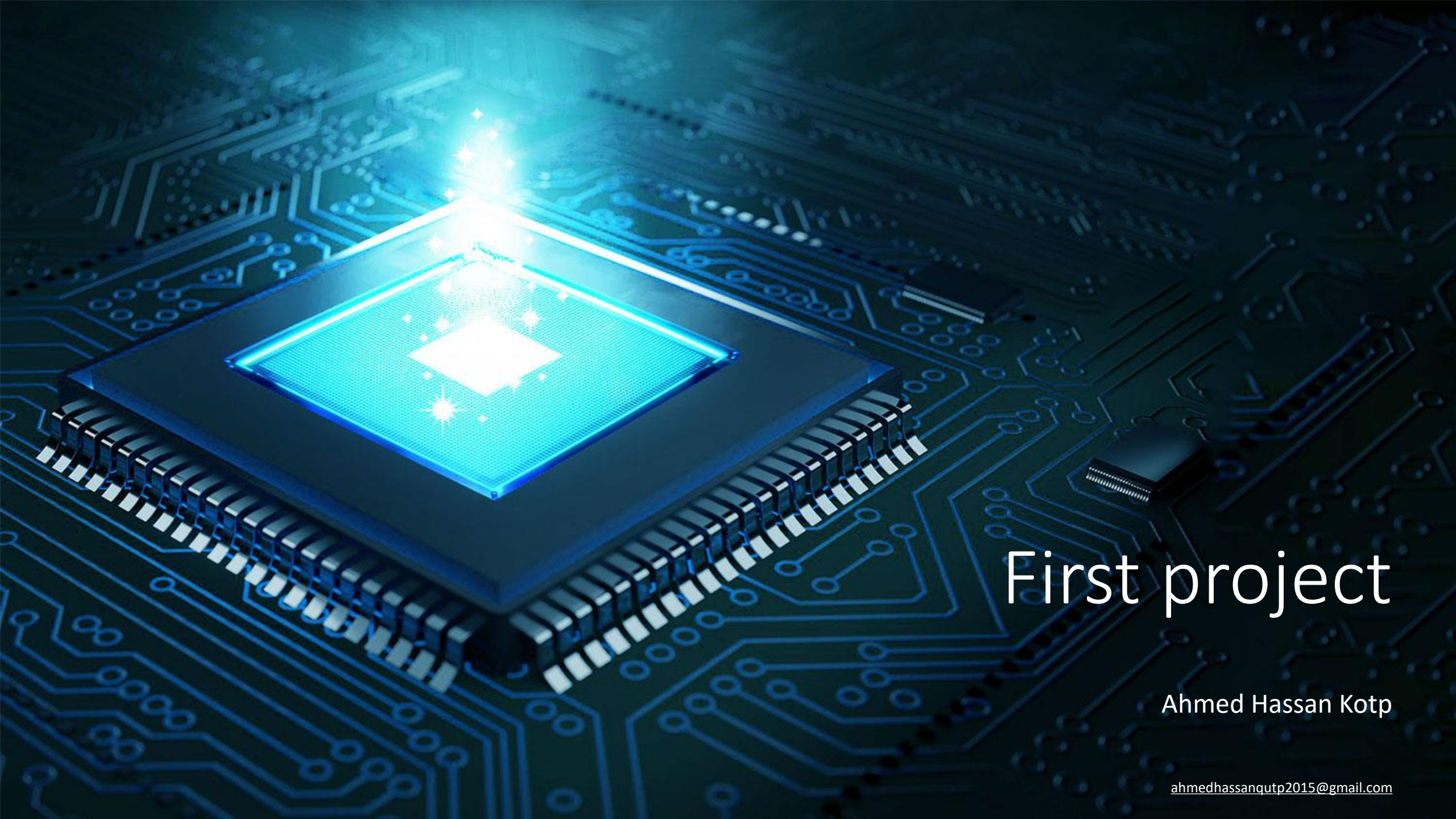
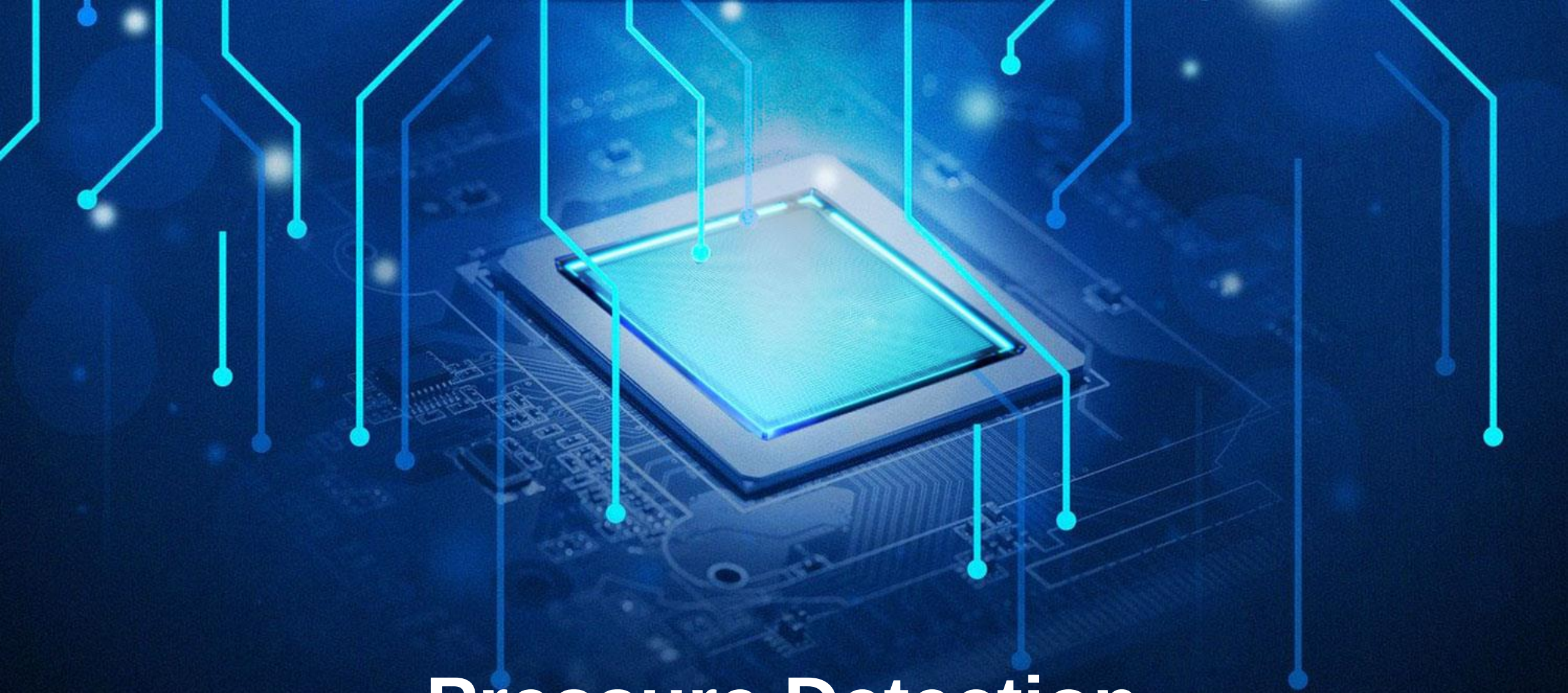


