School of Electronic and Electrical Engineering
FACULTY OF ENGINEERING

Craig A. Evans* and Alexander Valavanis

UNIVERSITY OF LEEDS

Use of a Hand-held Gaming Platform for Teaching Object-oriented Programming to Embedded Systems Students

Overview

Embedded systems modules feature in all Electronic and Electrical Engineering degree programmes. Many of these courses will involve programming microcontrollers (MCUs).

C has historically been the most commonly used language for embedded systems programming; however, the growth of development platforms such as Arduino[1] and Mbed[2] has seen an increasing use of C++ and its object-oriented features.

Many software libraries (classes) are now available that make interfacing with different types of hardware (such as sensors and displays) very simple. It is therefore important that students are familiar with object-oriented programming (OOP).

With this in mind, our Embedded Systems Project module was re-designed. We use the online Mbed™ compiler along with an Arm® Cortex™ MF4-based MCU development board.

We use an engaging project-based learning approach in which students develop a game for a hand-held gaming device. This allows students to develop their creativity as well as learning important software engineering principles that are embedded within the curriculum.

Delivery

The module is made up of 11 three-hour laboratory sessions. There are 7 taught laboratory sessions covering Printed Circuit Board (PCB) design, PCB assembly, tasks and interrupts, finite state machines, software libraries, joystick and LCD interfacing and an introduction to game design patterns.

The remaining laboratory sessions are used for independent project work with students developing an Atari-style game e.g. Pong, Asteroids, Pac-Man or Snake.

Assessment

A range of assessment is used to reflect the broad range of skills and knowledge that are required in order to complete the project.

An in-course test is used to assess the understanding gained during the taught laboratory sessions. The code developed during the game project is submitted to assess the use of correct software engineering techniques and the level of creativity demonstrated.

The students also write a technical report which details their approach to the software design and implementation. Finally, a project exhibition is held in which the students demonstrate their completed projects.

Game Development

Developing a game provides an engaging and creative project for students to work on.

A game also provides a good base for introducing OOP concepts. Characters, levels, vehicles, weapons etc. map nicely into classes and offer a more tangible way in which to picture objects and understand how they interact via methods. This is in stark contrast to the more abstract ways in which OOP can be taught in terminal-based programs.

It also offers a nice balance between writing the game logic and low-level hardware interfacing (e.g. Joystick, LCD, LEDs, Piezo, Accelerometer).

Hardware

During the taught laboratory sessions, the students follow a screen-cast and learn how to design PCBs using EAGLE software[3]. During this activity, they create a hand-held gaming platform (the 'Gamepad').

The Gamepad contains a MCU development board, LCD, joystick, several push buttons, LEDs, a potentiometer and a piezo buzzer.

They assemble the designed Gamepad (including some surface-mount soldering) and this is used in all subsequent taught laboratory sessions. Prototyping boards ('breadboards') are often used in embedded systems courses but circuits built on these are liable to loose connections making them hard to transport without requiring them to be re-built.

The use of a robust PCB-based platform (with a protective carry-case) makes it trivial to safely transport and better allows students to work on their projects outside of scheduled laboratory time.

Software Engineering

Software engineering is often taught in modules separate from embedded systems. In this module re-design, we took the opportunity to embed software engineering within the curriculum.

The taught laboratory sessions introduce software engineering techniques such as:

- Object-orientation
- Code structure
- Version control
- Code documentation

The assessment criteria for the project then explicitly covers these aspects, encouraging students to follow best practice through the course of the project and not just focus on the final deliverable, as often can be the case.

Object-orientation

During the taught laboratory sessions, students are introduced to the concept of C++ classes through the development of software libraries for the I²C accelerometer and magnetometer that is on-board the MCU development board.

They are also provided with software libraries (C++ classes) for the Gamepad and LCD which allows easy interfacing with the peripherals and enables them to better focus on the game implementation.

A sample project (Pong) is also provided for reference which demonstrates how to break down a game into appropriate classes (e.g. Paddle, Ball, Physics Engine).

Code Structure

The taught laboratory sessions introduce concepts such as functions, arrays, structs, finite state machines and common game design patterns to help enable the students to write better structured code.

To further encourage improvements to the structure of the project code, limits on the length of functions and depth of nested if-statements were imposed with penalties applied for exceeding these suggested lengths. It is very common to see students use deeply nested if-statements when implementing game logic with the code becoming extremely difficult to debug and maintain.

Version Control

The online Mbed compiler has a built-in version control system. As well as being useful for distributing example and template code during the taught laboratory sessions, it is also used for the assessment and submission of project code.

During the project, students are encouraged to regularly commit changes to their code. This allows them to implement new features into the game without the risk of breaking the overall functionality of the code and means that they always have an up-to-date backup on the Mbed servers.

The commit logs are studied on completion of the project. This allows the module staff to easily observe the development of the code and makes it easier to spot potential plagiarised code (i.e. large amounts of code that suddenly appears between commits).

Code Documentation

The online Mbed compiler has a built-in documentation system based on Doxygen[4].

Commenting and documenting software is often a task neglected by students, who instead prefer to focus on the functionality of the software, in the belief that it will lead to higher marks. This is understandable as they are rarely exposed to a situation in which they must rely on documentation in order to maintain or debug code that has been developed by others.

The assessment criteria for the project requires a fully-documented Application Programming Interface (API) for every class along with useful and through in-line commenting.

Outcomes

Since the re-design of the module, there has been a noticeable improvement in the quality of the code submitted by students. Feedback from students has been very good, with many noting how engaging they found the project. In addition, their understanding of OOP improved as they saw how the classes interacted as the game played out on the LCD.

References

[1] Arduino, arduino.cc
[2] Mbed, os.mbed.com
[3] EAGLE, https://www.autodesk.com/products/eagle/overview
[4] Doxygen, doxygen.org

*C.A.Evans@leeds.ac.uk