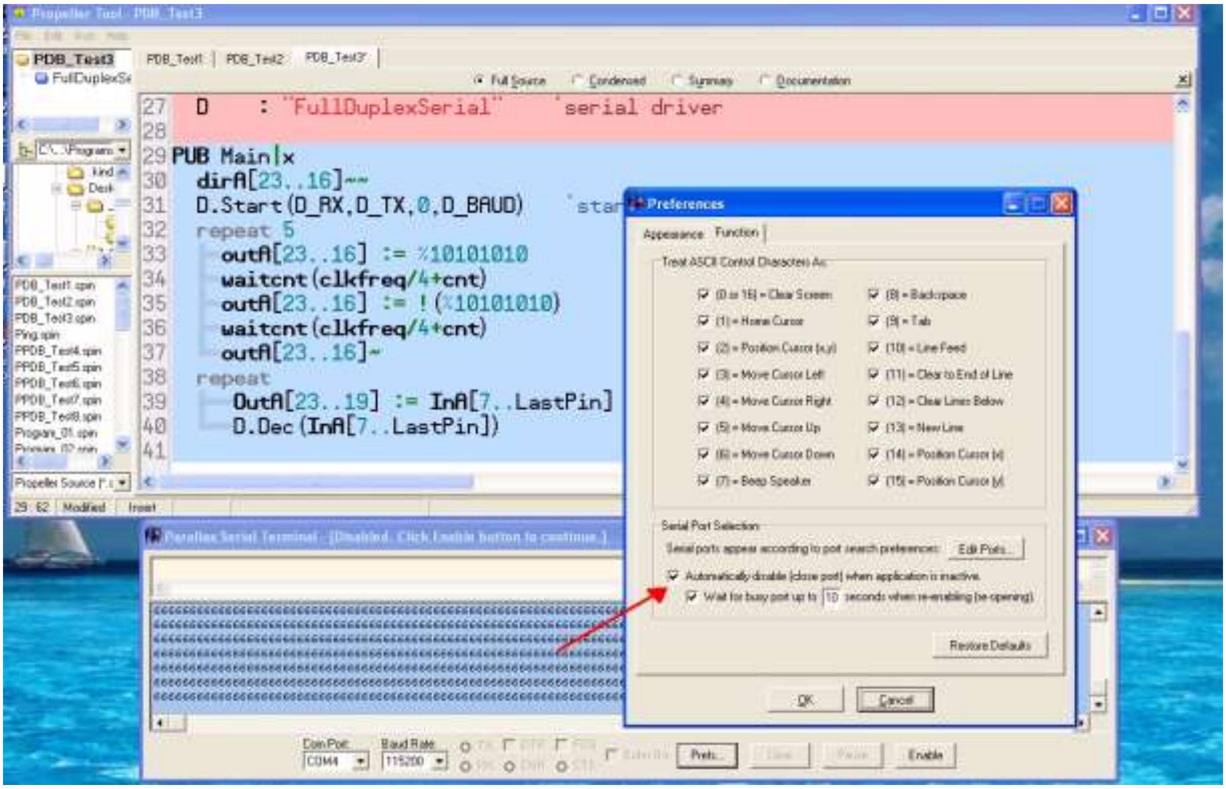
Unable to open COM4.
Scanned COM8.

sor to see source information

Com Port: COM4

Baud Rate: 115200

Figure 2.13: What you will see if you try to program the Propeller when the port is busy.



```
  D     : "FullDuplexSerial"     'serial driver

PUB Main|x
  dirA[23..16]~~
  D.Start(D_RX,D_TX,0,D_BAUD)     'star
  repeat 5
    outA[23..16] := %10101010
    waitcnt(clkfreq/4+cnt)
    outA[23..16] := !(%10101010)
    waitcnt(clkfreq/4+cnt)
    outA[23..16]~
  repeat
     OutA[23..19] := InA[7..LastPin]
     D.Dec(InA[7..LastPin])
```

Preferences

Treat ASCII Control Characters As:

☑ (0 or 16) = Clear Screen
☑ (1) = Home Cursor
☑ (2) = Position Cursor (x,y)
☑ (3) = Move Cursor Left
☑ (4) = Move Cursor Right
☑ (5) = Move Cursor Up
☑ (6) = Move Cursor Down
☑ (7) = Beep Speaker
☑ (8) = Backspace
☑ (9) = Tab
☑ (10) = Line Feed
☑ (11) = Clear to End of Line
☑ (12) = Clear Lines Below
☑ (13) = New Line
☑ (14) = Position Cursor (x)
☑ (15) = Position Cursor (y)

Serial Port Selection

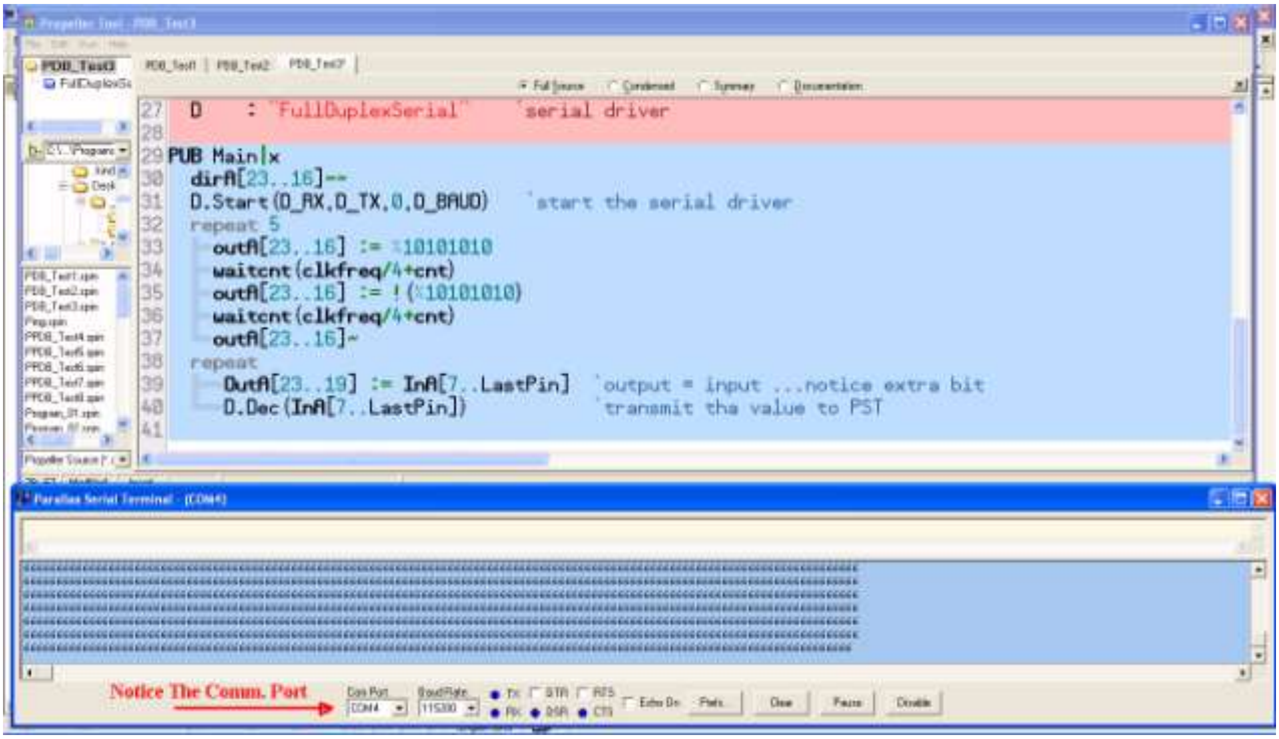
Serial ports appear according to port search preferences:

☑ Automatically disable (close port) when application is inactive.

☑ Wait for busy port up to 10 seconds when re-enabling (re-opening).

Com Port: COM4
Baud Rate: 115200

Figure 2.14: Configuring the Propeller Serial Terminal (PST) to release the com port when it loses focus.



```
  D     : "FullDuplexSerial"     'serial driver

PUB Main|x
  dirA[23..16]~~
  D.Start(D_RX,D_TX,0,D_BAUD)     'start the serial driver
  repeat 5
    outA[23..16] := %10101010
    waitcnt(clkfreq/4+cnt)
    outA[23..16] := !(%10101010)
    waitcnt(clkfreq/4+cnt)
    outA[23..16]~
  repeat
     OutA[23..19] := InA[7..LastPin]     'output = input ...notice extra bit
     D.Dec(InA[7..LastPin])              'transmit the value to PST
```

Com Port: COM4
Baud Rate: 115200

Figure 3.1: PST with correct com port and showing output.

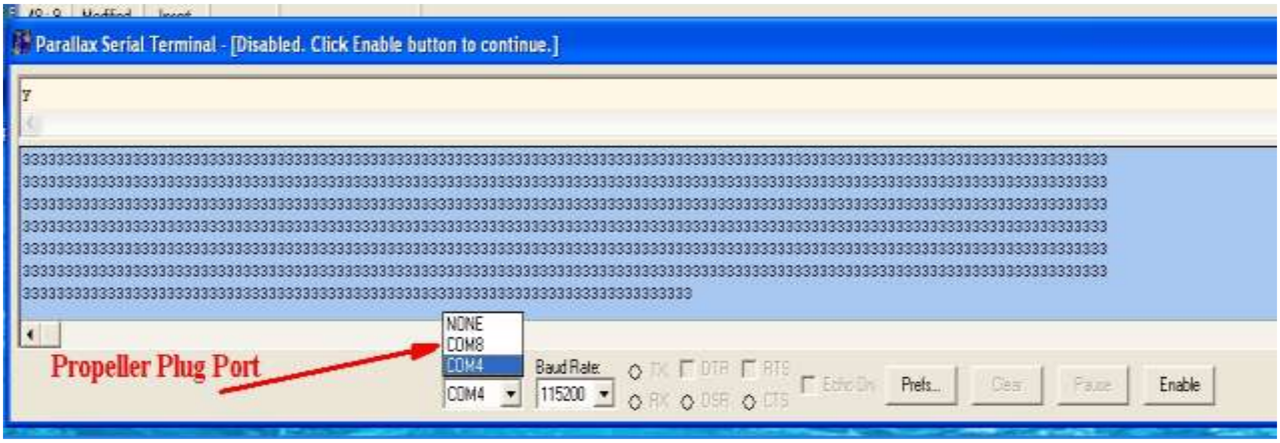
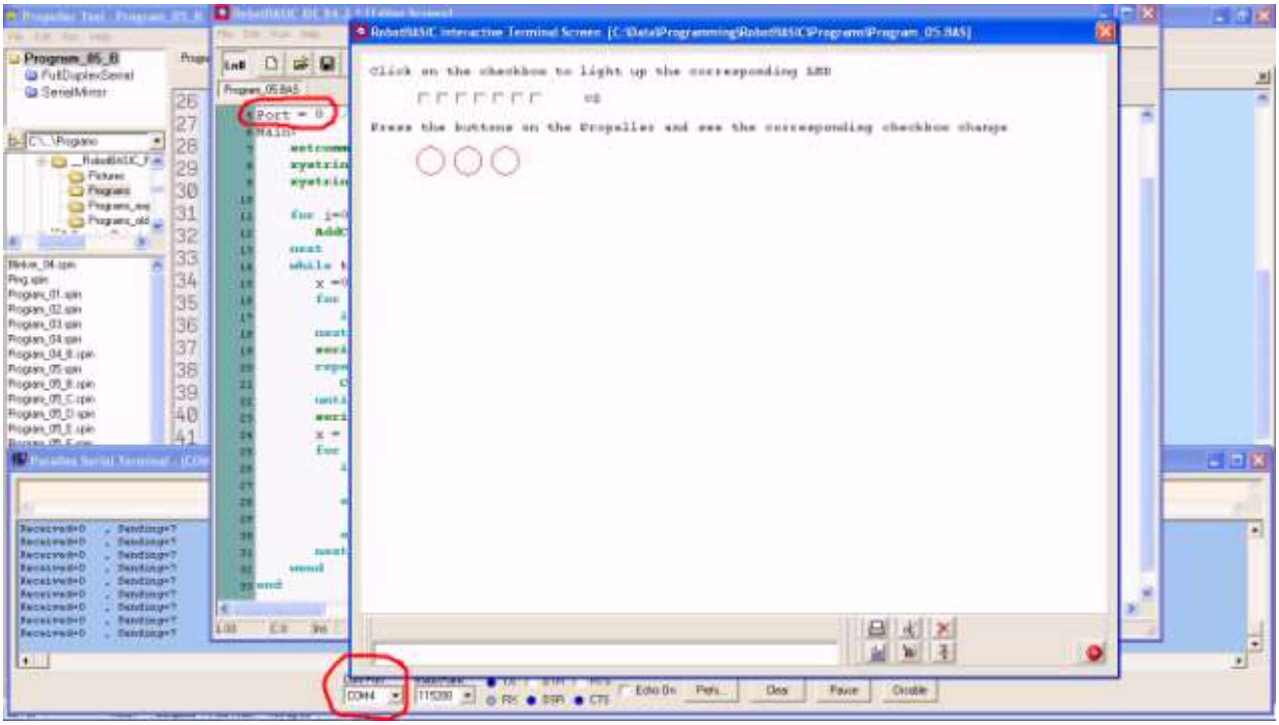

Parallax Serial Terminal - [Disabled. Click Enable button to continue.]

