Lab 3 – Microcontroller programming
Interfacing to Sensors and Actuators with iRobot

1. Objective

In this lab, you will:

i. Become familiar with the iRobot and AVR tools.
ii. Understand how to program a "bare iron" (no operating system) system in C.
iii. Interface with external sensors and learn to poll or setup interrupt systems.

2. Equipment

i. Computer with AVR tools installed
ii. iRobot USB interface
iii. iRobot and the command module with accelerometers connect.

3. Theory

a. Introduction

The iRobot[1] is a complete robot development kit that allows you to program new robot behaviors without having to worry about mechanical assembly and low-level code. In this lab, you will explore the capabilities of the iRobot using a command module that is added on to the iRobot Create. The command module (Figure 1) contains a microcontroller (Atmel AVR ATMega168 microcontroller, see [2] page 29 for overview, see [3] for microcontroller documentation) which allows you to write programs in C/C++ to control the iRobot. It is important to understand that you are not directly programming the iRobot, but you are programming the microcontroller on the command module to send OI (open interface, see below) commands to the iRobot through the cargo bay connector (Figure 2), which then the iRobot processes the commands and actuates itself based on the command given.

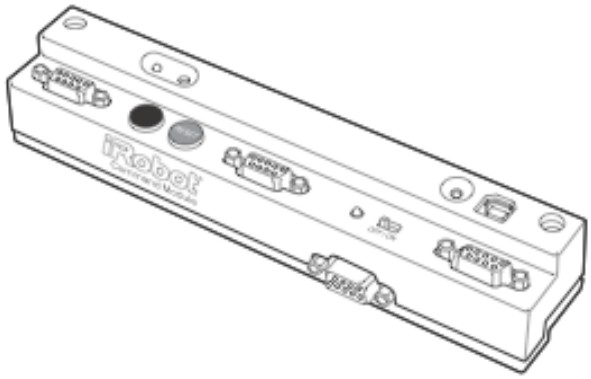

Figure 1. Command module added to the iRobot

b. iRobot platform

See Figures 2,3 and 4 for the sensors and actuators on the iRobot. In this lab, you will be using the cliff sensors, bump detectors and wheel drops to detect collision or cliffs. We will also attach accelerometers, which you will use for the final iRobot project. Feel free to use the LEDs for debugging purposes.

