Massachusetts Institute of Technology

Robotics: Science and Systems I

Lab 1: System Overview and Introduction to the ORCboard

Distributed: Wednesday, 2/4/2009, 3pm

Due: Wednesday, 2/4/2009, 5pm

Objectives

Your objectives in this lab are to:

- Gain perspective on the computational architecture of your robot
- Gain experience in reading a circuit schematic
- Gain experience in reading a PCB component layout
- Familiarize yourself with using a multimeter
- 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.

Materials

At your workstation:

- MASLab^{TM} ORCboard and ORCpad
- Robot power supply

Paper handouts:

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

Electronic resources for extra information are available online at
`http://courses.csail.mit.edu/6.141/spring2009/pub/labs/GeneralInfo/Orcboard-ORC4/`

- ORCboard instruction manual
- Datasheets for ORCboard

1 Equipment check

Check that your group has the following materials in your bin, and tell a LA if something is missing. Although not necessary for this lab, these tools will be needed in later labs. You may also label your equipment with your group number, if you wish, to reduce confusion later in the term.

- Hardware:
  - 2 sets of English unit Allen wrenches, one detachable and one loose
  - Small Phillips screwdriver
  - 5/8" wrench
- Measuring:
  - Protractor
  - Metric measuring stick
  - Measuring tape
- Wiring:
  - Multimeter with 2 probes
  - Wire cutters
  - Pliers
  - Roll of electrical tape
- Soldering:
  - Soldering iron with stand and sponge
  - Solder
  - Copper wick for desoldering
  - Desoldering pump
  - Helping hand
- Other:
  - 2 Sharpies
  - Roll of masking tape
  - Velcro tape (both sides)
  - X-acto craft knife
  - Wooden robot stand
  - C-clamp

2 Overview of robot and development environment

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, a Sun computer under your lab table. It is running Ubuntu, a distribution of the Linux operating system. Your programs will execute on a laptop mounted to the robot, in a runtime environment labeled on the diagram 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.

Next week, we will introduce a version-based software repository, so that your team members can work without coding clashes. We backup these repositories. None of the other directories on the workstation or the laptop are backed up, so keep any important files in the repository.

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. Running debugging program on the romote workstation reduces the computational load on the laptop and improves the robot's preformance.