First project

Ahmed Hassan Kotp

ahmedhassanqutp2015@gmail.com



Pressure Detection

Learn_in_Depth

Agenda Style

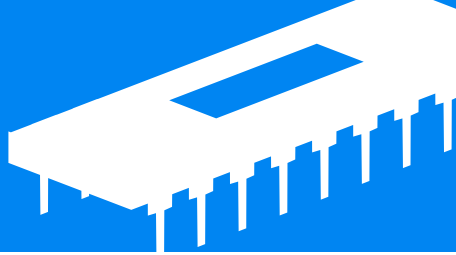
01 Design Architecting

02 Code implementation

03 Compilation processes

04 Project simulation

Design Architecting



4 — Space Exploration

Analyzing various functionally equivalent implementation alternatives

3 — Requirement

Establishing the goals and constrains of system

2 — Method

SDLC/STLC

1 — Case study

Understanding Specification of client

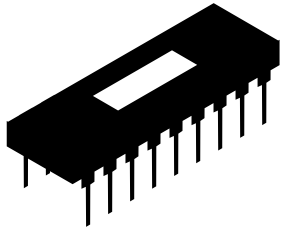
5 — System Analysis

Understanding what a client wants

6 — System Design

Making what a client wants.

7 — Testing



Burn code in MCU

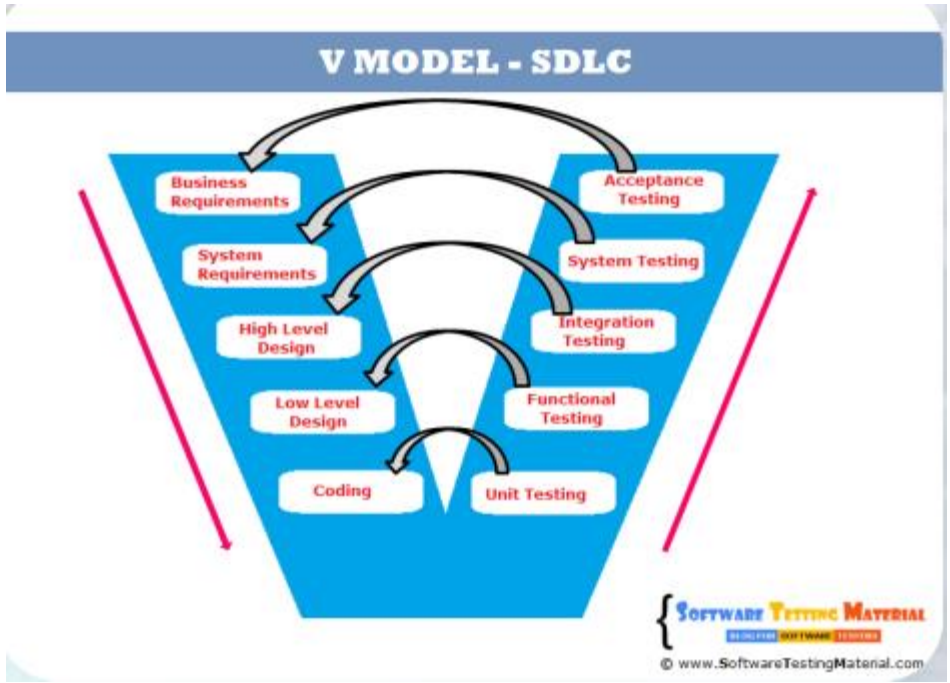
Case Study

A ”client” expects you to deliver the software of the following system:

- ❖ Specification (from the client)
  - ➢ A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin .
  - ➢ The alarm duration equals 60 seconds.
  - ➢ keeps track of the measured values.

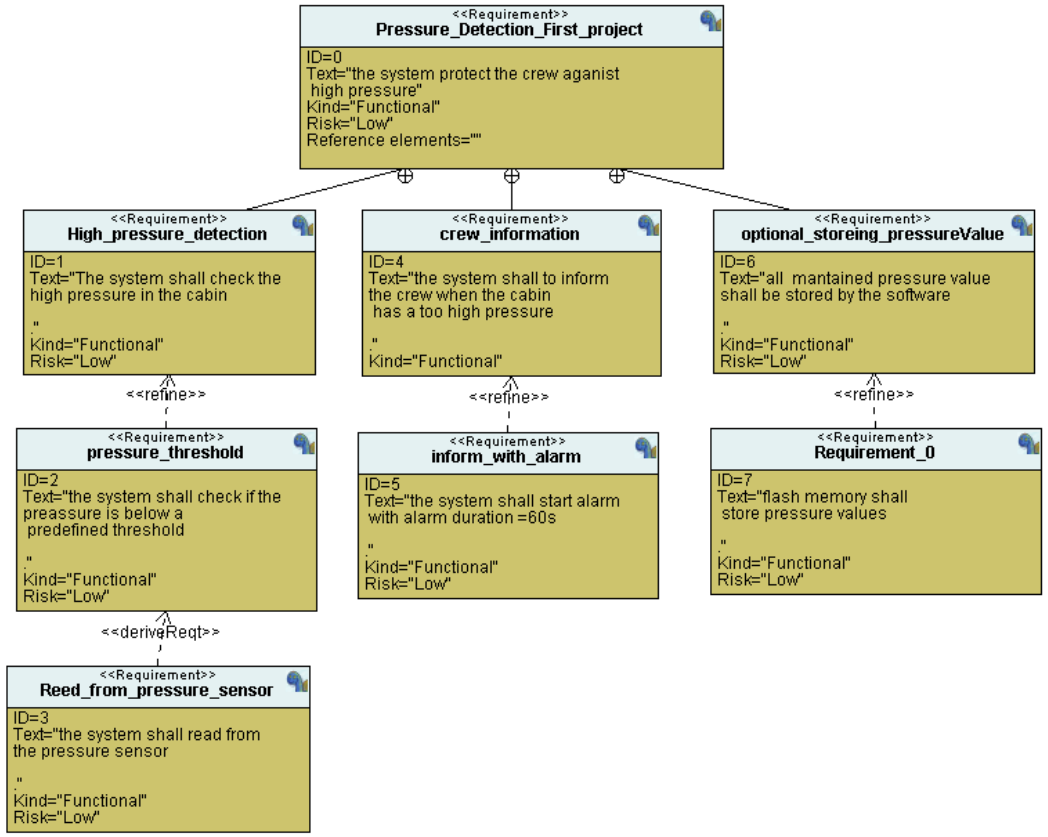
- ❖ Pressure Controller: Assumptions
  - ➢ The controller set up and shutdown procedures are not modeled .
  - ➢ The controller maintenance is not modeled
  - ➢ The pressure sensor never fails
  - ➢ The alarm never fails
  - ➢ The controller never faces power cut

Methods



The Software Development Life Cycle (SDLC) is a structured approach to software development that encompasses the processes, methodologies, and activities involved in creating and maintaining software systems. While the SDLC is applicable to various software development domains, including embedded software development, there are specific considerations and methodologies that are commonly used in the context of embedded software.

