MARINADE

Technology Investigation Report

Prepared by

Curt Henrichs

Hunter Parks

Alex Roberts

Larry Skuse

Andrew Stoehr

Submitted on October 26th, 2017

Milwaukee School of Engineering Senior Design Team Beta

Electrical Engineering and Computer Science Department | Advised by Professor Barnekow

Table of Contents

Existing Technologies

The technologies listed in the existing technologies section of this report include products similar to the system our team is designing. It also lists existing toolchains and frameworks that have potential to assist our team in development.

ARMSim

- Description:
    - ARMSim is a basic ARM simulator that allows the user to run or step through their ARM assembly code and view the contents of both registers and memory.
- Notes:
    - ARMSim is currently used in CE-1921 to write single file ARMv4 assembly programs
    - Some functionality from ARMSim is missing (i.e. stdin functionality is not implemented)
    - This simulator does not offer an IDE for editing ARM assembly code
    - Source code is not available
    - Stakeholders have noted displeasure with is product
- URL:
    - http://armsim.cs.uvic.ca/

MDK-Lite Edition

- Description:
    - MDK stands for Microcontroller Development Kit. According to the ARM Keil website, "It is intended for product evaluation, small projects, and the educational market." (http://www2.keil.com/mdk5/editions/lite)
- Notes:
    - Dr. Meier and Dr. Rothe are opposed to the complexity of MDK
    - The lite-edition limits the functionality of the development kit and does not make it a viable option for undergraduate courses at MSOE
- URL:
    - http://www2.keil.com/mdk5/editions/lite

Visual

- Description:
    - Visual is a ARM emulator designed for undergraduate computer architecture courses.
- Notes:

    - Has in-depth simulation tools including pointer information for individual ARM instructions, a memory view, a stack view, a dialog showing status bits, a branch destination indicator, more intuitive error messages, and infinite loop detection
    - Cross-platform (works on macOS, Linux, Windows)
- URL:
    - https://salmanarif.bitbucket.io/visual/

Unicorn

- Description:
    - According the the unicorn website, “Unicorn is a lightweight multi-platform, multi-architecture CPU emulator framework.” (http://www.unicorn-engine.org/)
- Notes:
    - Unicorn supports both ARM and ARM64
- URL:
    - http://www.unicorn-engine.org/

GNU ARM Embedded Toolchain

- **Description:** According to the ARM Developer website, “The GNU Embedded Toolchain for ARM is a ready-to-use, open source suite of tools for C, C++ and Assembly programming targeting ARM Cortex-M and Cortex-R family of processors.” (https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads)
- Notes:
    - Cross-platform (works on macOs, Linux, Windows)
- URL:
    - https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads

Possible Solutions

The solutions listed in the possible solutions section of this report includes all considered development solutions for MARINADE. Under each possible solution, our team listed any applicable languages and technologies needed to implement the solution. Following the applicable languages/technologies list, a list of pros and cons were shown. The pros and cons were used in the next section to assist our team in rating each possible solution. Finally, the cost of the solution, in terms of money and time, was evaluated.

Web-Technology Based

- Applicable Languages/Technologies:
    - Frontend
        - Electron
        - HTML

        - CSS
        - Javascript/Typescript
        - Angular or React
    - Backend
        - C/C++
        - Python
    - Need to supply compiler such as GCC in install
- Pros:
    - Cross-platform (works on macOS, Linux, and Windows)
    - Easy to design UI/UX (CSS styling)
    - Clear separation between frontend and backend
    - Team member experience
    - Backend (given appropriate technology) will allow for hardware interaction
    - Simulator visualization within capabilities of front-end framework
- Cons:
    - Slower than native apps, runs a Chrome instance in the background
    - Not everyone on the team has Javascript/Typescript experience
    - Can be intimidating for non-web-developers, documentation is still being developed, ecosystem is always changing
    - Javascript/Typescript does not work well at the bit/byte level, need a separate backend to handle that (Python, C/C++.)
    - Have to build editor in addition to simulation functionality
- Cost:
    - Development tools: free
    - Effort: Moderate workload

Qt

- Applicable Languages/Technologies:
    - C++
    - Qt Framework
    - Need to supply compiler such as GCC in install
- Pros:
    - Native performance
    - Adapts to system UI, can be styled using QSS
    - Good documentation
    - Cross-platform (works on macOS, Linux, and Windows)
    - Everyone on the team has some experience with C++
    - Hardware device interaction should not be a hurdle
    - Simulator visualization is within capabilities of the framework
- Cons:
    - Harder to organize, harder to focus on UI/UX
    - Have to build editor in addition to simulation functionality

- Cost:
  - Development tools: free
  - Effort: Moderate to heavy workload

Eclipse with Addons

- Applicable Languages/Technologies:
  - Eclipse CDT for C/C++ development
  - Java or C for custom plugin development
- Pros:
  - Much of the framework is already built
  - Cross-platform (works on macOS, Linux, and Windows)
  - Strong and mature community already exists
  - Hardware device interaction either exists or can be easily integrated
- Cons:
  - Eclipse can be slow/buggy
  - Limited control of UI look
  - May involve other restrictions from the Eclipse environment
  - Simulator visualization may be a concern
- Cost:
  - Development tools: free
  - Effort: Moderate workload

Atom Editor with Plugin

- Applicable Languages/Technologies:
  - HTML
  - CSS
  - Javascript/Coffeescript
  - jQuery
  - Need external compiler such as GCC
- Pros:
  - Cleaner design and less buggy than Eclipse
  - UI framework is already built up
  - Cross-platform (works on macOS, Linux, Windows)
  - Strong and mature community already exists
- Cons:
  - Limited control of UI
  - Not everyone on the team has Javascript/Coffeescript experience
  - Can be intimidating for non-web-developers, documentation is still being developed, ecosystem is always changing
  - May involve other restrictions from the Atom environment
  - Hardware device will most likely need to be customly integrated
  - Simulator visualization may be a concern