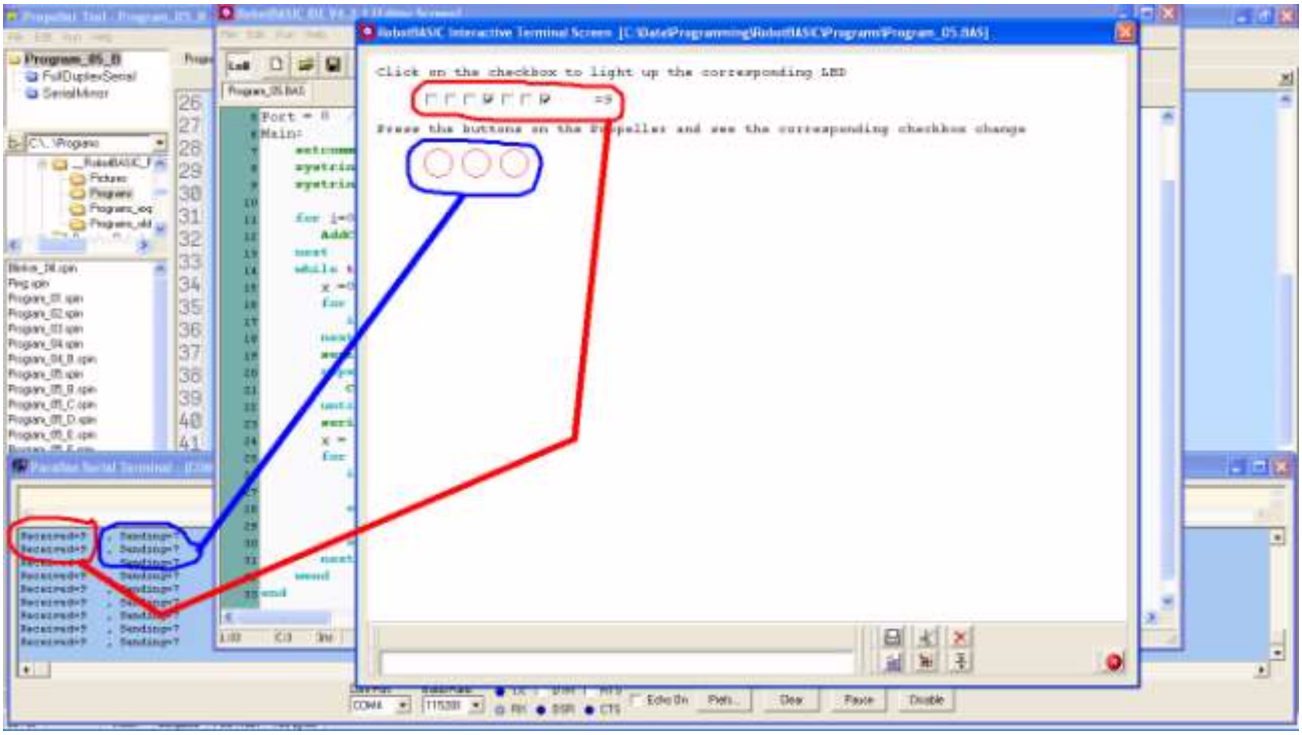
Figure 3.2: PST showing a list of available com ports.


RobotBASIC Interactive Terminal Screen [C:\Data\Programming\RobotBASIC\Programs\Program_05.BAS]

```
Click on the checkbox to light up the corresponding LED

Press the buttons on the Propeller and see the corresponding checkbox change
```

Figure 5.1: The Arrangement of the windows during the debugging session of Program_05_B.Spin



```
Click on the checkbox to light up the corresponding LED
    ☐ ☐ ☐ ☑ ☐ ☐ ☑      =9
Press the buttons on the Propeller and see the corresponding checkbox change
```

Figure 5.2: Example output of the debugging session of Program_05_B.Spin

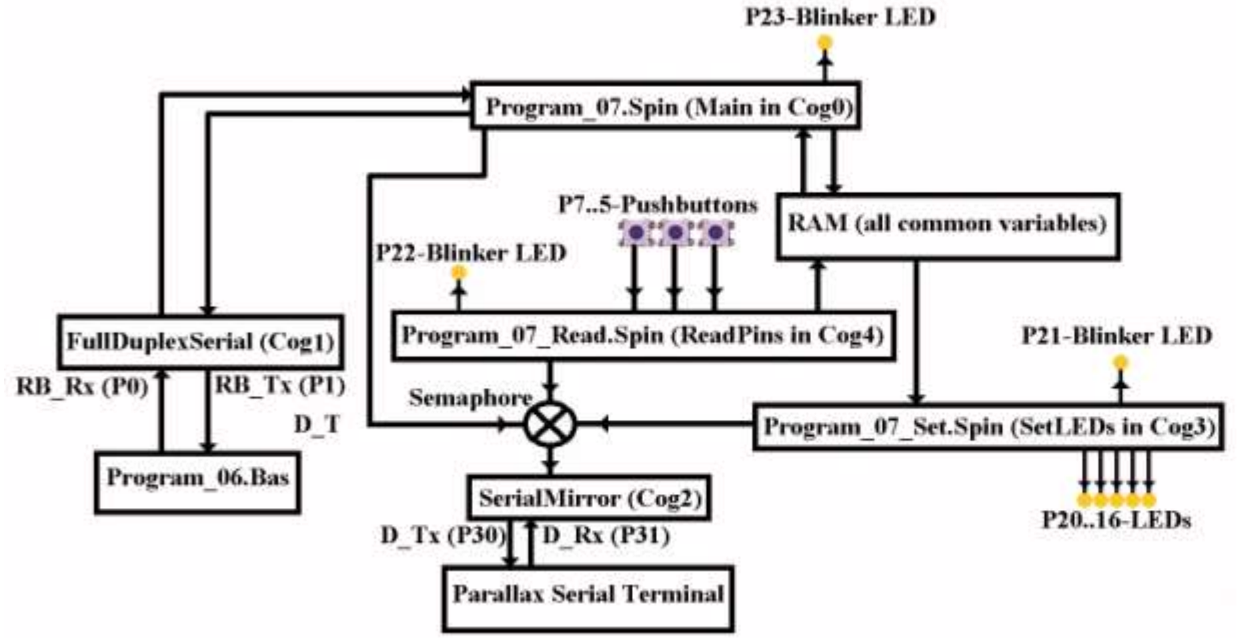
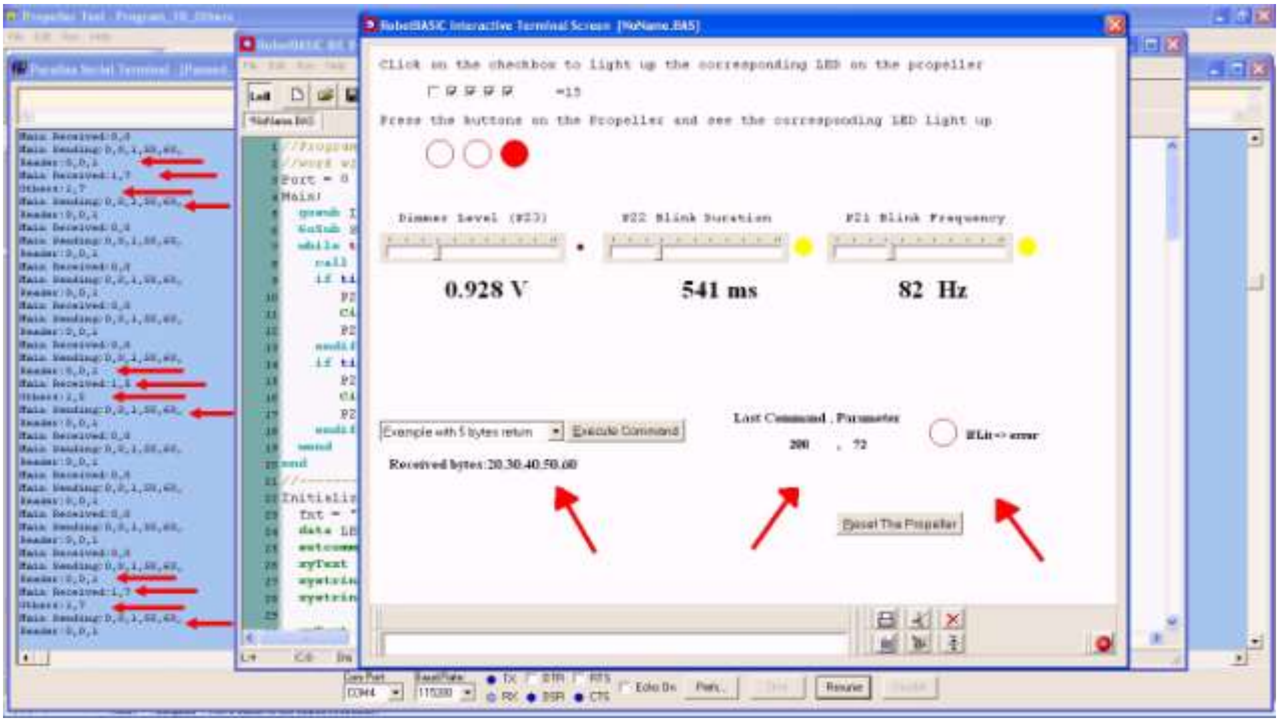


Figure 5.4: A schematic of the various objects and cogs in the new system showing how they interact. Cog numbers are just for reference, they are not necessarily the actual order.



Click on the checkbox to light up the corresponding LED on the propeller

☐ ☑ ☑ ☑ ☑ =15

Press the buttons on the Propeller and see the corresponding LED light up

○ ○ ●

Received bytes:20,30,40,50,60

Figure 6.1: A screenshot of Program_10_B.Bas in action. The arrows point to areas of interest as discussed in the text.

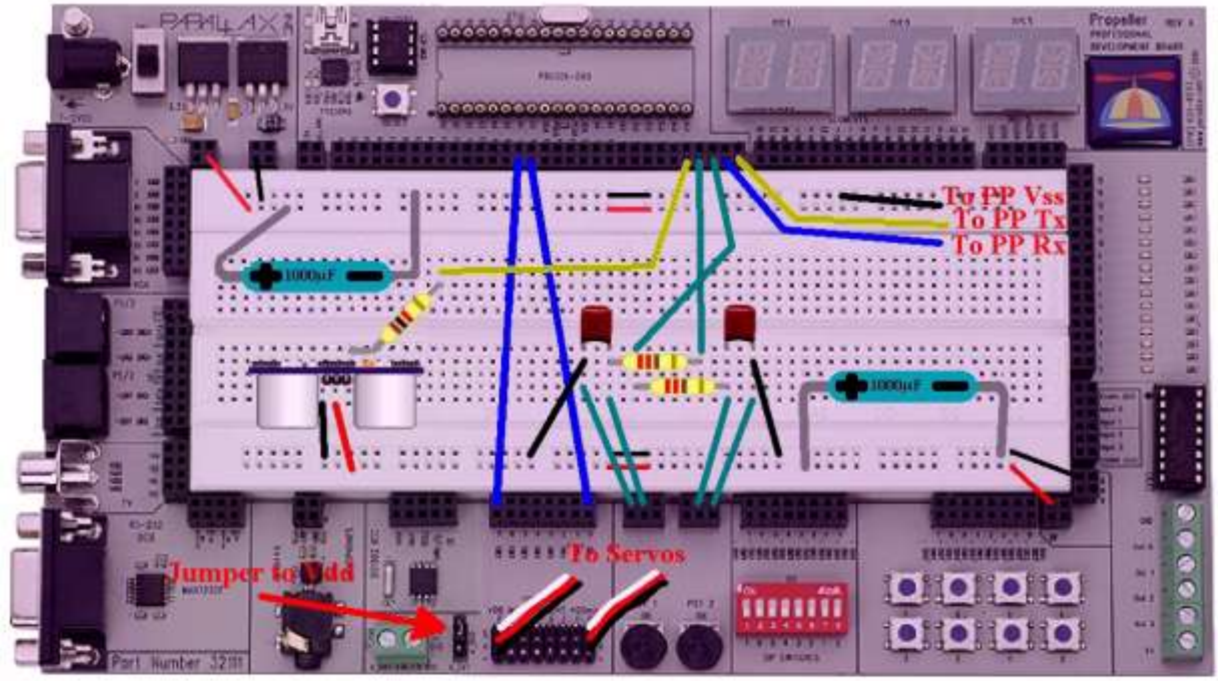


Figure 7.1: Some hardware connections on the PPDB. LED and Pushbutton connections are not shown; see Figure 2.4, but P16 and P17 are now used for Servomotors. Also see Figure B.3 for connection schematics.

