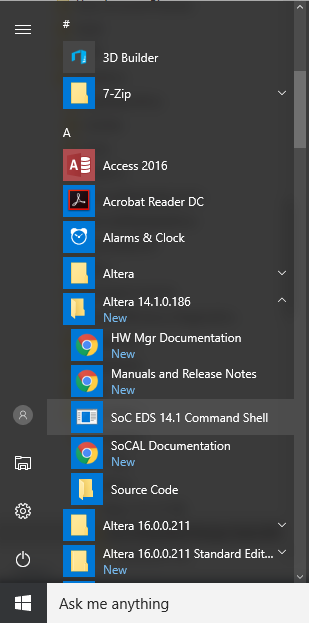
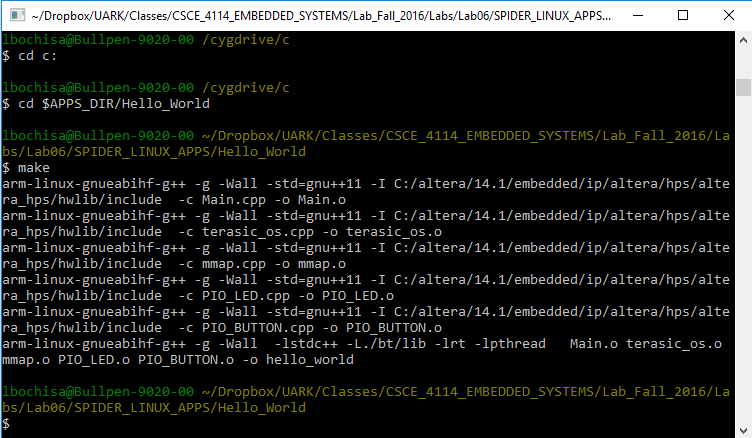
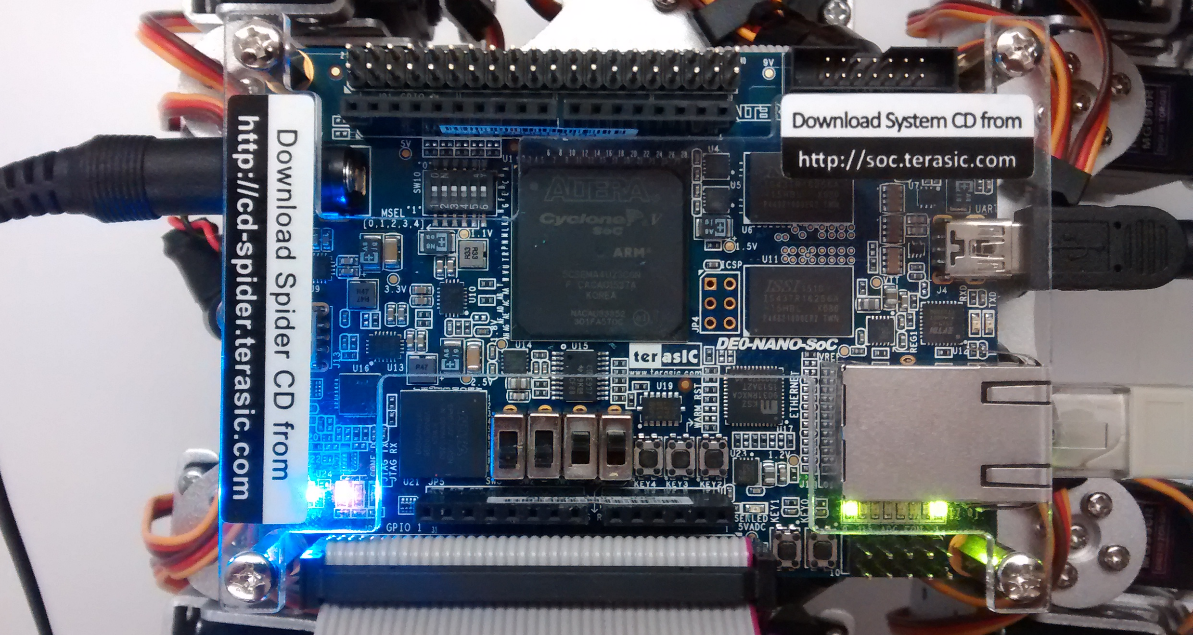
**Spider Robot Starter Kit**

**Introduction**

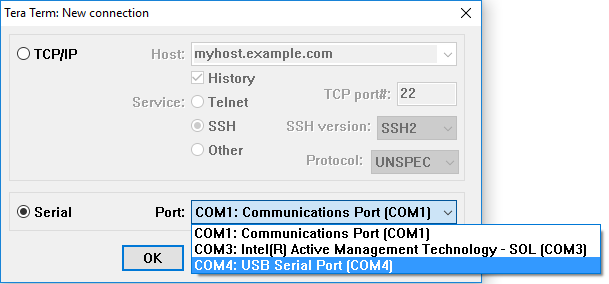
In short, there are modules in the FPGA that control the different parts of the Spider (legs, LEDs, pushbuttons, etc.) that are accessible through memory mapped registers (which we will use the embedded ARM processor in the FPGA.)

