Embedded Controller Lab Report

Design Problem :

Industrial Workspace Safety Automation

Date: 2022. 12. 17

Author/Partner: Dongmin Kim / Jinho Kook

Github: https://github.com/DongminKim21800064/EC_dmkim-064/tree/main/Final

Demo Video: https://www.youtube.com/watch?v=e--00xkY3is

I. Introduction

Design an embedded system to realize a simple smart factory safety system with the following design criteria.

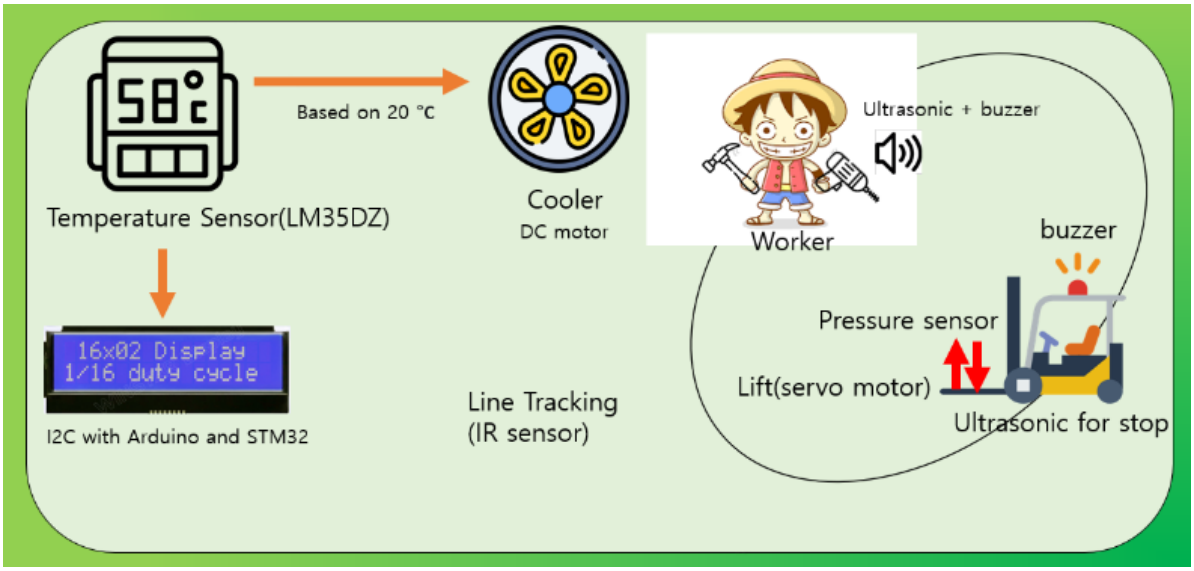
Purpose of Design Problem

(1) Smart factory system to prevent accidents of workers and create a pleasant working environment

(2) Efficient division of tasks through automation.

(3) Systems that can be applied anywhere, regardless of region.

Overview of the Problem



Description

- The forklift stops in front of the worker by line tracing.
- When the worker releases the box, Supervisor controls the forklift in manual mode and transports the box to the target location.
- After that, proceed with line trace again.

- The smart factory also has a temperature control function.
- If the temperature rises above a specific temperature, the cooler fan operates. Beside of cooler fan, Real-time temperature appears on the LCD.

A. Forklift

1A. Line tracing Mode

Mode	Description
Normal Mode	The forklift moves along the designated line. The forklift maintains the path using the IRsensor.
Danger Mode	Recognize workers or objects whose forklift is within 12 cm. When something is in front of the forklift, it stops.
Lift up Mode	When a forklift is loaded, the piezo sensor recognizes it. If the pressure is detected, the lift rises after 2 seconds.
Manual Mode	Press 'E' button on PC to switch to Manual Mode. In Manual Mode, the supervisor can directly move the forklift.

2A. Manual Mode

Mode	Description
Direction Control Mode	Can adjust the direction of the forklift on the PC by pressing the keys 'W', 'A', 'S', and 'D'. When reversing by pressing the 'S' key, the buzzer rings to alert you of danger.
Stop Mode	Press'B ' to stop the forklift
Lift Control Mode	Press 'L', Lift UP, Press 'K', Lift Down
Line tracing Mode	Press 'Q' button on PC to switch to Line tracing Mode again.

B. Cooler Fan & Alarm

Mode	Description
Cooler Fan OFF Mode	Turn Off the Cooler Fan when factory was cool down.
Cooler Fan ON Mode	Turn On the Cooler Fan when factory was HOT.
Danger Alarm Mode	Recognize Forklift which come to close the worker. And buzzer On

Requirement

Hardware

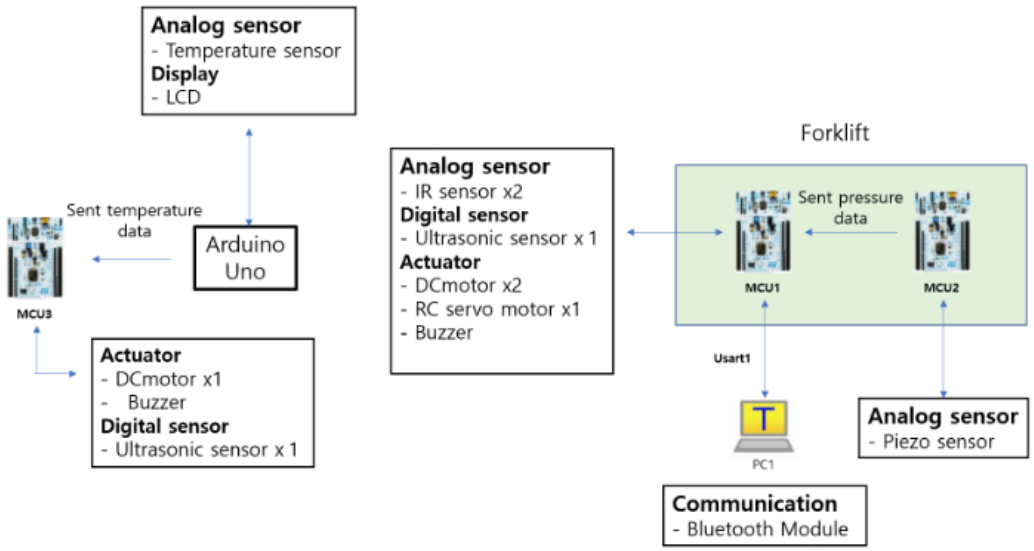
Item	Model/Description	Qty
MCU	NUCLEO-F411RE	3
	Arduino UNO	1
Analog Sensor	Piezo sensor(FSR 406 Solder Tabs)	1
	Temperature sensor(W1209)	1
	IRsensor	2
Digital Sensor	Ultrasonic distance sensor(HC-SR04)	2
Actuator	DC Motor	3
	DC motor driver(L9110s) for RC car	1
	DC motor driver(Packed with RC car module) for Cooler Fan	1
	Buzzer	2
	RC Servo motor for lift	1
Display	I2C 1602 LCD (SZH-EK101)	1
Communication	Bluetooth Module(HC-06)	1
Craft	Forklift Module	1
	Minifan	1
	Acryls for background	6
	Boxes	1

