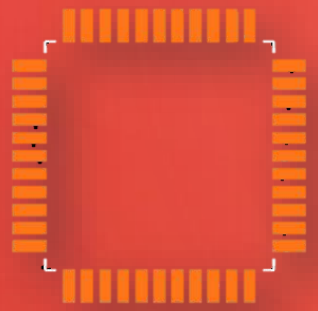
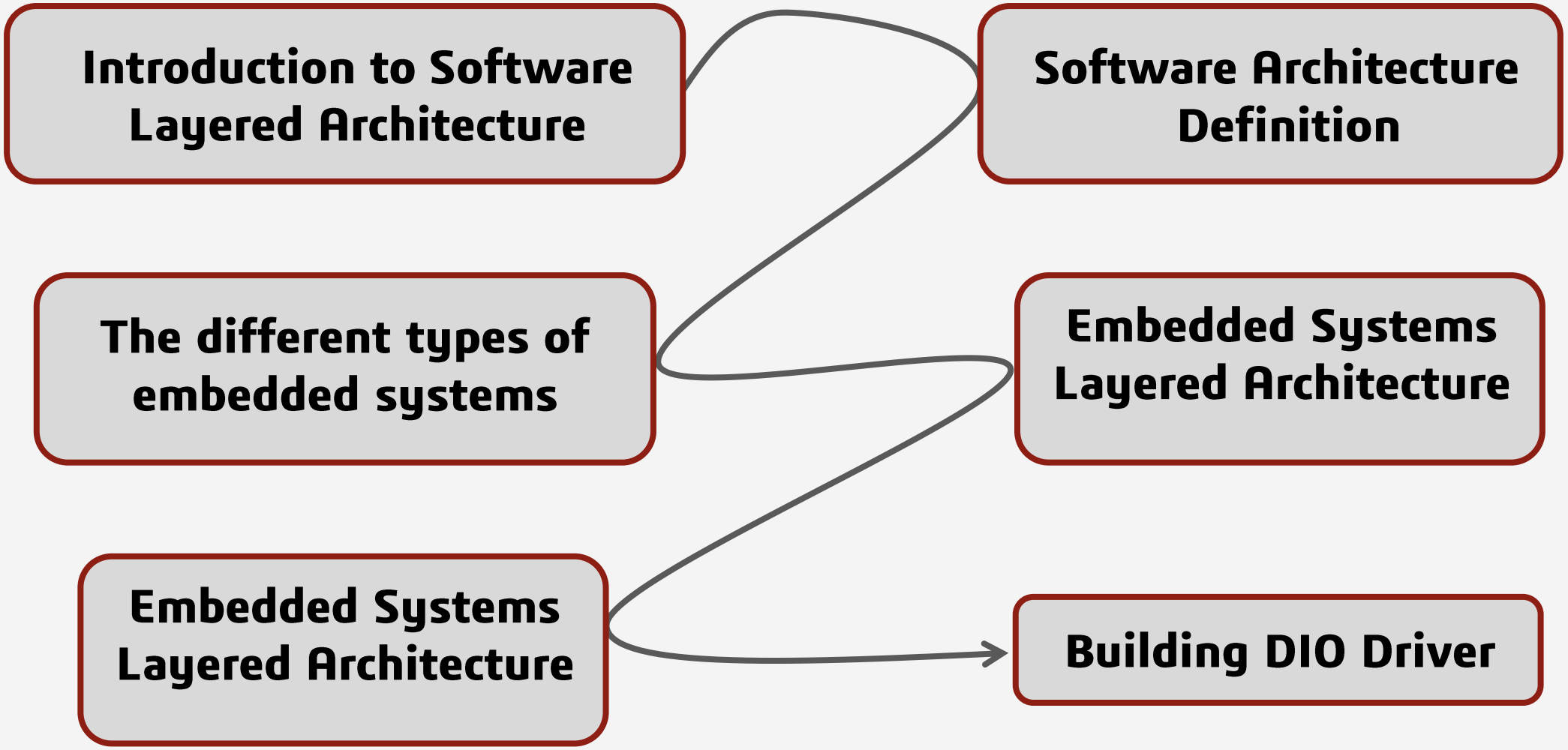


