SEP 4E information

Report demands:

Your task is to hand in two reports: a process report and a project report.
The process report is the least important: a diary (a couple of pages) and a page or two about how you used your chosen project methodology is enough.
The project report is more important but it doesn't have to be long. 10-15 well written pages and diagrams are plenty.
Regardless of your project methodology your project report should be structure as

- Requirements
- Analysis
- Design
- Test / Implementation

Requirements. The requirements part is where you describe what the user(s) want your system to do. You describe user interaction (use case diagrams) and describe the system functionality.

Analysis. The analysis part is where you analyse the requirements, and try to understand the essence of these.
This is also the place where you describe the physical constrains set by electronic and mechanics (how often can a user shift a contact, what is the best update rate for the display etc.).

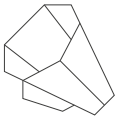
Design. The requirements and analysis of the system are used to make the software design. What kind of tasks do you need? How often should they run? Which are hard real-time tasks and which are soft? etc. Use task diagrams like the ones you can find in the books to show your tasks and their interaction. Use the theory to find a scheduling strategy of your tasks and prove they are schedulable (i.e. that hard real-time tasks will always meet their deadlines).
Remember there is not a single right design: there can be many alternatives with strengths and weaknesses. So it is important that you explain why you chose the design you did with its strengths and weaknesses. And if you are aiming at a high grade: discus what alternative you could have used.

Test / Implementation. Since you are only making a prototype there is not much to write about implementation. You should however write something about how to test software and hardware. It is very important to use an oscilloscope to determine how much time your tasks are using in order to prove that your system is schedulable (Design and Test is really an iterative process: in order to make your design you need to have an idea of how much time your tasks are spending and in order to prove our design you need to know how much time your tasks are spending).
It would be nice if you would show the use of unit-testing of parts of the code.

Design principles:

Good overview and modularization.
This is an important goal and will make it easier to debug, expand and reuse your software.
With timer controlled tasks this is well archived and choosing RMS, DMS or some other scheduling order with fixed periods you can easily prove that your system will hold the deadlines.
Time delay controlled tasks are simpler to implement but it is harder to exactly determine their period and will thus give a too conservative estimate if they are hard real-time.
Finally, you need to consider sporadic tasks.

Information sharing and signalling between tasks.



Obviously you should not put everything into one task and therefore the tasks have to communicate. One simple and safe way is through semaphores or mutexes so regardless of your system you must use semaphores or mutexes for protecting data and also for signalling events between tasks.
About data exchange between tasks you could use a generic linked list. The nodes could consist of pointer to data and an opcode describing the type of data.
Alternatively, you could just make global variables for data or better collect them in a struct. The data should be protected by encapsulation which can be done by declaring the struct static and then call functions to read/write from the struct.
What design you choose is up to you, but if you have ambitions about getting a high grade you should use an advanced solution (even if it is an overkill).

Minimum requirements to the system

- You must implement a system based on the document *SEP-RIE Project Idea .docx*.
- All mandatory parts must be implemented in order to pass.

Minimum requirements to the embedded software

- You must use at least three tasks.
- At least two tasks must be hard real-time.
- Some data must be used by more than one task.
- You must use semaphores or mutexes to protect data.

Minimum requirements to use of hardware

- Your software must run on your Game Controller board and make use of the following hardware components:
  - Serial Communication
  - Joystic
  - DOT-Matrix display
- R-2R D/A converter for measurements of task times. If you haven't made this circuit in the RIE course, you will have opportunity to make one during the project period.

Minimum requirements to the documentation and the project report:

- You must unit test parts of your application.
- You must use an oscilloscope to find computation time (C) for your tasks.
- Your system must give real-time guaranties and you must argue for this.
- Your documentation must contain at least one task diagram and a timeline diagram.

Minimum requirements for PC part

You must be able to control something (E.g. one of the users in a two user game) via the serial connection on the Game Controller board.

If you fail to meet the minimum requirements you will not pass the course.