

Computing Projects

1 Introduction

My main areas of interest are embedded systems, real-time, networking and formal methods.

The key requirement for doing a project with me is that you should be committed to developing a piece of software. If you're trying to avoid programming as much as possible, you should probably talk to someone else.

The next section gives a list of particular topics that I have in mind for this year. Any one of them could form the basis of an interesting project. If one or more of these topics grabs your interest, do some googling to get a rough idea of what might be involved and then speak to me to develop your ideas further. If you don't like any of these ideas but you think you'd like to do something in a related area, speak to me about it.

2 Project ideas 2012-13

- Deploy and compare FreeRTOS and uC/OS-III on ARM platform
- Add support for FAT filesystem to embedded OS. Analyse performance
- Interface a communications module (e.g. Bluetooth, Zigbee, GSM, GPS, ...) to embedded system development board. Design and implement drivers and API. Test and analyse performance.
- Develop Controller Area Network (CAN) driver and high-level API
- Investigate a real-time communications protocol, e.g. Flexray, TTA, TT-Ethernet, MILCAN
- Design, implement and test 802.15.4 to CAN gateway
- Implement and analyse performance of 6LoWPAN
- Develop application and analyse performance of accelerometer module
- Investigate tightness of bounds of WCET analyser
- Apply model checking to analysis of security properties of a communication protocol
- Develop a tool to generate automatically correct ACLs from a specification of required security properties
- Investigate automatic code generation for CAN-based, time-triggered embedded systems with multiple tick rates derived from a shared clock.
- Investigate the use of the precision time protocol [IEEE 1588](#) in distributed embedded systems.

3 Writing your dissertation

Content and Presentation

Some of your content will be based on your literature review. This will involve reading some literature. Unless you have a plan, your reading will be inefficient. [How to read a research paper](#) is an excellent guide to forming a plan for reading a paper. Follow its advice.

[Writing for Computer Science](#) is a good general text book about effective communication in Computer Science.

If you don't want to splash out on a text book, there are several sites with links to a variety of advice about how to write clearly. Not all of the advice is directly relevant to writing a B.Sc. or M.Sc. dissertation in Computing or Engineering. The advice may not even be consistent. However, it is well worth reading, as you begin to form your own ideas about how to write a good dissertation. The sites that I find most useful are:

- [William Stallings How-To](#)
- [Carnegie Mellon: Advice on Research and Writing](#)
- [Writing technical articles](#)

However, before rushing to devour all of the hints and tips available from these pages, you should remember that the most valuable document for you is the **project handbook** for the module that you are studying. You should pay particular attention to the marking guidelines and ensure that, for each part of your dissertation, you have addressed **all** of the requirements of the marking scheme. And when the project handbook requires an approach that differs from that suggested by some other document, it is the project handbook that you should follow. All of the general writing guidelines in the world are of little benefit if you do not adhere strictly to the specific requirements of the organization to which you intend to submit your document.

Once you're clear about the advice in your project handbook, you may find the following helpful for parts of your dissertation. Remember that much of the advice below is written for academics intending to submit a journal or conference paper. You'll need to interpret it carefully to make it fit the requirements of a dissertation.

Abstract

- [How to write an abstract](#): excellent advice from Philip Koopman.
- [How to have your abstract rejected](#): may not be directly relevant to you but it still makes me laugh.

Main report body

- [How to organize your thesis](#): intended mainly for Ph.D. dissertations - but if you disregard the references to the need to demonstrate an original contribution to knowledge, this works for M.Sc. and B.Sc. dissertations too.

- [How to write a great research paper](#): you can guess this is advice about writing a paper, not a dissertation - but if you ignore the advice about length of each section, almost everything else applies to writing a dissertation as well.

Bibliography, references, citations

- [Cite them right](#): essential reading.
- Manage your bibliography with BibTeX.
 - See [Managing Citations and Your Bibliography with BibTeX](#) and [LaTeX/Bibliography Management](#) to get started.
 - Keep an archive of the papers that you read for your own use. Use [bibtex2html](#) to produce html versions of your bibliography. Provide links to full text versions whenever possible.

Style

- [Guardian style guide](#): I like the Guardian's style. It's a good guide to modern British English.
- [Henning Schulzrine](#) and [John Owens](#) have more specific advice about technical writing style.
- [Markus Kuhn](#) offers good advice about preparing and preserving your publications.

Tools

Writing

It's hard to imagine why anyone would write a Computing or Engineering project dissertation using anything other than [LaTeX](#). Its key benefit is that it allows you to concentrate on the **content** of your writing rather than its **layout**. Also, it handles automatically the production of lists of contents, tables and figures, and, using BibTeX, it makes it easy to produce a bibliography and to manage citations. It also enables you to construct more [beautiful](#) documents.

I've made available some [resources](#) that show you how to use this tool to produce a dissertation. The [Not so short introduction to LaTeX](#) should be used to answer your general LaTeX queries. There's also a useful [LaTeX Wikibook](#) that is worth a look.

Version control

Writing a dissertation takes quite a long time. During this time you will produce many versions of your dissertation. Occasionally, you will be disappointed to discover that the latest version of your dissertation is not entirely your best. You will be even more disappointed if you cannot recover those parts from an earlier version that you now realise were better than your latest version. You can avoid this disappointment by using a version control system to maintain all of your dissertation sources. There are several good, modern version control systems that are freely available. I prefer [Mercurial](#) for its simplicity and power. [Git](#) and [Bazaar](#) offer similar functionality and are also highly regarded. If you are not already using one of these tools, you should begin at once.