Interfacing

DIO layered architecture

AMIT

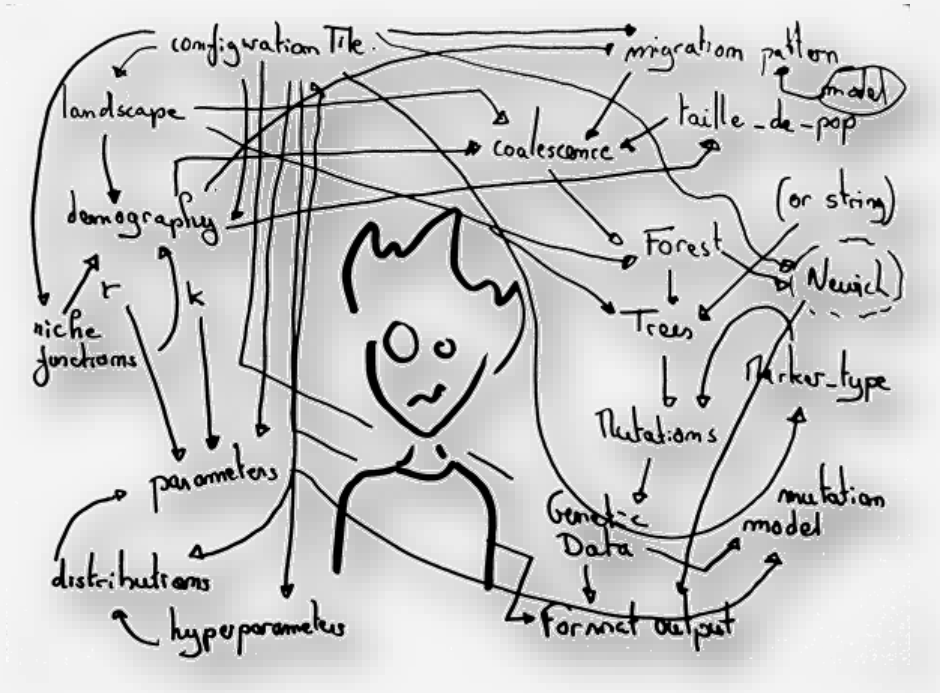
Content



AMIT

Introduction to Software Layered Architecture

Imagine having a source code file contains **10 thousands** lines of code. If you have a bug in this code, how complex is finding the bug ... ?

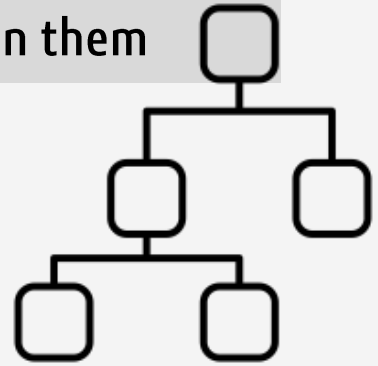


This is what's called **the Spaghetti Code**, which is the code that unstructured and difficult to maintain

AMIT

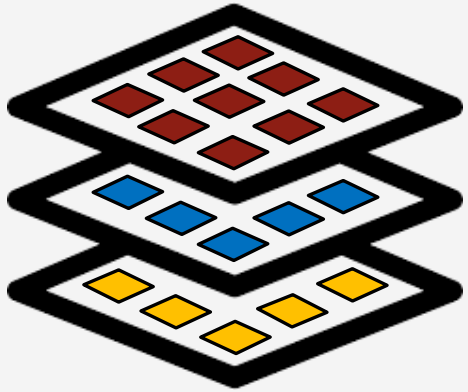
Embedded Systems Layered Architecture

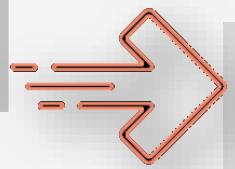
Definition **The software architecture** is a structuring way used to define software elements and relationships between them



In **Embedded Systems** we use a major type of software architecture called **Layered Architecture**

Layered Architecture In the layered architecture the software is divided into small parts called "Software Components**(SWC)**"





Software components related to each other are organized in a **horizontal layer**
Each layer is performing a **specific role**

AMIT

The different types of embedded systems

- Real-time embedded systems
- Standalone embedded systems
- Network, or networked, embedded systems
- Mobile embedded systems

Real-time embedded systems must provide results or outputs promptly.
Priority is assigned to output generation speed, as real-time embedded systems are often used in mission-critical sectors, such as defense and aerospace, that need important data

AMIT

The different types of embedded systems

- Real-time embedded systems
- Standalone embedded systems
- Network, or networked, embedded systems
- Mobile embedded systems

Standalone embedded systems don't require a host computer to function. They can produce outputs independently.