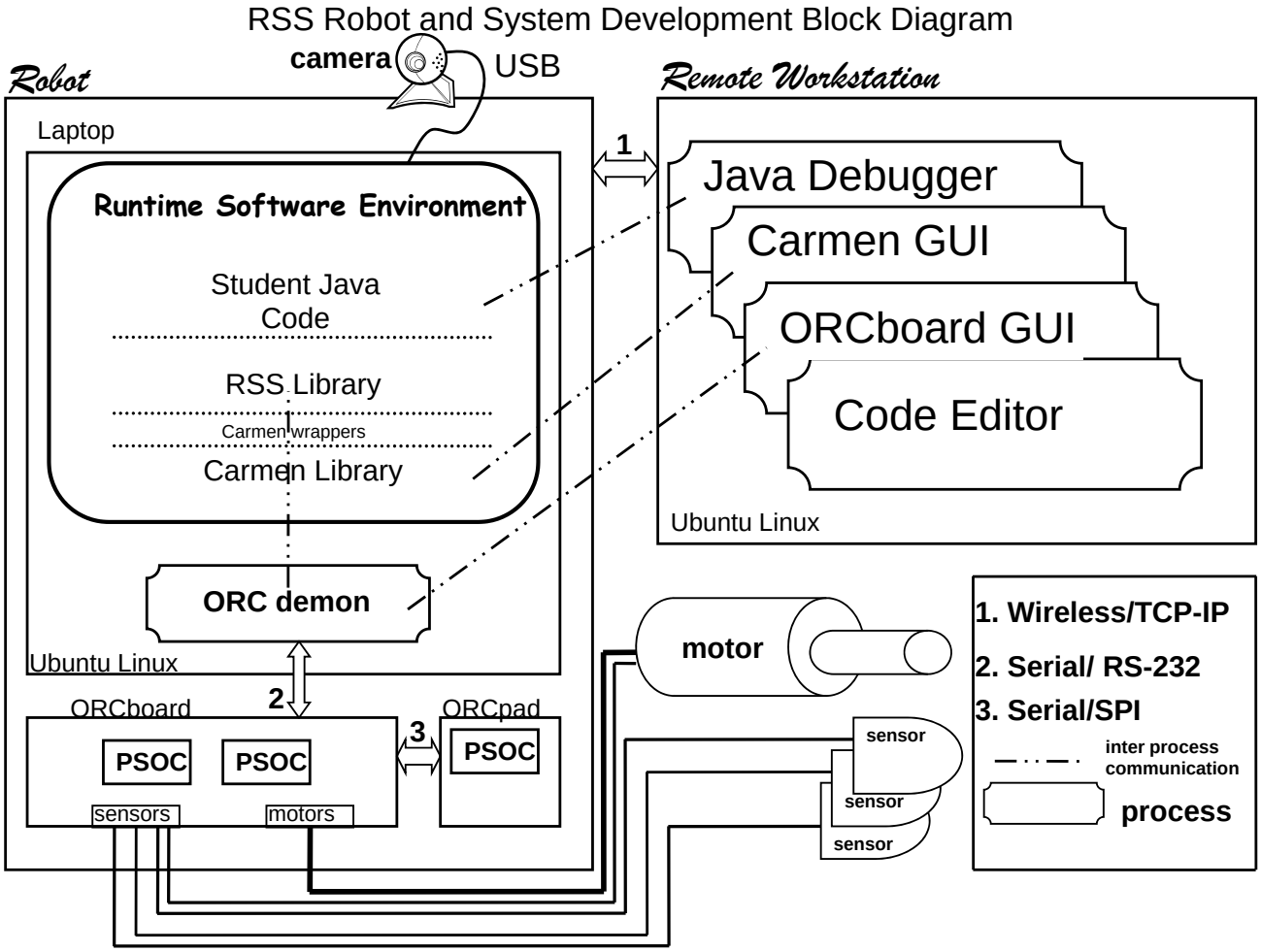

Figure 1: RSS Robot and Development System Block Diagram

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 type of 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. It is powered by either a rechargeable lead-acid battery or an AC power supply. When on, the Orcboard should always be connected to battery, and optionally to the AC power as well. Do not run the Orcboard solely off AC power; it is insufficient and may cause problems.

The ORCboard's software is executed on the microcontrollers within its two PSOC ICs (Programmable System-on-a-Chip Integrated Circuits, see the glossary at the end of this lab). 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 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 is such a high-bandwidth sensor that it needs a dedicated connection to the laptop, bypassing the ORCboard).

Deliverables: Nothing.

3 Powering up and learning about the ORCboard

1. Cover the back of your ORCboard in electrical tape, to prevent shocks.
2. 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.
3. 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.
4. 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.

5. 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.

Deliverables: 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.

4 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 "Agilent Scope How-To" 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.

1. 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 a main probe tip that has a removable hook on the end.
2. 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:
   - Set up the display to draw the output in the desired format – it may be better to have the grid and connect 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
3. 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.
4. 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

Deliverables: 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.

5 Using the Wiki

1. Log onto your group's Sun workstation with the username rss-guest and the password given to you by the TAs. This should be the only time you use rss-guest; you will have individual accounts on that machine by the next lab session.
2. Go to the course Wiki, at `http://projects.csail.mit.edu/rss/wiki/index.php/Main_Page`. Later in the course you'll be using it to submit assignments. You can also see work by groups from past years, but be careful - assignments may have changed.
3. Have each group member create a Wiki account.
4. Edit the page with your group's name, and make a table containing your group members' names, email addresses, cell numbers, and any IM contact information. You may find the help section of the Wiki useful as you get started.
5. Create a concise summary of each group members' schedule in a format of your choosing.
6. Upload a picture of your group's favorite robot, and post it to your group's page. You can use Google Image Search to find a picture; we just want you to have practice uploading files to the wiki.

