▶ Pressure Detection System

Mastering Embedded Systems Diploma

Presented By :

Arsany Ashraf Mounir

Presented To :

Eng. Keroles Shenouda

TABLE OF CONTENTS

01 Case Study

02 Methodolgy

03 Requirements

04 Space Exploration

05 System Analysis

06 System Design

07 Hardware simulation

08 Source Code

09 Software Analysis

01 Case Study

The jet cabinet pressure detection system is required to pressure ranges at different set points then actuate an alarm. The alarm consists of three LEDs. If the pressure sensor detected any range of pressure values below 10 bars, the green LED will be activated And if the pressure values in between 10-20 bars, the yellow LED will be activated . Lastly, if the pressure values are above 20 bars, the red LED will be activated. During the operation of the system the values read from the pressure sensor has to be stored in flash memory.

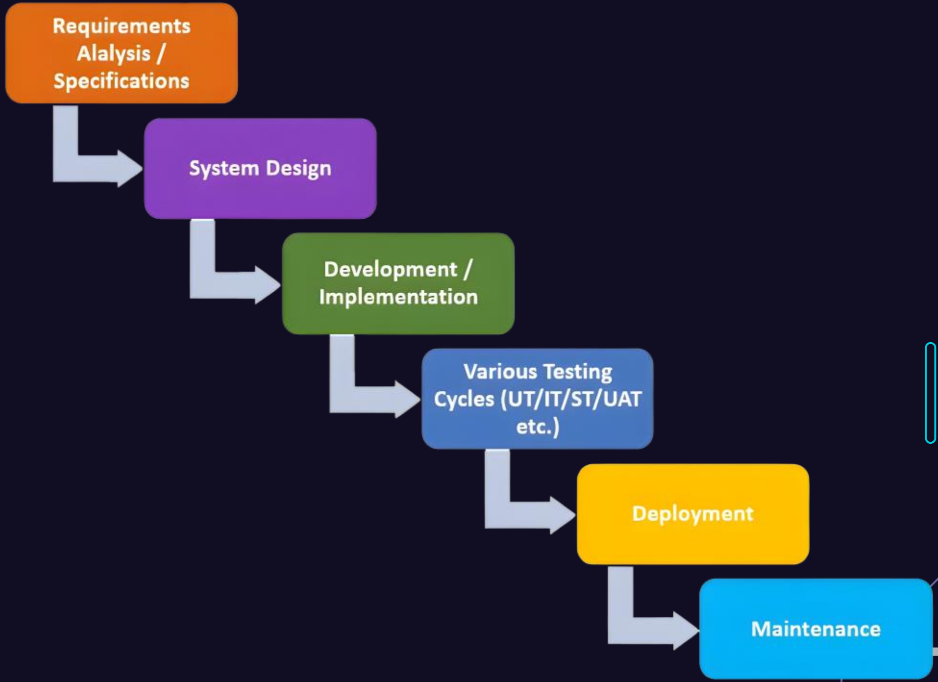
01 Case Study

There are some assumptions about the system that was discussed with the client which resulted in the following points :

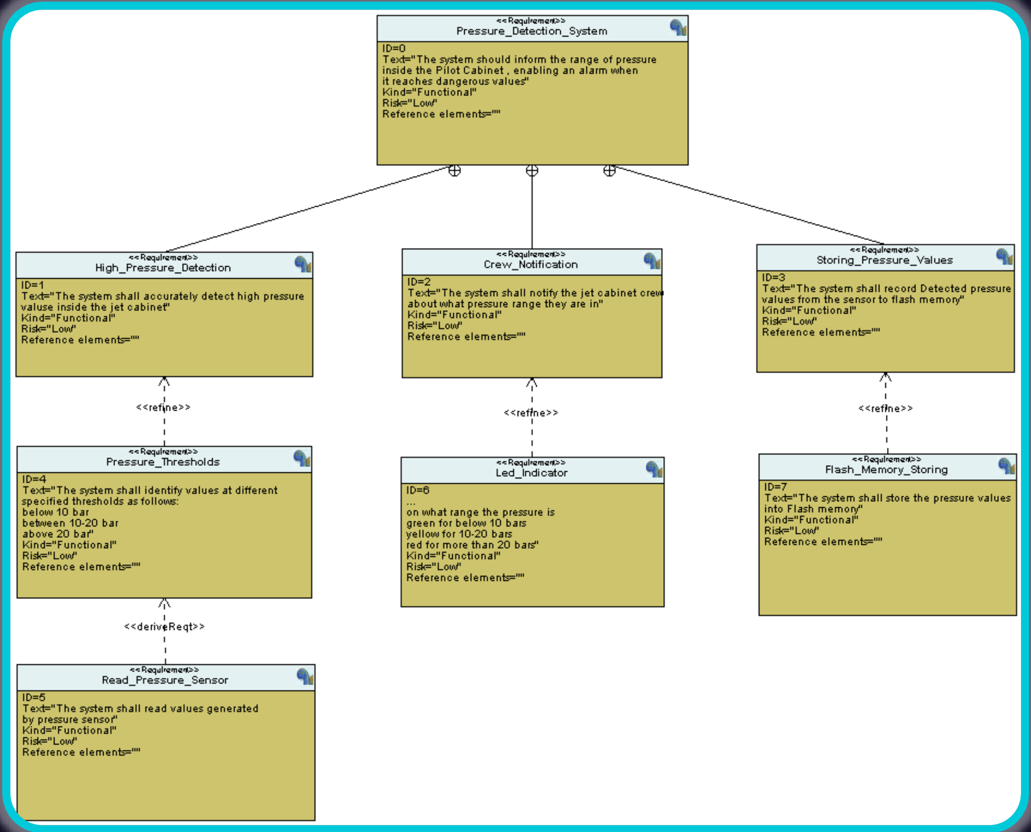
- Controller set up and shutdown procedures aren’t modeled
- Controller maintenance is not
- modeled
- Pressure sensor used never fails
- Alarm never fails
- No power loss

02 Methodolgy

Since the system has multiple modules that will communicate there are multiple methods available to develop this project , Waterfall method is chosen in our case .


Requirements Alalysis / Specifications
System Design
Development / Implementation
Various Testing Cycles (UT/IT/ST/UAT etc.)
Deployment
Maintenance

03 Requirments



<<Requirement>>
Pressure_Detection_System
ID=0
Text="The system should inform the range of pressure
inside the Pilot Cabinet , enabling an alarm when
it reaches dangerous values"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
High_Pressure_Detection
ID=1
Text="The system shall accurately detect high pressure
valuse inside the jet cabinet"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Crew_Notification
ID=2
Text="The system shall notify the jet cabinet crew
about what pressure range they are in"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Storing_Pressure_Values
ID=3
Text="The system shall record Detected pressure
values from the sensor to flash memory"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Pressure_Thresholds
ID=4
Text="The system shall identify values at different
specified thresholds as follows:
below 10 bar
between 10-20 bar
above 20 bar"
Kind="Functional"
Risk="Low"
Reference elements=""

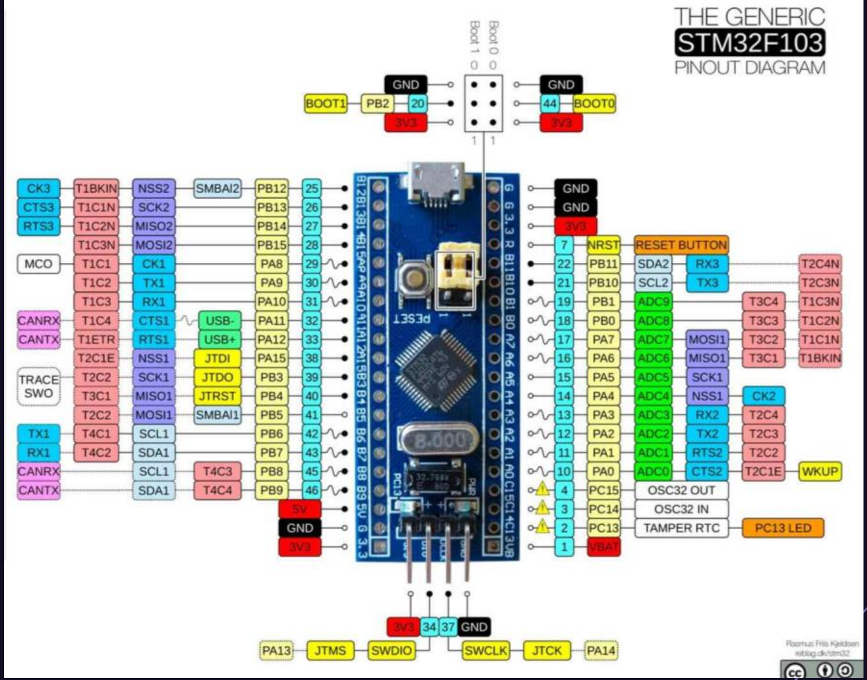
<<Requirement>>
Led_Indicator
ID=6
...
on what range the pressure is
green for below 10 bars
yellow for 10-20 bars
red for more than 20 bars"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Flash_Memory_Storing
ID=7
Text="The system shall store the pressure values
into Flash memory"
Kind="Functional"
Risk="Low"
Reference elements=""

