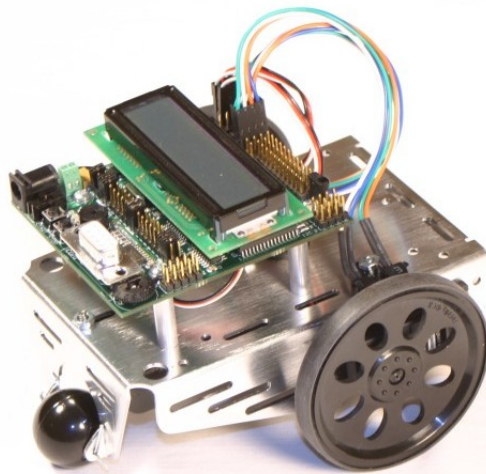




Beginning Robotics Course Outline



www.ridgesoft.com

Table of Contents

Lesson 1: Robot Hardware and Assembly	1
Lesson 2: Programming and Testing the IntelliBrain-Bot	2
Lesson 3: Maneuvering	3
Lesson 4: Introduction to Sensing	5
Lesson 5: Wheel Position Encoding.....	6
Lesson 6: Tracking Position using Odometry and Dead Reckoning	7
Lesson 7: Navigation.....	9
Lesson 8: Object Sensing	10
Lesson 9: Behavior-based Control	12
Lesson 10: Line Following.....	13
Lesson 11: Remote Control.....	15
Lesson 12: Sonar Range Sensing	16
Lesson 13: Dancing	17
Lesson 14: Creating a User Interface	18
Lesson 15: Vision Sensing	19

Lesson 1: Robot Hardware and Assembly

Learning Objectives

- Become familiar with robot hardware
 - Mechanics
 - Chassis
 - Motors
 - Wheels & Tires
 - Electronics
 - IntelliBrain robotics controller
 - LCD display
 - Sensors
 - Power Source - batteries
 - Serial cable

Resources

- IntelliBrain-Bot Kit
- *IntelliBrain-Bot Assembly Guide*

Activities

Inventory

1. Identify and inventory parts according to the *IntelliBrain-Bot Assembly Guide*

Assembly

1. Assemble the robot according to the *IntelliBrain-Bot Assembly Guide*

Verify Assembly

1. Visually inspect the mechanical assembly, comparing it with photos in the *IntelliBrain-Bot Assembly Guide*
2. Visually inspect electronic connections according to the *IntelliBrain-Bot Assembly Guide*

Lesson 2: Programming and Testing the IntelliBrain-Bot

Learning Objectives

- Become familiar with building and loading programs
- Learn basic robot testing and debugging techniques

Resources

- Assembled IntelliBrain-Bot
- IntelliBrainBotDemo program
- *Creating Your First IntelliBrain Program* tutorial
- *RoboJDE User Guide*
- *IntelliBrain User Guide*

Activities

Introduction to the RoboJDE Development Environment

1. Launching RoboJDE
2. Selecting and opening a project
3. Viewing the program source
4. Compiling the program

Powering the Robot On and Off

1. Installing batteries
2. Switching power on
3. Verifying the RoboJDE virtual machine starts
4. Switching power off

Downloading a Program to the Robot

1. Connecting the serial cable
2. Verifying host connectivity
3. Compiling and downloading the IntelliBrainBotDemo program

Testing the Robot Using the IntelliBrainBotDemo Program

1. Running the program
2. Viewing program output
3. Testing sensors
4. Debugging sensor problems
5. Verifying robot maneuvers

Lesson 3: Maneuvering

Learning Objectives

- Understand how to make the robot maneuver
- Understand how to create simple programs
- Understand how to command motors
- Understand that robots are imperfect and subject to forces outside of their control
- Learn how to use RoboJDE Application Programming Interface (API) documentation

Resources

- Assembled IntelliBrain-Bot
- Timed maneuvering example programs
 - IntelliBrainBotTimedForward program
 - IntelliBrainBotTimedRotate program
 - IntelliBrainBotTimedSquare program
 - IntelliBrainBotTimedOctagon program
- *Programming Your Robot to Perform Basic Maneuvers* tutorial
- *RoboJDE API Documentation* (Javadoc)
- *IntelliBrain API Summary* (1 page graphic)
- *IntelliBrain User Guide*