**C++ Development Tutorial**

1. Open the application "SoC EDS 14.1 Command Shell".
2. On the command shell, browse to the SpiderRobotStartKit directory and type "make" to build the applications.
3. Grab a Spider, connect the USB cable in the interface marked "UART", connect the Ethernet cable and use the **small power adapter** to turn on the board. We will be connecting our Ethernet cables to the lab’s floor ports. There are only a few open ports, so we will have to share.



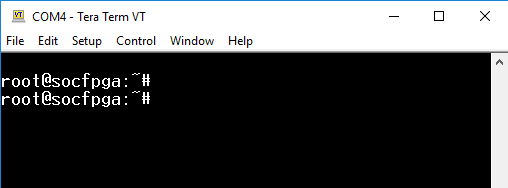
1. Open the application "Tera Term". Select "Serial" and "COM4: USB Serial Port". Note that in your system, the COM number may be different.



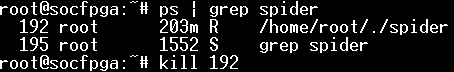
1. Click Setup->Serial Port… and select 115200 as the baud rate.
2. Restart the Spider by disconnecting and reconnecting the power supply. You should see the boot sequence of the OS on the Tera Term screen.

You can make the font bigger by clicking on Setup->Font.

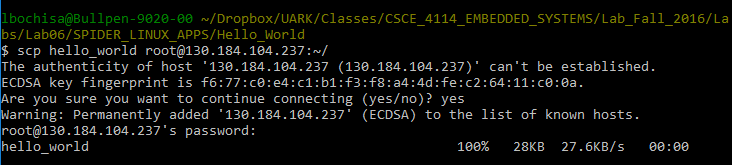
1. After the OS has finished booting, press ENTER a few times to see the terminal:



1. Change the permissions of the Spider application to no execution (to avoid interference with our app).



1. Acquire an IP address by typing "udhcpc". This step may have been performed upon booting
2. On the SoC EDS 14.1 Command Shell, use the command scp to copy the CustomSpiderProject application to the Spider (use the IP address returned by udhcpc). When prompted for a password, type "terasic".



1. Use the scp command shown above by navigating to the SpiderLinuxApplication and running with hello\_world replaced with your CustomSpiderProject.
2. Run the app. On the Tera Term screen, type: ./CustomSpiderProject

Does it work? (ignore the last LED, which is always blinking).

**Custom Application**

The Spider program code is located in the SpiderLinuxApplication directory. As you will be implementing a custom linux application, you will need to create a binary from your CPP code and transfer it to the Spider. Instead of using the 'Main.cpp' file for your code, I have created a 'Project.cpp' file that contains the essentials for starting your application the correct way. I've included all the header files from Main.cpp and added the ADC header file (See below). This will be built into binary file CustomSpiderProject. Its suggested to use Main.cpp as a reference when getting acquainted to this code base.

Utilizing the makefile, you only need to issue the 'make' command to compile your application. The above/below mentioned applications and the original Spider program are also built when issuing 'make'. Use 'make clean' to remove all compilation files.

The spider application is run after the system is booted up. Since the makefile will be making the CustomSpiderProject file for you to run your project, we don't need the spider application to run at boot. Simply change the permission of the spider binary file to not execute or just delete it. THIS IS IMPORTANT TO AVOID THE POWER QUIRK CAUSED BY THE LEGS.

Tips:

* File terasic\_os.h contains useful functions to tell time.
  + - OS\_TicksPerSecond() returns the number of ticks in 1 second, and OS\_GetTickCount() returns the current tick count of the processor. A frequency of 8 Hz, for example, has a period of OS\_TicksPerSecond()/8 ticks.
* Use only 7 LEDs, since the last one is always blinking to demonstrate that the OS is running.
* The pushbuttons are '1' when **NOT** pressed and '0' when pressed.
* To test if pushbutton 0 was pressed, you can do:
  + - If ((BUTTON\_PIO.GetBUTTON() & 0x01) == 0)
* To test if pushbutton 1 was pressed, you can do:
  + - If ((BUTTON\_PIO.GetBUTTON() & 0x02) == 0)
* To test both pushbuttons were pressed, you can do:
  + - If ((BUTTON\_PIO.GetBUTTON() & 0x03) == 0)