

# CS179M: Project in Computer Science: AI (Fall 2005)

October 5, 2005

## **Lab Information:**

- Time: 6:10 - 9:00pm, Wednesdays
- Location: Engineering Building II Room 226
- TA: Teddy Yap, Jr.
- Email: tyap@cs.ucr.edu
- Office Hours and Location: Mondays 2:00 - 4:00pm, Wednesdays 10:00 - 12:00nn, or by appointment, Engineering Building II Room 368

## **Important Safety Information:**

- Read the installation and operations instructions before using the equipment.
- Avoid using power extension cords.
- To prevent fire or shock hazards, do not expose the equipment to rain or moisture.
- Refrain from opening the unit or any of its accessories.
- Keep wheels away from long hair or fur.
- Never access the interior of the robot with charger attached or batteries inserted.

## **Some Do's and Don't's:**

- Do not drop the robot, run it off the ledge, or operate it in an irresponsible manner.
- Do not overload the robot by adding extra payload.
- Do not get the robot wet.
- Do not continue to run the robot with something wound around its wheels or axles.
- Do not open the robot with charger attached or batteries inserted.
- Always take good care of the robot and its accessories.
- Do not leave the robot and its accessories unattended at any time.
- Always use a set of three batteries of the same color every time you want to operate the robot.

## Equipment:

- Hardware:
  - 1 ActivMedia Robotics' P3DX mobile robot with attached camera (though we won't use the camera in this lab)
  - 2 Sets of colored-labelled batteries (red and green) (for a total of six batteries)
  - 1 Battery charger
  - 1 Serial cable for external connection
  - 1 IBM X32 notebook to be used as the control PC
  - 1 IBM X32 notebook docking station
  - 1 IBM X32 notebook A/C adapter
  - Lab computers (in Engineering Building II Room 226)
  - 1 Suction cup tool to grab each battery out of its bay
- Software:
  - Access to lab computers with Linux (in Engineering Building II Room 226)
  - Access to the IBM X32 notebook with Linux
  - ActivMedia Robotics Interface for Applications (ARIA)
  - Mapper3Basic - a software for creating a map for the robot's environment
  - MobileSim - a software for simulating an ARIA client program and its map or environment
  - ARIA Robotics libraries
  - Any text editor (e.g. emacs) - for writing ARIA client programs
- Others:
  - 1 Pioneer 3 Operations Manual

## Getting Started:

1. Make sure the robot is turned off.
2. Open the rear door to the robot's compartment.
3. Slide in three same-colored batteries. **Note:** Insert the batteries according to their correct orientation.
4. Close the rear door properly.
5. Turn on the power.
6. Place the IBM X32 notebook on the robot's deck and connect it to the robot using the serial cable provided.
7. Log-in to your account in the IBM X32 notebook.
8. Execute the provided compiled **demo** program. Start it with: `% ./demo`. **Note:** By default, the robot is in teleoperation mode.

9. In the teleoperation mode, press  $\uparrow$  to move the robot forward,  $\downarrow$  to reverse,  $\leftarrow$  to turn left,  $\rightarrow$  to turn right, and **SPACE** to stop. Pressing the arrow keys longer will increase or decrease the translational velocity ( $\uparrow$  or  $\downarrow$ ) or rotational velocity ( $\leftarrow$  or  $\rightarrow$ ).
10. When you are finished, press the **ESC** key to disconnect the ARIA client program from the server robot and exit the ARIA demonstration program. The robot should disengage its drive motors and stop moving, and its sonar should stop firing. You may now slide the robot's **Main Power** switch to **OFF**.

**Note:** Balance the batteries in the robot. It is recommended that you use three batteries to operate the robot. Otherwise, a single battery should be mounted in the center, or two batteries inserted on each side of the battery container.

**Some Notes on the Batteries and Recharging:** The User Control Panel has a bi-color LED labelled **BATTERY** that visually indicates current battery voltage. From approximately 12.5 volts and above, the LED glows bright green. The LED turns progressively orange and then red as the voltage drops to approximately 11.5 volts.

Aurally, the User Control Panel's buzzer, if active, will sound a repetitive alarm if the battery voltage drops consistently below the **FLASH LowBattery** (11.5 VDC, by default) level. If the battery voltage drops below the **FLASH ShutdownVolts** (11 VDC, by default) the controller automatically shuts down a client connection and notifies the computer, via the **HOST RI** (ring indicator) pin, to shut down and thereby prevent data loss or system corruption due to low batteries.

Typical battery recharge time using the recommended accessory (800 mA) charger varies according to the discharge state; it is roughly equal to three hours per volt per battery. **Note:** Do not overcharge the batteries.

### Some How To's:

- Create maps:
  - Run `Mapper3Basic` with: `% Mapper3Basic` and start building your own map.
  - Be sure to include a home point in your map.
  - Save your map with a `.map` as file extension.
- Run the simulator:
  - Run `MobileSim` with: `% MobileSim`.
  - Select the robot model (e.g. `p3dx` for this lab).
  - Select the map you want to load or select no map. **Note:** Once the map is loaded, `MobileSim` will listen and wait for a client program to control and command the robot.
  - Execute your compiled ARIA client program (e.g. `demo`).
- Create a simple program using ARIA:
  - Open a text editor (e.g. `emacs`) and create an empty file. Type in the statements under **Sample Program 1** and save the file under the name `ShyRobot.cpp`.
  - To compile:

```
% g++ -o ShyRobot ShyRobot.cpp -I/usr/csshare/include
-L/usr/csshare/lib -lAria -lpthread -ldl
```

– To execute: % LD\_LIBRARY\_PATH=/usr/csshare/lib ./ShyRobot

**Note:** It is highly recommended that you test your ARIA client program using MobileSim first before actually deploying it to the IBM X32 notebook and testing it with the actual robot.