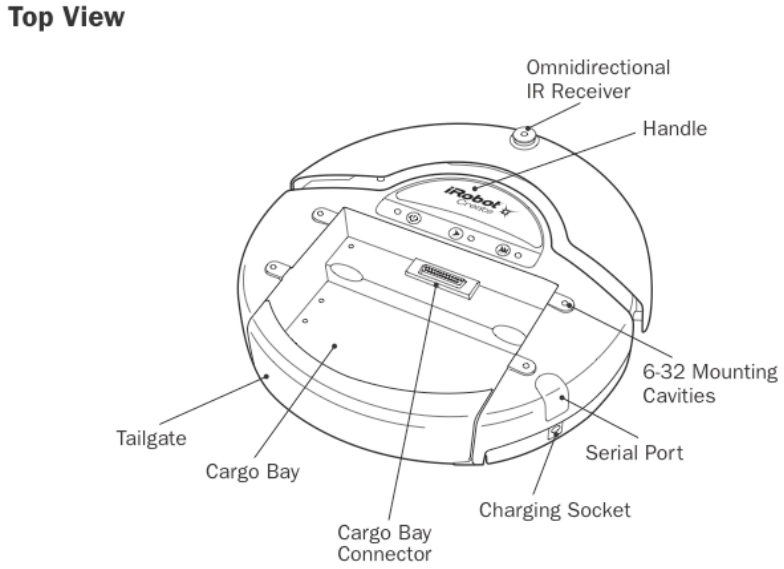

Figure 2. Top view of iRobot

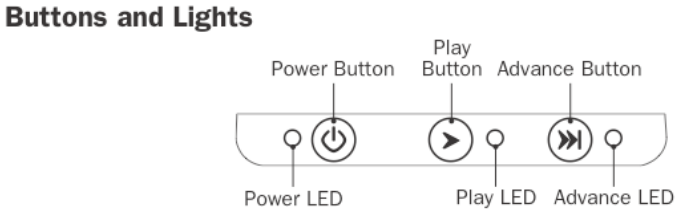

Figure 3. Buttons on the iRobot

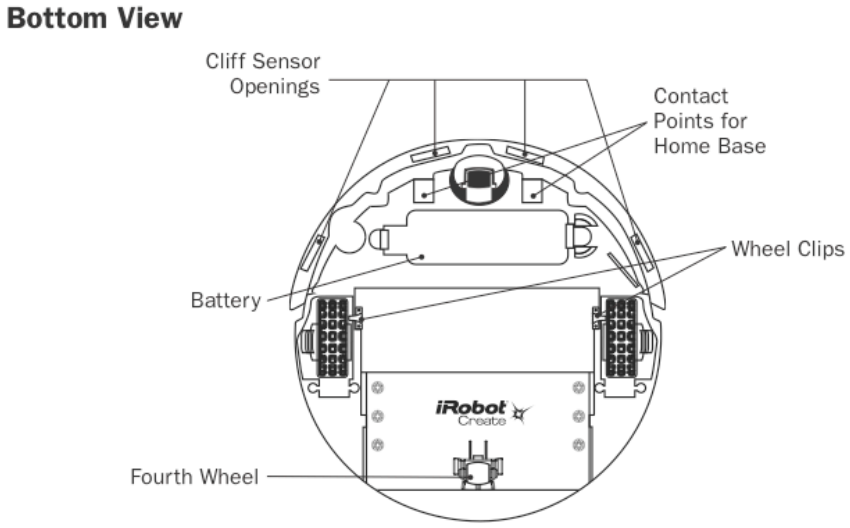

Figure 4. Bottom of the iRobot

c. Open Interface
The Open Interface (OI) defines the electronic and software interface for controlling the iRobot's behavior and reading its sensors. The software protocol is briefly explained in section 4 and 5 of [2]. A reference to all the commands can be found in [4]. Make sure read the section in [4] (page 17) on the OI sensor packets.

4. PRELAB EXERCISES
Read through the iRobot documentations posted on the iRobot page on the course webpage (http://chess.eecs.berkeley.edu/eecs124/iRobotDocs/index.html). There are a lot of documents; make sure you at least looked at them and know what's in which document. It will be useful to have a general idea so when you need to reference the documents you know which one to look at.

1. **Understanding the platform:**
    a. Which microprocessor is in the command module?
    b. How many sensors are on the bottom of the iRobot? Given the placement of some of the sensors, what problems might this cause?
    c. How many ports in the command module can be used for analog inputs? Digital?
2. **Understanding the protocols:**
    a. What sequence of bytes would you need to send to the iRobot to make it drive forward (velocity and radius values can be arbitrary)?
    b. What is the command sequence for reading sensor data from the iRobot?

5. IN LAB EXERCISES
The final goal of this lab is to program the iRobot to autonomously climb up a hill. This involves determining the correct orientation for the iRobot (you want to drive towards the top of the hill); avoid driving off the hill (using the built in iRobot cliff sensors and wheel drops); and collision detection (there might be obstacles along the path to the top of the hill).

We have split the lab into 3 parts to help you progress towards the final goal:
1. Making the iRobot autonomous
2. Finding the top of the hill
3. The autonomous hill climber iRobot

- **PART 1 – Autonomous iRobot**
    1. To get started on programming the command module, follow the step-by-step instructions in [2] to setup the programming environment. We will be using the WinAVR suite of open source development tools to compile and download your C or C++ programs.
    2. The example programs mentioned in the documentation should be installed in your computer [C:/WinAVR/examples/iRobot]. You may want to load in different example programs to get familiar with how to load programs in the

command module and run them.

3. Look at the "drive" example that's in the examples folder. You can also reference the lecture on "Interfacing with sensors and actuators" which gives the layout of drive.c on the last slide (with some code fragments removed). It also contains the references to the header files used to define the macro-names used to access the memory mapped registered. Make sure you understand the drive.c example and how to drive the iRobot and read sensor packets.
4. **Question**: How is the software for drive.c example architected? Is it an interrupt driven system? Polling Loop? Explain.
5. Build on the drive.c example. Program the iRobot so that it becomes fully autonomous using the built in sensors of the iRobot. We will test your implementation in the following conditions:
    a. Effective use of the cliff sensors and wheel drop. The iRobot should not drive off any cliffs, but instead maneuver away when it detects one.
    b. Effective use of bump sensors. The iRobot should not be stuck when there are obstacles in front of it, nor should it get stuck in corners. It should be able to maneuver in the right manner to avoid the obstacles.
    c. Basic sense of direction. We will point the iRobot in an initial direction, and expect the iRobot to roughly head towards that direction despite the recovery mechanisms that were implemented in the previous two conditions.