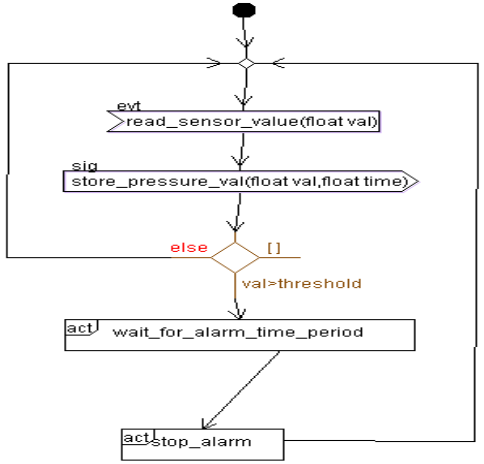
Requirements

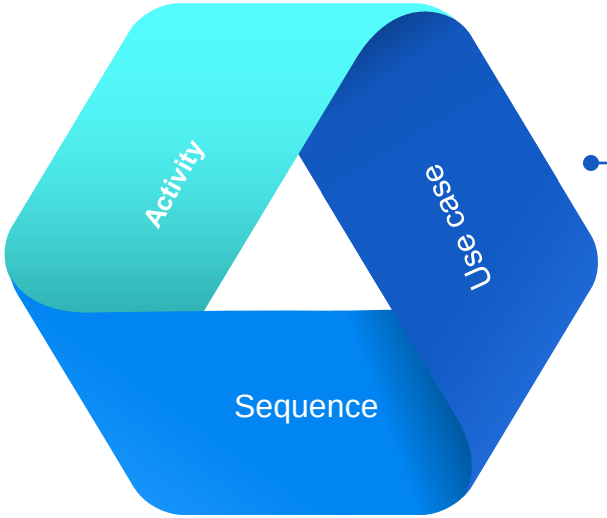


- A ”client” expects you to deliver the software of the following system:
- ❖ Specification (from the client)
- ✓ A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin.
- ✓ The alarm duration equals 60 seconds.
- ✓ Optional: keeps track of the measured values.