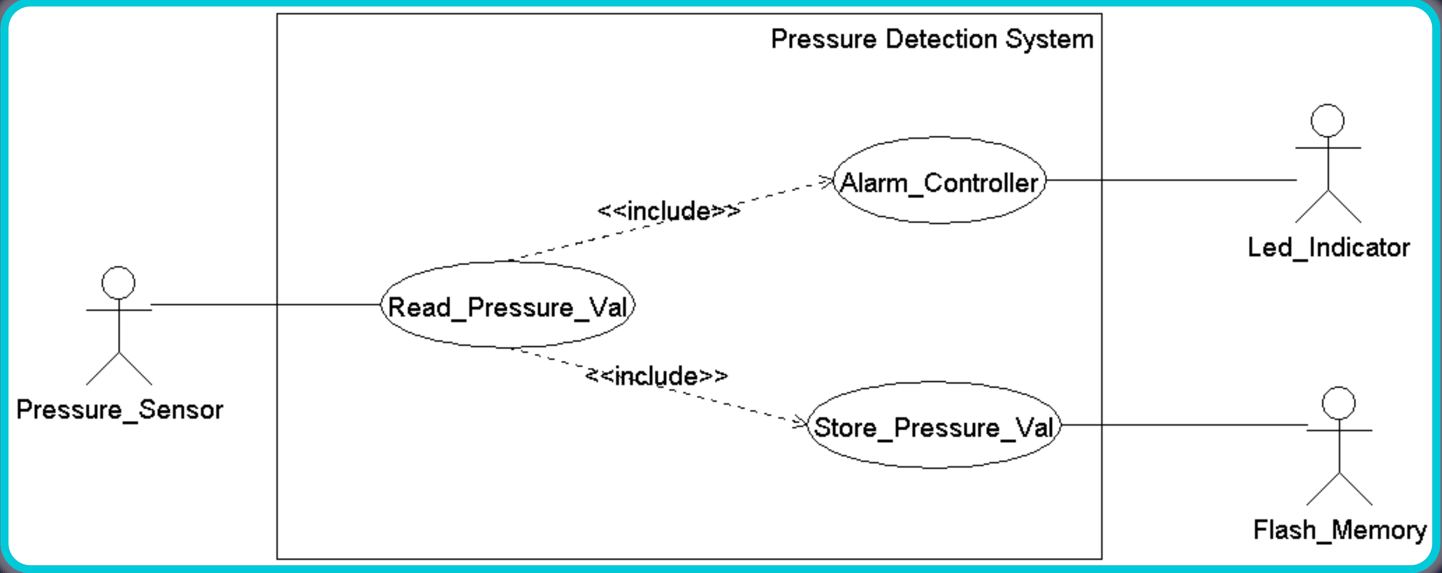
<<Requirement>>
Read_Pressure_Sensor
ID=5
Text="The system shall read values generated
by pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

04 Space Exploration

STM32F103xx SoC is more than enough for the desired application, with ARM Cortex-M3 32bit core operating at 72 MHz. If cost was considered a factor in this project another SoC would be chosen for cost efficiency .



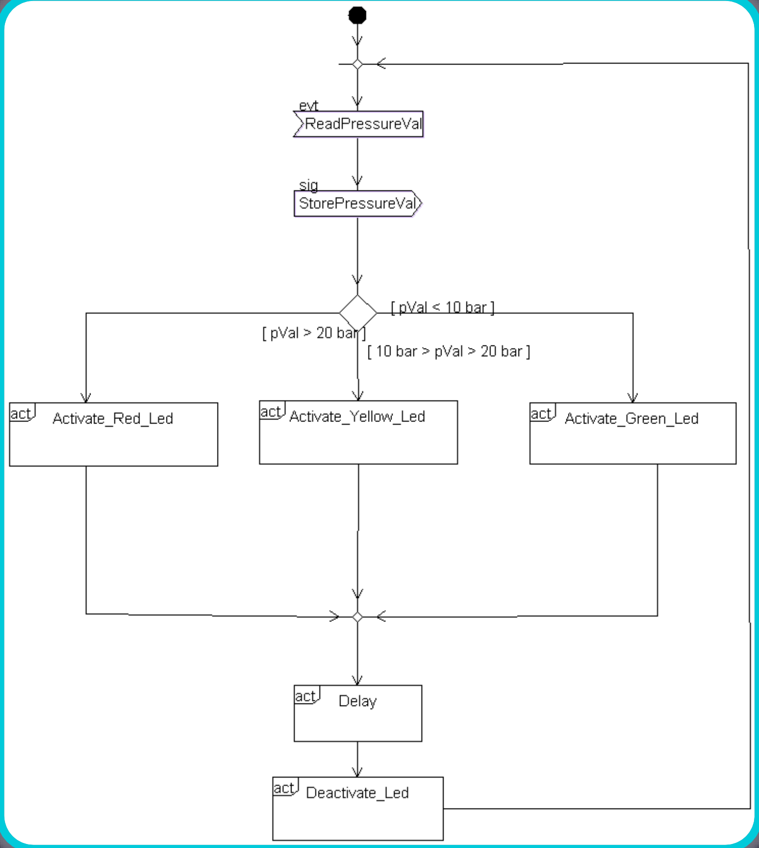
05 System Analysis



Use Case Diagram

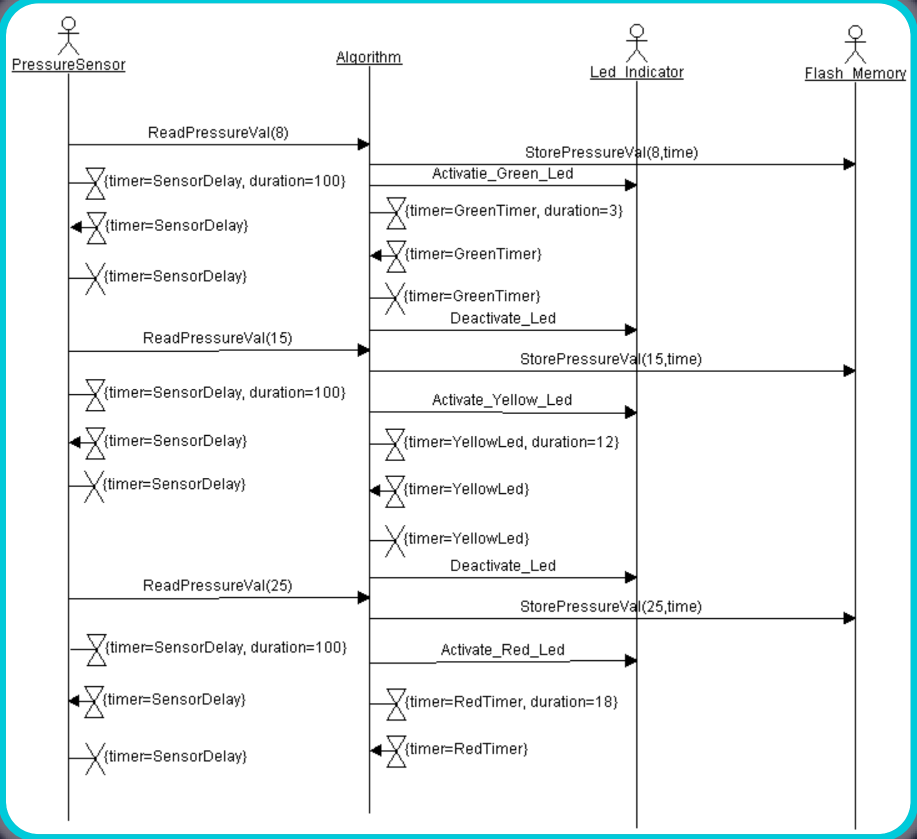
05 System Analysis

Activity Diagram

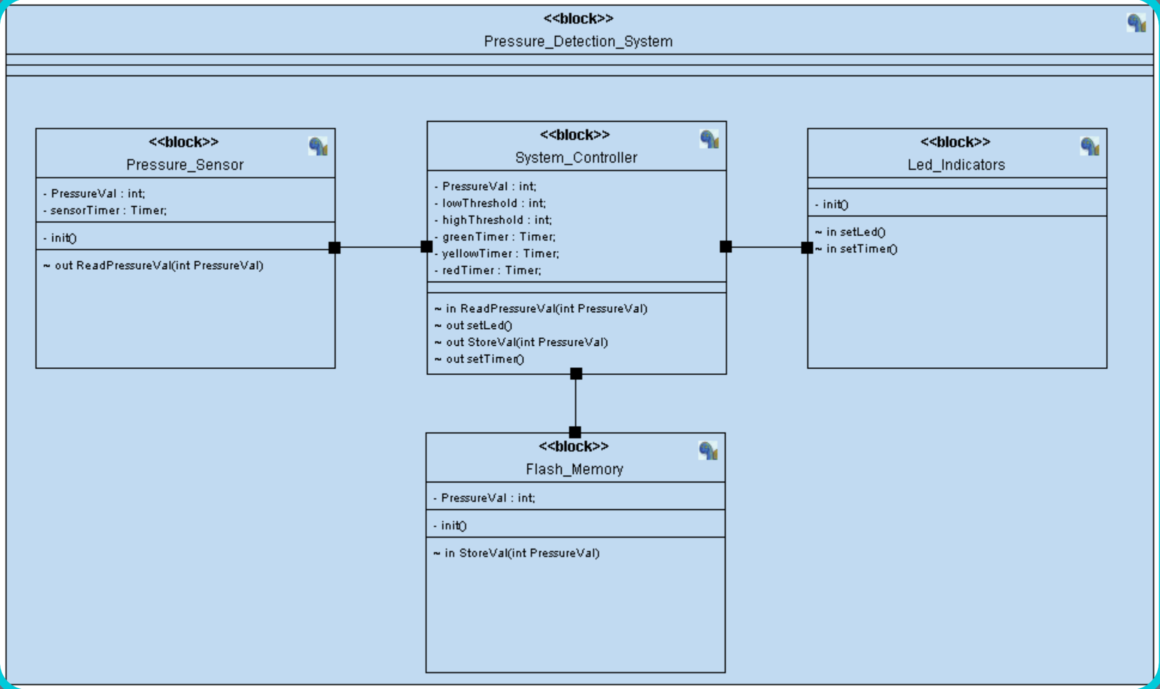


05 System Analysis

Sequence Diagram



06 System Design



<<block>>
Pressure_Detection_System

<<block>>
Pressure_Sensor

- PressureVal : int;
- sensorTimer : Timer;

