SE/EE/CPR E/CYB E 492 – sdmay24-33
PrairieLearn Senior Design Team
Bi-Weekly Report 4

Feb 24 - March 30
Faculty Advisors: Phillip Jones

Team Members:

Chris Costa - Auto-Drawing
Matt Graham - Emulator
Mitch Hudson - Technical Lead - ARM Assembly Auto grading
Carter Murawski - Note Taker
Tyler Weberski - Project Manager, Auto-Drawing
Andrew Winters - *Role not yet assigned*

Summary for Progress

Week 1

- Created ARM autograder
- Working on HW12
- Continued peer reviews and getting HWs ready for TA review
- Implementing Prairielearns recommended drawing technique

Week 2

- Finished up randomization for H5_Q2
- Finished up randomization for H8_Q1
- Write-up for PrairieLearn Auto-drawing created
- Finished homework 12 questions 2-5
- Wrote bare-metal ARM auto-grader
- Made write-up for bare-metal ARM auto-grader
- HW4 and 7 are ready for TA review
- Updated question names / assessment formatting for hw1-3
- Got HW4 and 7 ready for external review

Week 3

- Completed Midterm Peer Review class assignment
- Addressed emulator comments from the previous meeting
    - UARTSR/UARTECR(missing)
        - RP2040 does have this register, but the developers of the emulator chose not to implement it.
    - UARTLCRH vs UARTLCR_H
        - Both are the line-control registers that serialize parameters(data length, stop bits, and parity)
    - UARTCTL vs UARTTCR
        - Control register on both TM4 and RP2040
    - UARTIM vs UARTIMSC
        - Interrupt Mask register for both.
    - UARTCC(missing)

        - TM4 Clock configuration
        - There does not appear to be a clock register in the UART for the RP2040
- Reviewed homework 10
- HW 12 finished - waiting on peer review **(See note below!)**

Week 4
- Completed midterm peer review assignment
- Reviewed homework 12

Week 5
- Configured emulator environment and added to git
- Set up the emulator to be able to run any standard C file
- Completed emulator usage writeup **(see the end of this document here)**
- Wrote homework 8
- Wrote homework 6
- Reviewed homework 8
- Reviewed homework 6
- Reviewed homework 11

Pending Issues
NA

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Chris Costa	**Week 1**:Started working on hw 9 comments and review, worked on hw 5, and researched drawing in prairieLearn **Week 2**:Worked on H5_Q1 creating the drawing and and experimenting with inputs, started working on HW9_Q2 question with drawing **Week 3**: NA **Week 4**:Worked on and completed the midterm peer review for class, worked on homework 5 question 1 and started fixing homework 9 **Week 5**:worked on implementing hw5 q1 and getting better interactiveness, continued editing hw9	28	95
Matt Graham	**Week 1**:Reviewed homework 9, made changes from peer review of homework 5, researched pi pico emulator **Week 2**:Completed revisions for homeworks 4 and 7, refactored C autograder questions in homeowrks 4 and 7,developed pi pico comparison to CPR E 288 microcontroller started experimenting with the pi pico emulator	18	98

	Week 3: Reviewed homework 10, updated the website, completed the work progress section of the midterm peer review assignment, and worked to address the emulator comments from the last meeting **Week 4:** Completed midterm peer review assignment, reviewed homework 12, looked more into emulator **Week 5:** Got the emulator environment configured and added it to git under the pico-emulator-dev branch, set up the emulator to be able to run any standard C file, and created a writeup of how to use the emulator once you have it pulled from the repository **(see the end of this document here)**		
Mitch Hudson	**Week 1**:Wrote assembly autograder using QEMU Started process of adding ARM syntax highlighting to ACE editor Started process of adding ARM autograder to PrairieLearn repo Wrote HW12 question 2 Wrote assembly autograding writeup **Week 2**:Completed code for bare-metal ARM grader Made writeup for bare-metal ARM grader Cleaned up HW7 and responded to some feedback Finished hw12 q2-5 Updated question names and formatting for hw1-3 Pushed hw 4 and 7 to master for TA review **Week 3:** Wrote detailed design slides for Midterm peer review Fixed login bug on Chrome and Edge Finished HW 12 - waiting on peer review **(See note below)** Responded to HW 10 peer review Updated HW 11 with instructions **Week 4:** Finished midterm peer review, responded to some feedback for hw 12, waiting on peer reviews for 10-12 **Week 5:** Wrote hw 6 and 8 Responded to peer reviews for hw 6, 8, 11 Linked PrairieLearn to Canvas environment Started working on QEMU emulation platform for Tiva TM4C123G microcontroller Started writing Python system for handling configuration register setup	65	235

Carter Murawski	**Week 1**:Researched pi pico **Week 2**: developed pi pico comparison to CPR E 288 microcontroller started experimenting with the pi pico emulator **Week 3:** Finished the intro section of the midterm peer review assignment and worked to address the emulator comments from the last meeting **Week 4:** Finished the midterm peer review recording and looked into the emulator. **Week 5:** Worked on emulator environment, it can now run C files and helped Matt with the write up about how to set it up	25	88
Tyler Weberski	**Week 1**:wrote hw5-q2 over with Prairie Learns drawing method that they recommend. Completely randomized all inputs for that drawing, and can identify inputs, outputs, and ports associated with the problem **Week 2**:I have almost finished H5_Q2 with the randomization and autograder (1 note about question for part b grading in comments/extended discussion). Randomized H8_Q1 picture, and did the writeup for the PrairieLearn drawing **Week 3:** Wrote the challenges and solutions slides for the midterm peer review assignment we had. Working towards integrating the UART GPIO code into PrairieLearn still **Week 4:** Worked on midterm peer review, as well as overlooked GPIO code and started implementing with HW5 Q2, starting to write own question code **Week 5:** Reviewed homeworks 6, 8, and 11. On top of this continued working with GPIO code, making my own code work for the problem, still working on integrating	19	90
Andrew Winters	**Week 1**:wrote HW12 question 1, looked into starting 3 **Week 2**: NA **Week 3:** Wrote the demonstration slides for the Midterm per review. Will be reviewing hw 10 in the next couple of days **Week 4:** did the midterm peer review, I am traveling with the men's basketball team as a part of the Pep band so I won't be at the meeting this week and potentially future meeting **Week 5:** NA	9	77