Examples:

- Digital cameras
- Digital wristwatches
- MP3 players
- Appliances, like refrigerators, washing machines
- Calculators

AMIT

The different types of embedded systems

- Real-time embedded systems
- Standalone embedded systems
- Network, or networked, embedded systems
- Mobile embedded systems

Network, or networked, embedded systems rely on wired or wireless networks and communication with web servers for output generation.

Frequently cited examples:

- Home and office security systems
- Automated teller machines (ATMs)
- Point-of-sale (POS) systems

AMIT

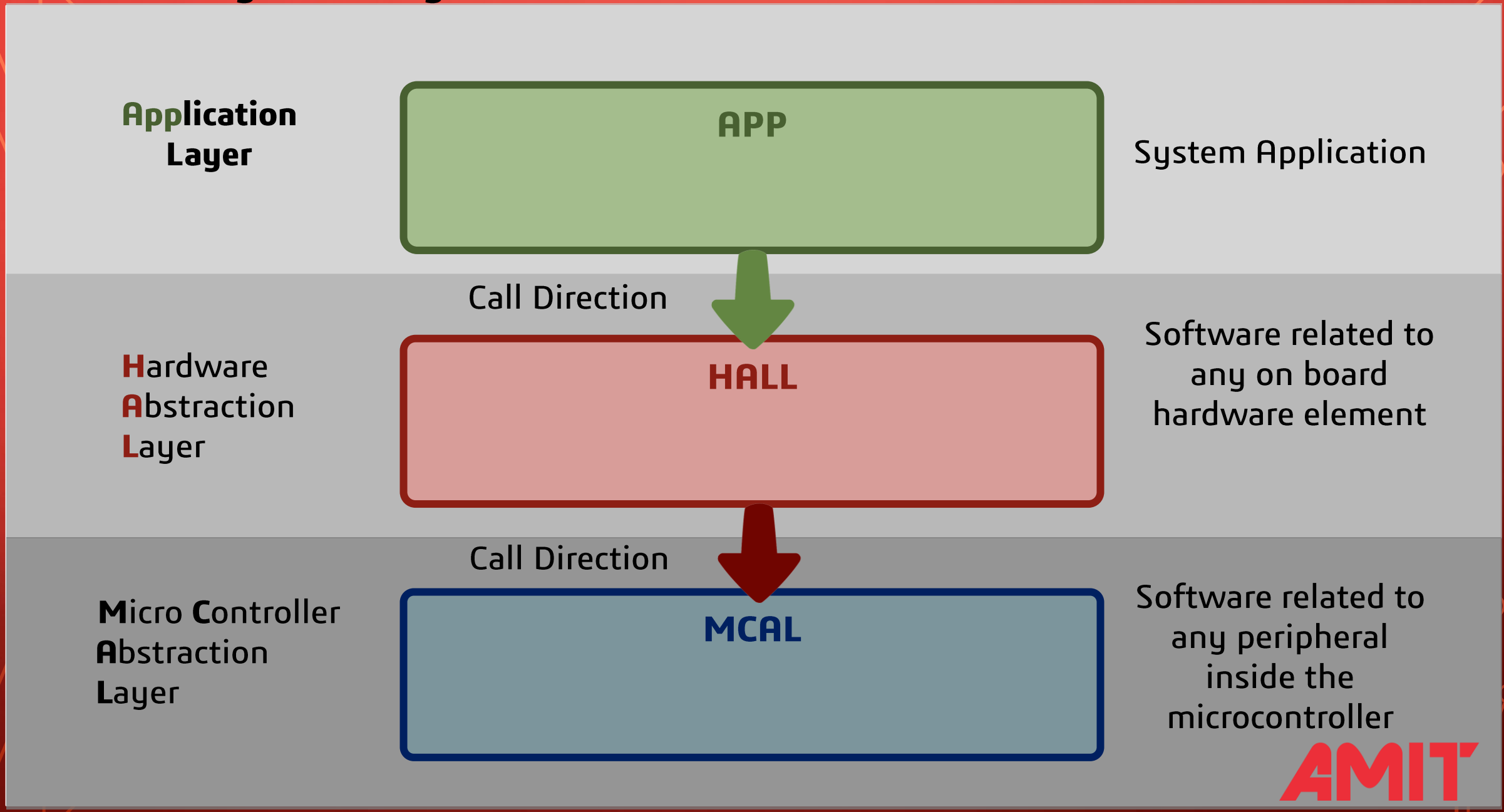
The different types of embedded systems

- Real-time embedded systems
- Standalone embedded systems
- Network, or networked, embedded systems
- Mobile embedded systems