- init()

~ out ReadPressureVal(int PressureVal)

<<block>>
System_Controller

- PressureVal : int;
- lowThreshold : int;
- highThreshold : int;
- greenTimer : Timer;
- yellowTimer : Timer;
- redTimer : Timer;

~ in ReadPressureVal(int PressureVal)
~ out setLed()
~ out StoreVal(int PressureVal)
~ out setTimer()

<<block>>
Led_Indicators

- init()

~ in setLed()
~ in setTimer()

<<block>>
Flash_Memory

- PressureVal : int;

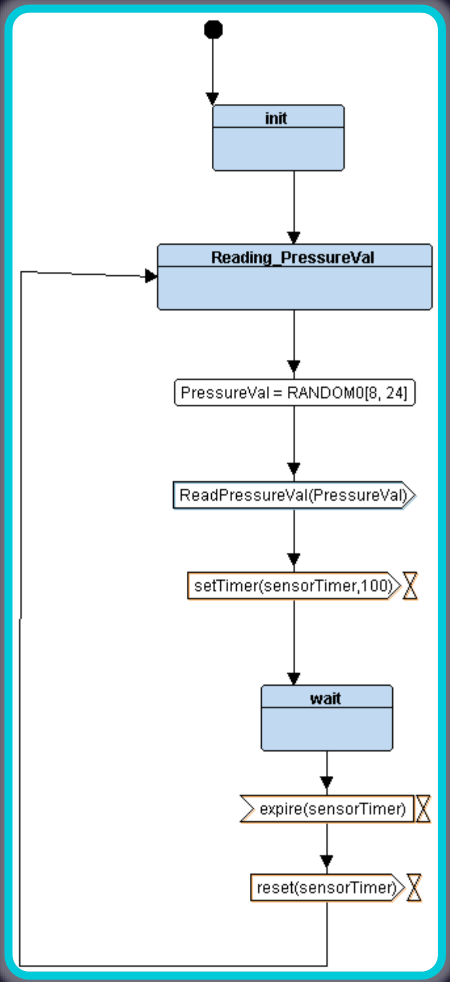
- init()

~ in StoreVal(int PressureVal)

Block Diagram

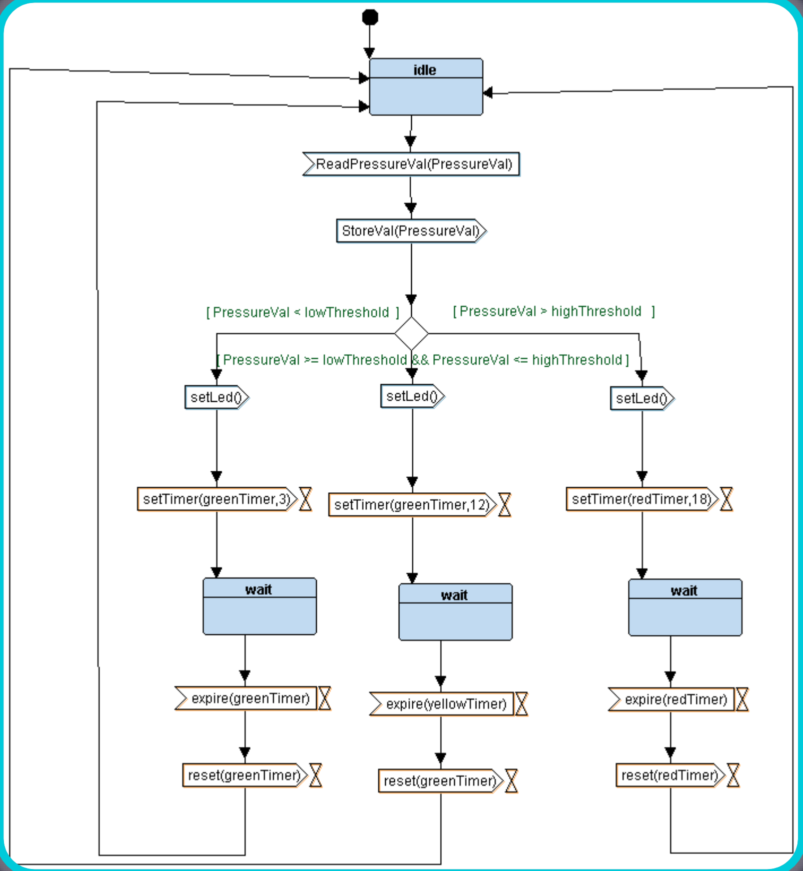
06 System Design

State Machine Diagram (Pressure_Sensor)


init
Reading_PressureVal
PressureVal = RANDOM0[8, 24]
ReadPressureVal(PressureVal)
setTimer(sensorTimer,100)
wait
expire(sensorTimer)
reset(sensorTimer)

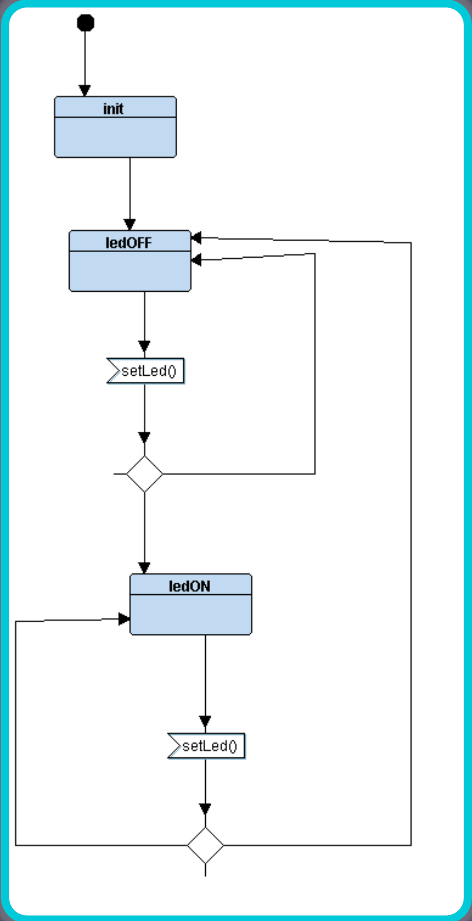
06 System Design

State Machine Diagram (System_Controller)



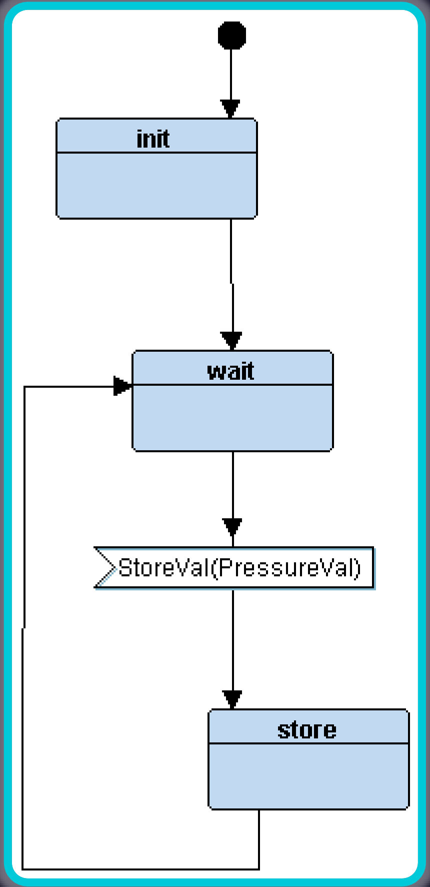
06 System Design

State Machine Diagram (Led_Indicators)



