CAOS Project Report: Match4 Game with Arduino

Group Members: Tim Bachmann, Daniel Weissen
27.01.2019

Introduction

The goal of our project was to create a match4 game which runs on an Arduino Uno, is played with buttons and is displayed on an LED Matrix. Initially we planned to have 7 buttons, each corresponding to a column in the match4 game, where the press of a button would result in a pin in the game dropping in said column. The rules and design of the game perfectly match the real-world counterpart. The game was to be displayed on an LED Matrix, we wanted the game to have a “retro” feeling, perhaps like something that you would find at an arcade. We encountered many difficulties during the development process and some plans had to be changed, but we are pleased with the finished product.

Background

We decided to use the Arduino Uno for our project, mainly due to the fact that the Arduino was heavily recommended in the lecture and previous experience with the device. The LED Matrix had to have a bigger width then height in order to mimic the real match4 game, we decided on a 32x16 LED Matrix. For the buttons we wanted something that was appropriately big and satisfying to press, we decided on red buttons with a tactile clicking mechanism.

The code for the game was relatively simple, the only special library we needed to use was the library to communicate with the LED Matrix. No other special code resources/libraries were used for the project.

Implementation, Setup, Development

A quick start

After the group decided on the necessary products they needed for the project, an order was placed with an estimated delivery time of about 2 weeks. In those first two weeks, Daniel decided to write the code for the match4 game. The plan was to write the game in C, and have it be playable through the console. The game was quickly written, the main hurdle was the, to Daniel, unfamiliar C Language.

One thing that is certainly worth noting about the initial code for the game was a lack of foresight by Daniel. Perhaps somewhat naively, Daniel assumed that, if the code runs on the PC, then surely it would run just fine on the Arduino as well. Needless to say that was not the case, it is something we will come back to later.

The assembly

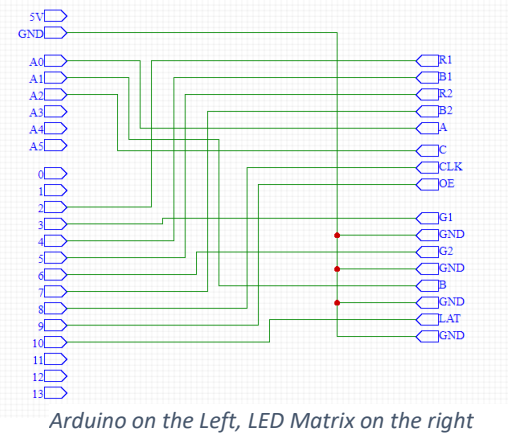
After all the parts arrived Tim could start the assembly process. This however proved to be the most difficult and cumbersome part of the project. The first issue came up when Tim figured out, that the LED Matrix needed an additional power supply. Initially we thought the LED Matrix would ship with a power supply included, but it didn't, so we ordered a suitable power supply but shipping took unexpectedly long. The power supply didn't ship until well into the holiday season. Originally, we planned to have the project finished before the end of the year so we didn't have our attention split between exams and the project, now we barely started!

Eventually the power supply arrived, and Tim could start the assembly process.

I theory the assembly was not overly complex. The LED Matrix had good documentation and schematics online. But it became clear quite soon that because the Arduino and the LED Matrix had totally different layout of pins, it would be more work than expected to connect the two devices. This might have been easier with a 5$ circuit board but to ship that from the USA would have potentially cost us another two weeks.

So, Tim went for the quicker although more labor-intensive way of soldering the two connections manually with cables together.

We then realized that because the Matrix uses a big portion of the Arduinos pins we wouldn't be able to fit all 7 buttons on our circuit. The solution to this was to install just 3 buttons. Two buttons that control the position of a cursor and one button to "drop" a pin on the position of the cursor.



Arduino on the Left, LED Matrix on the right

Getting the Match4 code on the Arduino

After everything was finally connected and working properly, it was time to get Daniel's code running on the Arduino. The first obvious problem with Daniel's code however, was that it assumed, that the game would be controlled with 7 buttons and not 3. Each of the 7 buttons would correspond to a column in the game, now, however, there are 3 Buttons, one to lower the chips and 2 to choose in which column to place the chips. The code had to be altered slightly to account for this change. Additional code also had to be written to read and process the button inputs properly and to display the game on the LED Matrix.

After all the code was written and tested in isolation, it was time to put everything together and hopefully have a working match4 game, played with 3 buttons and displayed on the LED Matrix. Unfortunately, that first attempt at testing all the code working together was far from successful, since something quite weird happened. When the code tried to load onto the Arduino, the LED Matrix began to flicker widely, as if the device was broken. This could not be the case though, since the matrix worked perfectly fine with other code. After some experimentation Tim discovered that the Arduino was out of memory and could not handle Daniel's match4 code.

Daniel wasn't even thinking of the possibility that they could run out of memory on the Arduino, his code wasn't well adjusted to the concept of limited memory consumption. 3 examples how Daniel's code "wasted" memory follow:

1. Using expensive data types: Daniel used a lot of Strings to represent data in the game which could have just as well been represented using much cheaper data types.
2. Unnecessary code: In order to check if someone won the game, Daniel's code would go over the entire gameboard and check from every single chip, in every single direction, if a match4 occurred. It was perfectly sufficient however to only check from the last chip inserted.
3. Wrong priorities: Daniel took the lecture "Software Engineering" while writing the match4 code and tried to write his code in a fashion representative of the principles taught in that lecture. One of those principles says that modules should be as non specific and reusable as possible, another one states that modules ought to only fulfill one specific purpose and a third one states to always take possible future changes on the code into account. When writing code for something that has very limited memory however, it appears that writing your code in that fashion simply uses too much memory to be feasible.