> In the case of Piezo sensor is originally analog, but we tune it to digitally by software .

Software

- Keil uVision, CMSIS, EC_HAL library
- Arduino library

II. Problem Description



A. MCU Detail

- **MCU1**

Function	Sensor/Actuator	Configuration	Comments
Worker detect	Ultrasonic distance sensor	10 us couter step Check object presence within 12cm Genarates Danger_MODE DC motors stop	Since it is a sensor that detects from 2cm to 400cm, do not detect 0-2cm.
Manual Direction control	Bluetooth communication, buzzer, DC motor	The forklift is wirelessly controlled using PC1.	When the forklift move backward, buzzer on
Manual Lift up&down	Bluetooth communication, RC servomotor	The forklift is wirelessly controlled using PC1.	In lift up state, PWM duty is 0.125, lift down state, PWM duty is 0.0972
Line tracing	IR sensor, DC motor	If IR value up to 1000, detect it and feedback to control forklift.	
Lift up	RC servomotor	A digital pressure flag is received from MCU2 to decide whether to turn up the lift.	In lift up state, PWM duty is 0.125

- **MCU2**

Function	Sensor/Actuator	Configuration	Comments
Pressure Detect	Piezo sensor	If it detects more than 100g of an object, it sends a digital signal to MCU1	Use ADC_IRQHandler Set ADC_TRGO

- **MCU3**

Function	Sensor/Actuator	Configuration	Comments
Cool down	Temperature sensor, DC motor	Check the temperature which is up to set_temperature, Cooler Fan on	Need to check for outlier measurements Set +- 2 degree offset for prevent on & off very quickly
Display the Temperature	Temperature sensor, I2C LCD	Using the Arduino UNO library File	
Alert the danger to Worker	Ultrasonic sensor, Buzzer	10 us couter step Check object presence within 12cm Buzzer on	

B. MCU Configuration

- **MCU1**

Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		1ms
Pressure flag	Digital Input	PC0	Read ditial typq pressure flag
Bluetooth	USART1	TXD: PA9	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
		RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
TIMER	TIMER1	PA8	Ultra sonic sensor, For PWM(Trig), 50ms, timer interrupt
		PB0	IR_sensor, ADC_TRGO, 100ms
		PB1	IR_sensor, ADC_TRGO, 100ms
	TIMER2	PA1	Servo Motor AF mode, 20ms
	TIMER3	PC9	DC motor PWM right
		PC8	DC motor PWM left
		PA6	Forklift Direction control
	TIMER4	PB6	Ultra sonic sensor, For input capture(echo), 10us, timer interrupt
		PB8	Forklift Direction control

- **MCU2**

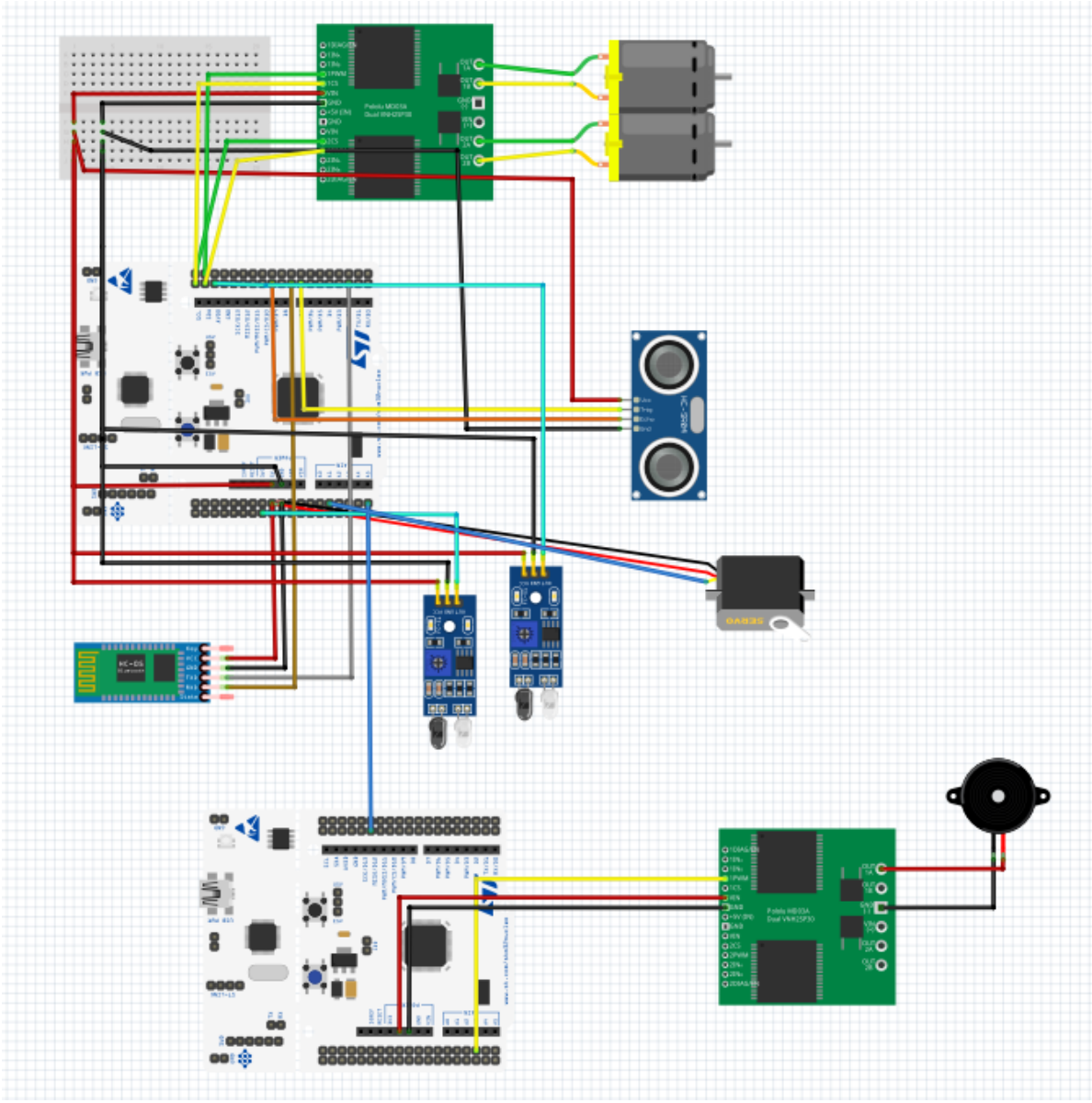
Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		1ms
GPIO_write	Digital out	PA5	To send digital type pressure flag MCU1
Timer	Timer1	PB0	Piezo_sensor, ADC_TRGO, 100ms

