ARIZONA STATE UNIVERSITY

HONORS THESIS

Building a Mobile Device that Uses the Power of a Desktop Computer

Author:
Dylan LATHRUM

Supervisor:
Dr. Robert HEINRICHS

A thesis submitted in fulfillment of the requirements for the degree of Software Engineering

for

Barrett, The Honors College

March 20, 2022

Declaration of Authorship

I, Dylan LATHRUM, declare that this thesis titled, "Building a Mobile Device that Uses the Power of a Desktop Computer" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Premature optimization is the root of all evil."

Sir Tony Hoare

ARIZONA STATE UNIVERSITY

Abstract

Barrett, The Honors College

Software Engineering

Building a Mobile Device that Uses the Power of a Desktop Computer

by Dylan LATHRUM

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

List of Abbreviations

CM4	Raspberry Pi **C**ompute **M**odule **4**
CPU	**C**entral **P**rocessing **U**nit
CRD	**C**hrome **R**emote **D**esktop
GPU	**G**raphics **P**rocessing **U**nit
IDE	**I**ntegrated **D**evelopment **E**nvironment
PC	**P**ersonal **C**omputer
PCB	**P**rinted **C**ircuit **B**oard
RDP	**R**emote **D**esktop **P**rotocol
SBC	**S**ingle **B**aard **C**omputer
SMD	**S**urface **M**ounted **D**evice
SSH	**S**ecure **Sh**ell (Protocol)
VNC	**V**irtual **N**etwork **C**omputing

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Power and Portability

TODO

1.2 Purpose of this Thesis

TODO

1.3 Thesis Overview

TODO

Chapter 2

Background

Computers have a long history of being built to serve a particular purpose, often compromising some aspects of the system to bolster others. Chapter 2.1 discusses the history of how computers were built for a particular function, and how the advancement of technology has brought the "Personal Computer" seen know today. Chapter 2.1.1 discusses the benefits of a desktop computer, Chapter 2.1.2 discusses the benefits and drawbacks of a laptop computer, and Chapter 2.1.3 discusses the rise of the gaming laptop and the closing gap between power and portability at the cost of increased price. Chapter 2.2 briefly discusses thin clients, a corporate solution for remote access to a computer. Chapter 2.3 introduces current world wide context that has an impact on the computing world and the need for power and portability.

2.1 Specialization of Computers

In the age of punch cards and monolithic computers the size of entire rooms, it was expected of a computer to only be capable of serving a single function. Before the technology could be miniaturized, every single computer had to be specialized for the job at hand. As time passed and technology progressed, a single computer could be produced that served multiple functions as well as be adaptable programmable to handle new tasks that were not considered when the machine was originally built. This gave rise to what was known as "Autonomic Computing", a system of characteristics that are built into a computer to help it self-manage it's resources and adapt to it's administrator's requirements without the need to be redesigned for it's new purpose [12]. Autonomic Computing was designed to combat the exponential complexity crisis that came from the widespread availability of computers in different disciplines, since now anyone who could afford a computer could program it for any purpose. As new use cases were found for computers in day-to-day life and new technologies were being developed to interface with the world around us, the need for componentization became ever more apparent. Researchers at IBM knew that the best way to address the looming problem of runaway complexity in computers was to develop a way for the computers to automatically interface with any new components that are installed, and to configure itself to only use what is necessary for the job at hand [12, p. 43]. This allowed hardware and software developers to focus on building their products to work with a common standard rather than having to manually integrate their products with every computer.

Once the idea of modularity began to take hold in the computer industry, attention was turned once again towards specializing individual computers. Now that a computer's physical footprint can be minified, and peripheral components can be added and removed without a complete reengineering of the device, a single computer can be optimized to handle a single task without the overhead cost of

building a monolithic computer [3]. Now a computer can be specialized to serve a particular purpose or set of purposes while keeping development time comparatively short. The world has seen this realized through numerous applications such as cell phones, designed to be user-friendly portable devices, computing clusters, built for high-performance computing, or even internet routers, which are built to be a plug-and-play solution for a problem posed to users of all levels of familiarity with computers. Without this idea of Autonomic Computing, each of these devices would have to be reengineered from the ground up every time a new use case was developed.