Activities

Programming the Robot to Go Forward

1. Understanding IntelliBrainBotTimedForward program and how to use the *RoboJDE API Documentation*
 - a. Investigate each line of the program, referring to *RoboJDE API Documentation* for a description of each library class and method used
 - b. Review of servo and motor classes in detail
2. Downloading and running IntelliBrainBotTimedForward program

Creating a New Program to Program the Robot to Rotate in Place

1. Creating a new project
2. Copying from a previous example
3. Modifying the program to make the robot rotate in place
4. Compiling the program
5. Debugging compilation errors
6. Downloading the program
7. Testing the program
8. Debugging the program

Timing Movements to Move the Robot in a Square

1. Tune the timing such that the robot rotates approximately 90 degrees

2. Modify the program such that the robot drives forward for a fixed period of time before rotating
3. Add a loop around the forward and rotate operations to do them 4 times, causing the robot to move in a square
4. Tuning the timing

Learning Why the Robot's Movement isn't Very Predictable

1. Compare the timing for different robots and discuss why they are different
2. Run the program with old batteries and new batteries and observe the difference in performance
3. Discuss findings and how factors outside of the robot's control (friction, battery charge) affect its performance
4. Discuss approaches to enable the robot to cope with this such as adding sensors for feedback (compass, encoders, accelerometers, GPS, etc).

Experimenting with Maneuvering Other Patterns

1. Program the robot to move in a more complex pattern

Lesson 4: Introduction to Sensing

Learning Objectives

- Understand theory of operation of infrared photo-reflector sensors
- Understand how to sample analog inputs
- Interpreting sampled data

Resources

- Assembled IntelliBrain-Bot
- IntelliBrainAnalog example program
- *Creating Shaft Encoders for Wheel Position Sensing* tutorial
- *RoboJDE API Documentation*
- *IntelliBrain API Summary* (1 page graphic)
- *IntelliBrain User Guide*

Activities

Theory of Operation of Infrared Photo-reflector Sensors

1. See Theory of Operation section of *Creating Shaft Encoders for Wheel Position Sensing* tutorial

Sampling Sensors from Your Program

1. Open the IntelliBrainAnalog example project
2. Review program and API documentation
3. Download and run program
4. Record data from sensor while you manually rotate the robot's left wheel
5. Compare sensor readings from different robots

Modify Program to Sample the Right Wheel

1. Create a new program to sample and display sensor readings from both sensors

Lesson 5: Wheel Position Encoding

Learning Objectives

- Understanding the theory of operation of shaft encoders
- Understanding how data from a simple sensor can be used to provide higher level information
- Using the AnalogShaftEncoder class from the RoboJDE class library

Resources

- Assembled IntelliBrain-Bot
- IntelliBrainTimedForward and IntelliBrainBotNavigateForward example programs
- *Creating Shaft Encoders for Wheel Position Sensing* tutorial
- *RoboJDE API Documentation*

Activities

Theory of Operation of Shaft Encoders

1. See *Creating Shaft Encoders for Wheel Position Sensing* tutorial

Using the AnalogShaftEncoder Class in the RoboJDE Class Library

1. See *Creating Shaft Encoders for Wheel Position Sensing* tutorial
2. See IntelliBrainNavigateForward example program
3. See *RoboJDE API Documentation* for AnalogShaftEncoder class

Create a Program to Monitor and Display Encoder Counts

1. Start by copying from the IntelliBrainTimedForward program
2. Add AnalogShaftEncoder objects for each wheel
3. Print the encoder count values to the screen once per second

Modify the Program to Navigate a Different Pattern

1. Modify program to navigate a different pattern
2. Observe the encoder output

Lesson 6: Tracking Position using Odometry and Dead Reckoning

Learning Objectives

- Understand dead reckoning
- Using geometry and trigonometry to track movements
- Summing small movements (integrating) to track position
- Using the `OdometricLocalizer` class
- Understanding systematic error
- Approaches to reducing and avoiding error

Resources

- Assembled IntelliBrain-Bot
- `IntelliBrainBotNavigateSquare` example program
- Nubotics WheelWatch WW-01 wheel encoder sensors (2) (optional)
- `IntelliBrainWW01` example program
- Devantech CMPS03 compass sensor (optional)
- `IntelliBrainCMPS03` and `IntelliBrainCMPS03I2C` example programs
- *Enabling Your Robot to Keep Track of its Position* tutorial
- *RoboJDE API Documentation*