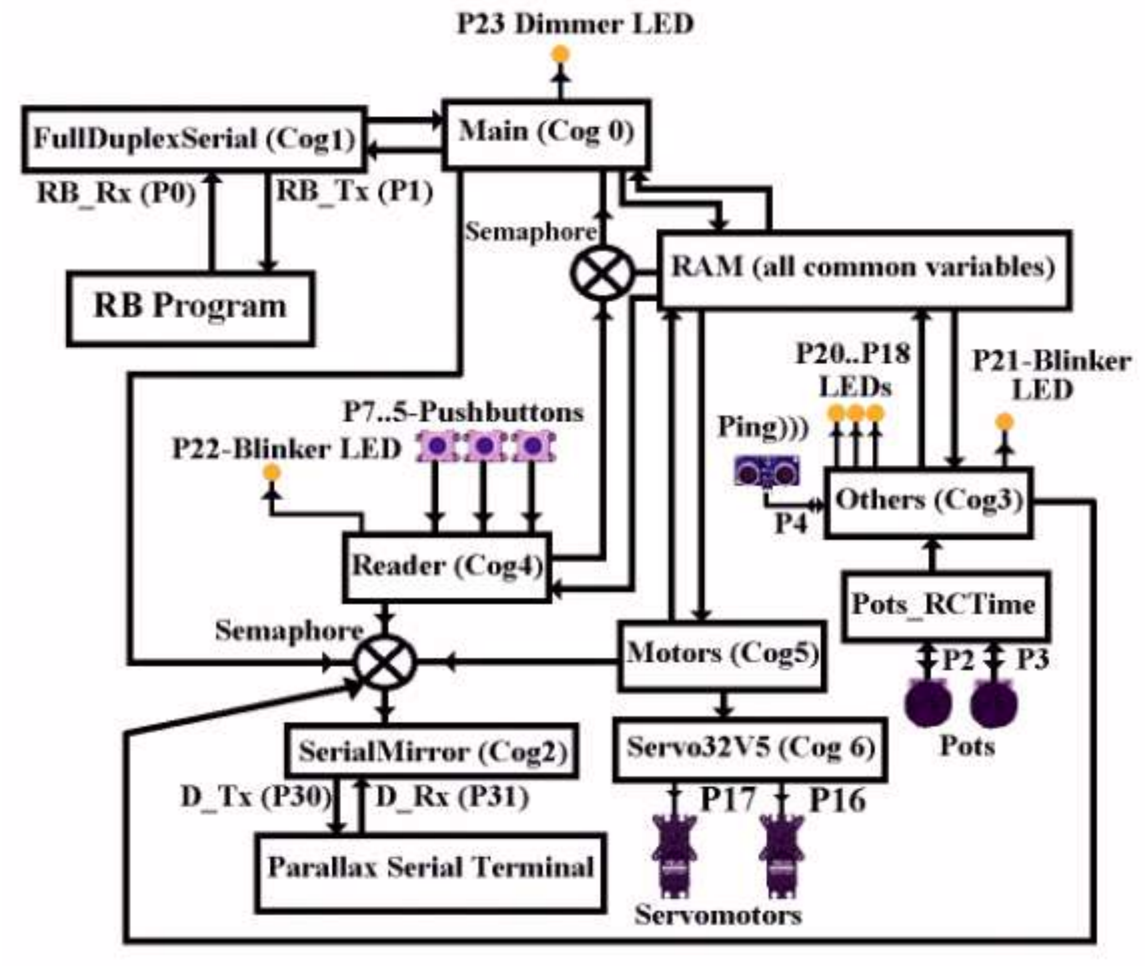
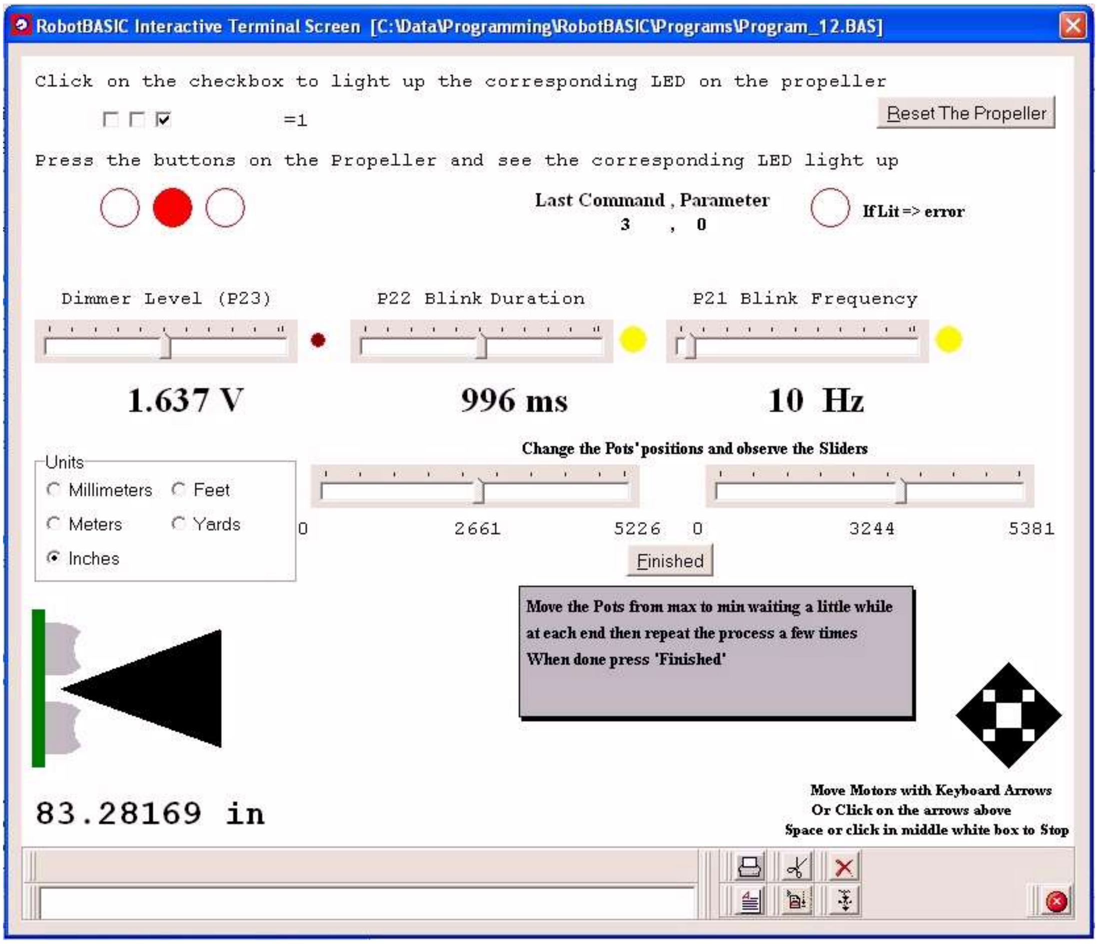


Figure 7.8: Conceptual Schematic of the System. Cog numbers are not actual numbers.



RobotBASIC Interactive Terminal Screen [C:\Data\Programming\RobotBASIC\Programs\Program_12.BAS]

Click on the checkbox to light up the corresponding LED on the propeller

☐ ☐ ☑ =1

Press the buttons on the Propeller and see the corresponding LED light up

○ ● ○

Last Command , Parameter
3 , 0

○ If Lit => error

Dimmer Level (P23) 1.637 V

P22 Blink Duration 996 ms

P21 Blink Frequency 10 Hz

Change the Pots' positions and observe the Sliders

0 2661 5226

0 3244 5381

Units
○ Millimeters ○ Feet
○ Meters ○ Yards
● Inches

Move the Pots from max to min waiting a little while at each end then repeat the process a few times
When done press 'Finished'

83.28169 in

Move Motors with Keyboard Arrows
Or Click on the arrows above
Space or click in middle white box to Stop

Figure 7.9: Screen Shot of Program_11_Advanced.Bas in action. Notice that a pots calibration is currently taking place.

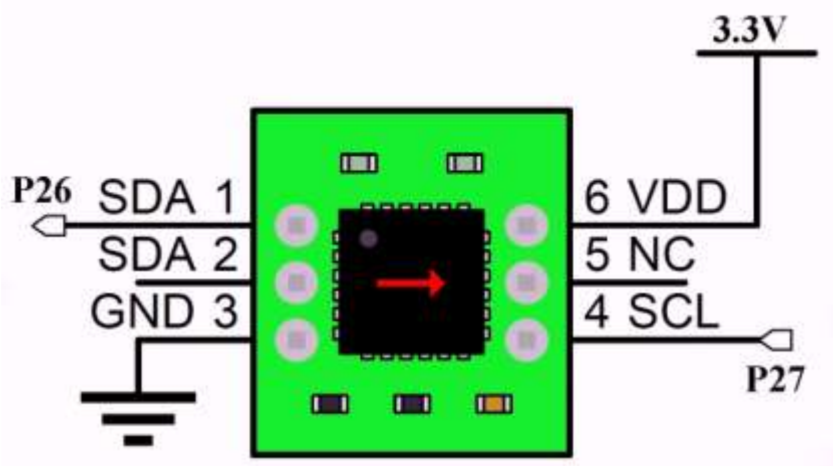


Figure 8.1: HMC6352 Connection Schematic. Notice the reference axis of the module is important.

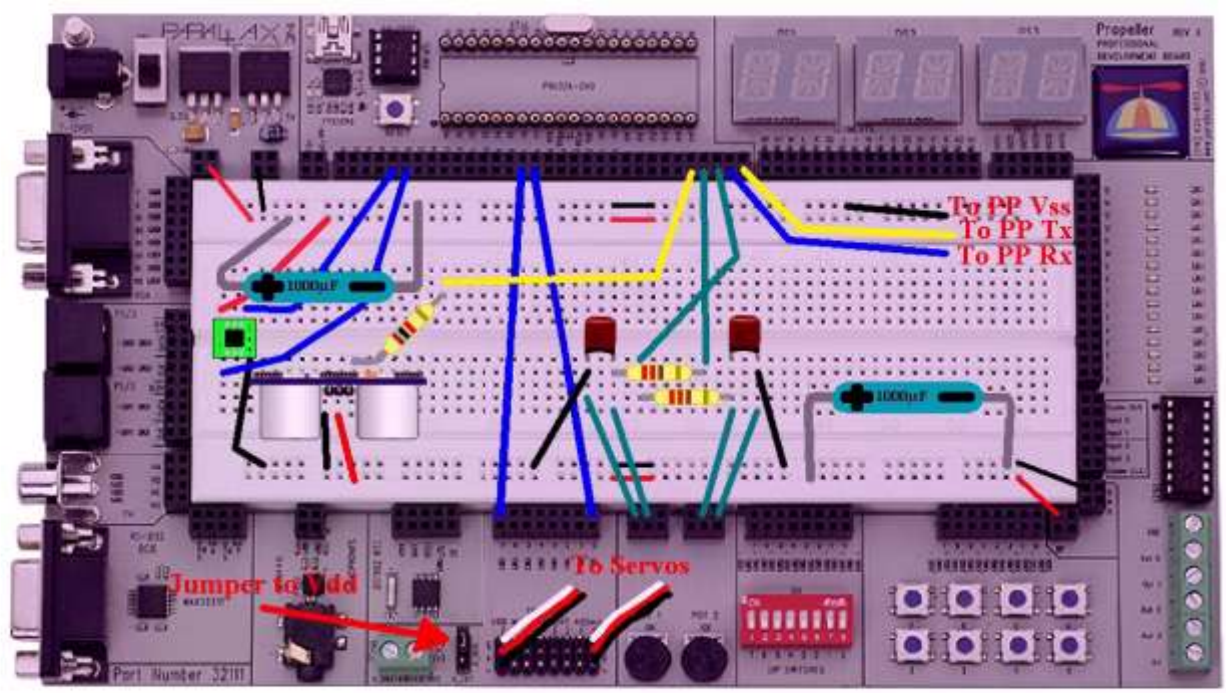


Figure 8.2: PPDB With the HMC6352 and all the previously installed hardware. LED and Pushbutton connections are not shown, see Figure 2.4, but P16 and P17 are now used for Servomotors.

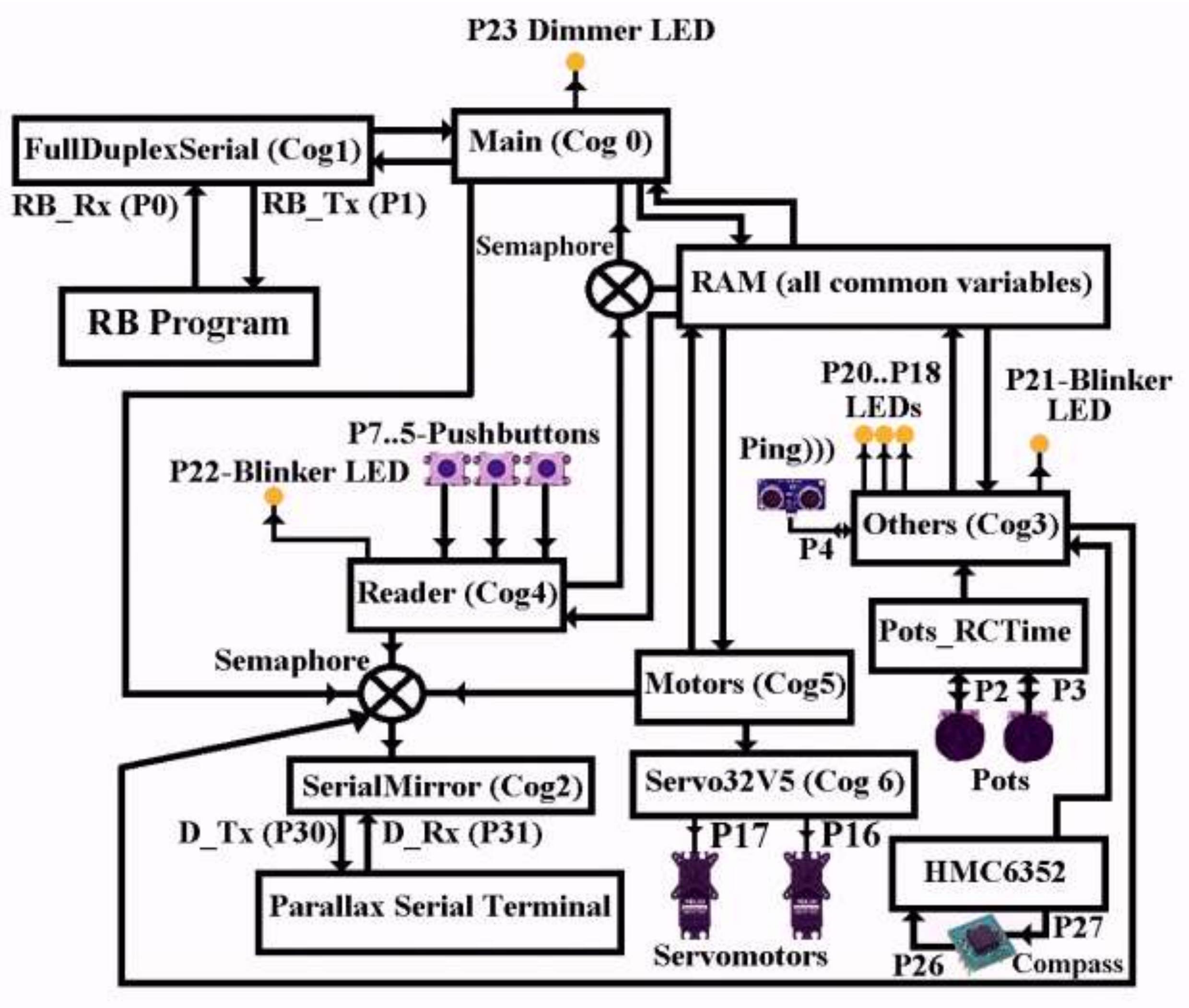
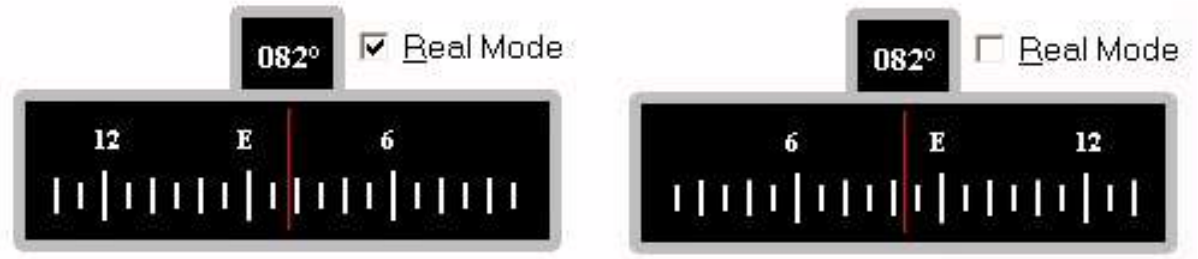


Figure 8.3: The Conceptual Schematic of the system with all the new hardware.


082°

☑ Real Mode

082°

☐ Real Mode

Figure 8.4: Screenshot of Compass_Animation.BAS simulating an authentic looking GUI Compass Instrument. Notice the difference between the real mode (left) and enhanced mode (right).

8.2 A Procedural Strategy for Adding Other Hardware

As you have seen so far, because of the way the system is designed, adding hardware is extremely simple and routine. As a matter of fact, the hardware we added covers almost every category of hardware that you are likely to want to incorporate into your system.