- **MCU3**

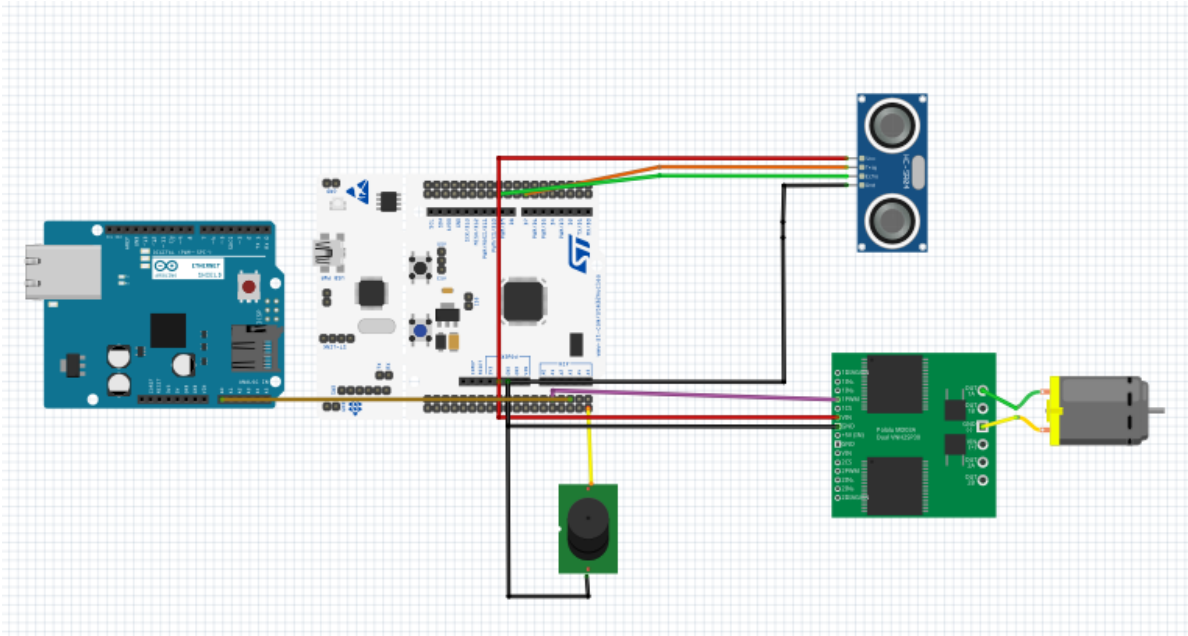
Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		1ms
GPIO_toggle	Digital output	PC3	Buzzer On, toggle 1s
TIMER	TIMER1	PA8	Ultra sonic sensor, For PWM(Trig), 50ms, timer interrupt
	TIMER2	PA1	Cooler DC motor, 20ms
	TIMER4	PB6	Ultra sonic sensor, For input capture(echo), 10us, timer interrupt

C. MCU Wiring Connection

C.1 Forklift Part (MCU1, MCU2)



C.2 CoolerFan and Alarm(MCU3, Arduino uno)



III. Algorithm

State Table

1. Fork Lift

1.1 Auto Mode (Push the 'Q' on the PC)

	Present State	Next State				Output		
		Ultra Sensor F		Ultra Sensor T		Motor	Lift	Buzzer
		Pressure F	Pressure T	Pressure F	Pressure T			
Line tracing	S0	S0	S1	S1	S1	1	0	0
Stop / Lift Down	S1	S0	S2	S1	S2	0	0	0
Stop / Lift Up	S2	S1	S2	S1	S2	0	1	1

- Input : IR sensing Data, Ultra Sonic sensing Data, Piezo sensing Data
- Output : Liftup_flag, Liftdown_flag, Motor_flag, Buzzer_flag

1.2 Manual Mode (Push the 'E' on the PC)

Manual Mode(Push the E)

- It moves according to the value entered by the PC.

Inputs

- Move Fork Lift : W, A, S, D
- Lift Up / Down : L, K

Outputs

- If the forklift is reversing or Lift Up and Down, the buzzer on.

