NTURACING

MCU Programming

W5: System Design

by 羅紀翔

2024-25 FSAE Season

Agenda

NTURACING

Software Development	1	2	C Technique 5 `container_of`
Embedded System Design	3	4	LAB 5 OOP in C

Software Dev

NTURACING

Software Development

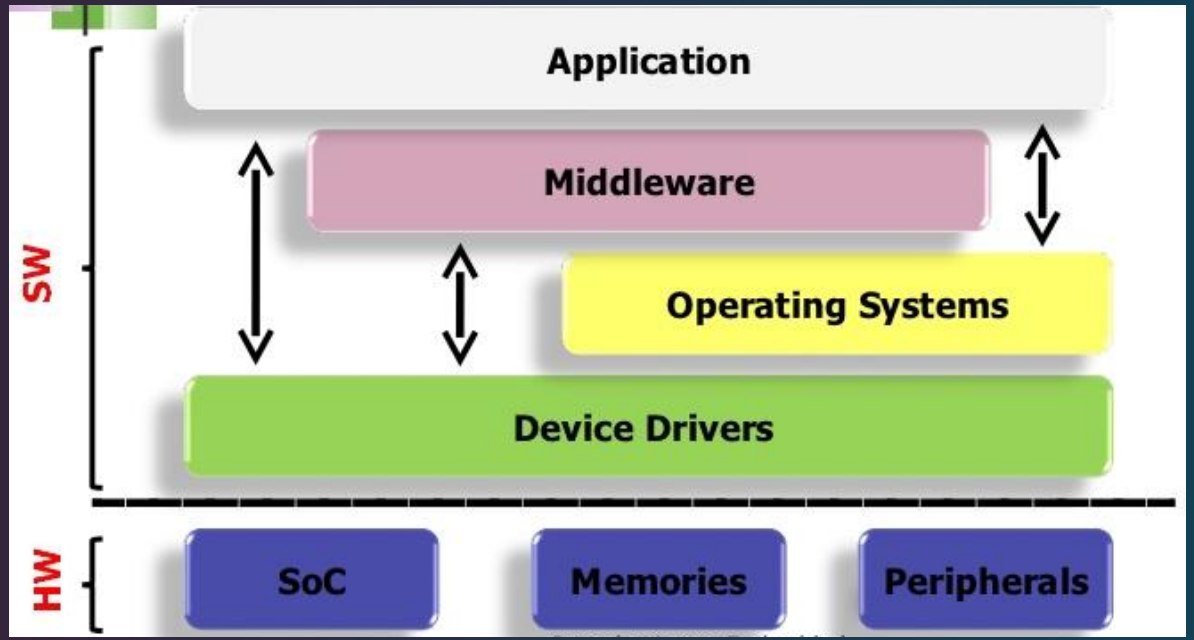
References: *Embedded System Design*

Software Dev

NTURACING

Typical Embedded Software Architecture

- Layered design
- Each layer exposes common interfeces
- Different requirements may have less or additional layers

