LEARNING

GUIDE

Faculty of Engineering and Information Technology

Electrical Engineering

48434
Embedded Software

Spring 2015

SUBJECT GUIDE

Welcome

Embedded Software is a subject in the Engineering Degree course for students who major in Electrical Engineering or Information and Communication Technologies (ICT).

This subject focuses on developing a set of field-of-practice skills and knowledge:

- It develops an applied base for your field-of-practice knowledge.
- It develops competence in the use of software development tools through laboratory-based project work and problem-based learning.
- It lets you apply core skills and knowledge to a field-of-practice.

Embedded Software is a subject in the last year of the course – it assumes that you have developed a proficiency in academic and information literacy skills, and provides specialised knowledge for a part of your field-of-practice. It also helps you to apply that specialist knowledge to practical, real-world problems in a laboratory setting and prepares you for the graduate workplace.

You will be expected to take on a significant responsibility for your own learning. While self-managed learning offers you choices about how and when you study, we also understand that you will learn best if there are convenient opportunities for you to interact with fellow students and course staff.

Therefore, the subject provides a balance between the convenience of independent learning and the stimulation of academic life. We hope you enjoy the content, learning experiences and assessment tasks that make up this subject as well as the benefits of managing your own learning.

Your Subject Coordinator

Dr Peter McLean is a Senior Lecturer at UTS in the School of Electrical, Mechanical and Mechatronic Systems within the Faculty of Engineering and Information Technology. Subjects taught include Electronics and Circuits, Introductory Digital Systems, Fundamentals of Electrical Engineering, Circuit Analysis, Signals and Systems, Data Acquisition and Distribution, Digital Electronics, Analog Electronics, Signal Processing, Embedded Software, Power Circuit Theory and Power Systems Operation and Protection.

He has undertaken numerous research projects in collaboration with industry that normally involve the development of embedded systems hardware and software. These include microcontroller-based power system protection devices, DSP-based power-line carrier systems and a broadband Internet distribution system for the home.

Where this subject fits into the course

This subject is a Stage 7 field-of-practice subject in the Embedded Systems thread which is a part of the Electrical Major and ICT Major within various Bachelor of Engineering Degrees.

The need for this subject

It is assumed that you have already been introduced to and attained competence in the C programming language, digital logic design fundamentals, and learnt to design simple sequential programs for a device other than a PC. In this subject you will gain experience in the design of software for an embedded application. It will be seen that embedded software draws on many fields of engineering expertise, and that techniques of synthesis are highly dependent on system specifications.

The subject lays the foundation for many areas of further interest to the engineer – autonomous systems and robotics, software architecture, real-time operating systems, signal processing and numerical methods.

Subject aims and objectives

The objective of this subject is for you to design and test software for an embedded system.

You will bring together many elements of engineering – system specification, design, simulation, testing and management – all in the context of a high level of integration between electronics and software.

The technical content of the subject aims to develop the basic structure, operation and design of embedded systems from the software perspective. The subject will give you practice in designing software for real embedded systems.

"The artist is nothing without the gift, but the gift is nothing without work. "
- Emile Zola (1840-1902)

Skills in writing software, interfacing, debugging and experimental verification are developed through a series of laboratories. A project in which you analyse, design, implement and test part of an embedded system contextualises nearly all the technical content and makes use of the previously acquired skills.

Three engineering themes permeate the subject. The first theme is the need for a systems perspective in engineering – you will need to analyse and dissect (through a requirements specification) and eventually synthesise in a hierarchical manner (through software design). The second theme is that you will be expected to draw knowledge from a wide variety of sources – previous subjects, industrial experience, industry-produced datasheets and application notes and the Internet. The third theme is that of the need for engineers to take responsibility for their own professional development. You will be responsible for your own learning – which will encompass requirements specifications, mathematical modelling, electronics interfacing, software design and testing.

Finally, the subject will prepare you for more advanced topics on software systems, operating systems and signal processing which you may encounter in professional practice and in further subjects.

