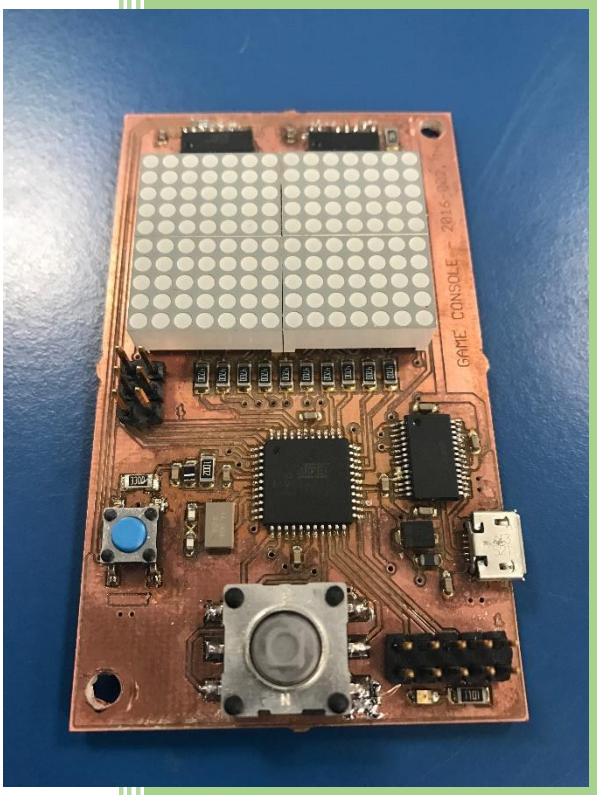
2016

Process Report For Game-Console



Jimmi Andersen, Marek Mikitovic and David Papp

Semester Project fourth Semester

16-12-2016

Table of Content.

Beginning of the project

Our project supervisor was Ib Havn, with whom we spent few introduction hours at the beginning of our project, so we could revise our knowledge from RIE (Realtime-Programming Interfacing & Electronics) class. He advised us to focus on recapitulating knowledge we know about data layer and mainly about the protocols. What kind of design issues we can face or how to stuff bytes in to the frames (framing methods). What error control and flow control are. Exercises we have done were supposed to give us better understanding how before mentioned methods could be implemented in real life, thus it helps us creating our own protocol based on serial port communication.

Workflow

We started as two students group, while we were developing our own solutions for given exercises such as different calculations for error corrections. There were some functional requirements that have to be met. Small brainstorm session revealed that the good direction would be to design ping pong based game on the display of this size and we started adding other non-functional requirements to the project.

The plan was simple divide project into smaller parts make them work and put it together. Then move one to the another part. Our approach was simultaneously working on diagrams and code implementation of it, which seemed like not one of the best. Next time we would like to change that, by creating more detailed diagrams of different tasks and system, would help us more during the implementation. Later by irrefutable occasions was our group extended by one student. Which doesn't change much but we had one extra worker to keep better focus on the project. We encountered a lack of theory in some fields as creating task diagrams and unit testing.

This has been solved by discussing it with our supervisor and then we were pointed to a book where we found some inspiration and clue how could such a diagram look like. In the final implementation of the game, we found out that there were some bugs. Those were solved by testing and reviewing the implementation.

Brief timeline

- Implementing first error checking codes
- Design and implementation of byte stuffing and framing
- Design of the first version protocol
- Design of game logic and diagrams
- Implementation of display function
- Implementation of player 1 movement
- Implementation of ball task, bouncing
- Design of early version of state machine and system diagram
- Testing single player stage of game
- Implementation and testing of protocol and queue for received data
- First version of Java PC protocol communication
- Change of the game logic mapping of displaying position of players
- Added implementation of player 2 movement
- Testing the end of the round logic ball can bounce off the playground
- Update on state machine diagram
- Reimplementation to C# PC protocol communication
- Idle and game state working properly
- Implementation of displaying score state
- Further testing and debugging

Improvements

Next time we could spend more time on proper diagrams especially state machine from the beginning of the project. It's more efficient to start "coding" design which is well described than, running into the problems in production.

Our protocol is very simple since we are sending only one byte of payload and as a response ACK or NACK. We could implement CRC or other error checking methods, but we got only two types of messages for moving up or moving down even if the message would be corrupted we still got enough time to resend another one. So the gameplay still wouldn't be affected.

Learning experience

During this project, we gained knowledge about protocols and different approaches to handling the data frames over the physical layer. We were introduced to different techniques of error checking codes as parity bits, checksums and CRC (cyclic redundancy codes) and even error repairing, Hamming codes. We improved our skills in using Free RTOS framework. How to use semaphores/mutexes or queues. Least but not last we improved our software developing skills in embedded field since this was the larger and more complex project than previous projects.

Self-reflection Jimmi

This project has been really excited to work on especially because I like to work with programming this kind of hardware, however, I must admit I am not the best programmer but have learned a lot working on this project but could still use more experience on the codding part. Doing this project I have handled most of the project documentation and this is something I have always handed to other group members, therefore, I am satisfied that I got to do this part to get better at documentations.

Self-reflection Marek

This project was one of the most interesting projects I have been introduced to. We combined our knowledge and skills to create PCB board, which we developed and designed with help of our teacher. Further, we were able to program microcontroller on that board. It was actually the best thing to create something completely from scratch. During this project, I achieved a lot of knowledge in my specialization. I think haven't got the opportunity to experience hardware programming in such amount as now. Since I joined this group in the middle of the project some of the work has been done. Mainly I have done the process report, but I also was kind of group supervisor on the implementation of the code, and I found out that having an another pair of eyes on obvious things is really handy, and would save a lot of time debugging.

Self-reflection David

The project offered many challenges and way to gain knowledge and experience. This was the first that I had to work with a real time operating system outside the class exercises and at first this provide difficult and challenging. We need to figure how to use the theoretical knowledge gain through the RIE in a complex system. We were fortunate as the communication between us, the team members, was really good and we could share knowledge easily. I liked to work with this team and this way and in the future I would change just a few things, like stricter time table. In my opinion the best thing was that we did most of the coding in paired programming manner, thus saving us valuable time debugging error coming from typos and missing pointers.