武漢大學
WUHAN UNIVERSITY

Embedded Systems Programming

--Introduction

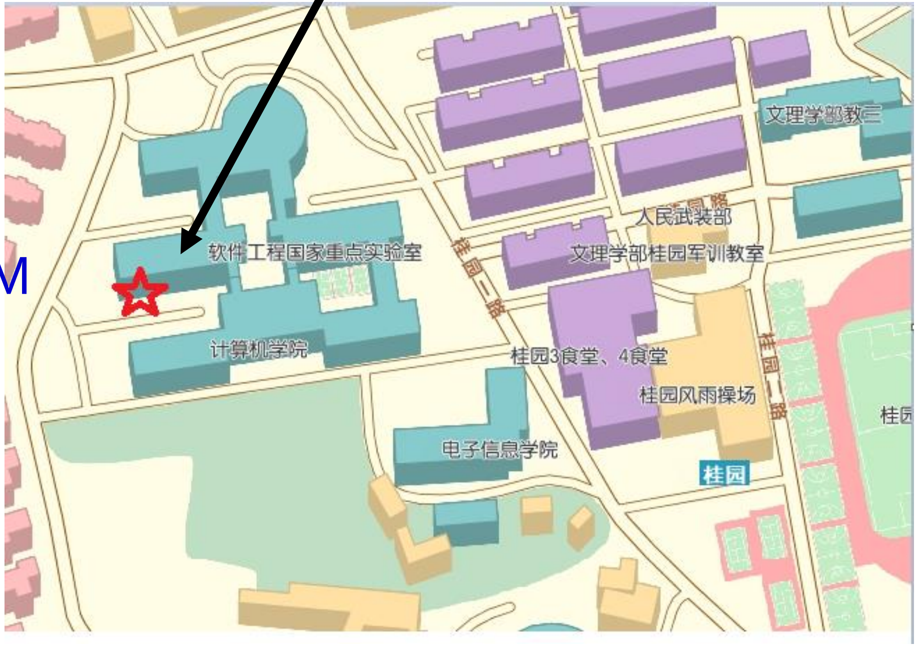
- ❑*Dr. Jianfeng Yang(杨剑锋) , yjf@whu.edu.cn*
- ❑*Dr. Yann-Hang Lee (李雁航) ， yhlee@asu.edu*

Background

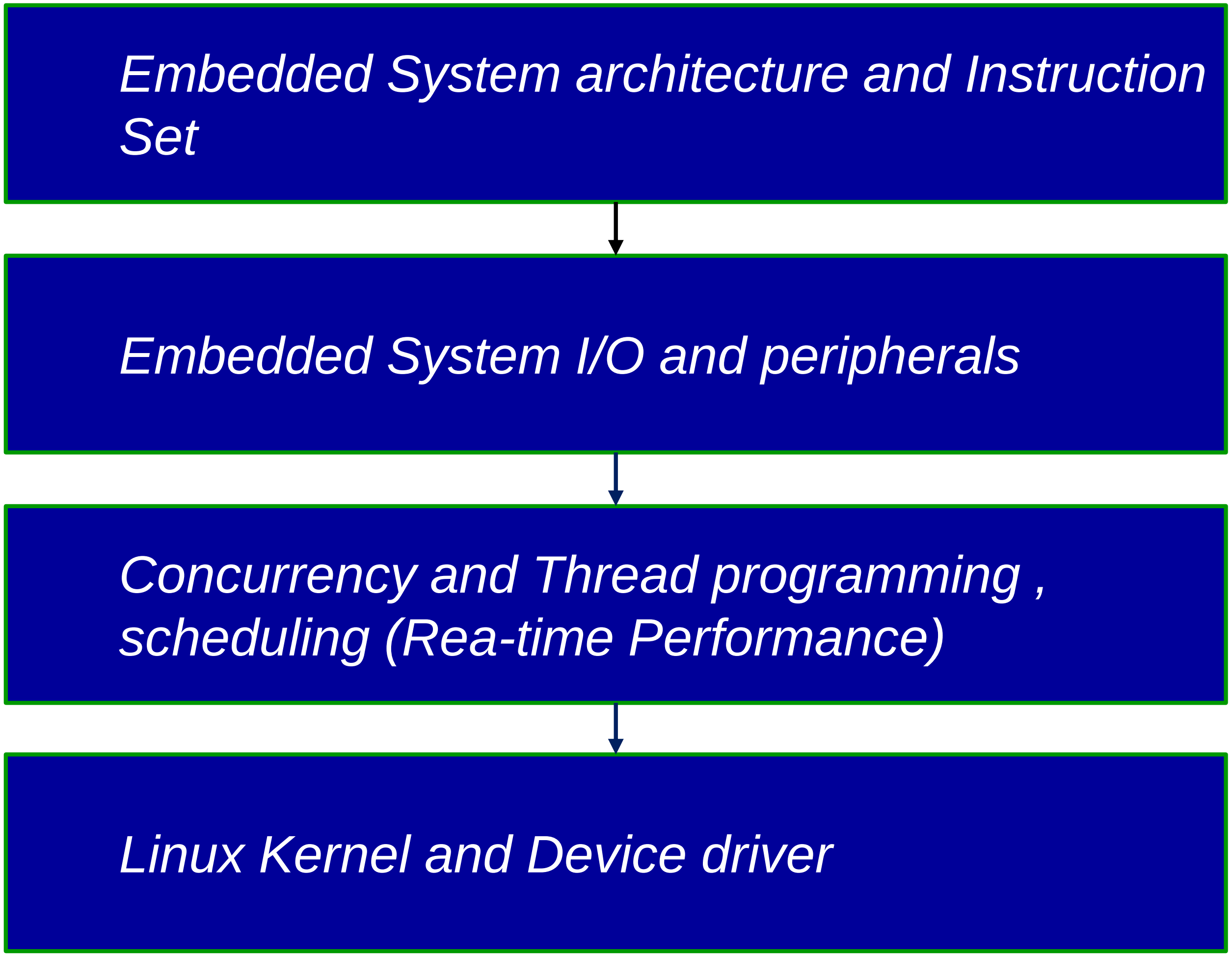
- **Teachers：**
  - Dr. Jianfeng Yang(杨剑锋）, yjf@whu.edu.cn
  - Dr. Yann-Hang Lee (李雁航)， yhlee@asu.edu
  - Dr. Yinbo Xie（谢银波）,xyb@whu.edu.cn
- **Teaching Assistant: Homework and lab assignment**
  - Rui Huang（黄睿）
  - Yun Yu (余韵）
- **Recording Assistant：**
  - Gang Yang(杨刚)
  - Hui Zhao (赵辉)