06 System Design

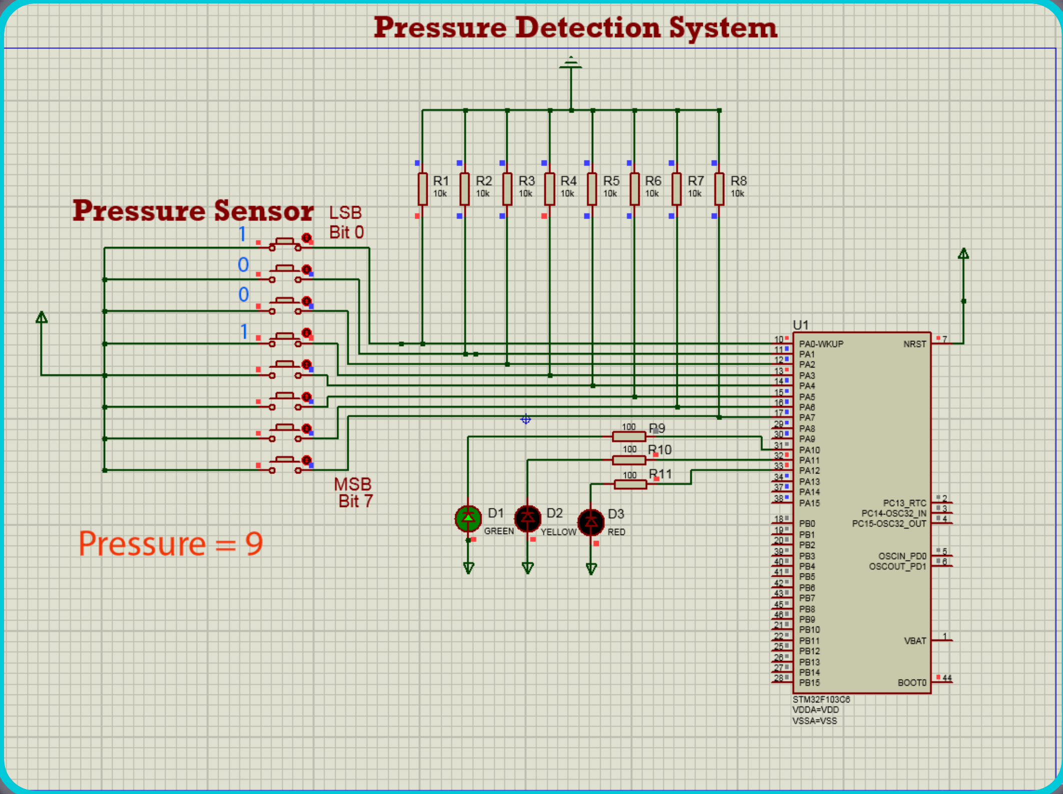
State Machine Diagram (Flash_Memory)



