Lab Assignment 6

Controlling the Robotic Arm by Hardware

Zhenming Yang, Connor McEleney

[yang.zhenm@northeastern.edu](mailto:yang.zhenm@northeastern.edu), [mceleney.c@northeastern.edu](mailto:mceleney.c@northeastern.edu)

Instructor: Prof. Zagieboylo

[d.zagieboylo@northeastern.edu](mailto:d.zagieboylo@northeastern.edu)

Due Date: October 26th, 2023

Submit Date: October 26th, 2023

**Abstract**

In this lab, we wrote three c++ programs on the DE1-SoC board’s Linux system using nano. The first program is a Hello-World program, and the second/third programs use selection sort to sort an array of integers and strings.

# Introduction

The DE1-SoC operates on a variant of the Linux Ubuntu distribution known as DE1-SoC-UP Linux. While it's a somewhat reduced version of Ubuntu, it retains many features found in a complete distribution of this OS. It functions on the ARM processor and can accommodate the running of personalized C or C++ applications [1].

# Lab Setup

## Pre-Lab

A video on the selection sort algorithm was studied.

## Equipment

DE1-SoC:

* The DE1-SoC is a hardware design platform built around the Altera System-on-Chip (SoC) FPGA. The DE1-SoC is designed for experiments on computer organization and embedded systems. It includes embedded processors, memory, audio and video devices, and some simple I/O peripherals.

# Results and Analysis

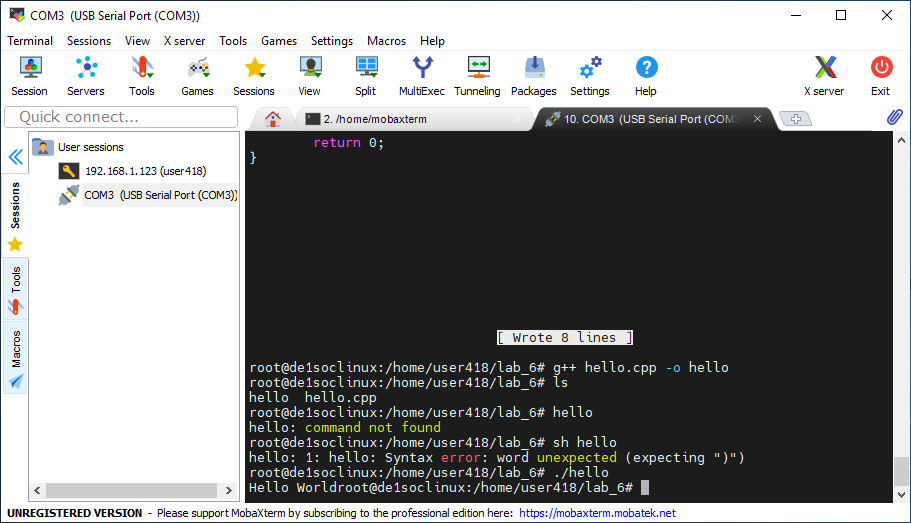
**Results**

## Part 1: Connecting to the DE1-SoC

We first established a connection between the DE1-SoC board with our computer. We attempted to connect to our board through ethernet with SSH. However, our computer can’t find the IP address of the board, so we decided to go with the serial connection. We followed the guide and established a serial connection using the micro usb port.