List 1: Categories of Hardware

a) Digital hardware with On/Off type I/O (Pushbuttons, LEDs)
b) Digital to Analog output (Dimmer LED)
c) Pulsating Frequency output (Blinking LEDs and Speaker)
d) Analog To Digital input with RC-Time (Pots)
e) Controlling Servomotors (Servomotors)
f) Counting Time Intervals (Ping and RC-Time)
g) I^2C I/O (Compass)
h) RS232 I/O (FDS, SM)
i) Using Counters (in Duty, NCO, and Edge Detector modes)

List 2: Programming Techniques Required to Develop the Firmware

j) Using Semaphores and Flags
k) Using Parallelism
l) Using Polling
m) Sharing RAM
n) Inter-Cog communications and control
o) Creating objects and methods

Just about any hardware that you are likely to want to add as well as the programming techniques required to add them to the firmware are most likely to belong to one of the above categories. Let's have a look at some hardware that we may wish to add to a project:

Table 8.1: Possible Hardware and its Category

Hardware	Category
Bumper Switch[30]	a
Infrared Proximity Sensors[31]	a or c
QTI Line Sensors[14]	a or d
PIR Movement Sensor[32]	a
Turret[33]	e
Accelerometer[34]	g
GPS[35]	h or g
DC motors[36]	e
Thermometer[37]	G
2-Axis Joystick[38]	D
Sound Impact Sensor[39]	A
5-Way button[40]	A
Piezoelectric Speaker[27]	I
Quadrature System[41]	H

8.2.1 Commands in the Protocol So Far

In our protocol so far we have allowed for many possible commands and Table 8.2 below is a good overview.

See Tables B.1, B.2, and B.3 in Appendix B below

8.2.2 A Procedural Strategy For Extending the Hardware

Much of the hardware you may wish to add is likely to be just a matter of deciding what category it is under (List 1 and List 2 and Table 8.1) and then looking at the commands in Table 8.2 to decide which command resembles it best. Once you have decided on this, use the command from Table 8.2 as a template for adding the new hardware. It is not just hardware that we might want to add. We may also want to add more housekeeping commands.

List 3: There are three types of commands:

1. Ones that set/change system parameters (**Main** object but can be any of the objects)
2. Ones that do something in the background and do not need to be commanded (**Reader** object)
3. Ones that carry out a task and then
    a. Do not return data (**Motors** object but can be **Others** too)
    b. Return data in the last two bytes of the primary send buffer (**Others** object)
    c. Return data in all or some bytes of the secondary send buffer (**Others** object)

<u>Procedure For Adding a New Hardware or Command</u>

To add a new command you need to

i. Decide which category of hardware it is from Lists 1 and 2 and Table 8.1.
ii. Decide what type of command it will be from List 3.
iii. Select a template command from Table 8.2.
iv. Modify the appropriate object to incorporate the methods needed to interact with the hardware and fill the send buffer if required. If you decide that you need a new object then use one of the existing objects as a template and modify it as needed.
v. Add any constants in the **CON** section.
vi. Add any variables in the **Var** or **Dat** section.
vii. Instantiate any required supporting objects in the **Obj** section and invoke their **Start()** methods in the **Initialization** method.
viii. Decide on a code for the command (make sure there is no clashing) and what parameters it has to be passed.
ix. Add the Case statement in the Case block to call the method. This should follow the template command.
x. Add the Case Statement in the **Main** object to allow for the new case statement in the subordinate object. This should follow the template command.

To illustrate the process we will now add new commands to:

❑ Allow for actuating the motors separately in any direction for a certain number of steps or to keep them on (Section 8.2.3).
❑ Allow the Ping))) mounted on a turret to be turned by 90 degrees right and left before measuring the distance (Section 8.2.4).
❑ Save all the system parameters to the EEPROM. We will also extend the system to read them from the EEPROM upon boot up if there are any valid saved ones (Section 8.3) and also allow for resetting them to factory settings.
❑ Add an Accelerometer (Section 8.4).
❑ Add three QTI infrared line sensors (Section 8.5).
❑ Add a speaker similar to Chapter 5.8 (Section 8.6).

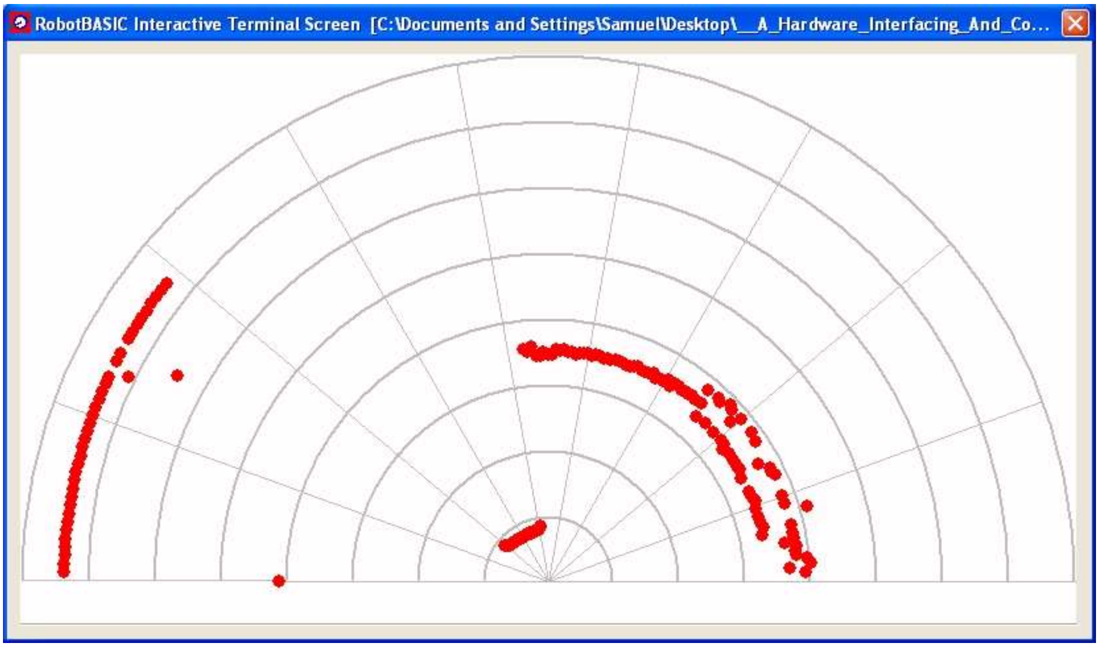
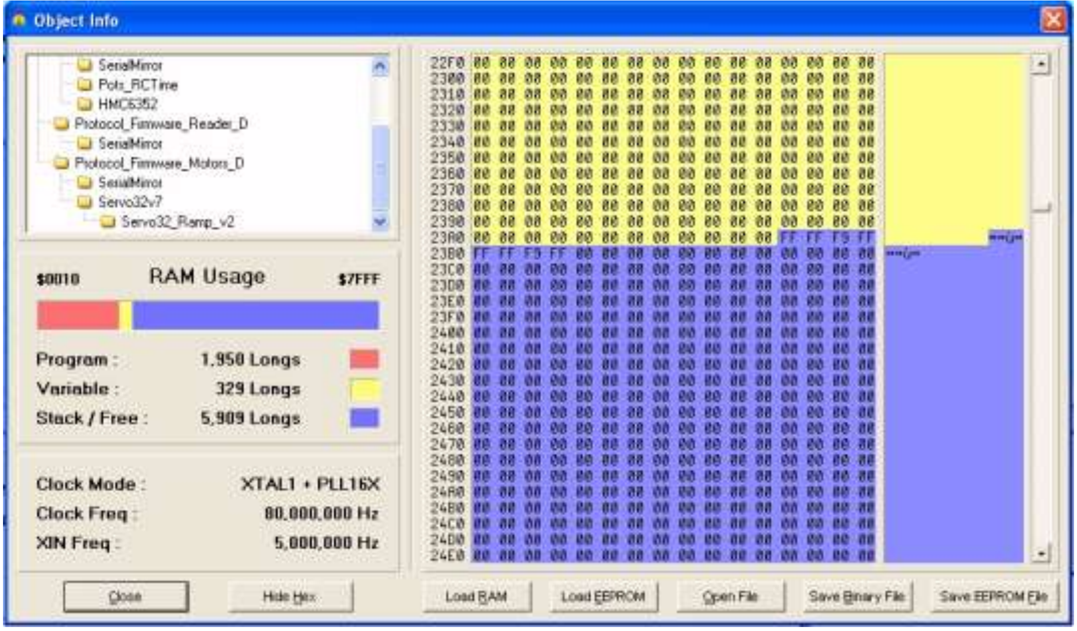

RobotBASIC Interactive Terminal Screen [C:\Documents and Settings\Samuel\Desktop\__A_Hardware_Interfacing_And_Co...

Figure 8.5: Screenshot of Turret_Radar.Bas in action.



$0010 RAM Usage $7FFF

Program : 1,950 Longs 🟥
Variable : 329 Longs 🟨
Stack / Free : 5,909 Longs 🟦

Clock Mode : XTAL1 + PLL16X
Clock Freq : 80,000,000 Hz
XIN Freq : 5,000,000 Hz

Figure 8.6: Memory Utilization

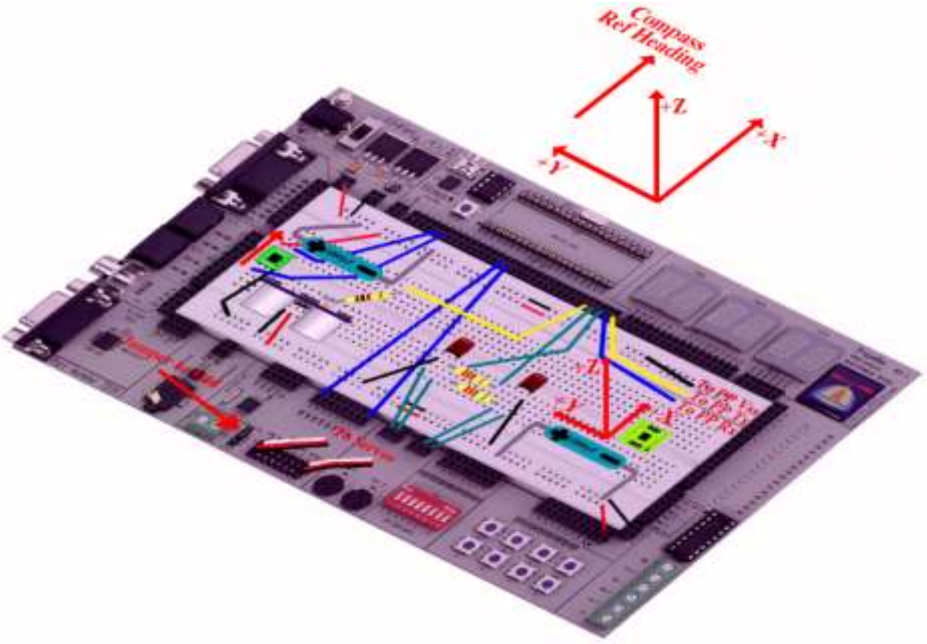
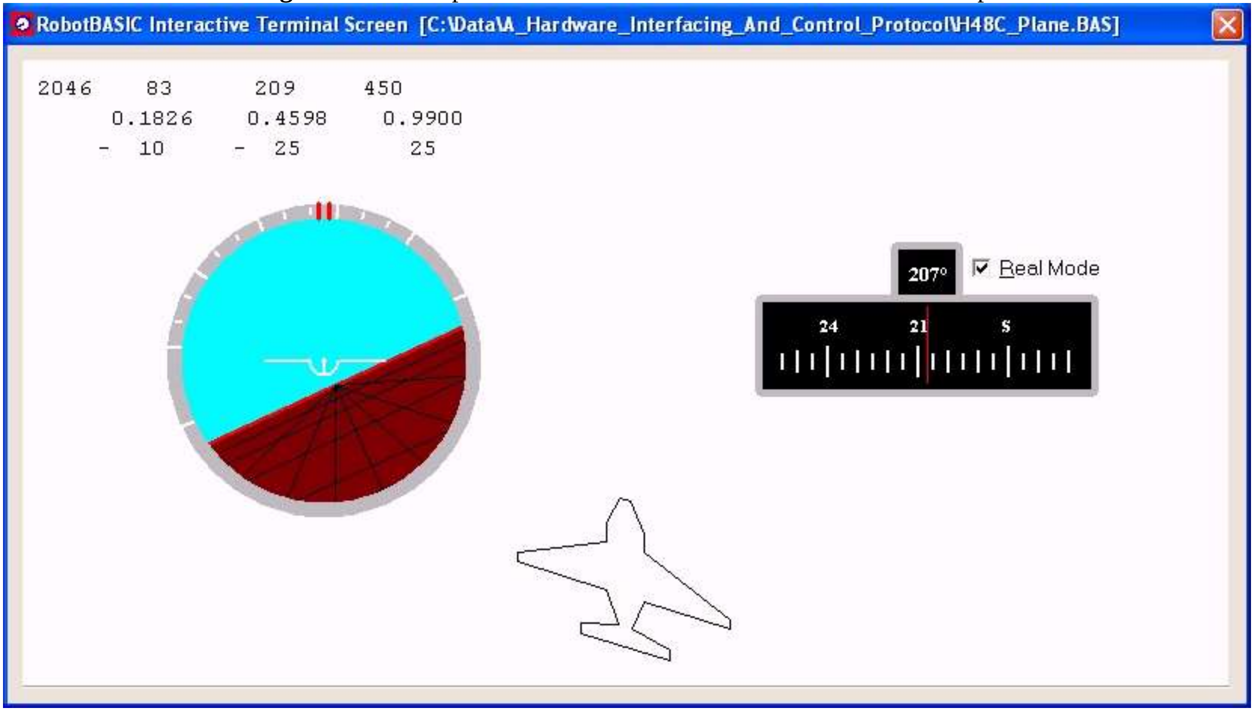
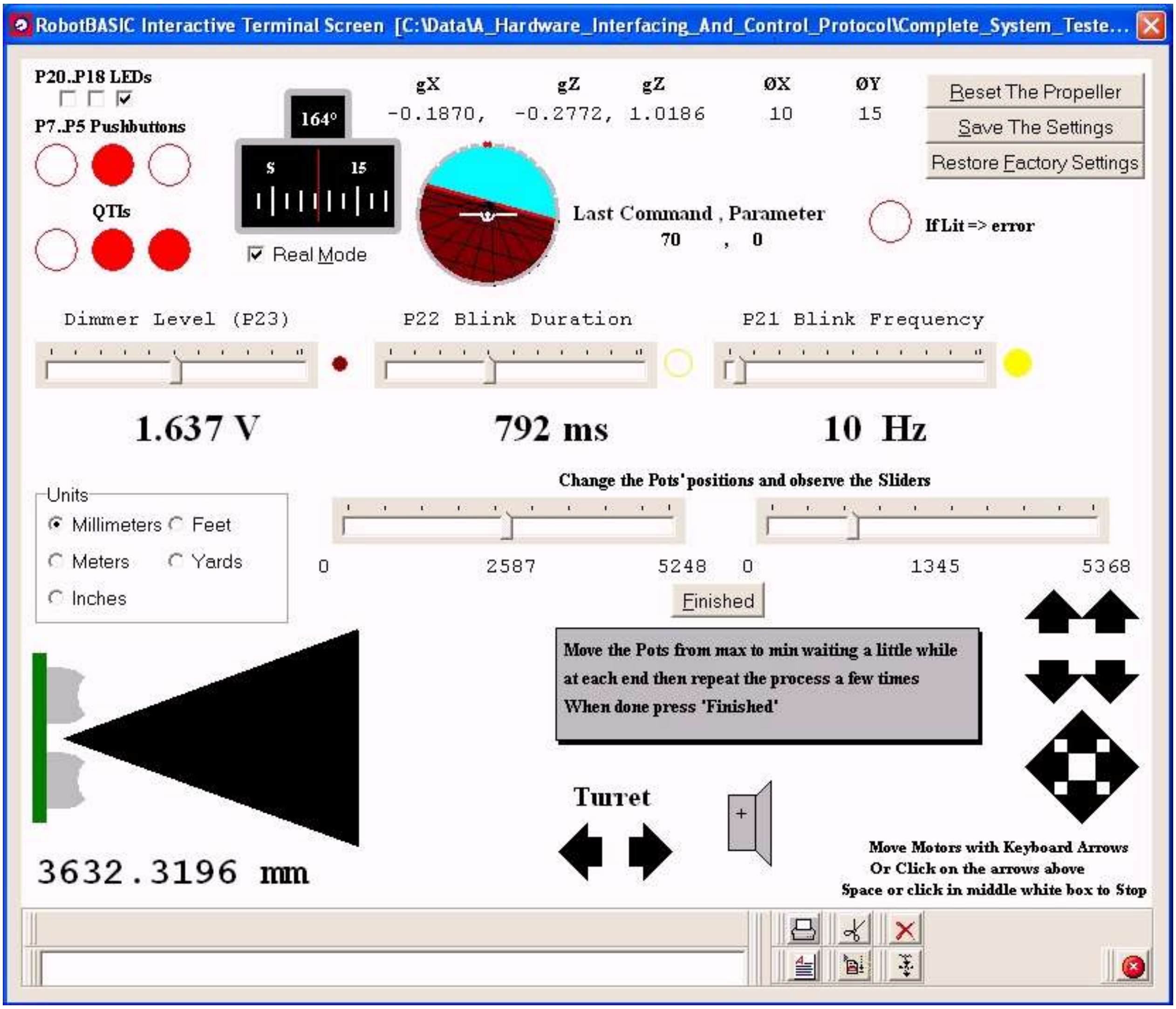