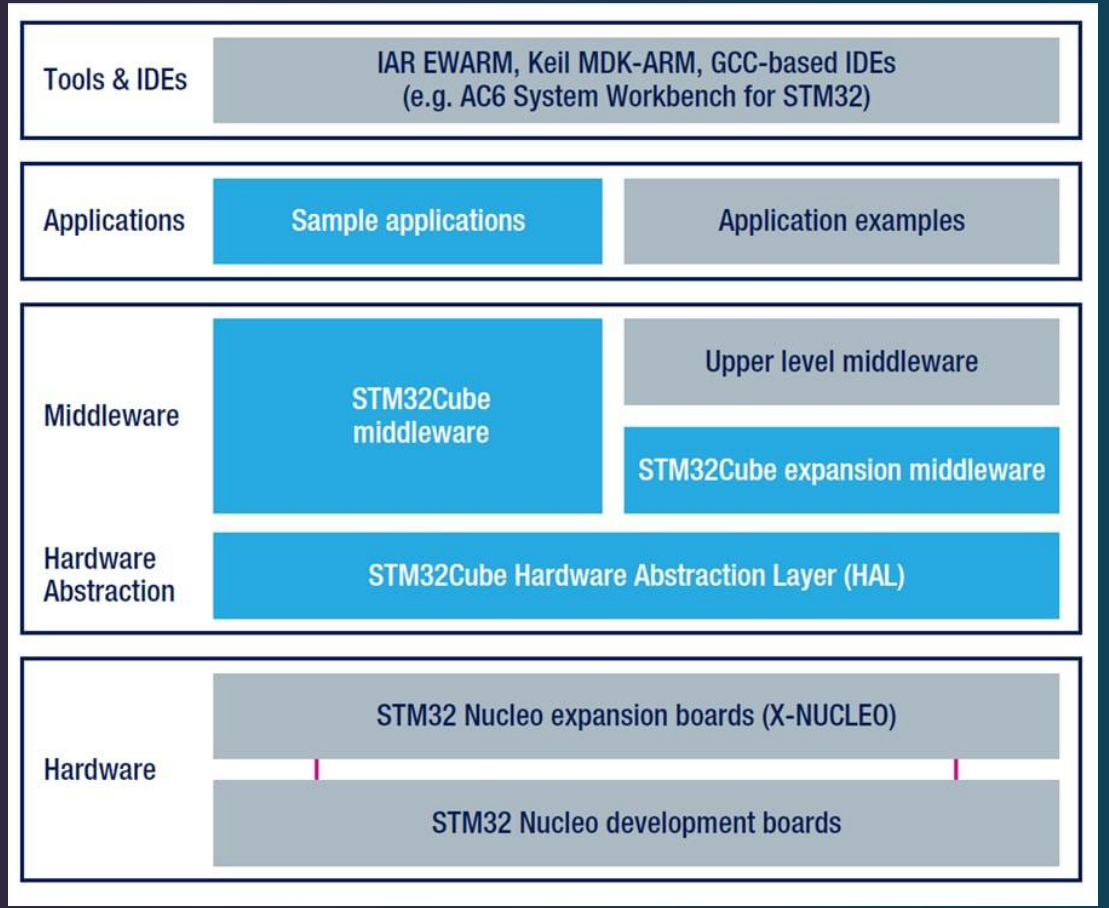


Software Dev

NTURACING

STM32Cube Architecture

- OSes are provided as low-level middleware



Software Dev

NTURACING

Layer: Device Driver

- Controls hardware
  - Microcontroller's peripherals
  - Device attached to microcontroller
- Provided by each hardware vendor
  - Since different vendors have different implementations of similar functionality
  - e.g. ST' HAL and LL library, Espressif's ESP-IDF
  - Or third-party users
- Ex.
  - Two IMUs MPU6050 via I2C and CH100 via CAN
  - Want to use the same set of functions (`imu_init()`, `imu_read()`, etc.) to control them

NTURACING

Layer: Operating System (Kernel)

- Manages the resources of the microcontroller
  - CPU: Task scheduling
    - Multi-tasking, synchronization
  - Memory: Stack, heap
- Typically in MCUs OSes guarantee real-time
  - Predictable runtime behavior
- Different OSes may provide different interfaces
  - POSIX functions
  - OS abstraction layer

