HOGESCHOOL ROTTERDAM

Real-Time Operating Systems

ROS01

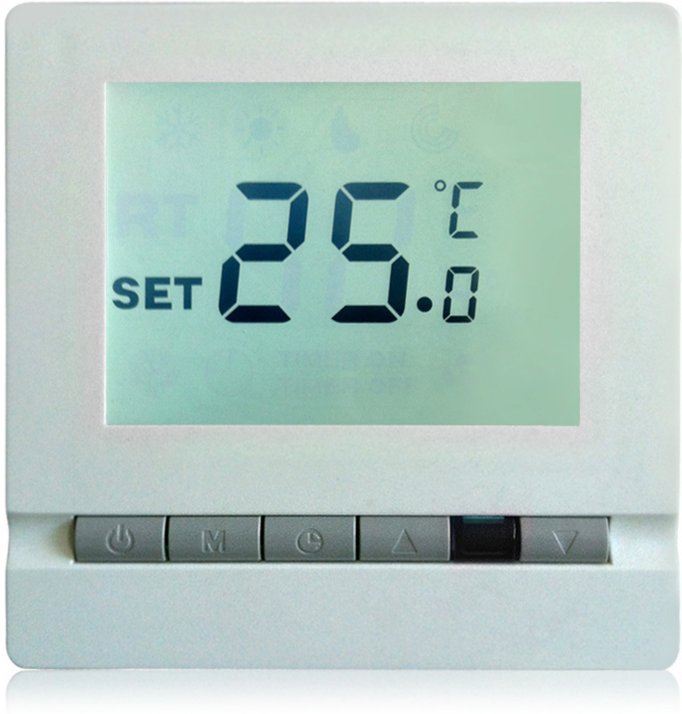
Minor Embedded Systems

Week 1
Introduction

versd@hr.nl
brojz@hr.nl

What is an embedded system?

- Traditional
  - Specific purpose
  - Very limited resources
  - Very limited interface



What is an embedded system?

- Modern
  - Limited set of purposes
  - Bigger but still limited resources
  - Intuitive interface
  - Connected to the IoT

Problem with modern applications

Reading out multiple sensors to drive multiple actuators while performing multiple algorithms while responding within a set period of time

- One CPU / Microcontroller
- Limited I/O
- Limited processing power
- Limited time

Creating suitable software that is independent of the type and amount of sensors and actuators allowing precise control over the use of time

What is a real-time system?

- System for which the response times for unpredictable inputs must be predictable.
- System for which the output must not only be correct but also on the right time.

ROS01

- What will be taught?
  - Working with an advanced microcontroller platform
  - Different types of architectures to deal with the problem
  - Using a real-time operating system to promise:
    - Response times
    - Tested and reliable code
    - Expandable code
  - Schedulability and response time analysis

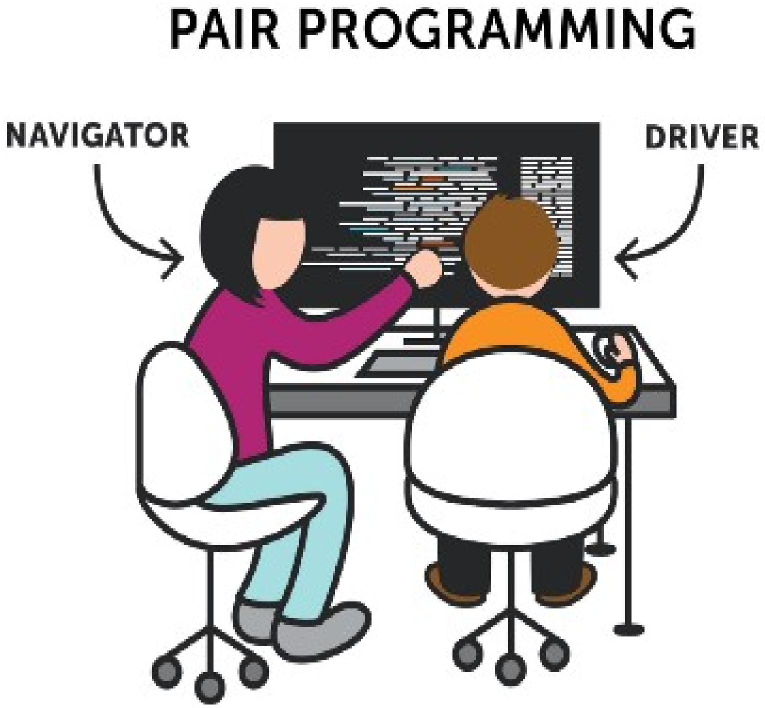
How to pass this course

- 50% of your grade is based on the weekly assignments up to and including week 4
  - Concluded by a report

