2016

Jimmi Andersen, Marek Mikitovic and David Papp

Semester Project fourth Semester

16-12-2016

Project Report For Game-Console

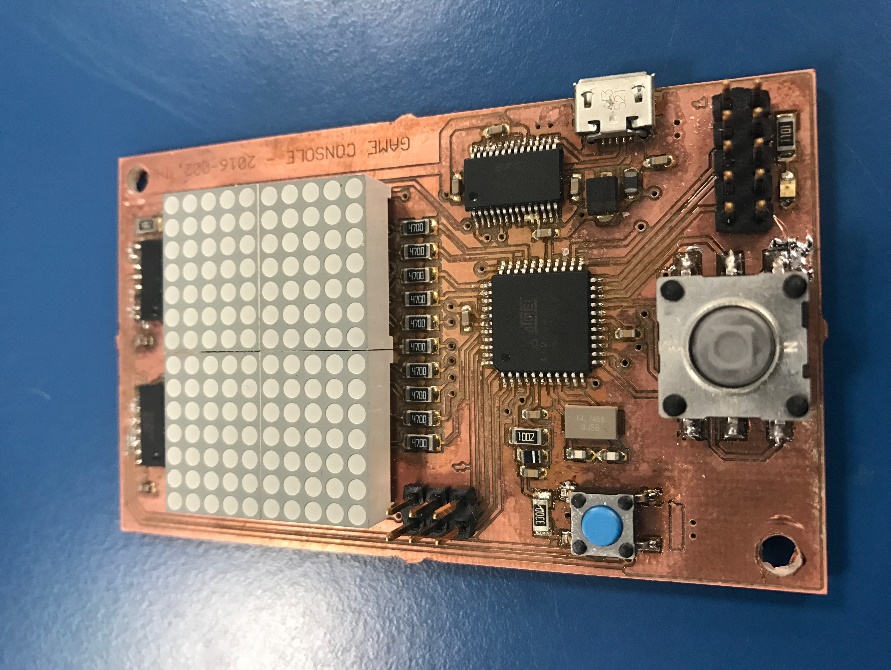


Table of Content.

[Abstract. 2](#_Toc469481313)

[Introduction. 2](#_Toc469481314)

[Analysis. 3](#_Toc469481315)

[Requirements. 3](#_Toc469481316)

[Functional: 3](#_Toc469481317)

[Non-functional: 3](#_Toc469481318)

[System Diagram. 3](#_Toc469481319)

[System timeline. 3](#_Toc469481320)

[Design. 4](#_Toc469481321)

[Game Protocol 4](#_Toc469481322)

[Game Protocol Diagram 4](#_Toc469481323)

[Task 1 Serial Connection 5](#_Toc469481324)

[Task 1 Serial Connection Diagram 5](#_Toc469481325)

[Task 2 Ball 6](#_Toc469481326)

[Task 2 Ball Diagram 6](#_Toc469481327)

[Task 3 Player on board 7](#_Toc469481328)

[Task 3 Player on board Diagram 7](#_Toc469481329)

[Task 4 Player Pc 8](#_Toc469481330)

[Task 4 Player Pc Diagram 8](#_Toc469481331)

[Implementation. 9](#_Toc469481332)

[Test. 9](#_Toc469481333)

[Result. 9](#_Toc469481334)

[Discussion. 9](#_Toc469481335)

[Conclusion. 9](#_Toc469481336)

[References. 9](#_Toc469481337)

# Abstract.

*This project addresses the learning experience of creating a two-player cross platform game. This report covers all the steps taken from the introduction of the system until the results and conclusion. In the first section of this report, the introduction provides a thorough description of the case, the purpose the system serves and how it will be relevant. Documentation of the analysis process follows, including the requirements made from the project description, the use cases created based on those requirements, the diagrams and descriptions regarding the systems tasks is found in the “Design” section. The “Implementation” chapter gives an overview of how a few Tasks work. Our collective opinion on the outcome of our Game system and the project is described in the “Discussion & Conclusion” section.*

# Introduction.

Ping Pong Two Player Game has been created by a small group of students at VIA University College. The Reason for this game is to help learn how to make small embedded systems and programming over different platforms using serial connections using a USB port. The task is to develop and program a small micro control board and use this for making a small game of Ping pong the players will be implemented as player one and is controlled by the joystick on the micro control unit(MCU) and player two is controlled by the keyboard on a computer using the USB connection. The game will be displayed on the small Dot-matrix display. There will not be displayed any Graphical user interface on the computer part because in the future we would like to use two MCU for playing the Ping Pong Game using only the computer as a relay station for transmissions between the MCU.