Logistics

- □ **Address:**
  - ❖ Class room: 1-4-204
  - ❖ Lab projects: EIS experimental center, room #403.
- □ **Class time: 9:00AM - 12：00PM**
- □ **36 hours, 3 credits**
- □ **Lab Projects**
  - ❖ 1st group: 14:00PM - 17:00PM
  - ❖ 2nd group: 18:00PM - 21:00PM
- □ **MOOC**



Embedded System



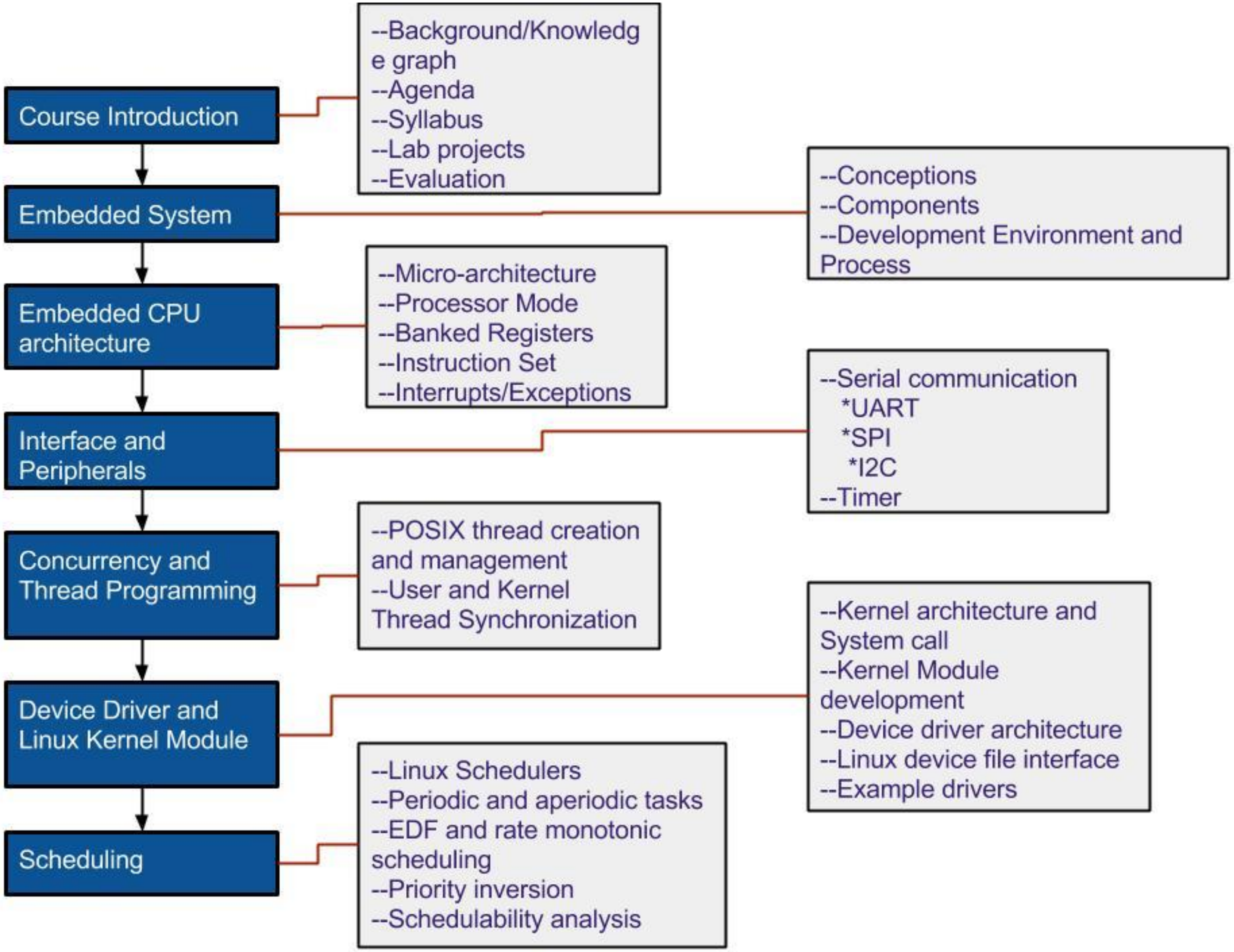
Embedded System architecture and Instruction Set

