

A collection of historical artifacts is arranged on a light-colored surface. On the left, a portion of a wooden chessboard with a blue and white checkered pattern is visible, featuring several chess pieces. Below the chessboard, there are several medals and orders, including a prominent white Maltese cross with a central emblem and a blue ribbon. A pair of round, gold-rimmed glasses with thin temples lies across the center. In the bottom left corner, a circular compass with a white face and black markings is partially visible.

The Role of Programming Languages in the Curriculum for Teaching Embedded and Real-Time Systems

Alan Burns
University of York, UK



Overview

- ◆ Issues to consider when attempting to produce a curriculum for embedded systems
- ◆ Experiences at York



Issues

- ◆ Curriculum is broad but not shallow
- ◆ Systems is hard to teach at undergraduate level
 - Hard to be coherent
 - Hard to attract students
- ◆ Postgraduate courses (MSc)
- ◆ Some refocusing of undergraduate courses



Computer Science

- ◆ Moving more towards Engineering
- ◆ Has many (but not all) of the basic material needed for an embedded systems curriculum
- ◆ Can move students towards careers in embedded systems engineering



York CS Degree

- ◆ A lot of maths
- ◆ Analogue and digital electronics
- ◆ Usual OS material, device drivers etc
- ◆ Model-based view of design
- ◆ Students can choose an Embedded Systems strand to their degree



Embedded Systems Strand

- ◆ Development work in 2nd year
- ◆ Networking, comms etc in 2nd year
- ◆ Real-Time Systems in 3rd year
- ◆ Control theory in 3rd year
- ◆ Hardware/Software co-design in 4th year
- ◆ High Integrity Systems in 4th year



Real-Time Systems

- ◆ To give cohesion to the material, and student enjoyment, we give a programming centred course
- ◆ Computer languages provide a framework to discuss many embedded system concepts
- ◆ We use language abstractions, experimental languages and mainstream languages (eg Ada, C with POSIX, Java)



Key Concepts

- ◆ Concurrency
 - Various models
 - Form of analysis, eg model checking
 - Common idioms – control loops, sampling etc
- ◆ Fault tolerance
 - Various primitives
 - Fault models etc
 - Atomic actions etc



Key Concepts

- ◆ Scheduling
 - Resource usage, including fault tolerance
 - Various forms of analysis, eg RTA
 - Distributed systems
- ◆ Low level programming
- ◆ Lab sessions
 - Student build and analyse embedded systems



Conclusions

- ◆ Programming and the study of languages is one way to bind a number of concepts together
- ◆ Advanced courses, MScs, are one way to approach the full embedded system curriculum – but ensuring students have a common set of pre-requisites is difficult