Figure 8.10: Compass and Accelerometer orientation and axes setup.


RobotBASIC Interactive Terminal Screen [C:\Data\A_Hardware_Interfacing_And_Control_Protocol\H48C_Plane.BAS]

```
2046     83       209      450
       0.1826    0.4598    0.9900
     -   10     -   25       25
```

Figure 8.11: Screenshot of H48C_Plane.Bas



P20..P18 LEDs ☐ ☐ ☑

☑ Real Mode

gX gZ gZ ØX ØY

-0.1870, -0.2772, 1.0186 10 15

Last Command , Parameter

70 , 0

Dimmer Level (P23) 1.637 V

P22 Blink Duration 792 ms

P21 Blink Frequency 10 Hz

Units: ● Millimeters ○ Feet ○ Meters ○ Yards ○ Inches

Change the Pots' positions and observe the Sliders

0 2587 5248

0 1345 5368

Move the Pots from max to min waiting a little while at each end then repeat the process a few times When done press 'Finished'

3632.3196 mm

Move Motors with Keyboard Arrows
Or Click on the arrows above
Space or click in middle white box to Stop

Figure 8.15: Complete_System_Tester.Bas Screenshot. Pots calibration is in progress.

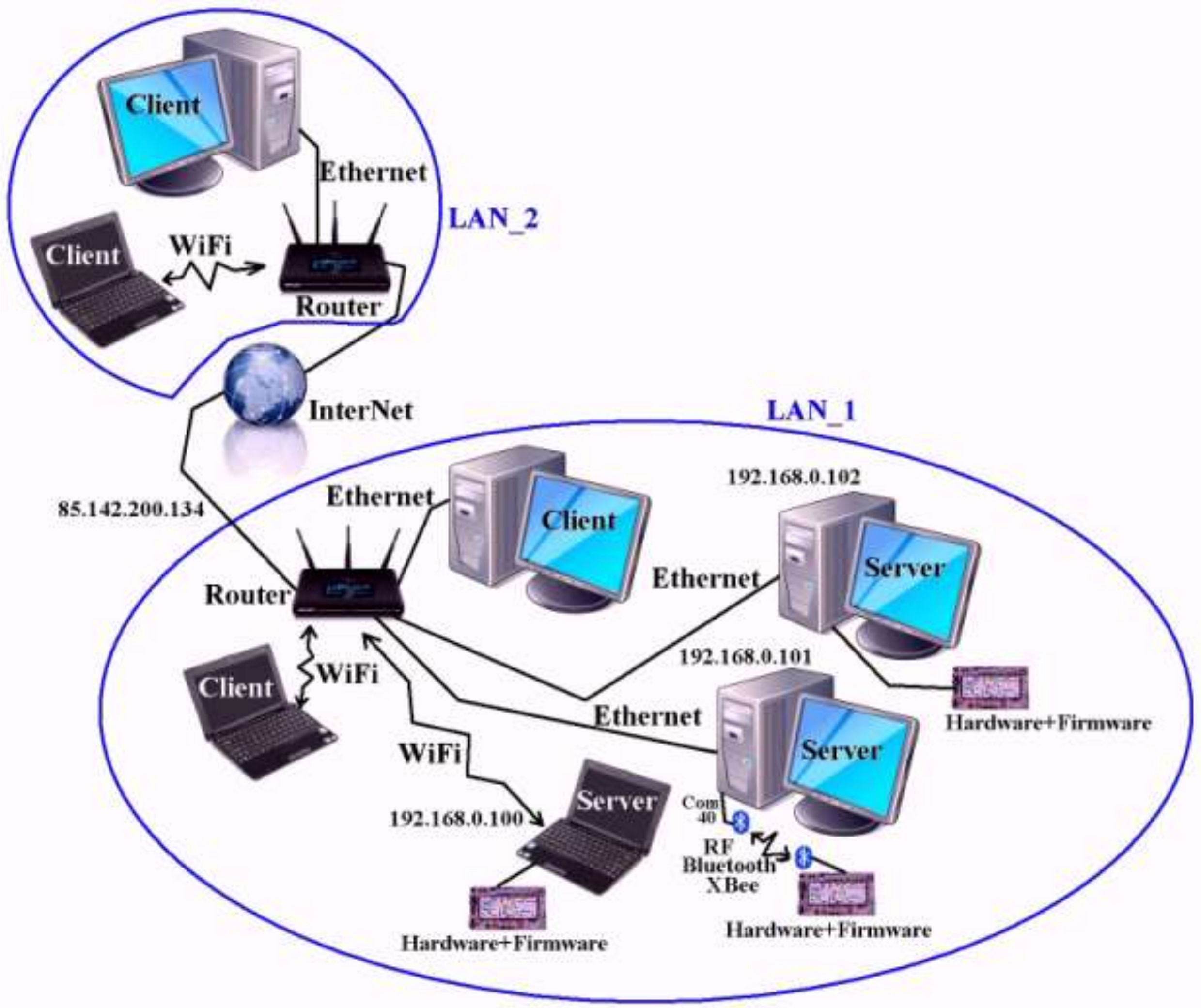


Figure 9.7: Various configurations for LAN or Internet Remote Control. Shown is the TCP model but the same topology would also apply to UDP.

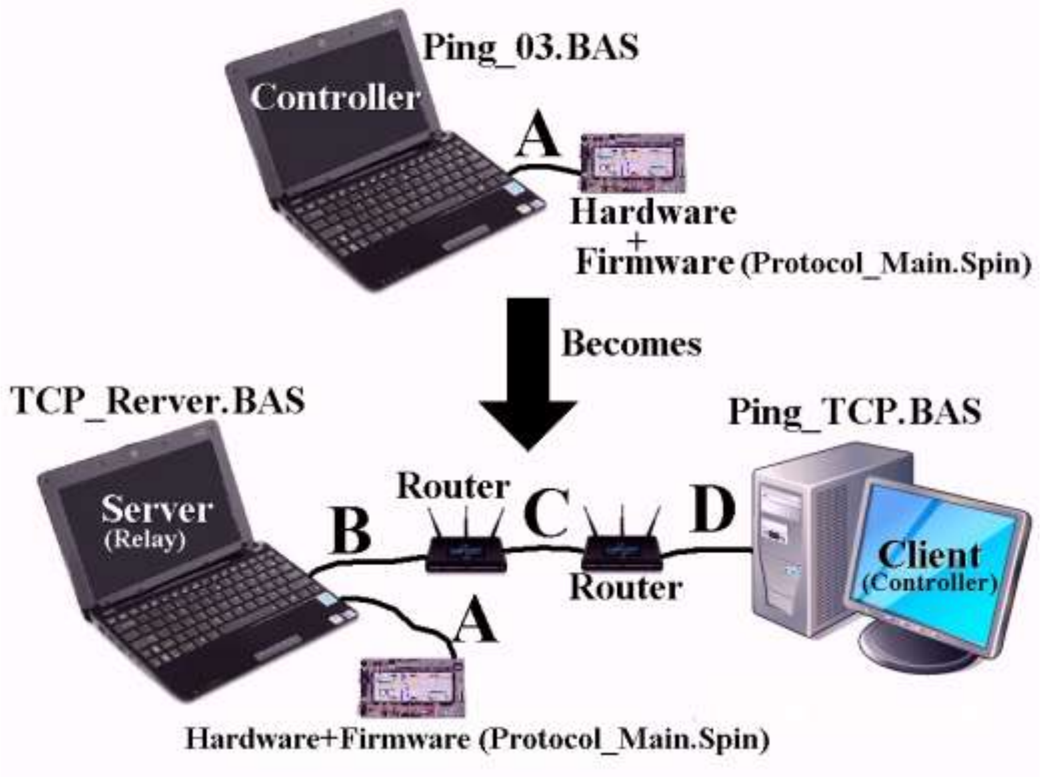


Figure 9.8: Converting our setup to work over a LAN or the Internet.