Embedded System I/O and peripherals

Concurrency and Thread programming , scheduling (Rea-time Performance)

Linux Kernel and Device driver

Course description



Course Introduction

- --Background/Knowledge graph
- --Agenda
- --Syllabus
- --Lab projects
- --Evaluation

Embedded System

- --Conceptions
- --Components
- --Development Environment and Process

Embedded CPU architecture

- --Micro-architecture
- --Processor Mode
- --Banked Registers
- --Instruction Set
- --Interrupts/Exceptions

Interface and Peripherals

- --Serial communication
  - *UART
  - *SPI
  - *I2C
- --Timer

Concurrency and Thread Programming

- --POSIX thread creation and management
- --User and Kernel Thread Synchronization

Device Driver and Linux Kernel Module

- --Kernel architecture and System call
- --Kernel Module development
- --Device driver architecture
- --Linux device file interface
- --Example drivers

Scheduling

- --Linux Schedulers
- --Periodic and aperiodic tasks
- --EDF and rate monotonic scheduling
- --Priority inversion
- --Schedulability analysis

Course Schedule

Day	Date	Content	Homework
Monday	2017/7/10	Course introduction	
		Embedded System—Introduction, case study.	
Tuesday	2017/7/11	Embedded CPU architecture	1st
Wednesday	2017/7/12	IO and Peripherals	
Thursday	2017/7/13	Concurrency and Thread Programming	
Friday	2017/7/14	User and kernel level synchronization	2nd
Saturday	2017/7/15	no class	
Sunday	2017/7/16	no class	
Monday	2017/7/17	Kernel structure and Loadable module	3rd
Tuesday	2017/7/18	Device driver - kernel module development, Driver architecture	
Wednesday	2017/7/19	Device driver - Linux device file interface, exemple driver	
Thursday	2017/7/20	Scheduling - task models, Linux Schedulers	4th
Friday	2017/7/21	Scheduling - EDF and rate monotonic analysis	
Saturday	2017/7/22	Scheduling - Priority inversion and schedulability analysis	
Sunday	2017/7/23	Review and exam	

Course Syllabus (Goals)

- **Course Goals:**
  - Understand the Embedded System conception, architecture
  - Understand the design issues of embedded software and gain an in-depth knowledge of development and execution environment.
  - Understand the functions and the internal structure of device interfaces, drivers, and real-time operating systems.
  - Acquire the skill to develop Linux kernel modules and multi-threaded embedded software in target environment.
  - Develop feasible task scheduling and carry out system performance and task schedulability analyses.
- **Pre-requisites:**
  - Assembly language and computer organization, microprocessor interfaces, and experience of C programming language
  - Knowledge of operating systems and computer architecture

Course Syllabus (Contents)

- ❑ Introduction: characteristics of embedded applications
- ❑ Embedded processor architecture: processor architecture, IO interface, exceptions and interrupts, and system memory map.
- ❑ Embedded software and thread programming: task model and specification, periodic and aperiodic tasks.
- ❑ Basic RTOS and services for multiple threads or tasks, mutexes, semaphores and software timers.
- ❑ Device interface and programming approaches: interconnection architecture, serial buses, and device controllers
- ❑ Device driver: software structure of device driver, Linux loadable kernel module, blocking and non-blocking IO, top-half and bottom-half ISR.
- ❑ Scheduling algorithms and analysis: cyclic, rate-monotonic, and EDF, scheduling, priority inheritance, and analysis.