07 Hardware Simulation

1st case

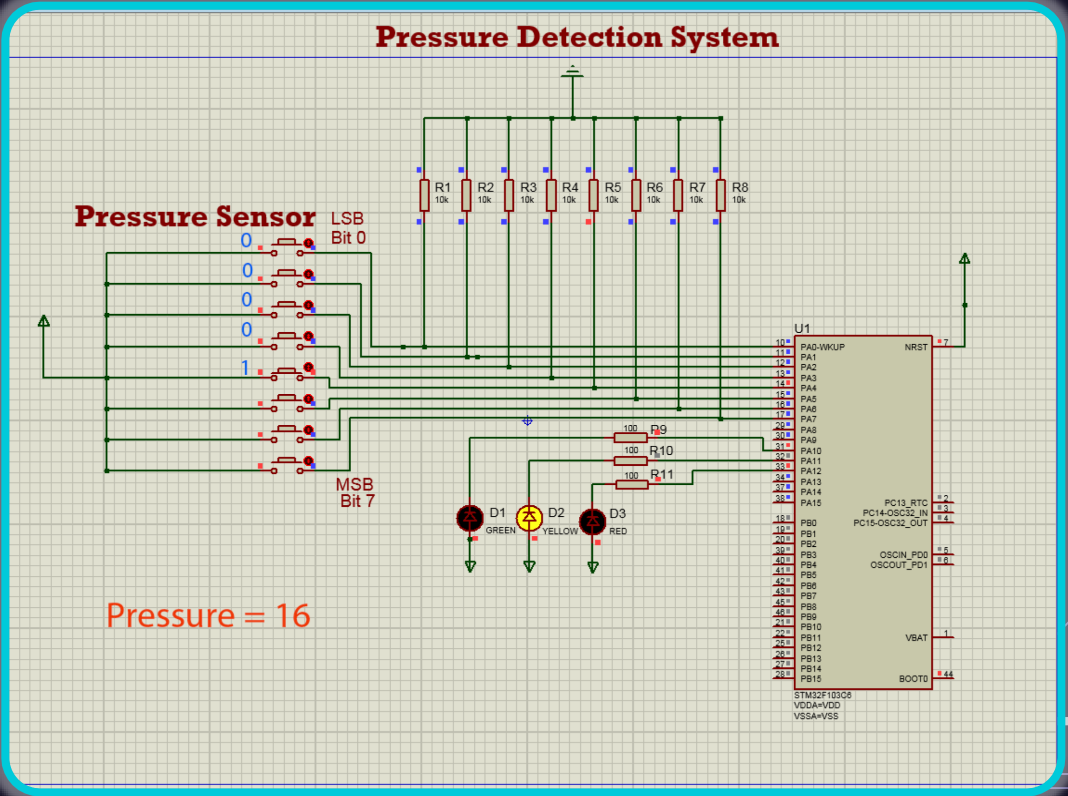
Pressure = 9



Pressure Detection System

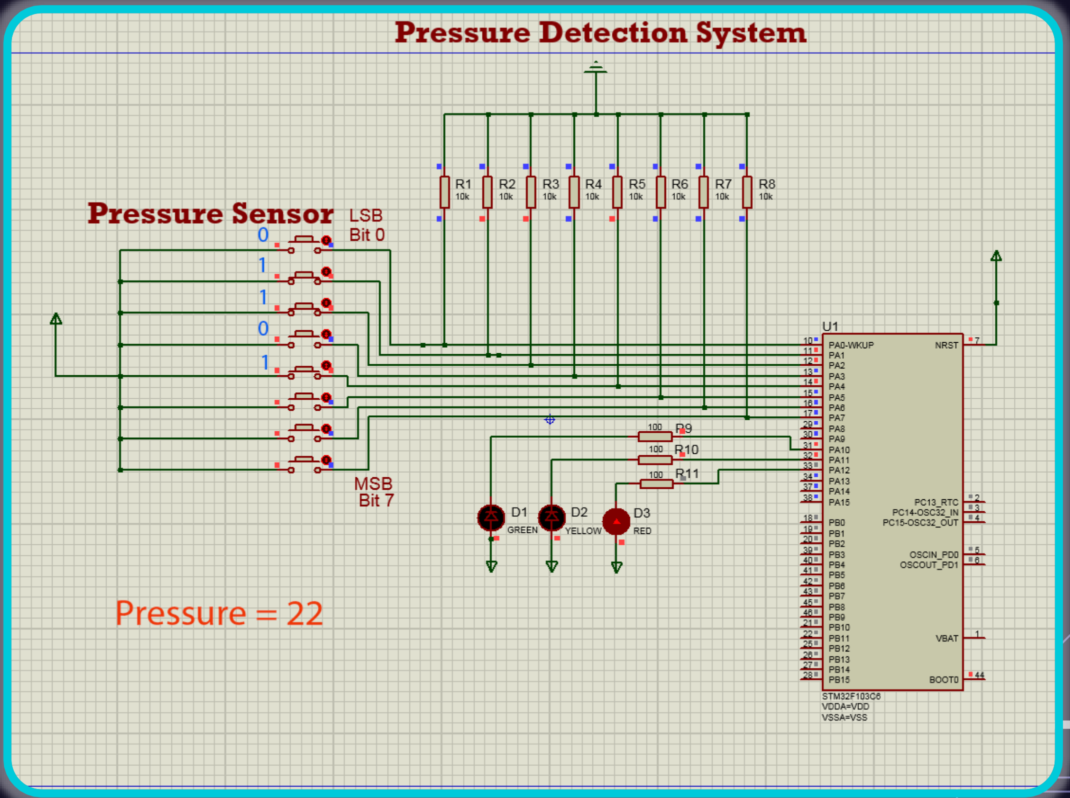
07 Hardware Simulation

2nd case
Pressure = 16



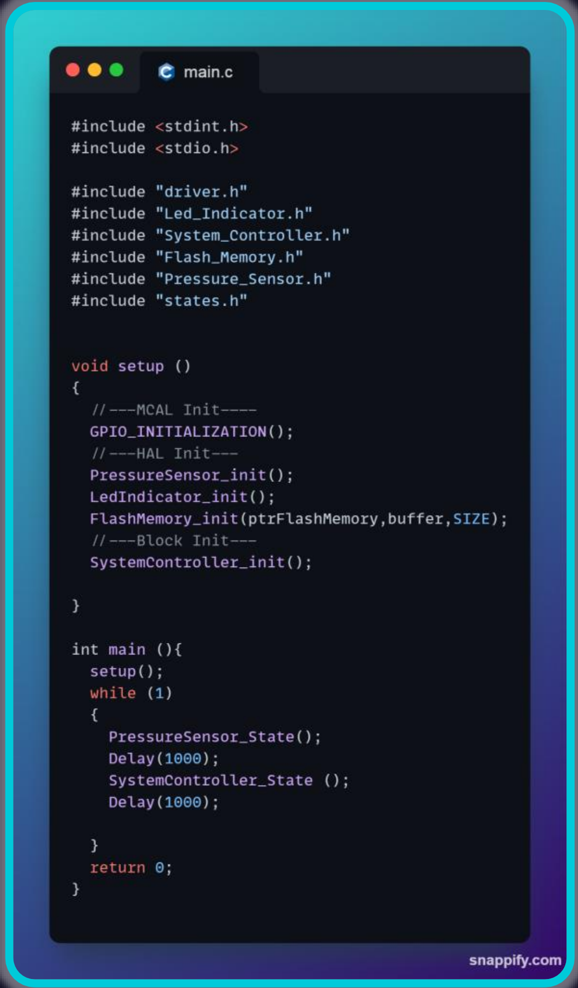
07 Hardware Simulation

3rd case
Pressure = 22



08 Source Code

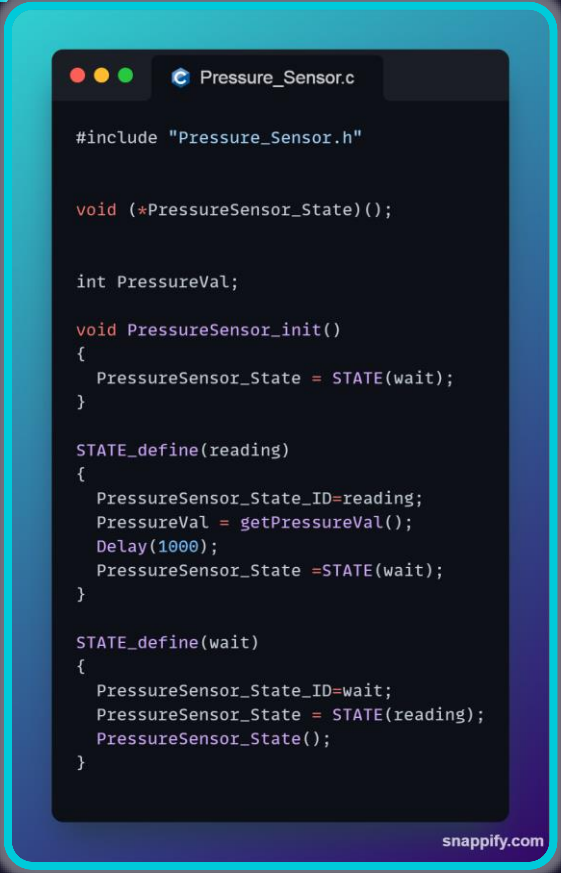
Main .c



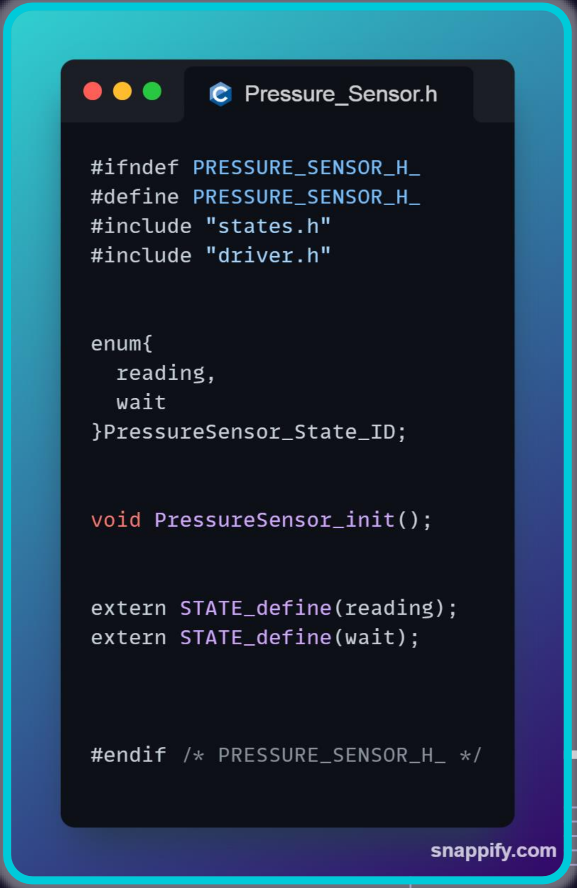
```c
#include <stdint.h>
#include <stdio.h>

#include "driver.h"
#include "Led_Indicator.h"
#include "System_Controller.h"
#include "Flash_Memory.h"
#include "Pressure_Sensor.h"
#include "states.h"


void setup ()
{
  //---MCAL Init----
  GPIO_INITIALIZATION();
  //---HAL Init---
  PressureSensor_init();
  LedIndicator_init();
  FlashMemory_init(ptrFlashMemory,buffer,SIZE);
  //---Block Init---
  SystemController_init();

}

int main (){
  setup();
  while (1)
  {
    PressureSensor_State();
    Delay(1000);
    SystemController_State ();
    Delay(1000);

  }
  return 0;
}
```

08 Source Code

PressureSensor



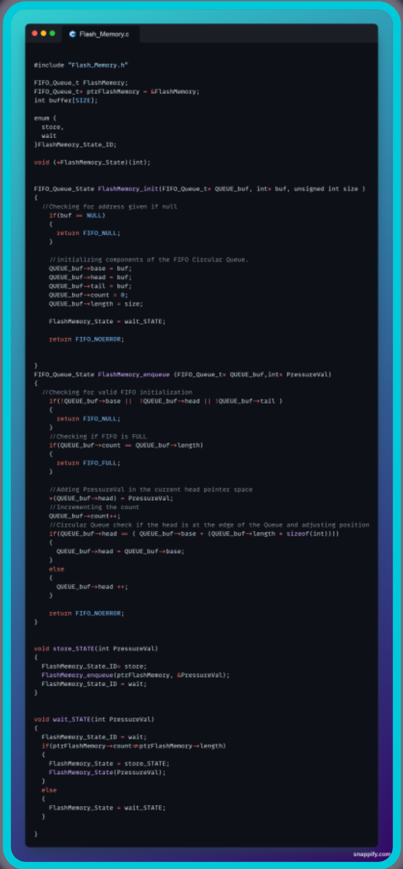
```c
#include "Pressure_Sensor.h"


void (*PressureSensor_State)();


int PressureVal;

void PressureSensor_init()
{
  PressureSensor_State = STATE(wait);
}

STATE_define(reading)
{
  PressureSensor_State_ID=reading;
  PressureVal = getPressureVal();
  Delay(1000);
  PressureSensor_State =STATE(wait);
}

STATE_define(wait)
{
  PressureSensor_State_ID=wait;
  PressureSensor_State = STATE(reading);
  PressureSensor_State();
}
```



```c
#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_
#include "states.h"
#include "driver.h"


enum{
  reading,
  wait
}PressureSensor_State_ID;


void PressureSensor_init();


extern STATE_define(reading);
extern STATE_define(wait);



#endif /* PRESSURE_SENSOR_H_ */
```

08 Source Code

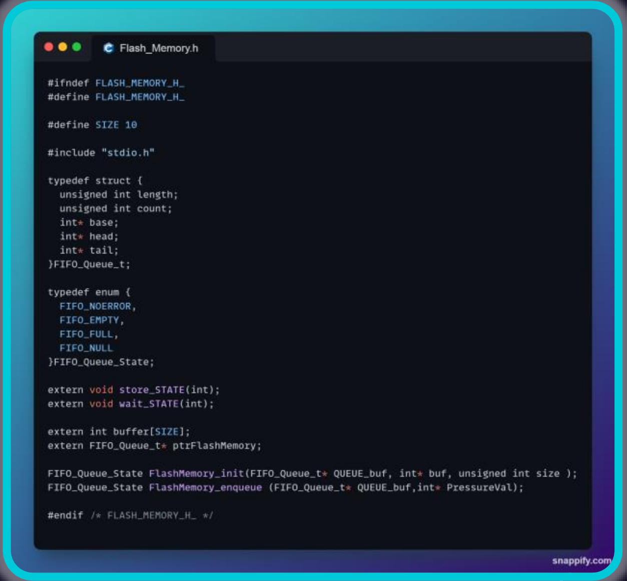
FlashMemory



Flash_Memory.c

```c
#include "Flash_Memory.h"

FIFO_Queue_t FlashMemory;
FIFO_Queue_t* ptrFlashMemory = &FlashMemory;
int buffer[SIZE];

enum {
  store,
  wait
}FlashMemory_State_ID;

void (*FlashMemory_State)(int);


FIFO_Queue_State FlashMemory_init(FIFO_Queue_t* QUEUE_buf, int* buf, unsigned int size )
{
  //Checking for address given if null
    if(buf == NULL)
    {
      return FIFO_NULL;
    }

    //initializing components of the FIFO Circular Queue.
    QUEUE_buf->base = buf;
    QUEUE_buf->head = buf;
    QUEUE_buf->tail = buf;
    QUEUE_buf->count = 0;
    QUEUE_buf->length = size;

    FlashMemory_State = wait_STATE;

    return FIFO_NOERROR;


}
FIFO_Queue_State FlashMemory_enqueue (FIFO_Queue_t* QUEUE_buf,int* PressureVal)
{
  //Checking for valid FIFO initialization
    if(!QUEUE_buf->base ||  !QUEUE_buf->head || !QUEUE_buf->tail )
    {
      return FIFO_NULL;
    }
    //Checking if FIFO is FULL
    if(QUEUE_buf->count == QUEUE_buf->length)
    {
      return FIFO_FULL;
    }

    //Adding PressureVal in the current head pointer space
    *(QUEUE_buf->head) = PressureVal;
    //Incrementing the count
    QUEUE_buf->count++;
    //Circular Queue check if the head is at the edge of the Queue and adjusting position
    if(QUEUE_buf->head == ( QUEUE_buf->base + (QUEUE_buf->length * sizeof(int))))
    {
      QUEUE_buf->head = QUEUE_buf->base;
    }
    else
    {
      QUEUE_buf->head ++;
    }

    return FIFO_NOERROR;
}


void store_STATE(int PressureVal)
{
  FlashMemory_State_ID= store;
  FlashMemory_enqueue(ptrFlashMemory, &PressureVal);
  FlashMemory_State_ID = wait;
}


void wait_STATE(int PressureVal)
{
  FlashMemory_State_ID = wait;
  if(ptrFlashMemory->count!=ptrFlashMemory->length)
  {
    FlashMemory_State = store_STATE;
    FlashMemory_State(PressureVal);
  }
  else
  {
    FlashMemory_State = wait_STATE;
  }

}
```

