Open-Source Keypad for Diacritic Characters

Security Assessment Report

Version 1.0

April 27, 2023

Table of Contents

1. Summary 3
    1. Assessment Scope 3
    2. Summary of Findings 3
    3. Summary of Recommendations 4
2. Goals, Findings, and Recommendations 4
    1. Assessment Goals 4
    2. Detailed Findings 5
    3. Recommendations 5
3. Methodology for the Security Control Assessment 7
        3.1.1 Risk Level Assessment 7
        3.1.2 Risk Assessment 8
        3.1.3 White Box Testing 8
        3.1.4 Tools Used 8
        3.1.5 Arduino Vulnerabilities 9
4. Figures and Code 9
        4.1.1 Processing Cycle Flowchart 9
        4.1.2 Project Build and Test Flowchart 10
        4.1.3 Sample Logging Code 10
5. Works Cited 11

1. Summary

We have determined that the security of the open-source Diacritic Keypad project was mediocre through product observation and white box testing. The major issues were centered around lack of explanation of systems over bad design or implementation.

1. Assessment Scope

Arduino IDE on Windows 10 and Windows 11 was utilized during testing, with the C project code being tested. GitHub was tested through Firefox, Edge, and Chrome, focusing primarily on the content more than the formatting.

2. Summary of Findings

Of the findings discovered during our assessment, 1 was considered High risk, 0 Moderate risks, 2 Low, and 5 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 1.

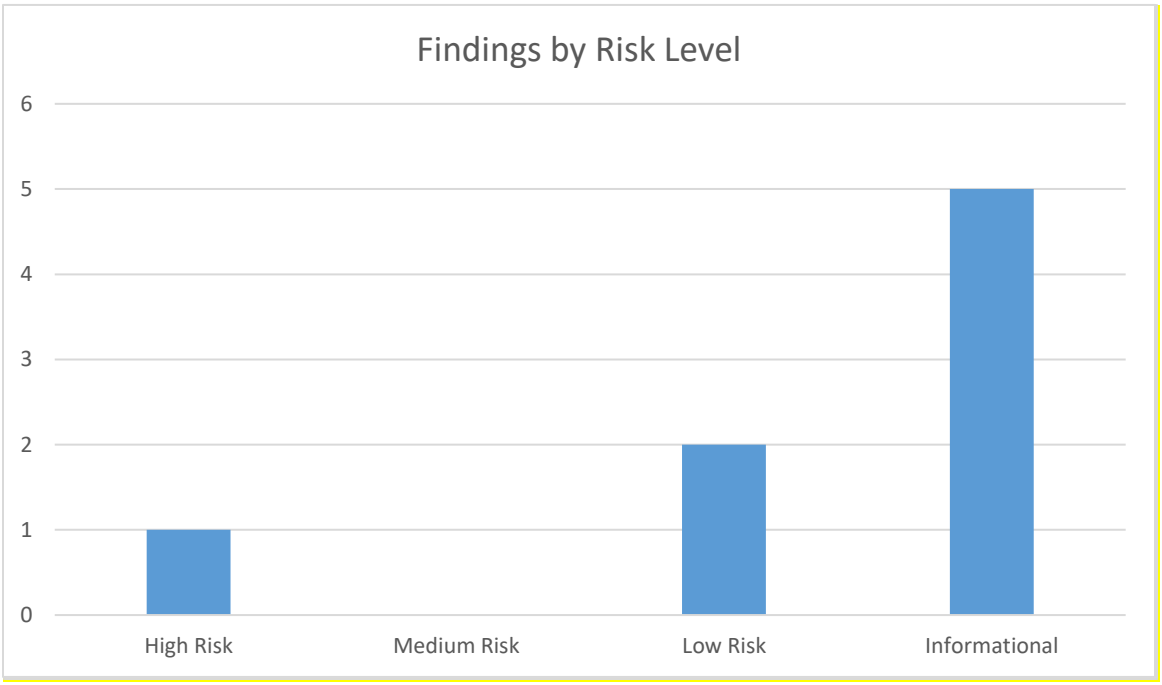


Figure 1. Findings by Risk Level

The top risk was a simple issue that the code of the device when finalized can still be easily overwritten, leading to many issues for the end user, and the potential for dangerous security vulnerabilities to be added to an otherwise secure computer. The remaining risks were either already handled or were more minor changes focused on improving the explanations with the project.