This leads into the specialization of home computers in from the perspective of a consumer. It used to be that a "home computer" was a device that was bought off the shelf as-is and served it's purpose. Now a home computer can be a desktop PC that can be upgraded with time and seldom moves from it's place, or it could be a laptop that is portable and easy to cary around. These specializations bring more choices for the consumer to pick a computer that best suits their needs, but they often come with compromises.

2.1.1 Power of the Desktop

The classical manifestation of a personal computer is the desktop computer. Known for it's componentization, user repairability, and direct connections to power and the network, desktop computers are the best option for a user looking for a workhorse system. Pieces of the computer could be bought separately, and single parts could be upgraded or replaced by anyone will a little hardware knowledge. Even though they aren't very portable, desktop computers allow consumers to access greater processor power for a lower cost compared to portable devices [14]. Due to this, a desktop PC is often seen at one's home or place of work hardwired to the wall where it remains unmoving unless it needs upgrading, cleaning, or repairs. Adding on the fact the desktop is modular, it can be easily customized to fit any user's needs, as well as enable particular parts to be replaced or upgraded without needing to rebuild the entire system.

While desktops were the staple of personal computers for decades, advancing technology and the rising need of portability and flexibility has led to the growth of laptops.

2.1.2 Convenience of the Laptop

As the world moved towards a more mobile lifestyle, the laptop shifted from being a luxury to being a necessity. People began to need computing power on the go, whether it be for working remotely, attending classes and following along, or simply needing more power than their phone can offer. Companies began to take note too and started building applications to work fully within a web browser, such as Google's implementation of Google Docs, allowing users to edit documents on the go without having to install software like Microsoft Word directly to their computers. This shift towards mobility provided the general public the ability to be more flexible with how they used computers, no longer requiring them to dedicating a spot in their house to be taken up by a desktop computer.

Reducing the barrier to entry for computers, such as the knowledge to buy the computer the best fits their needs and the restrictions that come with setting up and keeping a desktop, led to the widespread adoption of laptops. In fact, by 2021, laptops accounted for 80% of total computer sales [18]. For most consumers, the

laptop is all they need; it may not have the raw power of a desktop, but it handles everything they need it to do. The biggest limiting factor then becomes technology, with manufacturers constantly trying to minimize existing desktop technology into the footprint of a laptop while keeping energy efficiency high enough to be feasible in a mobile device.

2.1.3 Rise of the Gaming Laptop

All computer products are a balance, it simply isn't possible to have the perfect combination of portable, powerful, serviceable, and cheap. While desktops focus on power and serviceability, laptops often focus portability at the expense of other traits. As mobility becomes ever more important, gaming laptops have become more popular to bring power to the portable form factor. This understandably comes with a cost, with the literal cost of a gaming laptop being the most obvious compromise. The componentization of such a laptop also often takes a hit with parts being directly soldered to the motherboard, reducing a user's options for repairs or upgrades. It is more expensive, and often less user-serviceable, to buy a gaming laptop that has the same performance of a desktop computer – but given that it is an all-in-one device, it's portability gives it a niche that a desktop cannot hope to rival. The compromise presented by a gaming laptop helps close the gap between power and portability, but it still falls short of being an all around good solution.

Theres a word at the tip of my tour thats a lot better here dangit

2.2 Thin Clients

On the corporate side of computing, the solution for portability has been the thin client. Rather than integrating the power of a desktop to a portable machine, a thin client will connect to a powerful server and provide a thin interface to the user. This allows the user to utilize the power of the host machine from a cheap computer that usually doesn't have enough power to be used as a standalone computer. Because of it's low power, the thin client cannot be used as a standalone computer, but it's low cost and portability make it a great solution for a corporate user. While these clients work well for simple interfacing tasks such as data entry or system management, they are not designed to be used for complex tasks that require quick response times or graphical fidelity.