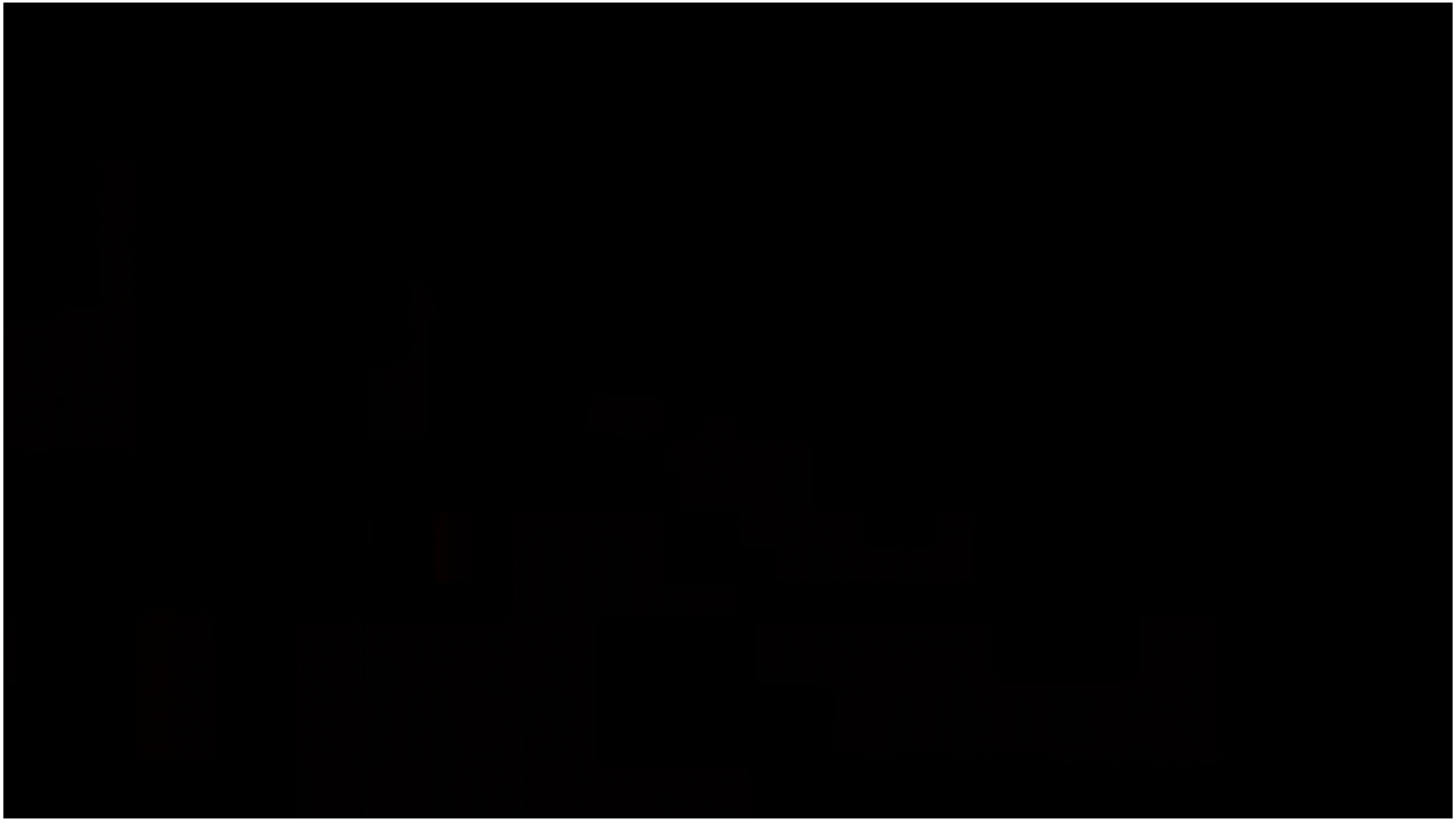
Course Syllabus (Evaluation)

Scores will be accumulated from the following activities.

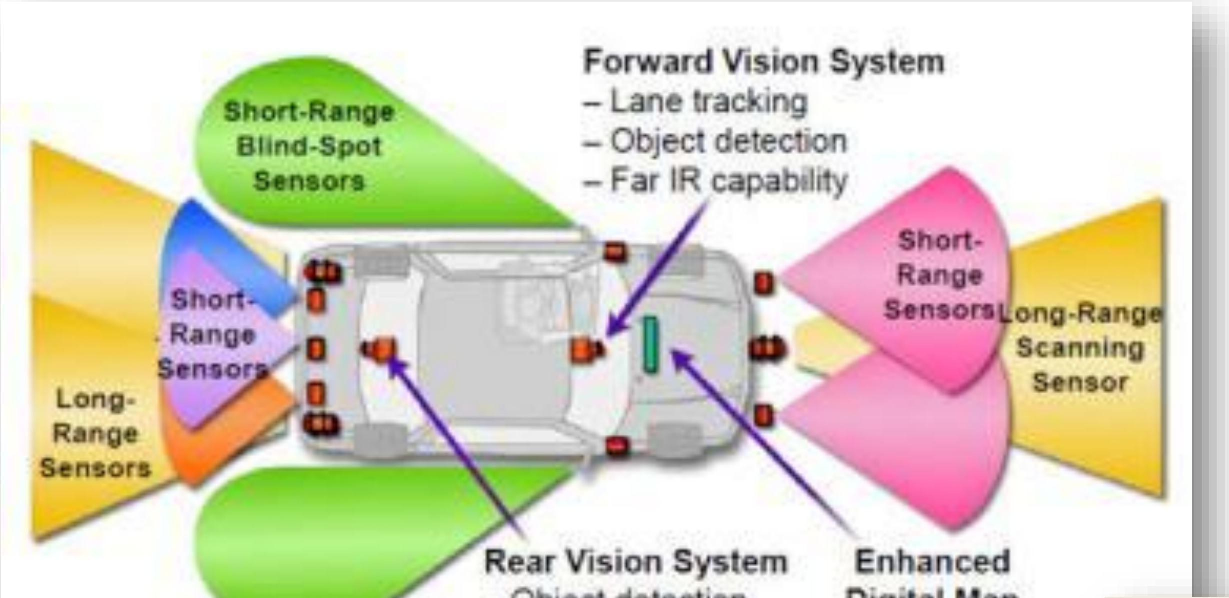
- ❑ **Class attendance and participation: 5%**
- ❑ **Homework:10%**
- ❑ **Lab assignments: 20%**
- ❑ **Exam(closed book and closed notes): 60%**