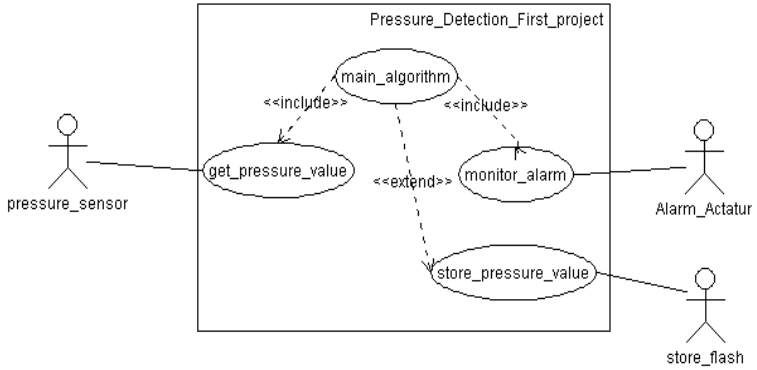
System Analysis

Diagram

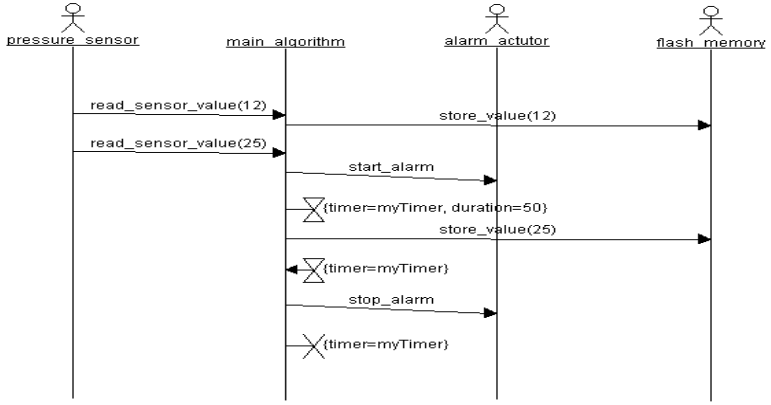




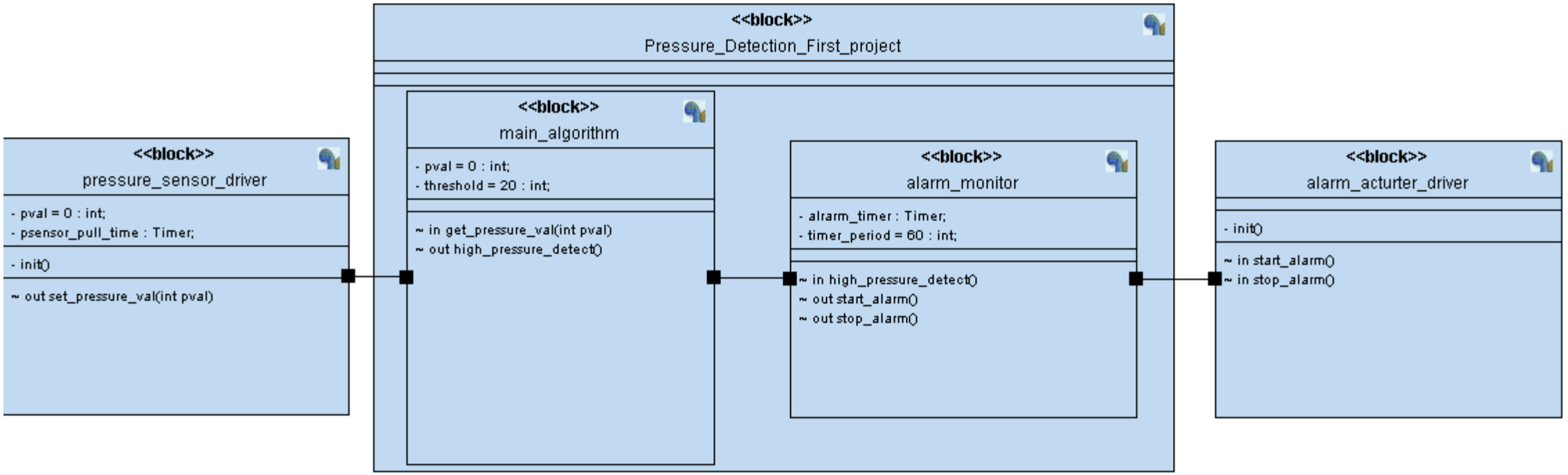
Diagram



Diagram

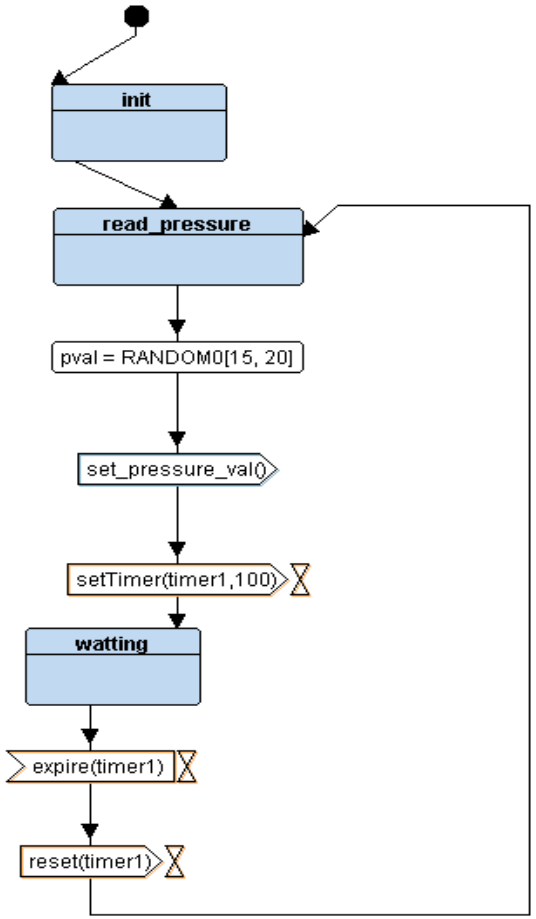


System Design

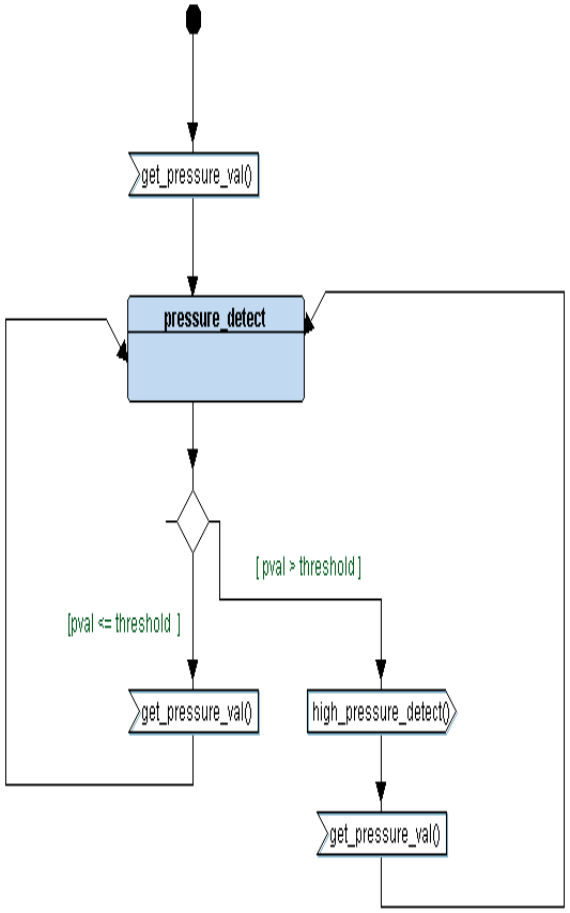


System Design

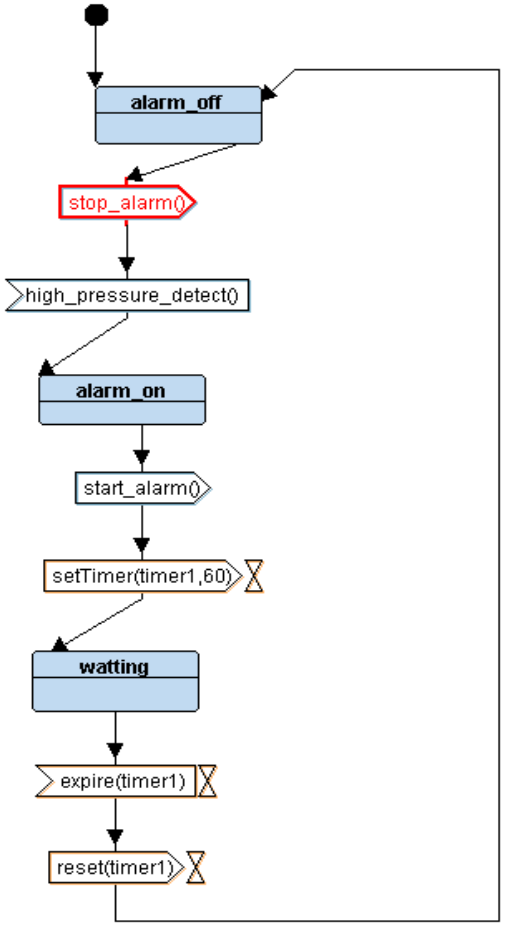
Pressure sensor



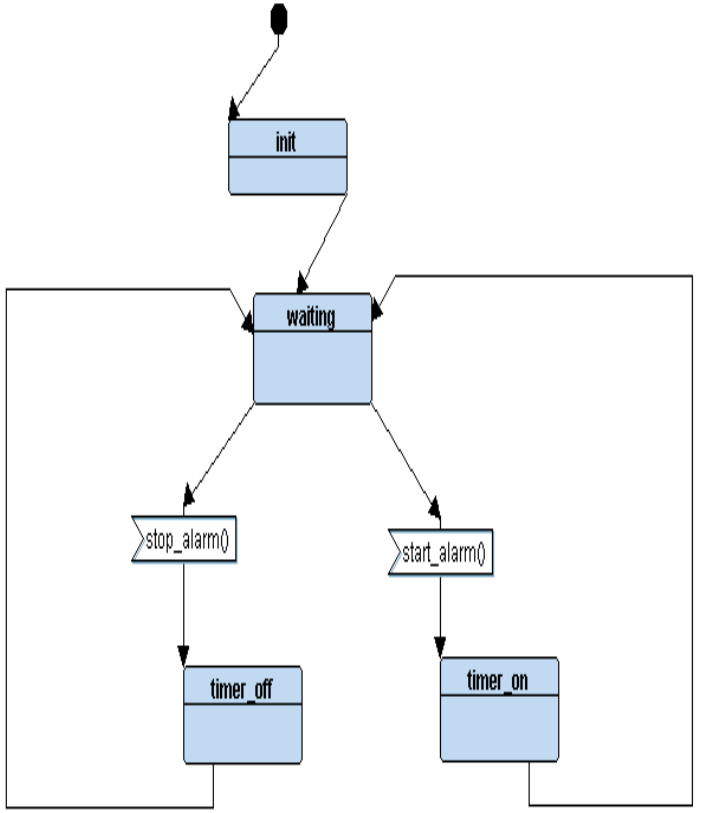
Main algorithm

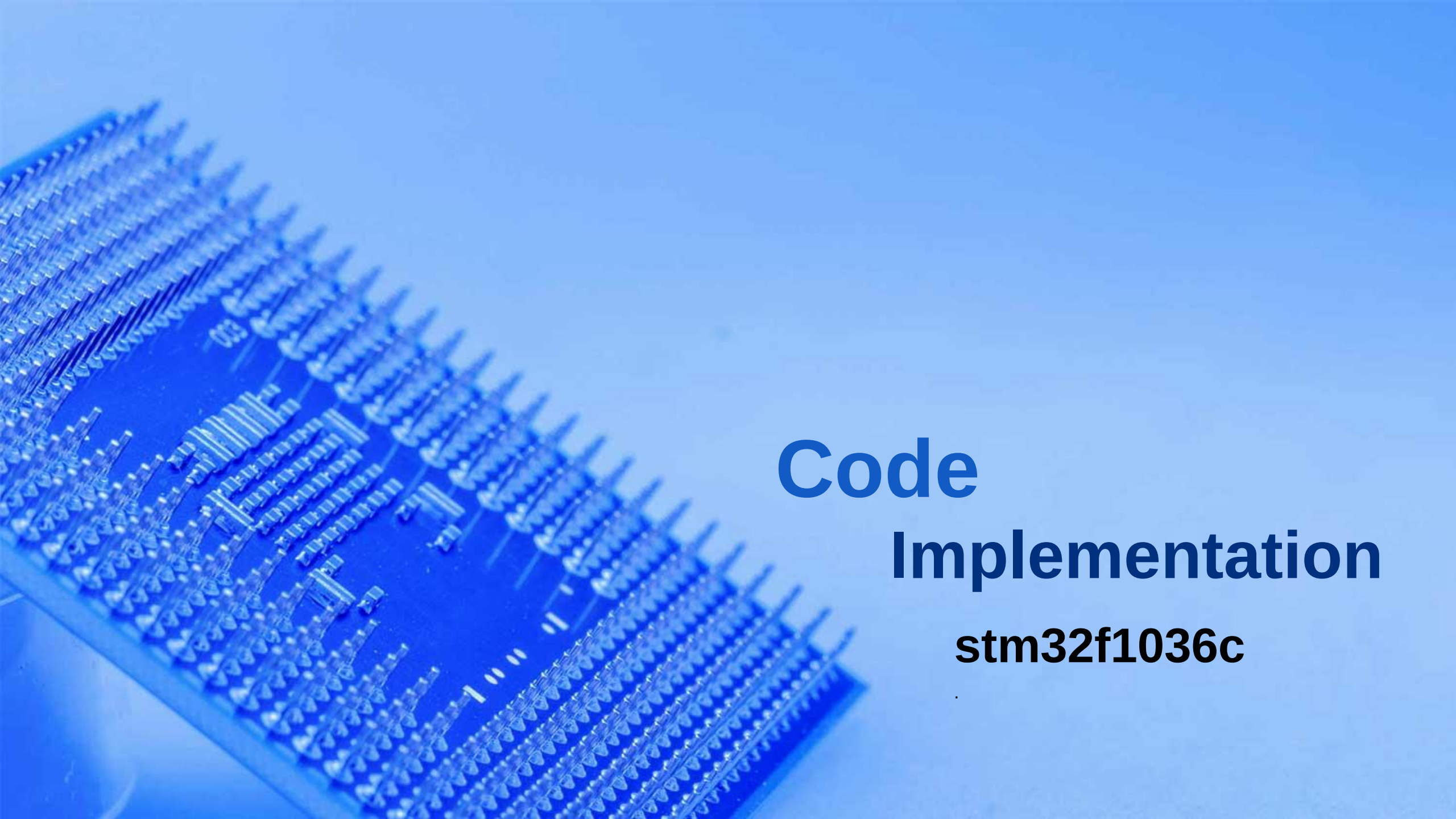


Alarm monitor



Alarm acteuater





Code Implementation

stm32f1036c

```c
/*
 * state.h
 *
 *  Created on: Jun 27, 2023
 *      Author: Egypt_Laptop
 */

#ifndef STATE_H_
#define STATE_H_

#include "STD_TYPES.h"

#define STATE_define(_state_)  void st_##_state_()
#define STATE(_state_)         st_##_state_

void set_pressure_val(uint8 prev);

void high_presure_detect();

void start_alarm();

void stop_alarm();

#endif /* STATE_H_ */
```

```c
#include "driver.h"
#include "pressure_sensor.h"
#include "alarm_monitor.h"
#include "alarm.h"
#include "detect.h"
#include "STD_TYPES.h"

void setup()
{
    GPIO_INITIALIZATION();
    pre_sen_init();
    detect_st=STATE(pressure_detect);
    alr_mon_st=STATE(alarm_off);
    alarm_init();
}
int main(){
    setup();
    while(1)
    {
        pre_st();
        detect_st();
        alr_mon_st();
        alarm_st();
    }


}
```

```c
/*
 * pressure_sensor.h
 *
 *  Created on: Jun 27, 2023
 *      Author: Egypt_Laptop