As a means to cut down on memory consumption Tim reworked a lot of Daniel's code. Some of the data types were changed to "unsigned char" and a lot of the unnecessary code was cut out. After the necessary changes were made, Tim finally got the game working.

The final steps

The final things the team wanted to do, was to improve the UI of the game and make everything pretty and cool to look at. However, even after significantly cutting down on the code, the Arduino was still pretty close to having no free memory left, which meant that the team was limited when writing further improvements. The team also decided against buying additional features such as a little sound box to play sound effects through, since due to unreliable shipping times it would be uncertain whether those features could be properly implemented in time.

Some of the last features that Daniel implemented were an animation for the chips falling down into their correct position when the button is pressed. Some minor improvements to the readability of the game were made, for example, the match4, once it occurred, changes color so the players can see where the match4 occurred. There was also some time spent testing the game and checking if all the different

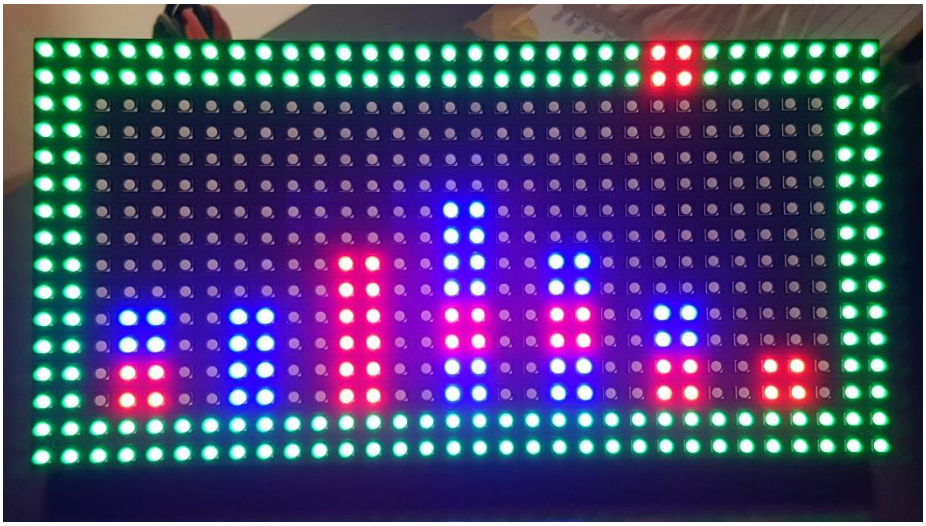
possibilities a match4 could happen, would be correctly registered by the game, and in fact one such bug was found that had to be fixed.

Daniel wanted to implement additional functionality to the game, but the memory was so close to being full, that he decided that there were no further improvements that could be made.

After some final testing and improving the readability of the code, the team decided that the technical part of the project was now finished. One final step was to build a Arcade like enclosure for the Project. A black was used, holes were cut out for the buttons and the LED matrix is connected to the box using strings.

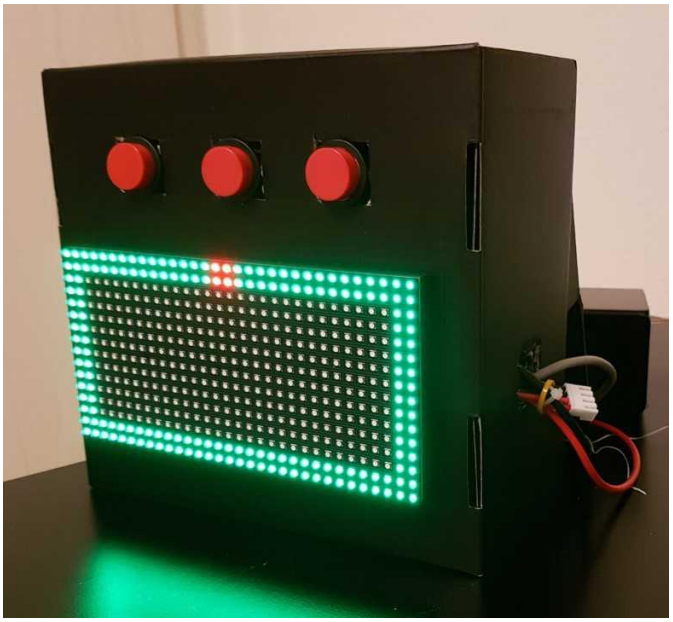
Results

This is the game after a couple turns have been played.



These are the Buttons, the Arduino and all the connections.

This is the final Project in its arcade like enclosure.



Conclusion / Lessons learned

While we are happy with how the project turned out, there certainly are some things that we would do differently in the future for project such as this. The most obvious thing we could improve on is a more precise analysis of all the things we need for a project. We lost months because we didn't have the necessary equipment we needed to continue. The fact that shipping times in combination with a set deadline can be a huge cause of many problems is something we are now very aware of.

A more complete understanding of the Arduino itself would have helped us to predict certain problems we faced, such as the lack of ports for 7 buttons or the sparse memory on the device. Especially the memory issues were a really cool learning experience. It made us more aware of the way a computer stores memory and from now on, we know how to cut down on memory quite effectively if the problem would arise in the future.

Although not perfect, we consider the project a success, since it delivers on all of the functionality we wanted it to have. The many issues we ran into were valuable learning experiences and we now feel much more capable to deal with a similar project in the future.