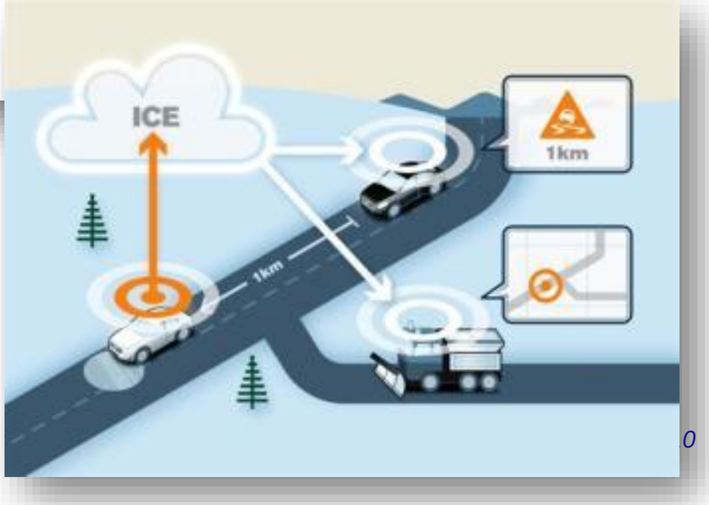
- ❑ **2 students can form a team to carry out lab assignments.**

