Interactive Embedded Systems Learning using the Prairie Learn Framework

Team: sdmay25-33

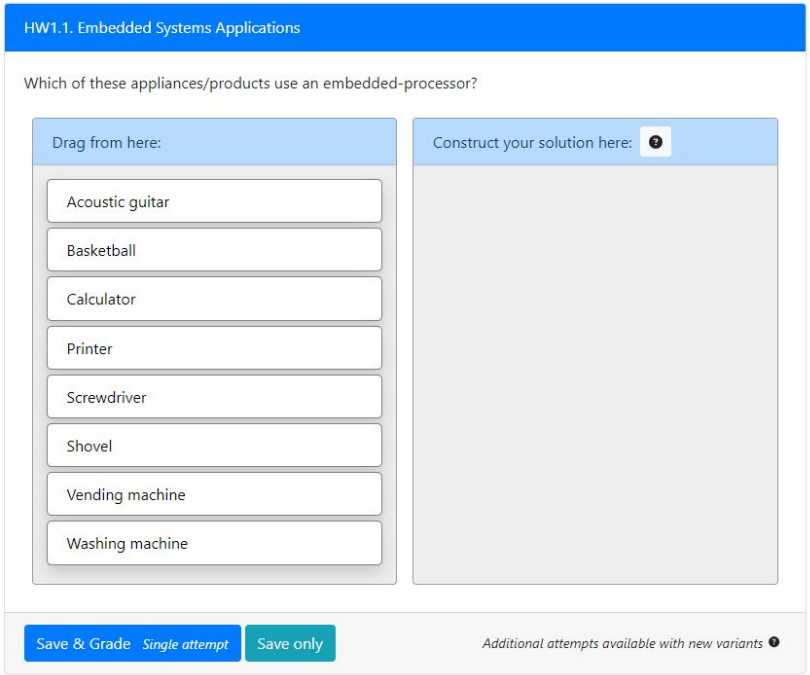
Project Manager: Caden Otis
Consultant: Devin Alamsya
Technical Lead: Justin Cano
Quality Assurance: Joseph Krejchi
Notetaker: Rachel Druce-Hoffman

Client and Advisor: Phillip Jones

Project Plan

Project Overview

- Create an interactive application for CPRE 2880 students to better understand the concepts
  - HWs and quizzes
  - Randomized questions and autograding
  - Use emulation tools to simulate microcontrollers
  - Potentially have an emulated Cybot robot interface
- PrairieLearn framework to host the application
- Utilize Python, HTML, C and other programming languages
- Hope to inspire other professors to build similar interactive tools for their students



Problem Statement

- Students don’t get enough practice of concepts
  - Little feedback on Canvas HW submissions
- Not always availability to practice programming on the microcontroller in the lab
- Limited time to meet with Professor and TAs
  - Lab, class, office hours
- Limited capabilities with Canvas platform

Functional Requirements

- All homeworks should be implemented
  - Code of each homework should also be documented for future development
- Most questions should be autograded
  - Includes student-written coding segments
- All questions should be randomized for unlimited practice
  - As many parameters within the problem should be randomized as possible

Non-functional Requirements

- USER EXPERIENCE: New question types designed and implemented focusing on interactiveness
- USER EXPERIENCE: Questions formatted to be easy for the user to understand and interact with
- RESOURCE: Implement the Virtual Cybot/Emulated Cybot interface so students can test their code.
- RESOURCE: Documentation written about each aspect of our implementation
    - Allow for continued development
    - Tutorials for other classes setting up PL
- AESTHETIC: No bugs/typos

What Makes Our Project Unique

- Tailored to just CPRE 2880 content
- Uses emulation tools to create unique, engaging questions
    - Simulates real hardware (LM3S6965 and TM4C123GH6PM boards)
- In complete control of what's created
- Free
- Immediate Feedback

zyBooks

coursera

Potential Risks and Mitigation

- Team member is not completing their work or showing up to meetings
  - Communicate with team member and advisor
  - Give reminder of team contract
- Answer Leakage
  - Ensure correct answers aren't accessible until after the student has submitted their own answer
- Product may perform worse than Canvas in beta testing
  - Adapt using student feedback to ensure that the end product performs better than other solutions

