CS148 Laboratory Notes Part B

Semester II, 1997-98

This Document is intended to supplement Part A of the Lab Notes.

The Kit

You should already have the following components, each of which is described in Part A of the Lab Notes:

- **Legos**
- **1 Motor Battery Pack and Charger**
- **2 Motors**
- **1 Motor Switch**

And you should have just now received the following components, which completes your kit. Please take good care of these materials. **Do not use them until you have read and understood this handout.**

- **6.270 Board**

  This computer on a board, specifically designed to make reading your robot's sensors and controlling its motors easy, is described in Section 1, "6.270 Board Introduction" and Section 3, "Using the 6.270 Board"

- **Serial Line**

  This is the connection between your 6.270 and the Sparcs for downloading programs. Downloading programs is covered in Section 2.

- **Sensors**

  Each kit contains two light sensors and two bump sensors. Section 4, "Sensor Design", covers the operation of these sensors and others that are available in the Lego Lab.

1. 6.270 Board Introduction

The Board

The 6.270 board itself was designed for the MIT 6.270 course in robotics. It is still used for that purpose, and has been adopted for use in Brown's CS148. It is based around the Motorola MC68HC11A1 microcontroller chip, and has: 32K of battery-backed RAM, 4 motor control outputs, a Serial port (connects to RS-232), an LCD display, a small speaker/buzzer,2 user-controlled LED's, 8 digital input ports, 13 available analog input ports, 4 dip switches, a "frob" knob, and two push-button inputs.

The general layout of the 6.270 board is as follows: Set the board in front of you, with the LCD screen in the far right corner.There is a control knob for the LCD's contrast to its left and down on the main board. The connector for the serial line will be on the far left corner, with the choose and escape buttons to its right (they are small and white). The four motor ports are coming toward you from the serial line jack. Each has a read and a green LED next to it, which indicate which direction the motor is going when it is on. On the near left corner is the motor battery jack. You will need to have a battery pack charged and plugged in here to use the motors or the IR beacon. Next to the battery jack is the main power switch (large and black). To its right is the large red reset button. Above the reset button is the frob knob. To the right of the reset button are the digital sensor ports (0-7). To the right of the digital ports, near the front edge of the board, are the two user-controlled LED's, labelled LED17 and LED18. Finally, the usable analog ports (15-27) are on the right side of the board, and the dip switches are just above them in the far right corner. All of the ports and the dip switches have their numbers printed on the circuit board or on the front of the component.

The major caveat with the board is to treat it with care. It is a sensitive piece of precision electronic equipment. Note also that the components are exposed, especially on the bottom of the board. The main concern besides physical damage is therefore static electricity, or a direct short-circuit between components. Be careful not to expose the board to these dangers as well as not dropping, hitting, or spindling, mutilating or mauling it. Also be careful of the battery pack for the AA's: do not let it hang from the cards or you could damage it and take your board out of commission.

AA batteries

The AA batteries which go in the battery case attached to your computer board are standard alkaline or nicad batteries. Under most situations they will last quite some time, as the computer and sensors draw very little power. To extend battery life, you should of course always turn off the computer when it is not in use. If you do drain your AA batteries, see a TA for a new set, or simply pick up a pack of four alkaline AA batteries, such as Duracell or Energizer brands, from any store.

One condition which can cause the AA batteries to drain more quickly is the use of infrared sensors later in the course. These sensors actively send out infrared light and look for its relfection off of nearby obstacles. Whenever your computer is turned on and the infrared sensors are plugged in the sensors are using battery power to emit infrared light, whether your program is running or not. As a result, battery life can be drastically reduced if you leave your computer on with infrared sensors plugged in. To increase battery life, you may unplug the infrared sensors or turn off the computer when you are not using the robot or downloading a program.

If your batteries are running out, your board may restart seemingly randomly or hang in mid-execution, or it may even refuse to go into or come out of download mode. There is some software that will hopefully warn you if the power is too low, and there is a red battery-low indicator LED next to the serial LEDs, but these tend to kick in too late, so if you cannot solve a hanging or resetting problem in software, try using batteries you know are good. You should also know that a set of batteries may be able to run other devices, but not have enough juice for the 6.270 boards, which are fairly picky about getting full voltage. We may have some extra new batteries in the LegoLab, in case yours run out, but please don't rely on this generosity often. Make sure you turn off your board when you aren't using it.

2. Communicating with the 6.270 board from the Sparcs

The primary method of communication between the 6.270 and other computers, like our Sparc workstations, is through a serial port. Your kit contains a serial cable for this purpose. We do not have any extra serial lines, so please keep track of yours!

Connecting your robot board to the Sparcs

- Plug the large RS-232 type connector on one end of the serial cable into the port labeled "A" on the back of the Sparc. Plug the phone-type connector into the 6.270 board (upside down).
- The green LED at the opposite end of the board labeled SER RCV should light. If it is not on, then unplug the whole thing and start again. If this is a persistant problem, make sure that your serial cable is not damaged or frayed.
- If the green SER RCV light is on, all is well. You are ready to communicate with the 6.270. To do this from the Sparcs, you will use a program called "dl" to download compiled programs.