Content

"In theory, there is no difference between theory and practice. But, in practice, there is. "
- Jan L.A. van de Snepscheut

The content covered is divided into the following sections:

1. Embedded Systems
2. Embedded C
3. Microcontroller Architecture
4. Memory
5. Interrupts
6. Timing Generation and Measurements
7. The Embedded Software Tool Chain
8. Concurrent Software
9. Interfacing
10. Fixed-Point Processing
11. Real-Time Operating Systems
12. Design Project

Each of these sections addresses an important aspect in modern embedded systems. The intention is that, as you work your way through the subject, your learning will be cumulative. That is, the content you cover in one section should directly help you to understand the topics that follow. A weekly learning schedule, based on a recommended study sequence of the sections, is given in the Study Guide. For each of the above sections, a separate list of topics and suggested reading is also provided in the Study Guide.

Below is a brief summary of the content that is later covered in detail in the lecture notes.

Prerequisite knowledge

You are expected to have successfully completed subjects in the C language and Introductory Digital Systems.

Section 1 – Embedded Systems

An overview of embedded systems is given, before a specific example is treated in detail. The architecture of a popular 32-bit microcontroller is given, in terms of hardware modules and a programming model. Various features of the microcontroller are highlighted, including its architecture, memory map, universal serial bus, serial peripheral interface, enhanced capture timers, analog-to-digital converter, pulse width modulator, and non-volatile Flash memory. Schematics for the hardware platform will be given, showing various pieces of peripheral and interfacing hardware that will be used in the laboratory.

Section 2 – Embedded C

Aspects of quality programming, self-documenting code, modular software development and layered software systems will be covered. Special attention is given to the application of the C language to microcontrollers with limited resources.

Section 3 – Microcontroller Architecture

The microcontroller architecture will be examined. Microcontrollers have a wealth of built-in peripherals. Some of these peripherals will be examined in depth. The clock and reset generator module will be examined in detail. The universal serial bus (USB) is often used to connect to external hosts such as a PC. The encapsulation of external devices in software using “device drivers” will form part of the laboratory program.

Section 4 – Memory

Modern 32-bit microcontrollers have many types of memory, such as internal Flash, SRAM and external SDRAM; as well as special function registers that are memory-mapped peripherals. Utilising these different memories and registers requires special software techniques in both C and assembly language.

Section 5 – Interrupts

Interrupts are the key to building real-time embedded systems. The interrupt structure and hardware support for interrupts on a 32-bit microcontroller will be examined. Special compiler support for interrupt service routines will be highlighted.

Section 6 – Timing Generation and Measurements

The enhanced capture timer module of a 32-bit microcontroller will be examined to see how periodic and aperiodic interrupts are generated as well as how external event capturing can be used to simplify software tasks.

Section 7 – The Embedded Software Tool Chain

The tool chain specific to a 32-bit microcontroller will be examined in depth, including projects, editing, compiling, libraries, linking, the map file, startup code, vector table initialization, application loading and ASCII encoding of binary data. Debugging strategies will also be covered.

Section 8 – Concurrent Software

Foreground and background threads will be covered, as well as multithreaded applications and the basis for real-time operating systems. The concept of shared resources and some mechanisms for accessing them using semaphores and critical sections will be discussed. The concept of thread scheduling will be reviewed.

Section 9 – Interfacing

"Whether you think that you can, or that you can't, you are usually right."
- Henry Ford (1863-1947)

Standard parallel digital interfacing of external devices such as input switches and keyboards, liquid crystal displays and output LEDs is looked at in terms of hardware and software. A microcontroller often needs to handle analog data, such as an automatic control system, or a measurement system. The theory behind choosing a sample rate, data scaling and limiting and quantization will be briefly examined. Methods for obtaining, operating on, and producing analog data at the required rate will be reviewed. Peripherals such as an analog-to-digital converter and a pulse width modulator (PWM) will be looked at in detail. The serial peripheral interface (SPI) will be examined – it is used to connect to chips such as analog-to-digital converters, EEPROM and Flash memory, LED display drivers and many other special purpose chips.