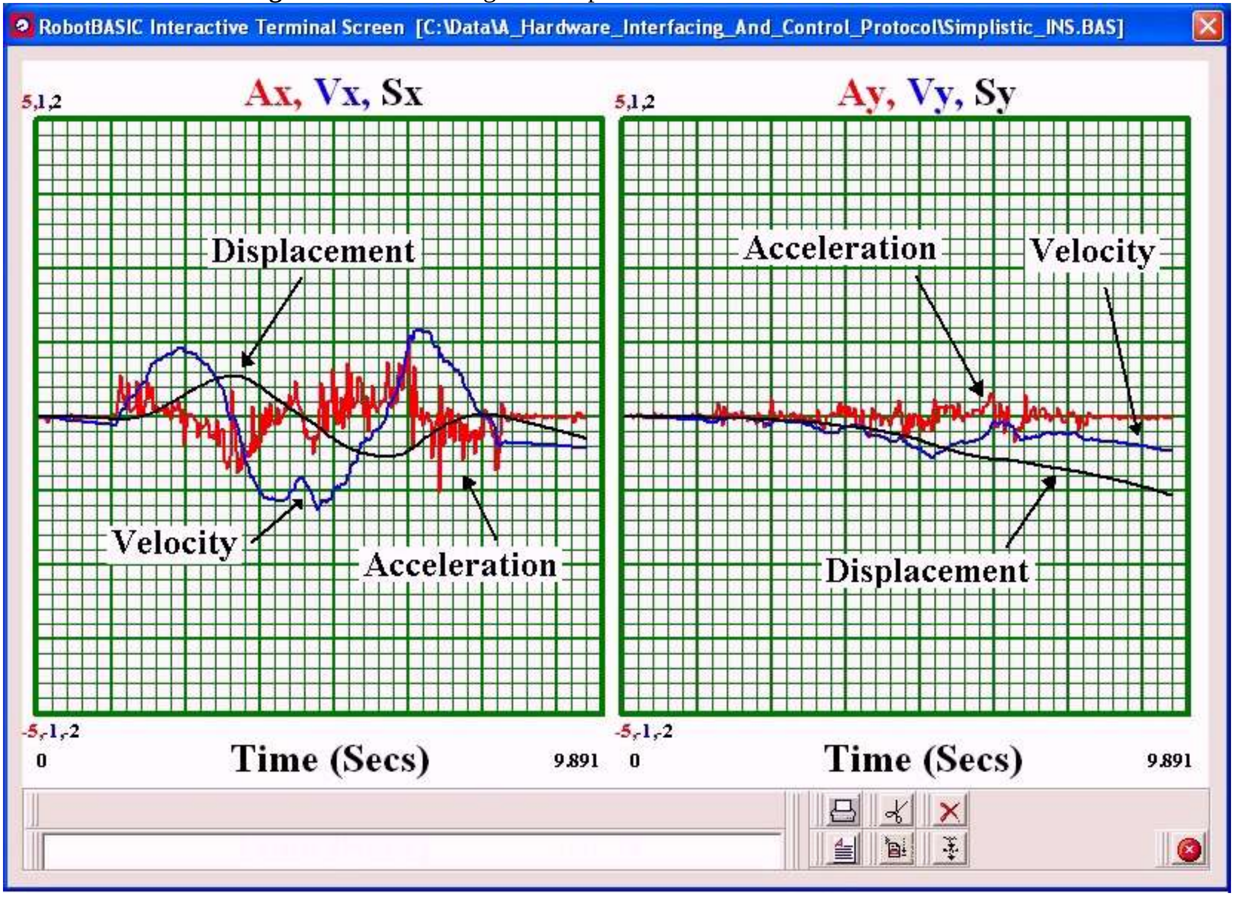


Figure 10.4: Results of running Simplistic_INS.Bas

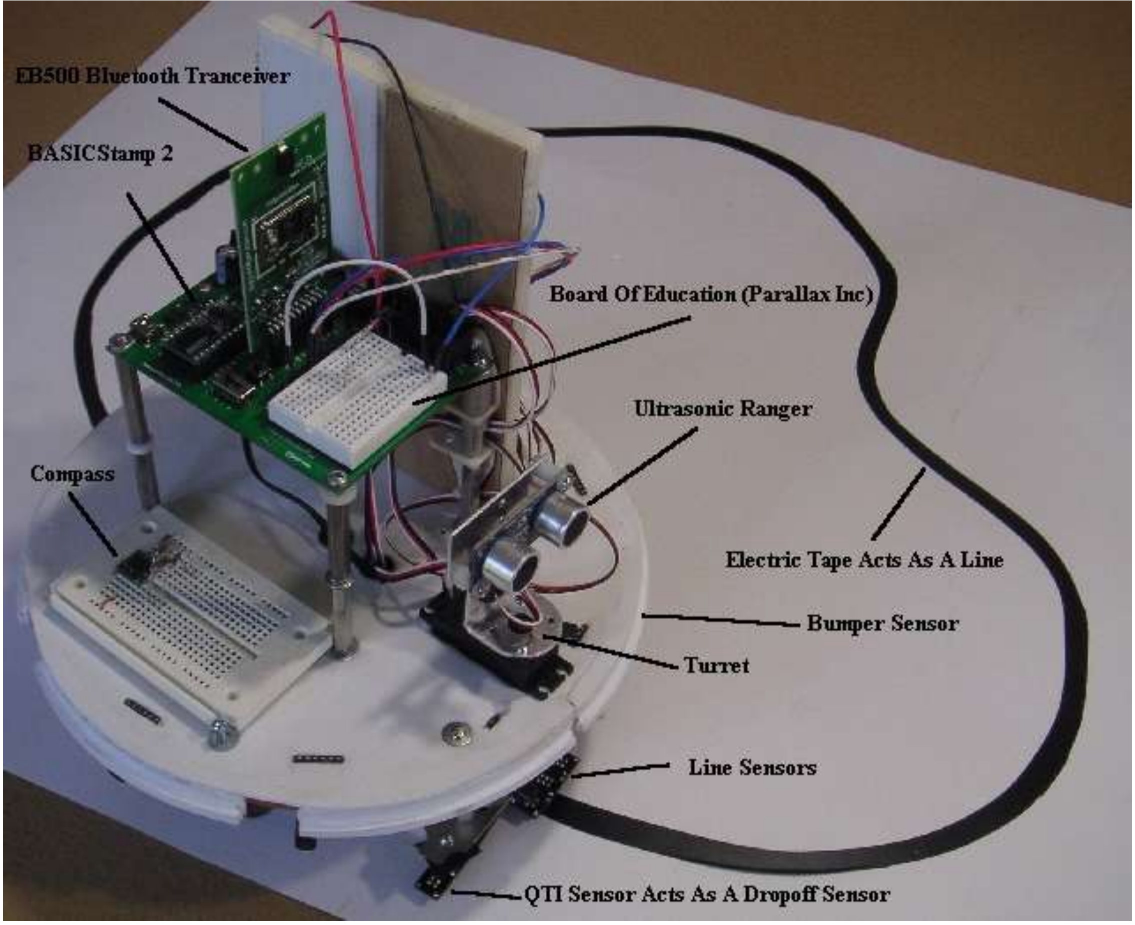

EB500 Bluetooth Tranceiver
BASICStamp 2
Board Of Education (Parallax Inc)
Ultrasonic Ranger
Compass
Electric Tape Acts As A Line
Bumper Sensor
Turret
Line Sensors
QTI Sensor Acts As A Dropoff Sensor

Figure 11.3: Enhanced Boe-Bot for working with the RobotBASIC Protocol

Appendices

Appendix A
Web Links

1- Propeller Chip
www.parallax.com/propeller
2- RobotBASIC
http://www.RobotBASIC.Com
3- USBmicro
http://www.usbmicro.com/
4- A researcher at MIT
mit.edu/whall/www/heli/paper/node3.html#SECTION00030000000000000
5- Propeller Demo Board (#32100 at www.parallax.com)
http://www.parallax.com/Store/Microcontrollers/PropellerDevelopmentBoards/tabid/514/CategoryID/73/List/0/SortField/0/Level/a/ProductID/340/Default.aspx
6- Propeller Professional Development (#32111 at www.parallax.com)
http://www.parallax.com/Store/Microcontrollers/PropellerDevelopmentBoards/tabid/514/CategoryID/73/List/0/SortField/0/Level/a/ProductID/515/Default.aspx
7- Propeller Plug (#32201 at www.parallax.com)
http://www.parallax.com/Store/Microcontrollers/PropellerTools/tabid/143/CategoryID/19/List/0/SortField/0/Level/a/ProductID/398/Default.aspx
8- Parallax USB2Ser Development Tool (#28024 at www.parallax.com)
http://www.parallax.com/Store/Accessories/CablesConverters/tabid/166/CategoryID/40/List/0/SortField/0/catpageindex/1/Level/a/ProductID/32/Default.aspx
9- Parallax USB to Serial (RS-232) Adapter (#28031 at www.parallax.com)
http://www.parallax.com/Store/Accessories/CablesConverters/tabid/166/CategoryID/40/List/0/SortField/0/catpageindex/1/Level/a/ProductID/379/Default.aspx
10- Ping))) Ultrasonic Sensor (#28015 at www.parallax.com)
http://www.parallax.com/Store/Sensors/AllSensors/tabid/760/CategoryID/46/List/0/SortField/0/catpageindex/3/Level/a/ProductID/92/Default.aspx
11- Parallax Continuous Rotation Servo Motors (#900-00008 at www.parallax.com)
http://www.parallax.com/Store/Robots/RoboticComponents/tabid/198/CategoryID/70/List/0/SortField/0/catpageindex/2/Level/a/ProductID/102/Default.aspx
12- Honeywell HMC6352 Compass Module (#29323 at www.parallax.com)
http://www.parallax.com/Store/Sensors/CompassGPS/tabid/173/CategoryID/48/List/0/SortField/0/Level/a/ProductID/596/Default.aspx
13- H48C Tri-Axis Accelerometer module (#28026 at www.parallax.com)
http://www.parallax.com/Store/Sensors/AccelerationTilt/tabid/172/CategoryID/47/List/0/SortField/0/Level/a/ProductID/97/Default.aspx

14- QTI Infrared Line Sensor (#555-27401 at www.parallax.com)
    http://www.parallax.com/Store/Sensors/ObjectDetection/tabid/176/CategoryID/51/List/0/SortField/0/Level/a/ProductID/100/Default.aspx
15- Propeller Object Exchange Library (ObEx)
    http://obex.parallax.com/
16- SerialMirror.Spin (SM)
    http://obex.parallax.com/objects/189/
17- FullDuplexSerial.Spin (FDS)
    http://obex.parallax.com/objects/54/
18- RobotBASIC_Serial_IO.pdf
    http://www.robotbasic.org/resources/RobotBASIC_Serial_IO.pdf
19- RobotBASIC_To_PropellerChip_Comms.pdf
    http://www.robotbasic.org/resources/RobotBASIC_To_PropellerChip_Comms.pdf
20- RobotBASIC_To_BS2_Comms.pdf
    http://www.robotbasic.org/resources/RobotBASIC_To_BS2_Comms.pdf
21- Motor controller module (HB-25 Motor Controller (#29144 at www.parallax.com))
    http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/SortField/0/Level/a/ProductID/64/Default.aspx
22- RobotBASIC forum.
    http://tech.groups.yahoo.com/group/RobotBasic/
23- Parallax Propeller Forum.
    http://forums.parallax.com
24- Servo Pal (#28824 at www.parallax.com)
    http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/SortField/0/catpageindex/2/Level/a/ProductID/481/Default.aspx
25- PWM Pal (#28020 at www.parallax.com)
    http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/SortField/0/Level/a/ProductID/67/Default.aspx
26- Servo32V7.Spin
    http://obex.parallax.com/objects/51/
27- Piezoelectric Speaker (#900-00001 at www.parallax.com)
    http://www.parallax.com/Store/Accessories/Sound/tabid/164/CategoryID/38/List/0/SortField/0/Level/a/ProductID/106/Default.aspx
28- HMC6352.Spin
    http://www.parallax.com/Portals/0/Downloads/docs/prod/sens/HMC6352-Propeller-Examples-V1.0.zip
29- Gyroscopic Device (LISY300 Gyroscope Module #27922 at www.parallax.com)
    http://www.parallax.com/Store/Sensors/AccelerationTilt/tabid/172/CategoryID/47/List/0/SortField/0/Level/a/ProductID/588/Default.aspx
30- Bumper Switch
    http://www.pololu.com/catalog/product/1403
31- Infrared Proximity Sensors
    http://www.pololu.com/catalog/product/1134
32- PIR Movement Sensor (#555-28027 at www.parallax.com)
    http://www.parallax.com/Store/Microcontrollers/BASICStampModules/tabid/134/ProductID/83/List/1/Default.aspx?SortField=UnitCost,ProductName
33- Turret (Ping))) mounting Bracket Kit #570-28015 at www.parallax.com)
    http://www.parallax.com/Store/Robots/AllRobots/tabid/755/ProductID/248/List/0/Default.aspx?SortField=ProductName,ProductName
34- Accelerometer (Hitachi H48C Tri-Axis Accelerometer Module #28026 at www.parallax.com)
    http://www.parallax.com/Store/Sensors/AccelerationTilt/tabid/172/CategoryID/47/List/0/SortField/0/Level/a/ProductID/97/Default.aspx
35- GPS (Parallax GPS Receiver Module (#28146 at www.parallax.com)
    http://www.parallax.com/Store/Sensors/CompassGPS/tabid/173/CategoryID/48/List/0/SortField/0/Level/a/ProductID/396/Default.aspx

http://www.parallax.com/Store/Sensors/CompassGPS/tabid/173/CategoryID/48/List/0/SortField/0/Level/a/ProductID/396/Default.aspx

36- DC motors (HB-25 motor controller #29144 at www.parallax.com)
http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/SortField/0/Level/a/ProductID/64/Default.aspx

37- Thermometer (Sensirion Temperature/Humidity Sensor #28018 at www.parallax.com)
http://www.parallax.com/Store/Sensors/TemperatureHumidity/tabid/174/CategoryID/49/List/0/SortField/0/Level/a/ProductID/94/Default.aspx

38- 2-Axis Joystick (#27800 at www.parallax.com)
http://www.parallax.com/Store/Accessories/HumanInterfaceDevices/tabid/822/CategoryID/90/List/0/SortField/0/Level/a/ProductID/581/Default.aspx

39- Sound Impact Sensor (#29132 at www.parallax.com)
http://www.parallax.com/Store/Accessories/Sound/tabid/164/CategoryID/38/List/0/SortField/0/Level/a/ProductID/614/Default.aspx

40- 5-Way button (#27801 at www.parallax.com)
http://www.parallax.com/Store/Accessories/HumanInterfaceDevices/tabid/822/CategoryID/90/List/0/SortField/0/Level/a/ProductID/615/Default.aspx

41- Quadrature System (Position Control Kit #27906 at www.parallax.com)
http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/SortField/0/Level/a/ProductID/665/Default.aspx

42- Standard Servomotor (Parallax Futaba Standard Servo #900-00005 at www.parallax.com)
http://www.parallax.com/tabid/768/ProductID/101/Default.aspx

43- SD card reader (micro-SD card adapter #32312 at www.parallax.com)
http://www.parallax.com/Store/Microcontrollers/PropellerAccessories/tabid/786/CategoryID/85/List/0/SortField/0/Level/a/ProductID/597/Default.aspx

44- FSRW26
http://obex.parallax.com/objects/92/

45- H48C_Tri-Axis_Accelerometer.spin

46- Tutorial on how to use the QTI
http://forums.parallaxinc.com/forums/default.aspx?f=6&m=132921

47- XBee (#32405 at www.parallax.com)
http://www.parallax.com/go/XBee

48- XBee USB Adapter Board (#32400 at www.parallax.com)
www.parallax.com/go/XBee

49- XBee SPI Adapter Board (#32402 at www.parallax.com)
www.parallax.com/go/XBee

50- XBee X-CTU Software
www.parallax.com/go/XBee

51- Excellent tutorial on the XBee (#122-32450 at www.parallax.com)
www.parallax.com/go/XBee

52- Easy Bluetooth Module (#30085 at www.parallax.com)
http://www.parallax.com/Store/Accessories/CommunicationRF/tabid/161/CategoryID/36/List/0/SortField/0/Level/a/ProductID/550/Default.aspx

53- D-Link DBT-120 dongle
http://www.amazon.com/D-Link-DBT-120-Wireless-Bluetooth-Adapter/dp/B00006B7DB/ref=sr_1_1?ie=UTF8&qid=1289628417&sr=8-1

54- Parallax Easy Bluetooth documentation
http://www.parallax.com/Portals/0/Downloads/docs/prod/comm/30085-EasyBluetooth-v1.3.pdf

55- Tutorial on the EBT at the Parallax Forums
http://forums.parallax.com/showthread.php?112822

56- 433 MHz RF Transceiver (#27982 at www.parallax.com)
http://www.parallax.com/Store/Accessories/CommunicationRF/tabid/161/CategoryID/36/List/0/SortField/0/Level/a/ProductID/582/Default.aspx

57- RobotBASIC_Networking.Pdf
http://www.robotbasic.org/resources/RobotBASIC_Networking.pdf

58- Spinneret Web Server (#32203 at www.parallax.com)
http://www.parallax.com/Store/Microcontrollers/PropellerDevelopmentBoards/tabid/514/ProductID/710/List/0/Default.aspx?SortField=ProductName,ProductName

59- Boe-Bot (#28832 at www.parallax.com)
http://www.parallax.com/Store/Robots/AllRobots/tabid/128/CategoryID/3/List/0/SortField/0/Level/a/ProductID/296/Default.aspx

60- Photo Resistor (#350-00009 at www.parallax.com)
http://www.parallax.com/Store/Components/Optoelectronics/tabid/152/CategoryID/50/List/0/SortField/0/Level/a/ProductID/175/Default.aspx