FIGURE 2.1: A thin client next to a traditional desktop computer [9].

2.3 Application to Modern Day

In 2022, the world is currently going through some chaotic times that demand flexibility and adaptability like it hasn't faced before. From the COVID-19 pandemic to massive chip shortages and supply line disruptions, people are now more than ever looking for a way to be flexible in their work and personal lives now and in the future. The COVID-19 pandemic has forced people to be able to operate remotely, whether that be for work, school, or communicating with friends and family [13]. Many office workers used to working on a desktop computer onsite are now needed

to work from home, or classes usually taught in person are now held remotely [6]. This shift to remote operation has fueled the need for mobile computers, many of which need to be powerful enough to handle the workload of a desktop computer from anywhere in the world. The global chip shortage is also causing issues for the purchase of new devices [11]. Parts which are usually readily available for purchase are out of stock with months of delay on back order. Resellers are taking advantage of the situation by hiking prices to two or three times the original listing price, and it's become harder than ever to build a new PC [19]. All of these factors combined lead to a very difficult situation for anyone looking to purchase a new computer or parts for their existing computer.

It doesn't make sense to build or purchase a powerful PC for use at home and another power laptop to handle working on the go, especially with rising prices, so many people are moving to exclusively powerful laptops or looking for other solutions.

Chapter 3

State of the Art

This chapter seeks to provide a summary of existing solutions for remote computer access. It will not cover specific applications, but rather the protocols that are used to communicate with a remote computer. Generally speaking, there are two types of solutions that have been developed, software solutions, which are covered in Chapter 3.1 and hardware solutions, which are covered in Chapter 3.2. Various protocols and implementations are covered in the subsections of those two categories.

3.1 Software Solutions

While there are many applications in existence that allow a user to remotely control a computer from another location, few of them offer the capabilities required to stream high-performing applications to a remote device. The following subsections will introduce existing solutions to this problem as well as drawbacks that come with each implementation.

3.1.1 Remote Desktop Protocol

Microsoft's Remote Desktop Protocol (RDP), is a protocol that defines communication between a terminal server and terminal server client for multimedia purposes [21]. Coming pre-installed on every Windows machine since Windows XP and with clients available for Windows, Mac, and Linux, RDP is an easy to use solution for remotely accessing Windows machines graphically.

Although it is the built-in solution for Windows machines, it isn't without it's drawbacks. Firstly, only one graphical session is active at one time while using windows. This means if a user is logged into the host computer and someone initiates a connection with RDP, the user using the host computer will be logged out. While this isn't always an issue, sensitive programs that don't take well to being logged out may run into issues. Secondly, RDP does not support relative mouse movement [4]. This means every mouse input is sent to the host computer as an absolute position on the screen, rather than as a relative distance from it's previous location. This means any program that moves the user's mouse for them, such as 3D applications with a virtual camera, will be sent the absolute position of the client's mouse. This will cause the application to detect a large movement of the mouse instantly, which can result in erratic camera movement as the program interprets the mouse moving a large distance every time the cursor is reset to the center of the screen. Again, while this isn't always an issue, tasks such as 3D modeling and animation or game development cannot be controlled properly.

3.1.2 Virtual Network Computing

Virtual Network Computing (VNC), is a platform-independent system originally developed by Olivetti & Oracle Research Labs and later bought and shelved by AT&T [20]. While the original implementation is no longer used in a wide capacity, the protocol it was developed on has been expanded and improved to become one of the most flexible yet simple ways to control a computer remotely. VNC can run on any modern operating system, and even a web browser can serve as a VNC client. It was originally built as the simplest form of remote computer control over LAN, allowing system administrators to control almost any kind of computer with a simple, robust, and extremely compatible system. However, since its use has gradually outgrown it's original scope, VNC isn't usually the right tool for any job greater than remote system management. Because of it's simplicity and focus on being as straightforward as possible, VNC is often found to be lacking in terms of security and speed [20]. It is not recommended to use VNC over the internet without some secondary form of security, such as SSL or a secure VPN.

