



PrairieLearn

sdmay24-33

Project Manager: Tyler Weberski

Technical Lead: Mitch Hudson

Construction: Andrew Winters

Consultant: Matt Graham

Quality Assurance: Chris Costa

Scrum Master: Carter Murawski

Client and Advisor: Dr. Phillip Jones

Introduction - PrairieLearn

- Online assessment and learning system for any class
- Interactive learning environment for students
- Easy auto grading system for instructors
- Open-source project

Introduction - Our project within PrairieLearn

- CprE 288 Embedded Systems
- Homeworks
- Auto Grader
- Setting up a server
- Emulator development

Functional Requirements

- Questions will be autograded as much as possible
- Implement all current questions and assessments
- Organize students into course sections
- Connect with ISU Okta and Canvas
- Integrate an emulator for running student code (**constraint**)

Non-Functional Requirements

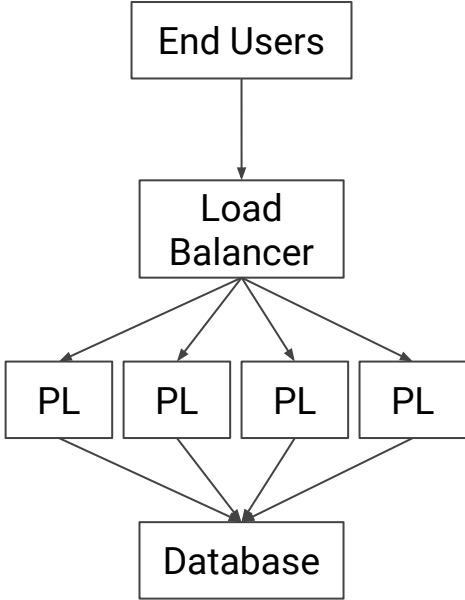
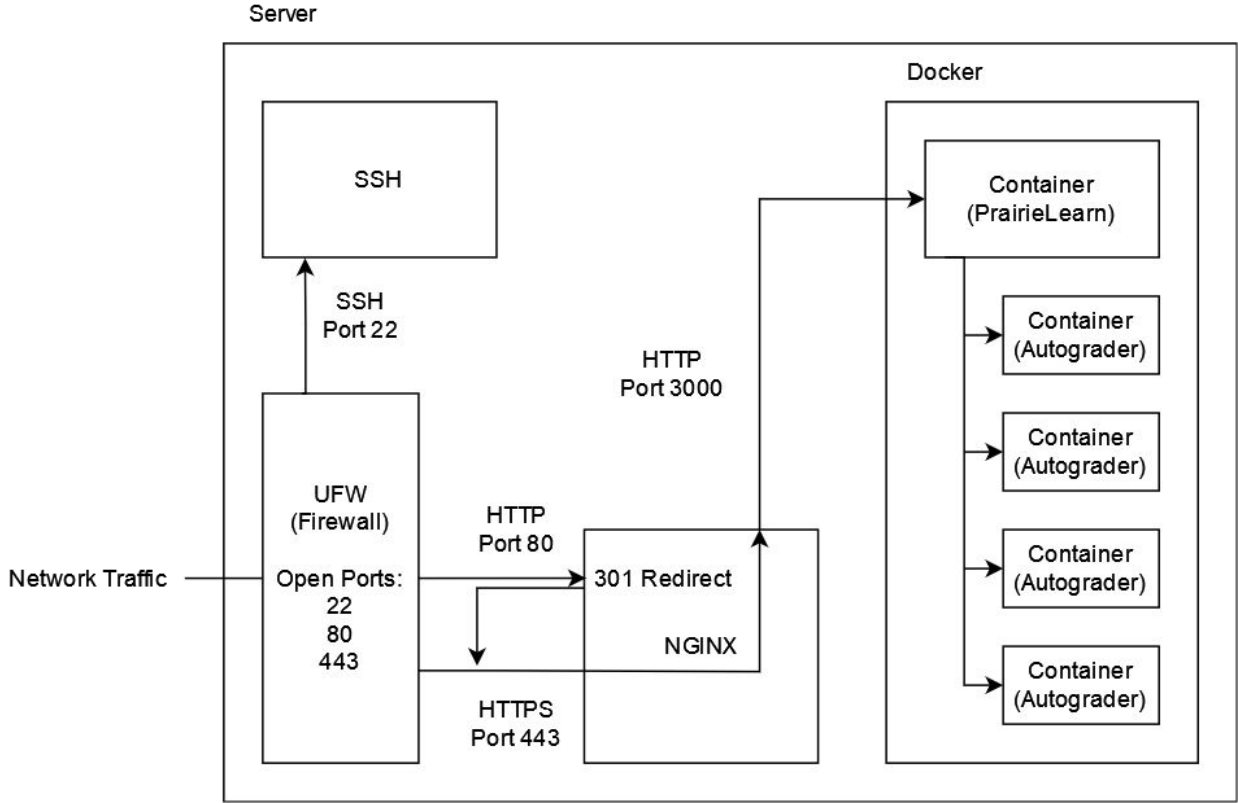
- Easy question/assessment updating
- Fast Assignment loading time
- Questions are intuitive and comprehensible
- Grading is efficient with quick feedback
- Feedback must be understandable and helpful to students
- Create quality documentation of our work

