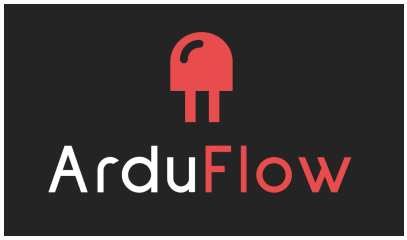

Personal Project Technology Research

Michał Raczkowski
OL S6
4465024

version 0.1v

Contents

1 Overview 2

2 Research 2

2.1 Native Desktop Application 2

2.2 Web-Based Application 2

2.3 Cross-Platform Framework 3

2.4 IDE Extension 3

3 Comparison Table 3

4 Conclusion 4

4.1 Development Complexity 4

4.2 Code Reusability 4

4.3 Learning Curve 4

4.4 Platform Compatibility 4

Version	Date	Author	Comment
0.1v	01.06.23	M. Raczkowski	Overview, Research, Comparison table, Conclusion

1 Overview

This research paper aims to determine the most suitable technology for developing a desktop application called ArduFlow, which is designed for programming LED animations on an 8x8 matrix controlled by Arduino. The paper explores various technologies commonly used in desktop application development, including native desktop applications, web-based applications, cross-platform frameworks, and IDE extensions. By evaluating factors such as performance, platform compatibility, development complexity, and user experience, the research seeks to identify the technology that best aligns with the requirements and objectives of ArduFlow. The findings of this research will assist developers in making informed decisions and selecting the optimal technology stack for developing the ArduFlow desktop application.

2 Research

This section provides a brief overview of the different options considered for developing the ArduFlow desktop application. The possible solutions include creating a native desktop application, developing a web-based application, utilizing a cross-platform framework, or extending an existing IDE. Each solution has its own advantages and factors to consider. By evaluating these options, developers can make an informed decision on the best approach to develop the ArduFlow desktop application based on their specific needs and requirements.

2.1 Native Desktop Application

One potential solution for developing the ArduFlow desktop application is to create it as a native desktop application using programming languages like C++ or Java. This approach would involve leveraging platform-specific frameworks such as Qt or JavaFX for GUI development. The advantages of this approach include high performance, direct access to hardware resources, and the ability to provide a seamless user experience. However, it's important to consider that developing native desktop applications can require more advanced programming skills and can be more time-consuming compared to other options. Additionally, ensuring platform compatibility across different operating systems may present a challenge.

2.2 Web-Based Application

Another option is to develop ArduFlow as a web-based application using standard web technologies like HTML, CSS, and JavaScript. This approach would involve utilizing popular front-end frameworks such as React, Angular, or Vue.js to facilitate efficient development. One of the key benefits of a web-based application is its cross-platform compatibility, as it can run on any device with a web browser. Web applications also offer easy deployment and accessibility. However, it's important to consider performance limitations and the limited access to hardware resources typically imposed by web browsers.

2.3 Cross-Platform Framework

Utilizing a cross-platform framework like Electron or Flutter is an alternative solution for developing the ArduFlow desktop application. These frameworks enable developers to write code once and deploy it on multiple platforms, including Windows, macOS, and Linux. This approach offers a balance between performance, platform compatibility, and development efficiency. However, it's worth noting that applications built with cross-platform frameworks may have slightly lower performance compared to native solutions. Developers must also consider the learning curve associated with these frameworks and the potential trade-offs between code sharing and platform-specific optimizations.

2.4 IDE Extension

An additional approach is to extend an existing Integrated Development Environment (IDE) like Arduino IDE or Visual Studio Code to incorporate ArduFlow functionality. By leveraging the capabilities and user base of the IDE, developers can provide a seamless development experience for users. This approach offers integration with existing tools, familiarity for users, and ease of use. However, it's important to consider that GUI customization within an IDE Extension may be limited by the capabilities and APIs provided by the IDE. Developers may have restricted control over the overall look and feel of the GUI compared to other development approaches.

When evaluating these possible solutions, developers should consider the project requirements, target audience, available development resources, and desired outcomes. Careful analysis of these factors will help determine the most suitable solution for developing the ArduFlow desktop application, ensuring a successful and efficient development process.

3 Comparison Table

Solution	Advantages	Considerations
Native Desktop	High performance, direct hardware access	Platform compatibility, development complexity
Web-Based	Cross-platform compatibility, easy deployment	Performance limitations, limited hardware access
Cross-Platform Framework	Code reusability, platform compatibility	Slightly lower performance than native solutions, learning curve
IDE Extension	Integration with existing tools, familiarity	GUI customization limitations, reliance on IDE capabilities

4 Conclusion

Based on the functionalities and features required for the ArduFlow project mention in ”Personal Project Idea, Specification and Requirements” document, and considering the trade-offs of each solution, the most suitable option appears to be a Cross-Platform Framework Here’s the rationale behind this decision:

4.1 Development Complexity

Cross-platform frameworks like Electron or Flutter simplify the development process by allowing developers to write code once and deploy it across multiple platforms, such as Windows, macOS, and Linux. This eliminates the need to build separate codebases for each platform, reducing the overall development complexity and effort required.

4.2 Code Reusability

Cross-platform frameworks facilitate code reusability, as a significant portion of the codebase can be shared across platforms. This minimizes duplication and eases maintenance, making it more manageable to implement and update functionalities consistently across different platforms.

4.3 Learning Curve

While there may be a learning curve associated with cross-platform frameworks, it is generally less steep compared to developing native applications for multiple platforms. These frameworks provide extensive documentation, resources, and vibrant developer communities that can assist with learning and troubleshooting, helping to mitigate some of the complexities.

4.4 Platform Compatibility

Cross-platform frameworks are designed to ensure compatibility across multiple platforms. This means that the ArduFlow application can be distributed to users on various operating systems without the need for separate builds or modifications, simplifying the distribution and deployment process.