3.1.3 Chrome Remote Desktop

uld this chapter be called Chromot- "Chrome Remote Desktop" is the h more recognizable name though

Google Chromoting, implemented publicly as Chrome Remote Desktop, is an open source protocol that enables remote desktop control through any web browser running on the Chromium engine [5]. It prioritizes low bandwidth usage and ease of access, allowing easy server install on every major desktop operating system, and enables any web browser or mobile device to act as a client. By employing the VP8 compression format (Most commonly seen in internet video streaming), Chromoting is able to keep network traffic low by only sending a few full images of the server's screen as keyframes and transmitting motion with compressed partial frames [10]. These partial frames are calculated by the server using the difference between the previous and current frame, allowing the server to only send the difference between the two frames.

This save in bandwidth and processing done by the client makes it the perfect solution for Google's ChromeOS laptops coming out around the same time. These low powered machines were built to be laptops running on the power of a web browser with a link back to more powerful computers when the tasks were too great [22]. While the protocol works well for remote access, it is limited in certain applications by it's conservative approach to bandwidth usage and reliance on the web browser.

3.1.4 Secure Shell Protocol

The Secure Shell Protocol (SSH), is a protocol for securely transmitting data over an insecure network [23]. Usually used for remote server management, SSH has become the de facto standard for connecting to a remote computer when nothing more than terminal access is needed. Paired with the associated SSH file transfer (SFTP) or secure copy (SCP) protocols, SSH provides an incredible amount of flexibility for working with remote computers. It's largest and most obvious drawback is limited graphical support. While not a problem for it's traditional use-case, it does limit what high-intensity applications can be used with it.

One recent innovation in remote computing has been the introduction of Remote Development using SSH through the Visual Studio Code IDE. By opening multiple SSH connections to the host machine at once, Visual Studio Code is able to run the IDE's User interface on the client's machine, and simply send all the written code and commands through to the host machine, utilizing it's power to compile

code, run applications, and debug software [8]. This enables a development experience comparable to working on a local machine by only transmitting the data that is needed over the internet and constructing the visual user interface on the client. Because of this, high-intensity computational programs such as Artificial Intelligence training and simulation can be run on a powerful host machine and only need to transmit the output to the lower-powered client, but graphical applications are still limited.

3.2 Hardware Solutions

While Hardware solutions are not as popular or widespread as software solutions, there has been some innovation in the field that is worth considering. Hardware solutions usually focus on building a device that has just enough power to connect and stream data to a remote computer in order to leverage its power for the user to use.

3.2.1 Thin Clients

As discussed in Section 2.2, Thin Clients are a family of devices commonly seen in the corporate world to allow employees to access data centers and computing clusters from their desks. Due to their low cost and ease of deployment, they are often the device of choice to enable employees to access the full power of the business' computing infrastructure without needed to purchase a powerful device for each employee. However, since a typical thin client is a headless device, meaning it doesn't come with a monitor, keyboard, mouse, or other peripherals, they are often less portable as one might think. A thin client is usually set up once, and then left in that place for as long as a desktop would stay. Especially with laptop sales on the rise, thin clients do not have the benefit of being more portable than the alternatives.