Compiling and downloading programs

How you will compile a program for use on the 6.270 will depend on the source language, and will be described seperately later. For the moment we assume that you have a valid program and have compiled it into the S-record format (*.s19). Once your robot board is connected to a Sparc via the serial line, perform the following steps to download your program:

- If the board is off, turn it on with the big black switch. Be sure that the serial line is connected as above and the green LED is lit.
- You need to put the board in download mode so that it will accept a program on the serial line. To do this, hold the "choose" button down while you press and release the red reset button. The yellow SER XMIT LED next to the green LED should turn off. This may take a few tries at first but you will get the hang of it. Make sure that the yellow LED stays off, rather than just blinking.
- Once the board is in download mode, you are ready to initiate comunication from the Sparc side. To do this you use the "dl" program. If your program was compiled into a file called "robot.s19", run "dl robot.s19" from the directory that the s19 file is in. The green and yellow LEDs on the 6.270 will flicker for a bit, and you will be able to keep track of the download from a display in your shell.
- When the download is finished, simply press the red reset button or power cycle the 6.270 to start your program running.

3. Using the 6.270 board

Programming the board

The board itself is programmed in 68hc11 assembly. At MIT, a C interpreter with a library of routines specific to the 6.270 board's hardware was written to allow students to write in C. Here at Brown we will be programming in Rex and perhaps C, and compiling the code into 68hc11 assember which will be loaded onto the boards without the need for a C interpreter.

C and REX

Rex is the robot control language we will be using to control our robots. This language uses a conception of a synchronous control "circuit", with data carried between small function processors on "wires". This language can also be compiled into C, and thence compiled and transferred to the 6.270 boards. In Rex, you will be provided some software "wires" that correspond to the robots inputs which can be read, and wires corresponding to robot outputs which can be set. If we have not covered Rex in class yet, don't worry if this sounds a bit confusing. In some cases, students may want to access some part of the 6.270 board which is not yet supported by Rex. In this case, some support code for your robot program may have to be written in C.

The Robot Inputs and Outputs

The 6.270 board provides a number of inputs your program can read to get information about the world and a number of outputs it can set to control the robot.

MOTORS

The motors are bidirectional and can be run at seven different power levels in either direction. The lower power levels may well be too low for use in propulsion, but might be useful for controlling the motors of a claw or a light. The motor outputs are numbered 0-3. The odd and even numbered motors will run in opposite directions. This is convenient if the odd and even motors are mounted on opposite sides of a symmetrical robot. In any case, the easiest thing to do is wire up your robot and test the motor directions.

LCD

The allows you to print out debugging information from your robot programs.

SENSORS

There are two types of sensors on the 6.270 boards, digital and analog. Ports 0-7 are digital ports and return values of 0 or 1. Ports 8-14 are analog ports, but are covered or obstructed by the design of the board. Sorry. Ports 15-27 are analog ports also, and can return values from 0-255. If you put a digital sensor in an analog port, you will typically get very low (0-5) or very high (240+) values.

MISC.

There are two LED's that you can control from your programs, located to the right of the digital ports on the expansion board. There are two input buttons on the board also, “choose” and “escape”. There are four dip switches for your use to the right of the analog sensor ports. . There is also an arbitrary analog input, the “frob knob”, next to the red reset button, which will return 0-255 like an analog sensor port.

Input and Output from REX

Rex defines two structures for you to use; lego-in and lego-out. The first of these, lego-in, allows you to read the values which the sensors and various switches and knobs had at the start of the current execution cycle. The structure is defined as follows:

```
(define-rex-struct lego-in
    ((ana12 int) (ana13 int) (ana14 int) (ana15 int)
        (ana16 int) (ana17 int) (ana18 int) (ana19 int)
        (ana20 int) (ana21 int) (ana22 int) (ana23 int)
        (ana24 int) (ana25 int) (ana26 int) (ana27 int)
        (dig0 bool) (dig1 bool) (dig2 bool) (dig3 bool)
        (dig4 bool) (dig5 bool) (dig6 bool) (dig7 bool)
        (dip0 bool) (dip1 bool) (dip2 bool) (dip3 Bool)
        (choose bool) (escape bool)
        (frob int) (cycle int)))
```

Each of the members of this structure is a Rex wire and can be used by your program to determine the robot’s current state. There are six types of input that you have access to:

- Analog input ports. These are the members ana12, ana13, *etc.*
- Digital input ports. These are the members dig0, dig1, *etc.*
- DIP Switches. These are members dip0, dip1, dip2, dip3.
- Buttons. These are inputs choose and escape.
- Knob. This is input frob.
- Cycle. The number of execution cycles performed thus far.