Software Dev

NTURACING

Layer: Middleware

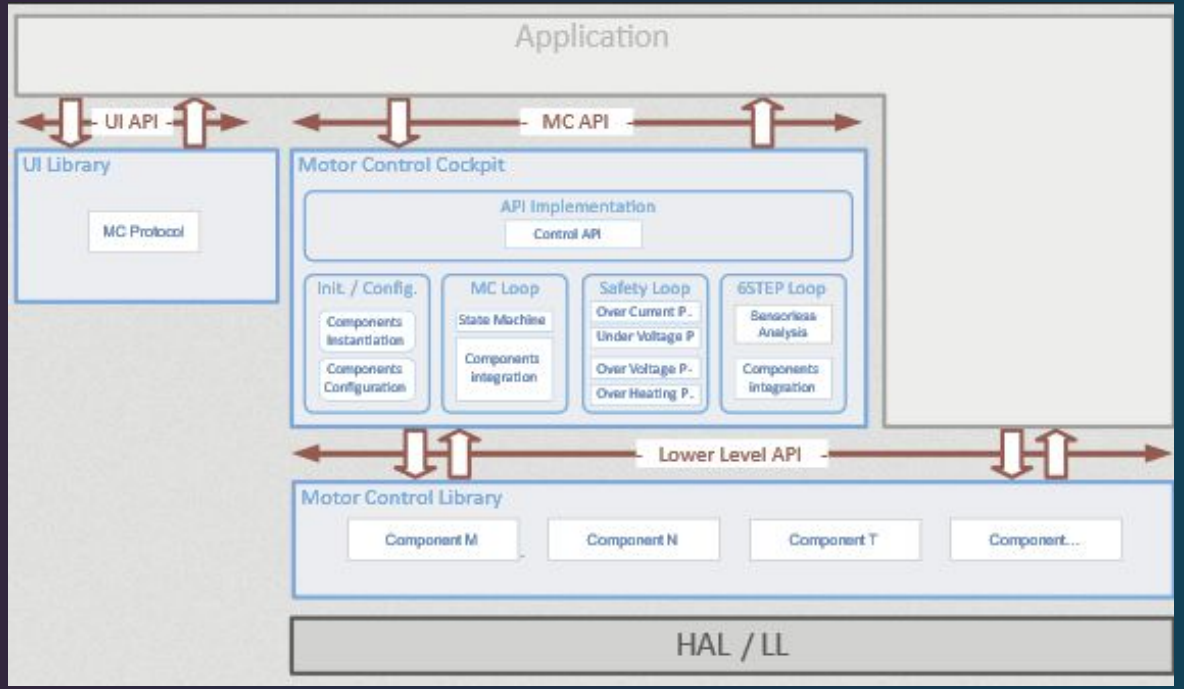
- Library for common tasks
- e.g. file system, SD, USB, internet, etc.
- Some OSes provide them built-in
- STM32CubeMX provides some