- The other 50% consists of a two-part final assignment
  - Implementing the assignment using TI-RTOS
  - Calculation the feasability of the requirements

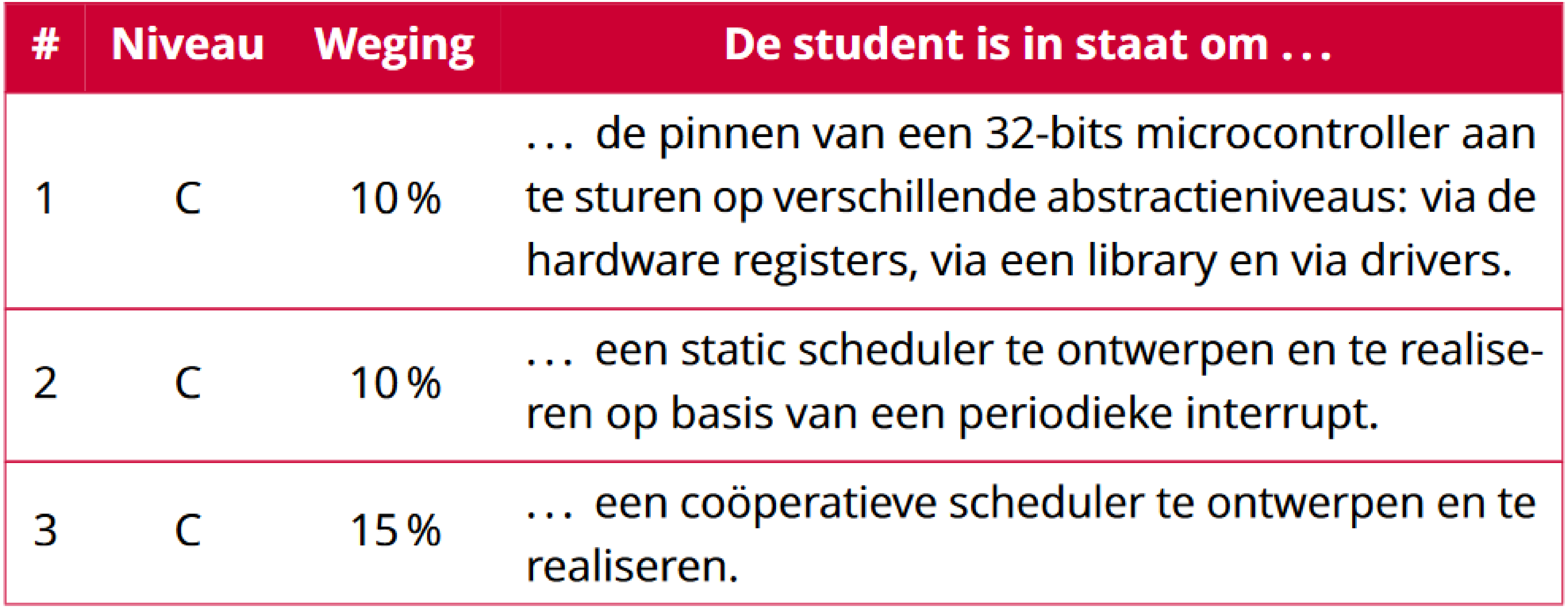
Pair programming

- You work, and are graded, in pairs (pairing is done by the instructors).