Mobile embedded systems refer specifically to small, portable embedded devices, such as cellphones, laptops, and calculators.

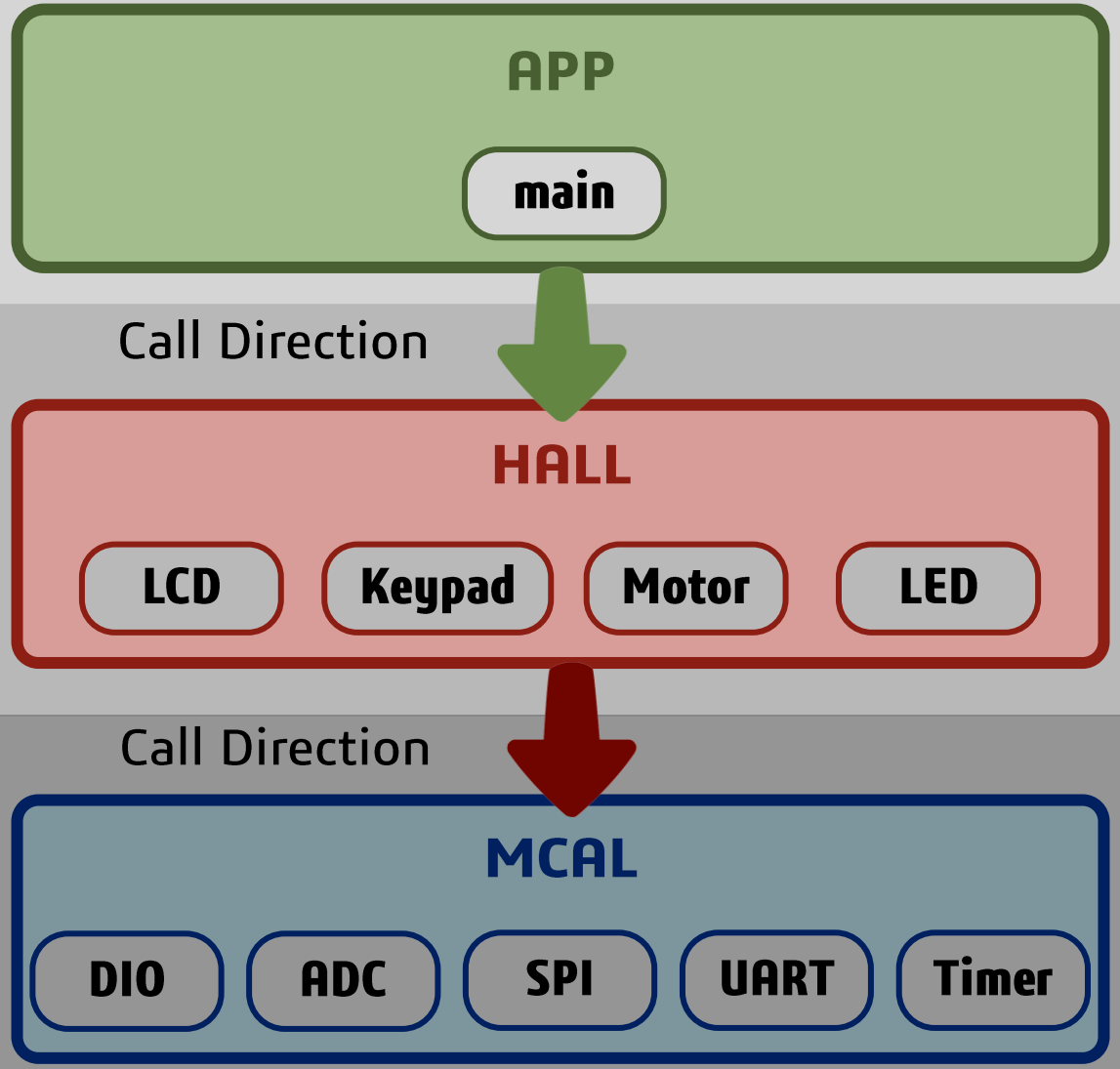
AMIT

Embedded Systems Layered Architecture



AMIT

Embedded Systems Layered Architecture



System Application

Software related to any on board hardware element

Software related to any peripheral inside the microcontroller

AMIT

Embedded Systems Layered Architecture

Advantages of Layered Architecture

1. **Modularity**
   In a Layered architecture we separate the user application from the hardware drivers from the microcontroller specific drivers
2. **Portability**
   Changing any part of the software part would change its layer only. For example, if we need the same application with a new microcontroller, we shall only change the MCAL
3. **Maintainability**
   Debugging and **Testing** is now much easier in small parts of the software instead of having a very long and complex one
4. **Reusability**
   Code could be easily **reused** in different applications and systems.

AMIT

Building DIO Driver

The Simplest driver consists of only 2 files

Program_Source file
Program.c

Contains the implementations of the functions provided by the driver
EX: DIO_program.c

Header file
Program.h

contains the prototypes of the functions provided by the driver to be used by other SWCs that need to use this driver
Ex: DIO_interface.h

AMIT

Building DIO Driver

Lets think about the DIO_Driver " DIO_Program.c & DIO_Interface.h "

Remember

DIO

Stands for " Digital Input / Output ": Is a peripheral that deals with **digital signals**, either by generating a digital signal **(Output Mode)** or by receiving it **(Input Mode)**

We want to Implement APIs or Functions that allow us to:

- API to Set a PORT Direction
  - I/P
  - O/P
- API to Set a PIN Direction
  - I/P
  - O/P
- API to Read a PIN's Value
  - 1
  - 0
- API to Read a PORT's Value
- API to Toggle a PIN
  - High
  - LOW

AMIT

Building DIO Driver

DIO_Interface.h → We will Write the prototypes of the functions

