UNIVERSITÄT BERN

Bern University of Applied Sciences
Engineering and Information Technology

SuPong

Project documentation

by *The F-Team*
Fabian Page
Fabio Bernasconi

last Update June 30, 2008

Contents

1 Introduction

This document describes the implementation of the SuPong game for the Keil ARM STR9x board. SuPong is a project written for the Microcontroller Programming course that is part of the Biomedical Engineering Master at the University of Bern.

2 SuPong - the game

SuPong is an implementation of the well known video game pong [1]. The implementation of the game itself was straight forward and thus will not be considered any further in this document although some parts of the logic are incomplete.

3 STR9x Implementation

Prior to implement the game for the STR9x, the project team decided to write an x86 version of it (without input handling though). This decision was taken because testing of the game implementation itself was a lot easier on a x86 platform than it is on the the Keil STR9x kit.

The display of the Keil STR9x board is only 2x16 characters large (small). To make the game more fun to play the project team decided to use a second board. Using two boards the screen dimensions doubles, that is it grows to 4x16 characters.

The boards communicate over a network. The project team used the TCP/IP stack implementation of the easyweb demo pack. Using the Keil RTX would have been easier since the easyweb TCP/IP implementation is kind of a hack. This resulted in a long and painful development time for the network realted stuff. Once the boards are running the slave tries to connect to the server. Thus it is important that the master is running before the slave does. After the initialization phase the master and slave alternatively send and receive a “state token”. The state token contains the complete game state.

[1] see for a description of the game http://en.wikipedia.org/wiki/Pong

4 Developer Documentation

This section gives a short introduction on how to setup the development environment that was used to develop the SuPong, how to flash the board or how to compile a x86 version of the game.

4.1 Development Environment

Eclipse CDT[2] and μVision IDE[3]. See 4.2 and 4.3 for platform setup.

4.1.1 Directory structure

- SuPong
  - src ; all the source files except the keil libs
    - core
      - engine.h ; main include file, game logic
      - engine.c
      - supong.h ; game structs and the like
      - supong.c
      - types.h ; some type definitions
    - x86
      - renderer/consoleRenderer.h ; provides functions for game rendering
      - renderer/consoleRenderer.c
      - main.c ; main entry point
      - README ; some notes for the x86 development platform
    - str9
      - str9Renderer.h
      - str9Renderer.c
      - main.c ; main entry point
      - ...
  - bin ; temporary output directory
  - doc ; documentation and other text
  - keil ; keil related stuff, some source files
  - .git ; git stuff
  - .settings ; eclipse project settings
  - README

4.1.2 Versioning System

Git was used as versioning system. The public project repository is located at `http://github.com/fbern/supong/tree/master`. For instructions on how to checkout the project check the link above or google after git tutorial.

[2] `http://www.eclipse.org/cdt/`
[3] `http://www.keil.com`

4.2 Compiling on a x86 platform

Using Eclipse you can import the project file. The project definition file is located in the SuPong root directory. You may need to change the compiler/linker configuration to sweet your needs first (depending on if you work with Linux, Mac OS X or Windows). You should now be able to successfully compile the project. Remember that the x86 version depends on the ncurses library. Run the program from the command line.

4.3 STR9x version

A project file exists for the Keil μVision IDE. It is located in the keil directory of the project. This directory contains also contains other source files which are needed to successfully compile the game for this platform. If you compile and load the program to the board's flash make sure that you compile one version with project target 'Slave' another with 'Master' as target. This is necessary because the targets defines two different symbols, MASTER or SLAVE namely.

5 User Documentation

This section contains the user documentation. It describes what you need, how to setup the 2 str9x boards and how to play the game.

5.1 Prerequisits

You need to successfully play the game

- 2 STR9x board kits
- 2 network cable
- 1 hub/switch
- 2 players

5.2 Hardware Setup

First setup the boards as shown in figure 1. Plugin the network cables and connect them together through a hub or a switch. You should now be ready to run the game.

A video of the game can be found in the doc folder.

5.3 Booting

After flashing the rom you should first start the server board. Then start the client. As soon as booth boards are up and running the game starts. So be ready for the challenge.

5.4 Howto Play

A player can interact (go left/go right with pong) with the potentiometer by turning the wheel to the left or the right.

5.5 Game Rulez

The game rulez of the pong video game are simple. The first player who let the ball hit the wall behind the pong looses the game.