Activities

Calculating Distance Traveled and Heading Changes

1. See *Enabling Your Robot to Keep Track of its Position* tutorial

Calculating Movements in Cartesian Coordinates

1. See *Enabling Your Robot to Keep Track of its Position* tutorial

Summing (Integrating) Movements to Track Position

1. See *Enabling Your Robot to Keep Track of its Position* tutorial

Using the `OdometricLocalizer` Class to Track and Display Position

1. See `IntelliBrainBotNavigateSquare` for example of using `OdometricLocalizer` class
2. Start with the program created in the previous lesson
3. Add an `OdometricLocalizer` object to track position
4. Print the position and heading to the display once per second
5. Compare manual position measurements with the displayed position over several test runs

Investigate Sources of Error

1. See *Enabling Your Robot to Keep Track of its Position* tutorial

2. Investigate the effects of doubling, quadrupling, halving and quartering the period of the OdometricLocalizer and its effect on error and CPU utilization
3. Calculate the precision of heading measurements and the effect of using higher resolution encoders
4. (Optional) Experiment with higher resolution wheel encoders (for example, Nubotics WheelWatcher WW-01) sensors to reduce error by using higher resolution encoders
 - a. See IntelliBrainWW01 example program
5. (Optional) Experiment with adding a compass sensor (for example Devantech CMPS03) to improve heading accuracy and reduce position estimation error
 - a. See IntelliBrainCMPS03 and IntelliBrainCMPS03I2C example programs

Lesson 7: Navigation

Learning Objectives

- Using dead reckoning to navigate
- Using proportional control to steer
- Using RoboJDE's navigation classes

Resources

- Assembled IntelliBrain-Bot
- *Programming Your Robot to Navigate* tutorial
- Navigation examples: IntelliBrainBotNavigateForward, IntelliBrainBotRotate180, IntelliBrainBotNavigateSquare
- Nubotics WheelWatch WW-01 wheel encoder sensors (2) (optional)
- IntelliBrainWW01 example program
- *RoboJDE API Documentation*

Activities

Using Trigonometry to Determine the Heading to Next Waypoint

1. See *Programming Your Robot to Navigate* tutorial

Using Proportional Control to Steer to the Next Waypoint

1. See *Programming Your Robot to Navigate* tutorial

Determining When Robot has Reached its Next Waypoint

1. See *Programming Your Robot to Navigate* tutorial

Overview of RoboJDE's Navigation Classes

1. See Navigation and Localization Class Diagram in *Programming Your Robot to Navigate* tutorial
2. See *RoboJDE API Documentation*: Navigator, DifferentialDriveNavigator

Navigating simple patterns

1. See examples: IntelliBrainBotNavigateForward, IntelliBrainBotRotate180, IntelliBrainBotNavigateSquare
2. (Optional) Experiment with WheelWatcher WW-01 sensors for improved accuracy
 - a. See IntelliBrainWW01 example

Lesson 8: Object Sensing

Learning Objectives

- Using Sharp GP2D12 infrared range sensors

Resources

- Assembled IntelliBrain-Bot
- Range Sensor Kit (**required add-on**)
- IntelliBrainBotDemo program
- IntelliBrainAnalog example program
- IntelliBrainGP2D12 example program
- Sharp GP2D12 datasheet (available online from many sources such as www.digikey.com)
- *RoboJDE API Documentation*

Activities

Assemble and Mount Range Sensors on IntelliBrain-Bot

1. Mount sensors on robot
2. Connect left and right range sensors to analog ports 1 and 2, respectively

Create a Program to Display Raw Sensor Readings

1. See IntelliBrainAnalog example program for example of sampling and displaying one sensor input
2. Create a program to sample and display both sensor inputs

Test the Range Sensors

1. Using the program created above, verify the range sensors perform as expected
2. Alternatively, use the IntelliBrainBotDemo program to verify the range sensors perform as expected

Experiment with a Sensor to Learn about its Characteristics

1. Hold the robot with the sensor facing the wall, record samples taken in 1 inch increments
2. Graph the samples with distance on the x-axis and sampled value on the y-axis
3. Compare your graph with graphs in the GP2D12 datasheet
4. Discuss behavior of the sensor
 - a. What happens when you move the robot toward the wall?
 - b. Does the behavior change when the sensor is very close to the wall?
 - c. What happens when you move it away from the wall?
 - d. What's its effective range?
 - e. Use your graph to convert from a sensor measurement to distance. Note how there are two possible distances for some sensor readings.