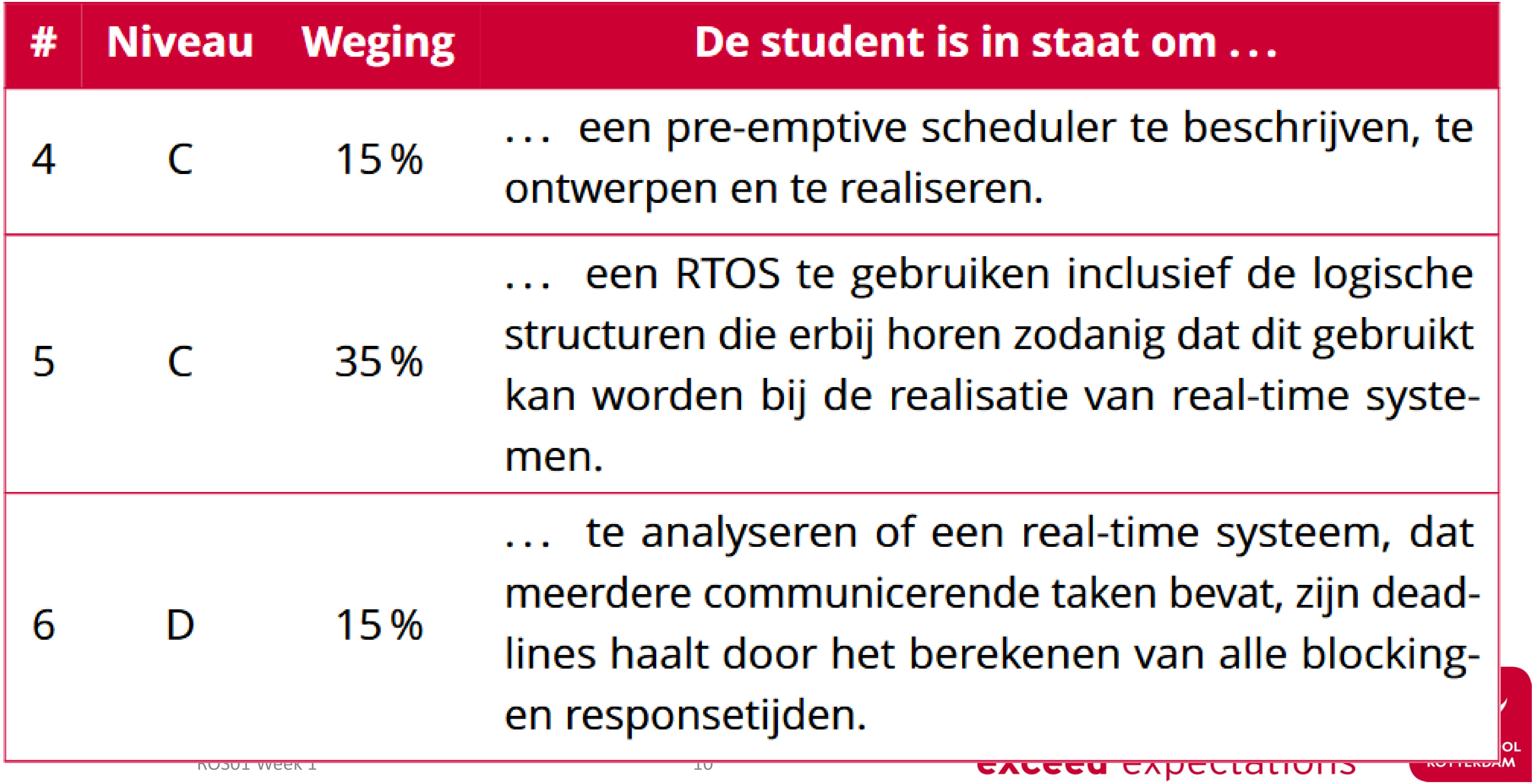
https://medium.com/@tomspencer_uk/pair-programming-and-problem-solving-4531ef3bf171

Leerdoelen (1 van 2)



#	Niveau	Weging	De student is in staat om …
1	C	10 %	… de pinnen van een 32-bits microcontroller aan te sturen op verschillende abstractieniveaus: via de hardware registers, via een library en via drivers.
2	C	10 %	… een static scheduler te ontwerpen en te realiseren op basis van een periodieke interrupt.
3	C	15 %	… een coöperatieve scheduler te ontwerpen en te realiseren.

Leerdoelen (2 van 2)



#	Niveau	Weging	De student is in staat om …
4	C	15 %	… een pre-emptive scheduler te beschrijven, te ontwerpen en te realiseren.
5	C	35 %	… een RTOS te gebruiken inclusief de logische structuren die erbij horen zodanig dat dit gebruikt kan worden bij de realisatie van real-time systemen.
6	D	15 %	… te analyseren of een real-time systeem, dat meerdere communicerende taken bevat, zijn deadlines haalt door het berekenen van alle blocking- en responsetijden.

Planning ROS01

- **Week 1: Introduction – Blinking leds**
- Week 2: Super loop construct with an ISR
- Week 3: Cooperative Scheduling
- Week 4: Pre-emptive Scheduling
- Week 5: TI-RTOS
- Week 6: Schedulability Analyses, Priority Assignment
- Week 7: Response Time Analyses
- Week 8: Finalizing Final Assigment