5.6 Status LEDs

Keil STR9x board only. The GPIO7 leds are used to give feedback of the program state to the user.

- GPIO0: On if Master
- GPIO1: On if Slave
- GPIO2:
- GPIO3: shows network action

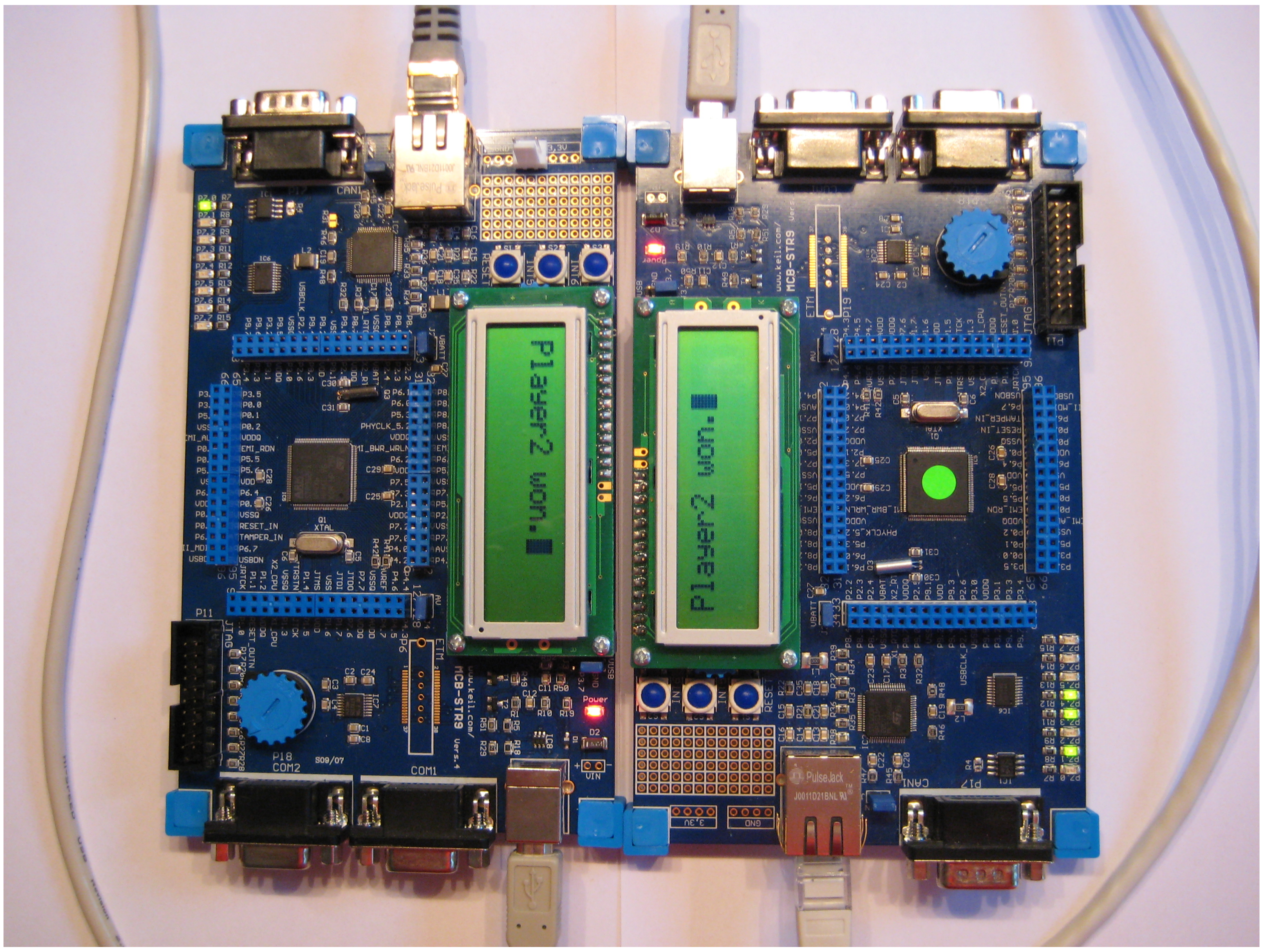

Figure 1: Hardware Setup

- GPIO4: shows network action
- GPIO5:
- GPIO6:
- GPIO7: blinks if program is running

5.7 Screenshots and Videos

A video of the game can be found in the doc folder.