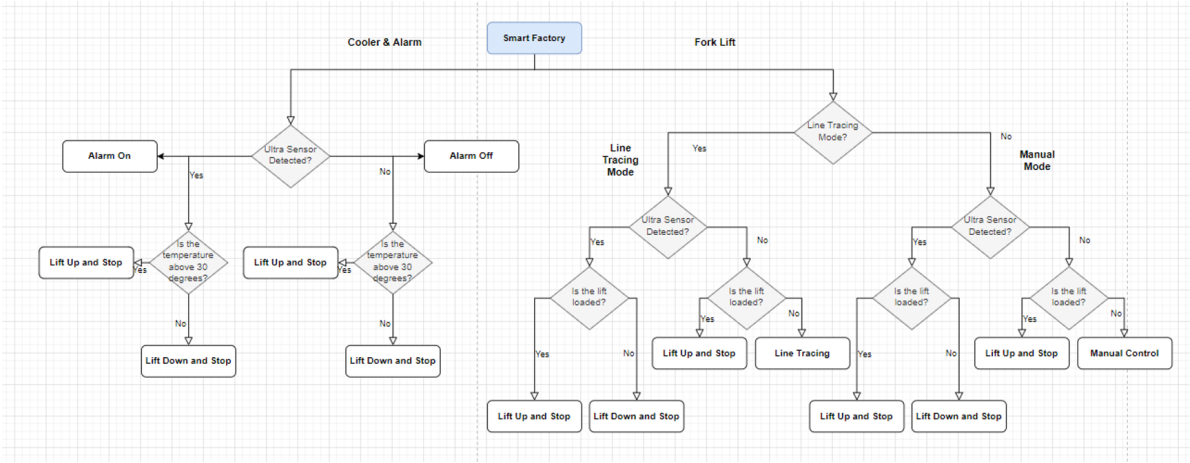
2. Cooler Fan & Caution Alarm

	Present State	Next State				Output	
		Cool		Hot		Cooler Fan	Alarm
		Safe	Danger	Safe	Danger		
Normal	S0	S0	S2	S1	S3	0	0
Cooling	S1	S0	S2	S1	S3	1	0
Alarm	S2	S0	S2	S1	S3	0	1
Cooling & Alarm	S3	S0	S2	S1	S3	1	1

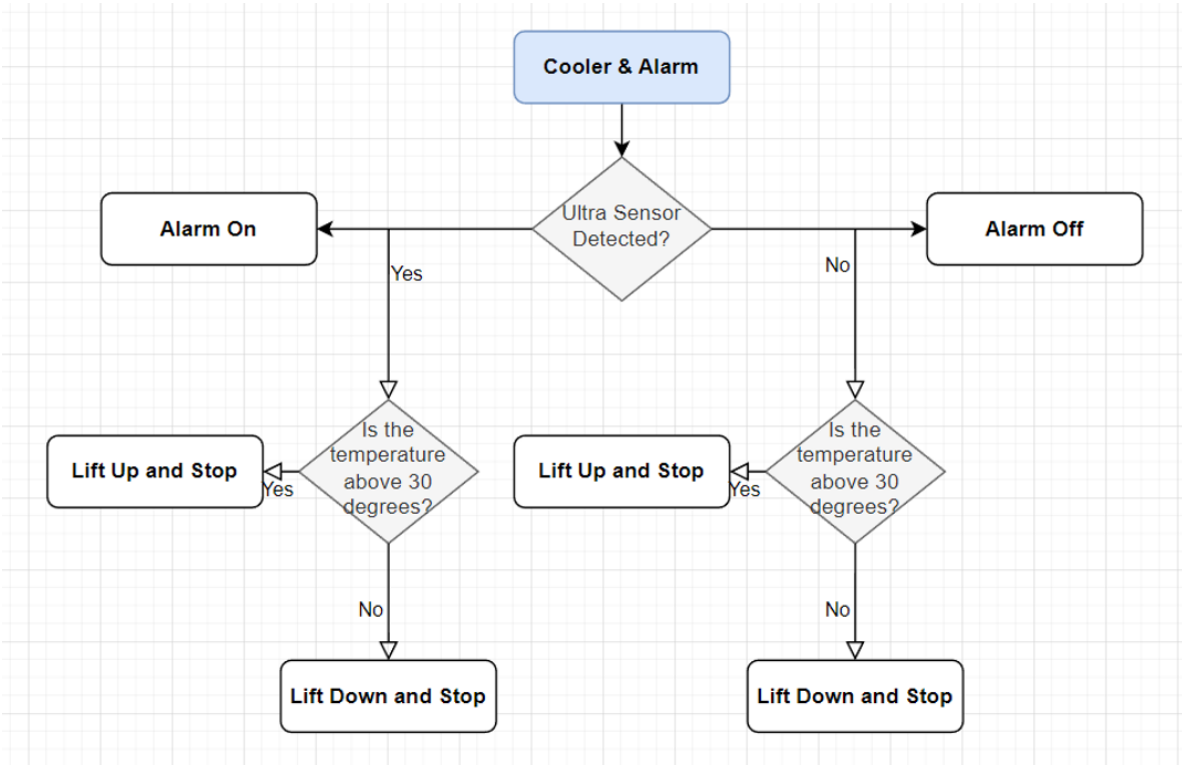
- Input : Temerature sensing Data, Ultra Sonic sensing Data
- Output : Motor_flag, Alarm_flag