Frontend Demo

- Live Demo Showing:
 - Student and Developer View
 - Randomization within each question
 - C compiler and Grader
 - ARM assembly compiler and Grader

Assessments		
	Available credit	Score
Intro to Embedded Systems		
HW1 Embedded Systems Basics	100% until 23:59, Thu, Apr 1 ●	0%
Intro to Embedded C		
HW2 Embedded C Basics	100% until 23:59, Thu, Apr 1 ●	Not started
HW3 TM4C123G and C Data Structures	100% until 23:59, Thu, Apr 1 ●	0%
HW4 C Practice and Microcontroller GPIO	100% until 23:59, Thu, Apr 1 ●	Not started
Low Level Embedded C		
HW7 UART and ADC	100% until 23:59, Thu, Apr 1 ●	0%
HW10 Timers	100% until 23:59, Thu, Apr 1 ●	Not started
ARM Assembly		
HW11 ARM and Assembly Basics	100% until 23:59, Thu, Apr 1 ●	Not started
HW12 Assembly Programming	100% until 23:59, Thu, Apr 1 ●	Not started

Backend Final Design



Writing Autograded Questions

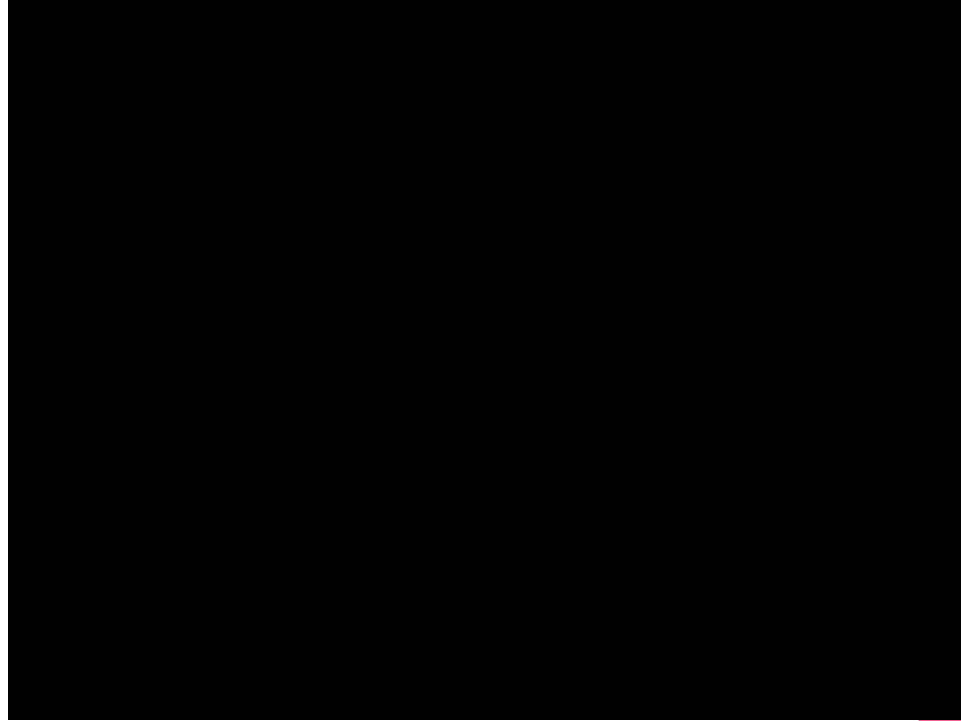
```
191 int main() {
192     /* Saves stdout in a new file descriptor */
193     int realStdoutNo = dup(STDOUT_FILENO);
194     FILE *realStdout = fdopen(realStdoutNo, "w");
195
196     /* Switches stdout/stderr to point to different file descriptor to
197     * avoid students passing code by printing the correct result. */
198     int devNull = open("/dev/null", O_WRONLY);
199     dup2(devNull, STDOUT_FILENO);
200     dup2(devNull, STDERR_FILENO);
201
202     int test;
203     int pin;
204
205     scanf("%d", &test);
206     scanf("%d", &pin);
207     switch(test) {
208         case 1: { // Test question 1
209             test_init(realStdout, pin);
210             break;
211         }
212         case 2: { // Test question 2
213             test_send(realStdout, pin);
214             break;
215         }
216     }
217
218     if (errno) {
219         fprintf(realStdout, "ERRNO: %d\n", errno);
220     }
221     return 0;
222 }
```

```
void c_entry() {
    int as[NUM_INPUTS];
    int bs[NUM_INPUTS];
    int *ptr0 = as;
    int *ptr1 = bs;
    for (int i = 0; i < NUM_INPUTS; i++) {
        scan_uart0("%d", ptr0);
        scan_uart0("%d", ptr1);
        ptr0++;
        ptr1++;
    }

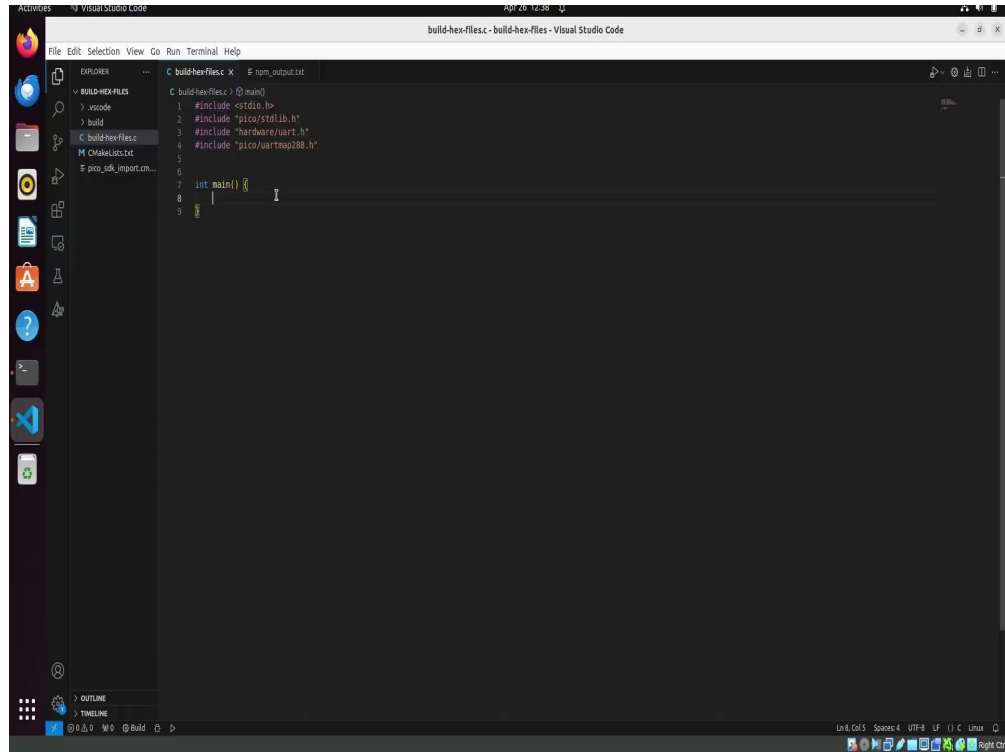
    for (int i = 0; i < NUM_INPUTS; i++) {
        a = as[i] % 0xff;
        b = bs[i] % 0xff;
        _multiplication();
        print_uart0("%d: a=%d\n", i, (int)a);
    }

    _exit_qemu(); // exits QEMU cleanly
}
```


