**List of requirements:**

Functional:

Player 1 (on game console):

* See the movement of the car on the display
* Move with the car to sides using joystick
* At the end of the game, display who won
* Timer controlled tasks DMS or RMS

Player 2 (on PC/Laptop):

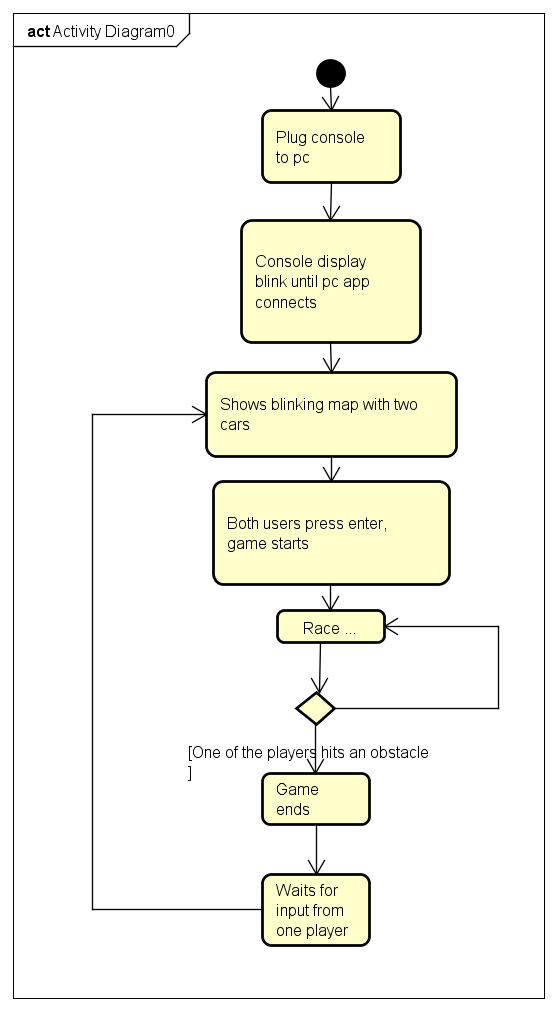
* Start/Load the game
* Play by using simple inputs (arrows) instead of the controller
* Be able to see the score, game time, progress of the game
* End/terminate the game
* At the end of the game, see who won

Non-Functional:

* The system must be implemented with FreeRTOS
* Need to have our protocol with flow control and error detection
* Serial connection between game-console and PC
* Code should be unit tested
* We must use semaphores or mutexes
* At least 3 tasks
* At least 2 hard real-time tasks

Constraints

* We use only given hardware (ATmega 3247, DOT matrix, usb serial communication, joystick, R2R DA)
* PC app in C
* Use oscilloscope for computation time measurements
* Only 2 players

****Activity diagram

**Design**

Resources

Matrix – represents the diodes lit up on display, might be represented as integer[14] or bit[10][14].

Input queue – accepts inputs from Joystick and Input samplers and stores them for processing by the Constrain checker.

**Tasks**

Controller

**Display updater –** serves the function of updating the display according to the matrix

* + resources: Matrix

**Game logic** – serves the purpose of updating the Matrix correctly based on the input queue but insures that executed inputs make sense and checks if cars and obstacles collide to decide the looser.

* + resources: Matrix, Input queue -> hard RT

**Obstacles –** updates the display with obstacles and takes care of generation and movement of them

* resources: Matrix

**Joystick sampler** – takes the input of the joystick and formats it into the suitable form by adding source.

* + resources: Input queue

**Communication sender** – resources: computer receiver -> hard RT

**Communication receiver** – resources: Input queue -> hard RT

Computer

**Communication sender** – resources

**Communication receiver** – resources

**Input sampler**

**We use rate monotonic scheduling**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Period** | **Deadline** | **Priority** | **Computation** |
| **DU** | **100ms** | **100ms** | **3** | **0.2ms** |
| **GL** | **75ms** | **75ms** | **4** | **0.08ms** |
| **JS** | **150ms** | **150ms** | **2** | **0.04ms** |
| **CS1** | **30ms** | **30ms** | **5** | **0.06ms** |
| **CR1** | **30ms** | **30ms** | **6** | **0.08ms** |
| **OBS** | **1000ms** | **1000s** | **1** | **1.5ms** |

Questions for teachers

Deadlines for tasks.

3.3 ms period for display updater?

How to use RTOS on pc.

How to use oscilloscope and R2R.

Should input be sporadic?

Can we implement existing protocol or should design one?

Group No.8