**Note:** To deploy your program to the IBM X32 notebook and compile and execute on the robot, use `scp` to transfer files, for example:

```
scp <username>@hill.cs.ucr.edu:<directory path>/<file> .
```

**Note:** It is recommended to set the following environment variables:

- LD\_LIBRARY\_PATH=/usr/csshare/lib
- LD\_CONFIG\_PATH=/usr/csshare/lib

#### Other Useful Information:

- The ActivMedia Robotics website: [www.activmedia.com](http://www.activmedia.com)
- The ARIA Reference Manual: </usr/csshare/pkgs/Aria-2.3.3/doc/Aria/index.html>

#### Sample Program 1:

```
/*
   This program implements a very simple ARIA program.
   This program implements a shy robot.
*/

// Don't forget to include Aria.h since we will be using it to create our
// ARIA client software.
#include "Aria/Aria.h"

using namespace std;

// Defines a new action called ShyAway, a subclass of ArAction.
class ShyAway : public ArAction {

public:

   // The constructor.
   ShyAway();
```

```

// The destructor.
virtual ~ShyAway();

// The action to be fired.
virtual ArActionDesired *fire(ArActionDesired currentDesired);

// Sets the robot.
virtual void setRobot(ArRobot *robot);

private:

// A pointer to the robot.
ArRobot *robot;

// The desired action.
ArActionDesired control;

};

// The construction definition.
ShyAway::ShyAway() : ArAction("Shy Away", "A Simple Behavior") {

    robot = NULL;

}

// The destructor definition.
ShyAway::~ShyAway() {

}

// The setRobot method definition.
void ShyAway::setRobot(ArRobot *r) {

    robot = r;

}

// This method implements the desired shy away behavior.
ArActionDesired *ShyAway::fire(ArActionDesired currentDesired) {

    double distance;
    double tolerableDistance = 1000;
    double minimumDistance = HUGE_VAL;
    int sonarIndex = -1;
    double velocity = 200;

// Resets the control.
control.reset();

// Poll all sonar sensors.

```

```

for (int i = 0; i < 15; i++) {

    distance = robot->getSonarReading(i)->getRange();

    if (distance < tolerableDistance) {

        if (distance < minimumDistance) {

            minimumDistance = distance;
            sonarIndex = i;

        }

    }

}

// Don't move if there is no close object nearby.
if (sonarIndex == -1) {

    velocity = 0.0;

}

// Back away if there is something close in front.
else if (0 <= sonarIndex && sonarIndex <= 7) {

    velocity *= -1;

}

control.setVel(velocity);
control.setRotVel(0.0);
control.setMaxVel(500.0);
control.setMaxNegVel(-500.0);
control.setMaxRotVel(500.0);

return &control;

}

// This is the main program heading. Not too exciting though.
int main(int argc, char** argv) {

    // Creates a simple connector object that simplifies connecting to the robot.
    // We need this in order to connect to MobileSim or to the actual physical robot
    // (if there is no MobileSim running).
    ArSimpleConnector simpleConnector(&argc, argv);

    // Creates a robot object.
    ArRobot robot;

```

```

// Creates a key handler so we can do our key handling.
ArKeyHandler keyHandler;

// Creates sonar device. Sonar must be added to the robot later.
ArSonarDevice sonarDevice;

// Creates a ShyAway action object.
ShyAway shyAway;

// Mandatory initialization step.
Aria::init();

// Lets ARIA know we have a key handler.
Aria::setKeyHandler(&keyHandler);

// Attaches the key handler to the robot.
robot.attachKeyHandler(&keyHandler);

printf("You may now press the ESC key anytime to exit.\n");

// Adds range device (in this case, sonar device) to the robot.
robot.addRangeDevice(&sonarDevice);

// Parses the command line. Fail and print the help if the parsing fails.
if (!simpleConnector.parseArgs() || argc > 1) {

    simpleConnector.logOptions();
    exit(1);

}

printf("This program implements a shy robot. It will make the robot move away
      if there is an object near it.\n");

// Tries to connect to the robot and exit when it fails.
if (!simpleConnector.connectRobot(&robot))
{

    printf("Sorry, could not connect to the robot...exiting...\n");
    Aria::shutdown();
    return 1;

}

// Turns on the motors, turns off Amigobot sounds.
// comInt is an ArRobot function that sends a command (ArCommands) to the robot
// with an integer as argument.
robot.comInt(ArCommands::ENABLE, 1);
robot.comInt(ArCommands::SOUNDTOG, 0);

// Adds the shy away action to the robot.

```

```
robot.addAction(&shyAway, 50);

// Starts the robot running, true so that if we lose connection the run stops.
robot.run(true);

// Shuts down ARIA.
Aria::shutdown();
return 0;

}
```

## References

- [1] ActivMedia Robotics Mobile Robots Pioneer 3 Operations Manual, 2004.