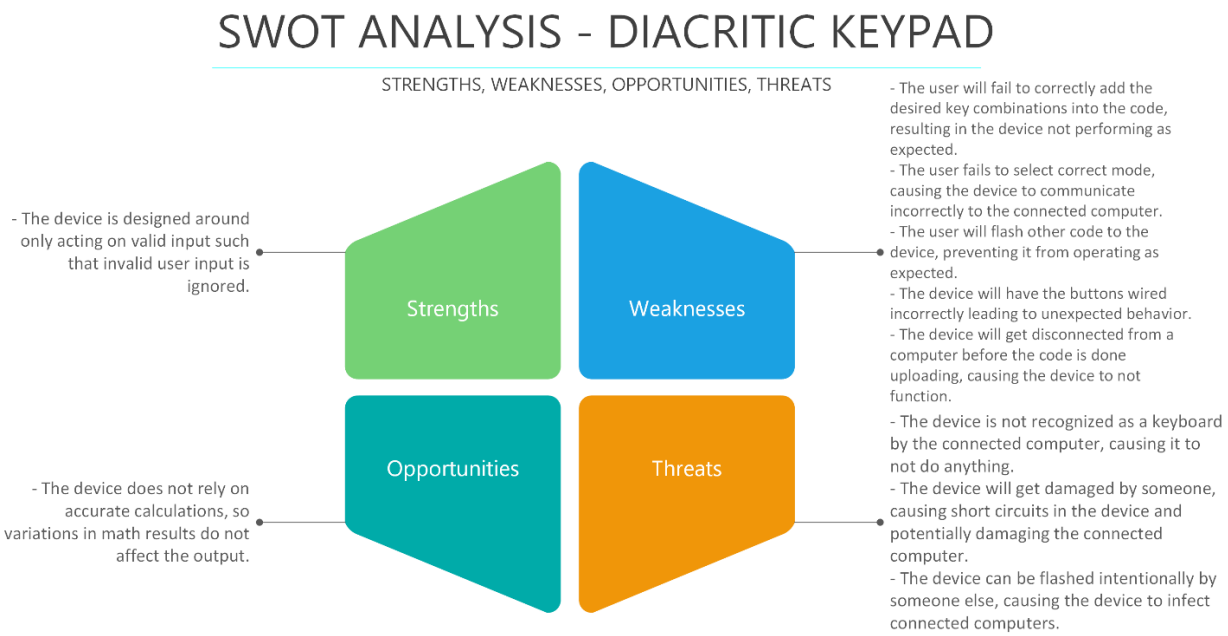


Figure 2. SWOT

The issues related to code being overwritten and bad assembly of project were addressed from the SWOT shown in Figure 2.

3. Summary of Recommendations

The provided supporting documentation and content needs to be improved to address assumptions made during the initial project, particularly focusing on assumed prior knowledge of electronics, wiring, and general safe practices of people considering attempting the project. Some guides have been built to address these issues, but more depth needs to be added to certain sections, such as modifying the code, as well as adding video-based content to demonstrate safe practices and techniques.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Ensure that the system was in compliance with basic user security as defined in CEN 3078.
- Determine if the application was securely built.
- Assess necessary corrective measures to reduce or eliminate identified issues.

