Implementation of a Tetris Video Game Console via an Embedded Computer System

Capstone Design ECE 4440 / ECE 4991

Technical Advisor: Harry Powell

Daniel Mizrahi

Arjun Deopujari

Patrick Thomas

Nick Moon

Statement of Work

1. Daniel Mizrahi

My main contributions to this project involved the software development of the logic for tetris, coordinating the team, making design decisions, as well as giving some input on other areas of the project. Primarily, I began my contributions by independently creating a working version of the classic tetris game, written in C++ and interfaced through terminal. This version of classic tetris was about 300 lines of C++ code that utilized multithreading and special libraries which allowed the user to interface with the game through the linux terminal. Although this version of the software would need to be refactored in the future in order to be compatible with the MSP-432 and Code Composer Studio (CCS), the main objective here was to ensure that all the game logic was correctly developed and tested properly before using it in any capacity with the hardware. This development process took a considerable amount of time and effort which included researching all the nuances of the game, creating the game logic, testing and debugging, as well as coming up with a way that the game could be played to ensure correctness.

In addition to writing the game logic, I also made many contributions later in the semester which involved the testing and integration of the hardware and software. Once the code had been refactored, in order to be run on CCS with the MSP-432, there were bugs in the code that were introduced in the refactoring process. I worked alongside Patrick and Nick during this stage to find and correct these bugs to ensure that the game would run as intended. Finally, although we initially anticipated to have a dual display which would represent other information to the user during the game such as the score as well as the next piece queue, this turned out to be outside the scope of what was achievable given the time and resources constraints. Thus, I came up with the idea to represent the score in binary rather than as numbers, in order to compromise an intended implementation.

2. Arjun Deopujari

My main contributions in this project were the low-level machine-specific firmware code and the hardware schematic design (which I did in unison with Nick. For the hardware, I designed the interface of the MCU chip with the level-shifting circuits, buttons, and LED matrix. Prof. Powell and Barnes also helped in providing feedback to our design for which Nick and I made changes to the schematic. In order to do this, I worked with Patrick and Nick to read datasheets of components and select which ones met the design requirements to be able to interface well with the other components in the circuit as well as the timing of the software and game logic. I then assisted Patrick and Nick with the layout of the PCB on Ultiboard.

For the firmware, I was the first member to read the MSP432 datasheet and create starter code which allowed other members to write interrupt service routines, timer code, I/O code for input and output ports (to buttons and level shifting ICs), and code which controlled the system and timer clocks. The macros I created allowed other members to more easily write code which wrote logic levels to different outputs pins on the MCU. I then worked with Patrick in order to test the code for basic I/O functionality, timer interrupts, and clock frequency.

I was also in charge of scheduling group meetings to discuss progress and weekly priorities outside of the regular meetings with Prof. Powell and Barnes. I was unfortunately not able to work in the lab on final testing due to an illness which forced me to leave grounds for the last month of the semester.

3. Nicholas Moon

One role I had was to design the schematic of our project, implementing connections between the MSP432, the buttons, the power, the LED matrix display, and auxiliary hardware (voltage regulator, voltage translator, static shield, etc.) with Arjun. This involved making decisions with regards to port pin usage (like buttons all feeding into the same port), pull-up resistors on the button inputs (as suggested by Professor Powell), and test pin connections for DFT. I also made the hierarchical block for the buttons, and the Multisim component blocks for all of the Digi-Key parts we included in our project. This involved studying the datasheets not only for correct pin placement and assignment, but for understanding the specific inputs and outputs of each component and how we might connect them in order to achieve desirable functionality.

I also helped in the component selection, both the specific components as well as hardware we need in general, for example choosing the MSP432 over the MSP430 due to memory limits, and over the TIVA board due to deprecation of support.

I did extensive work on the Ultiboard schematics and layout for our first and our final designs on the PCB. I routed the wires for the components, and also edited the database to correct for errors made in the first design. I also managed submissions to freeDFM and managed contact with WWW Electronics to get our board soldered on multiple occasions.

Finally, I helped Patrick with hardware debug and test, and managed VirtualBench operation for the duration we were physically in the lab.

4. Patrick Thomas

Towards the beginning of the project, my role revolved around finding and selecting components on Digi-Key. Of special importance was selecting the specific display for the device and finding one that both fit our specifications and had enough documentation to work with. This by association involved some work with the schematics for our design and involved coordinating with those working on the Multisim schematics and ensuring that the write components and component footprints were being used. In addition to selecting components, I also read and verified the datasheets to help make sure that they would fit our application.

After we selected our components and made our Multisim schematics, I helped to make the Ultiboard schematic. I worked heavily on the first Ultiboard layout and establishing the component layout on the board, however I did not work as much on the revision of our board.

While we were waiting for our board to be manufactured, components to arrive, and our board to be stuffed, I ported the game logic code that Daniel made into C code that was friendly to our microcontroller. For example, all dynamic allocations were replaced with static ones, and usages of standard library C++ data structures were replaced with regular C structures and arrays. In addition to porting the game code, I also wrote the display driver code, button interface code, and the code that interfaced the game with the display. In the National Instruments lab, I was the one to flash the MSP and code while we were debugging the device.

Table of Contents

Abstract 7

Background 7

Constraints 8

Design Constraints 8

Economic and Cost Constraints 9

External Standards 10

Tools Employed 10

Ethical, Social, and Economic Concerns 10

Environmental Impact 10

Sustainability 11

Health and Safety 11

Manufacturability 11

Ethical Issues 12

Intellectual Property Issues 12

Detailed Technical Description of Project 13

Design Explanation 13

Design Decisions and Tradeoffs 20

Problems and Revisions 22

Components 23

Project Timeline 24

Test Plan 27

Final Results 31

Costs 32

Future Work 34

References 35

Appendix 38

Detailed Budget 38

Complete Multisim Schematic 40

Complete Code Listing 40

Table of Figures

Figure 1. Voltage regulator. 13

Figure 2. Buttons outside of hierarchical block. 14

Figure 3. Inside of the hierarchical block, “The Buttons.” 14

Figure 4. Header pins for the MSP432. 15

Figure 5. Logic level translator one. 16

Figure 6. Logic level translator two. 16

Figure 7. The display connector. 17

Figure 8. Basic Software Game Flow 19

Figure 9. Display connector. Image from Adafruit. 20

Figure 10. Copper Top (left) and Copper Bottom (Right) Layers of Final PCB Design. 21

Figure 11. Display “smearing” is evident in the picture from the one LED wide columns that erroneously show up in what should be blank areas. 23

Figure 12. Original Gantt Chart. 25

Figure 13. Revised Gantt Chart. 26

Figure 14. Hardware Test Plan Pre Launchpad (Left) and Post-Launchpad (Right). 27

Figure 15. Test setup in the National Instruments Lab. 28

Figure 16. Oscilloscopes connected to the 3.3V (black oscilloscope) and 5V (grey oscilloscope) test points on the header board. 29

Figure 17. Oscilloscopes connected to the test point ROTATE which verifies that the voltage across the rotate button is about 3.3V when unpressed and 0V when pressed. 29

Figure 18. Demonstration of pressing the rotate button. 30

Figure 19. Oscilloscopes are placed on the test points R1_VL and R1 which correspond to the first red signal coming directly out of the MSP and that red signal after the level shifter. 31

Figure 20. A game of Tetris being played, where the left shows the initial game state, and the right shows the game after having lost. 32

Figure 21. Complete Multisim Schematic 40

Abstract

The Tetris game console is a self-contained and interactive game featuring everyone's favorite and timeless video game: Tetris. Featuring a wired controller and an LED Dot Display, this game console provides a retro experience to all players, one predicated on fun. The physical game will be created using an LED Dot Matrix and played with a custom built controller integrated with the electronics and logic. Behind the scenes, an MSP432 stores the C code for the game of Tetris and controls both the state of all the LED lights and the inputs from the controller. This version of Tetris provides the user with a simple, yet elegant, physical interface, scoring elements, and a custom built controller to play the classic version of Tetris with. The project makes use of new advancements in computer microarchitecture over the past four decades, like the development of energy efficient embedded CPU architectures such as MIPS, SPARC, and ARM, which have both made embedded processors faster as well as more energy-efficient. These advancements now enable classic arcade games to be played on low-power MCU which saves energy costs for video game consoles. Finally, the scoring implementation and display serves as a learning experience for children to develop a background in binary counting systems integral to computer science and engineering.

Background

Tetris, developed by Soviet video game programmer Alexey Pajitnov in 1984, achieved widespread popularity after its release to the rest of the world on Nintendo's NES and Gameboy. Now a classic title, the simple block puzzle game has sold at least 202 million copies and no doubt been played by more than a billion people. We chose this project because it would offer a great balance between hardware design, software implementation, and the dynamic configurability enabled by the two. This project also seeks to create a fun experience enjoyable both between friends and as a solo experience, in order to alleviate stress created by the worldwide COVID-19 pandemic.

Previous work has been done in implementing Tetris on FPGA hardware [1] and on an AT89S52 Microcontroller [2]. Of course there is also the wide variety of game consoles, computers, phones and digital devices that the Tetris code has been ported to, including Nintendo's Gameboy as a battery powered game console, which loads the game logic and assets from an interchangeable ROM to display on a monochrome LCD display [3].

Our work, especially compared with past published implementations of Tetris, has the advantage of extensible configurability. We plan on implementing scoring/high-score functionality, multiple color display, incoming tetrimino functionality, and pausing functionality. However, our ultimate design will be dynamic in implementation, through judgements made in response to the limitations of working through the COVID-19 pandemic, a smaller time allotment, and a limited budget.

This project draws upon skills, experience, and knowledge from a myriad of past coursework. First, the ECE Fundamentals (ECE 2630, ECE 2660, ECE 3750) series will guide the PCB design, soldering work, and analog circuitry necessary for inter-component communication. Introduction to Embedded Computer Systems (ECE 3430) will give essential knowledge on the MSP430 controller programming and basic embedded software engineering skills we will need to implement to the game software and interface the software with external circuitry. Advanced/Real-time Embedded Systems (ECE 4501) and Operating Systems (CS 4414) will help in interrupt handler and semaphore implementation [4]. Previous game design and display work from FPGA Design (ECE 4550) will help inform our work on the Tetris game logic and frame buffering in this project.

Constraints

Design Constraints

One of the largest constraints for the entire project was the amount of memory needed to both represent the current state of the game as well as the current state of the display. Tetris, which is traditionally played on a 20x10 board, needs a single 20x10 array of bytes in memory. Each byte encoded the values of an enumerated type, and the enumerated type encodes the color of the Tetris blocks in that location on the board. In addition to this, we also need to map this game state to the display's current state. Our display was a 64x32 RGB display. Our display did not have a notion of color intensity (e.g. the red component for a LED was either on or off), and this simplified the color space to three bits per pixel. We then used the same enumerated type for the 64x32 display, meaning that we needed a microcontroller that could handle at least 64x32+20x10=2248 bytes of memory, not including additional data structures needed to encode more of the game state. A microcontroller that satisfies this requirement is the MSP432, thus this requirement guided our choice to use the MSP432P401R LaunchPad. This particular LaunchPad has 64KB of RAM, which more than satisfies this requirement. This choice added the additional constraints that our board has to interface with the LaunchPad environment and thus follow the Texas Instruments BoosterPack guidelines. Additionally, this meant that we are also required to use TI's Code Composer Studio to program and flash the MSP432.