Deliverables: Have the LA/TA/instructor inspect your Wiki pages. This will complete your checkoff. This checkoff must be done before the deadline listed at the top of this handout.

Deliverables

Turn in a completed Lab Checkoff Sheet, having described the ORCboard, worked with the oscilloscope, and created a wiki page.

RSS-I Glossary and Acronyms

Carmen The Carnegie Mellon Robot Navigation Toolkit is an opensource collection of software for mobile robot control implemented in C. Its programs control the movement of the robot and accept inputs from the sensors plus handle more sophisticated tasks like mapping, navigation and localization. Carmen also includes a simulator. An update of Carmen has been developed for RSS and will later be released. For more information see: `http://www-2.cs.cmu.edu/~carmen/`

Daemon (From whatis.com) A daemon (pronounced DEEmuhn) is a program that runs continuously and exists for the purpose of handling periodic service requests that a computer system expects to receive. The daemon program forwards the requests to other programs (or processes) as appropriate.

DIP Dual Inline Package. "Dual inline" refers to two parallel sets of pins. DIP packaging is the standard packaging used for most regular integrated circuits.

Driver (From whatis.com) A driver is a program that interacts with a particular device or special (frequently optional) kind of software. The driver contains the special knowledge of the device or special software interface that programs using the driver do not.

Ethernet protocol A very common method of networking computers in a local area network involving bridges, routers and even switches. Ethernet will handle about a Gigabit of data per second and can be used with almost any kind of computer. For more see: `http://computer.howstuffworks.com/ethernet.htm`

GUI Graphical User Interface

IDE Integrated Development Environment

ORCboard ORC = "Our Robot Controller". The "OrcBoard" robotics interface board was designed and manufactured for the MASLab staff and is an evolved version from previous year's competitions. It contains an extensive list of capabilities, including

- Four high current motor drivers with dedicated motor enables and current sense.
- I2C expn port (400kbps)
- 12+ runtime reconfigurable digital IO ports (sonar, quadphase, current sensed servos, bump sensors, etc...
- 8+ analog ports
- OrcPad connector (and of course, an OrcPad)
- A "brain board" attachment header, allowing easy integration of a tiny microcontroller rather than a mongo MASLab computer.
- Size: 3.5" x 4.5".

In addition, the board has about 25 MIPS of CPU power built in, allowing additional features such as programmable lowpass filters on the A/D ports and programmable current limiting of motor ports.

ORCPad The OrcPad is a detachable accessory for the OrcBoard which provides human interface elements including buttons, a joystick, and a graphical 128x64 pixel display. The OrcPad was also designed and manufactured by the MASLab staff.

PSOC Programmable System on a Chip

PCB Printed Circuit Board. See `http://www.lvr.com/pcbs.htm` for more information.

RS-232 RS232 is a serial communications standard that provides asynchronous communication capabilities, such as hardware flow control, software flow control, and parity check. It has been widely used for decades. Almost all gears, instruments with digital control interface, and communications devices are equipped with the RS232 interface. The typical transmission speed of an RS232 connection is 9600 bps over a maximum distance of 15 meters. From `www.moxa.com/services/glossary.htm`

Serial protocol A communication standard where information is sent one at a time as opposed to in parallel.

TCP-IP (from whatis.com) TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet). TCP/IP is a twolayer program. The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination.

USB USB is an abbreviation of Universal Serial Bus. USB is a standard port that enables you to connect external devices (such as digital cameras, scanners, and mice) to personal computers. The USB standard supports data transfer rates of 12Mbps (million bits per second), a vast improvement over the serial port standard it is beginning to replace. Aside from speed advantages, USB devices can be connected or disconnected without the need to restart the computer.