ELEC1201
Programming
Steve Gunn
Klaus-Peter Zauner
What is Programming?

• Instructing a machine to undertake a task
• Machines are good at repetitive tasks
• Machines have their own language (machine code)
  – Architecture dependent, e.g. ARM, AVR, Intel, ...
• Often program in a different language to that of the machine
• Higher-level languages are easier to understand.
• Portability
• Require a compiler or interpreter to translate
Hold on, I signed up to do engineering ... Why do I need to program?
Hardware/Software Co-Design

- Design decisions over what is done in hardware and what is done in software

- Dedicated hardware
  - Fast, but expensive to develop
  - Expensive to update once deployed

- Many hybrid options now available and commonplace

- Reconfigurable hardware
  - CPLDs, FPGAs
  - Microcontrollers
Microcontroller Proliferation

There are now more microcontrollers than people on this planet.

Pictures reproduced from Atmel documentation.
The C programming language

- One of the most widely used programming languages of all time: it can be used on almost any platform
- Low-level access to memory
- Maps efficiently to machine instructions
- Requires minimal run-time support
- Often an alternative to assembly

From http://en.wikipedia.org/wiki/C_(programming_language)
Why C?

- Many other languages are based on it
  - e.g. C++, Java, C#
  - Python, Perl, PHP (written in C)
- Most popular language for embedded programming
- Programming microcontrollers
- High-level language with low-level access
- Maps efficiently to many embedded instruction sets
- Other languages come and go ...
Related Activities

- **MATLAB**  *Numerical Modelling*
  - Prototyping/Scripting Language
  - Circuit simulation, control simulation

- **System Verilog**  *Hardware Design*
  - Hardware Description Language

- **C++**  *Advanced Programming*
  - Object-oriented Programming Language

- **Mathematica**  *Mathematical Programming*
  - Functional/Procedural/OO Language
  - Arbitrary precision arithmetic, Symbolic equation solver
Do I need my own computer?

- Yes – it’s 2018.
- For those who have yet to get one:
  - Laptop or desktop?
  - Linux/Mac or Windows?
  - Should I get the most expensive machine I can find?
- Virtual Machines
- Backing up your work
- Course content is designed to be platform neutral
What about Software?

- Text Editor*
- gcc compiler
- avr-gcc compiler

* Your choice, e.g. notepad++, text wrangler, genie, ...

If you wish you may use an IDE such as Eclipse.
Textbook

- Some copies in the library
- Buy your own copy? (E version)
- Useful resource for this module and later modules
- Platform neutral content
- Also online resources which you may find a suitable substitute
But, nothing is perfect ...

Check that the program code looks exactly like the listing below, then add a final newline character (hit Return after the closing brace) and save the program as “hello.c”

```c
#include <stdio.h>

int main()
{
    printf( "Hello World!\n" ) ;
    return 0 ;
}
```
That’s better ...

```c
/*
   Name : Hello.c
   Author : Steve Gunn
   Version :
   Copyright :
   Description : Hello World in C, Ansi-style

*/
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello World!");
    return EXIT_SUCCESS;
}
```
### Teaching Methods

- Lectures (26); Labs (11); Clinics (8); Tests (2); Project (1)
Labs

• Undertaken in the Electronics Laboratories (Level 2, Zepler)

• Lab is 3 hours undertaken in standard laboratory pairs
  – Details about this in the lab induction meeting

• Assessed on four criteria by the demonstrators
  – Preparation, Progress, Understanding, Logbook
  – You fail the lab if you score zero in any of these categories

• **Preparation very important (3 hours)**
  – No preparation, you fail the lab.

• Your first lab is on Thursday (either a.m. or p.m.)
Clinics

• Purpose: to provide additional support to those new to programming. Opportunity to work through example code.

• Attendance not compulsory

• Who should attend?
  – Anyone who has never programmed before
  – Anyone who is finding the material difficult
  – Anyone scoring poorly in the laboratories
Module Overview

Hosted C Programming

Embedded C Programming

C Programming Project
Weeks 1-5

Hosted C Programming
Hosted C Programming

- No prior knowledge of programming assumed
- Aims to turn you into a C programmer in 5 weeks!
- “Teach Yourself Programming in Ten Years”, by Peter Norvig (Google) [http://norvig.com/21-days.html](http://norvig.com/21-days.html)
- “Hosted” is used to indicate the program is designed to run on a general-purpose computer (PC).
- Simple hosted programs usually execute some task and then terminate.
Outline

• Background reading from the course text
  – (2 chapters/week)
• 5 Laboratory exercises (C1-C5)
• 10 Lectures covering the material
• 4 Clinics (weeks 2-5)
• 1 Class test (week 5)
Weeks 6-11

Embedded C Programming
Embedded C Programming

• Builds on first 5 weeks.
• Aims to make you an embedded C programmer in 6 weeks.
• “Embedded” is used to indicate the program is designed for specific control functions as a component in a bigger system.
• Embedded programs can also be referred to as “firmware”.
• Embedded programs never normally terminate.
• Usually waiting to respond to events.
• Program flow more complex with the use of interrupts.
Your Embedded Platform

- AVR Microcontroller
  - 12 MHz
  - 64K Program Memory
  - 4K SRAM
  - Harvard Architecture
  - RISC Instruction Set
  - ADC, Timers/Counters
  - Serial Interfaces
  - 3.3V CMOS Logic Levels

- You build it in lab X2
- Used in labs C6-C11
- Used in the project D1
Microcontroller Architecture

Picture reproduced from Atmel documentation.
Outline

• No course text – datasheets/online resources.

• (X2 Lab in week 3-4 to build the hardware)

• 6 Laboratory exercises (C6-C11)

• 12 Lectures covering the material

• 4 Clinics (weeks 7-10)

• 1 Class test (week 10)
Project

- Brings together the material from weeks 1-11.
- Undertaken individually.
- Required to write two programs
  - Hosted program to read input and display responses.
  - Embedded program to control the hardware system.
- The two programs will communicate over a serial interface.
Outline

• One intensive week in the lab.
• First day develop hardware.
• Establish control of the hardware.
• Communicate from host to embedded platform.
• Integration and testing.
• Assessed on throughout the week.
Assessment
Assessment

• 5 Hosted C Labs (20%)
• Hosted C Test (20%)
• 6 Embedded C Labs (25%)
• Embedded C Test (20%)
• Project (15%) (A further 15% contributes to ELEC1200)

To pass the module you must score over 40% in total, you must pass 4 out of 5 Hosted C Labs, and you must pass 5 out of 6 Embedded C Labs.
Redemption

• There will be an opportunity to redeem failed labs during this semester. More details in the lab induction.

• If you fail to pass the module you will be referred and will have to be re-examined.

• Lab mark after opportunity to redeem failed labs is carried forward (50%)

• Remaining (50%) is re-assessed by an exam in September.

• Electronics students should also remember that there is a module in semester 2 called “Advanced Programming”.
How to fail the module

• Failing to do background reading
• Fail to do the laboratory preparation
• Fail to attend laboratories
• Fail to attend lectures
• Copy code from another source without acknowledging it
How to pass the module

• Program outside the labs
• Look at example code
• It’s ok to ask for help
• Develop your own programs for your own interests
Further Resources

• Website
https://secure.ecs.soton.ac.uk/module/1819/ELEC1201/29577/

• Web Tutorials

• Software Installation

• Student WiKi

https://secure.ecs.soton.ac.uk/student/wiki/w/ELEC1201