Resource/Cost Estimate

- Total development and use of our project is free
  - Use of PrairieLearn framework is free for development
  - Hosting PrairieLearn is free via ISU VM server
  - No cost for creating custom emulation tools

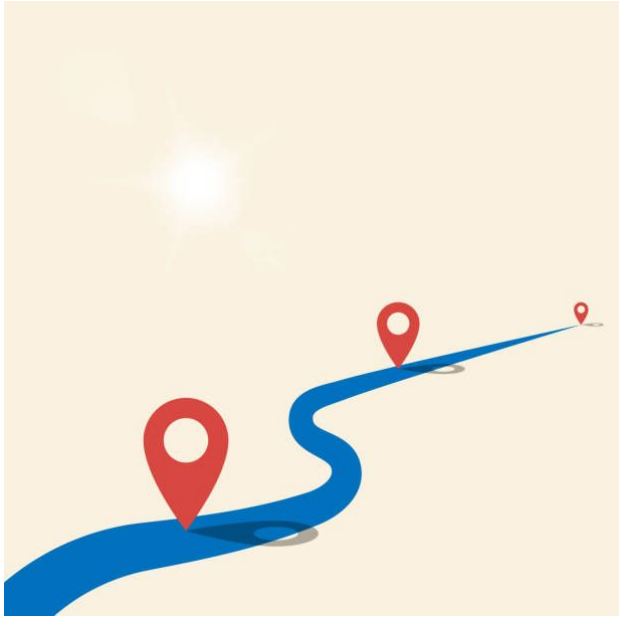
Project Milestones

Fall Semester:

- Complete partial Beta version of the application
    - Will be used for CPRE 2880 course in Spring 2025 semester
- HWs 1-6 implemented into Prairie Learn

Spring Semester:

- Implement all homeworks
- Update and complete documentation
- Full Canvas integration
- User feedback collection and improvements based on that feedback



Project Schedule/Timeline

Server Setup:
Get Prairie Learn server initialized
Get ASW to sign PrairieLearn Server Certificate for SSL
Get ISU Integration with Okta for student authentication

Begin Question Implementation:
Review CPRE 2880 concepts
Learn how to use PrairieLearn
Begin coding questions
Learn how to use Cybot emulator
Learn how to use student code autograder
Learn how to use the emulation tools that are already incorporated
Finish implementing questions for HW 9
Finish implementing questions for HW 12

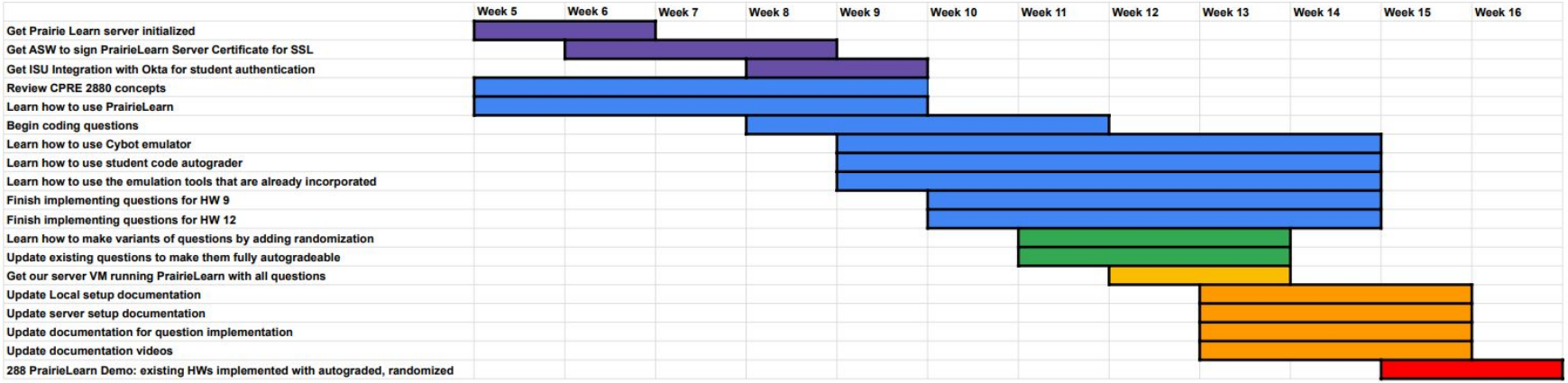
Improve Questions:
Learn how to make variants of questions by adding randomization
Update existing questions to make them fully autogradeable