61- Photo Diode (#350-00012 at www.parallax.com)
http://www.parallax.com/Store/Components/Optoelectronics/tabid/152/CategoryID/50/List/0/SortField/0/Level/a/ProductID/176/Default.aspx

62- Four books that deal with the RB simulator
http://www.amazon.com/s/ref=nb_sb_noss?url=search-alias%3Dstripbooks&field-keywords=robotbasic

63- Many YouTube video tutorials for RB
http://www.youtube.com/profile?user=john30340#g/u

64- Smart Sensors And Applications
http://www.parallax.com/Portals/0/Downloads/docs/prod/sic/3rdPrintSmartSensors-v1.0.pdf

65- Stamps In Class
http://www.parallax.com/education

66- Extensive tutorial on the USBmicro devices
http://www.robotbasic.org/resources/RobotBASIC_USBmicro_U4x1.pdf

67- USBToSerial (Pasrt#28030)
http://www.parallax.com/Store/Accessories/CablesConverters/tabid/166/CategoryID/40/List/0/SortField/0/Level/a/ProductID/378/Default.aspx

68- Basic_I2C_Driver.Spin
http://obex.parallax.com/objects/26/

69- Enhancing The Pololu 3pi With RobotBASIC
http://www.amazon.com/Enhancing-Pololu-3pi-RobotBASIC-ebook/dp/B0044XUU5E/ref=sr_1_2?ie=UTF8&qid=1295347995&sr=8-2

70- Robot Programmer’s Bonanza
http://www.amazon.com/Robot-Programmers-Bonanza-ebook/dp/B001AW2QKE/ref=sr_1_2?ie=UTF8&m=AG56TWVU5XWC2&s=digital-text&qid=1295484609&sr=8-2

71- Propeller Robot Controller Board (#28230 at www.parallax.com)
http://www.parallax.com/Store/Microcontrollers/PropellerDevelopmentBoards/tabid/514/CategoryID/73/List/0/SortField/0/Level/a/ProductID/584/Default.aspx

72- RobotBASIC_Subroutines.Pdf
http://www.robotbasic.org/resources/RobotBASIC_Subroutines.pdf

Appendix B
Tables & Schematics

Final Protocol Objects Hierarchy Map

The entire system requires 2,565 Longs (11 Kbytes) and 7 cogs. Thus we have 1 spare cog.

Out of the 28 available pins (32 less 2 for programming and 2 are for EEPROM) we are using 26 pins. So we have 2 spare pins. We do use the 2 EEPROM pins to write/read from it

```
───────────────────────────────────────
Parallax Propeller Chip Project Archive
───────────────────────────────────────

 Project :  "Protocol_Main"
    Tool :  Propeller Tool version 1.2.7
            Protocol_Main.spin
              ├──FullDuplexSerial.spin
              ├──SerialMirror.spin
              ├──Basic_I2C_Driver.spin
              ├──Protocol_Others.spin
              │    ├──SerialMirror.spin
              │    ├──Pots_RCTime.spin
              │    └──HMC6352.spin
              ├──Protocol_Reader.spin
              │    └──SerialMirror.spin
              └──Protocol_Motors.spin
                   ├──SerialMirror.spin
                   └──Servo32v7.spin
                        └──Servo32_Ramp_v2.spin
```

Extended Protocol Objects Hierarchy Map

The entire system requires 2,700 Longs (11 Kbytes) and 7 cogs. Thus we have 1 spare cog.

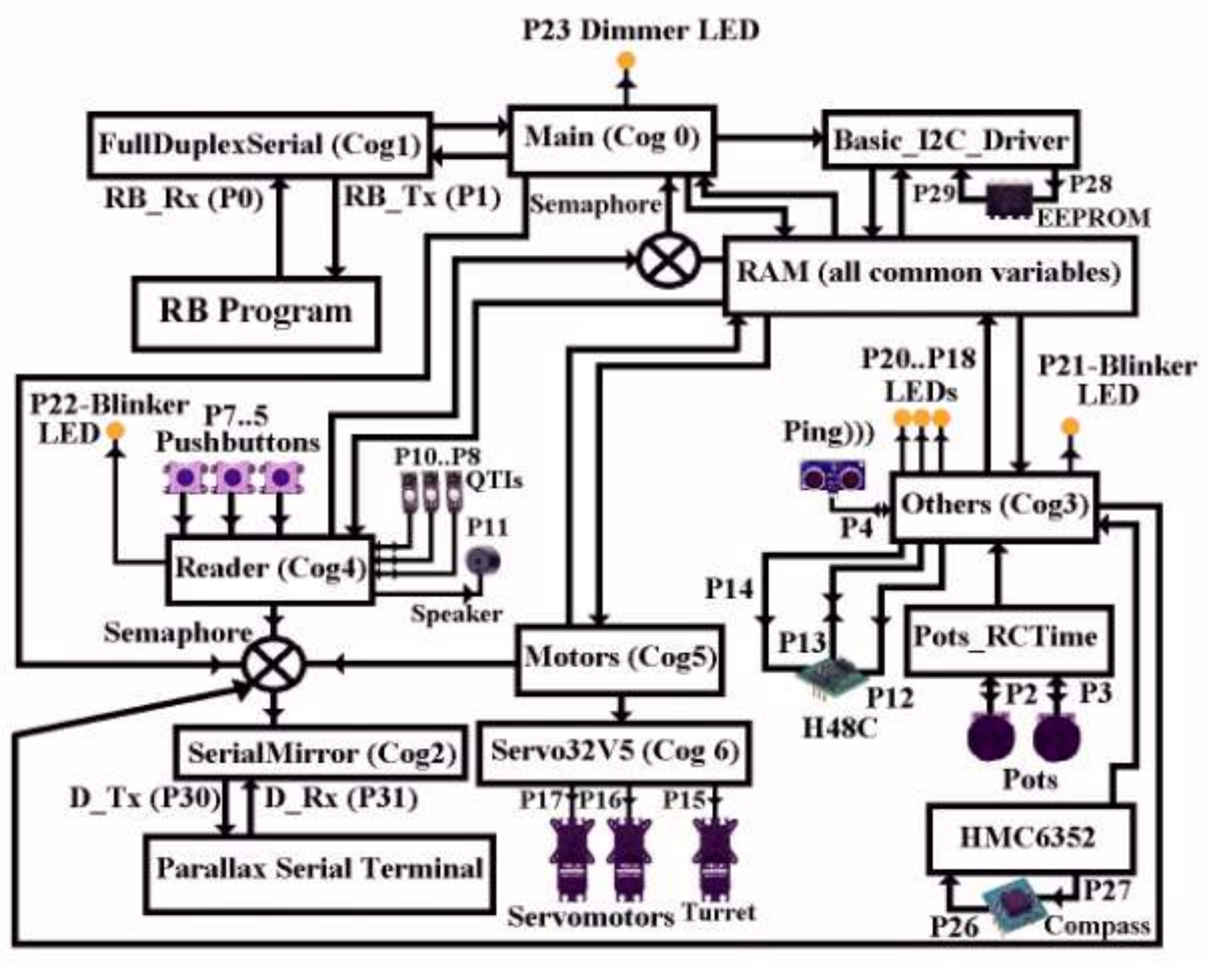
Out of the 28 available pins (32 less 2 for programming and 2 are for EEPROM) we are using 26 pins. So we have 2 spare pins. We do use the 2 EEPROM pins to write/read from it

```
───────────────────────────────────────
Parallax Propeller Chip Project Archive
───────────────────────────────────────

 Project :  "Extended_Protocol_Main"
Archived :  Sunday, January 16, 2011 at 10:52:21 AM
    Tool :  Propeller Tool version 1.2.6

            Extended_Protocol_Main.spin
              ├──FullDuplexSerial.spin
              ├──SerialMirror.spin
              ├──Basic_I2C_Driver.spin
              ├──Extended_Protocol_Others.spin
              │    ├──SerialMirror.spin
              │    ├──Pots_RCTime.spin
              │    └──HMC6352.spin
              ├──Extended_Protocol_Reader.spin
              │    └──SerialMirror.spin
              └──Extended_Protocol_Motors.spin
                   ├──SerialMirror.spin
                   └──Servo32v7.spin
                        └──Servo32_Ramp_v2.spin
```

Figure B.1: System's Conceptual Schematic



Cog numbers are only for reference.