Parts availability was mostly not a problem except for the LED dot matrix display. Options for RGB LED dot matrices that are large enough for our application that are also still in stock and production are very limited on Digi-Key, and no dot matrix that fits those requirements had a detailed datasheet. Since we were working with a relatively limited time frame, we could not choose a display that was out of stock as it would not arrive in time for us to complete the project. For the dot matrix that we eventually chose, it meant that we had to develop our own display driver as well as find a power adapter that could supply the 5V/8A maximally required by the display, thus introducing a power constraint. This was satisfied via a 5V/10A power supply that was also eventually used to power our entire project.

Another constraint from our choice of display was in the logic levels. The dot matrix display's driver integrated circuits operate at a 5V logic level while our MSP operates at 3.3V. This meant that we had to have a voltage level translator somewhere in our design for all 11 of the logic pins to the display. Since the MSP also runs off of 3.3V, we were also required to have a voltage regulator on the header board to regulate the 5V from the power adapter to 3.3V input voltage for the MSP.

The voltage level translators, voltage regulator, and transient voltage suppressors were extremely small and no one on our team had the skills or experience to solder such components to our header board. This required that we solder our board at WWW Electronics.

Economic and Cost Constraints

Since our project worked with relatively little asides from the display and the power adapter, costs were not too large of a problem for the scope of the project. However, if we had used a different LED dot matrix and went with our original plan to use two displays for the game, this would have cost about \$220 as compared to the \$40 that we actually spent on our single display. Our budget had a surplus of around \$160, so this would have gone over budget by approximately \$20.

External Standards

Since the MSP432's clock rate is somewhere in the range of 9 kHz to 3000 GHz, it is classified as an unintentional radiator (a digital device which contains signals which have frequencies in the radio frequency range) and is thus subject to 47 CFR Part 15 Subpart B.

The PCB that we designed is standard (not flexible, not microwave, etc.) and falls under IPC-2221 and IPC-2222 [5]. This is unavoidable but did not greatly affect how we proceeded to work on our project.

Tools Employed

We used National Instruments Multisim [6] and Ultiboard [7] in order to design the circuit schematic and PCB. In order to verify the PCB CAD files and ensure that there were no hidden errors in our schematics that Ultiboard missed, we used FreeDFM [8]for secondary verification.

All of the parts of the project were purchased through Digi-Key [9], and this involved using their online BOM manager and export tools. Skills had to be developed in order to efficiently and effectively find the desired component in Digi-Key. However, once this happened, the Digi-Key part filters proved to be very helpful. In order to combine both the Multisim BOMs and the Digi-Key BOMs, both Microsoft Excel [10] and LibreOffice Calc [11] were used. Microsoft Excel was also used in order to track the budget of the project.

The software repository for both the MSP firmware and the game logic is hosted through GitHub [12].The code in that repository was programmed through both Code Composer Studio [13] and Visual Studio Code [14] in C and C++. At one point the MSP432 became locked and needed to be factory reset, so we also used the MSP432 Security and Update Tool [15].

Ethical, Social, and Economic Concerns

Environmental Impact

All of the components used in the construction of the project are lead-free and RoHS-compliant. Almost all of the components are RoHS 3 compliant with the exception of the resistors and buttons used on the header board. Our components are also REACH-unaffected except for those same resistors and buttons as well as the MSP432 evaluation board. Our board is likely also lead-free and RoHS-compliant unless the solder used by either WWW or us is not lead-free. This makes the board itself much safer for daily usage, especially as the board in its current state has no enclosure. Additionally, the board is much easier to dispose of since it poses a much smaller to no lead-contamination risk and does not use many of the chemicals banned by REACH and RoHS.

Sustainability

The only sustainability concern related to our hardware version of Classic Tetris would be the consumption of energy. Given that we used a quite luminous board, this version of the game can consume quite a lot of energy when compared to not only other games of a similar nature, but also other versions of the game. If reducing such energy consumption was a design concern to us at the start we would have chosen a different means of displaying the game to the user, one in which the energy consumption is minimized. We quickly learned that the LED dot matrix used in our project is far too bright for a reasonable production version of the project and thus very energy inefficient for the given task.

Health and Safety

With respect to the way we designed the PCB and the utilization of our display, there are several small safety concerns. Firstly, because the display is connected directly to a wall outlet, any short on the board could cause the full power output of the wall to be sent through any electrical connections. Also, due to the high luminosity of the display, extended game-play could cause a strain to the user's eyes. Also, because there is no enclosure on the PCB controller and although it is unlikely, contact with any sharp components on the board could scratch or even cut the user.

Manufacturability

If this project were to be extended to be mass-produced and manufactured, several changes would have to be implemented in the design. Firstly and most importantly, the controller would have to be built and contained entirely in an enclosure. This is the only practical way to manufacture this device for the safety and comfort of the user. Likewise, the design of the controller would also have to be changed to include larger and more easily accessible buttons. Additionally, given the aforementioned enclosure, we would likely want to build one single board that performs all of the functionality of the PCB and MSP432 combined. Finally, the display would likely not be able to connect to a wall every time the user wanted to play, and thus a separate custom display would have to be designed in order to attach a battery pack to it. For convenience, this could all be designed and built in a single enclosure with a logic board, display, buttons and battery pack to power all the components. The way it is built right now, our Tetris game is not manufacturable by any reasonable means.

Ethical Issues

There are a couple ethical concerns. First, there is the issue of powering our device. Because the power used is purely for entertainment, it may be important to consider the source of electricity used to power our Tetris game console. Using the device in an area where fossil fuels or natural gas are the main power supply will have a negative impact on climate change, so being able to use sustainable energy supplies would help alleviate this issue.Secondly, there is the issue of potential video game addiction. Tetris is a game notorious for the ludic concept of "flow", which is the balance of difficulty and skill that can lead to players sinking into an addictive cycle. Thirdly, Carpal Tunnel Syndrome can cause intense pain for people who use computers or controllers for too long, and our controller is not ergonomically designed. Finally, we failed to consider the physically disabled when designing the system, so those with eyesight problems, or motor skill handicaps will struggle to play our game.

Intellectual Property Issues

Given that the goal of our project was to create a physical hardware version of the Classic Version of Tetris, which has several patents that protect its intellectual property, patenting our project would more than likely not be possible. Our design decisions were based on creating essentially a clone of the existing Tetris game. This clone includes copying specific details from Classic Tetris such as the 10x20 sized game board, specific geometric tetromino pieces, as well as the game rules, dynamics, and scoring mechanisms. For instance, The Tetris Company, who is the exclusive licensee of Tetris Holding LLC, has sole access to the following patents: US Patent Numbers: 9517412, 9138641, and 8313370. Collectively these three patents essentially protect, "Systems and methods for manipulating an object include a display for displaying an object where the object has a geometric shape and is arranged in a first orientation of the geometric shape. The display also displays at least a second orientation of the geometric shape in proximity to the object. The system includes a user interface for receiving a user input to select the second orientation of the geometric shape. A processor, in communication with the display and user interface, determines one or more possible orientations of the object including the second orientation and arranges the orientation of the geometric shape of the object to match the selected second orientation” as well as “Video game systems and methods provide one or more software-based skill adjustment mechanisms for video game systems to adjust a difficulty level of the video games based on an existing skill level or game play of a player. The difficulty level of the video games is adjustable to correspond to an existing skill level of the player for removing digital blocks from a digital matrix, clearing digital complete horizontal lines from the digital matrix and/or avoiding game ending conditions. The difficulty level of the video game may be adjusted to facilitate skill-building game play for the player. The difficulty level of the video games is adjustable during game play that is approaching or leading to a game ending condition via software for one or more primary and/or secondary skill adjustment mechanisms.” Additionally, given that Classic Tetris was the first game of its time, the specifications in the patents pertinent to the rules and game specific design are independent claims due to their novel nature. Likewise, the claims pertaining to for instance, “Systems and methods for manipulating an object include a display for displaying an object where the object has a geometric shape and is arranged in a first orientation of the geometric shape” are dependent because there are more than likely other games patented with geometric shapes, but not ones that are, “arranged in a first orientation of the geometric shape”. As previously mentioned, not only does our game have the same game elements, but also the same rules set or at least a strict subset of Classic Tetris. Therefore, in light of these patents and our blatant infringement of them, due to creating a clone of Classic Tetris with hardware, our project would certainly not be patentable.

Detailed Technical Description of Project

Design Explanation

First, power comes from an 5V/10A AC/DC wall adapter and onto the header board through a DC barrel jack. Since we want to power the MSP with the wall adapter too, we have to use a voltage regulator to step the voltage down to 3.3V. This is shown in Figure 1

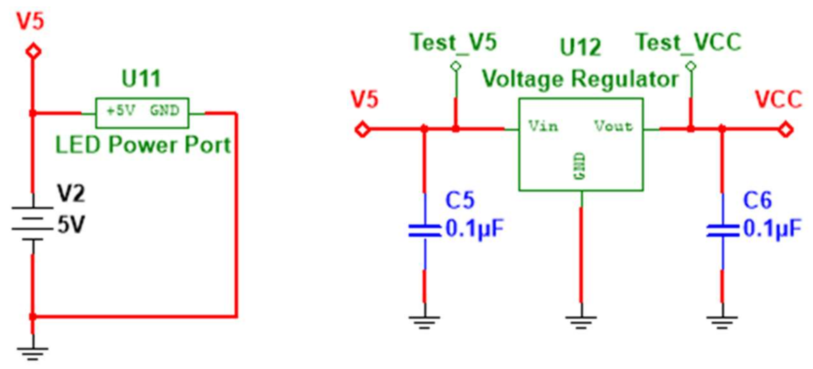

Figure 1. Voltage regulator. The subcircuit on the left represents the power coming from the wall adapter, named "LED Power Port" since the same adapter powers the LED dot matrix. On the right you can see the voltage regulator with test points and filtering capacitors.

According to the datasheet for our voltage regulator, we placed capacitors between the input to ground and the output to ground. Towards making a testable and verifiable design, we also placed test points immediately before and after the voltage regulator to be able to ensure that we are receiving 5V and outputting 3.3V with the voltage regulator.

After the input stage comes the buttons, where we take input from the user and subsequently inform the game logic with those inputs. This is shown in Figures 2 and 3. For our design we chose to use active-low buttons with pull-up resistors. Each of the pull-up resistors is 10kOhm. We chose this value since it is a happy medium from providing a small enough resistance that it is insignificant compared to the button when it is not pressed. According to the datasheet for our chosen buttons, the contact resistance is 100MOhm, which is much greater than 10kOhm as desired. Additionally, when the button is pressed, it has a contact resistance of around 100mOhm and requires a minimum load of 10uA. With our chosen resistors, the load on the buttons are 3.3V/10kOhm=330uA, which satisfies the conditions for the buttons and means that the voltage across the resistor is 0V when not pressed and 3.3V when pressed as desired.

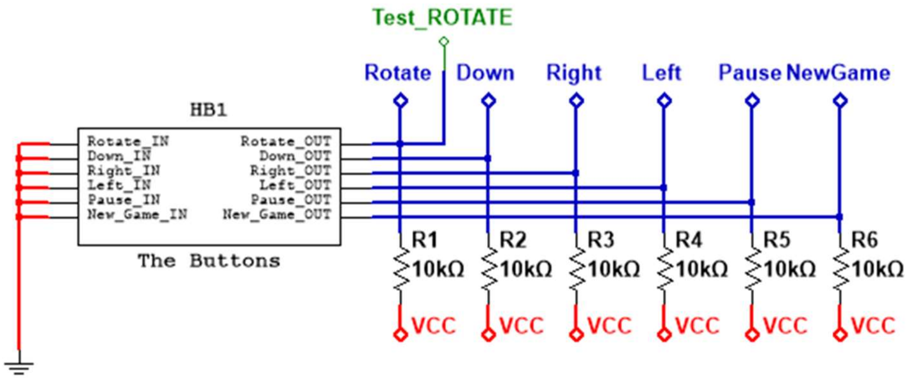

Figure 2. Buttons outside of hierarchical block. Obverse that there is a test point coming out of Rotate; this allows us to easily test one of the buttons. All of the on-page connectors other than VCC connect to pins on the MSP. We also used through-hole resistors. Those allow for easy measuring of voltages across those resistors and enable us to test each of the buttons.

