Obstacle avoidance car V1.0

DESIGN DOCUMENT

ASSINED BY:

MOHAMED SAYAD

Mohamed Sayed
SPRINTS | MAADI

Table of contents

Contents

Sprints

Firstly: Project Description:

Description

Hardware Requirements

1. ATmega32 microcontroller

2. Four motors (M1, M2, M3, M4)

3. One button to change default direction of rotation (PBUTTON0)

4. Keypad button 1 to start

5. Keypad button 2 to stop

6. One Ultrasonic sensor connected as follows

7. LCD

Software Requirements

1. The car starts initially from 0 speed

2. The default rotation direction is to the right

3. Press (Keypad Btn 1), (Keypad Btn 2) to start or stop the robot respectively

4. After Pressing Start:

1. The LCD will display a centered message in line 1 “Set Def. Rot.”
2. The LCD will display the selected option in line 2 “Right”
3. The robot will wait for 5 seconds to choose between Right and Left
   - When PBUTTON0 is pressed once, the default rotation will be Left
   - and the LCD line 2 will be updated
   - When PBUTTON0 is pressed again, the default rotation will be Right
   - and the LCD line 2 will be updated
   - For each press the default rotation will changed and the LCD line 2 is
   - updated
   - After the 5 seconds the default value of rotation is set
4. The robot will move after 2 seconds from setting the default direction of Rotation

5. For No obstacles or object is far than 70 centimeters:

1. The robot will move forward with 30% speed for 5 seconds
2. After 5 seconds it will move with 50% speed as long as there was no object or objects are located at more than 70 centimeters distance
3. The LCD will display the speed and moving direction in line 1: “Speed:00% Dir: F/B/R/S”, F: forward, B: Backwards, R: Rotating, and S: Stopped
4. The LCD will display Object distance in line 2 “Dist.: 000 Cm”

6. For Obstacles located between 30 and 70 centimeters

1. The robot will decrease its speed to 30%

2. LCD data is updated

7. For Obstacles located between 20 and 30 centimeters

1. The robot will stop and rotates 90 degrees to right/left according to the chosen configuration
2. The LCD data is updated

8. For Obstacles located less than 20 centimeters

1. The robot will stop, move backwards with 30% speed until distance is greater than 20 and less than 30
2. The LCD data is updated
3. Then preform point 8

9. Obstacles surrounding the robot (Bonus)

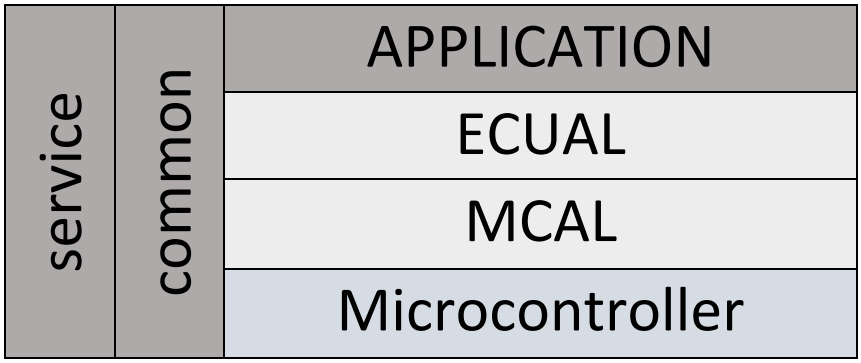
1. If the robot rotated for 360 degrees without finding any distance greater than 20 it will stop
2. LCD data will be updated.
3. The robot will frequently (each 3 seconds) check if any of the obstacles was removed or not and move in the direction of the furthest object

Sprints

Secondly: Layered architecture:

1- Microcontroller

2- MCAL
3- ECUAL
4- SERVICES
5- COMMON
6- Application



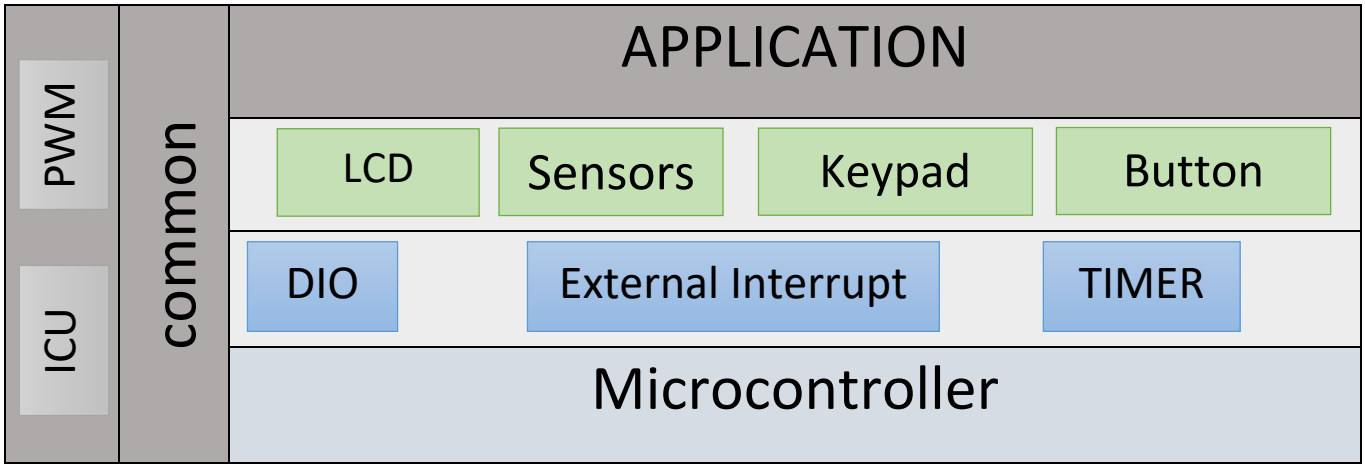
Thirdly : System modules:

1- Specify system modules/drivers:

- DIO, TIMER, External interrupt
- LCD, KEYPAD, SENSOR, Button
- APPLICATION

2- Assign each module to its related layer:

- By drawing



Sprints

Fourthly: APIs: **MCAL APIs**

DIO APIs

- void DIO_InitPin (PIn_name pin ,PIN_Status status);
  void DIO_init (void);
  void DIO_WRitePin (PIn_name pin ,Voltage_type s);
  Voltage_type DIO_ReadPin(PIn_name pin);
  void DIO_WritePort(PORT_Type Port,u8 data);

Configuration file:

```
typedef enum {
        PINA0,
        PINA1,
        PINA2,
        PINA3,
        PINA4,
        PINA5,
        PINA6,
        PINA7,
        PINB0,
        PINB1,
        PINB2,
        PINB3,
        PINB4,
        PINB5,
        PINB6,
        PINB7,
        PINC0,
        PINC1,
        PINC2,
        PINC3,
        PINC4,
        PINC5,
        PINC6,
        PINC7,
        PIND0,
        PIND1,
        PIND2,
        PIND3,
        PIND4,
        PIND5,
        PIND6,
        PIND7,
        PINs_Total
}PIn_name;

typedef enum{
        OUTPUT,
        INFREE,
        INPUT
}PIN_Status;

typedef enum {
        PA,
        PB,
        PC,
        PD
        }PORT_Type;

typedef enum {
LOW,
HIGH
}Voltage_type;
```

External Interrupt APIs

- void EXI_Enable (ExInterruptSource_type Interrupt);
- void EXI_Disable (ExInterruptSource_type Interrupt);
- void EXI_Trigger(ExInterruptSource_type Interrupt,TriggerEdge_type trigger);
- void EXI_SetCallBack(ExInterruptSource_type Interrupt,void(*pf)(void));

Configuration file:

```
typedef enum {
    LOW_LEVEL=0,
    ANY_LOGIC_CHANGE,
    FALLING_EDGE,
    RISING_EDGE
}TriggerEdge_type;

typedef enum{
    EX_INT0,
    EX_INT1,
    EX_INT2,
}ExInterruptSource_type
```

Timer APIs

- void Timer0_init (Timer0Mode_type mode ,Timer0Scaler_type scaler);
- void TIMER0_OV_InterruptEnable(void);
- void TIMER0_OV_InterruptDisable(void);
- void TIMER0_OV_SetCallBack(void(*local_fptr)(void));