Comments and Extended Discussion

Week 1

- TA review document link: https://docs.google.com/document/d/1U_E2vA8zJmP2FQb-ZPSnxSUaQaosqQoOGCLGKJOyeM0/edit
- Course content access roles documentation: https://prairielearn.readthedocs.io/en/latest/course/#course-content-access-roles

Week 2

- With part 2 of H5_Q2, how do we want to approach until we figure out UART, should I just wait, or change what previous group currently has for current work, so that we can push out Homework 5 for TA review
- For H8_Q1, I had an idea about randomizing with the code section, specifically using possibly SS1-3 as well compared to just SS0
- For scoring on the different questions, Can we just get an update on how each question should be scored points wise for you? (No point values set in homeworks so far)

Week 3

- HW 12 Question 6 Note: The Cortex M3 processor used by the LM3 board only supports Thumb and Thumb-2 instructions. This poses issues with several of the questions for ARM, but most notably with the loops, where you use LDR Rd, [Rt], #n which is not a supported addressing mode in Thumb.

Week 4

- Last week was spring break, no work was required over spring break
- We need to know the point values for questions / homeworks, but after that and peer reviews, all of the homeworks are going to be completed it seems

Week 5

- NA

Plans for Coming Week

- Work on getting the emulator working with more advanced C files, such as UART
- Continue to work through peer reviews and homework development
- Finish HW 5 and 9 peer reviews
- Work on QEMU emulation integration
- Work on Python register configuration scripts

Summary of Weekly Advisor Meeting

Week 1

- Come up with and sort module names in PL
- What will a manual grader see on their end
- Have HW 1,2 and 3 ready for review
- Setting up who is moving forward with different technologies

Week 2
- Add random generation and autograding to HW 12
- Contact previous group for autodrawing
- Change randomization from I/O to ports and wires for HW 4 Q2
- HW5-Q1, fully randomize values in the PCTL and AFSEL registers
    - See if we can have users add text to a drawing
- Change value ranges for HW7 Q6

Week 3
- Document the image drawing functionality
- Does Chrome cause issues logging into Okta?
- Add status register to UART
- Add UARTCC register turns clock on
- Check missnamed registers
- Try to run 288 homework's

Week 4
- There was no advisor meeting last week during spring break

Week 5
- Add lightning talks to website(slides with audio)
- Gave first demo of PL implementation
- Get last years questions
- Keep moving forward with PL question
- Improve communication and come to next meeting with more of a plan
- Try to find different meeting time

Midterm Feedback
Most of the feedback we received focused around the project's technical side. Specifically, a lot of the feedback was worried about potential vulnerabilities and security issues with the handling of student grades. The other really big point of feedback was a lack of demonstrations, diagrams, and documentation for the student side user experience. They also brought up concerns with the stability of the system and the flexibility of the auto grading. Finally, almost every group member mentioned that getting feedback on the system from current or former CPRE 288 students is crucial.

Some of the biggest questions were surrounding the platform's potential use in other courses, and the expansion of the platform used. They also asked a lot of questions about the UI/UX and resources for students using the system. Finally, they asked some questions about the ease of use on the professor's side and security issues.

The biggest insight we gained from this was seeing where our demonstrations lacked detail and what parts of the system needed more documentation and resources. To address this, we plan to dedicate the last few weeks of the semester to building documentation for both students and professors, as well as documentation marking the progress of the project for future senior design groups. This documentation will also help us better show our project to the final panel, and force us to create more demonstrations for the various aspects of our design. Finally, we plan to create some new diagrams to help better explain the more complicated components of the design, as well as a higher level diagram to show the general flow of the user experience.

How to Setup The Emulator and Run Any C File

- Open Terminal
- Run “cd ~”
- Git clone https://git.ece.iastate.edu/sd/sdmay24-33.git
- Run “cd sdmay24-33”
- Run “git pull”
- Run “git fetch”
- Run “git checkout “pico-emulator-dev”
- Run “sudo apt update”
- Run “sudo apt install”
- Run “sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi build-essential”
- Run “sudo nano ~/.bashrc”
- Add the following line to the end of the file:
    - “export PICO_SDK_PATH=/home/*user*/sdmay24-33/emulator/pico-sdk”
    - Replace “*user*” with the account name you are using
    - Note: this command assumes you have cloned the sdmay24-33 directory in your user’s home directory as specified above. If this is not the case, modify the path accordingly
- Open Visual Studio Code
- Select “File”
- Select “Open Folder”
- Choose the “sdmay24-33/emulator/build-hex-files” directory from your filesystem
- Here, you can edit the “build-hex-files.c” file to contain any C code you wish to run
- Select “Build” from the bottom of the screen (gear icon)
- You will see a prompt for “Select a kit for build-hex-files”
- Select “GCC 10.3.1 arm-none-eabi” from the dropdown