Learning Goals Week 1

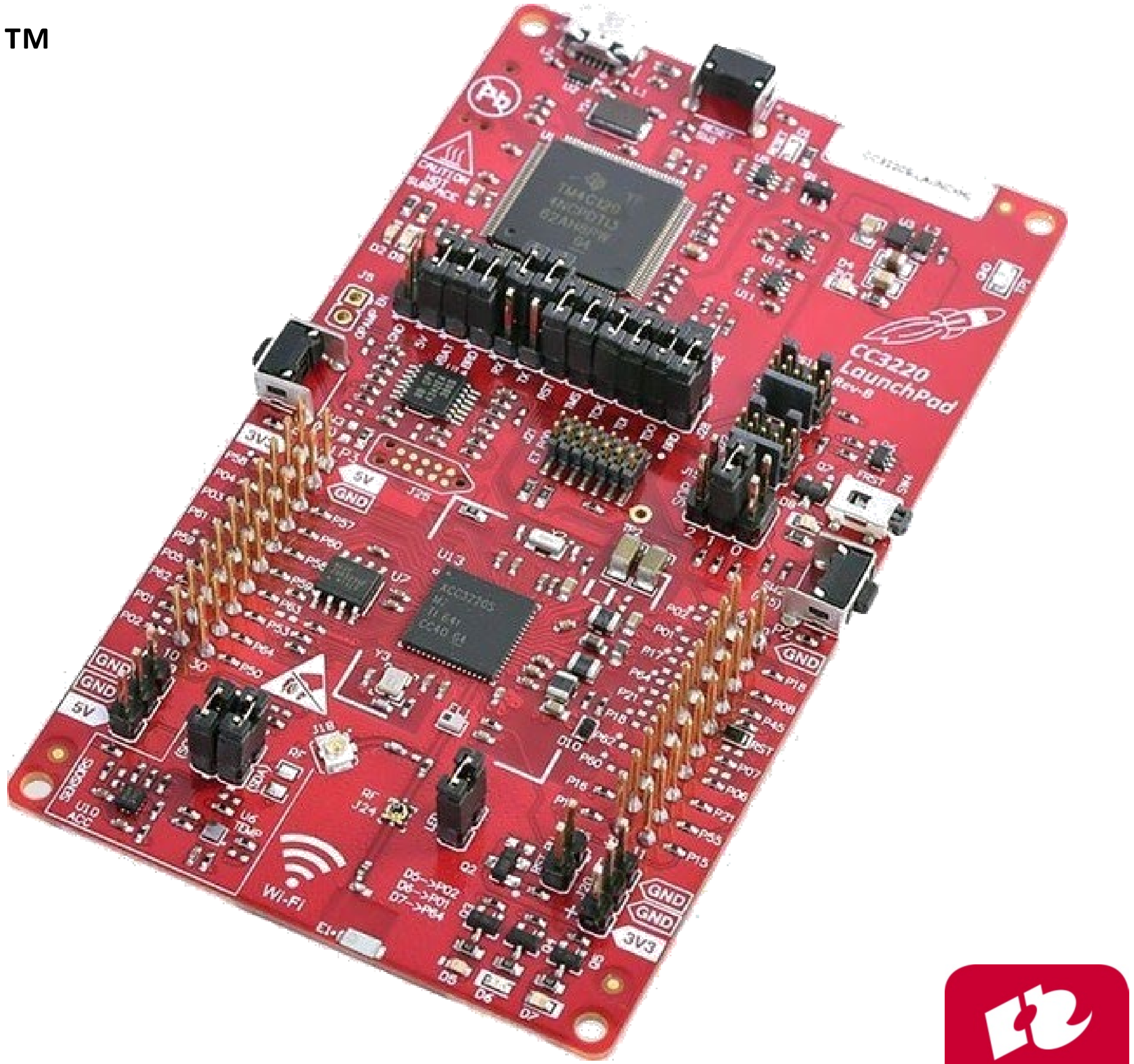
- You will learn:
  - the architecture of the CC3220S;
  - how to write software to control the leds on different abstraction levels.

Development Platform

SimpleLink™ Wi-Fi® CC3220S LaunchPad™

CC3220S SoC:

- 32-bit ARM® Cortex®-M4
- Wifi processor
- Wifi radio

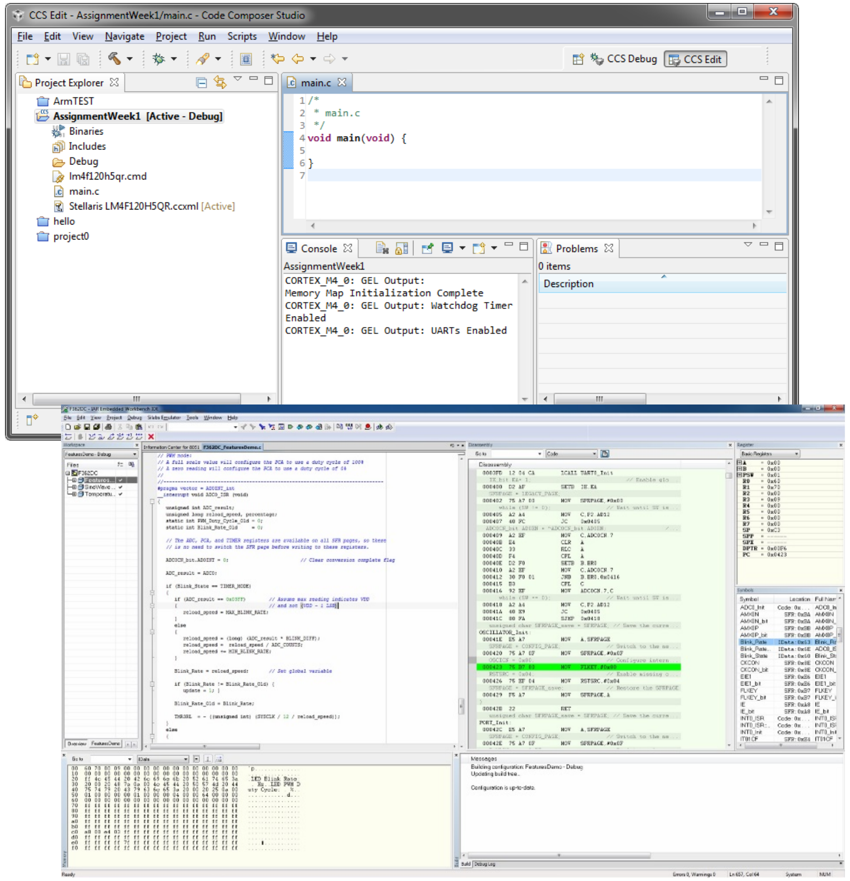


exceed expectations

HOGESCHOOL ROTTERDAM

Software

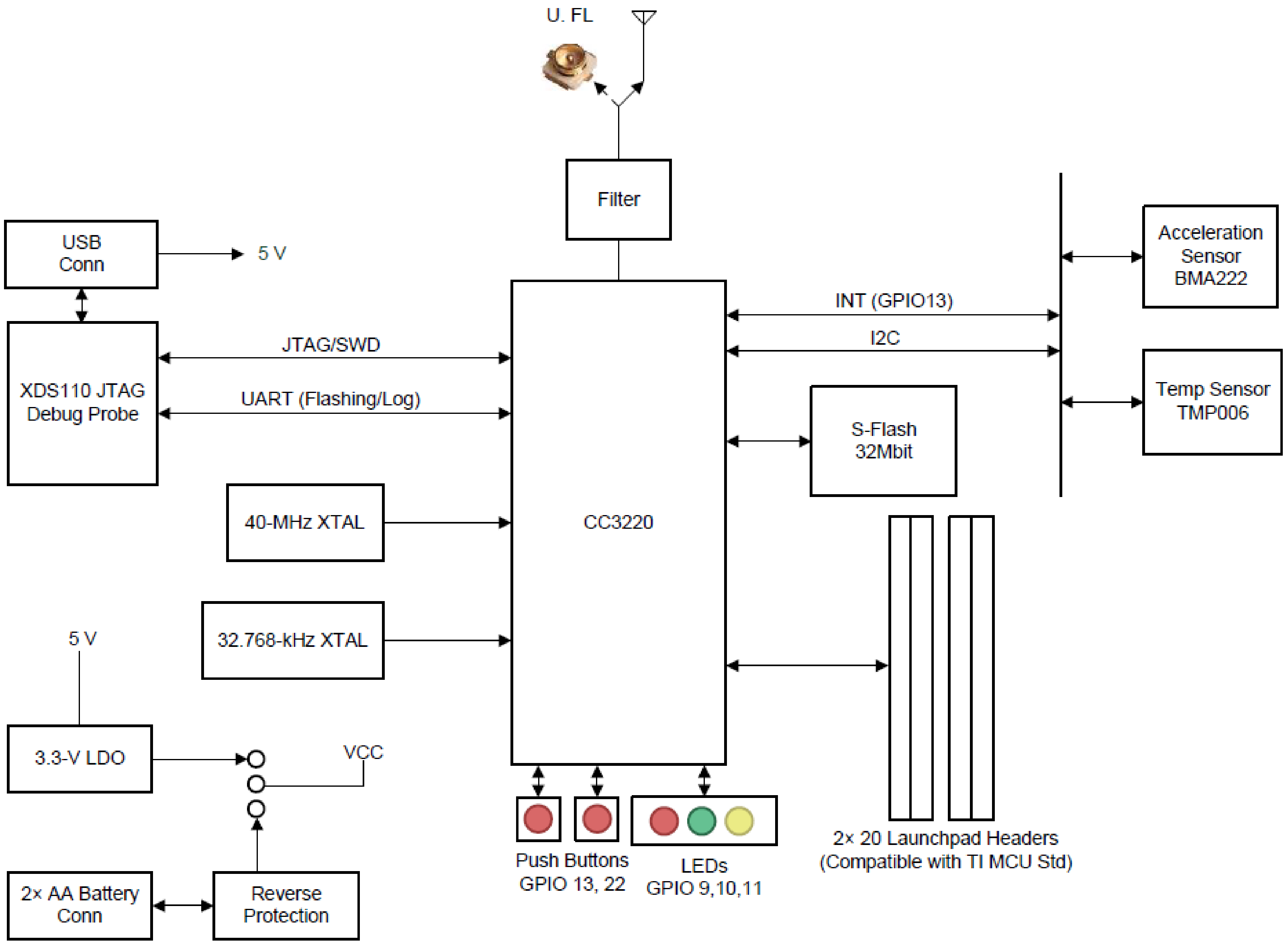
- Development Environment Options
  - **TI Code Composer Studio**
    - Based on Eclipse
    - Optimized compiler
    - Fully functional
  - IAR Embedded Workbench
    - Professional and well supported
    - Free version is size limited
  - DIY (e.g. Eclipse, arm-gcc, gdb ...)
    - Hassle setting up