Software Dev

NTURACING

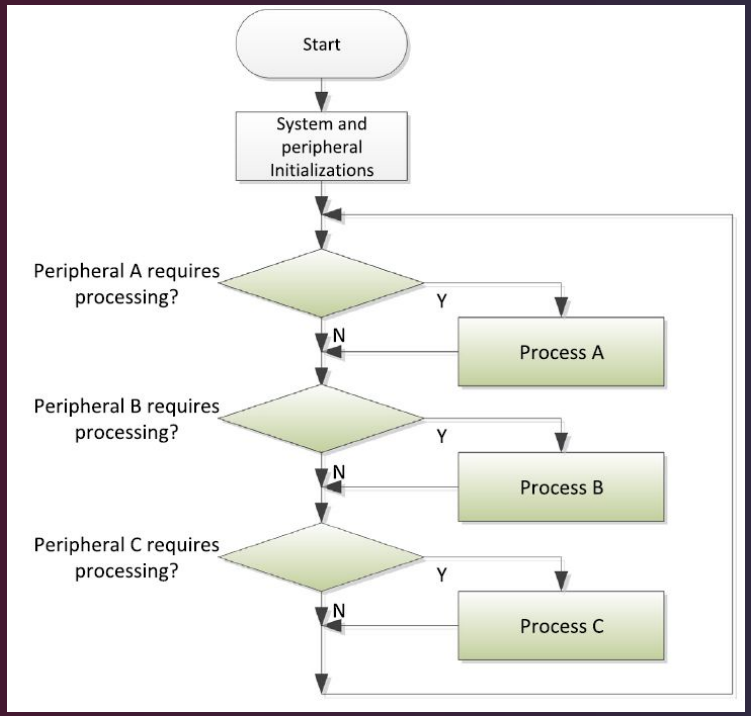
Layer: Application

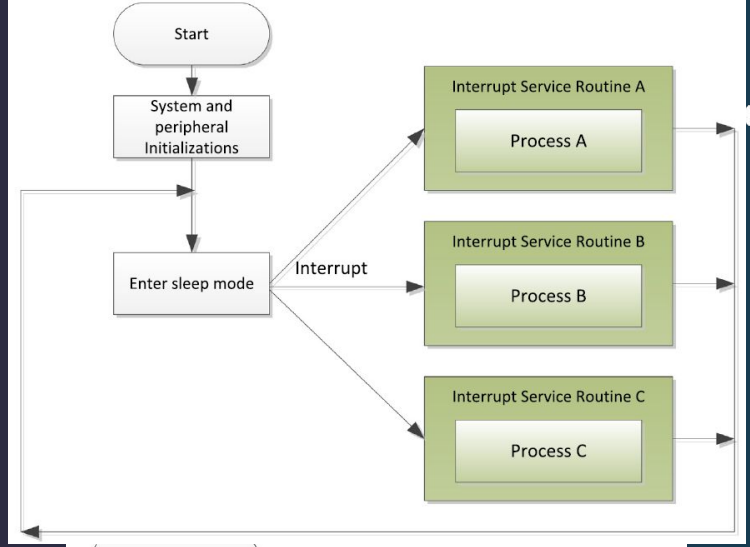
- Uses the interfaces from the lower-layers to implement the requirements
- Itself can also be layered to increase modularity
- Example of ST's motor control workbench
    - The application determines how the motor should run

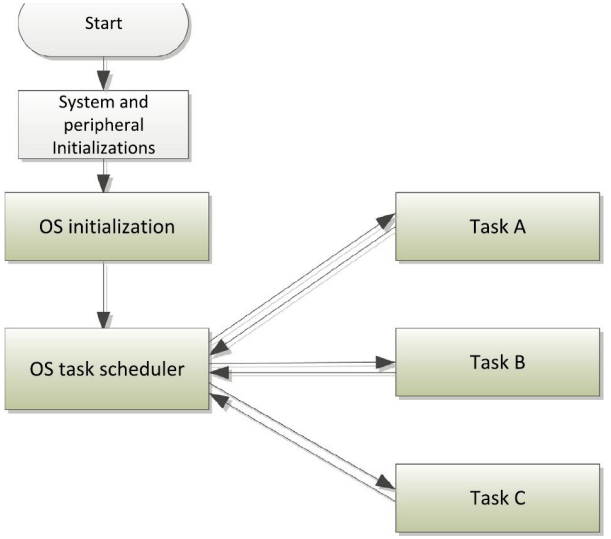


Software Dev

Revisit: Software Flow







NTURACING

C Technique 5: `container_of`

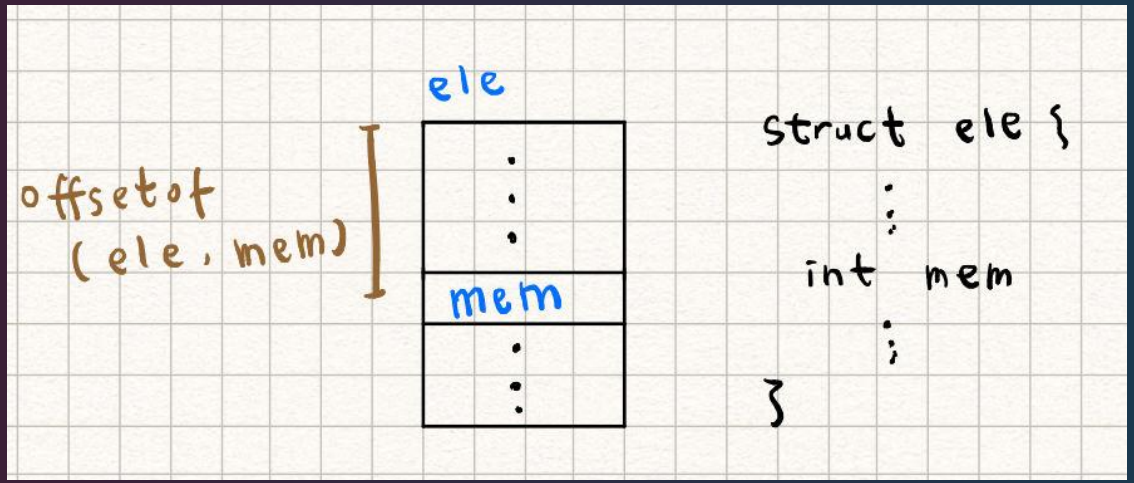
References: HackMD sysprog:
https://hackmd.io/@sysprog/linux-macro-containerof