Between all of the buttons and ground is a transient voltage suppressor. This is to protect the rest of the circuit and the MSP from static discharge since users of the device will be touching the buttons, which are soldered directly to the circuit board and not behind an enclosure of any type.

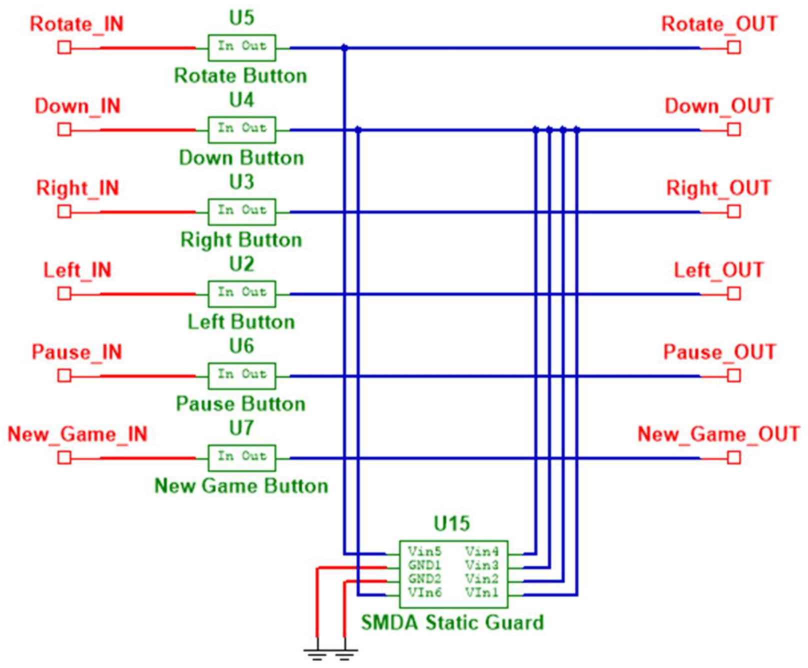

Figure 3. Inside of the hierarchical block, "The Buttons." Each of the buttons is a SPST, surface-mount, normally-off button.

After the buttons comes the microcontroller, the MSP432P401R. We only needed to pass our signals from the buttons to the MSP (represented in blue in the figure below) as well as take outputs from the MSP to the logic level translators (represented in green). In addition, we also passed power from the wall adapter to the MSP (represented in red). We chose the MSP for the reasons explained in the *Design Constraints* section: that is, we needed a microcontroller that had enough memory for our purposes. Since the MSP432P401R evaluation board has 20 pins on either side arranged in two rows of ten, we needed two 20-pin connectors to interface our header board with the MSP. This is represented via U8 and U9 in the figure below.

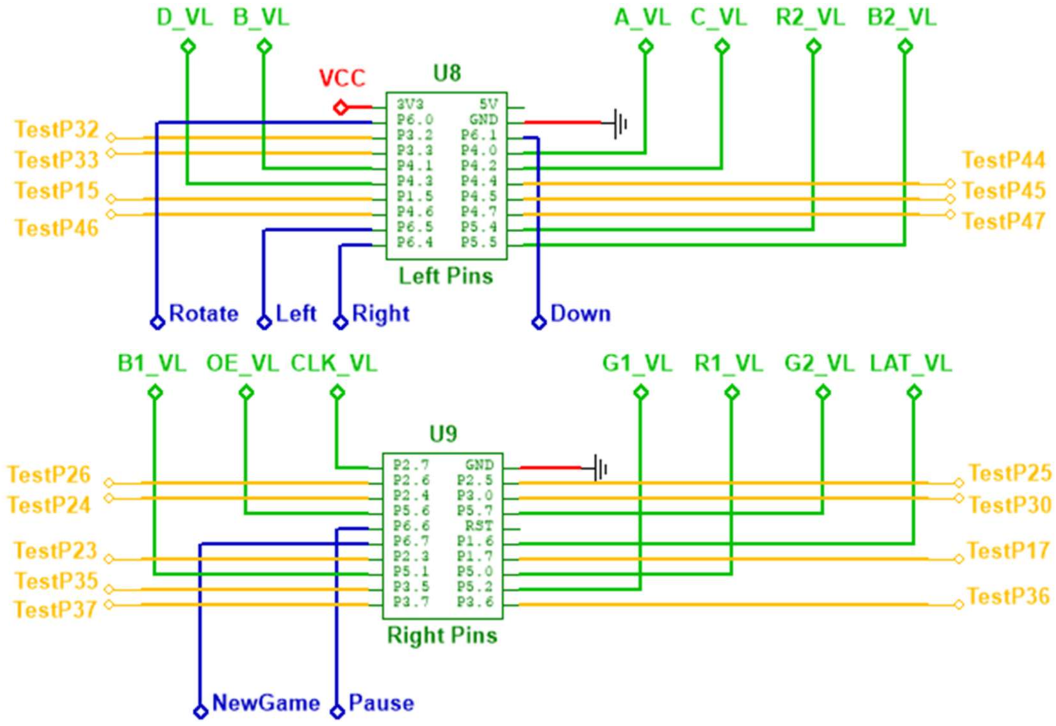

Figure 4. Header pins for the MSP432. In the figure, all of the test points (yellow) were not in the final design.

Once we have the outputs from the MSP to control the display, we need to translate the logic level to 5V. This is done with two unidirectional, eight-channel logic level translators. We specifically chose a unidirectional one since we only wanted to output to the display. We also made sure to choose a translator that could handle our input voltage range and output voltage range. We wired the enable for our translators to VCC to leave them always on. This was because we had no use for disabling them through software as we could simply clear the display if we wanted to turn it off. Towards testability we placed test points on the 3.3V and 5V sides of both the clock signal and R1. We chose these two signals to include both a logic signal and control signal. We also assumed that the clock signal would be the most rapid of these signals and the most likely to run into issues regarding data transmission rates when communicating with the display. The test point allows us to monitor the clock signal with an oscilloscope when testing the game to ensure that the signals are getting fully translated. We added capacitors like C3 and C4 to conform with the design recommendations from the data sheet for the translators.

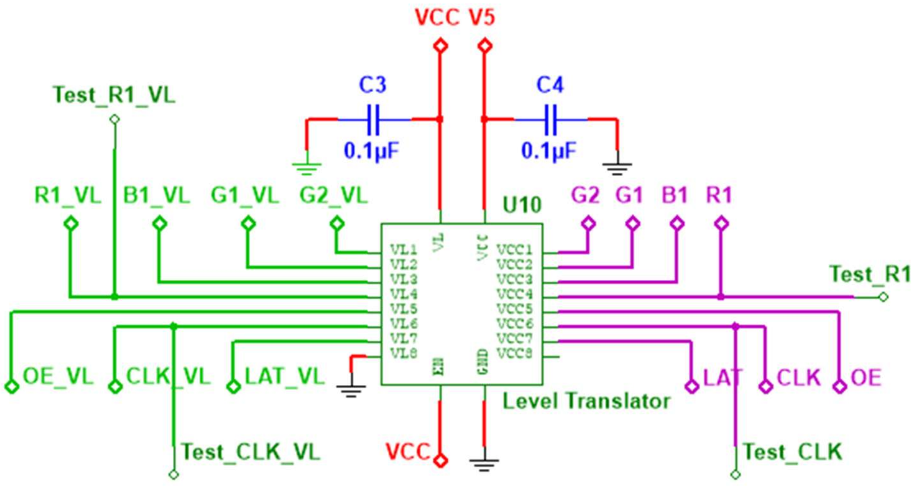

Figure 5. Logic level translator one. Note that one of the unused channels is placed on this device. The unused channel, VL8, is grounded such that it does not float and cause unexpected behavior.

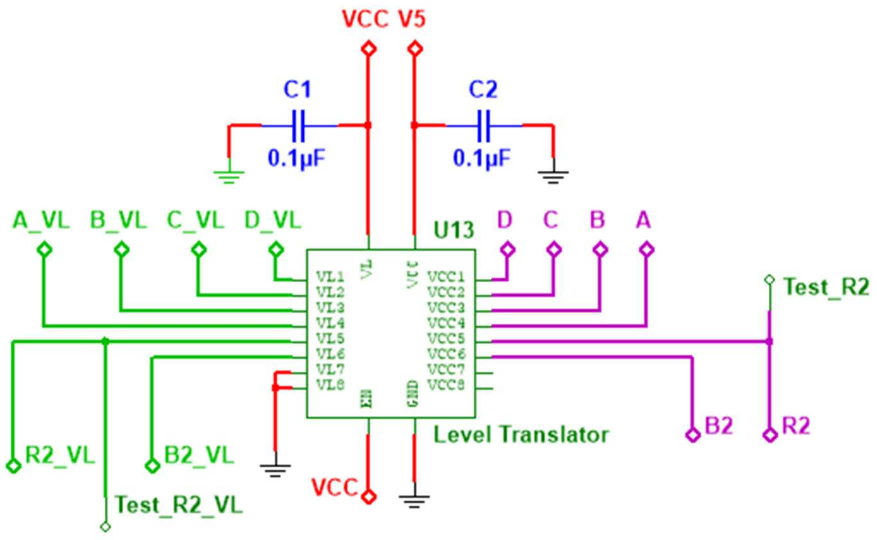

Figure 6. Logic level translator two. Note that the other two unused channels are on this device.

After the translator was the final component on the header board between the display: the display ribbon cable connector. This is shown in Figure 7 This was simple in that the signals from the translators directly went to the connector as well as some ground connections.

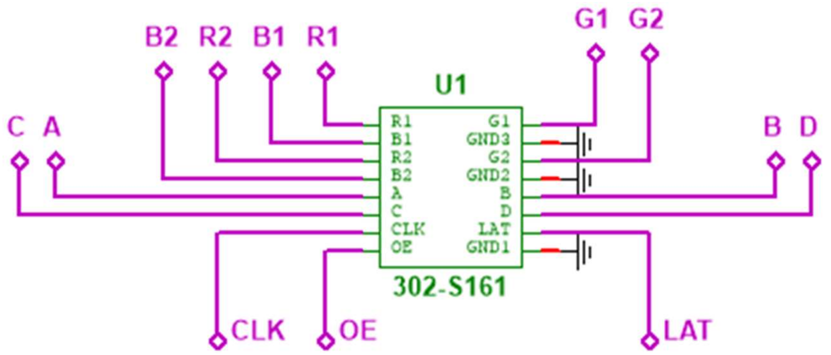

Figure 7. The display connector. The display's connector required three ground signals, thus we have GND1, GND2, and GND3.

