Teaching Team

SeungHwan Won  JingHuey Khor
Teaching Methods

• Lectures : Regularly three hours per Labs : Seven labs
• Clinics : Seven clinics
• Test : One final test
• Project : A final project

*It is noted that our current schedule can be changed whenever necessary.*
Lectures to be covered

• **C++ language**
  – Relationship between C and C++
  – Introduction to C++
  – Details of C++ and its examples

• **Data structures and Algorithms**
  – sorting an searching using structures, classes, templates, and maps

• **P1 to P7 labs and P20**: Each 3 hours except for P20
Labs

- Lab is officially 3 hours undertaken in the Electronic Engineering Laboratory
- **Preparation very important (3 hours)**
- Regarding P1 and P2, you must write your codes based on **C language**.
Lab schedule

Week 20: P1
Week 22: P2
Week 23: P3
Week 24: P4
Week 25: P5
Week 28: P6
Week 29: P7
Week 30, 31: P20 design exercise
Clinics

• Purpose: to go over example codes to improve programming skills

• The clinic will be used whenever you have a lab-based practice

• Who should attend?
  – Anyone who wants to learn example answers
  – Anyone who is finding the material difficult
  – Anyone scoring poorly in the laboratories
Course Textbook

- Use the e-book for your study
- It’s yours to keep
- Useful resource for this module
- Platform neutral content
- Extensive coverage
What about Software?

- gcc compiler, avr-gcc compiler, Eclipse IDE, AVRplugin, Qt4
Module Overview

C++ Programming

Software Engineering (Algorithms)

Raspberry Pi
# Brief comparison between C and C++

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language Type</strong></td>
<td>Procedural Oriented Language</td>
<td>Multi paradigm language including STL, procedural programming, functional programming and Object Oriented Programming</td>
</tr>
<tr>
<td><strong>Platforms</strong></td>
<td>Almost anything on the planet; requires recompile</td>
<td>Any, but libraries used can make it limited</td>
</tr>
<tr>
<td><strong>Developed by</strong></td>
<td>Dennis Ritchie &amp; Bell Labs</td>
<td>Bjarne Stroustrup</td>
</tr>
<tr>
<td><strong>Influenced</strong></td>
<td>awk, csh, C++, C#, Objective-C, BitC, D, Concurrent C, Java, JavaScript, Limbo, Perl, PHP</td>
<td>Ada 95, C#, Java, PHP, D, Aikido</td>
</tr>
<tr>
<td><strong>Influenced by</strong></td>
<td>B (BCPL,CPL), ALGOL 68, Assembly</td>
<td>C, Simula, Ada 83, ALGOL 68, CLU, ML</td>
</tr>
<tr>
<td><strong>Major Implementations</strong></td>
<td>GCC, MSVC, Borland C, Watcom C</td>
<td>GNU Compiler Collection, Microsoft Visual C++, Borland C++ Builder</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>C applications are faster to compile and execute than C++ applications</td>
<td>+-5% when compared with C if you know how to make a good use of C++. The performance of C++ and C are equal, since compilers are mature (now is 2012), if your C++ codes are much slower than C, don’t flame C++ but blame yourself.</td>
</tr>
<tr>
<td><strong>OOP (Object Oriented Programming)</strong></td>
<td>Not built in; has the freedom to setup structures to act like objects. Lacking the ability to encapsulate (announced private) is a big weak point.</td>
<td>Yes-polymorphism and inheritance, Classes.</td>
</tr>
<tr>
<td><strong>Appeared in</strong></td>
<td>1972</td>
<td>1985</td>
</tr>
<tr>
<td><strong>Execution Flow</strong></td>
<td>Top to Bottom</td>
<td>Bottom to Top</td>
</tr>
<tr>
<td><strong>Garbage Collection</strong></td>
<td>Manual; allows better management of memory.</td>
<td>Manual, you could manage the memory as C did, or use smart pointer, destructor to provide better and safer mechanism. According to your implementation, smart pointer can be zero runtime impact</td>
</tr>
<tr>
<td><strong>Usual filename extensions</strong></td>
<td>.c</td>
<td>.cc, .cpp, .hh, .hpp</td>
</tr>
</tbody>
</table>
Brief comparison between C and C++

[Keywords common to C and C++ languages]
auto, break, case, char, const, continue, default, do, double,
else, enum, extern, float, for, goto, if, int, long, register, return,
short, signed, sizeof, static, struct, switch, typedef, union, unsigned,
void, volatile, while

[Keywords specific to C++ language]
const_cast, dynamic_cast, reinterpret_cast, static_cast,
bool, catch, class, delete, explicit, false, friend, inline,
namespace, new, operator, private, protected, public,
template, this, throw, true, try, typename, using, virtual
C versus C++

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello World");
    return EXIT_SUCCESS;
}
```

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello World!";
    return 0;
}
```

- Your first thing to do is to compile, run and analyse the two codes.
Topics to be taught in S2

• Relationship between C and C++
• Introduction to C++
• Classes and Encapsulation
• Polymorphism
• Constructors and destructors
• Inheritance
• Template and Standard Template Library (STL)
• Objects
• Exception Handling
• Dynamic memory allocation
• GUI and UI design
• Use of OO modeling tools, including UML
• Introduction to the Raspberry Pi (including Qt5)
• Software lifecycle
• Storage (Files & Database)
• Introduction to data structure
• Stacks queues
Raspberry Pi 3 Model B

The Raspberry Pi is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.
# Raspberry Pi

<table>
<thead>
<tr>
<th></th>
<th>Raspberry Pi</th>
<th>Raspberry Pi 2</th>
<th>Raspberry Pi 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Released</strong></td>
<td>February 2012</td>
<td>February 2015</td>
<td>February 2016</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>ARM1176JZF-S</td>
<td>ARM Cortex-A7</td>
<td>ARM Cortex-A53</td>
</tr>
<tr>
<td><strong>CPU speed</strong></td>
<td>700MHz</td>
<td>900MHz</td>
<td>1,200MHz</td>
</tr>
<tr>
<td></td>
<td>single core</td>
<td>quad core</td>
<td>quad core</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>512MB</td>
<td>1GB</td>
<td>1GB</td>
</tr>
<tr>
<td></td>
<td>256MB Rev 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>Broadcom Videocore IV</td>
<td>Broadcom Videocore IV</td>
<td>Broadcom Videocore IV</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>SDHC slot</td>
<td>MicroSDHC slot</td>
<td>MicroSDHC slot</td>
</tr>
<tr>
<td></td>
<td>MicroSDHC Model A+ and B+</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>USB Ports</strong></td>
<td>2 on Model B</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>WiFi</strong></td>
<td>No built-in wifi</td>
<td>No built-in wifi</td>
<td>802.11n and Bluetooth 4.1</td>
</tr>
</tbody>
</table>
Weeks 14-15

P20: Collaborative Project
Outline

(1) Communication between two Raspberry Pi

• To experience interfacing a Linux-based system (Raspberry Pi)

• Should succeed in arranging communication between the two Raspberry Pi using a variety of techniques

• Lab time: Nine hours

• Notice: You will use your previous knowledge about serial communications.
Assessment
Assessment

• Seven Practical Lab Sessions (30 %)

• Final Test (40 %): 1.5 hours

• Collaborative Project (30 %)

* It is noted that percentage of each may be changed later if necessary.
Further Resources

• Website
  https://secure.ecs.soton.ac.uk/module/1314/ELEC1204/31154/

• http://www.cplusplus.com/

• Installation

• If necessary, further resources will be informed