2. Detailed Findings

Type	Severity	Detailed Finding
Weakness	Informational	The project code uses several C-style arrays throughout to store state data and reference keys. These arrays are easy to look beyond the bounds of as they decay to pointers readily, as discussed in CEN 3073 lecture on March 30th, 2023, particularly in loops, which is how this project utilizes the arrays.
Threat	Informational	The project is based on responding to user input, and not all inputs are valid, since only specific combinations of keys will produce a valid diacritic character, so faulty user input must be considered in every section that takes input from the user to ensure all scenarios are handled, even unexpected scenarios.
Threat	Moderate Risk	The code stored on hobby use microcontrollers such as the recommended Arduino Pro Micro have a built-in bootloader to allow for code to be easily uploaded when a computer is connected, as discussed online in the forum thread “Arduino pro Micro - Is It Possible to Remove the Bootloader While Flashing Using the Bootloader?”. However, this project intends to leave the microcontroller always connected to the computer, therefore code can always be written to it, overwriting the previous code, which can lead to unintended outcomes if the device is completed and another project’s code gets uploaded to this device.
Weakness	High Risk	The project code assumes that the pins defined at the top are correct, and if they are not updated to match each instance of the project built, the project may not work as intended or at all, as the communication wires expected are not actually there, but the code has no checks for such a condition.
Weakness	High Risk	The project requires that everything is wired correctly with no short circuits and the correct components. If this is not the case, the project is very likely to not function at all and even has the risk to physically damage components including the microcontroller and potentially also connected computers.
Threat	Observation	This project’s code includes 1 library, ‘Keyboard.h’. Since this library is included, the security of it must also be assessed since it is adding code to the project as well as being actively and extensively used in the project. However, due to the scale of the Arduino Pro Micro and the wide variance of it, there is very little focus on security holes in the library that may exist, as seen in the issues for the code on GitHub (“Issues · Arduino-Libraries/Keyboard.”). Most focus is instead pointed towards how the library can be used in security attacks on other systems, as seen by the research done by Seth Helgeson, et al., among many others.
Threat	Observation	The design of the project, particularly the case for the project, is very important as well. A poorly designed case will not hold things tightly and securely together, leading to the wires potentially getting ripped off, crushed, broken, touching each other (short circuit), or otherwise being exposed to being damaged over time. Many of these issues will not appear immediately but will instead manifest over time, leading to erratic behavior long after the project is completed, making the issue hard to identify for inexperienced users.
Weakness	Low Risk	This project features only minimal logging in the form of a debugging and test mode that can be enabled and disabled within the code. Additionally, even when this logging is occurring, a serial monitor must be running on an attached computer in order to see any of the log data as it occurs, as no information is saved to the microcontroller. Furthermore, no information is logged during normal use.

3. Recommendations

1. Add a Design and Wiring Advice section in ReadMe.md to help avoid potential regulatory issues by advising people to tailor their designs to be safe and not pose a risk

of electric shock or could otherwise cause damage to other systems. Ease of fix: Moderately Difficult – guide must be thoroughly written to be most effective.

2. Add a testing guide section in ReadMe.md to provide some basic training on how to test a project such as this for people with no prior experience. Also explain how to utilize the basic logging systems included in the provided code, since it is real-time only and done through the serial monitor in Arduino IDE. Ease of fix: Moderately Difficult – guide must be thoroughly written to be most effective.
3. Update end of ReadMe.md with fair use licensing remark and end user risk disclaimer. Ease of fix: Easy – simple template to follow, partially provided in GitHub ("Licensing a Repository.").
4. Vulnerabilities in included libraries (Keyboard.h). Perform extensive online research to identify any known and relevant security vulnerabilities included in that library. Ease of fix: Moderately Difficult – must check many sources and patch notes for issues.
5. Physical security of actual hardware. Add a disclaimer in the project ReadMe.md to make users aware of the risks they must consider for their situation when designing and building this product. Ease of fix: Easy – not an extensive guide, just a disclaimer.
6. Fix product code being overwritten. Update the ReadMe.md file of the project detailing different things that can be done to reduce or eliminate this vulnerability. Ease of fix: Moderately Difficult – guide must be thoroughly written for users not familiar with alternative bootloader tools ("Disabling Skethc Uploading on Arduino pro Micro.").
7. Add security design considerations section (including issues with adding wireless connectivity). Ease of fix: Moderately Difficult – many scenarios to consider, but not an extensive guide.