container_of

NTURACING

`offsetof`

- Macro defined in `stddef.h`
- Evaluates the offset (in bytes) of a given member within a struct or union type with type `size_t`.
- GCC implements using `__builtin_offsetof()`

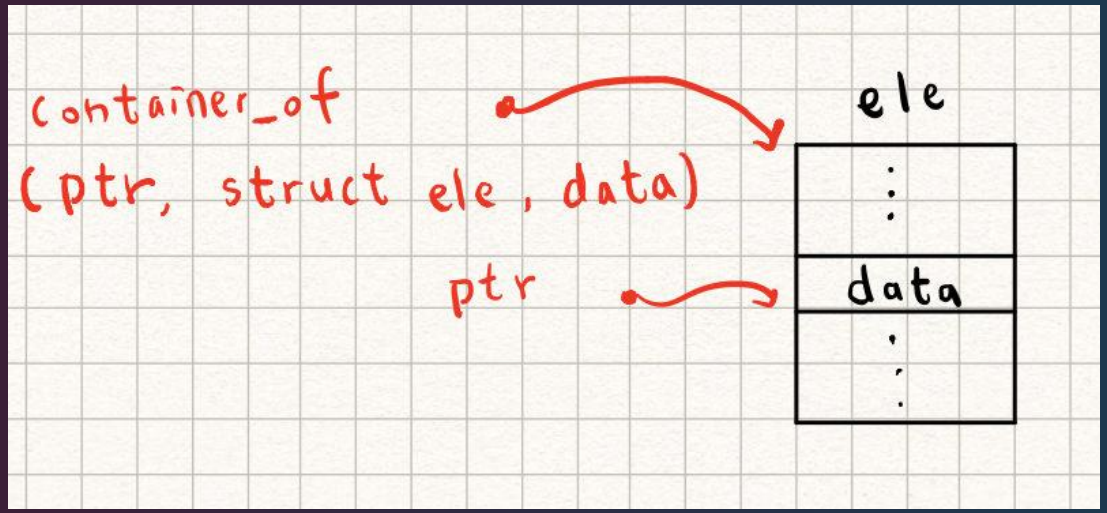


container_of

NTURACING

container_of

```
#define container_of(ptr, type, member) \
  ((type *)(((char *)(ptr)) - offsetof(type, member)))
```



container_of

NTURACING

Applications

- Object-oriented programming (refer to sample code)
- Use in sys/queue.h:
  (https://hackmd.io/@sysprog/linux-macro-containerof#%E6%87%89%E7%94%A8%E6%A1%88%E4%BE%8B-%E9%9B%99%E5%90%91%E7%92%B0%E7%8B%80%E9%8F%88%E7%B5%90%E4%B8%B2%E5%88%97)
- Accessing child struct data from parent struct