Flow Chart

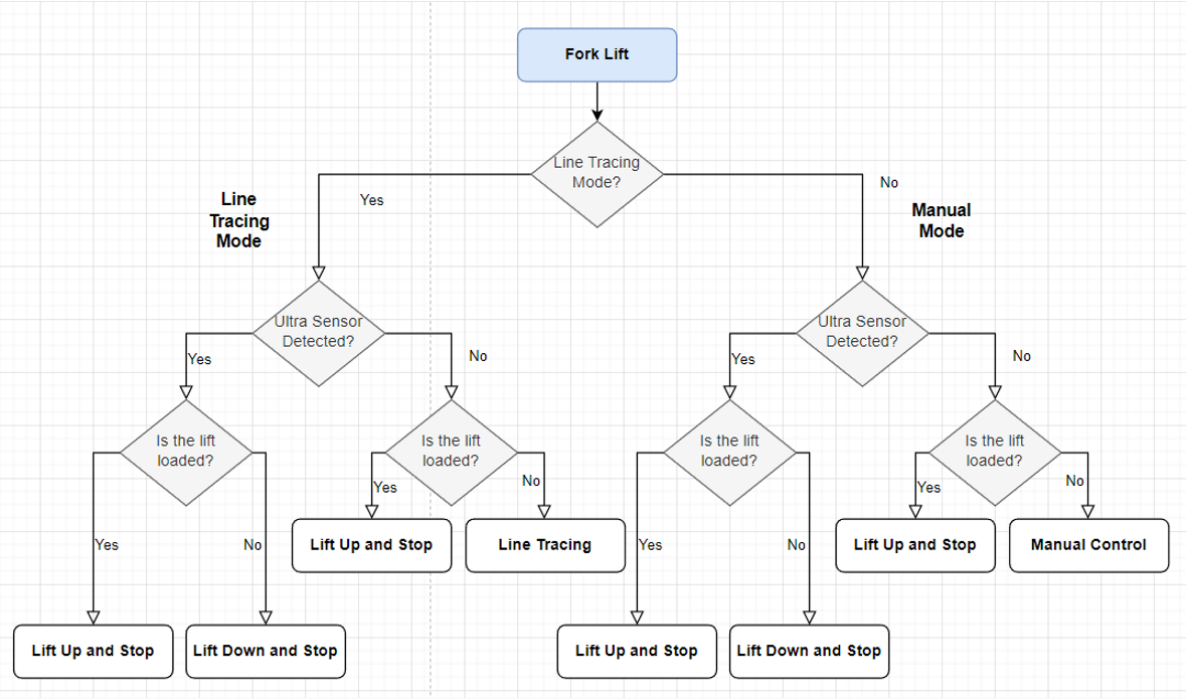
Total Flow chart



Cooler & Alarm



Fork Lift



IV. Demonstration

Scenario Checklist

Scenario	Contents	Confirmation
1	The forklift runs along the line designated as "Line Tracking Mode".	o
2	The forklift senses the worker while driving and stops.	o
3	The alarm goes on so that workers can alert the danger.	o
4	The worker loads the stopped forklift. The pressure sensor on the lift detects this and raises the lift.	o
5	Supervisor changes to 'manual mode' and moves forklift to specific location.	o
6	A forklift backs up to move to a specific location. The buzzer on the forklift signals danger.	o
7	The supervisor unloads at a particular location.	o
8	Supervisor matches the forklift on the line and changes it to "Line Tracking Mode".	o
9	The temperature inside the factory rises above 33 degrees, and the cooler fan operates.	o
10	With the cooler running, the forklift approaches the worker, and the buzzer operates.	o
11	The worker becomes safe, and the temperature inside the factory decreases, making the factory in normal mode.	o

Demo video Link : https://www.youtube.com/watch?v=e--00xkY3is

V. Trouble Shooting

1. As multiple timers were used, there were many errors caused by overlapping timers.
    - We solved this problem by designing a timer through pin map of ecPWM.c.
2. When the operating temperature of the temperature sensor is set to 20 degrees, the cooler fan stops near the operating temperature and repeats operation.
    - If the target temperature is set to 20 degrees, this phenomenon can be prevented by setting the operating temperature to +2 degrees and the stopping temperature to -2 degrees. I learned from my studies that many devices used in real industries work with algorithms like this