```
//Function that can set the direction of a PORT( A, B , C , D ) as Output or Input
        void   DIO_Set_PORT_DIR( PORT ,   DIR  );

//Function that can set the direction of a PIN( 0----> 7 ) as Output or Input
        void   DIO_Set_PIN_DIR(  PORT ,   PIN ,   DIR  );

//Function that can set the value allover PORT( A, B , C , D ) by a value
        void   DIO_Set_PORT_VALUE( PORT ,  Val  );

//Function that can set the value of a PIN ( 0----> 7 ) to 0 or 1
        void   DIO_Set_PIN_VALUE( PORT ,  PIN  ,  Val  );

//Function that can read the value of the PORT( A , B , C , D )
        void   DIO_Read_PORT( PORT ,   *val  );

//Function that can read the value of the PIN ( 0----> 7 ) to 0 or 1
        void   DIO_Read_PIN (  PORT ,  PIN  ,  *val  );

//Function that can toggle the pin value o to 1 and 1 to 0
        void   DIO_Toggle_PIN (  PORT ,  PIN  );
```

Let's Go and Implement these functions

AMIT

Building DIO Driver

DIO_Program.c

```
/*Function that can set the direction of a PORT( A, B , C , D ) as Output
or Input */
void  DIO_Set_PORT_DIR( PORTx ,  DIRx  )
      {
```

" To control the mode of the pins of PORTx to be output or input "

- The only step that we want here to assign the value of the **Data Direction Register "DDRx"**
- To set the PORTx's Direction to be **an output**, We will set **"1"** to all the bits in that Register Assuming that we deal with ATmega32 and its architecture is 1 byte which equals to **8 bit**, so we will assign **0xff " 1111 1111 " in the Data Direction** Register, and then we will successfully put **"1"** in each bit **" DDRx = 0xff "**

- To set the PORTx's Direction to be **an input**, We will set **"0"** to all the bits in that so we will assign **0x00 " 0000 0000 "** in the Data Direction Register , and then we will successfully put **"0"** in each bit **" DDRx = 0x00 "**

AMIT

Building DIO Driver

DIO_Program.c

- We will create a conditional statement to allow us to access to the actual location for the PORT selected and then, select the desired direction