3.2.2 Nvidia Shield and GameStream

The Nvidia Shield game console was Nvidia's first foray into the realm of remote streaming. Though it was focused on gaming and media streaming, it was one of the first attempts at using a portable device to stream intensive applications such as games from a host computer [1]. At the steep price point of $349, it definitely was enthusiast hardware for a device that focused on mobile and desktop gaming over running console games or demanding titles locally. But while the the physical device was praised for performance, battery life, and the experience of streaming games to the console, it didn't perform too well in terms of sales. It was still widely regarded as a technological breakthrough though, and Nvidia continued to develop the technology into a new device called the Nvidia Shield TV [7]. Focused on bringing the power of a gaming desktop to a TV, the Nvidia Shield TV dropped the idea of portability in favor of filling a different market focused on comfort. While this is no longer a valid device for the purposes of

FIGURE 3.1: An open Nvidia Shield displaying the home page [2].

this paper, the developments of the proprietary GameStream technology that powered the Shield and the Shield TV proved to be even more exciting than the physical devices themselves.

Though the GameStream technology that powers the Nvidia Shield devices is closed source and only officially compatible with the products Nvidia releases, its enticing power and potential drew a community to reverse engineer the protocol. In 2013, a group of students at Case Western Reserve University developed Moonlight, an open source implementation of the GameStream protocol [15]. This open source implementation allows the development of GameStream compatible clients that can run on other devices, from other computers to embedded ARM devices. This stands as a good starting place to develop the solution described in this paper.

3.3 Summary

TODO

3.4 Research Questions

While there are many existing solutions for accessing a computer remotely, many struggle to be performant enough or efficient enough to stream demanding applications to the client without compromising on the user's experience. This thesis will seek to build a solution that enables the use the power of a desktop computer from a mobile device in a way that is performant and responsive in response to the following questions:

1. **Hardware Feasibility:** *What hardware is needed in order to power a mobile device capable of acting as a client?*
2. **Hardware Cost:** *Is it feasible to construct such as device at a lower or comparable cost to existing solutions?*
3. **Communication Protocol:** *Does a protocol exist that is efficient enough to stream demanding applications to a client?*
4. **Software Application:** *Can a software solution be built to utilize this protocol in a manner that performant enough?*

Chapter 4

Developing the Hardware

TODO

4.1 Requirements

This section will explain the different constraints taken into account while developing the hardware portion of this project. First, the hardware must be performant enough to provide a positive user experience (4.1.1), and secondly the product must be cheap enough to justify it over similar alternatives (4.1.2).

4.1.1 Performance

When considering how much power needs to be incorporated in to the miniature computer, it is important to consider a number of factors, such as: having enough power to stream audio and video over the internet, not consuming too much power so as to require an expensive battery, have a fast enough connection to the internet to deliver data with the least possible latency, and having a reasonable footprint to be used as a mobile device.

Immediately, the device must be capable of decoding and rendering video that is streamed over the internet; a relatively expensive process that cannot easily be done with something such as a micro controller or even a single core CPU seen on Single Board Computers (SBC) such as the Raspberry Pi Zero. Though at the same time, it is important to not simply build a powerful system akin to a laptop that will consume more power. It makes no sense to built a powerful mini-computer when all it will ever be used to is to connect to another computer, and the added power draw would require a larger battery in order to stay operational for decent periods of time. Instead it should be cut down and distilled into something that can do it's job as efficiently as possible.

pic?

The second major performance requirement of the hardware is the ability to stream this data over the internet as quickly as possible. Because all of the data being sent to the device is live information from the host machine, the video and audio streams cannot simply be buffered in advance and played when it's ready like a recorded video. Instead, the data must be displayed as it is received so that the user can have a smooth experience without having to wait for their keystrokes or mouse movements to be registered. While Wi-Fi is growing ever quicker and more reliable, and should absolutely be included as a way to connected to the internet, the best way to ensure a stable and speedy connection is to use an Ethernet Cable. This will help the device work on a consistent connection when used at a desk, while also providing a way to connect while on-the-go.

This leads to the desire for a portable design that can be used both in a conducive environnement such as a desk or while mobile. The hardware should have the ability to be powered by a chord plugging into the device, or by an internal battery. This alone presents a number of difficult concerns, but those will be addressed later in this chapter. Such a requirement leads to a clear benefit of integrating crucial computers such as a battery and a screen into the hardware design, while leaving as many ports open such as ethernet and USB for the user to take advantage of.

