CEC 450
Real-Time Systems

Lecture 1 - Introduction
Dr. Sam Siewert

UC Berkeley – Philosophy/Physics

University of Notre Dame, BS - Aerospace/Mechanical Engineering

Johnson Space Center, U. of Houston – UHCL Computer Engineering, Mission Control Center

U. of Colorado, Boulder, MS/PhD – JPL, Colorado Space Grant, Computer Science

CU Boulder Senior Instructor, Adjunct Professor, CTO, Architect, Developer/Engineer in Local Start-ups

U. of Alaska, Anchorage, Assistant Professor, Computer Systems Engineering, Alaska Space Grant

Embry Riddle Prescott, Assistant Professor, CESE

© Sam Siewert
Related Industry Background

General Experience (~25 Years in Embedded and Scalable Systems)
- 12 Years NASA JSC, NASA JPL / CU, Ball Aerospace
- 12+ Years Commercial Telecomm, Storage/Networks, Embedded, Digital Video

Instrumentation and Machine Vision
- Spitzer Space Telescope
- Unmanned Aerial Systems
- Robotics at CU-Boulder, Arctic Sensor Systems at U. of Alaska Anchorage

Software Engineering
- NASA Johnson and JPL (Shuttle Ascent/Entry Guidance, Deep Space)
- Intel, Emulex, Start-ups

Consulting
- Graphics, Storage and Networking, UAS/UAV
Course Goals and Outline

- Real-Time Embedded Components and Systems with Linux and RTOS, 2nd Edition, Sam Siewert and John Pratt, October 2015, 978-1942270041, Mercury Learning, Amazon, Notes

- A balance of 1/3 Theory, 1/3 Practice, and 1/3 Development [Project]

- http://mercury.pr.erau.edu/~siewerts/ce450/

- http://mercury.pr.erau.edu/~siewerts/ce450/CEC450-Syllabus-Fall-15.html

Use Manuscript Notes
Get User and PASS in Class
Published in October
Embedded Linux and FreeRTOS

Option #1 – Jetson Embedded Linux – POSIX RT
- Linux Provides only Predictable Response (Not Deterministic) and Only with Use of POSIX RT Extensions
- Some Labs will Require Use of Linux
- Final Project can use Linux or FreeRTOS

Option #2 – Use Texas Instruments Microprocessor and FreeRTOS
- Labs Developed by Dr. Brian Davis
- Labs Developed by Dr. Siewert, Ported from VxWorks to FreeRTOS
- Some Labs will Require Use of FreeRTOS
- Final Project can use FreeRTOS or Linux
Administrivia

Introductions
- Instructor (Office Hours) - http://mercury.pr.erau.edu/~siewerts/Schedule-Fall-2015.pdf
- Students (Introductions) – Please do Collaborate, but cite well!
- Policies - http://mercury.pr.erau.edu/~siewerts/cec450/policies/

ERAU ERNIE and Canvas
- Primary Assignment Management Tool - https://erau.instructure.com/
- Access via ERNIE - https://ernie.erau.edu
- Mercury Website - http://mercury.pr.erau.edu/~siewerts/cec450/

Course Information
- Attendance & E-mail list (please sign up on sheet being passed around)
- Lecture Notes at http://mercury.pr.erau.edu/~siewerts/cec450/documents/Lectures/
- Will post on Canvas as well

You Will Each Have Access to an Embedded Linux Jetson System (Shared with CS 415, HCI)

We have Texas Instruments Boards for Use with FreeRTOS
Huge Range of Embedded Systems...

