

**RSS Spring 2008
Robot and Development Perspective
and
Schematics and PCB Layout**

**Distributed: Friday, February 8, 2008
To be signed off by: 3:00 PM, Monday, February 11, 2008**

Your objectives in this lab are:

- to gain perspective on the computational architecture of your robot
- to gain experience in reading a circuit schematic
- to gain experience in reading a PCB component layout
- to familiarize yourself with using a multimeter
- to learn to use the class Wiki

By the end of the lab, you will have a working, powered ORCboard and ORCpad and will be ready to tackle the Motor Control Lab.

To start the lab, you should have on your bench:

MASLab™ ORCboard and ORCpad
Robot power supply

Paper Handouts (in addition to this one, Handout 1-A, Lab Procedure):

- (1-B) RSS Robot and Development System block diagram
- (1-C) ORCboard layout (4 pages: front, back and ORCpad front and back)
- (1-D) ORCboard parts list
- (1-E) ORCboard schematic
- (1-F) Agilent Scope How-To
- (1-G) Lab Checkoff sheet. Only 1 per group is needed.

Electronic Resources (available online as per direction of the lab staff)

ORCboard instruction manual
Glossary
Datasheets for ORCboard

Part 1: RSS Robot and Development Environment Perspective

This part of the lab is largely an oral presentation by the lab instructor that refers to the RSS Robot and Development System block diagram in the Paper Handouts. Please review the block diagram while you listen and make notes of your own.

Over the coming term, in addition to designing and building robot hardware, you will develop software to process input from your robot's sensors (e.g. camera and sonar) and generate output to command its actuators (e.g. motors). The block diagram represents a high-level overview of how you will develop your software, how and where that software will execute and how that software connects to the robot hardware.

Starting on the top right-hand side of the block diagram:

You will program your robot at a remote workstation, which is a Sun workstation running the Linux operating system. It is under your lab table. Your programs will execute on a laptop mounted to the robot in a runtime environment labeled as 'Student Java Code'. You will create, edit, save and compile these programs on the Sun workstation. You are free to use any code editor of your choice, but we can only guarantee to support those which we have pre-configured for you: emacs, VIM, and Eclipse, a Java IDE. After compiling, you will launch your code by transmitting the binary executable to the laptop, then issuing another command on the laptop to start running it there.

We have arranged for automatic, nightly backup of your group workspaces on the Sun workstations. All other directories on the workstation---and **all** directories on the laptop---are **not backed up**, so do not keep the primary copy of any important file in those locations. Next week, we will introduce a version-based software repository, so that your team members can work without coding clashes.

To review: You will develop code (editing, saving, compiling) at the Sun workstation. When the code is ready to be tested, you will send the compiled binary over a network connection (labeled "1" on the block diagram) to run on the robot. The protocol is TCP-IP. This is explained in the glossary, if you want to learn more.

Robot control will be directed by the laptop that will be mounted on your robot. The remote workstation enables you to edit the software or monitor the activity of the laptop.

You will run other programs on the workstation that provide information about the robot. For example, you can run a GUI (graphical user interface) that shows the status of the ORCboard, motors, encoders and sensors. In later labs, you will also

run a Java debugger and the Carmen GUI to monitor the robot remotely. We will introduce the Java debugger, Carmen, and its GUI in the Visual Servo Lab.

The laptop, upper left quadrant of block diagram:

The laptop will handle high-level computations. For example, it will execute path-planning and localization algorithms. It supports the robot's runtime software environment. This consists of two software libraries: the RSS library and the Carmen library. Your software, the 'Student Java Code' layer, is logically above both libraries; it calls routines from them. All the software you need to see and develop is exclusively in Java. The RSS library is written in Java. Because the Carmen library is written in C, we have provided Java "wrapper" procedures that call the Carmen C procedures.

The laptop manages only one sensor for the robot: the camera. When you mount the camera (Visual Servo Lab), the camera images (or 'frames') will be sent through the USB connection to the laptop. To control the motors and use other sensors, the laptop communicates via a serial interface with the ORCboard. This physical interface has an accompanying software component---the ORC daemon (Daemon is a term for a resource process; process-based computation will be covered in the Software Engineering Lecture). Learning the gory details of the ORC daemon is not necessary now. It is sufficient to think of it as a conduit that A) makes information from the ORCboard available to the RSS and Carmen libraries, and B) sends configuration and control instructions to the ORCboard.

The ORCboard and robot motors and sensors, bottom of block diagram:

The ORCboard supports the lowest level of computation in your robot, and physically connects all of the robot's computations to its sensors and actuators.

The ORCboard is powered by either a rechargeable lead-acid battery or an AC power supply. When working exclusively on the workstation, the ORCboard power can be off.