Section 10 – Fixed-Point Processing

Microcontrollers are limited in their data handling capabilities, as they often need to process data in real-time and most do not possess hardware floating-point capabilities. Finite word length effects will be given and methods to overcome them will be examined. Some numerical methods will be presented for the integer evaluation of difficult results such as the square root.

Section 11 – Real-Time Operating Systems

An overview of a real-time operating system (RTOS) is given. The process of thread scheduling, preemption and thread switching is examined in detail with a simple implementation shown for a priority-based preemptive operating system. The design of application software for use in a system with an RTOS is discussed.

Section 12 – Design Project

This section brings all the other sections together in a project that requires the analysis and design of an embedded system. You will be required to interpret specifications and come up with sound engineering designs using a variety of methods. The designs will be implemented and experimentally verified.

Other subject information

The following information takes precedence over the default policies outlined in section 3.3.1 of the Faculty's Student Guide.

Internet

The subject uses UTSOnline which contains the subject documentation and links to important learning aids. The URL is:

http://online.uts.edu.au

web

You should regularly visit and explore the web site to keep informed of any important announcements such as timetable or assessment changes.

Lectures

You should attend all the lectures. They normally occur once a week with a one hour duration. During the lectures you will have the opportunity to meet with fellow students and with your subject coordinator who will answer questions and highlight selected topics.

face to face session

Software

A cross-compiler for the C language and an associated toolchain for the Kinetis series of microcontrollers will be used extensively throughout the subject as a means of programming an embedded system.

The computer laboratories have a freeware version of Freescale's Kinetis Design Studio Integrated Development Environment which you may wish to obtain:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KDS_IDE

The subject will rely heavily on a freely available Version Control System and Windows Explorer shell extension called TortoiseSVN:

http://tortoisesvn.net/

Laboratories

The laboratories are structured sessions that allow you to put into practice the knowledge delivered in lectures, using specialised equipment.

Twenty-four hour access to the Embedded Systems Laboratory will be given to students during the semester.

Laboratory access is contingent upon successful completion of the UTSOnline Safety Induction Quiz.

Assessment

Assessment for this subject is criterion-referenced. This means that your performance is measured against a set of criteria, not against the performance of other students.

The assessment criteria for this subject

In assessing your performance we will be looking for evidence that:

"Not everything that can be counted counts, and not everything that counts can be counted." - Albert Einstein (1879-1955)

- You are able to efficiently carry out an accurate analysis of the requirements of embedded systems which are similar to those dealt with in this subject.
- You have understood the concepts used in embedded software design and are able to apply them to the design of practical systems such as simple embedded systems.
- You can distinguish between the different methods of implementing a real-time embedded system and know their limitations and how to apply them correctly.
- You have understood the methods of hardware interfacing and are able to apply them to practical embedded systems such as those covered in this subject.

Assessment tasks

assessment task

The assessment tasks and their weighting are given in the Subject Outline.

Assessment dates

All assessment dates are shown in the Study Guide.

Calculators

Programmable calculators **are** allowed for the mid-semester test.

Please remember

- Check the subject web site regularly each week to make sure you don’t miss any important announcements about assessment items.
- Submit all assessment tasks on the date due as extensions are very difficult to arrange.
- Keep a copy of all assessment tasks you submit.

STUDY GUIDE

There are three components to completing your study of Embedded Software. They are:

- reading the lecture notes and associated “readings”
- attempting the laboratory assessment tasks
- completing the project satisfactorily

To guide you through these tasks there is a Timetable.