 */

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include "state.h"

extern void (*pre_st)();

typedef enum{
    read_pressure,
    waiting
}pre_state_en;

extern pre_state_en pre_state_id;

STATE_define(read_pressure);

STATE_define(waiting);

void pre_sen_init();
```

```c
#include "pressure_sensor.h"
#include "STD_TYPES.h"
#include "driver.h"
uint8 pval=0;

void (*pre_st)();

pre_state_en pre_state_id;

void pre_sen_init()
{
    //init  pre_sensor
    pre_st=STATE(read_pressure);
}

STATE_define(read_pressure)
{
    pre_state_id=read_pressure;
    pval=getPressureVal();
    set_pressure_val(pval);
    pre_st=STATE(read_pressure);
}

STATE_define(waiting)
{
    pre_state_id=waiting;
    pre_st=STATE(read_pressure);
}
```

```c
/*
 * detect.h
 *
 *  Created on: Jun 27, 2023
 *      Author: Egypt_Laptop
 */

#ifndef DETECT_H_
#define DETECT_H_

#include "state.h"

extern void (*detect_st)();

typedef enum{
    pressure_detect
}detect_st_en;

extern detect_st_en detect_st_id;

STATE_define(pressure_detect);



#endif /* DETECT_H_ */
```

```c
#include "detect.h"
#include "STD_TYPES.h"

uint8 pre_val=0;
uint8 last_val=0;
uint8 threshold=20;

void (*detect_st)();

detect_st_en detect_st_id;

void set_pressure_val(uint8 pval)
{
    last_val=pre_val;
    pre_val=pval;
}


STATE_define(pressure_detect)
{
    detect_st_id=pressure_detect;
    if (pre_val > threshold && pre_val!=last_val)
    {