Configuration file:

```
/*****************************
typedef enum{
     TIMER1_STOP=0,
     TIMER1_SCALER_1,
     TIMER1_SCALER_8,
     TIMER1_SCALER_64,
     TIMER1_SCALER_256,
     TIMER1_SCALER_1024,
     EXTERNAL0_FALLING,
     EXTERNAL0_RISING
}Timer1Scaler_type;

typedef enum
{
     TIMER1_NORMAL_MODE=0,
     TIMER1_CTC_ICR_TOP_MODE,
     TIMER1_CTC_OCRA_TOP_MODE
     TIMER1_FASTPWM_ICR_TOP_M
     TIMER1_FASTPWM_OCRA_TOP_

}Timer1Mode_type;

typedef enum
{
     OCRA_DISCONNECTED=0,
     OCRA_TOGGLE,
     OCRA_NON_INVERTING,
     OCRA_INVERTING

}OC1A_Mode_type;

typedef enum
{
     OCRB_DISCONNECTED=0,
     OCRB_TOGGLE,
     OCRB_NON_INVERTING,
     OCRB_INVERTING

}OC1B_Mode_type;


typedef enum{
     RISING,
     FALLING
}ICU_Edge_type;
```

```
/*****************************Timer0
typedef enum {
TIMER0_NORMAL_MODE=0,
TIMER0_PHASECORRECT_MODE,
TIMER0_CTC_MODE,
TIMER0_FASTPWM_MODE
}Timer0Mode_type;

typedef enum {
    TIMER0_STOP=0,
    TIMER0_SCALER_1,
    TIMER0_SCALER_8,
    TIMER0_SCALER_64,
    TIMER0_SCALER_256,
    TIMER0_SCALER_1024,
    EXTERNAL1_FALLING,
    EXTERNAL_RISING
}Timer0Scaler_type;

typedef enum
{
    OC0_DISCONNECTED=0,
    OC0_TOGGLE,
    OC0_NON_INVERTING,
    OC0_INVERTING

}OC0Mode_type;
```

Sprints

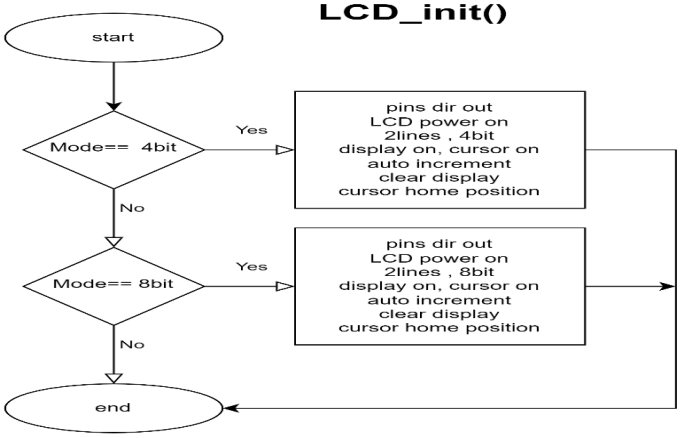
ECUAL APIs

LCD APIs

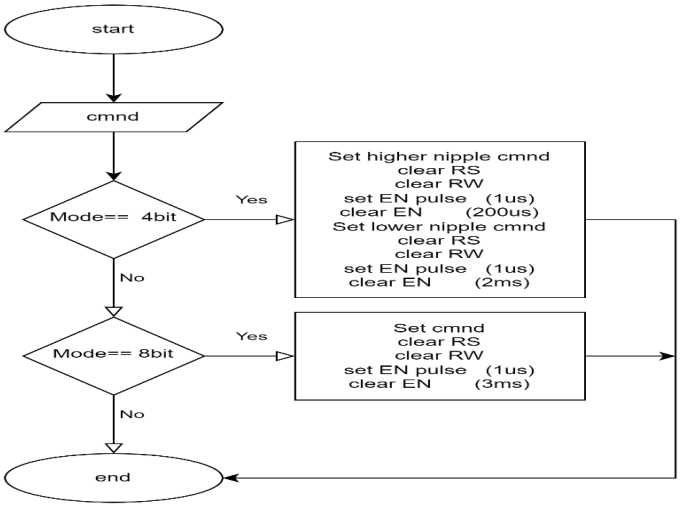
- void LCD_init (void);
- void LCD _sendcommand (u8);
- void LCD _sendstring(u8);
- void LCD_sendchar (u8);
- void LCD_setcursor (u8);
- void LCD_clear (void);
- void LCD_customchar(u8);
- void LCD_floattostring (f32_t float_value)

LCD FLOWCHARTS:

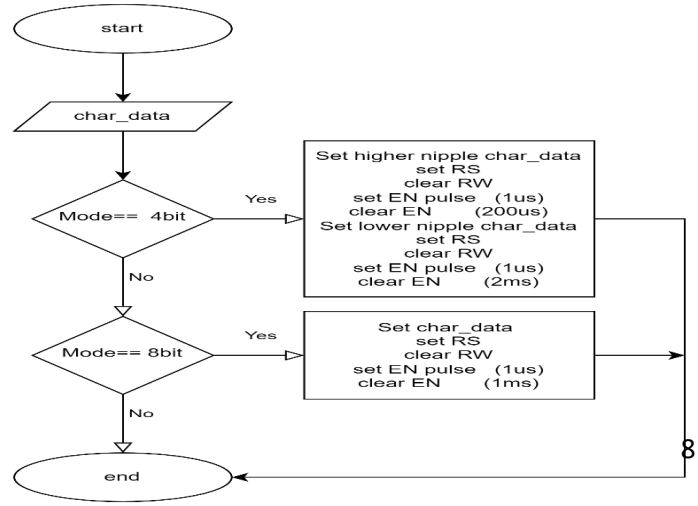
LCD_init()

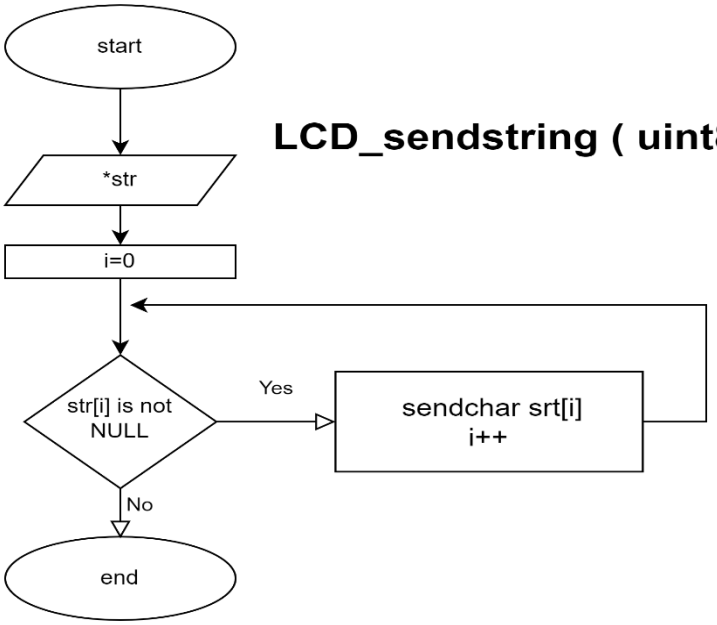


LCD_sendcommand(uint8_t cmnd)



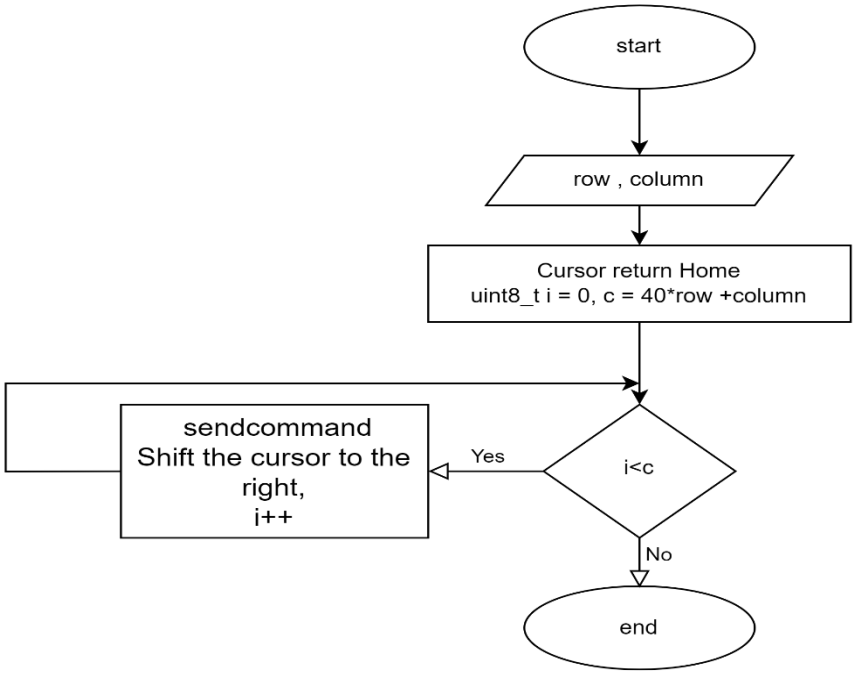
LCD_sendchar (uint8_t char_data)