Structure of the Timetable

The Timetable will help you manage your learning in Embedded Software. It does so through the following design features:

- It is organised in logical, linked and digestible steps, so that where your learning is headed remains clear. Each session of the Timetable refers to:
  - A lecture in the Lecture Notes. Each lecture may have associated readings that should be perused in that session.
  - Assessment tasks that should be started, or are due.
- The Timetable asks you, therefore, to be an active learner; not a passive reader. You should keep in mind that to achieve the necessary competence to pass this subject it is not sufficient to just read the pages of the lecture notes and readings a few weeks before assessment tasks are due. Apart from understanding the concepts given in the lecture notes, you also need to practice writing and debugging software and allow yourself sufficient time to reflect on what you have learnt.
- You can see what the learning tasks will be for each session of the Timetable before you begin. This enables you to mentally prepare for the learning tasks while you work through the session topic. In this way your learning stays focused on the main areas of the session; you don’t lose your way in the details.

The Lecture Notes

The lecture notes should be read before each session so that lectures can concentrate on particular topics of interest rather than trying to cover all the material.

Structure of the Lecture Notes

The Lecture Notes are an on-going development through a process of continual feedback from students attempting to learn the topics as well as a continual updating of the content as technology changes. Difficult or hard-to-grasp topics are expanded; or are presented in a different manner to the readings; or highlight the real-world application of the topic. Prerequisite material is often recapped. The focus of the Lecture Notes is towards the final project, so those topics that are important to this goal are treated fairly thoroughly. The Lecture Notes are therefore a complement to the readings, as well as a summary of the important topics.

Skim through first

Margin notes help you navigate the material

If you are already familiar with the material in any section or if you want to get an overall feel for what it contains, you may like to skim through it first, looking at the headings and margin notes.

Textbooks

Textbooks are where you find the detail of the topics covered in this subject.

Prescribed textbook

There is no prescribed textbook for this subject.

Reference textbooks

The following is a list of reference textbooks that delve deeper into the topics of this subject. They may be used for alternative explanations or you may consider purchasing them:

Valvano, J.W., *Embedded Systems: Introduction to ARM® Cortex™-M Microcontrollers, 5th Ed.*, CreateSpace Independent Publishing Platform, 2012. ISBN-13: 978-1-47-750899-2

Valvano, J.W., *Real-Time Interfacing to ARM® Cortex™-M Micro-controllers, 5th Ed.*, CreateSpace Independent Publishing Platform, 2015. ISBN-13: 978-1-46-359015-4

The Assessment Tasks

The Laboratories and Project are published in documents separate to the Lecture Notes. It is up to you to submit each assessment item on time. The due dates for assessment items are given in the Timetable.

Learning in partnership

Using a fellow student as a learning partner has been found repeatedly to be an important learning support. The idea is that you contact a fellow student, by whatever means is most convenient, to discuss your interpretation of a learning task, to check if your approaches are the same and to generally clear up any confusions which may have arisen. It has been found that well over half of the concerns students experience about their learning are to do with simply checking that they are 'on the right track' and can be solved using this method. If, however, the concern or uncertainty remains, it is then recommended that you contact your subject coordinator.

Your learning plan

Your time

Organising your time is a major challenge in learning. Leaving recommended readings and assessment tasks to the last minute is a common problem. To assist you with this challenge you may find it useful to plan your study time before you start work on this subject. First decide on the best place and time each week to study without distractions and then make sure to adhere to your own plan.

It is estimated that over a period of 13 semester weeks you should set aside a total of approximately 9 hours of study time each week. It is recommended that you break up those hours into at least two study sessions each on a different day of the week. This is a rough guide only, as people learn at different rates and from different levels of experience.