8. Update logging option to code, primarily for debugging, but with EEPROM as an option if desired. Ease of fix: Difficult – need to restructure code and deal with creating and exporting files.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment

Each Business Risk has been assigned a Risk Level value of High, Moderate, or Low. The rating is, in actuality, an assessment of the priority with which each Business Risk will be viewed. The definitions in **Error! Reference source not found.** apply to risk level assessment values (based on probability and severity of risk). While Table 2 describes the estimation values used for a risk's "ease-of-fix".

Table 1 - Risk Values

Rating	Definition of Risk Rating
High Risk	Exploitation of the technical or procedural vulnerability will cause substantial harm to the business processes. Significant political, financial, and legal damage is likely to result
Moderate Risk	Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity and/or availability of the system, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment to organization.
Low Risk	Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The confidentiality, integrity and availability of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment
Informational	An "Informational" finding, is a risk that has been identified during this assessment which is reassigned to another Major Application (MA) or General Support System (GSS). As these already exist or are handled by a different department, the informational finding will simply be noted as it is not the responsibility of this group to create a Corrective Action Plan.
Observations	An observation risk will need to be "watched" as it may arise as a result of various changes raising it to a higher risk category. However, until and unless the change happens it remains a low risk.

Table 2 - Ease of Fix Definitions

Rating	Definition of Risk Rating
Easy	The corrective action(s) can be completed quickly with minimal resources, and without causing disruption to the system or data
Moderately Difficult	Remediation efforts will likely cause a noticeable service disruption • A vendor patch or major configuration change may be required to close the vulnerability • An upgrade to a different version of the software may be required to address the impact severity • The system may require a reconfiguration to mitigate the threat exposure • Corrective action may require construction or significant alterations to the manner in which business is undertaken
Very Difficult	The high risk of substantial service disruption makes it impractical to complete the corrective action for mission critical systems without careful scheduling • An obscure, hard-to-find vendor patch may be required to close the vulnerability

Rating	Definition of Risk Rating
	• Significant, time-consuming configuration changes may be required to address the threat exposure or impact severity • Corrective action requires major construction or redesign of an entire business process
No Known Fix	No known solution to the problem currently exists. The Risk may require the Business Owner to: • Discontinue use of the software or protocol • Isolate the information system within the enterprise, thereby eliminating reliance on the system In some cases, the vulnerability is due to a design-level flaw that cannot be resolved through the application of vendor patches or the reconfiguration of the system. If the system is critical and must be used to support on-going business functions, no less than quarterly monitoring shall be conducted by the Business Owner, and reviewed by IS Management, to validate that security incidents have not occurred

3.1.2 Risk Assessment

Severity		Frequent	Probable	Likely	Possible	Rare
|	Emergency				People will intentionally flash other code to the device.	
|	Major					
|	Moderate	The user failed to configure device code correctly.		People will damage the device.		
|	Minor	The user enters an invalid input combination. The device communicating incorrectly to connected computer.		The user unintentionally flashes other code to the device.	The device is wired incorrectly.	The device fails to properly flash new code.
\/	Negatable		The device is not recognized by connected computer.			The device will have unexpected calculation results.

explain accuracy of it (what was and what wasn't)

3.1.3 White Box Testing

This was completed by checking both known-valid and known-invalid inputs on the device and verify the code can handle the all the inputs without breaking or otherwise working in unexpected or undocumented ways.