- From High Performance Computing…
  [link to IBM DeveloperWorks]

- PowerPC: from Play-Station/X-Box to Blue Gene/L
  [link to LLNL]

- Reconfigurable HPC - [link to SRC Comp]

To Cosmic Origins

- Embedded PowerPC Spitzer Telescope
  [link to NASA Jet Propulsion Lab and Cal Tech University]
To Be Embedded

A Compute Node That Provides Specific Services by Processing Inputs and Producing Required Responses

- Provides Specific Services Rather than General Purpose Computing
- Often No Direct Connection to User Input/Output
- Contained within a Larger System as a Sub-system

Operational Environment

Sensors

Embedded System

Actuators
Many Real-Time Embedded Systems

Real Time – Must Respond to Requests for Service by a Deadline relative to request
- Failure to Respond Prior to Deadline Results in a System Failure
- Request Rate for Service Driven by Real-World Events
- Controls Processes and Delivers Deadline Driven Services

Anti-Lock Braking
Streaming Media (Video and Audio)
Process Control
Aircraft Flight Control
Robotic Systems

© Sam Siewert
Why are RT Embedded Systems a Challenge?

- **Real-Time Services** – Correct Results on time – Deadlines!
  - Multi-service Concurrency Required, for Software, Multi-threaded
    - Multiple interfaces to service in addition to data processing
    - Multi-threaded compared to Main Loop + ISR Executive
    - Supports RT analysis and design (Rate Monotonic)
  - Function/Service Allocation – HW Service Off-load
  - Management of CPU, IO, and Memory Resources

- **CPU Resource**
  - Modern architecture – high throughput, less deterministic
  - pipelines, super-scalar, branch prediction, VM, split-transaction and burst transfer bus interfaces, multi-level caches

- **I/O Interface Resources**
  - Sensors / Actuators (Interaction with Real World)
  - Networks (Latency and QoS)
  - Off-load and Memory Devices (e.g. Flash, FPGAs, DSP)

- **Memory Hierarchy Resources**
  - Register file, L1/L2 cache, SRAM, dynamic RAM, Flash
How to Make RT Embedded Systems Easier!

**RT Service and CPU Resource Management**
- RT Theory, Practice, and Pitfalls (Theory -> System)
  - RMA and DMA Resource Theory
  - Prediction and Measurement of Performance
  - When to Allocate Services/Functions to HW, FW, or SW
- Multi-threaded RTOS Systems
  - Design Methods (DFD, SDL, EFSM and MSC methods)
  - RTOS Mechanisms (e.g. message queue, signal, semaphore)
  - Analysis Tools (e.g. Windview)
  - HW/FW Debug Tools (e.g. ICE / JTAG)

**I/O Device Interfaces and Drivers**
- Abstracted SW-HW Interfaces
- Interaction with Memory System (MMIO, DMAs, Plug-n-Play)

**Memory Hierarchy Analysis and Abstraction**
- Multi-level Cache Performance Models
- Abstracted Non-Volatile Memory Filesystems
Final Project (Assignment #5 & #6)

- Groups of 2-4 Students (Individual by Approval)
  - Must demonstrate **2+ Deadlines in Requirements**
  - Can use FreeRTOS or Linux with POSIX RT Extensions

- RT Emphasis Projects
  - Computer Vision Projects - Peak-Up, Optical-Nav, Video Compression / Transmission
  - Robotics (Proof-of-Concepts) – E.g. Tilt/Pan Servo Tracker

- Project of Your Own Design
Example Projects – Mobile Platforms

**Camera Line Followers**
- Image Processing
- Steering, Motor Control
- Collision Avoidance
- Tethered or Wireless
- Speed and Curvature

**Rovers**
- GPS (with compass)
- GPS + Optical Navigation

© Sam Siewert
Example Projects – Tilt/Pan Video Tracking

- **Camera Peak-Up**
  - Camera tilts and pans to track object

- **Target Tracker**
  - fixed camera, laser pointer tilts and pans to track

- **Stereo Vision Tracker**

- **Tracking Speed**

- **Intensive Image Processing**

- **Scanners**
  - Etch n Sketch
    - Scans Image
    - Redraws with Servos
Embedded Linux Overview