Timetable

DATE	LECTURE	READINGS	ASSESSMENT
1A 27 Jul	***L1: Embedded Systems*** Overview of Embedded Systems. Overview of Tower board. Freescale K70F120M architecture.	http://cache.freescale.com/files/microcontrollers/doc/user_guide/TWRK70F120MUM.pdf http://cache.freescale.com/files/microcontrollers/doc/ref_manual/K70P256M150SF3RM.pdf http://cache.freescale.com/files/32bit/hardware_tools/schematics/TWR-K70F120M-SCH.pdf	
1B 31 Jul	***L2: Embedded C*** Review of the C language. Kinetis Design Studio. Initializing and accessing I/O ports. Memory allocation. Self-documenting code. Modular software development. Layered software systems. Debugging.	Chapter 10 of K70P256M150SF3RM.pdf Chapter 11 of K70P256M150SF3RM.pdf	Start L1
2A 3 Aug	***L3: Microcontroller Peripherals*** Clocks and reset generator. UART. PC USB Interface. FIFOs. Polling. Tower serial protocol.	Chapter 56 of K70P256M150SF3RM.pdf http://cache.freescale.com/files/32bit/doc/quick_ref_guide/KQRUG.pdf Tower Serial Communication Protocol	
2B 7 Aug			

DATE	LECTURE	READINGS	ASSESSMENT
3A 10 Aug	***L4: Memory*** Flash memory. EEPROM. RAM. Special function registers. Memory-mapped peripherals.	Chapter 29 of K70P256M150SF3RM.pdf Chapter 30 of K70P256M150SF3RM.pdf	
3B 14 Aug			Start L2
4A 17 Aug	***L5: Interrupts*** Interrupts. Interrupt service routines. Hardware interrupts. Interrupt vectors and priority. Exceptions. Threads. Foreground and background threads. Re-entrant programming.		
4B 21 Aug			**L1 Due**
5A 24 Aug	***L6: Timing Generation and Measurements*** Timer module. Periodic timer. Output compare. Input capture. Pulse accumulator.		
5B 28 Aug			Start L3

DATE	LECTURE	READINGS	ASSESSMENT
6A 31 Aug	***L7: The Embedded Software Tool Chain*** Projects. Editing. Compiling. Libraries. Linking. The map file. Startup code. Vector table initialization. Loading.		
6B 4 Sep			**L2 Due**
7A 7 Sep	***L8: Concurrent Software*** Threads. Schedulers. Operating systems. The semaphore. Mutual exclusion with semaphores. Synchronisation with semaphores. The producer / consumer problem with semaphores.		
7B 11 Sep			Start L4
8A 14 Sep	***L9: Interfacing*** Input switches and keyboards. Analog to digital conversion. Digital to analog conversion. Serial Peripheral Interface (SPI). Inter-Integrated Circuit (I^{2}C).		
8B 18 Sep			**L3 Due**

DATE	LECTURE	READINGS	ASSESSMENT
9A 21 Sep	***Tutorial Week***		
9B 25 Sep	***Mid-Semester Test*** Lectures L1-L8 inclusive.		**Exam**
VC 28 Sep	***Vice-Chancellor's Week***		
VC 2 Oct	***Vice-Chancellor's Week***		
10A 5 Oct	***Public Holiday***		
10B 9 Oct			Start L5
11A 12 Oct	***Embedded Software Project*** ***Overview of project.*** **[PROJECT HANDED OUT]**		
11B 16 Oct			**L4 Due**

DATE	LECTURE	READINGS	ASSESSMENT
12A 19 Oct	***L10: Fixed-Point Processing*** *Q*-notation. Other notations. Fixed-point calculations. Square-root algorithm for a fixed-point processor.		
12B 23 Oct			Start Project
13A 26 Oct	***L11: Real-Time Operating Systems*** Real-time kernel concepts. Reentrancy. Thread priority. Mutual Exclusion. Synchronization. Interthread communication. Interrupts. Memory requirements. Advantages and disadvantages of real-time operating systems.		
13B 30 Oct			**L5 Due**
14A 2 Nov	***Embedded Software Project*** Project work. [NO LECTURE]		
14B 6 Nov			

DATE	LECTURE	READINGS	ASSESSMENT
15A 9 Nov			
15B 13 Nov			
16A 16 Nov			
16B 20 Nov			**Project Due**