VI. Appendix

Github ; https://github.com/DongminKim21800064/EC_dmkim-064/tree/main/Final

Main_1.c (Forklift main)

```
/**
******************************************************************************
* @author  Dongmin Kim & Jinho Kook
* @Mod       2022-12-16 by Dongmin Kim
* @brief   Embedded Controller:  Final Project_ Forklift_Main
*
******************************************************************************
*/
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecSysTick.h"
```

```c
#include "ecUART.h"
#include "ecADC.h"
#include "ecPWM.h"
#include "ecEXTI.h"
#include "math.h"
#include "string.h"


//IR parameter//
uint32_t IR1, IR2;
//int flag = 0;
int seqCHn[16] = {8,9, };
int forklift_flag = 0;

#define END_CHAR 13
#define A 0
#define B 1
#define MAX_BUF 100

PWM_t pwm11;

//USART parameter//
uint8_t pcData = 0;
uint8_t mcu2Data = 0;
uint8_t btData = 0;
uint8_t buffer[MAX_BUF] = {0, };
uint8_t buffer2 = '\r\n';
int bReceive = 0;
int idx = 0;

//Flag and Contol Mode//
int flag = 5;
int dis_flag =0;

int dirA =0;
int dirB =0;

int R_DC_velocity =0;
int L_DC_velocity =0;

float R_duty = 0.f;
float L_duty = 0.f;

int Auto_mode = 1;

uint16_t flagidx =0;

//Ultro Sonic parameters//
uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;

//Piezo Sensor//
int PS = 0;
//Buzzer//
int bz = 0;
```

```
void setup(void);



_Pin dcPwmPin[2] = {
   {GPIOC, 9}, // TIM3 Ch3 A-IA
   {GPIOC, 8}   // TIM3 Ch4 B-IA
};

PWM_t dcPwm[2];

_Pin dcDirPin[2] = {
   {GPIOB, 8}, //A-IB
   {GPIOC, 6}   //B-IB
};


int main(void) {

   // Initialiization --------------------------------------------------------
          setup();
      printf("Hello Nucleo\r\n");
      ADC_start();
      PWM_duty(&pwm11, (0.5+((2.5-0.5)/18)*13) /20); // Set 0 degree


   // Inifinite Loop ---------------------------------------------------------
   while(1){
       // Read the Piezo sensor
             PS = GPIO_read(GPIOC, 0);
               delay_ms(30);
             // If supervisor push the 'Q', Line Tracing Mode On
       if(btData == 'Q'){
          Auto_mode = 1;
             GPIO_write(GPIOC, 3, 0);
       }
       // If supervisor push the 'B', Stop Mode
       else if(btData =='B') {
          PWM_period_ms(&dcPwm[A], 40);
          PWM_period_ms(&dcPwm[B], 40);

          PWM_duty(&dcPwm[A], 1);
                    PWM_duty(&dcPwm[B], 1);
                    GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin, 1);
                    GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin, 1);
                    delay_ms(100);
       }
       // If supervisor push the 'E', Manual Mode On
       else if (btData == 'E'){
          Auto_mode = 0;
       }

       // Line Tracing Mode
       if(Auto_mode ==1){
          GPIO_write(GPIOA, LED_PIN, 0);
```

```
                    distance = (float) timeInterval * 340.0 / 2.0 / (10.0 *
100);
          printf("raw_distance = %f [cm]\r\n", distance);
          delay_ms(500);

               //Danger detecting
             if( distance < 12.0 && 2 < distance) {
                              //If object so close, Danger Mode On
                                     printf("disflag is on\n"); delay_ms(30);

                                     PWM_period_ms(&dcPwm[A], 40);
                                     PWM_period_ms(&dcPwm[B], 40);

                                     PWM_duty(&dcPwm[A], 1);
                                     PWM_duty(&dcPwm[B], 1);

                                   GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin,
1);
                  GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin, 1);
                  delay_ms(1000);

                                // If Pressure sensing, Lift up Mode On
                        if (PS == HIGH) {
                           printf("Lift on");
                           PWM_duty(&pwm11, (0.5+((2.5-0.5)/18)*18) /20); //LIFT
UP
                           delay_ms(300);
                           Auto_mode = 0;
                         }

                        }
                // Line Tracing Mode
        else {
                                     printf("IR1 = %d \r\n",IR1);
                                     printf("IR2 = %d \r\n",IR2);
                                     printf("\r\n");

                                     PWM_period_ms(&dcPwm[A], 40); //R
                                     PWM_period_ms(&dcPwm[B], 40); //L

                                     PWM_duty(&dcPwm[A], 0.7);
                                     PWM_duty(&dcPwm[B], 0.7);

                                     GPIO_write(dcDirPin[A].port,
dcDirPin[A].Pin, 1);
                                     GPIO_write(dcDirPin[B].port,
dcDirPin[B].Pin, 1);


                             if (IR1 > 1000){
                                              printf("GO LEFT\r\n");

                                              PWM_period_ms(&dcPwm[A], 40);
                                              PWM_period_ms(&dcPwm[B], 40);

                                              PWM_duty(&dcPwm[A], 1);
                                              PWM_duty(&dcPwm[B], 0.5);
```

```
                                        GPIO_write(dcDirPin[A].port,
dcDirPin[A].Pin, 1);
                                        GPIO_write(dcDirPin[B].port,
dcDirPin[B].Pin, 1);
                                        delay_ms(10);
                            }

                            if (IR2 > 1000) {
                                        printf("GO RIGHT\r\n");
                                        printf("\r\n");


                                        PWM_period_ms(&dcPwm[A], 40);
                                        PWM_period_ms(&dcPwm[B], 40);

                                        PWM_duty(&dcPwm[A], 0.5);
                                        PWM_duty(&dcPwm[B], 1);

                                        GPIO_write(dcDirPin[A].port,
dcDirPin[A].Pin, 1);
                                        GPIO_write(dcDirPin[B].port,
dcDirPin[B].Pin, 1);
                                        delay_ms(10);
                            }
                }
    }
                // Manual Mode
          else if (Auto_mode == 0){
             // Direction Control Mode
             if (btData == 'W'){ //go
                                 PWM_period_ms(&dcPwm[A], 50);
                                 PWM_period_ms(&dcPwm[B], 50);

                                PWM_duty(&dcPwm[A], 0.5);
                                PWM_duty(&dcPwm[B], 0.5);

                                GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin,
1);
                                GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin,
1);
                                GPIO_write(GPIOC, 3, 0);
                            }
                            else if (btData == 'N'){ //nutral
                                     PWM_period_ms(&dcPwm[A], 50);
                                     PWM_period_ms(&dcPwm[B], 50);

                                     PWM_duty(&dcPwm[A], 1.0);
                                     PWM_duty(&dcPwm[B], 1.0);
                                     GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin,
1);
                                     GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin,
1);
                                     GPIO_write(GPIOC, 3, 0);
                            }

                            else if (btData == 'S'){ //back
                                    PWM_period_ms(&dcPwm[A], 50);
                                    PWM_period_ms(&dcPwm[B], 50);
```

```
                                PWM_duty(&dcPwm[A], 0.5);
                                PWM_duty(&dcPwm[B], 0.5);

                                GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin,
0);
                                GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin,
0);

                                GPIO_toggle(GPIOC,3);
                delay_ms(500);
                        }

                        else if (btData == 'D'){ //right
                                PWM_period_ms(&dcPwm[A], 110);
                                PWM_period_ms(&dcPwm[B], 110);

                                PWM_duty(&dcPwm[A], 0.4);
                                PWM_duty(&dcPwm[B], 0.4);

                                GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin,
1);
                                GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin,
0);
                                GPIO_write(GPIOC, 3, 0);
                        }

                         else if (btData == 'A'){ //left
                                PWM_period_ms(&dcPwm[A], 110);
                                PWM_period_ms(&dcPwm[B], 110);

                                PWM_duty(&dcPwm[A], 0.4);
                                PWM_duty(&dcPwm[B], 0.4);

                                GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin,
0);
                                GPIO_write(dcDirPin[B].port, dcDirPin[B].Pin,
1);
                                GPIO_write(GPIOC, 3, 0);
                                 }

        // Lift Control Mode
                        else if (btData == 'L'){ //lIft UP
                PWM_duty(&pwm11, (0.5+((2.5-0.5)/18)*18) /20); // Set 0 degree,
lift up
                                }
             else if (btData == 'K'){ //lIft down
                PWM_duty(&pwm11, (0.5+((2.5-0.5)/18)*13) /20); // Set 0 degree,
lift down
                                }



   }
 }
 }
```

```
// Initialiization
void setup(void)
{
   RCC_PLL_init();                          // System Clock = 84MHz
   UART2_init();
   SysTick_init(1);
   TIM_INT_enable(TIM4);

   GPIO_pupd(GPIOA, 8, 0x00);
   GPIO_pupd(GPIOB, 6, 0x00);
   GPIO_otype(GPIOA, 8, 0);
   GPIO_ospeed(GPIOA, 8, 2);
   USART_begin(USART1, GPIOA, 9, GPIOA, 10, 9600);    // PA9: TXD , PA10: RXD
   GPIO_otype(GPIOB,6 , 0);
   GPIO_ospeed(GPIOB,6, 2);
   GPIO_init(GPIOC, 0, 0x00);
   GPIO_init(GPIOC, 3, 0x01);

   // ADC setting
   ADC_init(GPIOB, 0, TRGO);
   ADC_init(GPIOB, 1, TRGO);
   ADC_sequence(2, seqCHn);

   // ADON, SW Trigger enable
   ADC_start();

   GPIO_init(GPIOC, BUTTON_PIN, INPUT);
   GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);
   EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);                        // GPIOC pin 13
Initialization

    // Digital Out: RC ServoMotor
   GPIO_init(GPIOA, 1, AF);
   GPIO_pupd(GPIOA,1, 01);
   GPIO_otype(GPIOA, 1, 0);
   GPIO_ospeed(GPIOA, 1, 11);

   // PWM
   PWM_init(&pwm11, GPIOA, 1);
   PWM_period_ms(&pwm11, 20);

   // PWM configuration ---------------------------------------------------------
-------------
   PWM_t trig;                                    // PWM1 for trig
   PWM_init(&trig, GPIOA, 8);                          // PA_6: Ultrasonic trig
pulse
   PWM_period_us(&trig, 50000);       // PWM of 50ms period. Use period_us()
   PWM_pulsewidth_us(&trig, 10);      // PWM pulse width of 10us

// Input Capture configuration -----------------------------------------------
---------------------
   IC_t echo;                                    // Input Capture for echo
   ICAP_init(&echo, GPIOB, 6);          // PB6 as input caputre
   ICAP_counter_us(&echo, 10);         // ICAP counter step time as 10us
   ICAP_setup(&echo, 1, RISE);      // TIM2_CH3 as IC3 , rising edge detect
   ICAP_setup(&echo, 2, FALL);      // TIM2_CH3 as IC4 , falling edge detect
```

```
   PWM_init(&dcPwm[A], dcPwmPin[A].port, dcPwmPin[A].Pin);
   PWM_init(&dcPwm[B], dcPwmPin[B].port, dcPwmPin[B].Pin);

   for (int i = 0; i < 2; i++){
      GPIO_init(dcDirPin[i].port, dcDirPin[i].Pin,OUTPUT);
      GPIO_pupd(dcDirPin[i].port, dcDirPin[i].Pin, 01);
      GPIO_otype(dcDirPin[i].port, dcDirPin[i].Pin, 0);
      GPIO_ospeed(dcDirPin[i].port, dcDirPin[i].Pin, 11);
   }
 }

void ADC_IRQHandler(void){
   if((is_ADC_OVR())){
      clear_ADC_OVR();
   }

   if(is_ADC_EOC()){       //after finishing sequence
         if (flag==0){
            IR1 = ADC_read();
         }
         else if (flag==1){
            IR2 = ADC_read();
         }
      flag =! flag;
   }
}



void TIM4_IRQHandler(void){
   if(is_UIF(TIM4)){
      ovf_cnt++;
      clear_UIF(TIM4);
   }
   if(is_CCIF(TIM4, 1)){
      time1 = TIM4->CCR1;
      clear_CCIF(TIM4, 1);
   }
   else if(is_CCIF(TIM4, 2)){
      time2 = TIM4->CCR2;
      timeInterval = ( (time2 - time1) + ( (TIM4->ARR) + 1 )*ovf_cnt );
      ovf_cnt = 0;
      clear_CCIF(TIM4, 2);
   }
}


void USART1_IRQHandler(){         //USART1 INT
   if(is_USART_RXNE(USART1)){
      btData = USART_getc(USART1);
      USART_write(USART1,(uint8_t*) "BT sent : ", 10);
      USART_write(USART1, &btData, 1);
      USART_write(USART1, "\r\n", 2);
     // RC_control(&btData);
      printf("NUCLEO got : %c (from BT)\r\n",btData);
   }
}
```