Using the SharpGP2D12 Class from RoboJDE Class Library

1. See SharpGP2D12 class documentation in *RoboJDE API Documentation*
2. Experiment measuring distances to objects using the IntelliBrainGP2D12 example program

Lesson 9: Behavior-based Control

Learning Objectives

- Understand behavior-based control
- Using behavior classes from the RoboJDE class library
- Creating behaviors

Resources

- Assembled IntelliBrain-Bot with Range Sensor Kit add-on (**Range Sensor Kit add-on is required**)
- IntelliBrainBotAvoidObstacles example program
- *RoboJDE API Documentation*

Activities

Discuss Behavior-based Control

1. See API documentation for BehaviorArbiter, Behavior, Behavior2, BehaviorEvent, BehaviorListener, AbstractBehavior2, AvoidBehavior, GoToBehavior and StopBehavior. The BehaviorArbiter class documentation discusses using behaviors.
2. See IntelliBrainBotAvoidObstacles example program

Experiment with IntelliBrainBotAvoidObstacles Example Program

1. Place an object in the robot's path and observe the robot's behavior
2. Modify the program such that the robot attempts to navigate a different pattern, such as a triangle, avoiding obstacles in its path

Lesson 10: Line Following

Learning Objectives

- Understand how a robot can follow a line
- Understand organizing a program as a state machine

Resources

- Assembled IntelliBrain-Bot
- Line Sensor Kit (**required add-on**)
- White surface
- 1" black electrical tape
- IntelliBrainBotLineFollower example program
- *RoboJDE API Documentation*

Activities

Assemble and mount line sensors on IntelliBrain-Bot

1. Mount line sensors on the robot less than 1 inch apart
2. Connect left and right range sensors to analog ports 6 and 7, respectively

Create a Program to Display Raw Sensor Readings

1. See IntelliBrainAnalog example program for example of sampling and displaying one sensor input
2. Create a program to sample and display both sensor inputs

Experiment with Line Sensors to Learn About Their Characteristics

1. Place a piece of black electrical tape on a white surface
2. Observe and record the sensor readings while holding them over the white surface
3. Observe and record the sensor readings while holding them over the tape
4. Hold the sensor at varying distances from white and black surfaces and observe how the readings are affected by distance
5. Move a sensor over the edge of a line at different distances and observe how its sensitivity to the line is affected by its distance from the line

Discuss State Machine Programming

1. See IntelliBrainBotLineFollower program
2. List and discuss the states for line following with two sensors. States are:
 - a. 2 left: 2 sensors left of line
 - b. 1 left: 1 sensor left of line
 - c. centered: both sensors over line
 - d. 1 right: 1 sensor right of line
 - e. 2 right: 2 sensors right of line
 - f. unknown: both sensors over white
3. Draw a state diagram

- a. Draw each of 6 states as a bubble
- b. Draw all possible state transitions as lines with arrows between the bubbles and label with the events which cause the transitions
 - i. any -> centered: both sensors over line
 - ii. any -> 1 left: left sensor over white, right sensor over line
 - iii. any -> 1 right: right sensor over white, right sensor over line
 - iv. 1 left -> 2 left: both sensors over white
 - v. 1 right -> 2 right: both sensors over white
 - vi. centered -> unknown: both sensors over white
- c. Note how the robot should respond in each state
 - i. centered: drive straight ahead
 - ii. 1 left: drive slightly right
 - iii. 2 left: drive hard right
 - iv. 1 right: drive slightly left
 - v. 2 right: drive hard left
 - vi. unknown: stop

Develop a Line Following Program

1. Working only from the state diagram, develop a program to:
 - a. poll the sensors
 - b. determine the new state
 - c. power the wheels according to the new state

Experiment with More and Fewer Sensors

1. Write a program using one sensor and 4 states
 - a. Follow the edge of the line with the following states:
 - i. over edge
 - ii. over line
 - iii. over white
 - b. Compare the performance of this program to the one with two sensors
2. Write a program using 3 or 4 sensors
 - a. Is the robot able to follow the line more accurately?