snappify.com



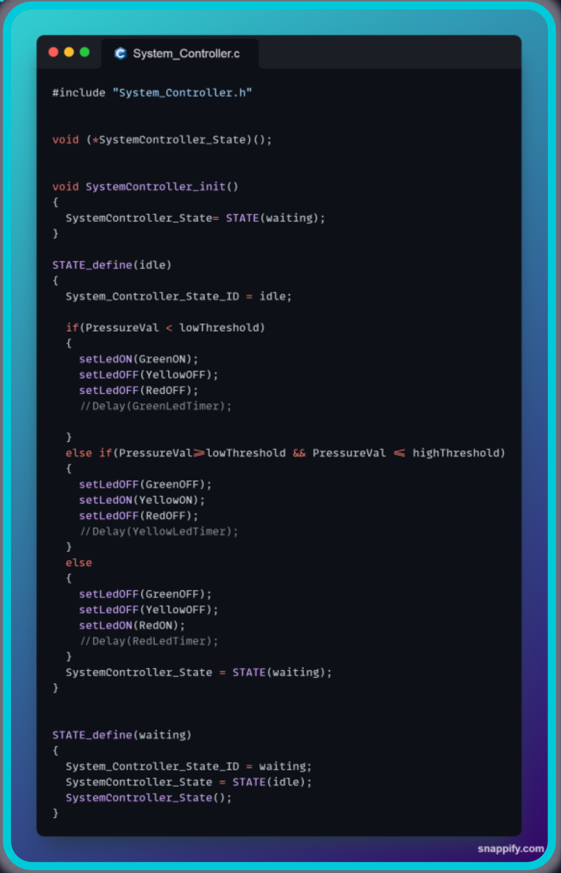
Flash_Memory.h

```c
#ifndef FLASH_MEMORY_H_
#define FLASH_MEMORY_H_

#define SIZE 10

#include "stdio.h"

typedef struct {
  unsigned int length;
  unsigned int count;
  int* base;
  int* head;
  int* tail;
}FIFO_Queue_t;

typedef enum {
  FIFO_NOERROR,
  FIFO_EMPTY,
  FIFO_FULL,
  FIFO_NULL
}FIFO_Queue_State;

extern void store_STATE(int);
extern void wait_STATE(int);

extern int buffer[SIZE];
extern FIFO_Queue_t* ptrFlashMemory;

FIFO_Queue_State FlashMemory_init(FIFO_Queue_t* QUEUE_buf, int* buf, unsigned int size );
FIFO_Queue_State FlashMemory_enqueue (FIFO_Queue_t* QUEUE_buf,int* PressureVal);

#endif /* FLASH_MEMORY_H_ */
```

snappify.com

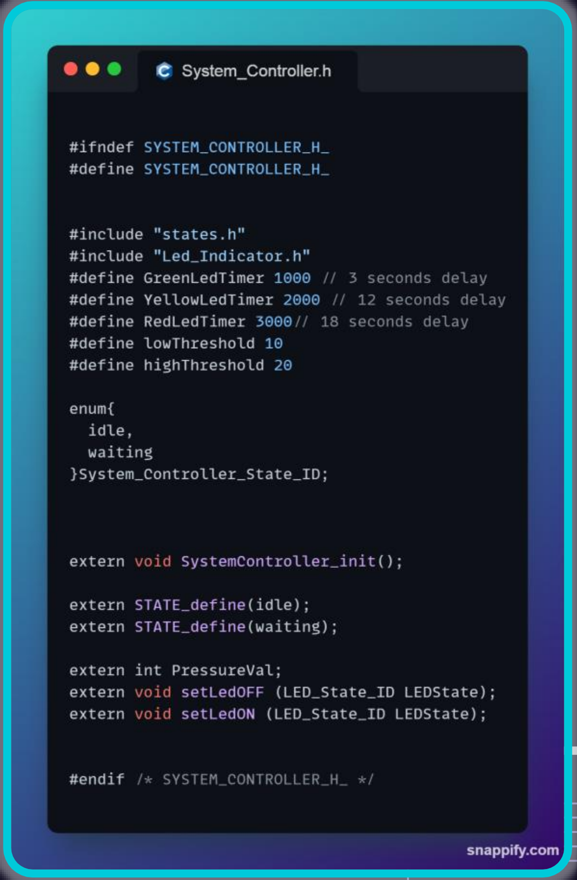
08 Source Code

SystemController



System_Controller.c

```c
#include "System_Controller.h"


void (*SystemController_State)();


void SystemController_init()
{
  SystemController_State= STATE(waiting);
}

STATE_define(idle)
{
  System_Controller_State_ID = idle;

  if(PressureVal < lowThreshold)
  {
    setLedON(GreenON);
    setLedOFF(YellowOFF);
    setLedOFF(RedOFF);
    //Delay(GreenLedTimer);

  }
  else if(PressureVal>=lowThreshold && PressureVal <= highThreshold)
  {
    setLedOFF(GreenOFF);
    setLedON(YellowON);
    setLedOFF(RedOFF);
    //Delay(YellowLedTimer);
  }
  else
  {
    setLedOFF(GreenOFF);
    setLedOFF(YellowOFF);
    setLedON(RedON);
    //Delay(RedLedTimer);
  }
  SystemController_State = STATE(waiting);
}


STATE_define(waiting)
{
  System_Controller_State_ID = waiting;
  SystemController_State = STATE(idle);
  SystemController_State();
}
```

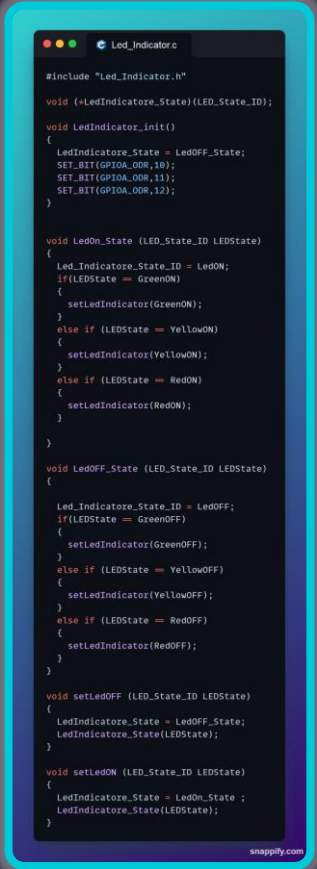


System_Controller.h

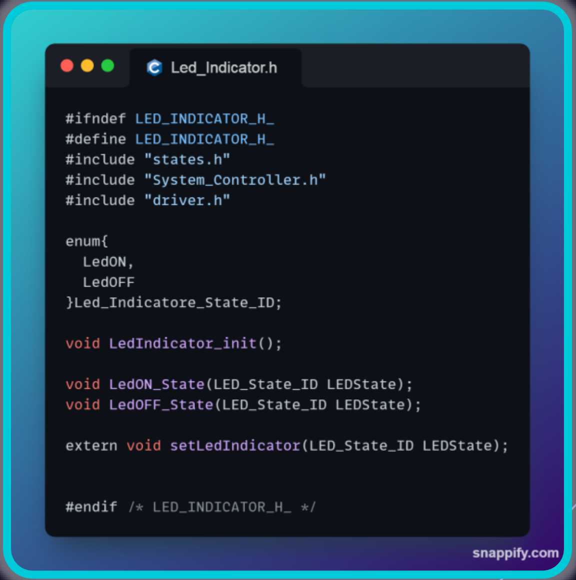
```c

#ifndef SYSTEM_CONTROLLER_H_
#define SYSTEM_CONTROLLER_H_


#include "states.h"
#include "Led_Indicator.h"
#define GreenLedTimer 1000 // 3 seconds delay
#define YellowLedTimer 2000 // 12 seconds delay
#define RedLedTimer 3000// 18 seconds delay
#define lowThreshold 10
#define highThreshold 20

enum{
  idle,
  waiting
}System_Controller_State_ID;



extern void SystemController_init();

extern STATE_define(idle);
extern STATE_define(waiting);

extern int PressureVal;
extern void setLedOFF (LED_State_ID LEDState);
extern void setLedON (LED_State_ID LEDState);


#endif /* SYSTEM_CONTROLLER_H_ */
```