        high_presure_detect();
        detect_st=STATE(pressure_detect);
    }
    else
    {
        detect_st=STATE(pressure_detect);
    }
}
```

```c
/*
 * alarm_monitor.h
 *
 *  Created on: Jun 27, 2023
 *      Author: Egypt_Laptop
 */

#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_

#include "state.h"

extern void (*alr_mon_st)();

typedef enum{
    alarm_off,
    alarm_on,
    timer_waiting
}alr_mon_st_en;

extern alr_mon_st_en alr_mon_st_id;

STATE_define(alarm_off);
STATE_define(alarm_on);
STATE_define(timer_waiting);

#endif /* ALARM_MONITOR_H_ */
```

```c
#include "alarm_monitor.h"
#include "driver.h"

void (*alr_mon_st)();

alr_mon_st_en alr_mon_st_id;

void high_presure_detect()
{
    alr_mon_st=STATE(alarm_on);
}

STATE_define(alarm_off)
{
    alr_mon_st_id=alarm_off;
    stop_alarm();
}
STATE_define(alarm_on)
{
    alr_mon_st_id=alarm_on;
    start_alarm();
    alr_mon_st=STATE(alarm_off);
}
STATE_define(timer_waiting)
{
    alr_mon_st_id=timer_waiting;
    alr_mon_st=STATE(alarm_off);
}
```

```c
/*
 * alarm.h
 *
 *  Created on: Jun 27, 2023
 *      Author: Egypt_Laptop
 */

#ifndef ALARM_H_
#define ALARM_H_

#include "state.h"

extern void (*alarm_st)();

typedef enum{
    alarm_start,
    alarm_stop,
    alarm_waiting
}alarm_st_en;

extern alarm_st_en alarm_st_id;

STATE_define(alarm_start);
STATE_define(alarm_stop);
STATE_define(alarm_waiting);

void alarm_init();

#endif /* ALARM_H_ */