Course Functional on VM:
Get our server VM running PrairieLearn with all questions

Documentation:
Update Local setup documentation
Update server setup documentation
Update documentation for question implementation
Update documentation videos

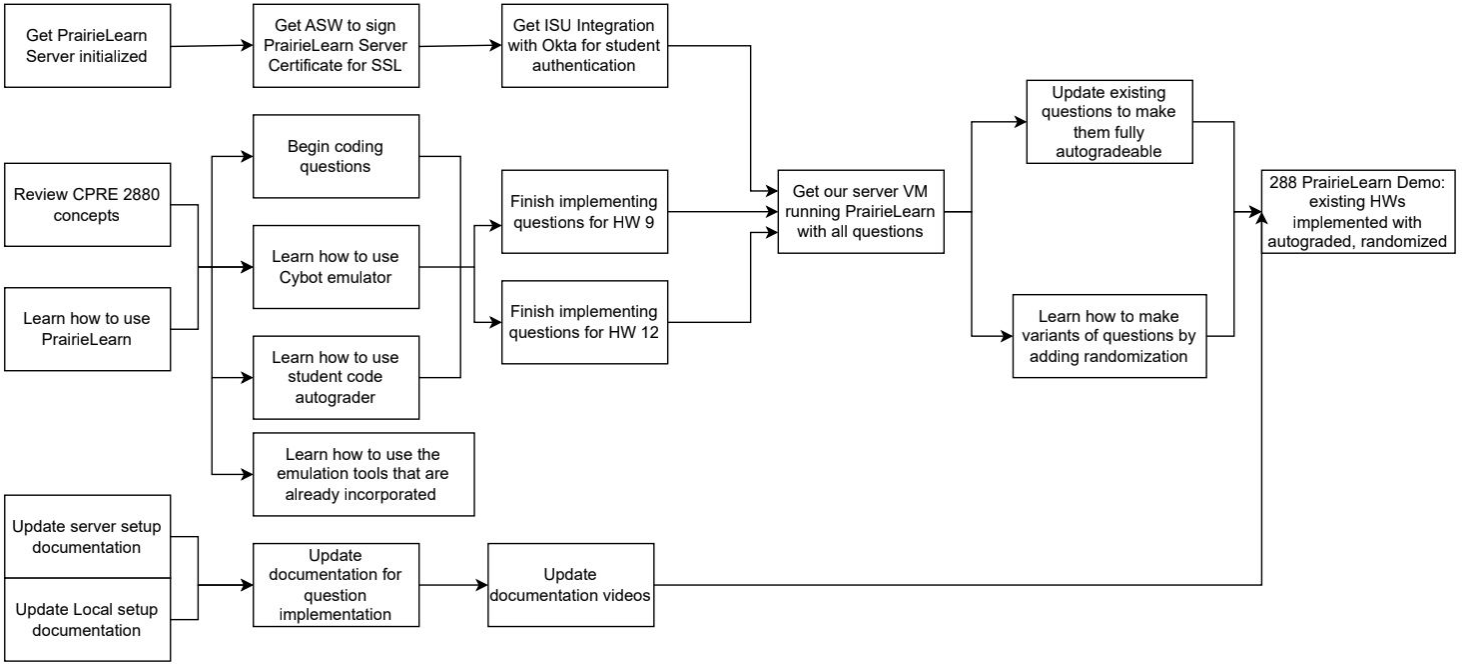
288 PrairieLearn Demo: existing HWs implemented with autograded, randomized