Lesson 11: Remote Control

Learning Objectives

- Understand how an infrared TV remote works
- Understand how to control a robot remotely using a TV remote control

Resources

- Assembled IntelliBrain-Bot
- Universal remote control with Sony remote control compatibility (**required**)
- <http://electronics.howstuffworks.com/remote-control2.htm>
- IntelliBrainBotRemoteControl example program
- *RoboJDE API Documentation*

Activities

How TV Remote Controls Work

1. See <http://electronics.howstuffworks.com/remote-control2.htm>

Discuss Using Remote Control Classes

1. See IrRemote, IrReceiver and SonyIrRemote classes in *RoboJDE API Documentation*

Create a Program to Receive Data from a Remote Control

1. See IntelliBrainBotRemoteControl example program
2. Write a program to display the received code each time a button is pressed on the remote control

Program the IntelliBrainBot to Perform “Tricks”

1. See IntelliBrainBotRemoteControl
2. Program the robot to take different actions based on the command it receives from the remote control. For example:
 - a. buzz the buzzer
 - b. blink an LED
 - c. play a tune
 - d. print a message
 - e. perform a maneuver

Lesson 12: Sonar Range Sensing

Learning Objectives

- Understand sonar sensing
- Using sonar range finding classes from RoboJDE class library

Resources

- Assembled IntelliBrain-Bot
- Parallax Ping sonar range sensor and mounting hardware or Devantech SRF04/SRF05 sonar range finder
- Wire and connectors
- IntelliBrainPing or IntelliBrainSRF04 example program
- IntelliBrainBotSonarFollower example program
- *Sonar Made Simple* tutorial
- *RoboJDE API Documentation*

Activities

Theory of Operation of Sonar Sensors

1. See Theory of Operation section of *Sonar Made Simple* tutorial

Using Sonar Sensing to Measure Distance

1. Open the IntelliBrainPing or IntelliBrainSRF04 example program
2. Review program and API documentation
3. Download and run program
4. Experiment using the sensor to measure distances to various objects noting its effective range and accuracy

Using Sonar to Create a “Tractor Beam”

1. Mount the sensor on the IntelliBrain-Bot
2. Download and run the IntelliBrainBotSonarFollower program
3. Place hand six inches in front of sonar sensor then move your hand slowly in either direction to “push” or “pull” the robot using the sonar tractor beam

Lesson 13: Dancing

Learning Objectives

- Understanding how to make the robot move
- Using Random class for random number generation
- Using the *RoboJDE API Documentation*

Resources

- Assembled IntelliBrain-Bot
- IntelliBrainBotRandomDance example program
- *RoboJDE API Documentation*

Activities

Understanding the Random Dance Program

1. Open the IntelliBrainBotRandomDance example program
2. Review the program and API documentation
3. Download and run the program

Create Your Own Dance

1. Develop you own program to perform a dance
2. Create methods to perform individual steps which move the robot in simple ways
3. Combine the steps in various sequences to make the robot perform sophisticated dances

Lesson 14: Creating a User Interface

Learning Objectives

- Understanding how to create a simple user interface to a robot
- Understanding that a simple user interface is extremely valuable for testing and debugging a robot
- Interfacing to the LCD display, push buttons and thumbwheel on the IntelliBrain robotics controller
- Creating reusable software components

Resources

- Assembled IntelliBrain-Bot
- *Creating a User Interface for Your Robot* tutorial
- IntelliBrainBotDemo program
- *RoboJDE API Documentation*

Activities

Develop a User Interface

1. Develop classes from scratch according to the *Creating a User Interface for Your Robot* tutorial or use pre-built classes from the RoboJDE class library as in the IntelliBrainBotDemo program

Lesson 15: Vision Sensing

Learning Objectives

- Understand theory of operation of the CMUcam vision sensor
- Using the CMUcam classes in the RoboJDE class library
- Tracking colored objects using the CMUcam

Resources

- Assembled IntelliBrain-Bot
- CMUcam vision sensor (available from Seattle Robotics)
- CMUcam/ Mounting Cabling Kit (add-on kit)
- IntelliBrainCMUcam example program
- IntelliBrainBotColorTracker example program
- *Vision Sensing with the CMUcam* tutorial
- *CMUcam User Guide*
- *RoboJDE API Documentation*

Activities

Theory of Operation of CMUcam Vision Sensor

1. See CMUcam User Guide

Using the CMUcam

1. See *Vision Sensing with the CMUcam* tutorial
2. See the IntelliBrainCMUcam example program
3. See CMUcam class documentation in *RoboJDE API Documentation*

Tracking Colorful Objects with the CMUcam

1. See IntelliBrainBotColorTracker example program