Embedded System

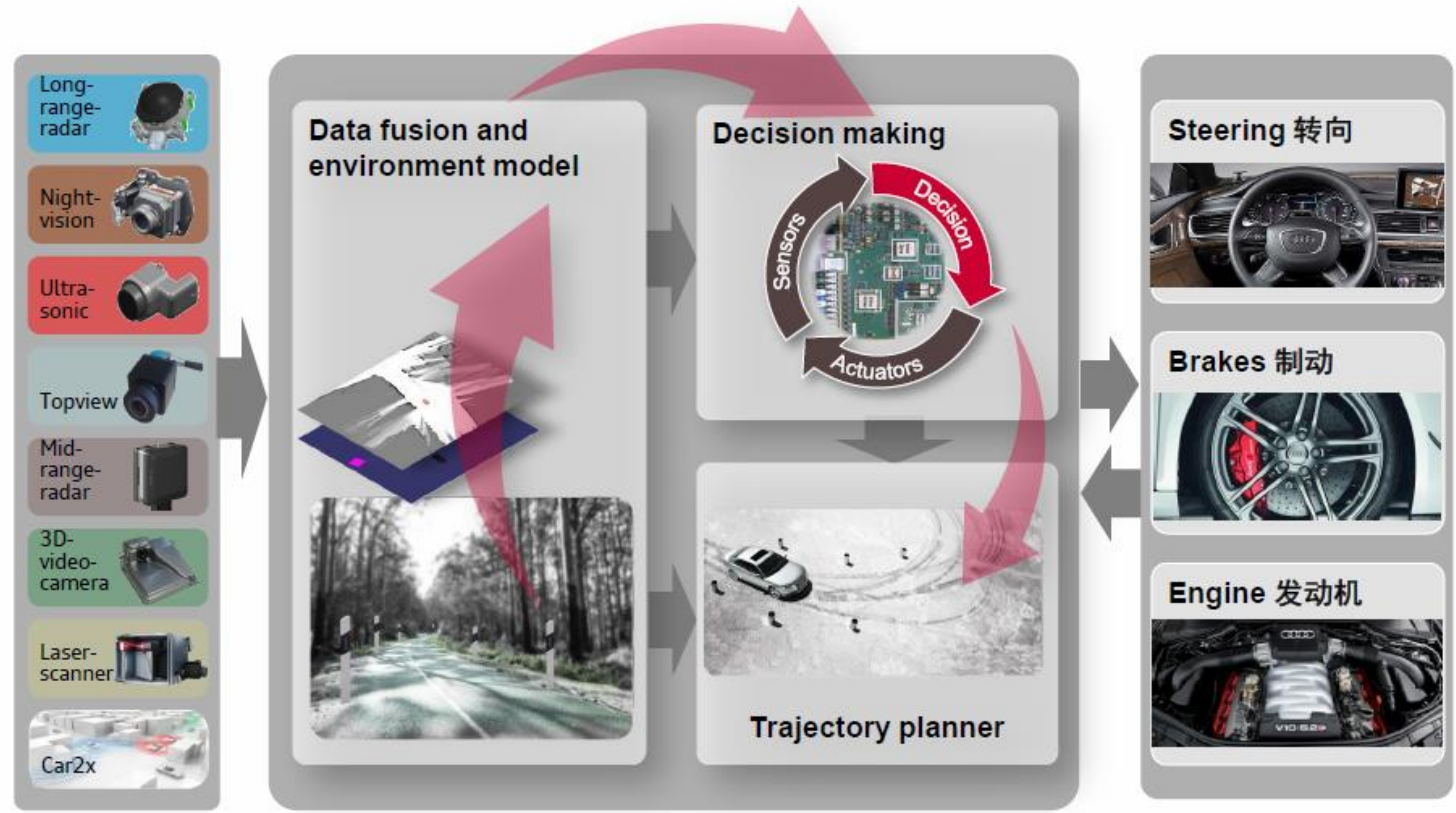


Autonomous Vehicles

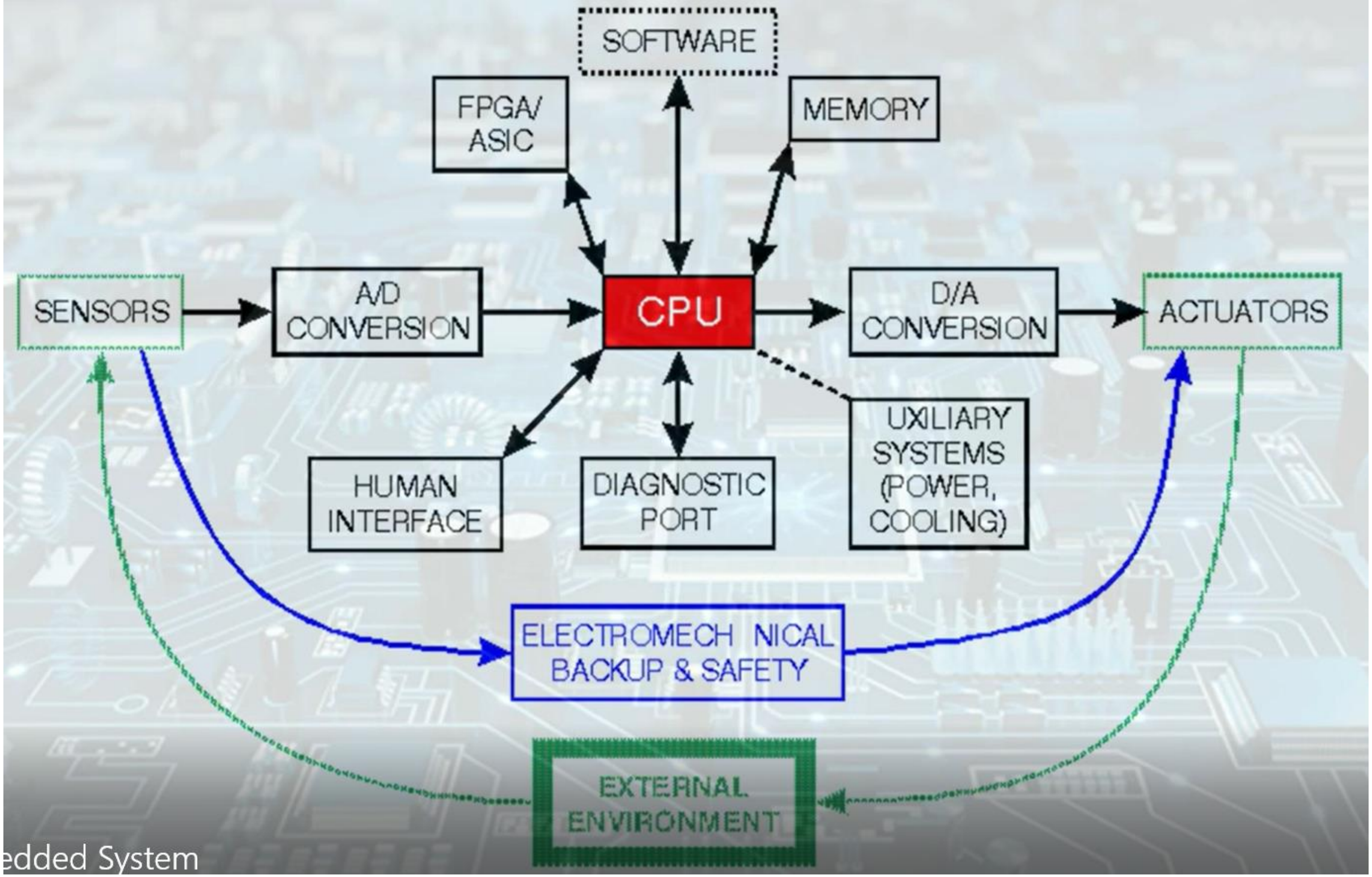




Autonomous Vehicles



Embedded System



What’s Embedded System?

❑ Embedded mean to combine different features into a single object.

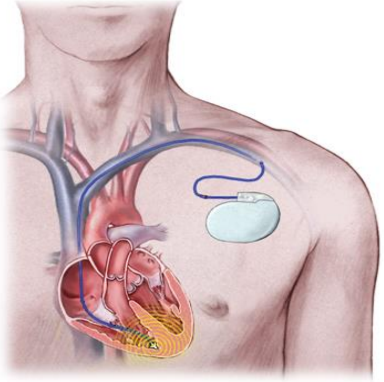
+

❑ System is a way of performing one or many tasks according to a fixed way.

=

❑ Embedded System is a combination of hardware and software performing specific task.

What’s Embedded System?