Project Schedule/Timeline

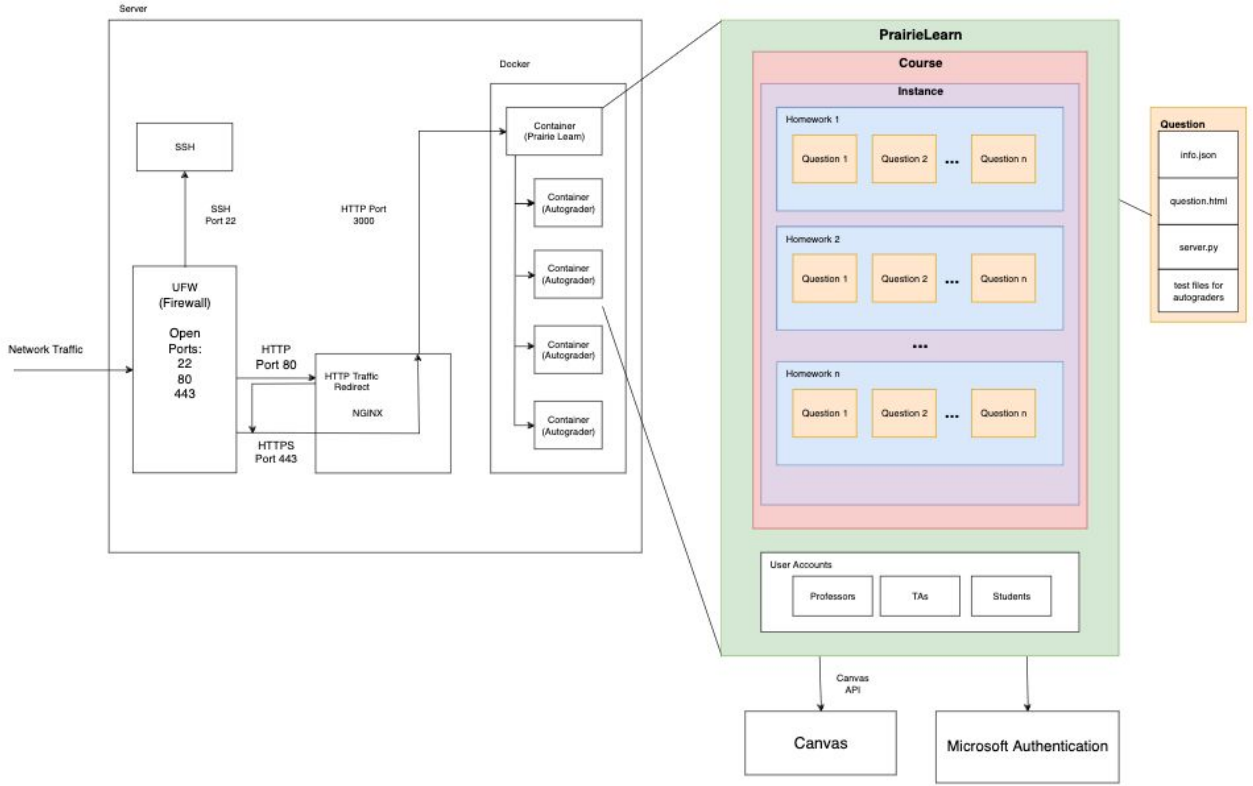


System Design

Functional Decomposition



Detailed Design



Software Platforms Used

- PrairieLearn Framework
- Git
- Linux
- Docker
- ISU VM

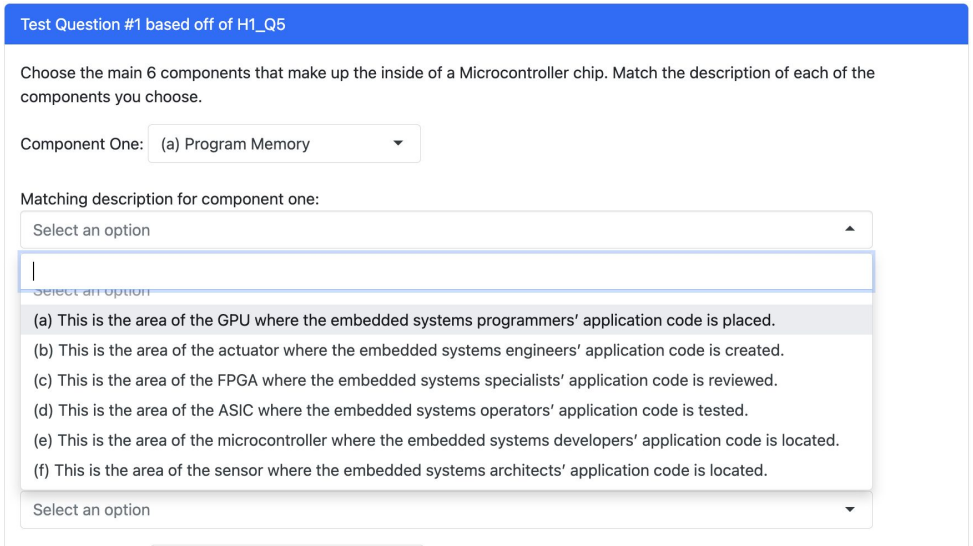
Test Plan

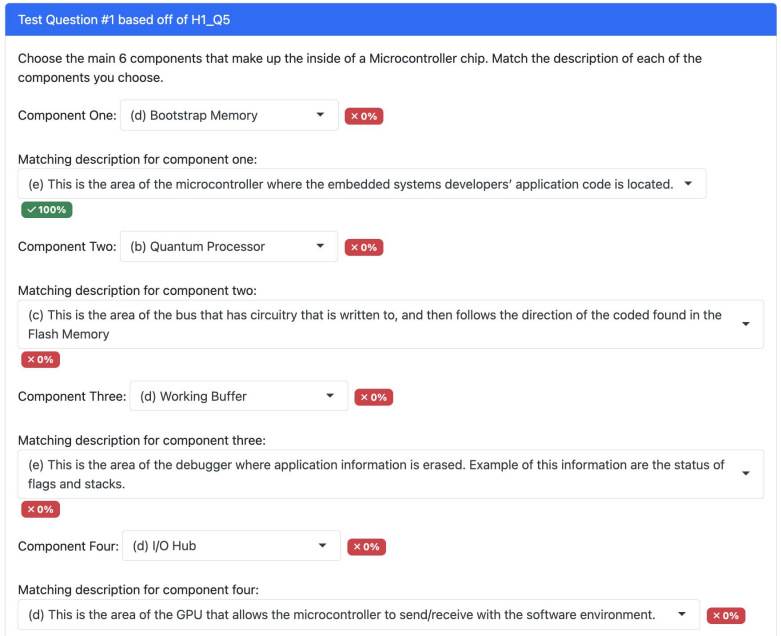
- Unit testing
  - Verify individual homework questions, autograding, and server setup
- Interface testing
  - Ensure "student view" and "dev view" interfaces function correctly and provide a seamless user experience
- Integration testing
  - Test the synchronization of grades between the application and Canvas using mock courses
- System testing
  - Conduct end-to-end testing of all functionalities to confirm system coherence and reliability

Test Plan Continued

- Regression testing
  - Validate that new updates do not disrupt existing features
- Acceptance testing
  - Get feedback from client and students
- Security testing
  - Make sure only ISU students and professors can access our project

Prototype Implementations





Prototype Implementations

Coding Practice

Complete the following function, max_consecutive_1s, so that it returns the maximum number of consecutive 1's in the variable that is passed to it.