NTURACING

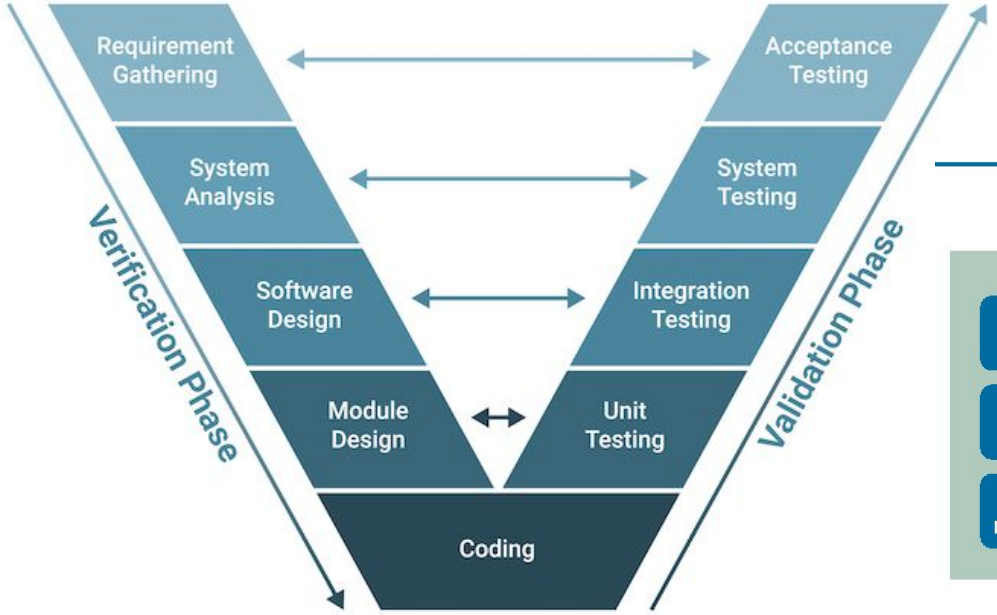
Embedded System Design

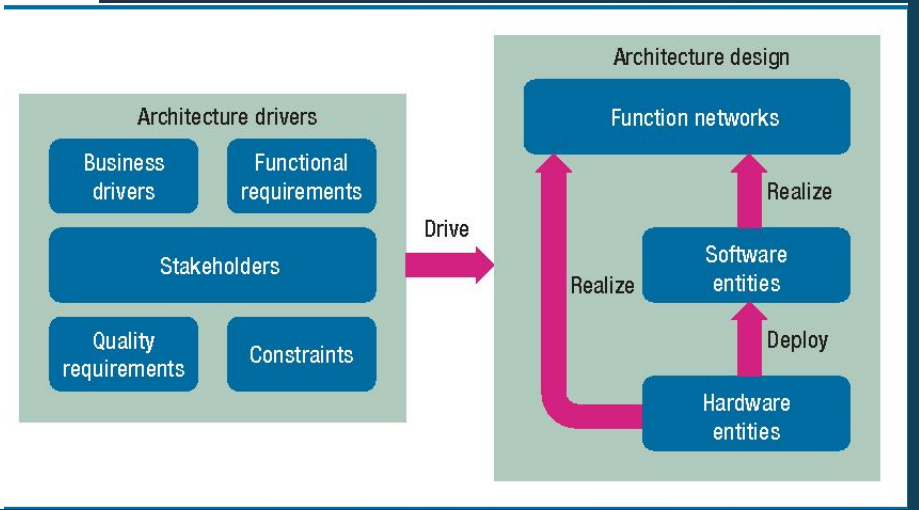
References: *Embedded System Design, ADD Paper*