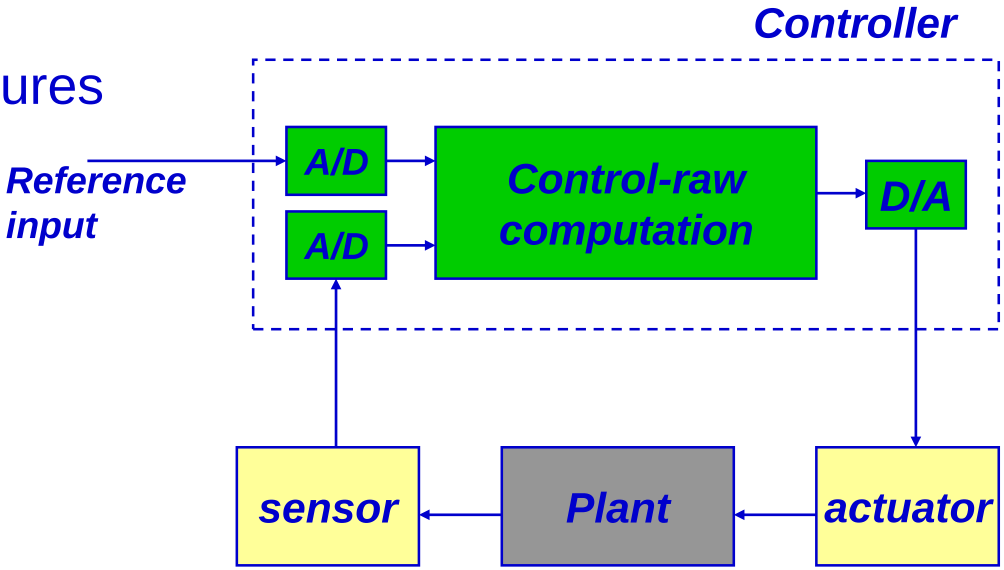


Pacemaker--A pacemaker is a small device that's placed in the chest or abdomen to help control abnormal heart rhythms.

Emerging Embedded Systems

Embedded Systems

- ❑ ***Embedded system***
  - ❖ **the software and hardware component that is an essential part of another system**
- ❑ **Why add a computer to the larger system?**
  - ❖ Better performance
  - ❖ More functions and features
  - ❖ Lower cost
  - ❖ More dependability



Embedded System

❑ Economics

- ❖ Microcontrollers (used for embedded computers) are high-volume, so recurring cost is low
- ❖ Nonrecurring cost dominated by software development

❑ Networks

- ❖ Often embedded system will use multiple processors communicating across a network to lower parts and assembly costs and improve reliability

Real-time Embedded Systems

- ***Real-time system***
  - provide well-timed computation
  - deadlines, jitters, periodicity
  - temporal dependency

Microcontroller vs. Microprocessor

- **Both have a CPU core to execute instructions**
- **Microcontroller has peripherals for concurrent embedded interfacing and control**
  - Analog
  - Non-logic level signals
  - Timing
  - Clock generators
  - Communications
  - Reliability and safety

Attributes of Embedded Systems

❑ Concurrent, reactive behaviors

- ❖ Must respond to sequences and combinations of events
- ❖ Real-time systems have deadlines on responses
- ❖ Typically must perform multiple separate activities concurrently

Constraints

- ❑ **Cost**
  - ❖ Competitive markets penalize products which don’t deliver adequate value for the cost
- ❑ **Size and weight limits**
  - ❖ Mobile (aviation, automotive) and portable (e.g. handheld) systems
- ❑ **Power and energy limits**
  - ❖ Battery capacity
  - ❖ Cooling limits
- ❑ **Environment**
  - ❖ Temperatures may range from -40°C to 125°C, or even more

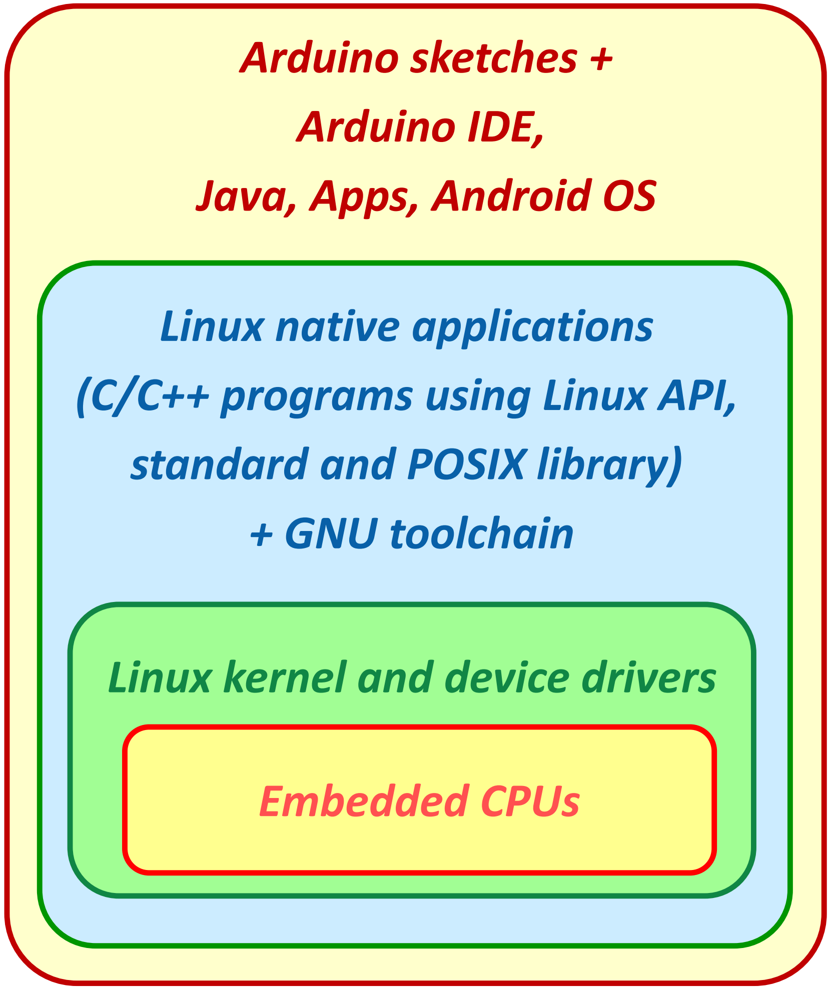
Impact of Constraints

- Microcontrollers used (rather than microprocessors)
  - Include peripherals to interface with other devices, respond efficiently
  - On-chip RAM, ROM reduce circuit board complexity and cost
- Programming language
  - Programmed in C rather than Java (smaller and faster code, so less expensive MCU)
  - Some performance-critical code may be in assembly language
- Operating system
  - Typically no OS, but instead simple scheduler (or even just interrupts + main code (foreground/background system)
  - If OS is used, likely to be a lean RTOS