```

```c
/*
 * alarm.c
 *
 *  Created on: Jun 27, 2023
 *      Author: Egypt_Laptop
 */

#include "alarm.h"
#include "driver.h"

void (*alarm_st)();

alarm_st_en alarm_st_id;

void alarm_init()
{
   //alarm_init
    alarm_st=STATE(alarm_waiting);
}

void start_alarm()
{
    alarm_st=STATE(alarm_start);
}

void stop_alarm()
{
    alarm_st=STATE(alarm_stop);
}
```

```c
STATE_define(alarm_start)
{
    alarm_st_id=alarm_start;
    Set_Alarm_actuator(0);
    Delay(2000000);
    alarm_st=STATE(alarm_waiting);
}

STATE_define(alarm_stop)
{
    alarm_st_id=alarm_stop;
    Set_Alarm_actuator(1);
    alarm_st=STATE(alarm_waiting);
}

STATE_define(alarm_waiting)
{
    alarm_st_id=alarm_waiting;
}
```

```h
#include <stdint.h>
#include <stdio.h>

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))


#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)


void Delay(int nCount);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();
```

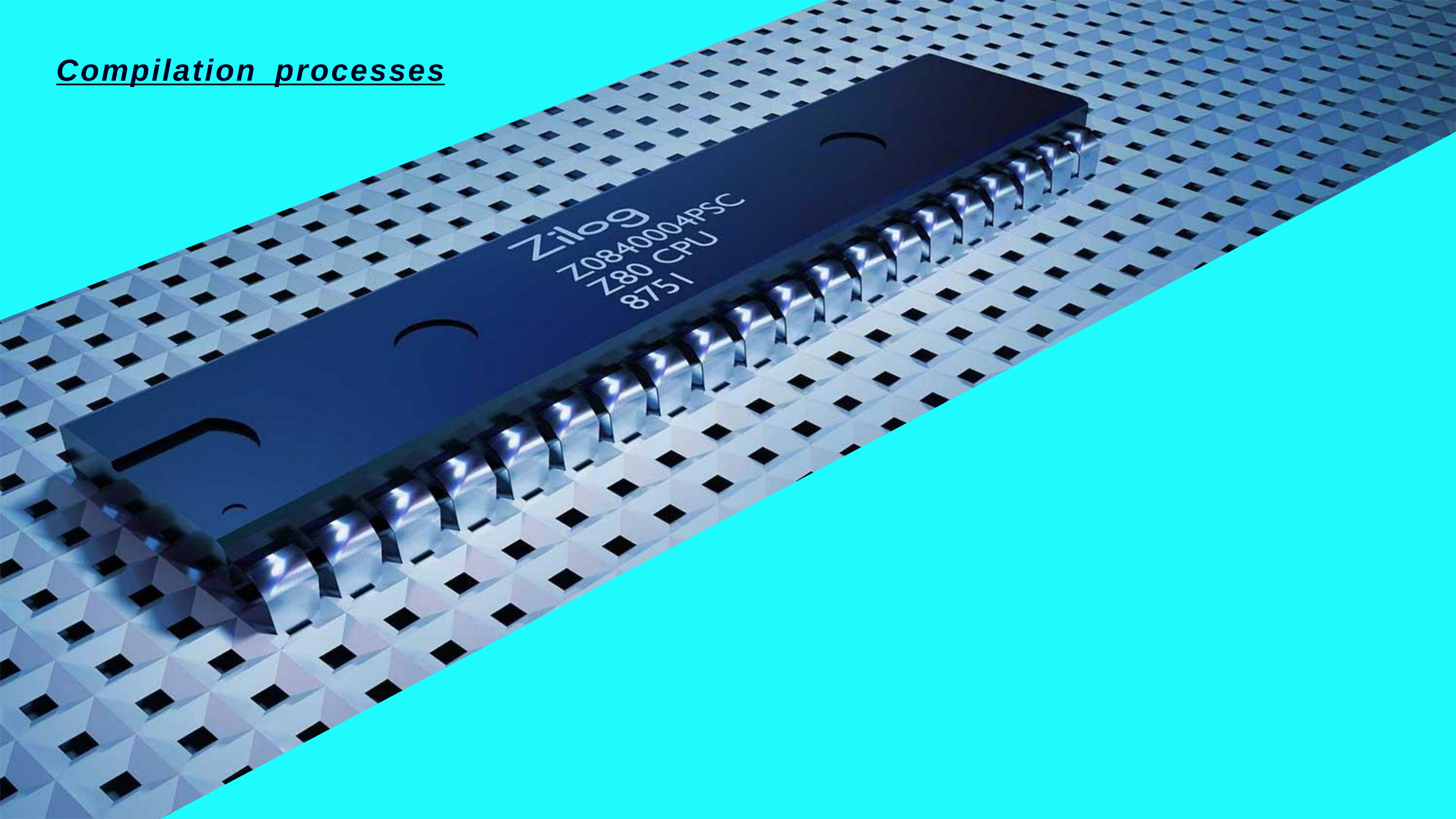
```h
#include "driver.h"
#include <stdint.h>
#include <stdio.h>
void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}

int getPressureVal(){
    return (GPIOA_IDR & 0xFF);
}

void Set_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);
    }
}

void GPIO_INITIALIZATION (){
    SET_BIT(APB2ENR, 2);
    GPIOA_CRL &= 0xFF0FFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFF;
    GPIOA_CRH |= 0x22222222;
}
```