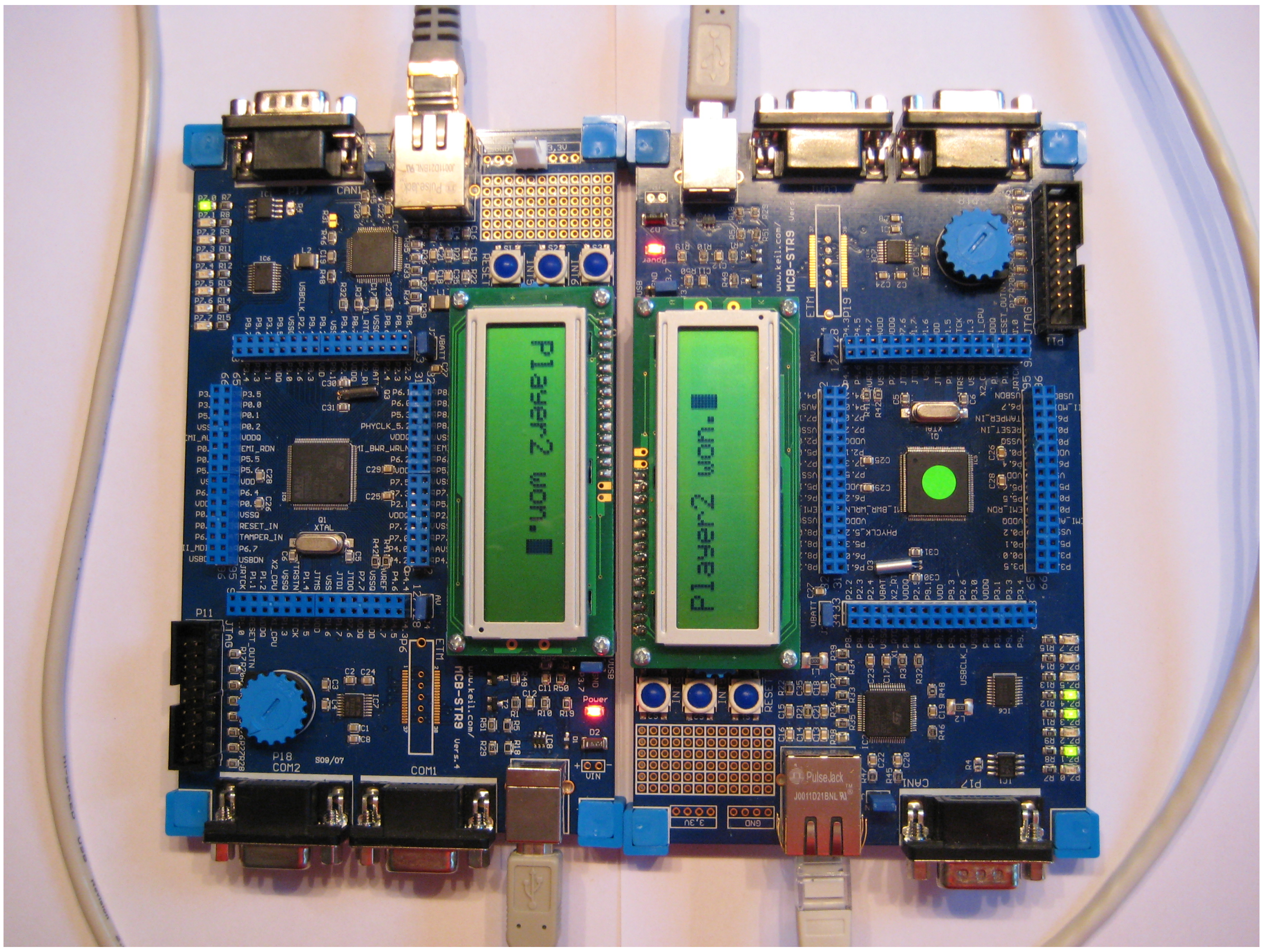

Figure 2: Demo rendering

6 Improvements

The current implementation is very basic. Further simple improvements like

- Alternating ball speed
- Improved pong steering
- ...

would make the game much more fun to play. As always in software projects the code documentation is not perfect.

7 Conclusion

Hardware close programming is very time consuming. It took the project team a long time to figure out how to build the network related stuff for the game. This was a really pain in the ass. Especially because the team used a network stack implementation for the STR9 board that is more or less “a hack” and thus is very sensible to ie. order of header file include, operation count between network related calls and the like.