Robotics Club - Colony
Introductory Lab #0 – Dance Competition

Release Date: 9/11/09

Demo Date: 9/18/09

Objectives

1. Set up a work directory and get starter code
2. Open, compile, download, and run a simple robot program using Programmers Notepad (or equivalent)
3. Write your own program using the motors, orbs, buttons, potentiometer, and serial port to make the robot dance like there is no tomorrow
4. Work with group members to make the robot to be able to execute all of your dances without reprogramming the robot

Objective 1: Download code

We have placed some starter code and a simplified version of the robot library (all of the code that works behind the scenes) on a computer in the Robotics club. If you are in Roboclub, you can access it like any other drive from a Windows machine as the W: drive. If you are prompted for login credentials, ask a senior member in the club and they will log you onto the drive. Make a new work folder in W://colony/lab0 with your Andrew ID and copy and unzip W://colony/starter_code.zip to your work folder. (If you want the code, take it with you).

Objective 2: Programming Robots

The colony robots are programmed using C. A robot can only store and run one top level (i.e. has a function named main) C program at a time. Because of this, every time you want to put a new program on the robot or modify an existing program, you will need to compile and download the program. How you do this depends on your operating system. If this is your first time writing and compiling C code it will be easier to use a Windows computer with Programmers Notepad installed. Programming the robot in a Linux environment requires some basic knowledge of package management.

Linux Option:

1. Install avr-gcc, avr-libc, and avrdude using your favorite package manager.
2. Type “make” from your work directory to compile.
3. Plug the robot in and put it in programming mode as listed above.
4. Type “make program”.
5. Reset the robot.

Windows Option: Programmers Notepad is an Integrated Development Environment (IDE) that makes it easy to program the Colony robots. It should be installed on all of the Roboclub computers, and is free to get it on your own computer. If you are programming the robot from your own computer you will also need to install the FTDI usb-serial drivers.

1. Once you have downloaded the starter code from the W drive, run Programmers Notepad and open FirstProgram.c.
2. To compile FirstProgram.c select Tools>[WinAVR] Make All. There should be some text scrolling through the output window ending with “Process Exit Code: 0”.
3. Put the robot in programming mode by turning the power switch from off to on while holding down button 1. The orbs should blink blue and green for a bit and then one orb should stay lit green.
4. Plug the robot in to the computer with a USB cable. If a bubble pops up saying it is installing the serial port, you will have to change the port to COM4 in Control Panel>System>Hardware>Device Manager>Ports(COM & LPT)>USB Serial Port>Properties>Port Settings>Advanced>COM Port Number. You will need to unplug and re-plug the cable for the change to take effect.
5. Program the robot by selecting Tools>[WinAVR] Program. Text will scroll through the output window and again end in “Process Exit Code: 0”.
6. To run the program reset your robot by turning the power switch off and on (without holding any buttons).

If all has gone well your colony robot should now be running FirstProgram.c.

Once you can program the robots, it is often necessary to debug your programs. Later in this lab you will need to use the serial ports to print the status of the robot to your computer. To communicate using the serial port, you will need to download and install TeraTerm Version 2.3 if you are using Windows or gtkTerm if you are using Linux.

Objective 3: Telling the Robot to Do Your Bidding

This zeroth lab will familiarize you with basic input and output on the colony robots. Detailed explanations of each function can be found in the API located on the website. To exhibit your mastery of these functions you should make the robot perform a sequence of motions and light displays that changes based on input from the buttons and potentiometer. You will also learn how to use the serial output terminal to read text output from the robot.

Motors

You can set the speed and direction of each motor individually using motor_x_set, where x is r or l. Direction can be 1 for forward and 0 for backward. We have also handily defined the constants FORWARD and BACKWARD. Speed can be 0-255, but the robot probably won’t start moving until you put a number above 170. The following code will make the robot spin at a pretty good speed:

```
motor_r_set(FORWARD, 200);
motor_l_set(BACKWARD, 200);
```

Orbs

The orbs are controlled with the orbN_set functions by telling each one a specific Red, Green, and Blue value, much the same as a computer pixel. The RGB value given to the function ranges from 0 to 255. You can also set the orbs to predefined colors using the orb_set_color function. For a list of predefined colors check the API documentation. This code sets Orb1 to a lovely aqua color and Orb2 to the predefined color PURPLE:

```
orb1_set(0, 200, 250);
orb2_set_color(PURPLE);
```

Serial

The robot represents text as strings, which are arrays of characters. For this lab you can explicitly write your strings by enclosing them in double quotes. To send a string across the USB cable to the computer use the function usb_puts (stands for usb put string). For example the code:

```
usb_puts(“Hello World!\r\n”);
```

will put the text *Hello World!* on the serial terminal followed by a newline (what happens when you hit the enter key). This also shows how to use escape characters. The sequence “\r\n” tells the terminal to return to the beginning of the current line (carriage return) and then go down a line (newline).

To display the text on the computer you need to run TeraTerm (check the start menu). Open a new serial connection on COM4. A word of warning, you cannot program the robot while TeraTerm is connected to the serial port. You can disconnect and reconnect TeraTerm from the serial port from the File menu.

Buttons

The robot buttons are easy ways to change the behavior of your robot based on user input. The button_click function will return a 1 if the user has pressed and released a button. The following code will change the orb1 color to red if the user has clicked button1:

```
if(button1_click())
{
     orb1_set_color(RED);
}
```

Potentiometer

The potentiometer wheel provides a way to input analog values to the robot. Calling the wheel function will return a number between 0-255. You can use this value to control other functions calls with variables. This example code stores the value read from the wheel in a variable, and then uses that variable to control the speed at which the robot spins.

```
int wheel_value = wheel();
motor_r_set(BACKWARD, wheel_value);
motor_l_set(FORWARD, wheel_value);
```

Use these functions to make the robot do a little dance. Your dance should be about 5 seconds and use all of the functions above. When button1 is clicked your robot should do something special for a second or so. To time your robot dance, you can use the delay_ms (delay for x milliseconds) function. To wait for one and a half seconds you would call delay_ms(1500). At the beginning of your dance print something like “I am doing Austin’s dance!” to the serial port.

Your Final Test

Work with a couple of other people to make a single robot cycle between all of your dances when you click button 2. You can do this by putting your sequence of dance commands in a function, which will be called from a loop in the main function. If you have some programming experience, try to make the framework for switching between dances yourself. Otherwise check out multi_dance.c for ideas.

Demo Date: 9/18/09