Compilation processes



```c
//startup code

/*LEARN_IN_DEPTH
UNIT2_LESSON2_LAP2
ENG: AHMED HASSAN
*/
#include "STD_TYPES.H"

extern uint32 stack_top;
extern uint32 _S_DATA;
extern uint32 _E_DATA;
extern uint32 _S_BSS;
extern uint32 _E_BSS;
extern uint32 _E_text;


extern int main(void);
void Rest_Handler();
void init_sect(void);

void Default_handler(void)
{
    Rest_Handler();
}

void NMI_Handler(void)          __attribute__((weak , alias("Default_handler")));;
void H_Fault_Handler(void)      __attribute__((weak , alias("Default_handler")));;
void MM_Handler(void)           __attribute__((weak , alias("Default_handler")));;
void Bus_Handler(void)          __attribute__((weak , alias("Default_handler")));;
void Usage_Fault_Handler(void)  __attribute__((weak , alias("Default_handler")));;
```

```c
uint32 vectors[] __attribute__((section(".vectors")))=
{
    (uint32) &stack_top,
    (uint32) &Rest_Handler,
    (uint32) &NMI_Handler,
    (uint32) &H_Fault_Handler,
    (uint32) &MM_Handler,
    (uint32) &Bus_Handler,
    (uint32) &Usage_Fault_Handler,
};
void Rest_Handler(void)
{
    init_sect();
    main();
}
void init_sect(void)
{
    uint32 DATA_size = (uint32*)&_E_DATA - (uint32*)&_S_DATA;
    uint8 *P_src = (uint8*)&_E_text;
    uint8 *P_dst = (uint8*)&_S_DATA;

    for (uint32 i=0;i<DATA_size;i++)
    {
        *((uint8*)P_dst++) = *((uint8*)P_src++);
    }

    uint32 bss_size = (uint32*)&_E_BSS - (uint32*)&_S_BSS;
    P_dst =(uint8*)&_S_BSS;
    for (uint32 i=0;i<bss_size;i++)
    {
        *((uint8*)P_dst++) = (uint8)0;
    }
```

```
/*LEARN_IN_DEPTH
   linker script
ENG: AHMED HASSAN
*/
ENTRY(Rest_Handler)

MEMORY
{
    flash (RX) : ORIGIN = 0X08000000, LENGTH = 512M
    sram (RWX)  : ORIGIN = 0x20000000, LENGTH = 512M
}

SECTIONS
{
        .text :
        {
            *(.vectors*)
            *(.text*)
            *(.rodata*)
            . = ALIGN(4);
            _E_text = . ;
        }> flash
    .data :
        {
            _S_DATA = .;
            *(.data)
            . = ALIGN(4);
            _E_DATA = .;
        }> sram AT> flash
    .bss :
        {
            _S_BSS = .;
            *(.bss) *(.COMMON)
            . = ALIGN(4);
            _E_BSS = .;
        }> sram
    . = . + 0x1000;  /*4kB of stack memory*/
    stack_top = . ;
}
```

```
#make file
#prepared by Ahmed Hassan (learn_in_depth)

project name=pressure_detector

CC=arm-none-eabi-
CFLAGS=-g -gdwarf-2 -mcpu=cortex-m3
INCS=-I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As  = $(wildcard *.s)
AsOBJ = $(As:.s=.o)

all:$(project name).bin
    @echo ">>>>>>>>>>>>>Duild is Done<<<<<<<<<<<<<"

%.o: %.c
    $(CC)gcc.exe -c  $(CFLAGS) $(INCS) $< -o $@

%.o: %.s
    $(CC)as.exe  -c  $(CFLAGS) $(INCS) $< -o $@

$(project name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ)  -o $@ -Map=output.map

$(project name).bin: $(project name).elf
    $(CC)objcopy.exe -O binary $< $@

clean_all:
    rm *.o *.elf *.bin *.map map.asm

clean:
    rm *.elf *.elf
```

```
gucts/First_Project (main)
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 main.c -o main.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitu
gucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t main.o

main.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 main.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     F .text  0000004c setup
00000000         *UND*  00000000 GPIO_INITIALIZATION
00000000         *UND*  00000000 pre_sen_init
00000000         *UND*  00000000 alarm_init
00000000         *UND*  00000000 detect_st
00000000         *UND*  00000000 st_pressure_detect
00000000         *UND*  00000000 alr_mon_st
00000000         *UND*  00000000 st_alarm_off
0000004c g     F .text  00000060 main
00000000         *UND*  00000000 pre_st
00000000         *UND*  00000000 alarm_st



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitu
gucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000ac  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000e0  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000e0  2**0
                  ALLOC
  3 .comment      0000004a  00000000  00000000  000000e0  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002a  00000000  00000000  0000012a  2**0
                  CONTENTS, READONLY

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitu
gucts/First_Project (main)
```

```
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 driver.c -o driver.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded syster
gucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t driver.o

driver.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 driver.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     F .text  00000040 Delay
00000040 g     F .text  00000028 getPressureVal
00000068 g     F .text  00000068 Set_Alarm_actuator
000000d0 g     F .text  0000008c GPIO_INITIALIZATION



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded syster
gucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h driver.o

driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000015c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000190  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000190  2**0
                  ALLOC
  3 .comment      0000004a  00000000  00000000  00000190  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002a  00000000  00000000  000001da  2**0
                  CONTENTS, READONLY
```

```
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 pressure_sensor.c -o pressure_sensor.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Maste
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t pressure_sensor.o

pressure_sensor.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 pressure_sensor.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     O .bss   00000001 pval
00000004 g     O .bss   00000004 pre_st
00000008 g     O .bss   00000001 pre_state_id
00000000 g     F .text  0000002c pre_sen_init
0000002c g     F .text  00000064 st_read_pressure
00000000         *UND*  00000000 getPressureVal
00000000         *UND*  00000000 set_pressure_val
00000090 g     F .text  0000003c st_waiting



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Maste
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h pressure_sensor.o

pressure_sensor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000cc  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000100  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000009  00000000  00000000  00000100  2**2
                  ALLOC
  3 .comment      0000004a  00000000  00000000  00000100  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002a  00000000  00000000  0000014a  2**0
                  CONTENTS, READONLY
```

```
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 detect.c -o detect.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t detect.o

detect.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 detect.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     O .bss   00000001 pre_val
00000001 g     O .bss   00000001 last_val
00000000 g     O .data  00000001 threshold
00000004 g     O .bss   00000004 detect_st
00000008 g     O .bss   00000001 detect_st_id
00000000 g     F .text  00000048 set_pressure_val
00000048 g     F .text  00000090 st_pressure_detect
00000000         *UND*  00000000 high_presure_detect



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h detect.o

detect.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000d8  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000001  00000000  00000000  0000010c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000009  00000000  00000000  00000110  2**2
                  ALLOC
  3 .comment      0000004a  00000000  00000000  00000110  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002a  00000000  00000000  0000015a  2**0
                  CONTENTS, READONLY
```

```
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 alarm_monitor.c -o alarm_monitor.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Master
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t alarm_monitor.o

alarm_monitor.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 alarm_monitor.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     O .bss   00000004 alr_mon_st
00000004 g     O .bss   00000001 alr_mon_st_id
00000000 g     F .text  0000002c high_presure_detect
00000058 g     F .text  00000040 st_alarm_on
0000002c g     F .text  0000002c st_alarm_off
00000000         *UND*  00000000 stop_alarm
00000000         *UND*  00000000 start_alarm
00000098 g     F .text  0000003c st_timer_waiting



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Master
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h alarm_monitor.o

alarm_monitor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000d4  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000108  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000005  00000000  00000000  00000108  2**2
                  ALLOC
  3 .comment      0000004a  00000000  00000000  00000108  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002a  00000000  00000000  00000152  2**0
                  CONTENTS, READONLY
```

```
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 alarm.c -o alarm.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded syst
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t alarm.o

alarm.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 alarm.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     O .bss   00000004 alarm_st
00000004 g     O .bss   00000001 alarm_st_id
00000000 g     F .text  0000002c alarm_init
00000118 g     F .text  00000028 st_alarm_waiting
0000002c g     F .text  0000002c start_alarm
00000084 g     F .text  00000050 st_alarm_start
00000058 g     F .text  0000002c stop_alarm
000000d4 g     F .text  00000044 st_alarm_stop
00000000         *UND*  00000000 Set_Alarm_actuator
00000000         *UND*  00000000 Delay



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded syst
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h alarm.o

alarm.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000140  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000174  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000005  00000000  00000000  00000174  2**2
                  ALLOC
  3 .comment      0000004a  00000000  00000000  00000174  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002a  00000000  00000000  000001be  2**0
                  CONTENTS, READONLY
```

```
$ arm-none-eabi-gcc.exe -c  -I .-mcpu=cortex-m3 startup.c -o startup.o

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gi
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -t startup.o

startup.o:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 startup.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .vectors       00000000 .vectors
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .ARM.attributes        00000000 .ARM.attributes
00000000 g     F .text  0000001c Default_handler
0000001c g     F .text  00000020 Rest_Handler
00000000  w    F .text  0000001c Usage_Fault_Handler
00000000  w    F .text  0000001c Bus_Handler
00000000  w    F .text  0000001c MM_Handler
00000000  w    F .text  0000001c H_Fault_Handler
00000000  w    F .text  0000001c NMI_Handler
00000000 g     O .vectors       0000001c vectors
00000000         *UND*  00000000 stack_top
0000003c g     F .text  000000f8 init_sect
00000000         *UND*  00000000 main
00000000         *UND*  00000000 _E_DATA
00000000         *UND*  00000000 _S_DATA
00000000         *UND*  00000000 _E_text
00000000         *UND*  00000000 _E_BSS
00000000         *UND*  00000000 _S_BSS



Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gi
nit5_First_term_progucts/First_Project (main)
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000134  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000168  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000168  2**0
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  00000168  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .comment      0000004a  00000000  00000000  00000184  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 0000002a  00000000  00000000  000001ce  2**0
                  CONTENTS, READONLY
```

```
Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Mastering-Embedded-System-online-/1-First_Term/U
nit5_First_term_progucts/First_Project (main)
$ make all
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . alarm.c -o alarm.o
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . alarm_monitor.c -o alarm_monitor.o
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . detect.c -o detect.o
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . driver.c -o driver.o
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . main.c -o main.o
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . pressure_sensor.c -o pressure_sensor.o
arm-none-eabi-gcc.exe -c  -g -gdwarf-2 -mcpu=cortex-m3 -I . startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld  alarm.o alarm_monitor.o detect.o driver.o main.o pressure_sensor.o startup.o  -o pressure
_detector.elf -Map=output.map
arm-none-eabi-objcopy.exe -O binary pressure_detector.elf pressure_detector.bin
>>>>>>>>>>>>>Duild is Done<<<<<<<<<<<<<
```

```
Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Mastering-Embedde
$ arm-none-eabi-objdump.exe -t pressure_detector.elf >>pressure_detector_SYMBOLES.TXT

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Mastering-Embedde
$ arm-none-eabi-objdump.exe -S pressure_detector.elf >>pressure_detector_ASEMBLY.TXT

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Mastering-Embedde
$ arm-none-eabi-objdump.exe -h pressure_detector.elf >>pressure_detector_HEADERS.TXT

Egypt_Laptop@DESKTOP-P9O9616 MINGW64 /d/Communication/02-embeded system/Gitup_Repo/Mastering-Embedde
$
```

Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements

Mastering Embedded System Online Diploma (KS)

www.learn-in-depth.com

FIrst Term Project 1

Eng: Ahmed Hassan Kotp