Introduction Session
Integrating Linux into RT System

- RTECS v2 – Chapter 11
- User Space POSIX Threads and RT API
- Kernel Modules
- Linux Kernel Patches to Improve Pre-emptibility
- Risky for HRT (Hard RT), Option for SRT (Soft RT)

Hybrid Solutions – FPGA + SoC

© Sam Siewert
Embedded Linux

- Jetson Systems
- Must Use **root** privilege POSIX RT Threads
- Possible to Get Predictable Response
- Never as Deterministic as FPGA State-Machine or RTOS

© Sam Siewert

https://developer.nvidia.com/jetson-tk1
http://coloradoengineering.com/tk1-som-nvidia-tegra-k1-ultimate-iot-m2m-platform/
POSIX RT Task Example

- $S_1$ – Computes Fibonacci for 10 milliseconds and quits
- $S_2$ – Computes Fibonacci for 20 milliseconds and quits
- $C_1=10$, $C_2=20$ milliseconds
- $T_1=D_1=20$ & $T_2=D_2=50$ milliseconds

https://www.mathsisfun.com/numbers/fibonacci-sequence.html

CPU Loading = ____ %

Feasible Schedule?

Safe? Easy to Emulate?

Violates Rate Monotonic Feasibility Test?

$$U = \sum_{i=1}^{m} \left( \frac{C_i}{T_i} \right) \leq m \left( 2^m - 1 \right)$$
What Does Theory Tell Us?

- Look over LCM of Periods (J Lehoczky, L Sha, Y Ding)
- If Schedule Feasible over LCM, Feasible for All Time

<table>
<thead>
<tr>
<th>Example 5</th>
<th>T1</th>
<th>2</th>
<th>C1</th>
<th>1</th>
<th>U1</th>
<th>0.5</th>
<th>LCM =</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T2</td>
<td>5</td>
<td>C2</td>
<td>2</td>
<td>U2</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3</td>
<td>10</td>
<td>C3</td>
<td>1</td>
<td>U3</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LCM Schedule**

- **RM Schedule**
  - S1
  - S2
  - S3

- **EDF Schedule**
  - S1
  - S2
  - S3

- **TTD**
  - S1
  - S2
  - S3

- **LLF Schedule**
  - S1
  - S2
  - S3

- **Laxity**
  - S1
  - S2
  - S3

© Sam Siewert
Next Time …

- What Does Theory Tell Us?

- Can We Emulate the Fibonacci Workload Accurately
  - In Linux?
  - In FreeRTOS?
  - With an ISR and Interrupts Asserted by a Programmable Interval Timer?

- What are Pitfalls?

- Why Would I want to Do This?
Complex Multi-Service Systems

- Multiple Software Services (Dynamic Admission – Reconfigurable)
- Synchronization Between Services
- Communication Between Services
- Multiple Sensor Input and Actuator Output Interfaces
- Intermediate IO, Shared Memory, Messaging

Multi-Service Pipelines

RT Management

Event releases / Service control

Control Plane

Data Plane

Remote Display

TCP/IP pkts

SW HW

P0

P1

P2

P3

frameRdy interrupt

frameRdy

RGB32 buffer

grayscale buffer

Bt878 PCI
NTSC decoder
(30 Hz)

Serial A/B

ISA ethernet

tBtvid
(30 Hz)

tFrmDisp
(10 Hz)

tOpnav
(15 Hz)

tTlmLnk
(5 Hz)

tThrustCtl
(15 Hz)

tCamCtl
(3 Hz)

tFrmLnk
(2 Hz)

tNet
(2 Hz)

frame

© Sam Siewert
So Why SW for HRT Systems?

- ASIC and FPGA State-Machine Solutions Offer Hardware Clocked Deterministic Solutions

- FPGAs Can be Updated with New Bit-streams (Synthesized HDL to Reconfigurable Logic Elements)

- Software (Firmware) Remains Simplest for Field Upgrade (Reconfigurable at Run-time)

- FPGAs Can be Costly Per Unit

- Cost of Software Engineering vs. Hardware Engineering