Lab 8

Controlling the Spider Robot with C++

1. Introduction

Object-Oriented programming is useful for controlling complex embedded systems such as the Spider Robot. The source code and header files C++ control code stack are given that will enable the robot to walk forward using the alternating tripod gait. You will cross-compile it on the lab desktop computer and transfer the executable file to Spider's DE10 board. You will run the code and note the performance of the Spider. Finally you will add functionality and attempt to fix any problems with the original code the executable code on the Spider robot.

2. Cross-Compilation

The DE1-SoC and the DE10Nano-Soc are running a version of the Ubuntu distribution of the Linux operating system. This is a slightly scaled-down version of Ubuntu but has a lot of the functionality of a full-fledged distribution of this operating system. The operating system runs on the ARM processor and supports the execution of custom C or C++ programs.

Because it is a custom distribution, whenever you write and compile code on your own machine, the resulting executable *will not run successfully on the boards*. While you are free to develop on whatever machine you like, whenever we run code on the DE1 board or DE10Nano of the Spider robot, you will need to follow a different procedure.

The DE10-SOC runs *Linux* and the desktop runs *Windows*, you will need to cross-compile your code. Cross-compilation is where you compile code to target a different kind of machine than you are running the compiler on; cross-compilation is slightly more complicated but is extremely common in embedded systems since the target usually doesn't have support for a compiler & complex development environment interface.

2.1 Connecting to the DE10-SOC

In this lab we will connect the DE10-SoC using an **Ethernet cable** to your computer:

Ensure that your DE10-SoC board is **powered off**, and that there is a microSD card in the

microSD card slot.

Before turning on the board, ensure that the MODE SELECT (MSEL) switches found on the bottom side of the board match the settings shown in **Figure 1**.

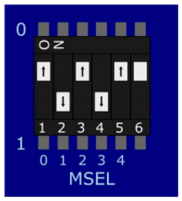


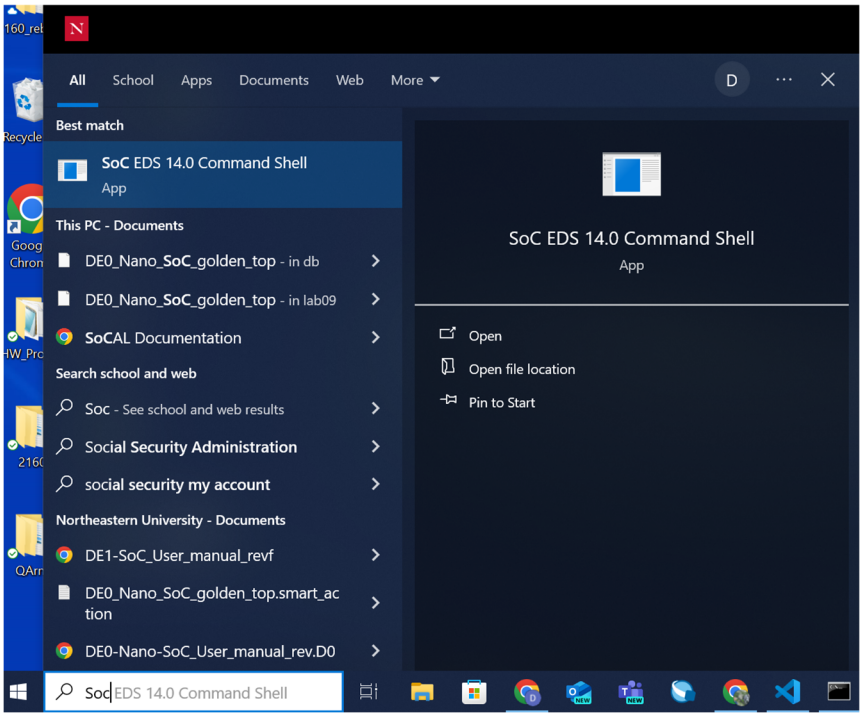
Figure 1: MSEL switches settings

- Connect your DE10-SoC board to your host computer using the provided *USB-to-Ethernet* adaptor cable.
- Plug in the power adaptor and press the switch on the DE10.
- Use *MobaXterm* to run a secure shell (*ssh*) to connect to the DE1-SoC. Select menu *Sessions* → *New session* → *SSH* →Fill in the basic SSH settings using Remote host as 192.168.1.113, no username, and the default SSH port is 22.
- Click *OK* and an *SSH* terminal will open.
- Login using the username: **root**, and the password: **terasic**.
- Once you are logged in, create a directory named "Lab8" to hold the Spider control code executable file.
- To transfer files between the board and the lab computer, **open a Terminal** window in *MobaXTerm* and use the **scp** commands. Or simply drag the file from the desktop to the DE10 board and drop it in the directory/file window on the left side of the screen. You can drag files in the other direction as well.