QEMU Emulator Demo



Pi Pico Emulator Demo



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays a project structure with folders for 'BUILD-HEX-FILES', 'vscode', 'build', 'C: build-hex-files', 'M: MakeLists.txt', and 'F: pico_sdk_import.cm...'. The main editor window is titled 'build-hex-files - build-hex-files - Visual Studio Code' and shows a C source file named 'main.c'. The code in the editor is as follows:

```
C: build-hex-files \> main()
1 #include <stdio.h>
2 #include "pico/stdlib.h"
3 #include "hardware/uart.h"
4 #include "pico/uarttrap288.h"
5
6
7 int main()
8 |
9 }
```

The status bar at the bottom of the editor indicates the current file encoding as 'LnB, GdS', the workspace as 'Spocsc4', the file encoding as 'UTF-8', and the file type as 'LF'. The system tray at the bottom right shows the date and time as 'Apr 26 12:38' and the system as 'Linux'.

Challenges and Solutions

- Many software engineering challenges
- Transitioned into embedded systems problem solving
- Improved on the previous learning experience
- Our emulators replaced the hardware device with a virtual version

Design Complexity

- Integrating Okta and Google SSO
- Auto Drawing Feature
- Emulator Development
- Auto Generation of Device Configuration
- Infrastructure Development

Test Completion

- Unit Testing
 - Individual Questions, Emulator Grading, Docker Containers
- Interface Testing
 - File Types, Assignments
- System Testing
 - Repetitive full assignments testing
- Security Testing
 - Firewall, MFA, Data Encryption, Okta
- Black & White Box Testing
 - Internal & External
 - Acceptance Testing

Milestones Reached

- Fall 2023:
 - Developed Interactive Questions
 - Server Establishment and Configuration
- Spring 2024:
 - Developed Unfinished Problems
 - Improved Auto Grader
 - Implemented Emulator
 - Completed Documentation for Created Content
 - Website Well Developed for Future Use

Goals for Future Groups

- Tweak homeworks after TA feedback
- Online Lab Environment
- Create Documentation for New Implementations

Conclusions

- Accomplished our goals
 - Homeworks completed
 - Emulator implemented
 - Autograding successfully

- Prepared next years group for success
 - Plenty of documentation