To provide output to control the motors and other external devices, Rex provides an output structure, lego-out. This is defined as follows:

```
(define-rex-struct lego-out
    ((motor0 int) (motor1 int)
        (motor2 int) (motor3 int)
        (led0 bool) (led1 bool)
        (lcd-int0 int)
        (lcd-int1 int)
        (lcd-int2 int)
        (lcd-int3 int)))
```

Again, each of the members is a Rex wire and can be assigned to, resulting in a change in the robot's actions. There are three types of output for your use:

- Motors. The boards can support up to four motors, although you will typically only be using two. The speed and direction of the motors is set with the members motor0, motor1, motor2, motor3. Positive speeds are forward.
- LEDs. The two LEDs on the board can be set on or off with the members led0 and led1.
- Display. The LCD currently can display four numbers from Rex. These numbers are set using lcd-int0, lcd-int1, lcd-int2 and lcd-int3.

One important thing to note is that somewhere during **each** execution cycle **all** of the members in the output struct must be assigned a value, otherwise the results are unpredictable. Just because they were assigned a value last time does not mean that they will keep that value the next time. Also, don't try to assign values to the input structure, as this might lead to chaos too.

C The Library Functions

In some cases a particular project may require direct access to parts of the 6.270 board not yet accessible from Rex, such as the speaker or serial port. In those specific cases, some C programming may be necessary. See your TA for more assistance. Below is an overview of the robot functions accessible from C.

INIT
At the beginning of any program, and in any case certainly before using the motors, analog sensors, dip switches or frob knob, or the LCD display, you should call "init_legobot();". This runs some maintenance and setup funtions and initializes the system.

SLEEP
To put a pause in your program for a certain amount of time, use the "msleep(s)" function, where s is the number of milliseconds to sleep.

MOTORS
To set motor n on in direction d, at speed s, call "setmotor(n, d, s);". To change the speed of motor n to s, call "setspeed(n, s);". To change the direction of motor n to forward or backward, call "fd(n);" or "bk(n);", respectively. To turn motor n off entirely, call "off(n);". To turn all the motors off at once, call "alloff();". Before using the motors, you should have called the "init_legobot();" routine, which resets the motors to off, forward, and maximum speed, and enables the code which lets them change speeds.

SPEAKER
The speaker is capable of producing sound at any given frequency. To sound a tone at freq Hz for msec milliseconds, call "tone(freq, msec);". To set the frequency of the speaker to freq Hz, call "set_pitch(freq);". To simply turn the speaker on or off at the current frequency and leave it that way, call "beep_on();" or "beep_off();", respectively.

LCD

The LCD screen is 2 rows by 16 columns. Using it is much like a normal C printf: use the function “LCDprintf(controlstring, args);”. The controlstring is a string (between double quotes) and cn have the special characters %d, %x, or %b, which will print the args in order in decimal, hexadecimal, or binary format, respectively. “\n” is a newline character. If your cursor is on the lower line, and you print a newline character, the cursor returns to the beginning of the upper line. The far left corner of the LCD is space 1,1.To clear the LCD and set the cursor back to space 1,1, call “LCDclear();”. You must call “init_legobot();” or the LCD will not respond to LCDprintf or LCDclear.

SENSORS

If you put a digital sensor in an analog port, you will typically get very low (0-5) or very high (240+) values from “analog(n);”. If you call “digital(n);” on ports 8-27, you will get either 0 or 1, depending on whether the analog value of the port if greater or less than 128. So to access sensor port n as digital, call “digital(n);”, and to access a port as analog call “analog(n);”. You must call “init_legobot();” before the analog ports will function properly.

MISC

- To set and LED on or off call “set_ledX(s);” where X is 0 or 1 for LED0 or LED1, and s is 0 for off or 1 for on.
- To read the choose button or esc button call “choosbut();” or “escbut();”, respectively.
- To read a dip switches, call “dswitch(n);”, which returns 0 if switch n is down and 1 if it is up.
- To read the frob knob’s value call the function “knob();”, which will return 0-255

4. Sensor Design

The sensors you provide to your robot will be its only source of information about the environment in which it must survive. Complicating matters is the fact that robots generally use cheap sensors, and cheap robots use really cheap sensors. The Lego robots you will be building are no exception, and their sensors may have a number of shortcomings. So it's a good idea to have at least some idea of how each sensor works and what exactly it is measuring, so that your system can account for, work around or even exploit its quirks.

Attached to this section is a chapter on sensor design from the MIT 6.270 course. It covers the photo transistors and switches your kit includes, which your robot will use to detect light and obstacles. It also covers a variety of other sensors, such as bend sensors and infrared-based obstacle sensors, most of which are available in the LegoLab for your projects.

On a philosophical note, one should keep in mind that no sensors are perfect, and any robot in a real environment is going to occasionally get sensory information which is incorrect or ambiguous. So, buying more expensive sensors does not eliminate the need to deal with incorrect sensor information, it only postpones or hides it. Starting with very simple, somewhat unreliable sensors like those available for your robot, while occasionally frustrating, is great exposure to kinds of problems robot software must face in a real environment, not to mention quite economical.

The file /course/cs148/handouts/sensors.ps, which is Chapter 5 of the 1993 MIT 6.270 course notes is the handout "Sensor Design".