Figure B.2: Propeller Pin Utilization

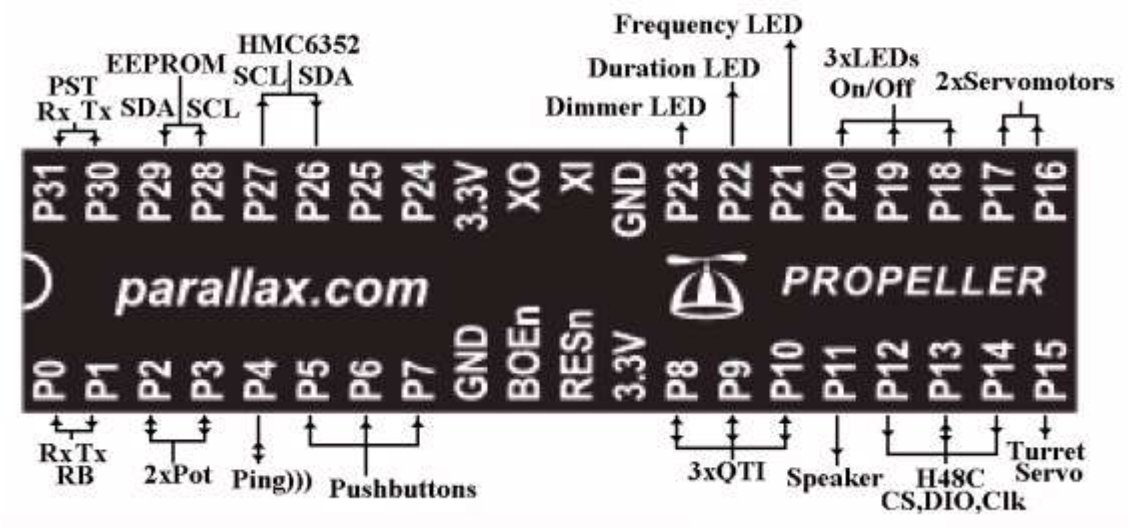


Figure B.3: Hardware Connection Schematics

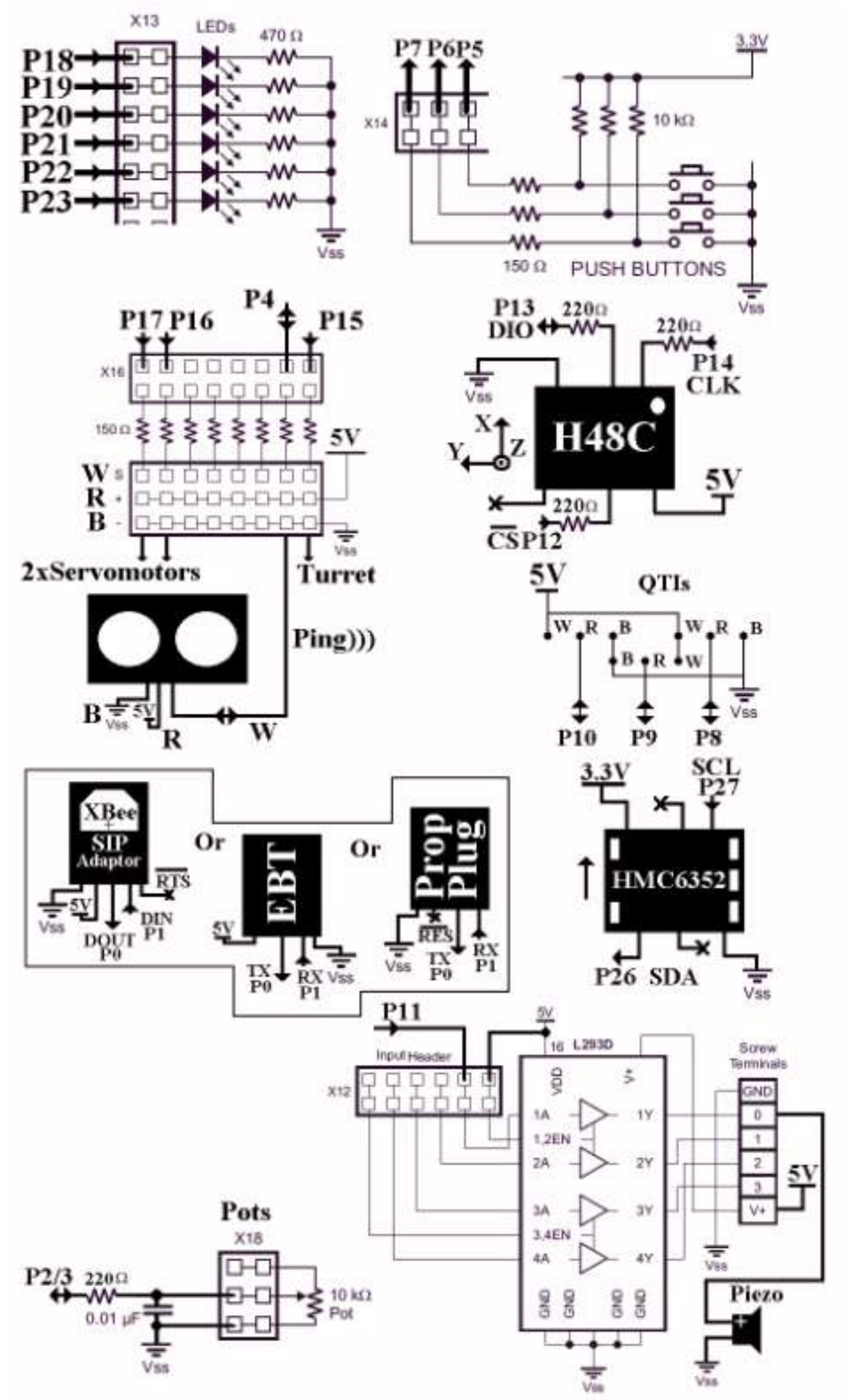


Figure B.4: Photograph of the Final PPDB Hardware Arrangement

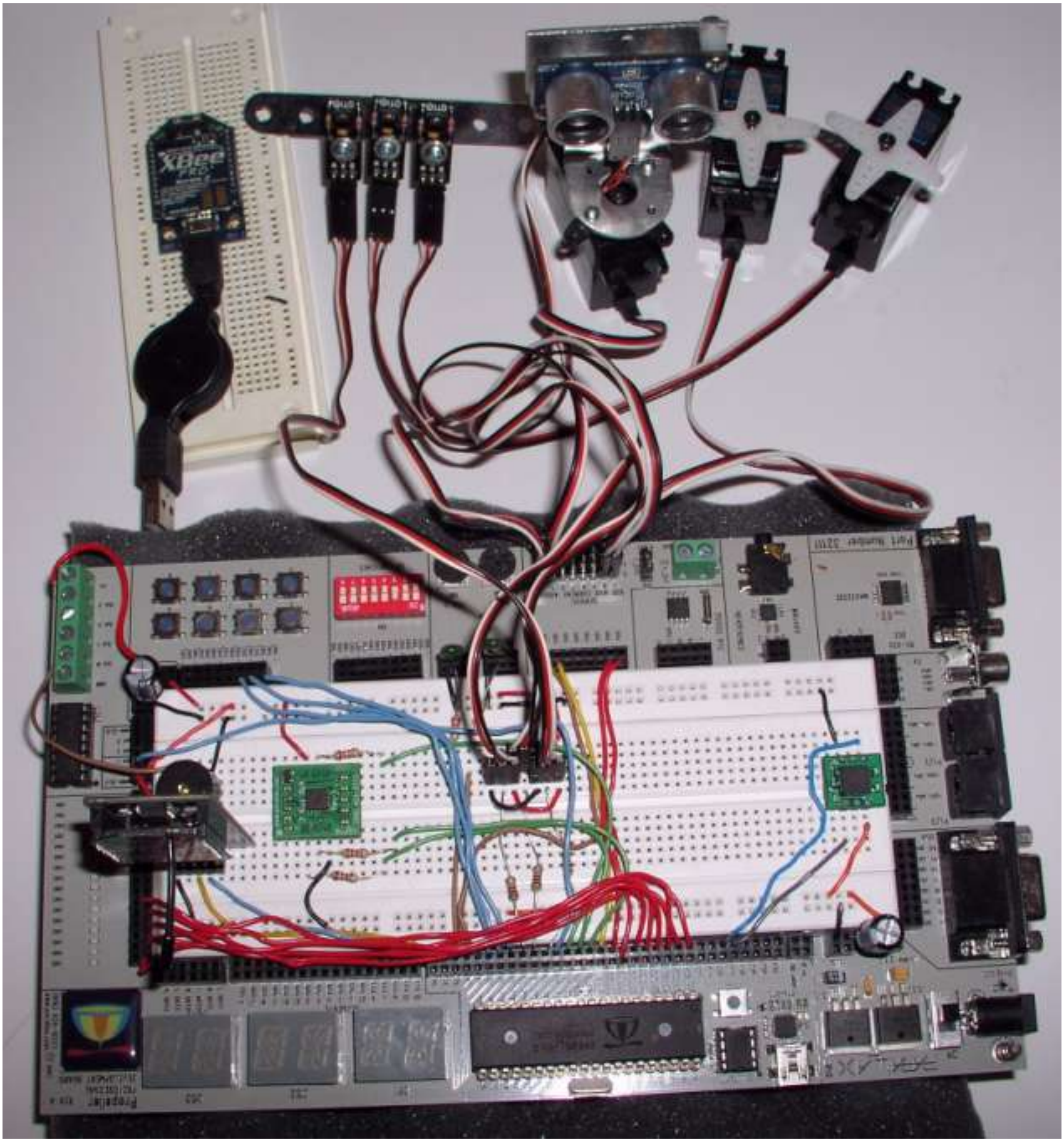


Table B.1: Final Firmware Protocol Command Codes

Command	Code	Parameter	Updates Three Critical Sensors	Data Returned
Stop Motors	0	0	Yes	None
Get System Parameter's value from RAM	4	Parameter# (See Table 8.6)	No	Value of Parameter Little-Endian in the first 4 bytes If success 5th bytes is 0 if failed then $FF
Save or Display System Parameters from RAM or EEPROM	5	0=Save 1=Restore Factory Settings 2=List EEPROM 3=List RAM	Yes	Only when saving. 5th byte is 1 if success or 0 if not
Motors On P17..P16 Forward **Backward**	6 7	Amount	Yes	None
Turn right motor on P16 forward **Backward**	8 9	Number Of Steps 0=off 255= all the time	Yes	None
Turn left motor on P17 forward **Backward**	10 11	Number Of Steps 0=off 255= all the time	Yes	None
Turn Motors on P17..P16 Right **Left**	12 13	Amount	Yes	None
Read the Compass On P27..P26	24	0	Yes	Last two bytes
Check if a compass is available	24	1	Yes	4th byte = 0, 5th byte =1 if available or 0 if not
Calibrate the Compass	24	2=Manual 3=Automatic	Yes No	None
Read the Pots On P3..P2	66	0	No	First 4 bytes
Read the H48C on P14..P12	70	0=Get Axis 1=Get vRef	Yes/No	vRef in 4th and 5th bytes Axis in 5 bytes
Play a note on the Speaker on P11	73	Note# (0-84)	Yes	None
Read the Ping))) on P4 and **Move Turret on P15 Right and Left**	192 193	Angle (0-90)	Yes	Last two bytes
Set P20..P18 LEDs	1	LED States	Yes	None
Set P21 Blink Frequency	2	Hz Value	Yes	None
Reset the Propeller	255	0	No	None
Set P23 LED brightness	200	Level	Yes	None
P22 LED Blink duration	201	Level	Yes	None
Set 2nd byte receive Timeout1	202	N x 10ms	Yes	None
Set operations Timeout2	203	N x 10ms	Yes	None
Set L_Speed	240	Speed	Yes	None
Set T_Speed	244	Speed	Yes	None
Set L_Timeout	241	N X 10 ms	Yes	None
Set T_Timeout	245	N X 10 ms	Yes	None
Set StepTime	242	N X 10 ms	Yes	None
Set TurnTime	243	N X 10 ms	Yes	None

Table B.2: System Parameters Mapping Formulas

When using the command to read back a system parameter the RB program will receive it from the Propeller as it is stored in the RAM, which is a 32-bit Integer. To convert the number into a value from 0 to 255 which is the value sent from an RB program to the Propeller, we will use a factor with which to multiply the value received from the Propeller. The table below shows these factors; F is the clock frequency of the Propeller, which should be 80 MHz for our purposes. In the top part are the values in the **Main** object and in the lower part are the ones in the **Motors** object.

Parameter Description	Command Number	Parameter Order	Multiply Factor
P22 blinker LED Duration	201	0	255/2/F
P21 blinker LED frequency	2	1	1
P23 dimmer LED brightness level	200	2	1
TimeOut1 for receiving second byte	202	3	2/F
TimeOut2 for a command to finish its working	203	4	2/F
L_Speed for speed of motors in linear movement	240	5	1
T_Speed for speed of motors in turning	244	6	1
Step_Time the time for keeping motors on to go one step linearly	242	7	0.1
TurnTime is the time to keep the motors to turn 1 degrees	243	8	0.1
L_TimeOut is the time to keep the motors on after finishing a step in linear movement	241	9	0.1
T_TimeOut is the time to keep the motors on after finishing a turn in turning movement.	245	10	0.1

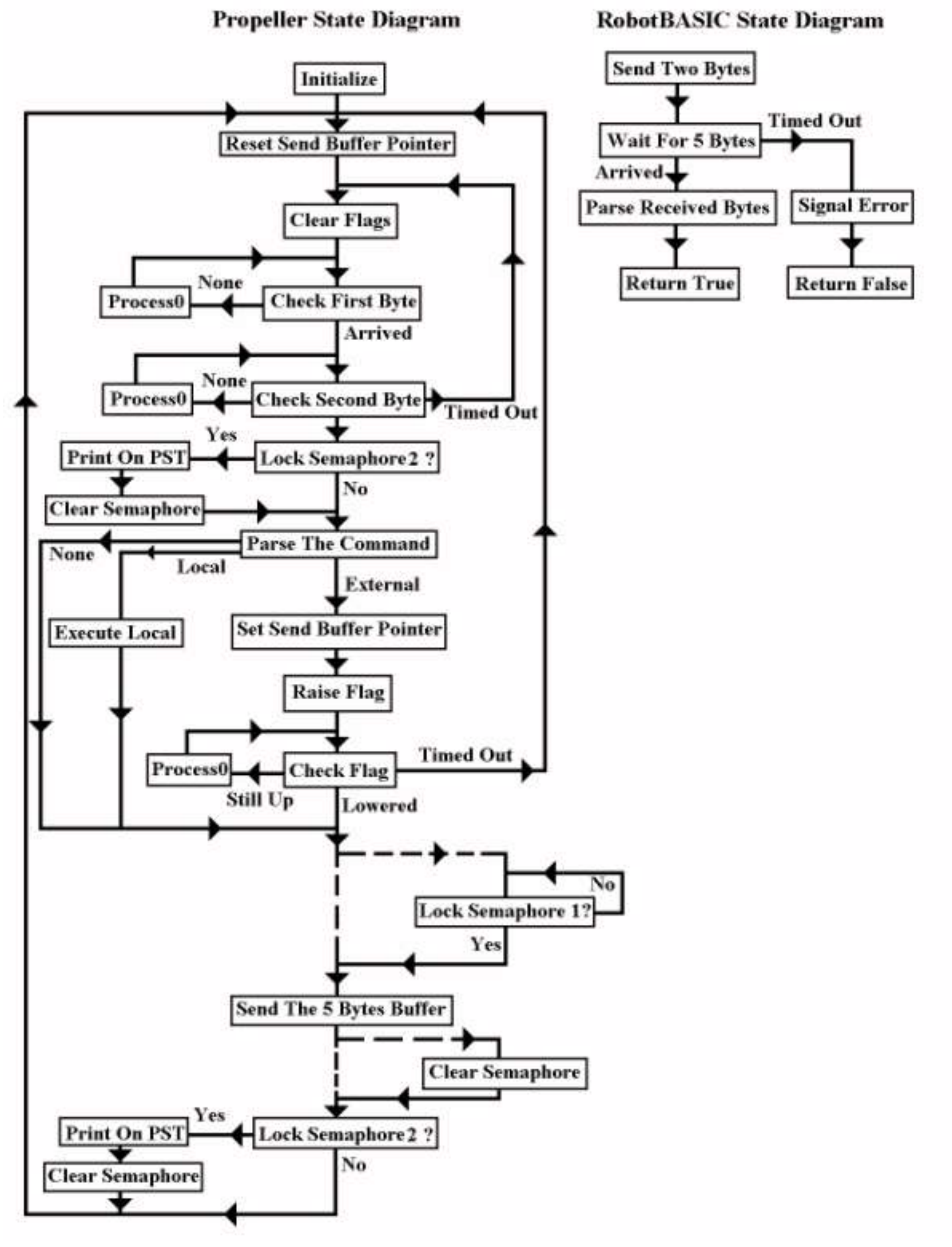
Table B.3: Extended Firmware Command Codes (In addition to Table B.1)

Command	Code	Parameter	Updates Three Critical Sensors	Data Returned
Read the H48C 3-Axis values adjusted for the reference voltage.	71	0 (1-byte)	No	3xLongs (12 bytes). Each is a 4-byte Little-Endian Long.
Get All System Parameter's value from EEPROM or RAM	254	0 = EEPROM 1 = RAM (1-byte)	No	12xLongs, 1st long is count (11). The other 11 longs are the values of the 11 system parameters. Little-Endian 4-byte longs.
P22 LED Blink duration as a duration in Propeller ticks where 80_000_000 is 1 second.	211	Time duration 4-bytes Little-Endian Long	Yes	None
Set 2nd byte receive Timeout1 as a duration in Propeller ticks where 80_000_000 is 1 second	202	Time duration 4-bytes Little-Endian Long	Yes	None
Set operations Timeout2 as a duration in Propeller ticks where 80_000_000 is 1 second	203	Time duration 4-bytes Little-Endian Long	Yes	None
Set L_Timeout in milliseconds	250	Milliseconds 4-bytes Little-Endian Long	Yes	None
Set T_Timeout in milliseconds	253	Milliseconds 4-bytes Little-Endian Long	Yes	None
Set StepTime in milliseconds	251	Milliseconds 4-bytes Little-Endian Long	Yes	None
Set TurnTime in milliseconds	252	Milliseconds 4-bytes Little-Endian Long	Yes	None

Table B.4: RobotBASIC Inbuilt Protocol Command Codes

Command/Function	Code	Parameter	Updates Three Critical Sensors	Data Returned	Error
rLocate ne_X,ne_Y	3	ne_X	Yes	None	None
rForward +ne_Amount	6	ne_Amount	Yes	None	Halts program
-ne_Amount	7	ne_Amount	Yes	None	Halts program
rTurn +ne_Amount	12	ne_Amount	Yes	None	Halts program
-ne_Amount	13	ne_Amount	Yes	None	Halts program
rCompass()	24	0	Yes	Last two bytes	-1
rSpeed ne_Speed	36	ne_Speed	Yes	None	None
rLook({+ne_Angle})	48	ne_Angle	Yes	Last two bytes	-1
({-ne_Angle})	49	ne_Angle	Yes	Last two bytes	-1
rGPS vn_X,vn_Y	66	0	No	First 4 bytes	-1,-1
rBeacon(ne_Color)	96	ne_Color	Yes	Last Two bytes	-1
rChargeLevel()	108	0	Yes	Last two bytes	-1
rPen ne_State	129	ne_State	Yes	None	None
rRange({+ne_Angle})	192	ne_Angle	Yes	Last two bytes	-1
(-ne_Angle)	193	ne_Angle	Yes	Last two bytes	-1
rCommand(ne_Command,ne_Data)	ne_Command	ne_Data	No	String with 5 bytes	Empty buffer

Figure B.5: Protocol State Diagrams


Propeller State Diagram
Initialize
Reset Send Buffer Pointer
Clear Flags
Process0
None
Check First Byte
Arrived
Process0
None
Check Second Byte
Timed Out
Yes
Print On PST
Lock Semaphore2 ?
No
Clear Semaphore
Parse The Command
None
Local
External
Execute Local
Set Send Buffer Pointer
Raise Flag
Process0
Check Flag
Timed Out
Still Up
Lowered
No
Lock Semaphore 1?
Yes
Send The 5 Bytes Buffer
Clear Semaphore
Yes
Print On PST
Lock Semaphore2 ?
No
Clear Semaphore
RobotBASIC State Diagram
Send Two Bytes
Wait For 5 Bytes
Timed Out
Arrived
Parse Received Bytes
Signal Error
Return True
Return False