For example, for 0xFF5F the max number of consecutive 1's is 8.

Another example, for 0x77FE the max number of consecutive 1's is 10.

```
unsigned char max_consecutive_1s(unsigned short x)
{

}


int main()
{
    unsigned char max = max_consecutive_1s(0xFF5F);
    return 0;
}
```

studentCode.c

```
#include <stdio.h>
#include <stdlib.h>

unsigned char max_consecutive_1s(unsigned short x)
{
    int i = 0;
    unsigned char count = 0;
    unsigned char max_count = 0;

    for(i=0; i < 16; i++)
    {
        if( (x >> i) & 0x1 )
        {
        count++;
        }

        else
        {
            count = 0;
        }

        if(count > max_count)
        {
            max_count = count;
        }
    }

    return max_count;
}
```

Score: 1/1 (100%)

Test Results

✓ **[1/1] Test command: ./studentCode**

Max points: 1

Earned points: 1

Message

```
Expected all of:
        TEST 1 PASSED
        TEST 2 PASSED
        TEST 3 PASSED
        TEST 4 PASSED
        TEST 5 PASSED
```

Output

```
TEST 1 PASSED
TEST 2 PASSED
TEST 3 PASSED
TEST 4 PASSED
TEST 5 PASSED
```

Prototype Implementations

HW12.3. Arithmetic

Arithmetic

Given the following declarations, write the assembly code to implement each functionality.

```
// Variable declaration assumptions
unsigned int a;         // Defined at 0x20000384
unsigned int b;         // Defined at 0x2000051C
```

For this function: perform a = b + 9

student.s

```
1  .global _arithmetic
2  _arithmetic:
3      MOVW R0, 0x0384
4      MOVT R0, 0x2000
5      MOVW R1, 0x051c
6      MOVT R1, 0x2000
7      LDR R2, [R1]
8      ADD R2, #9
9      STR R2, [R0]
10     bx lr
```

Restore original file

Test Results

✓ [1/1] Testing with random inputs

Max points: 1

Earned points: 1

Message

```
Expected all of:
        0: a=251 b=242
        1: a=123 b=114
        2: a=23 b=14
        3: a=39 b=30
        4: a=107 b=98
```

Output

```
Timer with period zero, disabling
0: a=251 b=242
1: a=123 b=114
2: a=23 b=14
3: a=39 b=30
4: a=107 b=98

(NO ENDING LINE BREAK)
```

Conclusion

Current Status of Project

- A partial beta version will be released at the end of this semester
  - Homeworks 1-6 Finalized
    - Focusing on randomization, autograde capabilities, and quality of format
- Working authentication
  - Google OAuth works
  - ISU SSO still needs to be implemented

Task Breakdown of Each Member

Member	Responsibilities	Contributions
Joseph Krejchi	Quality Assurance	Fixed HW4; Researched QEMU ARM autograder
Devin Alamsya	Consultant	Fixed HW2 along with various other short answer questions; Developed a new question format
Caden Otis	Project Manager	Fixed HW12 1a and HW3; Researched QEMU ARM autograder
Rachel Druce-Hoffman	Notetaker	Improved HW6; Developed new question format; Recorded meeting events and feedback for future reference
Justin Cano	Technical Lead	Initialized and secured server VM; Set up Prairielearn; Worked with ASW to get SSL encryption on the server; Implemented SSO

Plan for Next Semester

- Perform a beta test for Spring 2025
- Fully implement homeworks 7-12
    - Considering feedback from beta test
- Fix various bugs/issues
- Work on finishing the virtual cybot emulator
    - Switch from bare bones emulator to Cybot emulator
- Write new questions not based on currently existing homeworks

Q&A