System Design

NTURACING

Design Process





System Design

NTURACING

Design Requirements

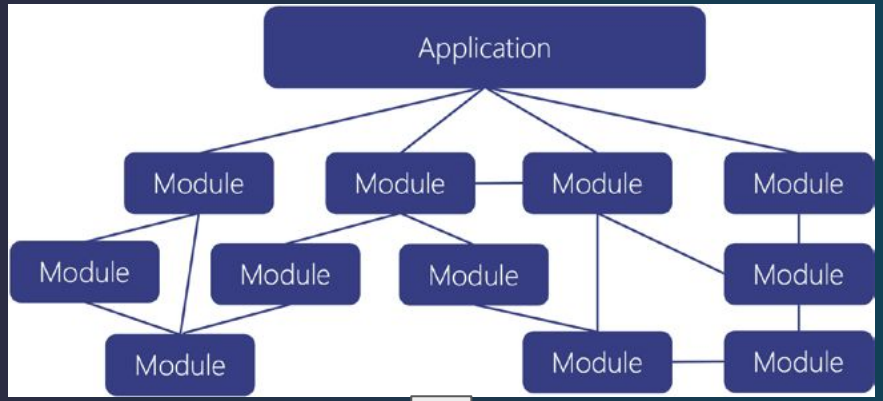
- **Functional requirements** : What functions a system must provide to meet stated and implied stakeholder needs when the software is used under specific conditions.
  - Ex. The system shall allow users to review account activity.
- **Quality attributes** : requirements that indicate the degrees to which a system must exhibit various properties.
  - Ex. The system shall process sensor input within one second. (performance)
- **Design constrains** : Decisions about a system's design that must be incorporated into any final design of the system.
  - Ex. The system shall be implemented using Visual Basic.

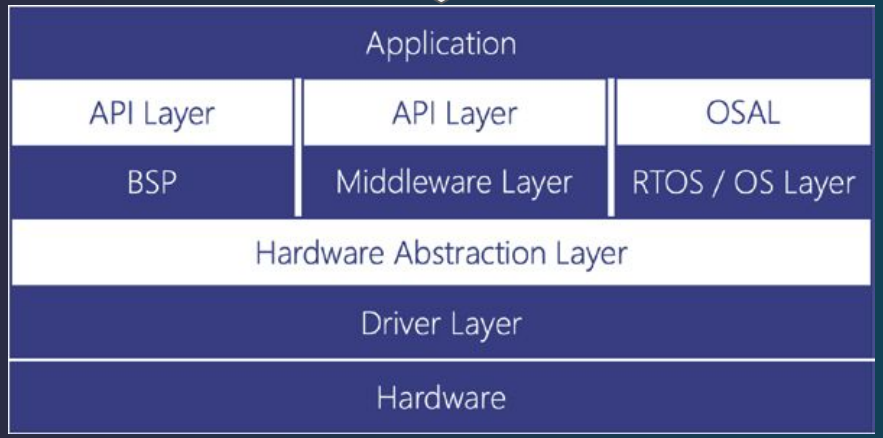
System Design

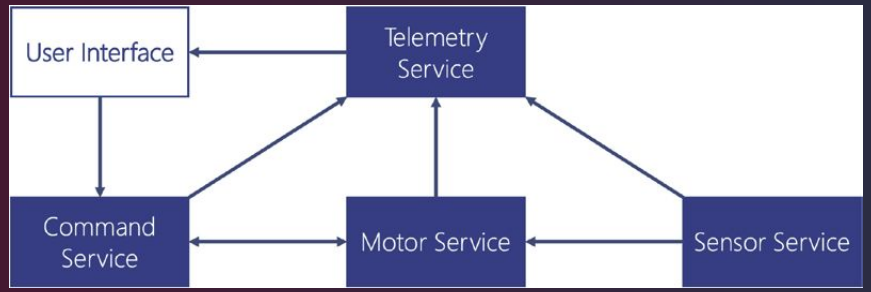
NTURACING

Architecture Design

- Goal
  - Lowly coupled code
  - Vendor independent
  - Flexible architecture
- Layered
- Event-driven
- Message-passing







System Design

NTURACING

Selecting Hardware

- What kind of hardware is required
  - Functionality, interface, dimension
- Microcontrollers (for us, it has become a design constrain)
  - Satisfies the performance, interface requirements
  - Support (software, community, etc.)
- List the requirements for the electrical design (PCB)