# Analysis.

The first step of developing our system is the analysis. All the stated requirements are read, analyzed, and discussed. The requirements are used to make Tasks and then Task diagrams to provide a decent overview of the actions the system should do when completed.

## Requirements.

Mareks part

Having a clear list of requirements is here to provide the good overview on how to start creating the system and how to know when it is going to be finished.

### Functional:

* System must use 3 tasks.
* System must have 2 real time tasks.
* System must have 2 task sharing resources.
* The system must use semaphores or mutexes.
* The pc must be able to control player two on game board.
* The system must give real time guarantee.

### Non-functional:

* Game must display the game on computer screen.
* Game gives score when ball hits paddle
* Game resets when ball is out of bounds.

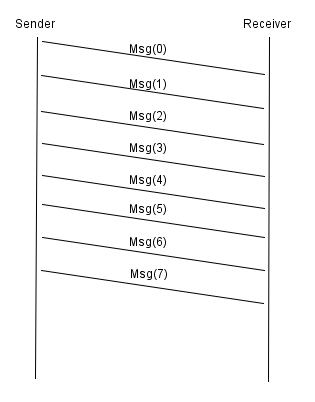
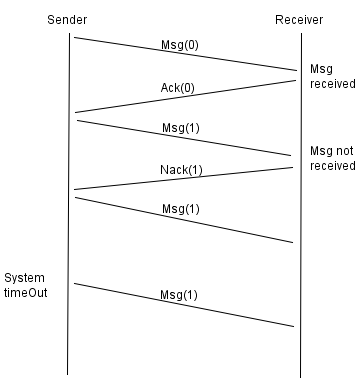
## Task Diagram.

# Design.

In the design part describes in detail what our different tasks in the game system is doing and why we chose to use this design over another. For better understanding the task diagrams are shown under the description.

## Game Protocol

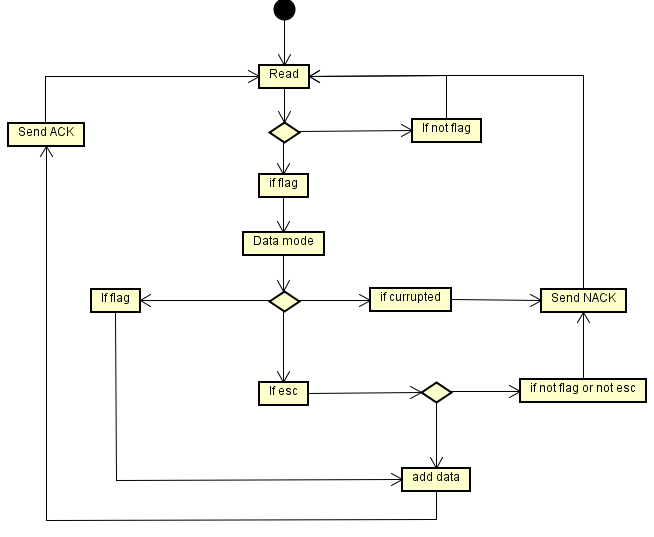
In this project, we have used a protocol design that ensures the security of the system by sending acknowledgments to the sender for every message received. The system also uses timers to ensure the messages is received synchronously. To explain further what the system does is when sender is sending the message it uses a timer for handling the fact that the system does not send or the system has some interrupted communication of the ANC (message acknowledgment) if the sender manages to throw a timeout exception we then resend the message to the receiver. The sender listens after the ANC or the NANC (message not Acknowledged) if the sender receives the ANC it will send the next message to the receiver on the other hand if the sender receives the NANC it will send the current message again until ANC is received this gives you the security of knowing that the system will receive all messages synchronously. This kind of protocol design will take more computation time than the simple protocol design where you just send messages as fast as possible for instance if you don’t care if the message is correct on the receiver side you just care that the system receives the messages. Game Protocol Diagram

 Diagram 1 Secure system protocol Diagram 2 Simple Protocol 1

## Task 1 Serial Connection

What the serial connection does is that it reads the queue of received messages and checks if the start of the message is a flag if it is a flag it will go to the data mode if the start of the message is not a flag then it will start itself again and continue to read until flag is received. When it is in data mode it will check for tree things it will check if it receives another flag then it will store the data if it reads an ESC it will check if the ESC were supposed to be there and then store the data otherwise it will send back to the sender a NACK for not received. If it in data mode does not receive neither a flag or an ESC, then it will send the NACK back to sender. When it stores the data for usage to update the game it will send back a ACK to sender for Acknowledgement. If the Serial Times out it will send back a NACK to sender to tell that the message were not received.