## Part 2: Hello-world on the DE1-SoCs

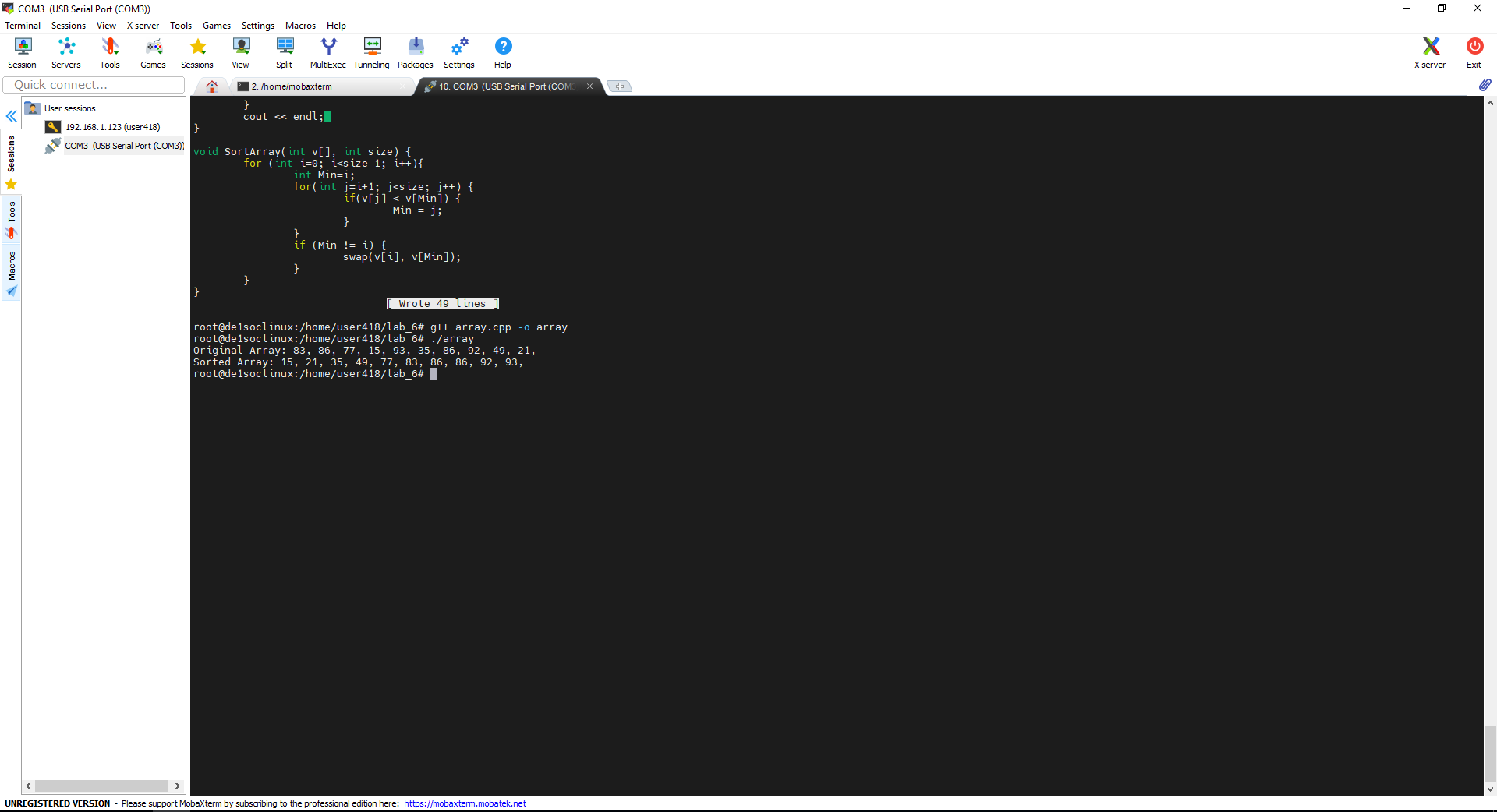
We then created a “hello.cpp” under our lab session’s directory. We opened the file using nano and coded the program. We then used the g++ compile to compile the cpp code into a executable. After running the executable, **we see the result “Hello World” as intended** (Figure 1).



**Figure 1**: The Terminal Output of the Hello-World program. Noted the correct output of "Hello World" appears in the terminal at the last line.