System Design

NTURACING

Implementing the Software

- According to the requirements and responsibilities of the module
- Generate a solution for the requirements
  - Choose patterns (commonly used solutions) and tactics (how to make the patterns fit)
  - Document the design tradeoff
  - Implement it
- Verify and refine the requirements
- Defining the interfaces between each modules

System Design

NTURACING

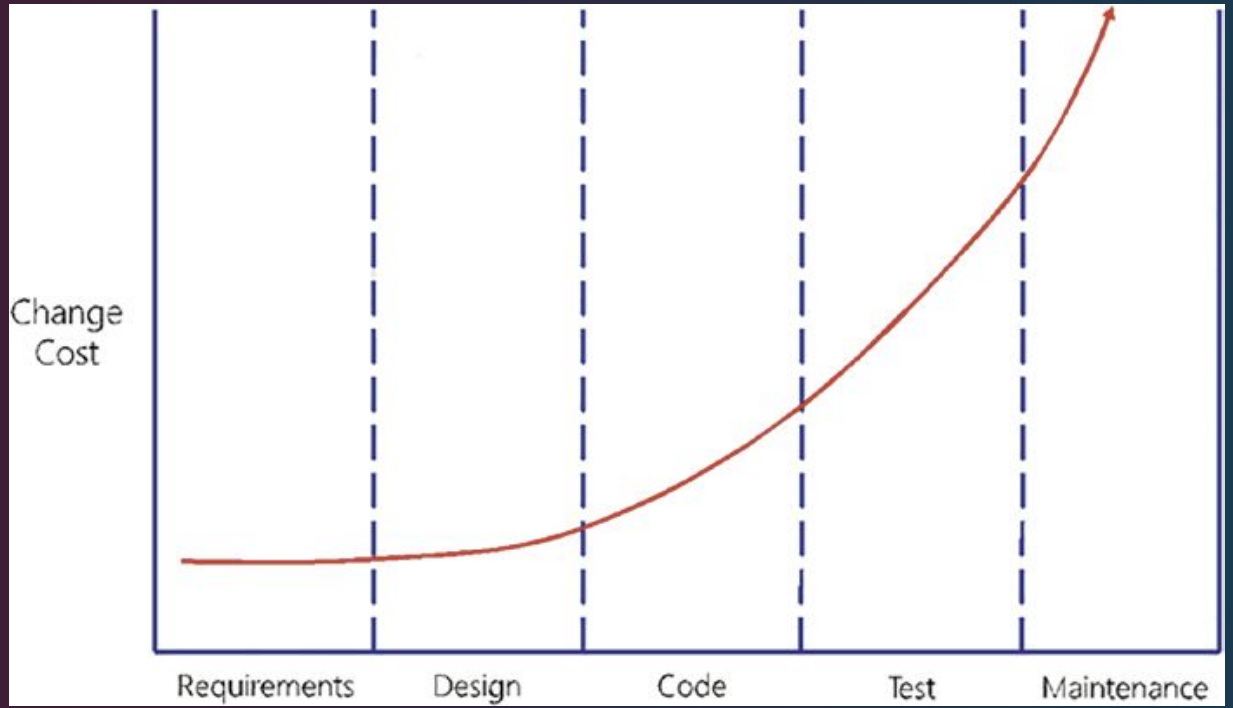
Testing/Validation

- Unit test: To ensure that individual module work as expected.
  - Typically in simulation/emulation. Requires hardware abstraction to make be testable.
- Integrating test: To verify the interaction between different modules and ensure they integrate correctly.
  - Can still be done in simulation/emulation.
- System test: To ensure the complete system meets the specified requirements and works as intended.
- Acceptance test: Determines if the application meets the business requirements and is ready for delivery to the end-users or stakeholders.

System Design

NTURACING

Cost of Changes



NTURACING

Project: RC Car

RC Car

NTURACING

Hardware Overview

- RC car chassis
- Motor, ESC, and steering servo
- Three phase incremental encoder, IMU

Goal

- Wirelessly control the rc car with speed and direction control
- Log the sensor data