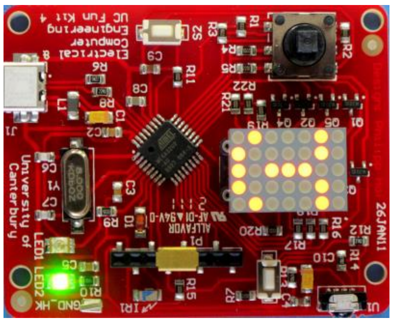
ENCE260 Embedded Programming Assignment

Semester 2 2025



Due: Your lab session during Week 6 of Term 4 2025
Worth: 10% of final grade
Groups: Students work in assigned groups of two.

1. Introduction

As a pair, you are to write a two-player game using two UCFK4 microcontroller boards, using the API functions provided. The goals of the assignment are:

1. To program a small-scale embedded system.
2. To use the API modules provided to abstract hardware dependence.
3. To structure the program in multiple C modules.
4. To actively use a version control system – git.
5. To collaborate on writing software with your partner.

2. General instructions

1. You will be allocated a project partner through Learn in Week 2 of Term 4. It is your responsibility to contact them as soon as possible. Your group is registered so each group will have an individual group number. Your git repository name is based on your group number. Login to your git repository and start using it once you have been assigned a partner. Contact Richard Clare (richard.clare@canterbury.ac.nz) if you haven't been allocated a partner or haven't been able to contact your partner by the end of Week 3.
2. Develop a two-player interactive game for the UCFK4 using the navswitch, pushbutton, display, and IR communications.
3. Demonstrate your program during your lab session during the last week of Term 4. The first hour of the lab session is for you to set up. You may volunteer to have your assignment marked in the first hour. In the second hour, when the lecturer or TA asks you to demonstrate your game, you need to demonstrate.
4. Your program must be compiled and demonstrated on the Erskine Lab computers, which run with avr-gcc 7.3.0.
5. Perform a final commit of your program source code to your git repository before the end of your demonstration session. The git repositories will be cloned at the end of the demonstration session. Only the default branch will be cloned and marked.
6. You are strongly encouraged to use the existing API functions in /utils , /drivers, and in /drivers/avr such as pio, timer and tinygl.
7. Both members of a group receive the same mark regardless of the contribution. Both members have to attend the demonstration to both be awarded the demonstration mark.
8. As the demonstration mark is given for a live demonstration, this mark cannot be appealed.
9. Before your demonstration, you will be asked to run `make clean` which deletes all compiled files so we can see you build your program from scratch. You should get in the habit of running `make clean` as you complete the assignment.
10. Your game must consist of a single application (binary). If you require different functionality on each board, then this must be determined at run-time, say by pushing a button. You cannot have two separate Makefiles. At the demonstration, both UCFK4 boards must be programmed from the same machine.
11. Ensure each of your source modules have both group members' names and usercodes at the top of each source file.
12. Place all your final application modules in your git cloned directory: /ence260-ucfk4/assignment/group_XXX, where XXX is your group number, e.g., group_123 (do not

create any subdirs in the dir). By doing this you will be able to *periodically* push your source files directly from your group_XXX directory to your repository.

13. Ensure that you have a working Makefile that will build your program. Note that a template Makefile is provided. Also note that marks will be lost if your application cannot be built by running make from the files in your repository.
14. Provide a README file with instructions on how to play your game and an AI statement.

3. Assessment

Your programs will be compiled and checked for compiler warnings and errors. Marks will be based on the following:

1. How well your program works during a short inspection
2. The complexity and originality of your game.
3. The modularity of your programs.
4. The readability of your programs (consistent formatting, commenting/documentation, avoidance of embedded constants (magic numbers), consistent naming).
5. The use of git for version control, and the README to describe how to play the game.

The marking rubric for the assignment (demonstration and source code) is available on Learn.

4. Plagiarism

Every year we detect plagiarism in this assignment. We use software to check your source code with material on the web (github etc.), against all groups from this and previous years, and the /apps directory. Any plagiarism will not only result in zero marks being awarded for this assignment, but a referral to the University Proctor. You can and should use the API functions provided. Students repeating the course must complete the assignment, and cannot re-use code from previous attempts at the course (this is still considered to be plagiarism).

5. Suggestions

1. Start with the 2-way communication program you developped in Lab 3.
2. Look at the example apps (in /apps) and the header examples, i.e., pacer.h, button.h, tinygl.h, ir serial.h, etc.
3. Look at the on-line documentation on Learn
   https://learn.canterbury.ac.nz/mod/page/view.php?id=4217084
4. Use the Blue LED to help debug your code.
5. Write a number of simple test applications, using one of the simple example applications (such as /apps/updown2) as a template.
6. Follow a consistent coding style. It does not matter what this style is (e.g. snake or camel).
7. You don't have a lot of memory (RAM or EEPROM) to play with, so keep things simple.
8. Ensure you commit your game regularly to your git repository.

Generative AI Tools Are Not Restricted for This Assessment
In this assessment, you are permitted to use generative artificial intelligence (AI) to assist you in any way within the bounds of academic integrity. You must appropriately acknowledge any use of generative AI in your work. Please include a statement of AI use in your README file, clearly indicating which AI tools were used and how they contribute to your assessment. If AI was not used in the assignment, this should be stated in the README file.