3.1.4 Tools Used

This was completed using Arduino IDE's Serial Monitor and dedicated serial port logging code in the project. Used to verify all button inputs work as expected, as well as key layout is known, and

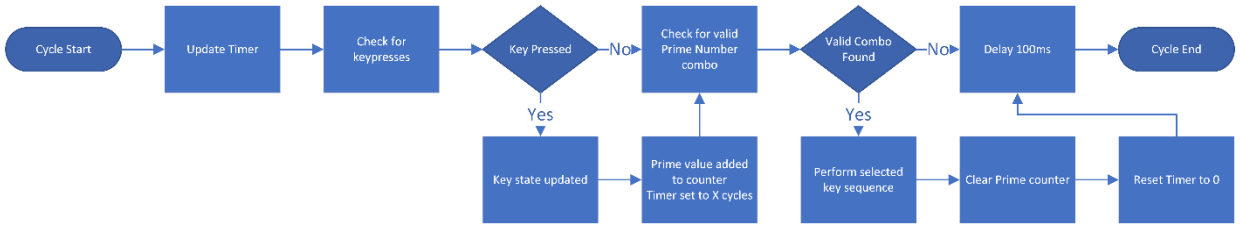
that there are no logical errors within the code, particularly since this part is defined by the end user and cannot be tested ahead of time.

3.1.5 Arduino Vulnerabilities

There are few major vulnerabilities in the Arduino hardware, specifically the Arduino Pro Micro, based on the ATMega32u4 processor. There are also no known vulnerabilities in the only included file in the C code of the project, Keyboard.h, at least as reported to the library's issues tracker in GitHub ("Issues · Arduino-Libraries/Keyboard."). There is currently very little interest in trying to hack these devices and libraries since they simply are not widespread enough nor containing valuable enough information for most hackers to target these, instead they may focus on consumer units that are similar but using different code and hardware ("Security Risk of Emulating Hid Devices with ATmega32U4 Devices."). The only major attention focused on these type of devices is how they can be utilized in attacks of other systems, primarily when a USB port can be accessed (Helgeson, Seth, et al.).