The underlying logic of our version of Tetris is quite simple and can be seen below in the *Basic Software Game Flow*. Upon starting the game, a Tetris piece is spawned. Once spawned, the piece will begin falling down one block of time based on a time control that is correlated with the user's progress in the game thus far. Between every time a Tetris piece is waiting to fall down, the user has an opportunity to give user input which includes, moving the position of the piece, the orientation of the piece, or placing the piece on the board. If the user action is possible the complete, it will be, otherwise, nothing happens. Once a piece is eventually placed, either by user intervention or by the game placing it for the user, the program will access whether this placed piece has completed any full rows in the game board. If it has, the appropriate lines will be removed, the lines above will shift down, scoring will be updated, and the cycle will continue from spawning another piece. All the while, internal game logic will update parameters such as the fall down speed as the user progresses as well as if at any time a piece is placed at the top row of the game, the game will conclude. Finally, all other implementation specific questions pertaining to software design decisions can be found under *Complete Code Listing* in the "Final Version" link. This link contains the full repository of all code used to create our version of the game. Likewise, there is also another full repository called "Initial Terminal Version" which is a Linux Terminal version of the software which was created for testing and debugging purposes.

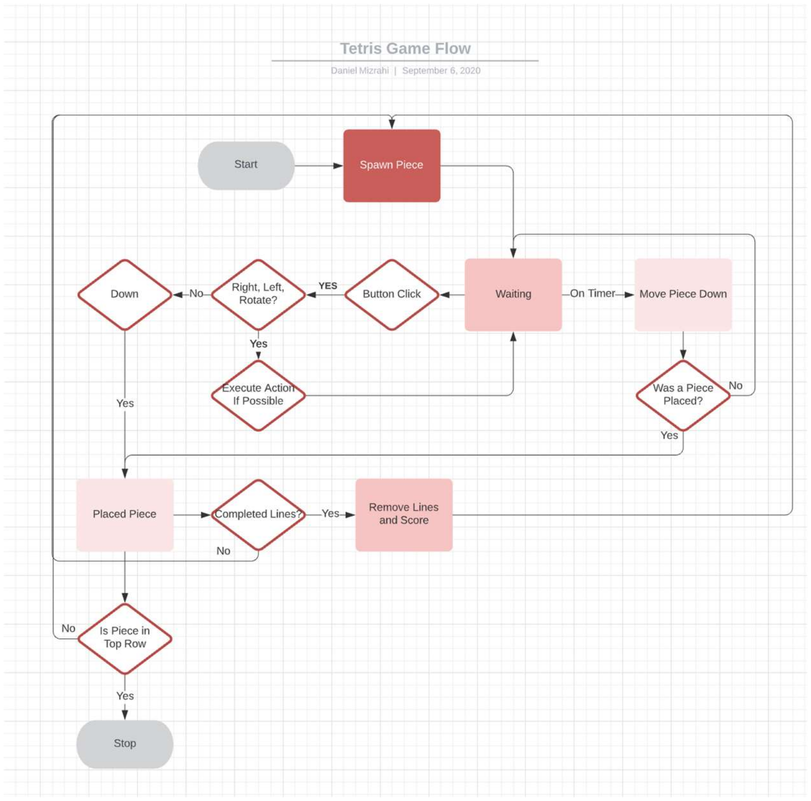

Figure 8. Basic Software Game Flow.

The code communicated with the display through a relatively simple interface. A labelled picture of the inputs to the display is below in Figure 9 All signals to the display are active high and falling edge triggered. The falling edge trigger only matters for the clock and latch signals. The general process was to translate the game status to the display array by simply scaling the game board to be three times larger and centered. Since the display writes to two rows at once, we have to keep track of both the start row and the start row offset by 16 rows. We first set the row address through the controls A, B, C, and D to the display. We then write color data to R1 through B2 for each of the two rows that are currently being written. Toggling clock both saves the data and increments us to the new LED in the row. We repeat this process until the end of the row. There, we set output enable to low to disable it, we toggle latch to fully save the data for the row, and then we re-enable output. This is repeated for each of the rows.

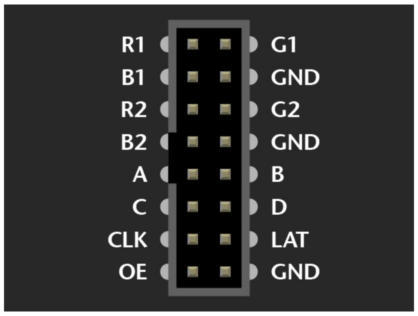

Figure 9. Display connector. Image from Adafruit [16]**.**

Design Decisions and Tradeoffs

Since the logic level voltage of the display and the MSP were different, we needed a way to translate the 3.3V output of the MSP to the 5V required by the logic circuit on the display. We only needed a unidirectional level shifter, and since we needed 13 outputs to be shifted, we eventually chose the Maxim Integrated MAX3008EUP+ logic level translator. This device gave us a more than adequate data rate, could easily handle voltages in the range that we required, and also had eight channels per device. We then needed two of these devices in all to let us interface with our display. This resulted in some wasted channels (three in all).

One of the most critical components to our design was the display. We wanted an RGB LED dot matrix capable of handling at least a 20x10 (since that is the dimensions of a standard Tetris board [17]) and has some wired interface similar to I2C or SPI. These constraints alone narrowed our selection to only two displays on Digi-Key, as many of the displays are either not in production anymore, interface through Bluetooth, or are meant for special applications and thus have very unique features (some displays were flexible). These were the Adafruit 2278 [18] and the Lumex LDM-6432-P5-BLE4-1 [19]. Neither of these displays had extensive documentation of detailed datasheets, but both are roughly the same size and are a 64x32 grid of RGB LEDs. The Lumex display, however, was interfacable with UART and BLE and was seemingly an already fully-developed smart display with its own driver and special software to be remote controllable through smartphone apps and desktop applications. This display at the time also cost \$108.53; this was roughly three times the price of the \$39.95 Adafruit display. The Adafruit display, however, did not have a well-defined interface or any detailed datasheet with a timing diagram. Despite this, we eventually chose the Adafruit 2279 64x32 dot matrix display [20], a slightly smaller version of the Adafruit 2278. This was because the Adafruit 2278 went out of stock before we could purchase it. Since the Adafruit display is quite popular among hobbyists, there were many graphics drivers online and the results of those who already reverse engineered the display, meaning that we did not have to fully reverse engineer the display.

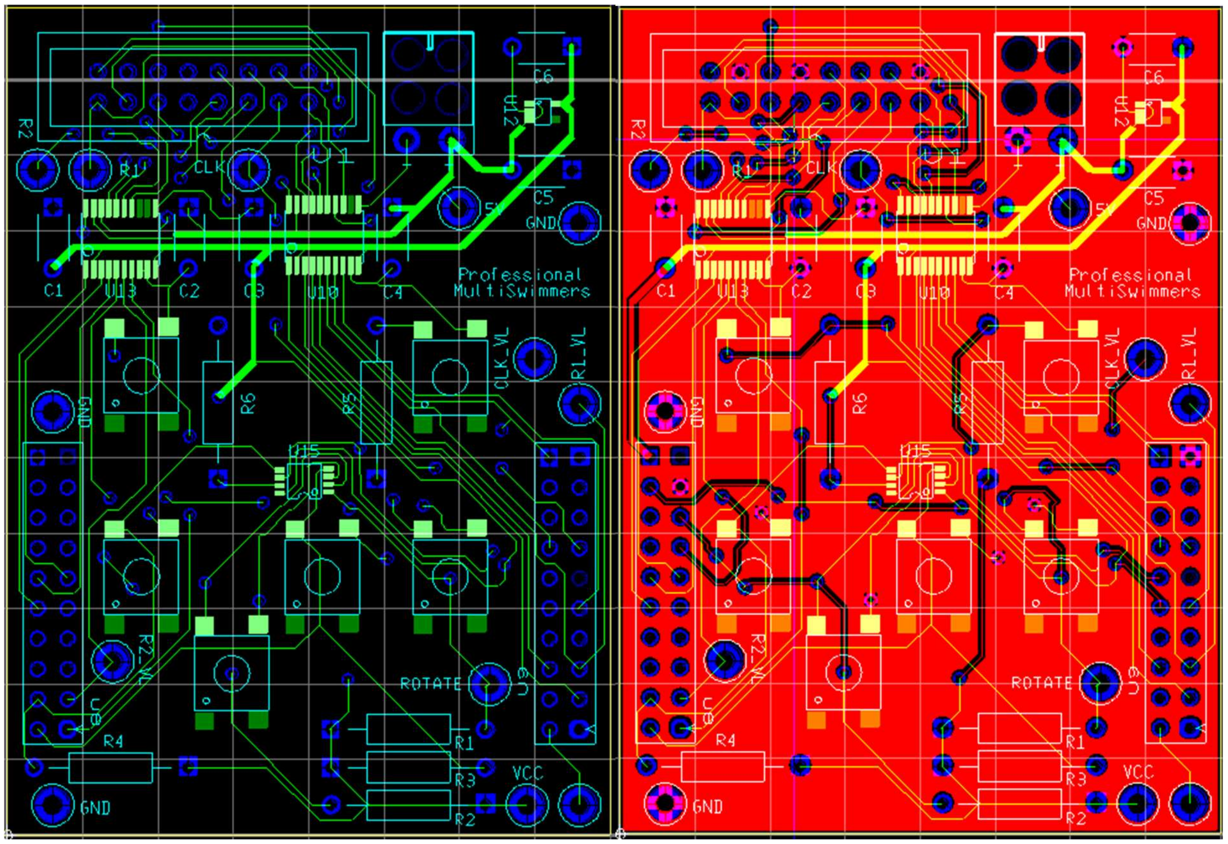

Figure 10. Copper Top (left) and Copper Bottom (Right) Layers of Final PCB Design.

Figure 10 shows the final layout of the PCB. A couple of layout decisions were made that were critical to our design. First, the buttons were placed as they were mostly due to time constraints on getting the PCB manufactured and soldered, especially as our first board didn't work and we were prioritizing functionality for the final revision over ergonomics. Given time, we would have liked to have the buttons on a separate PCB with connections to the logic PCB to separate the user handled layout and behind-the-scenes electronics. Our second big decision was including key test points. While they made routing the PCB much harder, they allowed us to easily test the digital and analog portions of our PCB in such a way where we could identify and isolate errors in hardware. Finally, a decision that was made in our layout that greatly simplified power access was using the wall power adapter as the power source for the LED dot matrix, the PCB electronics, and the MSP432. This allowed us to drive enough power through our board to maintain a clock speed on the MSP432 high enough to get a healthy refresh rate on our LED matrix.

Problems and Revisions

Our first design fabricated into PCB was mostly correct, however there were a couple of big issues that prevented us from being able to run our project. First, WWW Electronics soldered the female connectors to the MSP432 on wrong, which was a quick revision and revisit to the site. After that, the logic level translator pins were reversed on our Ultiboard component block, which led to incorrect wiring and nets when importing from the Multisim schematic and made the first revision of the board unable to communicate with the display. Additionally, the static guard component layout was never updated in Multisim or Ultiboard after finalizing the correct component to order from DigiKey, so the pin layouts were incorrect on that end too. In effect, these problems not only caused our PCB to function incorrectly, but it also caused magnified power consumption that permanently scarred our MSP432, leaving it barely functional. This meant that we wouldn't even be able to test the code while waiting for the new parts to ship in. However, once the corrections were made and a new MSP432 ordered, the board worked with no error or additional revisions to the design needed.