08 Source Code

LedIndicator



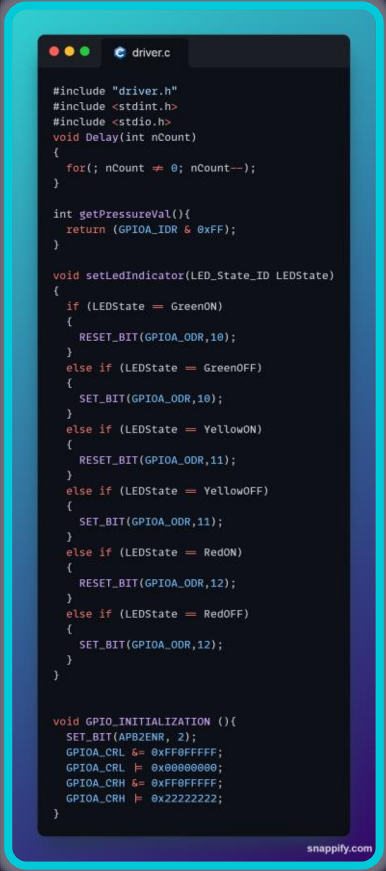
```c
#include "Led_Indicator.h"

void (*LedIndicatore_State)(LED_State_ID);

void LedIndicator_init()
{
  LedIndicatore_State = LedOFF_State;
  SET_BIT(GPIOA_ODR,10);
  SET_BIT(GPIOA_ODR,11);
  SET_BIT(GPIOA_ODR,12);
}


void LedOn_State (LED_State_ID LEDState)
{
  Led_Indicatore_State_ID = LedON;
  if(LEDState == GreenON)
  {
    setLedIndicator(GreenON);
  }
  else if (LEDState == YellowON)
  {
    setLedIndicator(YellowON);
  }
  else if (LEDState == RedON)
  {
    setLedIndicator(RedON);
  }

}

void LedOFF_State (LED_State_ID LEDState)
{

  Led_Indicatore_State_ID = LedOFF;
  if(LEDState == GreenOFF)
  {
    setLedIndicator(GreenOFF);
  }
  else if (LEDState == YellowOFF)
  {
    setLedIndicator(YellowOFF);
  }
  else if (LEDState == RedOFF)
  {
    setLedIndicator(RedOFF);
  }
}

void setLedOFF (LED_State_ID LEDState)
{
  LedIndicatore_State = LedOFF_State;
  LedIndicatore_State(LEDState);
}

void setLedON (LED_State_ID LEDState)
{
  LedIndicatore_State = LedOn_State ;
  LedIndicatore_State(LEDState);
}
```



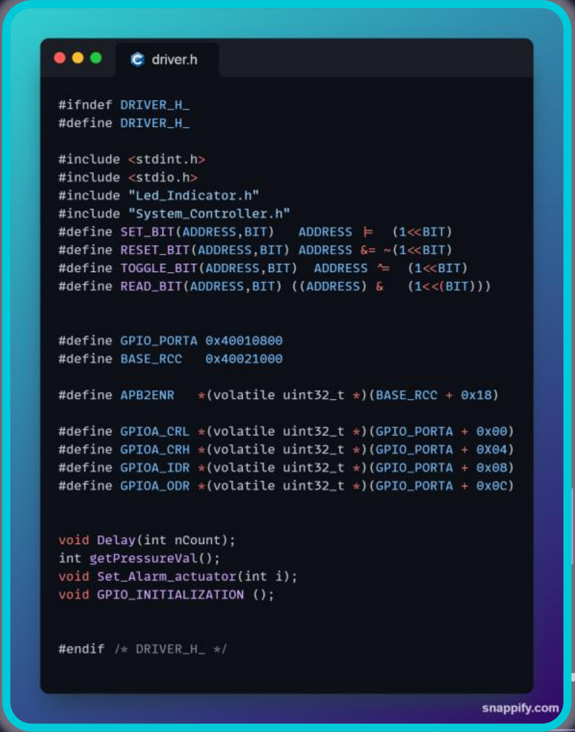
```c
#ifndef LED_INDICATOR_H_
#define LED_INDICATOR_H_
#include "states.h"
#include "System_Controller.h"
#include "driver.h"

enum{
  LedON,
  LedOFF
}Led_Indicatore_State_ID;

void LedIndicator_init();

void LedON_State(LED_State_ID LEDState);
void LedOFF_State(LED_State_ID LEDState);

extern void setLedIndicator(LED_State_ID LEDState);


#endif /* LED_INDICATOR_H_ */
```

08 Source Code

Drivers



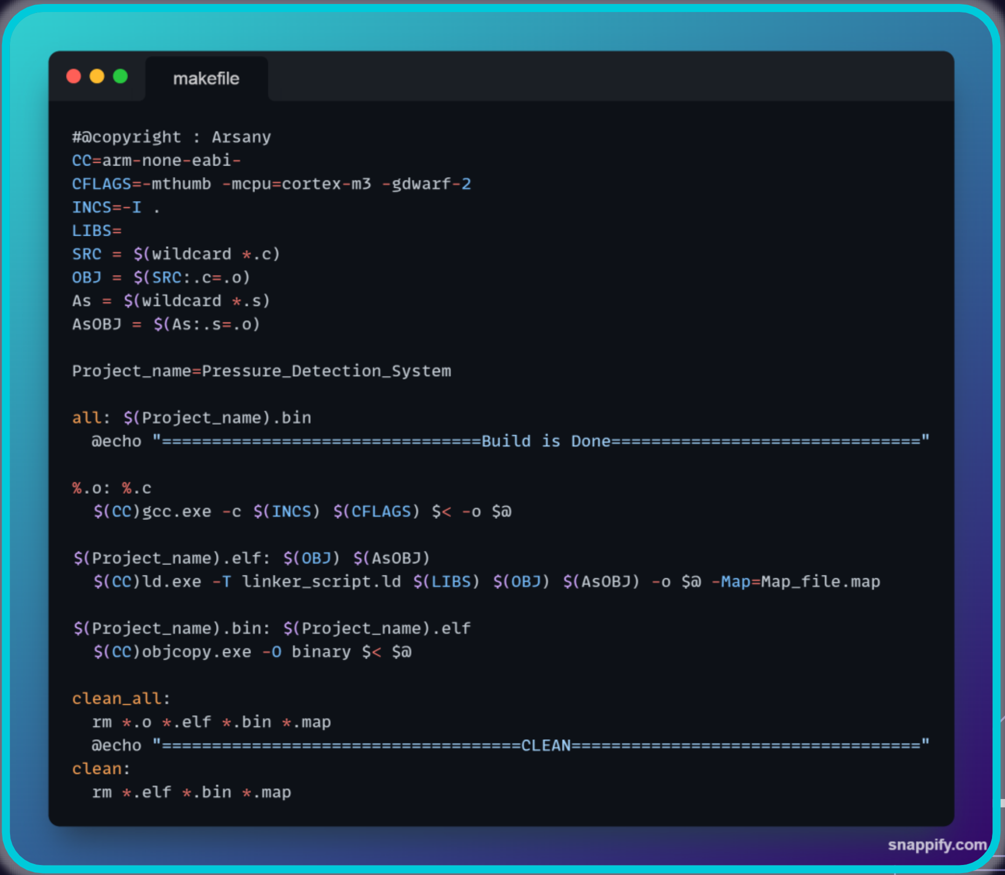
```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>
void Delay(int nCount)
{
  for(; nCount != 0; nCount--);
}

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

void setLedIndicator(LED_State_ID LEDState)
{
  if (LEDState == GreenON)
  {
    RESET_BIT(GPIOA_ODR,10);
  }
  else if (LEDState == GreenOFF)
  {
    SET_BIT(GPIOA_ODR,10);
  }
  else if (LEDState == YellowON)
  {
    RESET_BIT(GPIOA_ODR,11);
  }
  else if (LEDState == YellowOFF)
  {
    SET_BIT(GPIOA_ODR,11);
  }
  else if (LEDState == RedON)
  {
    RESET_BIT(GPIOA_ODR,12);
  }
  else if (LEDState == RedOFF)
  {
    SET_BIT(GPIOA_ODR,12);
  }
}


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



```c
#ifndef DRIVER_H_
#define DRIVER_H_

#include <stdint.h>
#include <stdio.h>
#include "Led_Indicator.h"
#include "System_Controller.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 ();


#endif /* DRIVER_H_ */
```

08 Source Code

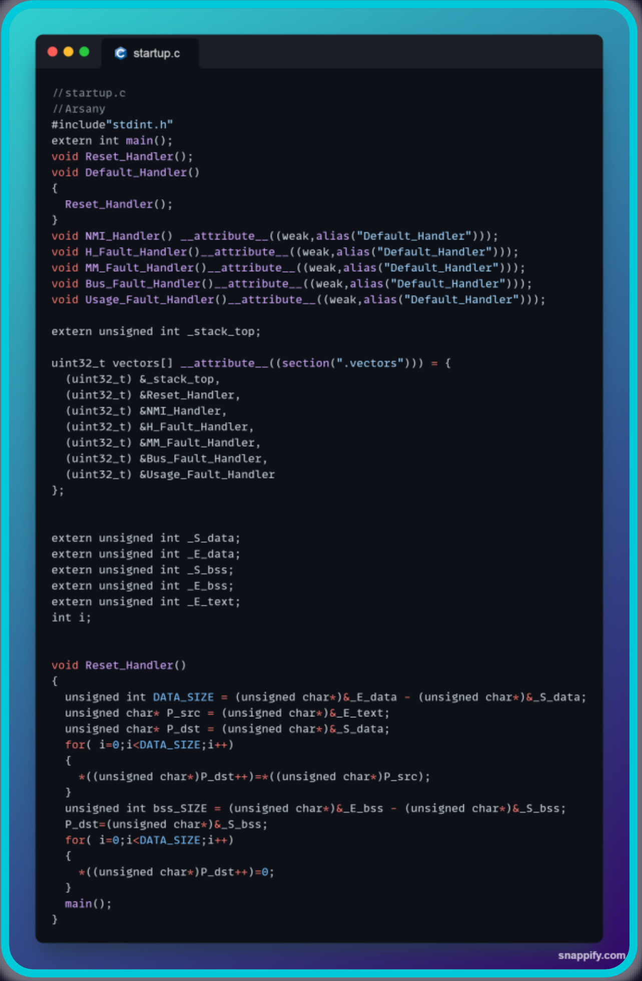
makefile