Embedded System Programming

- ❑ **Applications programs to control embedded devices**
  - ❖ Programming/development and execution environment
  - ❖ Should the programs be written in C/C++, Java, Arduino sketch, Simulink blocks, Android App, etc.

- ❑ **What would you like to learn about embedded systems?**
  - ❖ applications
  - ❖ software systems
  - ❖ hardware systems



Arduino sketches +
Arduino IDE,
Java, Apps, Android OS

Linux native applications
(C/C++ programs using Linux API,
standard and POSIX library)
+ GNU toolchain

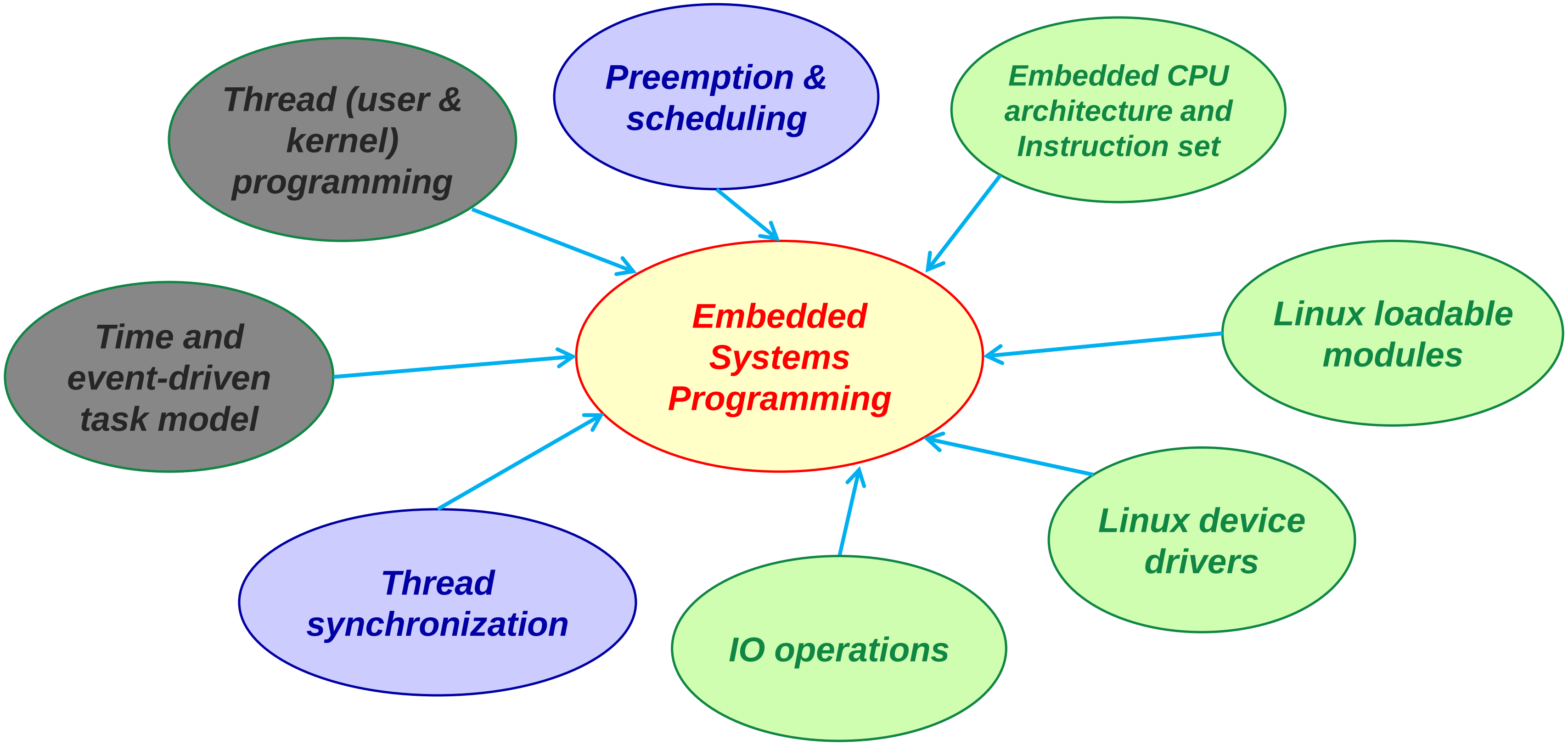
Linux kernel and device drivers

Embedded CPUs

Embedded System Programming Course

WUHAN UNIVERSITY

❑ Knowledge components


Thread (user & kernel) programming
Preemption & scheduling
Embedded CPU architecture and Instruction set
Time and event-driven task model
Embedded Systems Programming
Linux loadable modules
Thread synchronization
IO operations
Linux device drivers

NEXT :
Embedded system architecture