Main_2.c (Forklift_piezo sensor)

```
/**
******************************************************************************
* @author  Dongmin Kim & Jinho Kook
* @Mod       2022-12-16 by Dongmin Kim
* @brief   Embedded Controller:  Final Project_ Forklift_Piezosensor
*
******************************************************************************
*/
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecSysTick.h"
#include "ecUART.h"
#include "ecADC.h"
#include "ecPWM.h"
#include "ecEXTI.h"
#include "math.h"
#include "string.h"



//IR parameter//
uint32_t IR1, IR2;
//int flag = 0;
int seqCHn[1] = {8};
int forklift_flag = 0;


#define END_CHAR 13
#define A 0
#define B 1
#define MAX_BUF 100

PWM_t pwm11;

//USART parameter//
uint8_t pcData = 0;
uint8_t mcu2Data = 0;
uint8_t btData = 0;
uint8_t buffer[MAX_BUF] = {0, };
uint8_t buffer2 = '\r\n';
int bReceive = 0;
int idx = 0;

//Flag and Contol Mode//
int flag = 5;
int dis_flag =0;
int dirA =0;
int dirB =0;
```

```
int R_DC_velocity =0;
int L_DC_velocity =0;
float R_duty = 0.f;
float L_duty = 0.f;
int Auto_mode = 1;
int pressure = 0;

uint16_t flagidx =0;



uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;



void setup(void);




int main(void) {

   // Initialiization --------------------------------------------------------
   setup();
            printf("Hello Nucleo\r\n");
      ADC_start();

   // Inifinite Loop ---------------------------------------------------------
   while(1){
       if (pressure >= 100){
                    GPIO_write(GPIOA, 5, 1);
                    delay_ms(1000);
                }

       else if (pressure < 100){
                    GPIO_write(GPIOA, 5, 0);
                    delay_ms(1000);
                }

              printf("pressure = %d \r\n", pressure);
      }
        }



// Initialiization
void setup(void)
{
   RCC_PLL_init();                          // System Clock = 84MHz
   UART2_init();
   SysTick_init(1);
   GPIO_init(GPIOA, 5, 0x01);
   USART_init(USART2, 9600);
```

```
   // ADC setting
   ADC_init(GPIOB, 0, TRGO);

   // ADC channel sequence setting
   ADC_sequence(1, seqCHn);

   // ADON, SW Trigger enable
   ADC_start();

}

void ADC_IRQHandler(void){
   if((is_ADC_OVR())){
      clear_ADC_OVR();
   }

   if(is_ADC_EOC()){       //after finishing sequence

      pressure = ADC_read();

   }
}
```