- Cost:
  - Development tools: free
  - Effort: Moderate workload

JFrame (Swing) / JavaFX

- Applicable Languages/Technologies:
  - Java
  - JFrame (Swing) or JavaFX
  - Need external compiler such as GCC
- Pros:
  - Java is easy to work with
  - Cross-platform (works on macOS, Linux, and Windows)
  - UI/UX easy to design
  - Program can easily be made portable with .jar file
  - Everyone on the team has experience with Java
  - Simulator visualization within capabilities of libraries
- Cons:
  - Java is slow
  - Hardware device will most likely need to be customly integrated with possible hurdles
- Cost:
  - Development tools: free
  - Effort: Moderate to heavy workload

Tkinter

- Applicable Languages/Technologies:
  - Python
  - Need external compiler such as GCC
- Pros:
  - Python is simple to work with
  - Different functions could be added modularly as widgets
  - Simulator visualization is within the capabilities of libraries
  - Interaction with hardware devices should be no more challenging than C/C++ case
- Cons:
  - Python may not be as capable as languages such as C/C++
  - UI will not look as visually appealing as other solutions
- Cost:
  - Development tools: free
  - Effort: Moderate workload

ARMSim Plugin

- Applicable Languages/Technologies:
    - C#
    - .NET Framework
- Pros:
    - It has some features that meet requirements
    - Works on Windows and macOS
- Cons:
    - Currently being used in curriculum, noted displeasure with product
    - UI is incomplete and possibly unintuitive
    - Source code not available
    - May involve other restrictions from the ARMSim environment
    - Simulation visualization depends on the plugin’s relationship to the application
    - Hardware interaction depends on the plugin’s relationship to the application
- Cost:
    - Development tools: free
    - Effort: Moderate to heavy workload

VS Code Plugin

- Applicable Languages/Technologies:
    - Javascript/Typescript
    - VS Code Microsoft C/C++ extension (in preview)
    - Need external compiler such as GCC
    - ARM assembly syntax highlighting plugin (https://marketplace.visualstudio.com/items?itemName=dan-c-underwood.arm)
- Pros:
    - VS Code is growing in popularity
    - UI framework is already built up
    - Cross-platform (works on macOS, Linux, Windows)
    - Strong and growing community already exists
- Cons:
    - Limited control of UI
    - May involve other restrictions from the VS Code environment
    - Not everyone on the team has Javascript/Typescript experience
    - Hardware device will most likely need to be customly integrated
    - Simulator visualization may be a concern
- Cost:
    - Development tools: free
    - Effort: Moderate workload

Classic Client-Server Model using a Website

- Applicable Languages/Technologies:
    - Frontend
        - HTML
        - CSS
        - Javascript/Typescript
        - Angular or React
    - Backend
        - Java (Tomcat)
        - Python (Django REST/Flask)
        - Javascript (Express)
    - Need external compiler such as GCC on remote server
- Pros:
    - Nothing for users to install, simply navigate to the right URL
    - Could result in a business opportunity
    - Simulator visualization within capabilities of front-end framework
- Cons:
    - No support from IT
    - Would have to pay for hosting and server resources
    - Have to manage sessions
    - Have to upload code to run remotely
    - Hardware device interaction will be a major concern
    - Not everyone on the team has Javascript/Typescript experience
    - Can be intimidating for non-web-developers, documentation is still being developed, ecosystem is always changing
- Cost:
    - Development tools: free
    - Effort: Moderate to heavy workload

Ratings for Possible Solutions

Using the pros and cons discussed above, our team rated each possible solution on a scale of 1 to 10. A rating of 1 signifies the solution is insufficient and the system could not be designed appropriately using this technology. A rating of 10 signifies the solution is ideal and the system can be designed to meet or exceed all requirements given a reasonable amount of effort is invested in developing the solution. Each possible solution received an independent rating. The ratings can be seen in the table below. The solution with the highest average rating was selected as the technology our team will use to develop the project.

	Curt Henrichs	Hunter Parks	Alex Roberts	Larry Skuse	Andrew Stoehr	Average Rating
Web-technology based	7	9	9	8	8	8.2
Qt	2	4	5	7	3	4.2
Eclipse with Addons	5	2	2	3	4	3.2
Atom Editor with Plugin	5	5	4	4	4	4.4
JFrame (Swing) / JavaFX	1	6	7	6	4	4.8
Tkinter	3	2	3	2	5	3.0
ARMSim Plugin	2	2	2	4	3	2.6
VS Code Plugin	6	5	4	4	6	5.0
Client-server model using a website	4	2	2	2	6	3.2

Conclusion

The team rated the web-technology based solution the highest. This allows us to use electron to build the frontend of the application and C/C++ and Python to build the backend. This solution will allow the system to be both cross platform and aesthetically pleasing. Additionally, the software engineer, computer engineer double majors on our team have extensive experience with web technologies allowing our team to jump into the UI design right away. Given the nature of the project, getting rapid feedback on the UI from the stakeholders should allow for a better end product. However, electron does have some drawbacks. Although it provides a clear separation between the frontend and the backend, one of the greatest challenges our team will face will be getting the backend to talk with the frontend. Additionally, it runs a chrome instance in the background with will cause the system to use a significant amount of RAM. Finally, if time allows for hardware device interaction to be accomplished, this choice will afford the greatest flexibility to integrate with existing toolchains. Overall, this solution allowed will allow us to create the highest quality product.