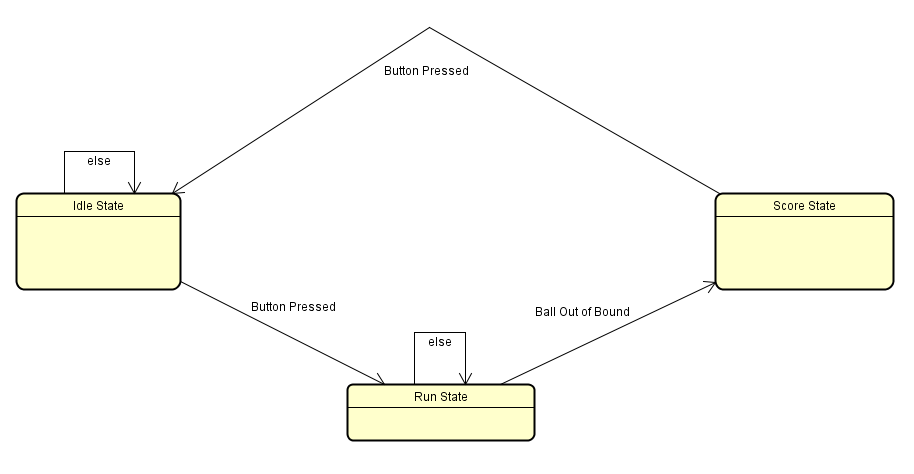
### Task 1 Serial Connection Diagram



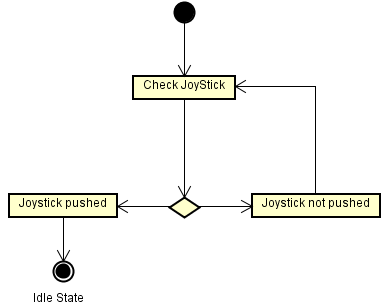
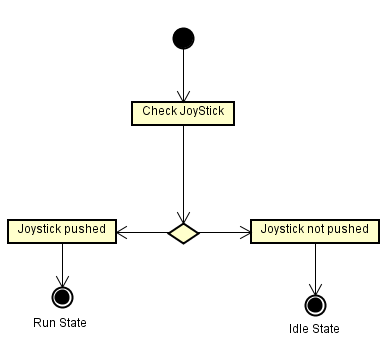
## Task 2 Game

