Intro

- In this course we're looking at programming physical hardware.
    - We're gonna program an Arduino in C.
    - An 8-bit microcontroller.
- We will have 2 hour labs for this module.
    - Probably on Wednesdays
    - There'll also be a project after a couple of weeks.
        - e.g. a digital piano, an IR morse code transmitter, tic tac toe
- website: http://cuc.ucc.ie/jpm/CS3514

C

- C is an old and popular language because it's extremely powerful
    - "high-level assembly language"
    - most large things are written in C (e.g. the JVM)
    - in C you have access to a lot of low-level control (e.g. specific registers, bit manipulations, memory allocations)

Microcontroller

- Whole board is the Arduino (open hardware)
- CPU you can get for like 50c
- A microcontroller is designed so you can connect individual chip pins to the environment (unlike PCs, microprocessors)
    - All about controlling, sensing, changing the environment

Arduino

3 things:

- Physical piece of hardware
- Programming environment (free software, made by MIT?)

  - Built around the hobby market (you write “sketches”, which are embedded into a C program)
  - Hopefully we’ll be going behind the scenes after a while and writing without this, using the shell and our own tools
- Community & philosophy
- Based on the AVR chipset – designed to be programmed in C
  - laid out so C can be compiled very efficiently to it
  - We’re using the ATMega328, which is quite old but it doesn’t matter

Hardware

- LilyPad (sensors for clothing)
- stamp-sized boards
- bluetooth
- boards are made to be stackable, so you can add e.g. an ethernet board to extend something
- Memory is quite limited, we’ll be looking at how we can squeeze as much memory out of the system as possible
- Flash memory is program
- Static ram is working memory (e.g. stack frame when a function is called, where variables are kept, etc.)
- EEPROM is for configuration data that will survive power outage
- Separation between program and data memory is called Harvard architecture (rather than Von Neumann)
  - You can have separate busses for them, do them at the same time, etc. ~

Peripherals

- counters, timers
  - registers that hold a number of bits and are incremented on every clock cycle
  - counter uses an external signal rather than the clock (e.g. increment every time a door opens)
- PWM
  - used to simulate analog voltage signals from digital signals

- ADCs
  - analog voltage (e.g. 5V) to a number (e.g. 1023 for 10 bits)
  - used for e.g. light sensing, temperature sensing
- Special power-saving modes
  - can be the difference between a battery lasting a day and lasting for months
  - factor of 2000

Sketches

Two mandatory functions:

- `setup()`
  - run once on startup (also reset)
- `loop()`
  - run continuously

Some built-in functions:

- `pinMode()`
  - determines whether a pin is working as an output or as an input
- `digitalWrite()`
  - allows you to write 0s and 1s (lows and highs) to pins

Compilation/Upload

- Bootloader takes up about 2 KiB
- Blink program uses 3% of the chip because the built-ins use a lot of code
  - We can decrease the number a lot by knowing how the background works and optimising it.