One issue that stopped our progress temporarily occurred when we tried to increase the master clock speed on the MSP432 to the maximum, 64MHz, in order to improve the display characteristics. This seemingly triggered some sort of protection mechanism on the MSP a left it unable to be flashed or communicated with through ordinary means. The MSP also would only give a single error saying that we needed something other than an evaluation board to use a clock speed that high. TI's documentation suggested that we used the MSP432 Security and Update Tool [15] to reflash the MSP, but this tool did not seem to help us. Eventually we rectified the problem by using a hidden function in Code Composer Studio to entirely wipe the memory of the MSP432. This erased whatever was causing the error and reset the clock settings and allowed the MSP to be flashed once again.

Once problem we also had was with the longer display cable that we ordered. We ordered this cable [21] since we did not expect the display to come with a cable, and we specifically chose this once since it fit the display's connector, is quite long, and would allow us to more comfortably use the MSP as the handheld input controller for the game. This cable, however, seemed to lose connection based on how the cable was physically arranged and generally had a poor connection. This would cause a display bug on our display where a single pixel would be smeared about 8 LEDs away from its desired location. This is shown in the figure below. This problem was solved by using the spare display ribbon cable that came with the display.

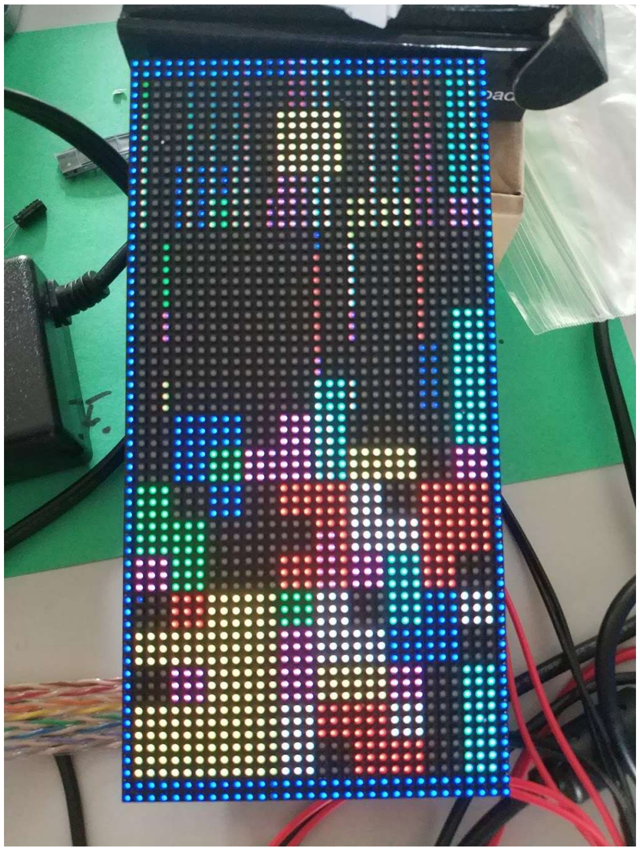

Figure 11. Display "smearing" is evident in the picture from the one LED wide columns that erroneously show up in what should be blank areas. The problem was eventually traced to the display cable that we were using, shown in the bottom left corner of the image.

Components

- Adafruit 2279 64x32 RGB LED dot matrix [20]
- Adafruit 5V/10A wall adapter [22]
- 2x MSP432P401R LaunchPad [23]
- Tensility International Corp. 2.5MM DC barrel jack female [24]
- 3M IDC ribbon cable [21]
- *Revision 1*
    - 1x CM1213-06MR transient voltage suppressor [25]

    - 6x B3FS-1000P tactile push button [26]
    - 12x Keystone Electronics 5011 test points [27]
    - 302-S161 connection header for display cable [28]
    - 6x CFR16J10K 10kOhm resistors [29]
    - 6x C315C104M5U5TA7303 0.1uF capacitors [30]
    - 2x MAX3008EUP+ logic level translators [31]
    - MIC5365 3.3V voltage regulator [32]
    - 2x M20-7831046 20 pin header for interfacing with MSP432 LaunchPad [33]
    - 839-1144-ND DC barrel jack female to wire leads [34]
    - CA-2185 DC barrel jack male to wire leads [35]
- *Revision 2 (final design)*
    - 1x CM1213-06MR transient voltage suppressor [25]
    - 6x B3FS-1050P tactile push button [36]
    - 12x Keystone Electronics 5011 test points [27]
    - 302-S161 connection header for display cable [28]
    - 6x CFR16J10K 10kOhm resistors [29]
    - 6x C315C104M5U5TA7303 0.1uF capacitors [30]
    - 2x MAX3008EUP+ logic level translators [31]
    - MIC5365 3.3V voltage regulator [32]
    - 2x M20-7831046 20 pin header for interfacing with MSP432 LaunchPad [33]
    - 839-1144-ND DC barrel jack female to wire leads [34]

Project Timeline

For our original Gantt Chart, we allocated four weeks for programming the game logic. Three weeks for creating the testing logic, three weeks for testing/debugging the PCB, six weeks for the LED board, five weeks for the game controller, and four weeks for integrating hardware and software.

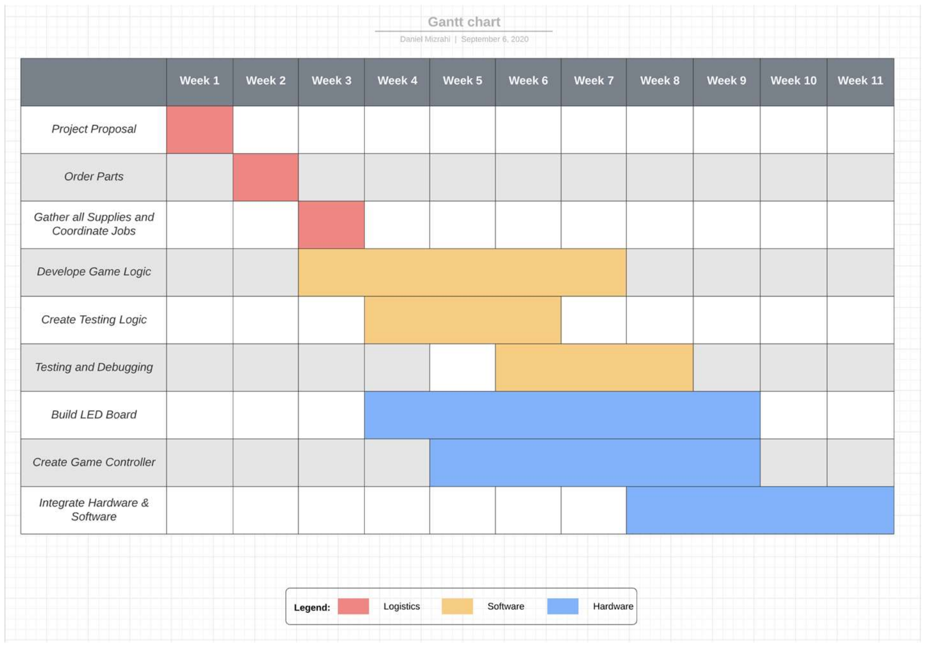

Figure 12. Original Gantt Chart.

We completed the initial Tetris game logic much quicker than we had anticipated, so our timeline was revised to allow more time for testing and assembly of the hardware and software as they come in and come together. Because the semester had no breaks or vacations, we worked weekly to complete these goals and hoped to be well within the bounds for the time allotment of the semester. In terms of the structure of the Gantt chart, a lot of changes in the hardware design have led to shifts in aspects of completion, especially with the controller housing both the MSP432 LaunchPad and its header board. Additionally, because we used a premade LED dot matrix instead of designing our own display from scratch, a lot of time soldering and driving a custom LED display has been removed.

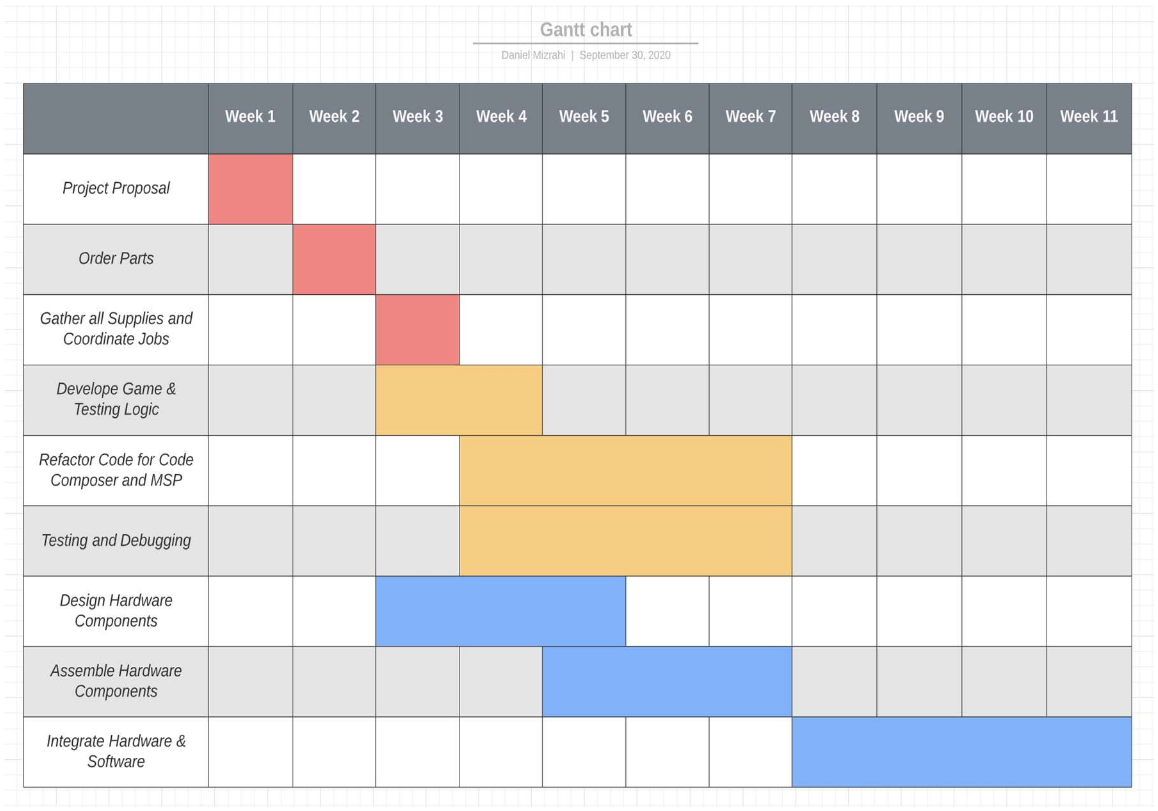

Figure 13. Revised Gantt Chart.

In summary, we would say that the timeline of the Gantt chart was very well adhered to. The quick development of the initial game logic, by Daniel, in the first couple weeks of the semester would allow the rest of the team to work on the design and layout of the PCB. Later, while Daniel was fixing all the bugs in the code and finalizing the software, Arjun, Nick, and Patrick worked tirelessly on the design and layout of the PCB. Also, as the team came closer to finishing the PCB design, Arjun and Patrick were able to work in parallel to refactor the code in C while Nick sent out the first iteration of the design. Later, when the first iteration came back, Nick, Patrick, and Daniel did initial hardware testing of the board while Arjun worked more on refactoring the score and Patrick did more research into the display driver code. Moreover, given this cushion of time for the hardware we were even able to iterate on our PCB after we ran into some design issues. Additionally, because we used a pre-built LED Dot Matrix, we saved a great deal of anticipated time allocated to building the game board. Lastly, once the second iteration of the board came back, all the team members met on several occasions to connect the hardware and software and eventually complete the project with a respectable amount of time remaining in the semester. However, it is noted that the debugging and hardware-software integration was done in series with the revisions to the second iteration of the board. Because the final software could not be tested until the revised board was completed, several group members were blocked for a week or two. Nonetheless, almost all of the tasks were able to be parallelized and this is why the project was able to be completed on time and without a great deal of last-minute work.