- switch(PORT)
  case: PORTA
  DDRA = Dirx
  case: PORTB
  DDRB = Dirx
  - - - - - - - -
  - - - - - - - -
  - - - - - - - -
  - - - - - - - -
  }

1 For Output

DIRx

0 For Input

B C

A PORTx D

AMIT

Building DIO Driver

DIO_Program.c

```
/*Function that can set the direction of a PIN( 0----> 7 ) as Output or Input*/
void   DIO_Set_PIN_DIR( PORTx , PINx ,  DIR  )
        {
```

" To control the mode of the PINx to be output or input "

- The only step that we want here to assign the value of the **Data Direction Register "DDRx"**
- To set the PINx Direction to be **an output**, We will set **"1"** to the corresponding bit in that Register

" DDRx = the corresponding bit for the pin selected "

- To set the PINx Direction to be **an input**, We will set **"0"** to the corresponding bit in the **Data Direction Register "DDRx"**

" DDRx = the corresponding bit for the pin selected "

```
        }
```

AMIT

Building DIO Driver

DIO_Program.c

- We will create a conditional statement to check the direction an then we will switch between the PORTs to set the desired PIN

```
switch(Dir)
 case: OUTPUT
     switch(PORT)
         case: PORTA
          Set_Bit(DDRA , PINx)
         case: PORTB
          Set_Bit(DDRB , PINx)
              - - - - - - - -
              - - - - - - - -
              - - - - - - - -
              - - - - - - - -
   case: INPUT
    switch(PORT)
         case: PORTA
         CLR_Bit(DDRA , PINx)
         case: PORTB
         CLR_Bit(DDRB , PINx)
              - - - - - - - -
              - - - - - - - -
              - - - - - - - -
}             - - - - - - - -
```

1 For Output

DIRx

0 For Input

AMIT

Building DIO Driver

DIO_Program.c

```
//Function that can set the value allover PORT( A, B , C , D ) by a value
void   DIO_Set_PORT_VALUE( PORT ,  Val )
{
```

- We will create a conditional statement to allow us to access to the actual location for the PORT selected and then, will assign the desired value

- switch(PORT)

```
 case: PORTA
 PORTA = val
 case: PORTB
 PORTB = val
 - - - - - - - -
 - - - - - - - -
 - - - - - - - -
 - - - - - - - -
}
```

AMIT

Building DIO Driver

DIO_Program.c

```
//Function that can set the value of a PIN ( 0----> 7 ) to 0 or 1
void  DIO_Set_PIN_VALUE( PORT , PIN , Val )
{
```

- We will create a conditional statement to allow us to access to the actual location for the PORT selected and then, will assign the desired value to the selected PIN in its register

- switch(PORT)
  case: PORTA
  PINA = val
  case: PORTB
  PIN = val

- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -

```
}
```

AMIT

Building DIO Driver

DIO_Program.c

```
        //Function that can read the value of the PORT( A , B , C , D )
void   DIO_Read_PORT( PORT ,   *val  )
{
```

- We will create a conditional statement to allow us to which PORT selected and then, will can read the actual value to the its PINx into val

- switch(PORT)
  case: PORTA
  *val = PINA
  case: PORTB
  *val = PINB

```
}
```

Building DIO Driver

DIO_Program.c

```
//Function that can read the value of the PIN ( 0----> 7 ) to 0 or 1
void  DIO_Read_PIN ( PORT , PIN ,  *val  )
{
```

- We will create a conditional statement to allow us to which PORT selected and then, will can get the actual value of the selected PINx into val

- switch(PORT)

```
 case: PORTA
 *val = Get_BIT(PINA, Pin)
 case: PORTB
  *val = Get_BIT(PINB, Pin)
 - - - - - - -
 - - - - - - -
}  - - - - - - -
 - - - - - - -
```

AMIT

Building DIO Driver

DIO_Program.c

```
        //Function that can toggle the pin value o to 1 and 1 to 0
void   DIO_Toggle_PIN ( PORT , PIN )
{
```

- We will create a conditional statement to allow us to which PORT selected and then, will can toggle the actual value of the selected PINx

- switch(PORT)
  case: PORTA
  Toggle_BIT(PORTA, Pin)
  case: PORTB
  Toggle_BIT(PORTB, Pin)

```
        }
```

THANK YOU!

AMIT