Main_3.c (CoolerFan & Alarm)

```
/**
******************************************************************************
* @author  Dongmin Kim & Jinho Kiik
* @Mod       2022-12-16 by Dongmin Kim
* @brief   Embedded Controller:  Final Project_ Cooler&Alarm
*
******************************************************************************
*/
#include "stm32f4xx.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecUART.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecSysTick.h"
#include "ecADC.h"
#include "math.h"

//#define VOLTS_PER_UNIT 0.00122F
//IR parameter//
static volatile double result_v =0;
static volatile double volts =0;
static volatile double temperature =0;

// Control the Setting of temperature//
static volatile double set_temperature = 20;

static volatile double temp1 =0;
static volatile double temp2 =0;
```

```
static volatile int flag =0;
int seqCHn[16] = {8,9,};

//Ultro Sonic parameters//
uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;



void setup(void);
void ADC_IRQHandler(void);
void TIM2_IRQHandler(void);

#define A 0
#define B 1

PWM_t pwm11;

int main(void) {
   // Initialiization --------------------------------------------------------
   setup();

   // Inifinite Loop ---------------------------------------------------------
   while(1){


         // Calculate the distance
      distance = (float) timeInterval * 340.0 / 2.0 / (10.0 * 100);
      printf("raw_distance = %f [cm]\r\n", distance);
      delay_ms(500);

         // If Forklift so close, Buzzer On!
         if(distance < 20.0 && 2 < distance) {
           printf("Caution! Danger detected!! \r\n");
           GPIO_toggle(GPIOC,3);
                 }

      // Calculate the temperature => result_v is output of temperature sensor
      temperature = ((float)result_v/10)-7;
      printf("temperature : %.3f 'c \r\n", temperature);

            // set_temperture = 20 C degree
            // If temperature up to 20+2, Cooler Fan On
       if(temperature > set_temperature+2 || temp1 >set_temperature+2 || temp2
>set_temperature+2){
                PWM_period_ms(&dcPwm[A], 40);
                PWM_duty(&dcPwm[A], 0.2);
                GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin, 1);
                       delay_ms(25);
          // If temperature down to 20-2, Cooler Fan Off
         } else if(temperature < set_temperature-2 || temp1 <set_temperature-2
|| temp2 < set_temperature-2)
             {
                PWM_period_ms(&dcPwm[A], 40);
```

```
                PWM_duty(&dcPwm[A], 1);
                GPIO_write(dcDirPin[A].port, dcDirPin[A].Pin, 1);
                        delay_ms(25);
        }
            // To save previous temperature
        temp2 = temp1;
        temp1 = temperature;
                delay_ms(500);
    }
}




// Initialiization
void setup(void)
{
    RCC_PLL_init();                         // System Clock = 84MHz
    SysTick_init(1);
    UART2_init();
    ADC_init(GPIOB, 0, TRGO);
    //ADC_continue(CONT);

//ADC_sequence(1, seqCHn);
    ADC_start();

// Buzzer => PC3
    GPIO_init(GPIOC, 3, 0x01);

// Enable TIMx interrupt         ------------------------------------------------
-----------------------
    TIM_INT_enable(TIM4);                 // TIM4 Interrupt Enable

// PWM configuration ---------------------------------------------------------
---------
    PWM_t trig;                                     // PWM1 for trig
    PWM_init(&trig, GPIOA, 8);                      // PA_8: Ultrasonic trig pulse
    PWM_period_us(&trig, 50000);       // PWM of 50ms period. Use period_us()
    PWM_pulsewidth_us(&trig, 10);      // PWM pulse width of 10us

// Input Capture configuration -----------------------------------------------
---------------------
      IC_t echo;                                    // Input Capture for echo
    ICAP_init(&echo, GPIOB, 6);          // PB10 as input caputre
    ICAP_counter_us(&echo, 10);         // ICAP counter step time as 10us
    ICAP_setup(&echo, 1, RISE);      // TIM2_CH3 as IC3 , rising edge detect
    ICAP_setup(&echo, 2, FALL);      // TIM2_CH3 as IC4 , falling edge detect


      PWM_init(&pwm11, GPIOA, 1);
    PWM_period_ms(&pwm11, 20);

    // PWM for cooler Fan DC motor
    PWM_period_ms(&pwm11, 20);


    }
```

```
void TIM4_IRQHandler(void){
   if(is_UIF(TIM4)){
      ovf_cnt++;
      clear_UIF(TIM4);
   }
   if(is_CCIF(TIM4, 1)){
      time1 = TIM4->CCR1;
      clear_CCIF(TIM4, 1);
   }
   else if(is_CCIF(TIM4, 2)){
      time2 = TIM4->CCR2;
      timeInterval = ( (time2 - time1) + ( (TIM4->ARR) + 1 )*ovf_cnt );
      ovf_cnt = 0;
      clear_CCIF(TIM4, 2);
   }
}


void ADC_IRQHandler(void){
   if((is_ADC_OVR())){
      clear_ADC_OVR();
   }

   if( is_ADC_EOC() ){       //after finishing sequence
     result_v = ADC_read();


 }
}
```

Arduino main.c

```
#include <Wire.h>
#include <LiquidCrystal_I2C_Hangul.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C_Hangul lcd(0x27, 16, 2);

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.print("LM35 Test");
  delay(1000);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 -
5V):
```

```
  float voltage = sensorValue * (5.0 / 1023.0);
  float temperature = voltage*100;
  // print out the value you read:
  Serial.println("Voltage : ");
  Serial.println(voltage);
  Serial.println("Temperature : ");
  Serial.println(temperature);

  lcd.setCursor(0,0);
  lcd.print("T :");
  lcd.print(temperature);
  lcd.print((char)223);
  lcd.print("C");
  delay(500);
}
```