Senior Design

Spring Week 6 Report

Interactive Embedded Systems Learning using the Prairie Learn framework

2/21 - 2/28
Faculty Advisor: Phillip Jones

Team Members:
- Ben Stroup
- Caden Last
- Jack Kennedy - Git Team Lead
- Emmanuel Paz - Server Lead
- Ryan Dela Merced - Project Manager
- Cody Prochaska - Technical Team Lead
- Ryan Bumann

Summary of Progress this Week:
- Continued to work on all aspects of our project

Questions:

Team Member	Contributions	Hours	Total Hrs
Ben Stroup	Researched “invalid: not gradeable” error and found work around solution. Applied solution to homework’s 1-5 and 11. Matching and order blocks not able to work around. Started revising earlier homework’s.	5	70
Caden Last		5	61
Jack Kennedy		3	51
Emmanuel Paz	Created video authentication on doc server. Helped Cody figure out autograding assembly (in additional notes)	6	69
Ryan Dela Merced		6	54
Cody Prochaska	Talked with Manny and found a promising method for autograding assembly	7	62
Ryan Bumann	Experimented way to make dynamic graphics faster with svg’s. Worked on autograder, specifically how to move data between all the different types of files, Python, JS, C. Worked on	14	75

	documentation for setting up external autograding for production and local.		

Plan for Next Week:

Additional Information:
[PrairieLearn Homework Tracking]

assembly autograding steps:

required packages:

- gcc-arm-linux-gnueabihf
- gcc-arm-linux-gnueabi
- binutils-arm-linux-gnueabi

take student file "assembly_file.s"

```
arm-linux-gnueabi-as -o first.o assembly_file.s      (Create object file from assembly code)
arm-linux-gnueabi-gcc -o first first.o               (Create binary from object file)
qemu-arm -L /usr/arm-linux-gnueabi/ ./first          (Run ARM binary using QEMU)
```

plan for going forward:

- install required packages during docker container setup to allow these commands to be run
- clean docker container after every submission similar to our c autograder
- create dynamic assembly questions
- look at specific questions and figure out how to best approach autograding each (print statements comparing registers/memory)
- possibly run the same assembly code on multiple randomized inputs to prove it's working correctly, for example a question might say "find largest value in an array of size x" we only want to have the autograder check the final memory value that the student stores. The student could just store a hard coded value of what they know to be the largest given some inputs, so we could randomize the inputs to prove it's working right.