4.1.2 Cost

Because this project consists of both a Hardware and Software component to make up a complete product, there must be reason for both to be developed. While the software has the benefit of being the foundation for the connection between host and client machines, there are already hardware solutions that would theoretically be able to run such software. Since the goal of this project is to save money by not needing to purchase an expensive mobile device to connect to a powerful desktop computer, there must be benefit in the hardware to warrant it's existence over repurposing something like an old laptop. The most obvious metric of this is the actual cost of the hardware. If the hardware can be produced at a high enough quality at a low enough cost to bring value over just a software solution, then it should be considered moving forward. Otherwise it may prove that the development of the software should focus on compatibility so the hardware can be provided by the end user.

4.2 Choosing Parts

oduce acronyms the first time 're used

s was introduced in 4.1.1, should it be done here?

When starting to choose parts for the hardware, there are many difficult questions to face. First and foremost, the hardware had to be built with the time and resources available for this project. Specifically, building a Single Board Computer from scratch would be completely infeasible, but the goal was also not to simply purchase an off the shelf SBC that was not tailored to the task at hand such as a traditional Raspberry Pi. Instead, a better suited solution would be to utilize a *Raspberry Pi Compute Module 4 (CM4)* which has "The power of Raspberry Pi 4 in a compact form factor for deeply embedded applications" [16]. Boasting an incredibly small size and streamlined I/O, there only way to communicate with the Compute Module was through two tiny mezzanine connectors as shown in Figure 4.2. With no other pin outs for anything such as power, HDMI, or USB, everything had to be built on a custom Printed Circuit Board (PCB).

This PCB would have to be designed so that the CM4 could be plugged into the board using the mezzanine connectors and rely on it for all inputs and outputs. This, however, is only the beginning of what is required to develop the product. Since the project would not use an off the shelf SBC like a regular Raspberry PI, the project would require only designing the PCB but also picking out all the components that would go onto the PCB, such as the USB ports themselves, the USB driver chip, and related components needed for operation. Because the CM4 is designed to be as low-powered and compact as possible, some features that would usually be expected of such a device, like the USB ports, are disabled by default and must instead be built by the board designer. While this helps greatly by removing extraneous components and features that aren't needed for this project, it also provides me with a great opportunity to learn exactly what parts are being used in the hardware.

s this sound sarcastic?

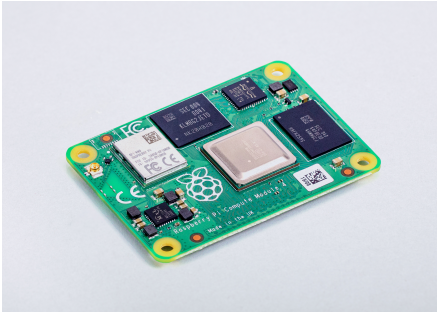

FIGURE 4.1: Raspberry Pi Compute Module 4, sizing 55 x 40mm

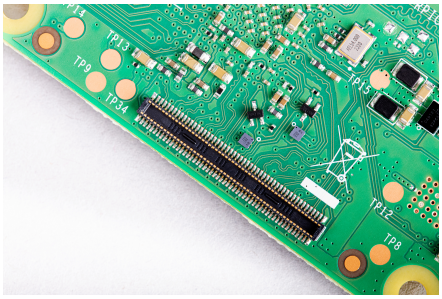

FIGURE 4.2: Close-up of the pin out, with a 0.04mm pitch between pins

An invaluable resource while designing the PCB was the Official I/O board for the CM4 [17]. Though the PCB board would still have to be custom built and components would need to be replaced due to the current ongoing chip shortage, it provides a great foundation instead of starting from scratch. Another massive benefit from working within the Raspberry Pi Ecosystem is their incredible documentation (re)move? and community surrounding the boards. Around the time this project was being started, the CM4 was just starting to get into the hands of makers and developers, and only a couple custom boards had actually been made by anyone other than Raspberry Pi themselves. Even though this leaves the community with less of a bulk of knowledge to work with, it has provided a sort of comradery with everyone helping each other learn the new Compute Module together. Members of the CM4 engineering team would even assist those on the forums to help them pick parts and design their own boards.