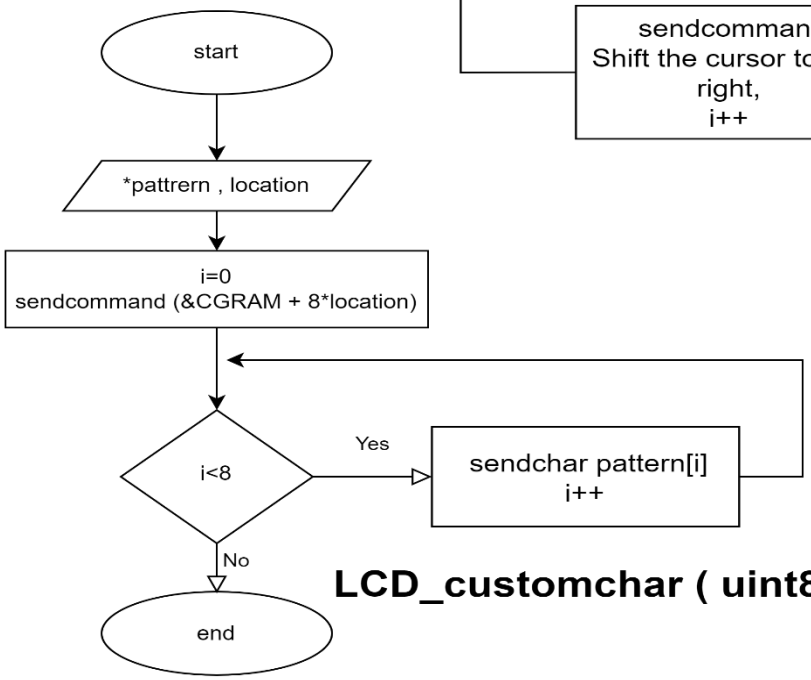




LCD_sendstring (uint8_t *str)

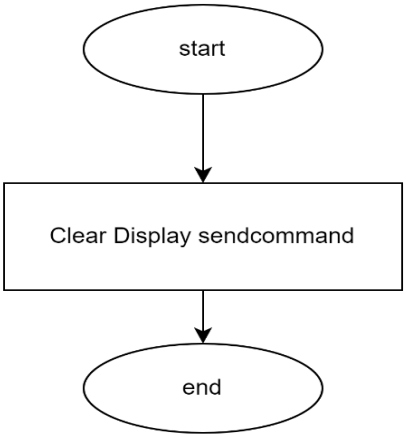
LCD_setcursor (uint8_t row,uint8_t column)



