MZ_APO Pong
Technical documentation

Adam Loucký, Vít Knobloch

May 2021

1 Introduction

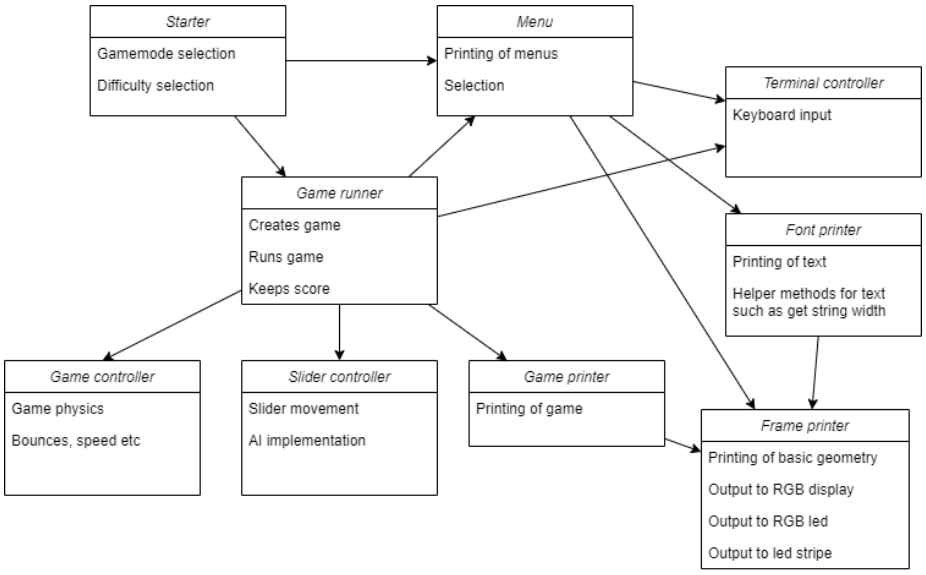
The program is an implementation of PONG game for MZ_APO kit, it directly accesses the kit's embedded and connected peripherals for output and ssh terminal connection for input. It was developed using remote lab access.

2 Used technologies

- MZ_APO kit and provided source code for low level peripheral access and memory mapping
- C programming language and GNU Make
- Big Blu Button, SSH and WSL2 for lab access
- Git and VS Code for source control
- Discord for team communication
- Overleaf online LaTeX editor for documentation and user manual

3 Building blocks

The application consists of several building blocks that are interconnected and call each other for their subroutines. The main building blocks and their dependencies are described in the diagram below.



Only the functions which are used by other blocks are defined in the header files. We tried to keep the building blocks as separated as possible.

4 Building and running

To run the application, you have to compile it first. You can use the included Makefile to do so, you will need to have *arm-linux-gnueabihf-gcc* cross compiler installed on your system. The target executable file's name is *pong*. Inside the Makefile change the *TARGET_IP* field to IP address of an unused MZ_APO microcomputer in the remotely accessable lab. Then use *make run* command to start the app via SSH. We recommend setting the *keyboard repeat rate* high and *keyboard delay* to the lowest possible setting on your system for better playability before launching the program (unfortunately, the application cannot change these settings via SSH, nor can it read the scancodes directly), you can use the following commands to do so:

```
# set key repeat rate. delay 50 milisecond, 40 per sec
xset r rate 50 40
# restore key repeat rate to default. 25 per sec, 660 milisecond delay
xset r rate
```