Test Plan

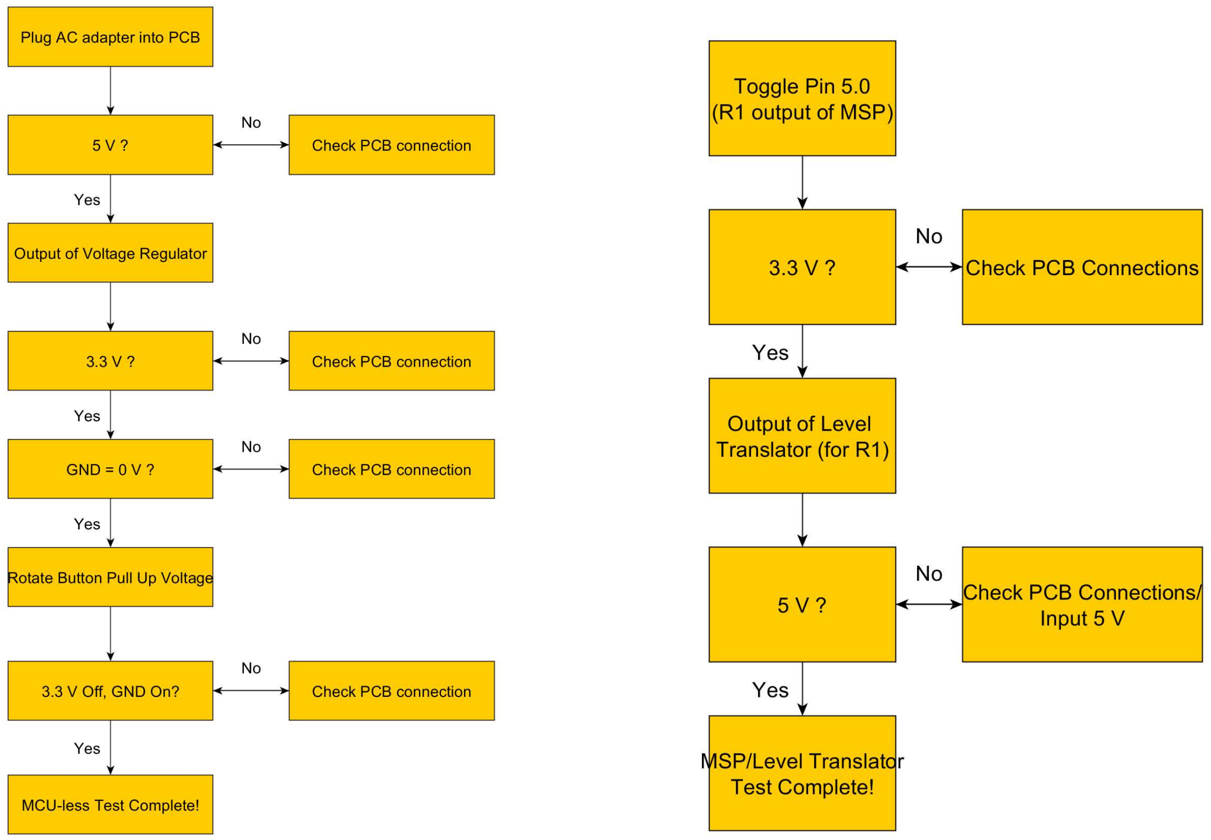

Figure 14. Hardware Test Plan Pre Launchpad (Left) and Post-Launchpad (Right).

Figure 14 above shows our test plans as produced for the midterm design review. Because we ended up getting our entire PCB soldered by WWW Electronics, we weren't able to solder and test incrementally, but we were still able to test starting with power supply and working through the entire PCB before attaching it to the launchpad. In fact, this testing helped us identify problems in the layout of our PCB, which we were then able to correct and rectify. Being able to separate our testing into pre-MSP432 and post-MSP432, as well as Hardware vs Software, we could contain issues in our hardware or software design to specific parts of our design.

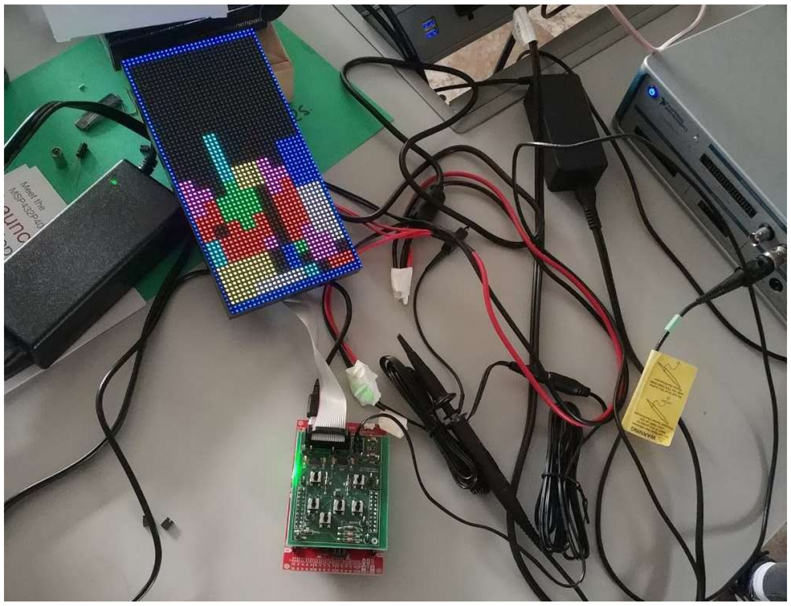

Figure 15. Test setup in the National Instruments Lab. On the right is the VirtualBench used with oscilloscopes connected. Out of picture is the laptop used to display the signals from the scope.

To begin with, we would plug the power source into a breadboard to verify that the power supply was producing 5V, shown in Figure 16. Then, we would connect this to our PCB, and check that the output of the voltage regulator was 3.3V, shown in Figure 16. We then also tested the voltage across the six buttons when they weren't pressed, and read 3.3V as well, as shown in Figure 17. Then, as shown in Figure 18, we would press each button, and verify that it would turn to ground.

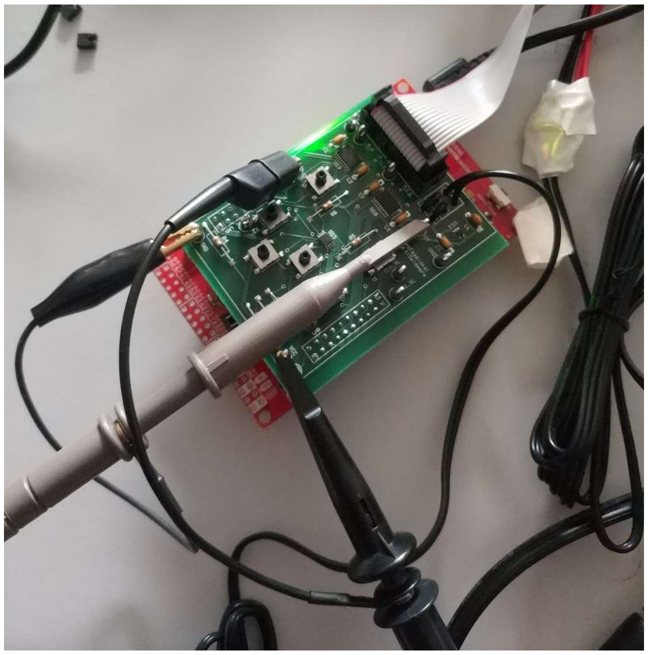

Figure 16. Oscilloscopes connected to the 3.3V (black oscilloscope) and 5V (grey oscilloscope) test points on the header board. This verifies that the power supply is working as intended, the voltage regulator is functioning as intended, and that there is less of a chance for there to be any shorts on the board.

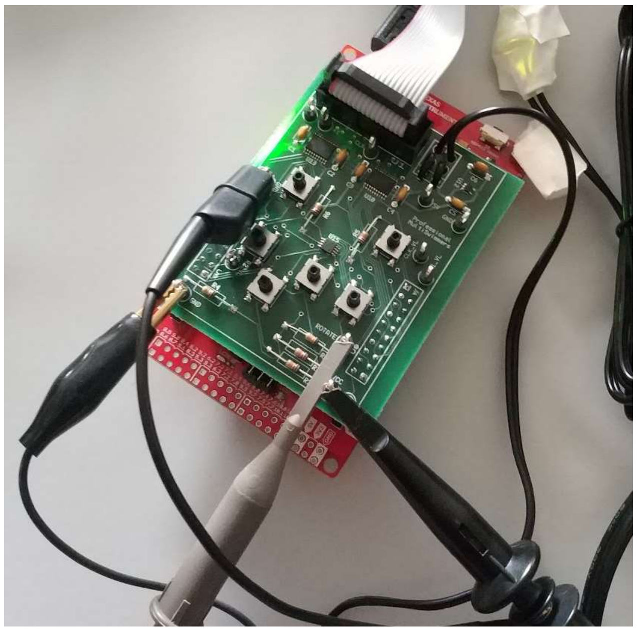

Figure 17. Oscilloscopes connected to the test point ROTATE which verifies that the voltage across the rotate button is about 3.3V when unpressed and 0V when pressed.

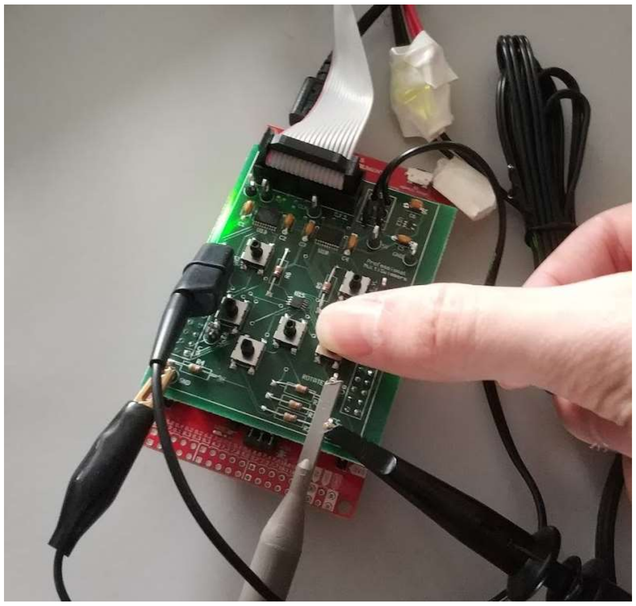

Figure 18. Demonstration of pressing the rotate button. We then expect the voltage across ROTATE to be 0V.

Once we verified the power and button subcircuits, we then verified the level shifters. We did this by making a simple test program for the MSP432 that toggled the R1, R2, and CLK signals to the display. Each of these signals had a test point on the 3.3V logic and 5V logic sides of the board. We placed breakpoints around this program to be able to manually step through it easily. We then placed oscilloscopes one the low and high voltage sides (3.3V and 5V) of the level shifters for each of R1, R2, and CLK and verified that changing the 3.3V side would result in a subsequent change on the 5V side.