```makefile
#@copyright : Arsany
CC=arm-none-eabi-
CFLAGS=-mthumb -mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
AsOBJ = $(As:.s=.o)

Project_name=Pressure_Detection_System

all: $(Project_name).bin
  @echo "================================Build is Done==============================="

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

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

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

clean_all:
  rm *.o *.elf *.bin *.map
  @echo "====================================CLEAN==================================="
clean:
  rm *.elf *.bin *.map
```

08 Source Code

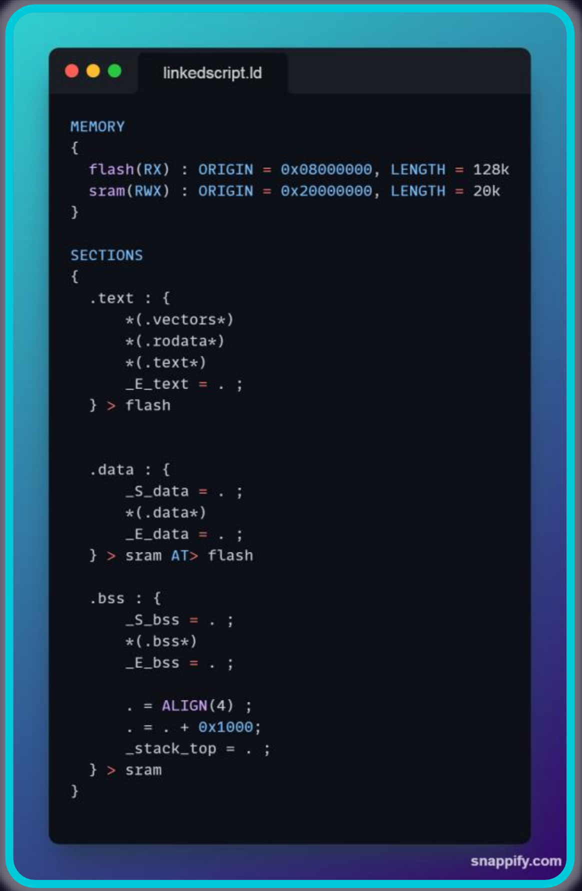
Startup



```c
//startup.c
//Arsany
#include"stdint.h"
extern int main();
void Reset_Handler();
void Default_Handler()
{
  Reset_Handler();
}
void NMI_Handler() __attribute__((weak,alias("Default_Handler")));
void H_Fault_Handler()__attribute__((weak,alias("Default_Handler")));
void MM_Fault_Handler()__attribute__((weak,alias("Default_Handler")));
void Bus_Fault_Handler()__attribute__((weak,alias("Default_Handler")));
void Usage_Fault_Handler()__attribute__((weak,alias("Default_Handler")));

extern unsigned int _stack_top;

uint32_t vectors[] __attribute__((section(".vectors"))) = {
  (uint32_t) &_stack_top,
  (uint32_t) &Reset_Handler,
  (uint32_t) &NMI_Handler,
  (uint32_t) &H_Fault_Handler,
  (uint32_t) &MM_Fault_Handler,
  (uint32_t) &Bus_Fault_Handler,
  (uint32_t) &Usage_Fault_Handler
};


extern unsigned int _S_data;
extern unsigned int _E_data;
extern unsigned int _S_bss;
extern unsigned int _E_bss;
extern unsigned int _E_text;
int i;


void Reset_Handler()
{
  unsigned int DATA_SIZE = (unsigned char*)&_E_data - (unsigned char*)&_S_data;
  unsigned char* P_src = (unsigned char*)&_E_text;
  unsigned char* P_dst = (unsigned char*)&_S_data;
  for( i=0;i<DATA_SIZE;i++)
  {
    *((unsigned char*)P_dst++)=*((unsigned char*)P_src);
  }
  unsigned int bss_SIZE = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
  P_dst=(unsigned char*)&_S_bss;
  for( i=0;i<DATA_SIZE;i++)
  {
    *((unsigned char*)P_dst++)=0;
  }
  main();
}
```

08 Source Code

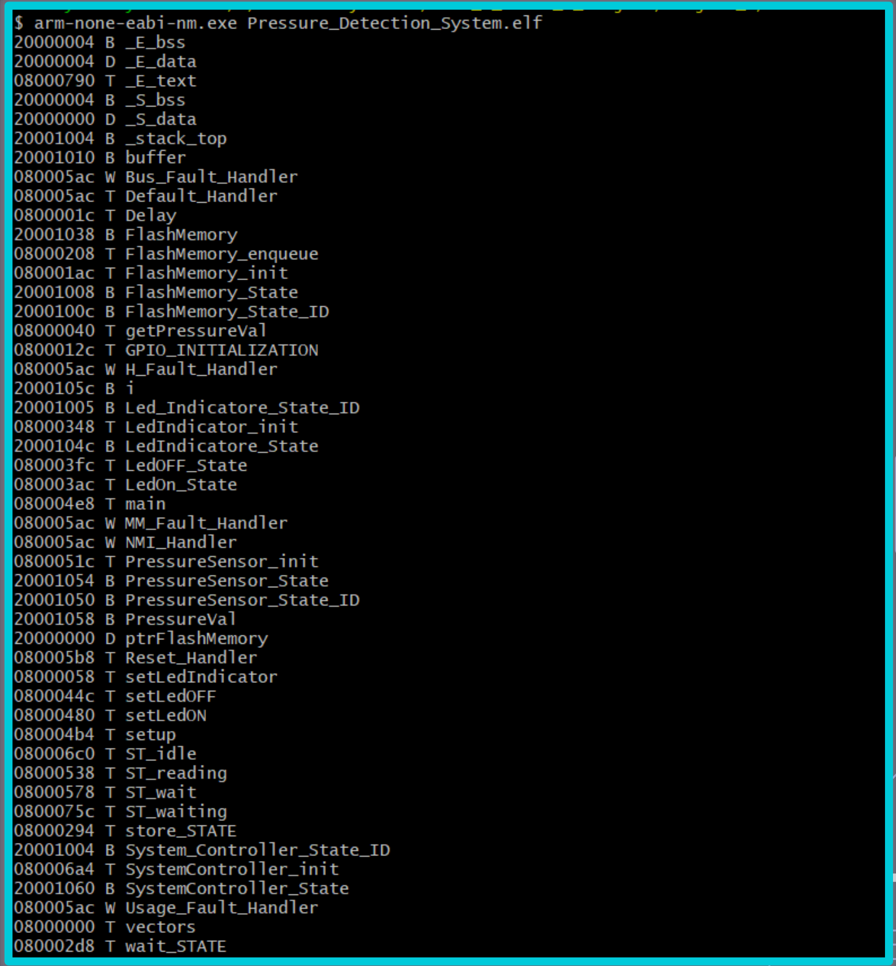
Linked Script



```
MEMORY
{
  flash(RX) : ORIGIN = 0x08000000, LENGTH = 128k
  sram(RWX) : ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{
  .text : {
      *(.vectors*)
      *(.rodata*)
      *(.text*)
      _E_text = . ;
  } > flash


  .data : {
      _S_data = . ;
      *(.data*)
      _E_data = . ;
  } > sram AT> flash

  .bss : {
      _S_bss = . ;
      *(.bss*)
      _E_bss = . ;

      . = ALIGN(4) ;
      . = . + 0x1000;
      _stack_top = . ;
  } > sram
}
```

09 Software Analysis

.elf Symbol table



```
$ arm-none-eabi-nm.exe Pressure_Detection_System.elf
20000004 B _E_bss
20000004 D _E_data
08000790 T _E_text
20000004 B _S_bss
20000000 D _S_data
20001004 B _stack_top
20001010 B buffer
080005ac W Bus_Fault_Handler
080005ac T Default_Handler
0800001c T Delay
20001038 B FlashMemory
08000208 T FlashMemory_enqueue
080001ac T FlashMemory_init
20001008 B FlashMemory_State
2000100c B FlashMemory_State_ID
08000040 T getPressureVal
0800012c T GPIO_INITIALIZATION
080005ac W H_Fault_Handler
2000105c B i
20001005 B Led_Indicatore_State_ID
08000348 T LedIndicator_init
2000104c B LedIndicatore_State
080003fc T LedOFF_State
080003ac T LedOn_State
080004e8 T main
080005ac W MM_Fault_Handler
080005ac W NMI_Handler
0800051c T PressureSensor_init
20001054 B PressureSensor_State
20001050 B PressureSensor_State_ID
20001058 B PressureVal
20000000 D ptrFlashMemory
080005b8 T Reset_Handler
08000058 T setLedIndicator
0800044c T setLedOFF
08000480 T setLedON
080004b4 T setup
080006c0 T ST_idle
08000538 T ST_reading
08000578 T ST_wait
0800075c T ST_waiting
08000294 T store_STATE
20001004 B System_Controller_State_ID
080006a4 T SystemController_init
20001060 B SystemController_State
080005ac W Usage_Fault_Handler
08000000 T vectors
080002d8 T wait_STATE
```

▶ THANKS!

Do you have any questions?
Arsanyashrafmounir@gmail.com

in