4.3 Prototyping

Whenever working with electrical projects such as this, it is always a good idea to build up prototypes before committing to a final design that will be produced at the highest quality. Usually breadboards and some breakout boards for Surface Mounted Devices (SMD) are a great way to develop a circuit without creating a full PCB; but due to the nature of the complex microchips required for this project as well as the dependence on the CM4's tiny mezzanine connectors, building a prototype using a breadboard is impractical given the time and resources available. For example, the USB controller chip only has 0.5mm of space between each of it's 64 pins (Figure 4.3), which would require a breakout board at least 7 times larger than the chip so that each each pin can be easily accessed by hand. Not to mention the difficulty in wiring up that many pins along with the other related components which would need their own breakout boards either purchased or custom-designed.

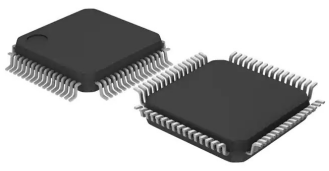

FIGURE 4.3: The USB controller, with 0.5mm between pins

Instead, the first board was built out in a piecewise fashion, allowing each section of the board a way to fail without interrupting the rest of the board. This allows each function of the board to be tested individually and in isolation so that changes

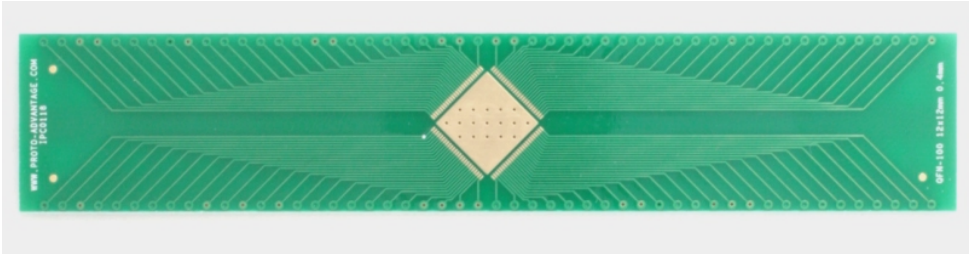

FIGURE 4.4: A breakout board similar to what would be needer for the USB controller used on the custom PCB

could be made incrementally without needing to worry about their integration with the rest of the board. For example, in order to build a system that would allow the board to be powered via USB or an internal battery, as well as charge the battery simultaneously, I would need to build a complex circuit consisting of parts neither I nor anyone I knew had used before. Since power is obviously a critical component of the board, A secondary way of feeding power directly the board was built to bypass all that circuitry in case it fails. In doing so, the board can still be powered and it's other features can be tested while a fix for that portion of the circuitry is developed for the next iteration.

vert first person to third person

4.4 Designing the Printed Circuit Board

TODO

4.5 Manufacturing the Circuit Board

TODO

Chapter 5

Developing the Software

5.1 Requirements

TODO

5.2 Potential Avenues

TODO

5.2.1 NVIDIA GameStream

TODO

5.2.2 Moonlight

TODO

5.3 Developing for ARM

TODO

Chapter 6

Evaluation

6.1 Testing Methodology

TODO

6.2 Responsivity and Latency

TODO

6.3 Performance

TODO

6.4 Quality

TODO

6.5 Summary

TODO

Chapter 7

Conclusion

7.1 Summary

TODO

7.2 Limitations

TODO

7.3 Future Research

TODO

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

Bibliography

[1] Eric Brown. *Nvidia Shield game console runs Android on tegra 4*. 2013. URL: https://linuxgizmos.com/nvidia-shield-android-game-console-shipping-soon/.