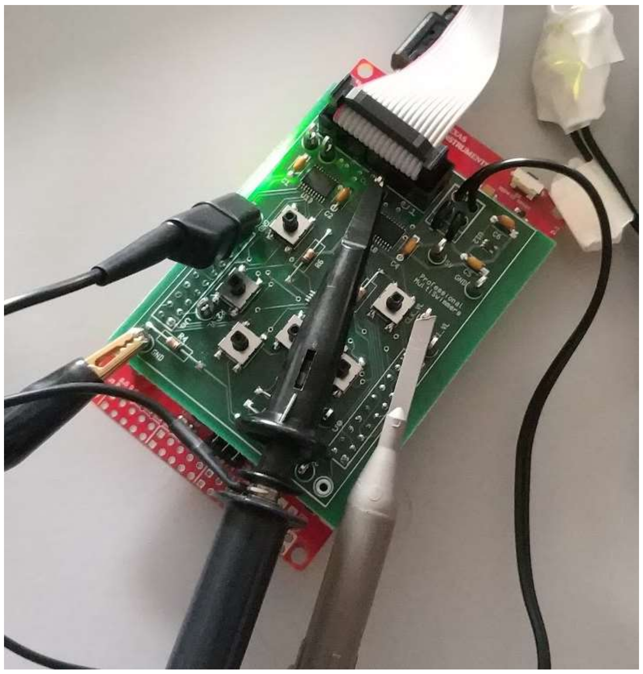

Figure 19. Oscilloscopes are placed on the test points R1_VL and R1 which correspond to the first red signal coming directly out of the MSP and that red signal after the level shifter. A change on R1_VL should show the same signal on R1 except that the logic high voltage is now 5V.

For the post-launchpad testing, we would run simple test programs (like toggling a single pin on the MSP) to verify that the input to the Level Translator would read 3.3V when toggled, and then the output would be 5V. We would then test each output of the MSP pins used in our configuration, to ensure that the MSP toggle could percolate through the level translator to the LED Matrix.

Final Results

As stated in the Project Proposal, the expectation for an A-range grade was as follows:

"A: The user should be able to play a complete game of tetris using the controller and the score-board with minimal to no bugs/defects in the game and the flow should be relatively, if not completely, smooth."

At the conclusion of the semester, as proven in a video demonstration of the Classic Tetris game being played, the full scope of A-range grade was met. The user of the game was able to play Classic Tetris on the LED dot matrix with no bugs in the game logic whatsoever. Likewise, the game had quite low latency and the game flow was very smooth, given the ability of the LED's.

The only small shortcoming of the project was the display of the score. Although we initially anticipated to represent the score as numbers on a second display, this turned out to be more difficult than we anticipated. Because we didn't use an Arduino etc, we didn't have access to any of the Adafruit libraries that one would normally have access to. These libraries are packed with tons of logic for boards just like our LED dot matrix display that allows the user to represent words, numbers, and graphics with ease. However, because we did not have access to any of this, we would have to code all the numbers, animations, and words pixel-by-pixel, just like the Tetris game was. Upon further discussion, the team decided that this would be too tedious and time-consuming to complete. Hence, although it is not specifically specified in the Proposal that the score representation had to be numeric and we compromised with binary, it would have been nice to have a more aesthetically pleasing user interface. In summary, we were successful in completing the entirety of the scope of our project, outlined in the proposal, that pertains to our A-range grade criteria.

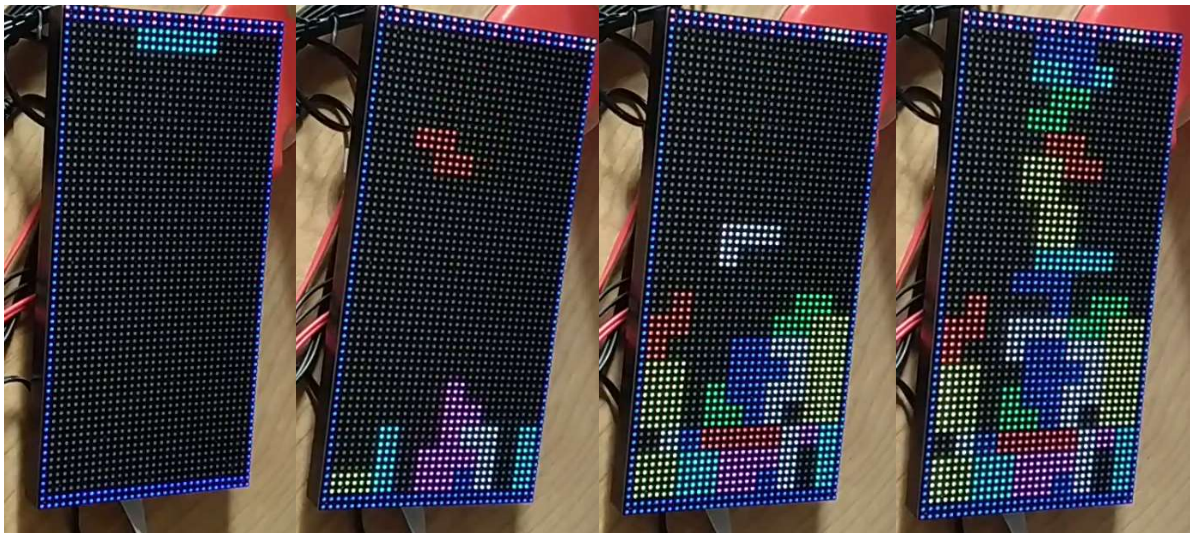

Figure 20. A game of Tetris being played, where the left shows the initial game state, and the right shows the game after having lost.

Costs

Costs are broken down into two broad categories: the first board sendout and the second board sendout. For brevity, individual component costs, save for the dot matrix display, power adapter, and microcontroller, are omitted from the outline and put under a generic category, "other components".

The difference in costs between the first and second board sendouts for the other components sections was that the second board sendout did not include lots of the cables that could be reused independent of the header board.

- *First board sendout*
    - MSP432P401R - $23.99
    - Board production costs
        - Manufacturing cost - $25
        - Shipping - $30
        - Stuffing at WWW - $20
    - 64x32 RGB LED matrix - $44.95
    - AC/DC 5V/10A adapter - $38.94
    - Other components (resistors, capacitor, cables, buttons, ICs) - $46.66
- *Second board sendout*
    - MSP432P401R - $23.99
    - Board production costs
        - Manufacturing cost - $25
        - Shipping - $30
        - Stuffing at WWW - $20
    - Other components (resistors, capacitors, buttons, ICs) - $38.06
- Total cost of prototyping and production - $346.59
    - Remaining budget - $153.41

Should we enter mass production, the total cost of the components for 10,000 units would be $1,176,474.20. Note that this is not accounting for the manufacturing and assembly costs of the header board itself and is still using the MSP432P401R evaluation board. This estimate also does not include the cost of the test points on the design since the product design would likely not need test points.

Several edits to the board would greatly reduce the cost of mass production. One is that we would likely integrate the microprocessor with the rest of the board that we designed, meaning that we save money on the MSP evaluation boards as the cost of increasing the complexity and the monetary cost of our design. The dot matrix and AC/DC wall adapter also should be sourced from manufacturers other than Adafruit as they do not offer discounts on higher quantities and cost \$449,500.00 and \$299,500.00 for 10,000 units respectively. The dot matrix display, AC/DC wall adapter, and MSP evaluation board compose roughly \$1,000,000 of the total cost. Beyond the larger subassemblies, we should use surface mounted alternatives for the resistors and capacitors and revise the copper bottom of the PCB to only cover relevant areas. Integrating the PCB that we designed with the display board itself would eliminate the need for a long ribbon cable from the board and display and allow for a much cheaper hand-held controller for the device, further reducing the cost.

Finally, a much more detailed and complete breakdown of the budget is attached in the appendix under the heading *Detailed Budget*.

Future Work

For further improvements to this project, our group proposes adding more complex game logic to the system. This would involve more logic for moving the pieces on the board. For example, one could implement new controls which allowed the game user to turn the pieces 180 degrees at a time. Another feature of a more ideal game logic would be functionality which increases the game speed (how fast a piece falls down) everytime the user completes a certain number of pieces. That is, the game would get progressively more difficult when the level increases. Doing so would create a more engaging experience for the user.

In terms of hardware, the buttons should be placed in more ergonomic positions. As it is currently, the buttons are spaced closely. These spacings make it a little difficult to play the game and also makes designing an aesthetically-pleasing and ergonomic enclosure for the project a difficult task. For future work, we would ideally make another PCB which would serve as the controller and would contain a more spaced-out arrangement of the buttons. To make the design more ergonomic, one should then also create a container inside which the controller and main PCB would reside. This enclosure would need a port for the controller cable, a display output cable, and a power input. This would make the whole hardware setup more aesthetically pleasing as it would more closely resemble a commercial gaming console.

When designing and implementing the aforementioned improvements to the game logic, one should consider timing. Our main difficulty in getting the game to work was speeding the game logic so the pixels could be written to the game board without causing the game to lag. Adding more complex game logic needs to be met with more software optimization in order to prevent lagging; for example, we could implement a more complex display driver that only writes to the display when the game status has changed.

References

[1] K. Liu, Y. Yang, and Y. Zhu, "Tetris game design based on the FPGA," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Yichang, China, Apr. 2012, pp. 2925–2928, doi: 10.1109/CECNet.2012.6201435.

[2] X. Chen and C. Lin, "Tetris Game System Design Based on AT89S52 Single Chip Microcomputer," in *2009 Third International Symposium on Intelligent Information Technology Application*, NanChang, China, 2009, pp. 256–259, doi: 10.1109/IITA.2009.239.

[3] Nintendo, *Tetris*. Nintendo, 1989.

[4] M. T. Elsir, P. Sebastian, and V. V. Yap, "A RTOS for educational purposes," in *2010 International Conference on Intelligent and Advanced Systems*, Kuala Lumpur, Malaysia, Jun. 2010, pp. 1–4, doi: 10.1109/ICIAS.2010.5716166.

[5] "IPC-2222 Sectional Design Standard for Rigid Organic Printed Boards," p. 5, 1998.

[6] National Instruments, "Multisim." https://www.ni.com/en-us/support/downloads/software-products/download.multisim.html#312060 (accessed Dec. 09, 2020).

[7] National Instruments, "Ultiboard." https://www.ni.com/en-us/shop/software/products/ultiboard.html (accessed Dec. 09, 2020).

[8] "Printed Circuit Board Design Check - FreeDFM.com | Advanced Circuits." https://www.4pcb.com/free-pcb-file-check/index.html (accessed Dec. 09, 2020).

[9] DigiKey, "DigiKey Electronics - Electronic Components Distributor." https://www.digikey.com/ (accessed Dec. 09, 2020).

[10] "Microsoft Excel Online, Spreadsheet Software, Free Trial." https://www.microsoft.com/en-us/microsoft-365/excel (accessed Dec. 09, 2020).

[11] "Calc | LibreOffice - Free Office Suite - Based on OpenOffice - Compatible with Microsoft." https://www.libreoffice.org/discover/calc/ (accessed Dec. 09, 2020).

[12] "GitHub," *GitHub*. https://github.com (accessed Dec. 09, 2020).

[13] Texas Instruments, "CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE)." https://www.ti.com/tool/CCSTUDIO (accessed Dec. 09, 2020).

[14] "Visual Studio Code - Code Editing. Redefined." https://code.visualstudio.com/ (accessed Dec. 09, 2020).

[15] "MSP432-SECURITY-AND-UPDATE-TOOL MSP432P4xx Security and Update Tool | TI.com." https://www.ti.com/tool/MSP432-SECURITY-AND-UPDATE-TOOL (accessed Dec. 10, 2020).