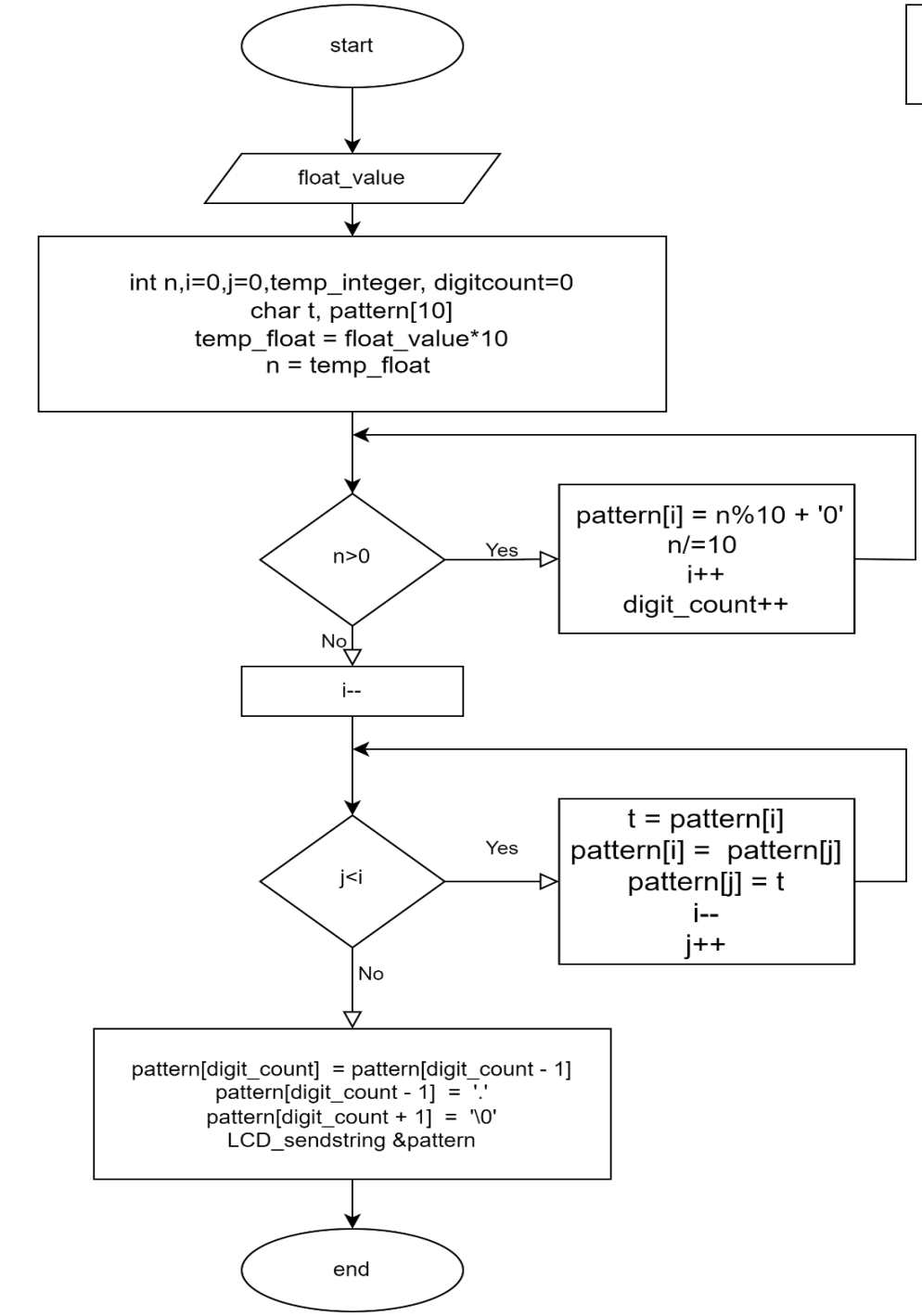


LCD_customchar (uint8_t *pattern,uint8_t location)

LCD_clear ()



LCD_floattostring (f32_t float_value)



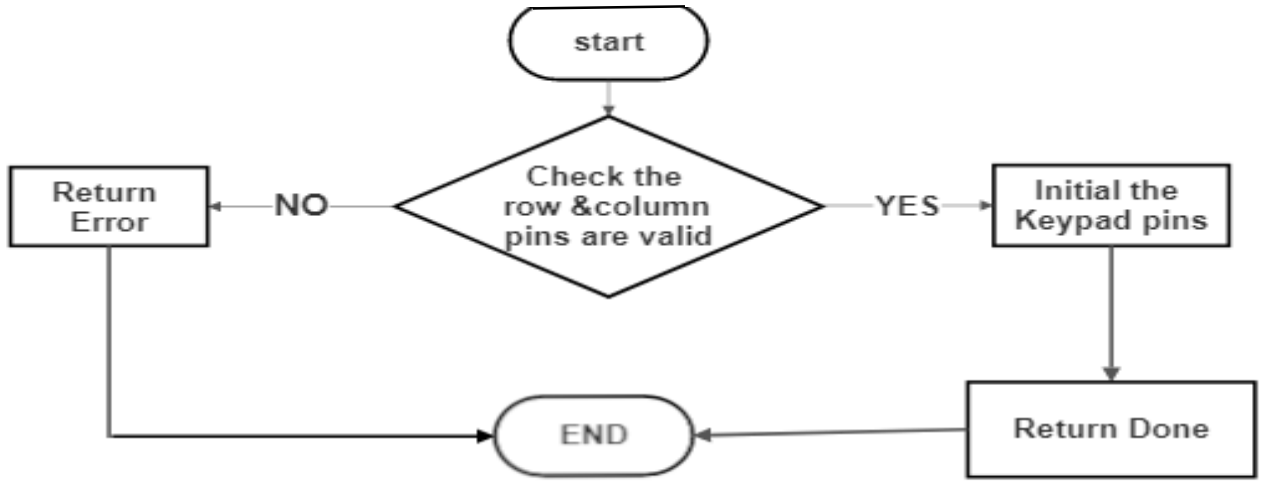
Configuration file:

```
#define  _4_Bit  1
#define  _8_Bit  2


/*******************pin config**********************/
#define LCD_Mode    _8_Bit

#define RS          PIND0
#define EN          PIND1

#define D7          PINA7
#define D6          PINA6
#define D5          PINA5
#define D4          PINA4

#define LCD_PORT    PC
/***************************************************/
```