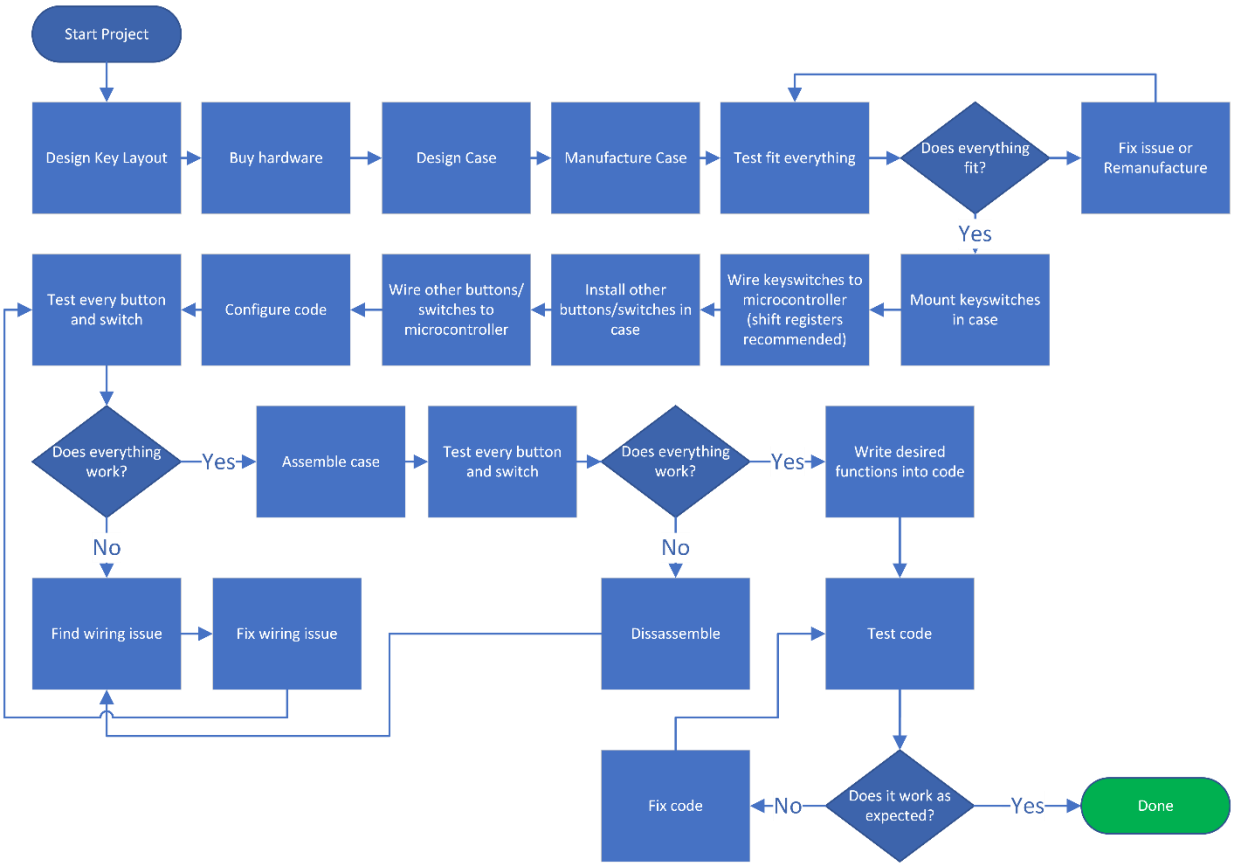
4. Figures and Code

4.1.1 Processing Cycle Flowchart



The processing flowchart shown above describes the execution path of the project code in a high level, including the branches reacting to user input, when it is given.

4.1.2 Project Build and Test Flowchart



The flowchart above details all the steps for completing this project as an outside user. Includes all steps between starting the project and having a complete and working product. The process has many points for users to make changes, necessitating several checks throughout that cannot be done by the project itself.

4.1.3 Sample Logging Code

```
// Print out extra debug information if enabled
#ifdef ExtendedTestMode
  Serial.print("Button Index: ");
  Serial.print(button);
  Serial.print("    Value: ");
  Serial.println(value);
#endif
```

The code snippet shown above is a sample of the logging code currently in the project's code. This section and several others could be expanded to include writing to an internal file. In adding that feature, another function would also need to be added for exporting that internal file to a computer to be used for debugging or maintenance.

5. Works Cited

"Arduino pro Micro - Is It Possible to Remove the Bootloader While Flashing Using the Bootloader?" *Arduino Stack Exchange*, arduino.stackexchange.com/questions/90543/is-it-possible-to-remove-the-bootloader-while-flashing-using-the-bootloader. Accessed 26 Apr. 2023.

"Disabling Skethc Uploading on Arduino pro Micro." *Arduino Forum*, 2 Nov. 2022, forum.arduino.cc/t/disabling-skethc-uploading-on-arduino-pro-micro/1048808. Accessed 26 Apr. 2023.

"Issues · Arduino-Libraries/Keyboard." *GitHub*, github.com/arduino-libraries/Keyboard/issues?q=is%3Aissue. Accessed 26 Apr. 2023.

"Licensing a Repository." *GitHub Docs*, docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository#disclaimer.

"Security Risk of Emulating Hid Devices with ATmega32U4 Devices." *Arduino Forum*, 5 July 2021, forum.arduino.cc/t/security-risk-of-emulating-hid-devices-with-atmega32u4-devices/881892. Accessed 26 Apr. 2023.

Helgeson, Seth, et al. *Wireless Exploitation through Keyboard Emulation*. Athens: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018. *ProQuest*. Web. 26 Apr. 2023.

Week 11 Lecture – March 30th - about code vulnerabilities (c-style arrays and pointers)