## Part 3: Sorting an array of integers

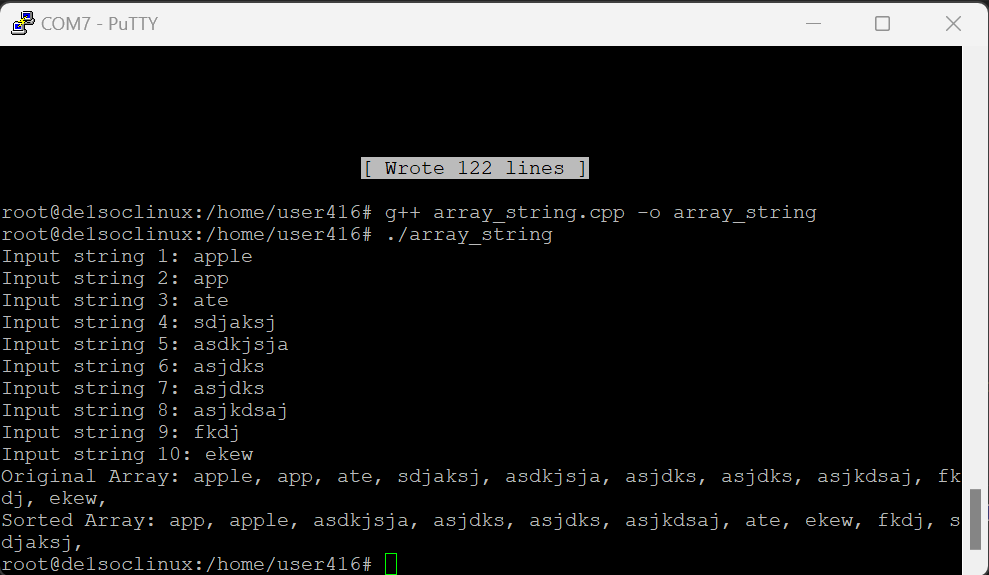
We then coded the selection sort algorithm to sort an array of random integers. We first created the list using the function RamdomArray() to generate an array of random integers between 0 and 99 using standard library rand(). Then, we print the original array with the function PrintArray() to show us the unsorted array of integers. Then we sort the array using SortArray() which implements the selection sort algorithm**. It checks each number from start to finish with every number after it and puts the lowest number to the front position**. At last, the program PrintArray() again with the newly sorted array to show us the final result. We tested our function, and it gives us the result (Figure 2). Noted the new array is sorted whereas the original array is totally random.



**Figure 2**: The terminal output of the integer array sorting program. Noted the difference between the original array and the sorted array.

## Part 3: Sorting an array of strings

We then modified the integer sorting program to make a string sorting program. Instead of generating an array of random integers, the new program prompts the user to input the string to the original array. The program first runs the **InputArray() function which loop through the empty array and prompt the user for a input** (Figure 3). Then the array is printed using the same PrintArray() function. The SortArray() function is slightly change to fit the new data type: instead of using the build in “<” comparator, **we created our own StringComparator() to compare the strings. The new comparator is very simple. It first checks the length of both strings and loops through both strings with the length of the shorter strings.** Each character we loop through is first converted to lower case and compared with each other. If the longer string contains the shorter string, the shorter string is “smaller” which will out put true at the end of the loop. We tested our program, and it yields promising results (Figure 3).



**Figure 3**: The terminal output of the sorting program for strings. Noted "app" comes before "apple" and the rest goes in alphabetical order.

**Analysis**

This lab is an introduction to programming on the DE1\_SoC board. We first had some challenges with the connection with the board through ssh but we then switched to using serial to communicate from our computer. The sorting algorithm was designed for us to learn to us abstraction to minimize the amount of work needed to do. Instead of changing the sorting algorithm to sort a new data type, we just have to define a new comparator and it algorithm still works with the new comparator.

# Conclusion

This lab introduced us to the concept of embedded programming. We learned how to connect to the DE1-Soc through SSH, although due to IP address errors we connected through serial. Practices for embedded coding were employed, such as not importing full libraries when needing just a few functions from them to conserve what may be very limited memory in some embedded systems. We were also introduced to the selection sort algorithm, where you increment through the list, combing through the values and swapping the lowest number ahead of the current with the current index and then increments. By coding the sorting algorithm ourselves we obtained a deeper understanding of the behind the scenes of some of the included functions in the C library.

# References

1. DE1-SoC User Manual