CCS Edit - AssignmentWeek1/main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
CCS Debug
CCS Edit
Project Explorer
ArmTEST
AssignmentWeek1 [Active - Debug]
Binaries
Includes
Debug
lm4f120h5qr.cmd
main.c
Stellaris LM4F120H5QR.ccxml [Active]
hello
project0
main.c
Console
Problems
0 items
Description

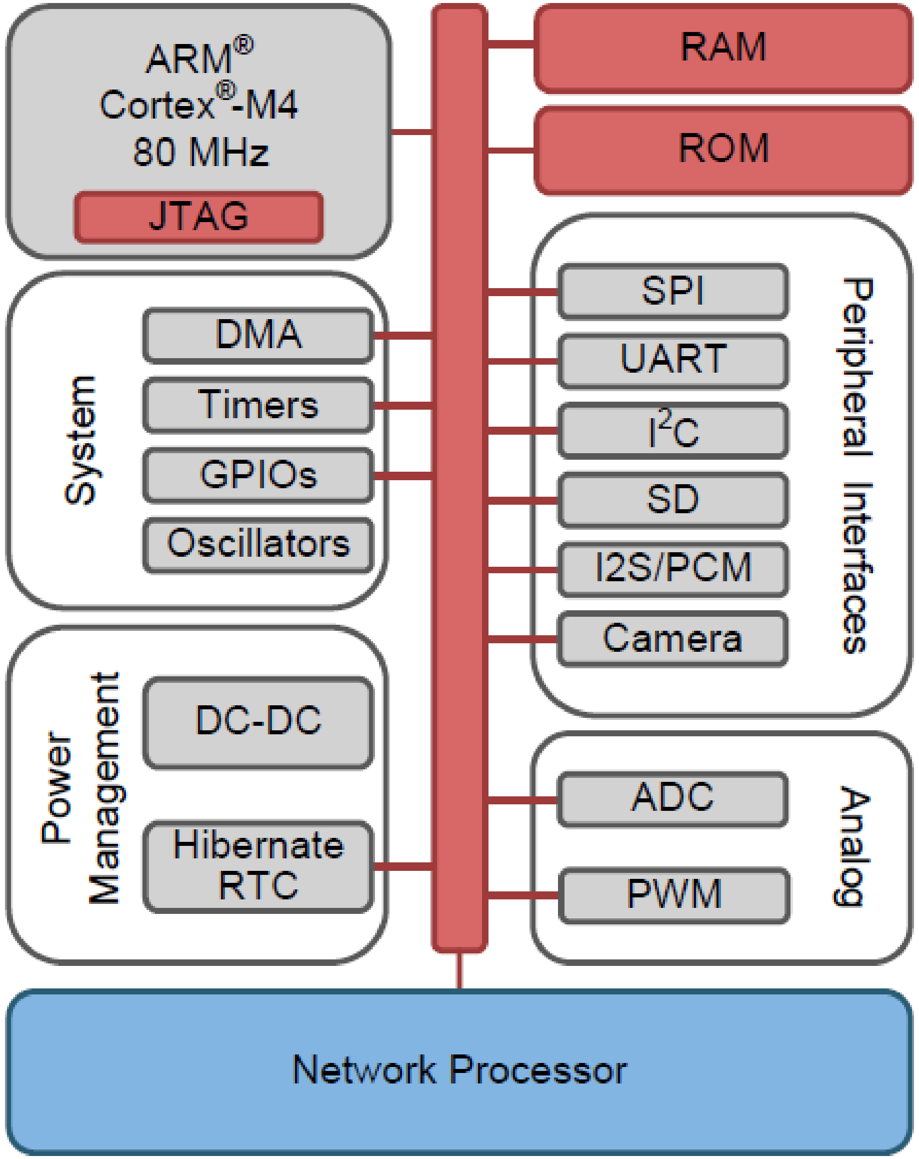
```
/*
 * main.c
 */
void main(void) {

}
```

```
AssignmentWeek1
CORTEX_M4_0: GEL Output:
Memory Map Initialization Complete
CORTEX_M4_0: GEL Output: Watchdog Timer
Enabled
CORTEX_M4_0: GEL Output: UARTs Enabled
```