### Task 2 State Machine

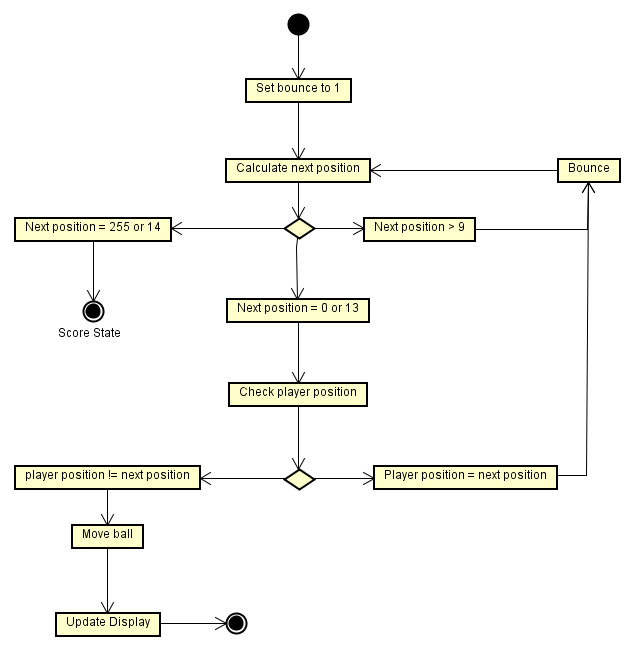
Game State Machine Diagram



Idle State Diagram. Score State Diagram



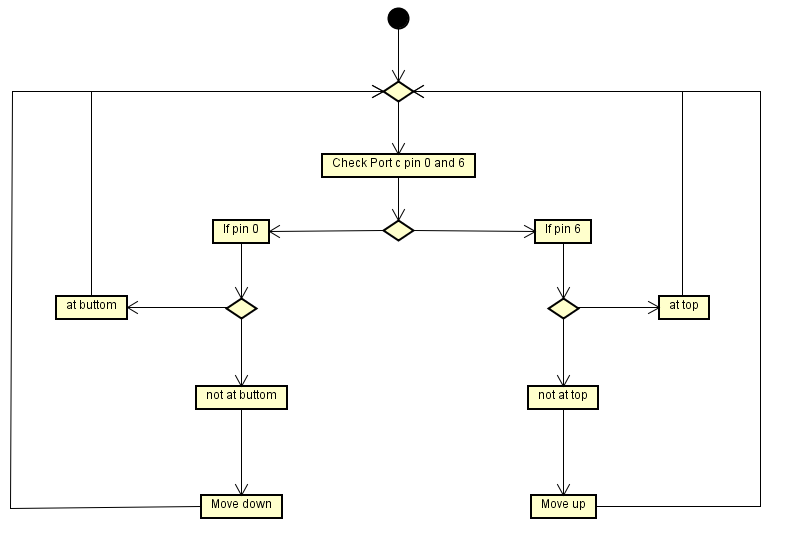
Run State Diagram.



## Task 3 Player on board

Task 3 or as it is called in the system, Board player is used to read the input from the player using the boards joystick to control one of the pads of the ping pong game what it does is that when called the task checks what the input value of port c on the microcontroller is and if port c pin 6 is high that means the joystick has been pressed upward what the task then does is that it checks the pads current position and if the pad is at the top of the game boarder it will then do nothing but keep looping the task to listen for downward press from the joystick but if the player pad is not at the top most position it will move one place up and then loop back to the beginning to listen for new input from user. If port c pin 0 is high at the beginning, we do the same as described for the upward movement but for the downward movement.

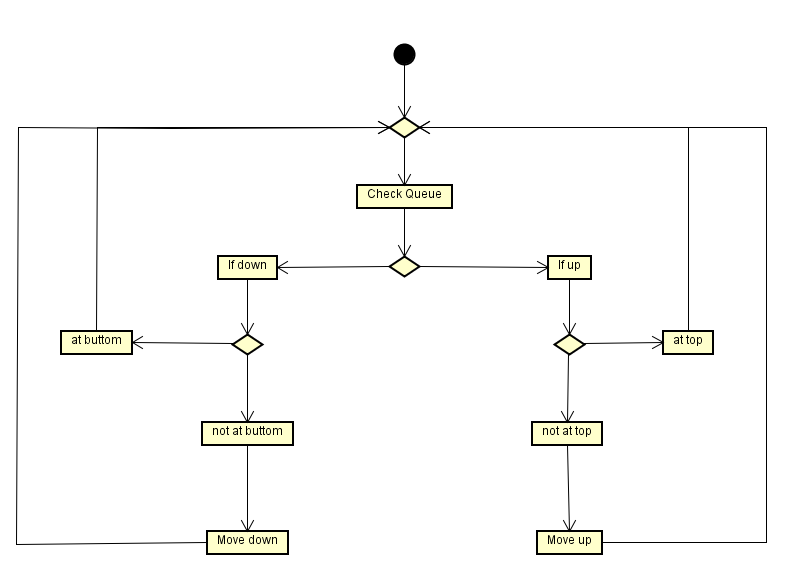
### Task 3 Player on board Diagram



## Task 4 Player Pc

Task 4 or as it is called in the system, Serial player is used for reading the input for player two from the pc´s keyboard using the up and down arrows. What this task does is the same as preciously describe in task 3 but instead of listening for port c the task checks the message queue for the upward key stroke and the downward keystroke the message queue is derived from the serial connection task described earlier in this report.

Task 4 Player Pc Diagram



# Implementation.

Code snippets here and explaination

# Test.

Picoscope and calculations here

## System timeline.

# Result.

In the result part, we have concluded from our testing of the Ping pong game what functions are working and what functions are not working. Every required and necessary function are listed below as working or not working.

Working:

* All Four tasks
* Shared resources (Display, serial connection)
* semaphores/mutexes
* input from joystick (Local player moves)
* input from Computer (Serial player moves)
* Game state machine (ball moves and resets)
* Display working as interrupt
* Protocol working.

Not working:

* CLC not implemented
* Point system not completed (displays points but does not reset)
* Use of two boards as players (not implemented)

# Discussion.

# Conclusion.

# References.