[2] Eric Brown. *Nvidia Shield: Shipped, praised, critiqued, dissected*. 2013. URL: https://linuxgizmos.com/nvidia-shield-reviews-and-teardown/.

[3] Steve Burbeck. "Complexity and the Evolution of Computing : Biological Principles for Managing Evolving Systems". In: *Complexity and the Evolution of Computing : Biological Principles for Managing Evolving Systems*. 2007. URL: https://evolutionofcomputing.org/Complexity%20and%20Evolution%20of%20Computing%20v2.pdf.

[4] Bryan S Burgin. *RelativeMouseMode*. 2013. URL: https://social.msdn.microsoft.com/Forums/en-US/ae516842-1e3d-4943-8144-1b225e74684e/relativemousemode.

[5] *Chromoting Build Instructions*. 2020. URL: https://chromium.googlesource.com/chromium/src/+/refs/tags/82.0.4058.2/docs/old_chromoting_build_instructions.md.

[6] *Covid-19 announcements archive*. URL: https://eoss.asu.edu/health/announcements/coronavirus/announcements-archive.

[7] Chris Daniel. *Shield TV brings the smart home to a TV near you: Nvidia blog*. 2017. URL: https://blogs.nvidia.com/blog/2017/01/04/shield-ai-home/.

[8] *Developing on remote machines using SSH and Visual Studio Code*. 2021. URL: https://code.visualstudio.com/docs/remote/ssh.

[9] Djjajah. *Clientron U700*. 2008. URL: https://commons.wikimedia.org/wiki/File:ClientronU700.jpg.

[10] *Google Chromoting - remote desktop management*. URL: https://www.miniorange.com/google-chromoting-remote-desktop-management.

[11] *How long will the chip shortage last?: J.P. Morgan Research*. 2021. URL: https://www.jpmorgan.com/insights/research/supply-chain-chip-shortage.

[12] J.O. Kephart and D.M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (2003), pp. 41–50. DOI: 10.1109/MC.2003.1160055. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1160055.

[13] Gad Levanon. "Remote work: The biggest legacy of covid-19". In: *Forbes* (2020). URL: https://www.forbes.com/sites/gadlevanon/2020/11/23/remote-work-the-biggest-legacy-of-covid-19/?sh=699b0767f590.

[14] L. Meyer. "5 Reasons Schools Still Need Desktop Computers: Despite the Growth of Mobile Learning, Desktops Still Play Important Roles in the 21st Century Classroom". In: *T.H.E. Journal Technological Horizons in Education* 41 (2014), p. 20.

[15] *Moonlight*. URL: https://moonlight-stream.org/.

[16] Raspberry Pi. *Compute Module 4*. URL: https://www.raspberrypi.com/products/compute-module-4/.

[17] Raspberry Pi. *Compute Module 4 IO Board*. URL: https://www.raspberrypi.com/products/compute-module-4-io-board/.

[18] *Solid notebook demand in consumer and education marks 2021 Q1 as the fourth consecutive quarter of double-digit growth in EMEA PC market, says IDC*. 2021. URL: https://www.idc.com/getdoc.jsp?containerId=prEUR147632021.

[19] Maxim Tamarov. *Chip shortage driving up PC prices, Wait Times*. 2021. URL: https://www.techtarget.com/searchmobilecomputing/news/252499402/Chip-shortage-driving-up-PC-prices-wait-times.

[20] *The VNC family of Remote Control Application*. URL: http://ipinfo.info/html/vnc_remote_control.php.

[21] *Understanding remote desktop protocol (RDP) - windows server*. URL: https://docs.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol.

[22] Linus Upson and Caesar Sengupta. *Next step in the chrome OS journey*. 2012. URL: https://blog.google/products/chromebooks/next-step-in-chrome-os-journey/.

[23] Tatu Ylonen. *RFC 4251 - the secure shell (SSH) protocol architecture*. Ed. by ChrisEditor Lonvick. 2006. URL: https://datatracker.ietf.org/doc/html/rfc4251.