[16] "32x16 and 32x32 RGB LED Matrix," *Adafruit Learning System*. https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/connecting-with-jumper-wires (accessed Dec. 09, 2020).

[17] "Tetris Guideline - TetrisWiki." https://tetris.wiki/Tetris_Guideline (accessed Dec. 09, 2020).

[18] "2278 Adafruit Industries LLC | Optoelectronics | DigiKey." https://www.digikey.com/product-detail/en/adafruit-industries-llc/2278/1528-2505-ND/7035036 (accessed Sep. 13, 2020).

[19] DigiKey, "LDM-6432-P5-BLE4-1 Lumex Opto/Components Inc. | Optoelectronics | DigiKey." https://www.digikey.com/en/products/detail/lumex-opto-components-inc/LDM-6432-P5-BLE4-1/9607203 (accessed Dec. 09, 2020).

[20] "2279 Adafruit Industries LLC | Optoelectronics | DigiKey." https://www.digikey.com/en/products/detail/adafruit-industries-llc/2279/7034991 (accessed Dec. 09, 2020).

[21] DigiKey, "M3CCA-1620K 3M | Cable Assemblies | DigiKey." https://www.digikey.com/en/products/detail/3m/M3CCA-1620K/28196 (accessed Dec. 09,

2020).

[22] “658 Adafruit Industries LLC | Power Supplies - External/Internal (Off-Board) | DigiKey.” https://www.digikey.com/product-detail/en/adafruit-industries-llc/658/1528-1519-ND/5774322 (accessed Sep. 13, 2020).

[23] DigiKey, “MSP-EXP432P401R Texas Instruments | Development Boards, Kits, Programmers | DigiKey.” https://www.digikey.com/en/products/detail/texas-instruments/MSP-EXP432P401R/5170609 (accessed Dec. 09, 2020).

[24] DigiKey, “50-00543 Tensility International Corp | Connectors, Interconnects | DigiKey.” https://www.digikey.com/en/products/detail/tensility-international-corp/50-00543/7087238 (accessed Dec. 09, 2020).

[25] DigiKey, “CM1213-06MR ON Semiconductor | Circuit Protection | DigiKey.” https://www.digikey.com/en/products/detail/on-semiconductor/CM1213-06MR/5213215 (accessed Dec. 09, 2020).

[26] DigiKey, “B3FS-1000P Omron Electronics Inc-EMC Div | Switches | DigiKey.” https://www.digikey.com/en/products/detail/omron-electronics-inc-emc-div/B3FS-1000P/277814 (accessed Dec. 09, 2020).

[27] DigiKey, “5011 Keystone Electronics | Test and Measurement | DigiKey.” https://www.digikey.com/en/products/detail/keystone-electronics/5011/255333 (accessed Dec. 09, 2020).

[28] DigiKey, “302-S161 On Shore Technology Inc. | Connectors, Interconnects | DigiKey.” https://www.digikey.com/en/products/detail/on-shore-technology-inc/302-S161/2794234 (accessed Dec. 09, 2020).

[29] DigiKey, “CFR16J10K TE Connectivity Passive Product | Resistors | DigiKey.” https://www.digikey.com/en/products/detail/te-connectivity-passive-product/CFR16J10K/3313505 (accessed Dec. 09, 2020).

[30] DigiKey, “C315C104M5U5TA7303 KEMET | Capacitors | DigiKey.” https://www.digikey.com/en/products/detail/kemet/C315C104M5U5TA7303/3726100

(accessed Dec. 09, 2020).

[31] DigiKey, "MAX3008EUP+ Maxim Integrated | Integrated Circuits (ICs) | DigiKey." https://www.digikey.com/en/products/detail/maxim-integrated/MAX3008EUP/5086189 (accessed Dec. 09, 2020).

[32] DigiKey, "MIC5365-3.3YC5-TR Microchip Technology | Integrated Circuits (ICs) | DigiKey." https://www.digikey.com/en/products/detail/microchip-technology/MIC5365-3.3YC5-TR/1868230 (accessed Dec. 09, 2020).

[33] DigiKey, "M20-7831046 Harwin Inc. | Connectors, Interconnects | DigiKey." https://www.digikey.com/en/products/detail/harwin-inc/M20-7831046/3727804 (accessed Dec. 09, 2020).

[34] "10-01097 Tensility International Corp | Cable Assemblies | DigiKey." https://www.digikey.com/en/products/detail/tensility-international-corp/10-01097/3507722 (accessed Dec. 09, 2020).

[35] DigiKey, "CA-2185 Tensility International Corp | Cable Assemblies | DigiKey." https://www.digikey.com/en/products/detail/tensility-international-corp/CA-2185/568576 (accessed Dec. 09, 2020).

[36] DigiKey, "B3FS-1050P Omron Electronics Inc-EMC Div | Switches | DigiKey." https://www.digikey.com/en/products/detail/omron-electronics-inc-emc-div/B3FS-1050P/5723299 (accessed Dec. 09, 2020).

Appendix

In this section you should include helpful information that does not fit into the above categories but will be helpful in understanding and assessing your work. Complete code listings should be in this section, and detailed cad drawings.

Delivered in a Word .docx format with track changes turned on

Detailed Budget

Date	Item	Quantity	Price per unit	Remaining budget

				$500.00
09/18/20	MSP-EXP432P401R	1	$23.99	$476.01
10/19/20	Board sendout 1	1	$55.00	$421.01
10/21/20	CM1213-06MR	2	$0.98	$419.05
10/21/20	B3FS-1000P	7	$0.59	$414.92
10/21/20	302-S161	1	$0.40	$414.52
10/21/20	M20-7831046	1	$1.81	$412.71
10/21/20	CFR16J10K	8	$0.10	$411.91
10/21/20	5011	12	$0.40	$407.11
10/21/20	C315C104M5U5TA7303	8	$0.21	$405.43
10/21/20	MAX3008EUP+	3	$5.21	$389.80
10/21/20	MIC5365-3.3YC5-TR	3	$0.12	$389.44
10/21/20	M20-7831046	1	$1.81	$387.63
10/21/20	10-01097	1	$3.30	$384.33
10/21/20	658	1	$38.94	$345.39
10/21/20	50-00543	1	$2.11	$343.28
10/21/20	2279	1	$44.95	$298.33
10/21/20	CA-2185	1	$2.42	$295.91
10/21/20	M3CCA-1620K	1	$5.45	$290.46
11/9/2020	Board stuffing at 3W	1	$20.00	$270.46
11/6/2020	Board sendout 2	1	$55.00	$235.46
11/9/2020	B3FS-1050P	6	$2.02	$223.34
11/9/2020	CM1213-06MR	1	$0.99	$222.35
11/9/2020	302-S161	1	$0.40	$221.95
11/9/2020	M20-7831046	1	$1.90	$220.05
11/9/2020	CFR16J10K	6	$0.10	$219.45
11/9/2020	5011	12	$0.40	$214.65
11/9/2020	C315C104M5U5TA7303	6	$0.21	$213.39
11/9/2020	MAX3008EUP+	2	$5.21	$202.97
11/9/2020	MIC5365-3.3YC5-TR	1	$0.12	$202.85
11/9/2020	M20-7831046	1	$1.90	$200.95
11/9/2020	10-01093	1	$3.55	$197.40
11/9/2020	MSP-EXP432P401R	1	$23.99	$173.41
11/9/2020	Board stuffing at 3W	1	$20.00	$153.41

Complete Multisim Schematic

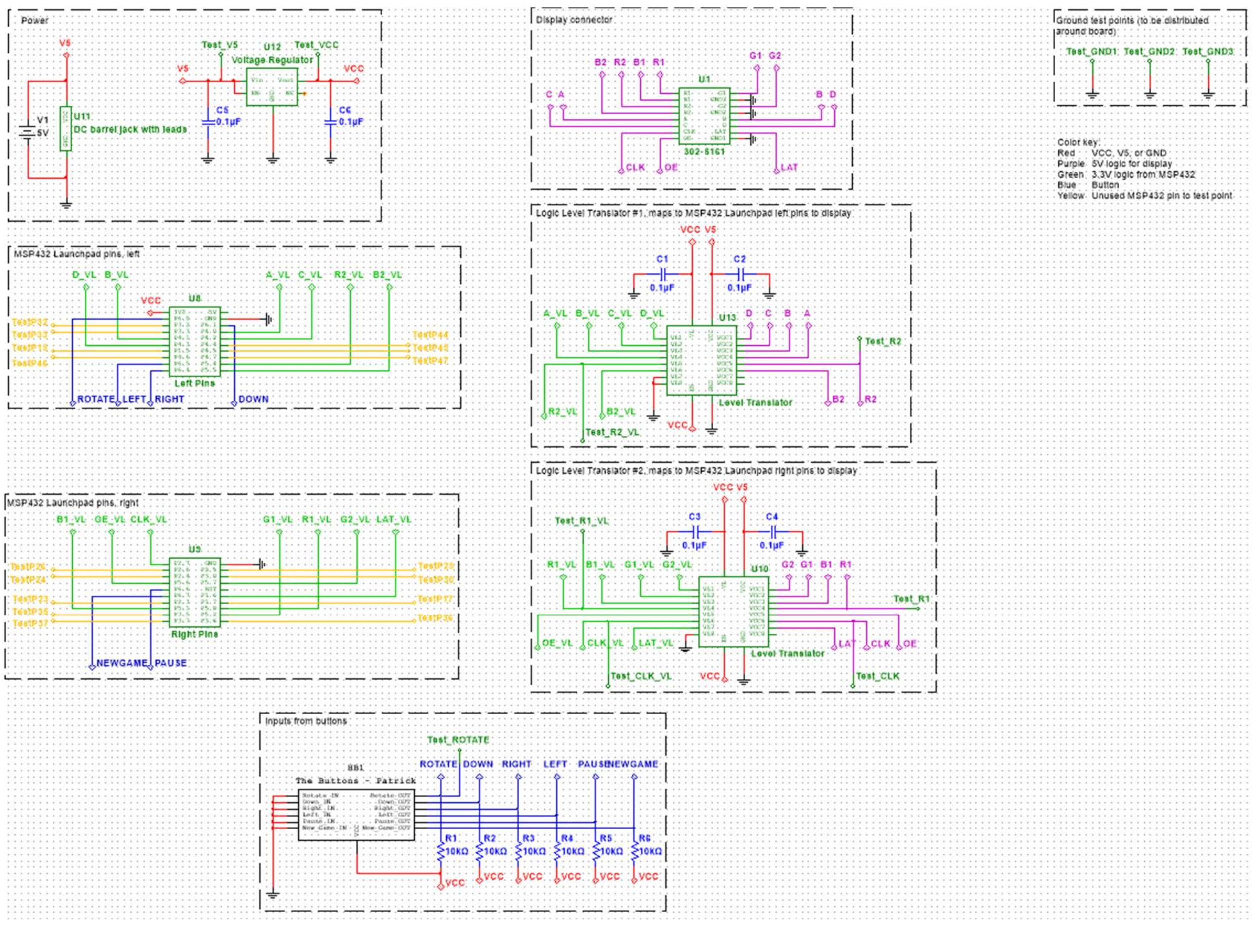

Figure 21. Complete Multisim Schematic

Complete Code Listing

All code is available on GitHub and available to browse online as well as attached below. The online version of the code is highly recommended for viewing over the archived code below.

- Final Version - https://github.com/arjundeopujari/Tetris
- Initial Terminal Version - https://github.com/daniel-mizrahi/ECE-Capstone-Tetris