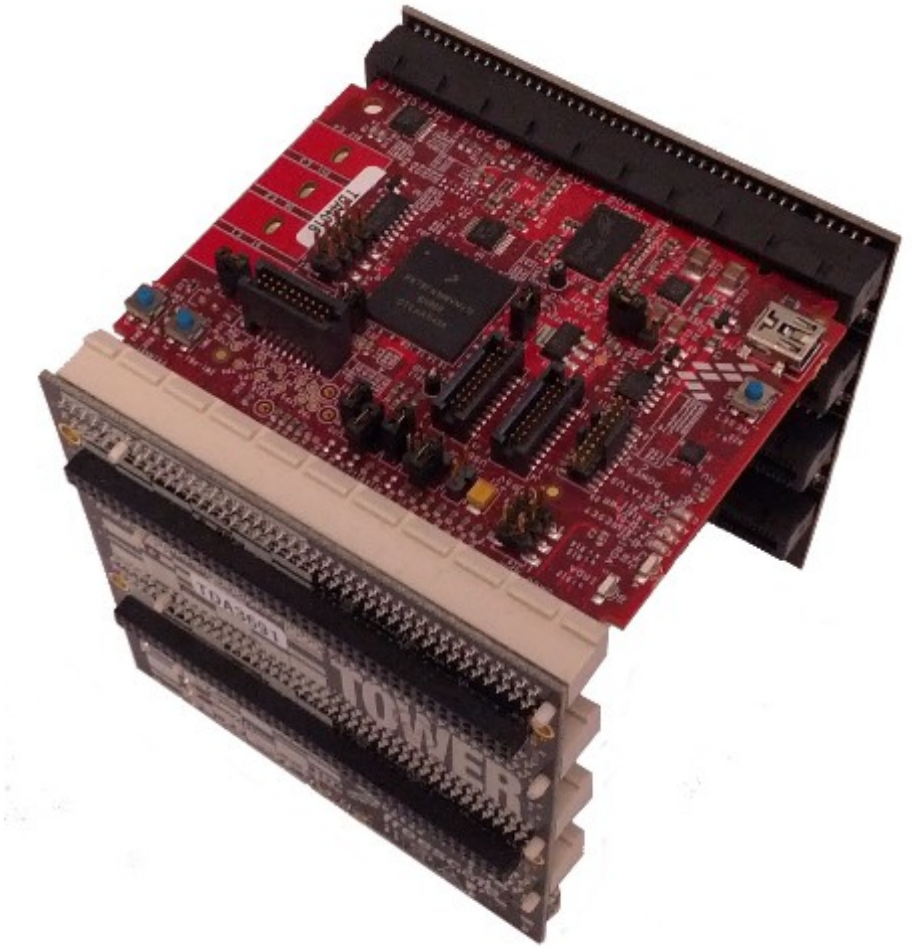
ES Coursework 1



Agenda

- Brief overview of coursework
- Introduction to hardware/software being used
- Information on 4 tasks to be completed
- Lab info
- Submission

Overview

- Coursework consists of programming for an embedded device running an RTOS
- First three tasks are introductory – these should be straightforward (most of the code you need is provided)
- Final task is more freeform and you may need to consult documentation on some features
- Final submission is by demonstration, although code should also be submitted

K70 Board

- Part of the Freescale Tower System
- K70 SOC contains:
  - ARM Cortex-M4F Core
  - Flash Memory
  - DRAM
  - Many peripherals (UART, Memory Controllers...)
- Fairly powerful for an embedded device
- Lots of memory – usually this is a constraint in embedded systems

MQX

- RTOS ported to various Freescale devices
- Contains standard library and drivers for many devices
- Lots of useful features and fairly easy to use

Freescale Toolchain

- Based on Metaware compiler – behaves differently to GCC!
- We're using the ARM backend (not surprisingly)
- Command line interface
  - A simple Makefile is provided
  - Could be wrapped by an IDE but you will need to do this yourself!
- Scripts are also provided for communicating with the board using OpenOCD (for flashing/debugging)

Coursework Tasks

- Three easy tasks:
  - Blink some LEDs
  - Set up the network interface/HTTP server
  - Set/display the Real Time Clock over HTTP
- One longer task
  - Security system

Task 1 – LED Control

- Mainly a 'getting started' task
- Introduction to using the board
- Two parts:
  - LED constantly on
  - LED on in response to button press

Task 2 – Web Server

- MQX has a built in network stack including an HTTP server
- This task involves serving a static web page over the network connection
- Only one part:
  - Serve static web page

Task 3 – Real Time Clock

- K70 board includes an accurate Real Time Clock module
- This task involves configuring and displaying the value of the RTC using the web server
- Three parts:
  - Handling CGI requests in MQX
  - Display the value of the RTC over HTTP
  - Set the value of the RTC over HTTP

Task 4 – Security System

- Much longer task – create a web-controlled security system running on the board
- Touch sensors represent sensors, LEDs represent alarms
- Each sensor/LED represents a different room

Task 4, Part 1 – Basic System

- Capacitive touch buttons represent motion sensors
- LEDs represent alarms
- Push buttons used to switch the alarm on or off, and stop the alarm if it has gone off
- When a touch button is pressed, the attached LED should flash

Task 4, Part 2 – Web Control

- Builds on Part 1
- The user should be able to:
  - Switch on/off the alarm via the web interface
  - Stop the alarm if it is going off
  - View the current status of each room
  - Individually enable/disable the sensor for each room

Task 4, Part 3 – Timing Control

- Builds on Part 2
- The user should be able to:
  - Use the web interface to view and set the current RTC value
  - Set times at which each alarm zone is enabled/disabled (or have a zone always enabled, or never enabled)

Labs

- The boards are in AT3.01 and can only be connected to specific network ports in this lab
- Boards and cables are in lockers at the back of AT3.01 and should not be removed from the lab
- Labs are Wednesday 14:10 – 16:00 and Friday 16:10 – 18:00 in AT3.01. A lab signup email was sent out a couple of weeks ago but there is some flexibility

Submission

- The first three tasks due 4PM February 13th
- Final task is due 4PM March 19th
- Submission consists of a demonstration during the lab, plus submission of code using submit system