The ORCboard's software is executed on the microcontrollers within its two PSOC ICs (Programmable System-on-a-Chip Integrated Circuits, see Glossary). This software happens to be written in C. Each microcontroller runs a minimal operating system that executes communications and driver software. The microcontroller software that processes a sensor input and readies the data for transfer on the serial port is called a sensor driver, and the microcontroller software that processes motor control commands from the laptop (coming in on the serial port) is called a motor driver. The communications software handles low-level message transfer and coordination of the serial port.

The ORCboard is the place where the 'brains meet the body' of your robot. A motor or sensor has some set of wires, and these wires are physically attached to appropriate connection points on the ORCboard, which then makes that device available to the robot software (one exception is the camera, which is such a high-bandwidth sensor that it needs a dedicated connection to the laptop, bypassing the ORCboard).

Part 1, to hand in: Nothing.

Part 2: Powering up and learning about the ORCboard

Plug your ORCpad to your ORCboard. Make sure the cable is oriented correctly. The red wire corresponds to the dot on the ORCboard and the dot on the ORCpad.

Power up the ORCboard. You should now see the Maslab startup screen on the display of your ORCpad. If you do not see the Maslab startup screen, you may need to adjust the contrast on your ORCpad, which is a potentiometer labeled R1. This will take a very small screwdriver which will be on hand at the hangar. If you still do not see the Maslab startup screen, ask a TA/LA for help.

Now, locate the following 12 components on the ORC schematic, and on the physical board.

(Page 1) J23 and H1 in D6 and C6 respectively

(Page 2) U1 in C5 and J1 in C1

(Page 3) U3 in C5

(Page 4) U2 in C5

(Page 6) J2 in D4, J4 in C4, J13 in B4, J14 in B4, U4 in C5, and U5 in A5.

These components play roles in three important circuits on the board: motor, power and communication. Identify these three circuits by working from the components you located on the schematic. Track the critical signal paths (considering which direction data or signals go) through them by referencing pin and signal labels and by briefly looking at the data sheets of the components. Try to surmise what you think the circuits do. Here is where team members with more expertise should share and teach their team mates. This exercise is going to vary in difficulty depending on your background.

Conclusion of Part 2: Show the circuits to an LA/TA/instructor on the schematic and on the board, and give a high level description of what you think the circuits do on your checkoff sheet.

Part 3: Scopes

We now want to introduce everyone to oscilloscopes. Oscilloscopes are a very useful tool for debugging circuits, allowing you to see how the voltage in your circuit changes with time. For this part of the lab, we will investigate how the OrcBoard provides power to drive the motors. One of the TAs or LAs will guide your team in using the scope, supplemented by the "Scope" handout.

The OrcBoard controls motor speed using a technique called pulse-width modulation (PWM). We'll learn much more about this next week, including implementing a PWM controller, but all you need to know about PWM for now is that it uses a square wave to control the average power received by the motor. Our goal will be to measure the properties of this square wave.

In order to measure the PWM, we need to find where the motor output channels are located. Use the OrcBoard schematic to find these outputs and then locate them on the actual OrcBoard. Once you have found the motor outputs, attach one of the scope's probes. The probes we use have an alligator clip to attach to ground and the main probe tip that has a removable hook on the end.

After attaching the probe to the motor output, we now need to adjust the settings on the scope to handle our signal. The steps for doing this are:

- Setup the display to draw the output in the desired format -- I prefer having the grid and connecting the measurement dots
- Change the sec/div and volts/div to fit the entire signal in the window
- Set the triggers for the signal -- using the auto-trigger is fine for this lab

With the scope properly calibrated for our signal, we can now set out to measure the outputs generated by the OrcBoard. To get the OrcBoard to output a signal, attach the OrcPad and change to Drive Mode. Move the joystick around to see the PWM output.

If the PWM is displaying correctly, then we can try to measure the properties of the signal. In particular, we want to find:

- The peak-to-peak period
- The duration of the high signal per period
- The duration of the low signal per period
- The magnitude of the signal

Conclusion of Part 3: Work through the scope demo with a TA/LA. Show the TA/LA that you can characterize the PWM signal; s/he will then check you off.

Part 4: Using the Wiki..

Later in the course you will be using the class Wiki to submit some of your assignment writeups. A Wiki, as many of you may already know, is a web site which has on-line edit functionality. The lab staff will give you the URL to a Wiki we have set up for RSS I. Bring it up and create an account for each group member. Then create a page with your group's name, and on that page make a bullet-list of your group members' names, email addresses, cell numbers, and any IM contact information. Then create a concise summary of each group members' schedule in a format of your choosing. You may find the help section of the Wiki useful as you get started.

Conclusion of Part 4: Have the LA/TA/instructor inspect your Wiki pages. This will complete your checkoff. Bring the checkoff sheet to the next lab. This checkoff must be done before 3:00pm on Monday, February 11th.