2.2 Cross Compiling from Windows to DE10-Linux

1. Create a directory on your lab computer and transfer the Spider control files from Canvas. Don't forget the Makefile.
2. Now, in the *Windows search bar* type "SoC EDS" and open the **SoC EDS 14.0 Command Shell** application (see screenshot below). This will open a new Linux-style Terminal window; use the **cd** command to navigate to the folder where you placed **the Spider code**. It

is convenient to cd to the top level directory to find your source code directory. Use the command cd c:/



3. In this terminal window, run the command **make**. This should cross-compile the file and produce an executable file name **spider**.

4. Copy hello to the DE10-SOC using scp and run it. You should rename this executable "spider1" to distinguish it from other executables you will generate as you modify the code.

2.3 Testing the Spider

When you execute the code the Spider may just sit and jitter. In that case power-cycle the robot and try again.

When the code runs successfully the left legs will stretch-out to maximum length and then the robot will slowly adjust its legs so it can stand-up. Hole the robot while this is happening and be careful not to jam you fingers in the robot's links. It is best if two people handle the robot during this phase.

You must hold the robot otherwise it may fall off the table.

After initialization press 'f' and the Spider should take a step forward. Repeated pressing of 'f' allows the spider to walk forward.

3 Modify the Spider Code

Study the Spider source code and understand how the Spider class invokes methods from the SpiderLeg class and the ServoMotor class to coordinate the legs for forward motion.

3.1 Implement Backward Walking

Study the slides on hexapod walking from Lecture 18. Using the void MoveForward() method in the Spider class. Add a method named MoveBackward() to Spider.cpp. Also modify main.cpp to add the menu choice 'b' to cause the Spider to take a step backward. Modify any other cource code or header files as needed.

Cross-compile and test your code. Don't forget to hold the Spider until its initialization is done.

3.1 Implement Turning

Study the slides on hexapod turning from Lecture 18. Add a method to the Spider class named LeftTurn(). Also modify main.cpp to add the menu choice 'l' to cause the Spider to make one rotation to the left. By one rotation I mean the equivalent to one step, this will be a partial rotation. Modify any other source code or header files as needed.

Cross-compile and test your code. Don't forget to hold the Spider until its initialization is done.

Add another method named RightTurn() to the Spider class. Also modify main.cpp to add the menu choice 'r' to cause the Spider to make one rotation step to the right. Modify any other source code or header files as needed.

Cross-compile and test your code. Don't forget to hold the Spider until its initialization is done.

3.2 Fix Initialization

Ideally, the Spider should extend its legs horizontally and then bend them so the spider stands up. As you can see the code causes the Spider to extend the left legs on one side maximally, and then slowly bend and swivel the legs to stand up. Examine this behavior carefully, especially note the rotation of the left legs vs the right legs. Modify the init() and/or the standup() methods from the Spider class to make the Spider stand-up more gracefully with less of a chance of flipping over or falling off the table. Modify any other source code or header files as needed.

Cross-compile and test your code. Don't forget to hold the Spider while testing.

4. Grading Rubric

Section	Items	Points
	Explanation of Spider Control code in its original form Classes Methods	15
Cross-compile and test initial Spider Code	Video of Spider robot walking forward	10
Move backward	Commented source code from modified files	10
	Video of Spider robot walking forward and backward	10
Turn	Commented source code from modified files	15
	Video of Spider robot walking forward and backward, and turning left and right	10
Fix initialization	Commented source code from modified files	15
	Video of Spider robot initializing gracefully, then walking forward and backward, and turning left and right	10
Lab Report Quality		5
	Total Points Lab Report	60
	Total Points Demonstrations	40
	Grand Total Points	100