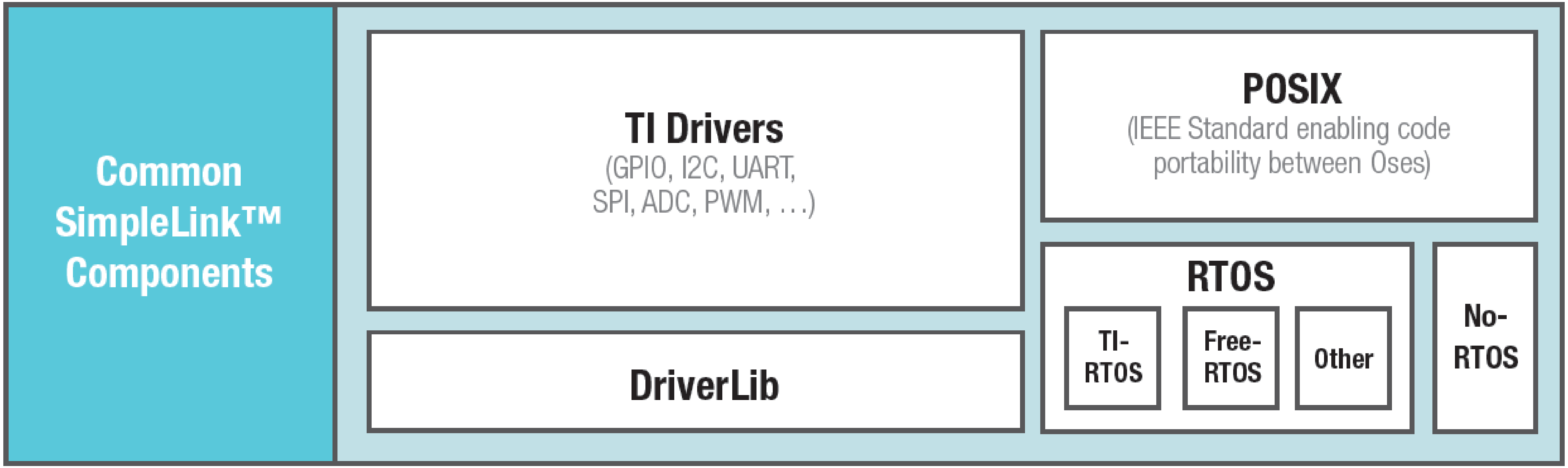
CC3220 SimpleLink™ Wi-Fi® LaunchPad™



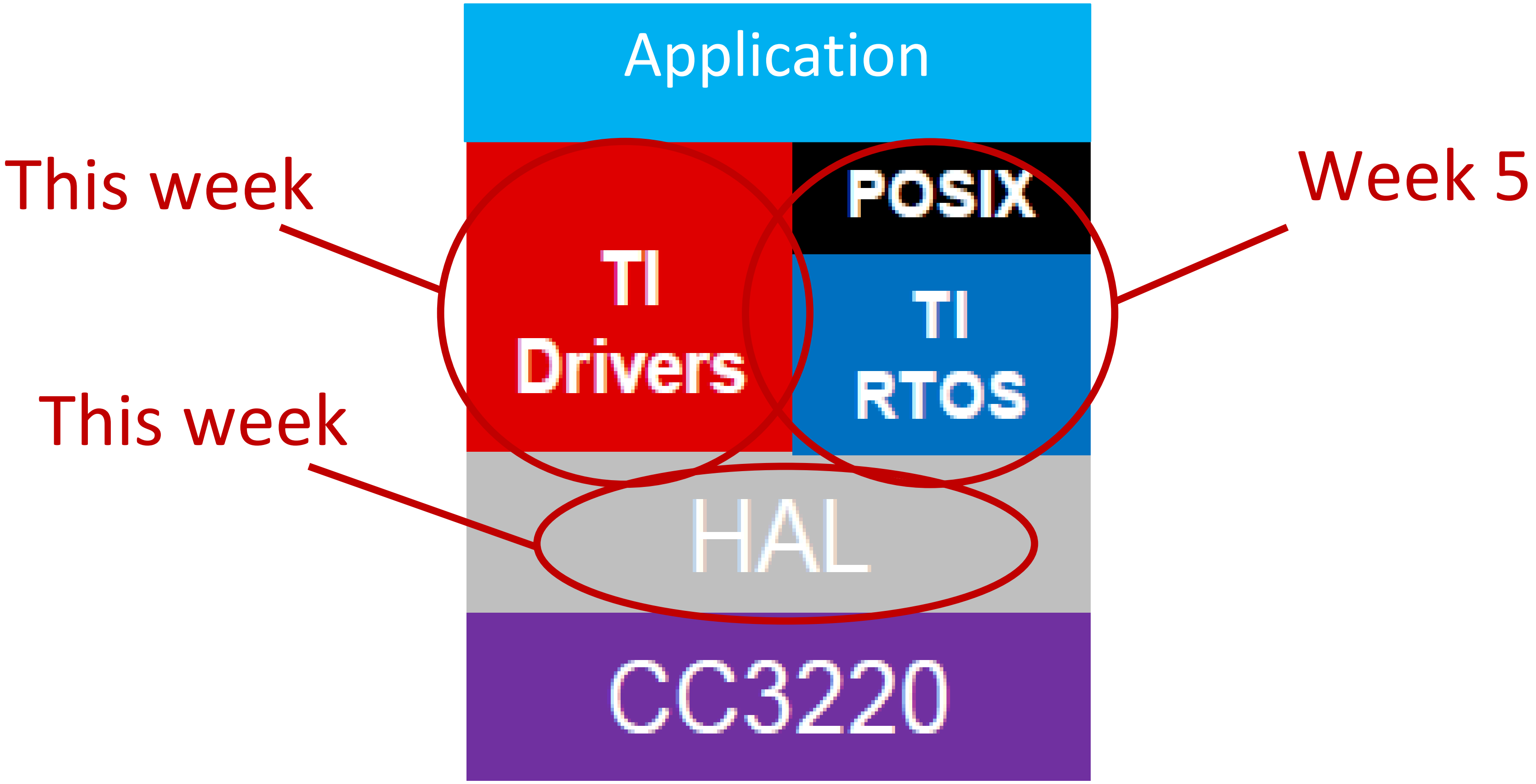
CC3220S


ARM® Cortex®-M4 80 MHz
JTAG
RAM
ROM
System
DMA
Timers
GPIOs
Oscillators
Peripheral Interfaces
SPI
UART
I^2C
SD
I2S/PCM
Camera
Power Management
DC-DC
Hibernate RTC
Analog
ADC
PWM
Network Processor

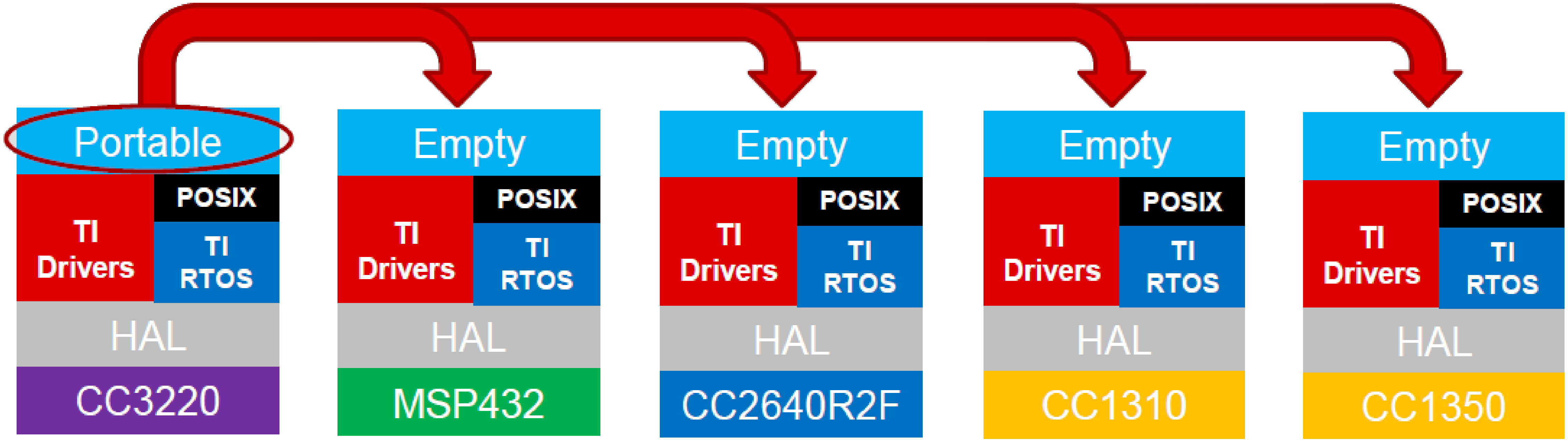
Abstraction layers


Common SimpleLink™ Components
TI Drivers
(GPIO, I2C, UART, SPI, ADC, PWM, …)
DriverLib
POSIX
(IEEE Standard enabling code portability between Oses)
RTOS
TI-RTOS
Free-RTOS
Other
No-RTOS

Abstraction Layers



TI SimpleLink™



wifi

BLE
Bluetooth low energy

Sub-1 GHz RF

Sub-1 GHz.
2.4 GHz

DIY

- Start working on assignment week 1