- **PART 2 – Hill Climbing**
    1. By now you should be familiar with how to program the microcontroller in the command module to interface with the sensors on the iRobot and how to drive the robot. The next step is to attach an extra sensor to the command module and interface to it.
    2. In lab you will be given an accelerometer attached to a breadboard along with connectors to allow you to connect it to the command module ports. It will look something like figure 5.

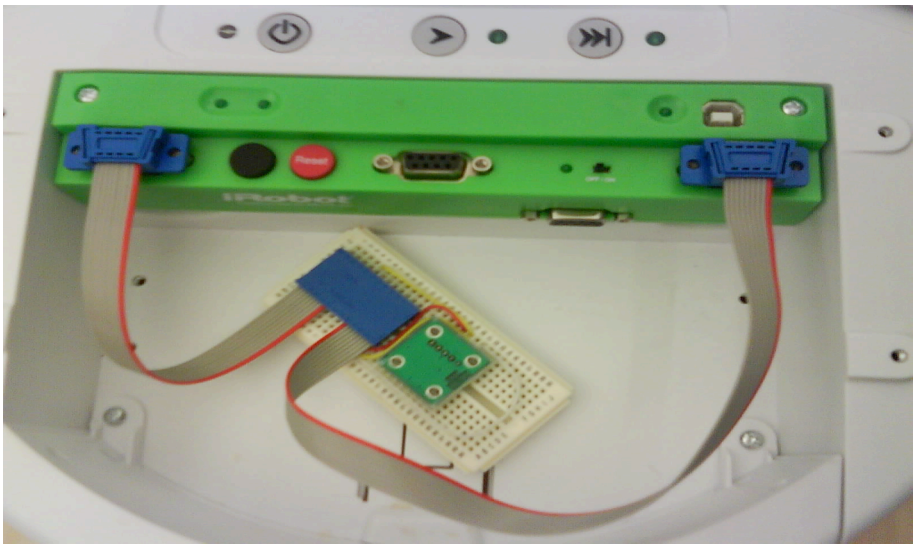

Figure 5. Accelerometer board

3. Using the data sheets of the command module and the Atmel processor, correctly wire the accelerometer to the ADC pins of the command module. You will need to keep track of which ADC pin you wired so you can access it via software.
4. From the previous lab you learned how to interpret data from the accelerometer and measure tilt and roll using it. Think about how you would easily use the 2-D accelerometer to orient the iRobot towards the top of the hill.
5. The purpose of this part is for the iRobot to be able to determine the orientation to the top of the hill. You don't have to worry about the collision detection or avoidance yet, focus on interfacing with the accelerometer to help determine where the top of the hill is.

- **PART 3 – Putting it all together**
  1. Now that you have successfully interfaced with both the sensors on the iRobot, and the accelerometer to find the top of the hill, your final job would be integrating the two together. Your iRobot should be able to climb to the top of the hill without falling off the cliff and avoiding obstacles on the hill.
  2. You will need to think about the priorities of the sensors and the accelerometer when you integrate the 2 functionalities. Which one should have the higher priority?
  3. In your lab report, explain any design decisions that had to be made, and why they were made. Also explain the structure of your code. You may use diagrams (FSMs etc) or graphs, and attach a printed copy of your code (Clean it up and comment it! Make sure it's readable) at the end of your lab report. Be sure to include stuff learned in lecture (is it interrupt driven? Polling loop?).
  4. Also in your lab report, include a one paragraph summary of the hardware/software interface. What conceptually is happening when your iRobot is running?

6. REFERENCES

[1] Documentation of the iRobot Create
http://chess.eecs.berkeley.edu/eecs124/iRobotDocs/CreateManual_Final.pdf
[2] Documentation of the Command Module
http://chess.eecs.berkeley.edu/eecs124/iRobotDocs/CommandModuleManual_v2.pdf
[3] Documentation of the ATmega168 microcontroller
http://chess.eecs.berkeley.edu/eecs124/iRobotDocs/ATmega168Reference.pdf
[4] Documentation of Open Interface
http://chess.eecs.berkeley.edu/eecs124/iRobotDocs/CreateOpenInterface_v2.pdf

7. REVISION HISTORY

Date/Author	Revision Comments