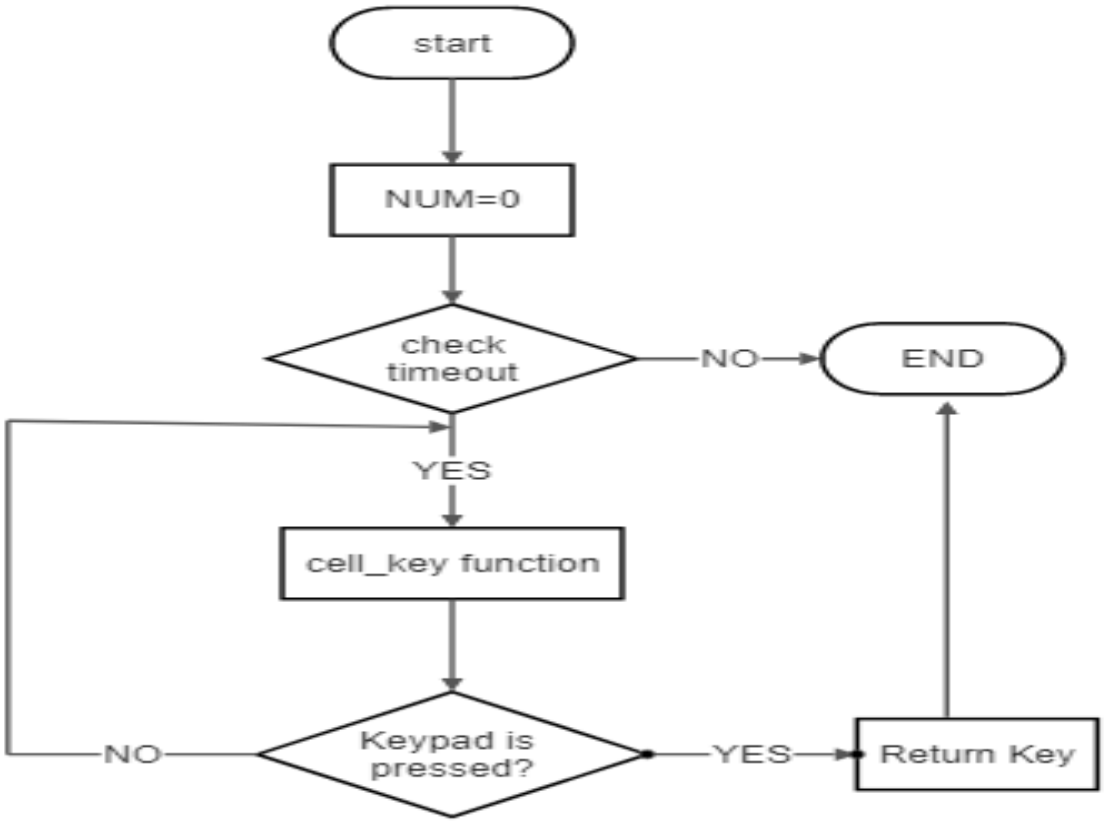
KeyPad APIs

- Keypad_Status_en KEYPAD_Init(PIn_name First_Output,PIn_name Firs_Input);
- Keypad_Status_en KEYPAD_GetNum_time(u8 timeout, u8* key);
- static u8 KEYPAD_GetKey(void);

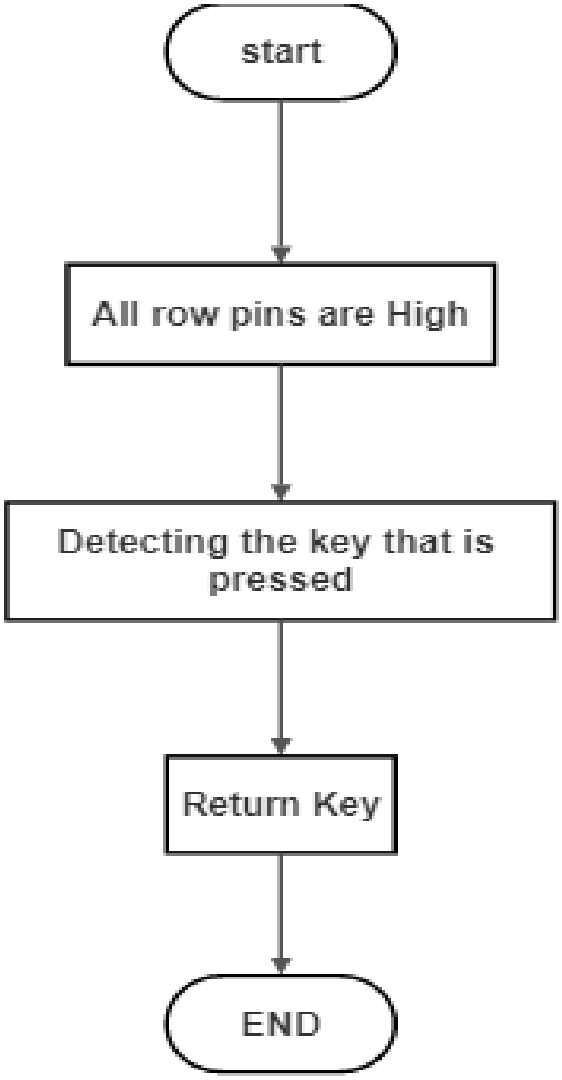
Keypad FLOWCHARTS:



Keypad_Status_en KEYPAD_Init(PIn_name First_Output,PIn_name Firs_Input);



Keypad_Status_en KEYPAD_GetNum_time(u8 timeout, u8* key);



static u8 KEYPAD_GetKey(void);

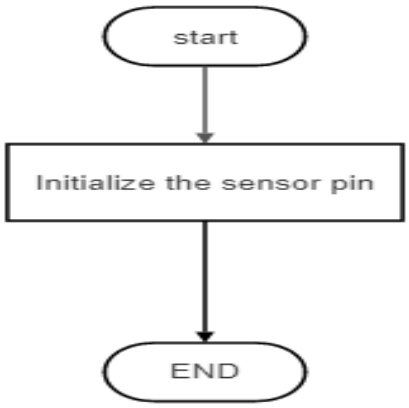
Configuration file:

```
#define COL   3
#define ROW   3

#define  NO_KEY  'N'

static u8 KeysArray[ROW][COL]={{'1','2','3'},
{'4','5','6'},
{'7','8','9'}};
/*************************************************/
```

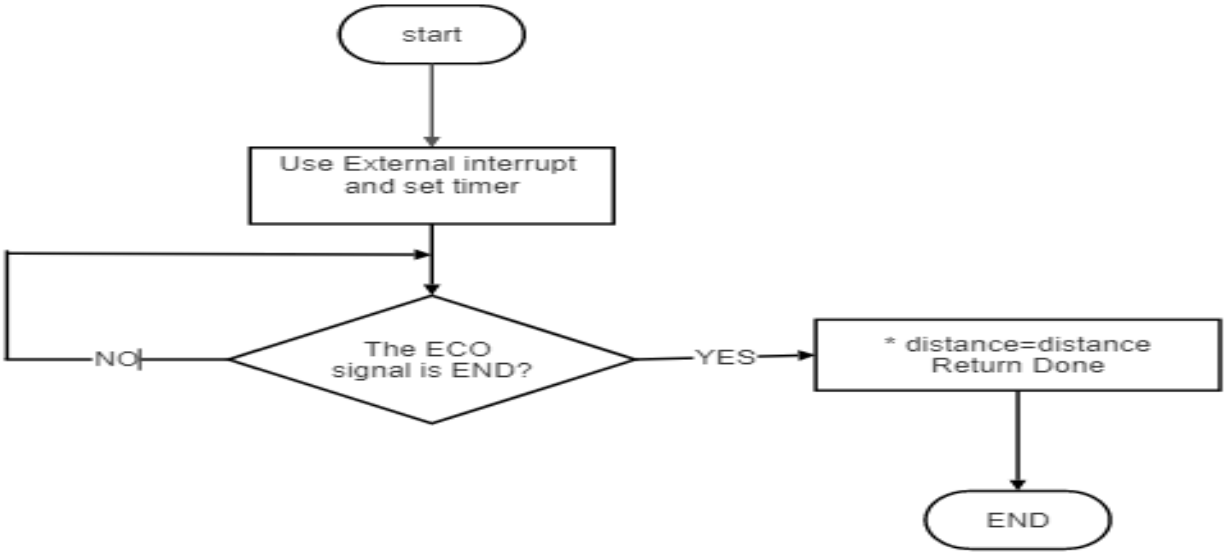
SENSOR APIs

- Ultrasonic_Status_en Ultrasonic_Init(u8 Eco_Pin , u8 Trigger_Pin);
- Ultrasonic_Status_en Ultrasonic_Get_Distance(u8* distance);

SENSOR FLOWCHARTS:



Ultrasonic_Status_en Ultrasonic_Init(u8 Eco_Pin , u8 Trigger_Pin);

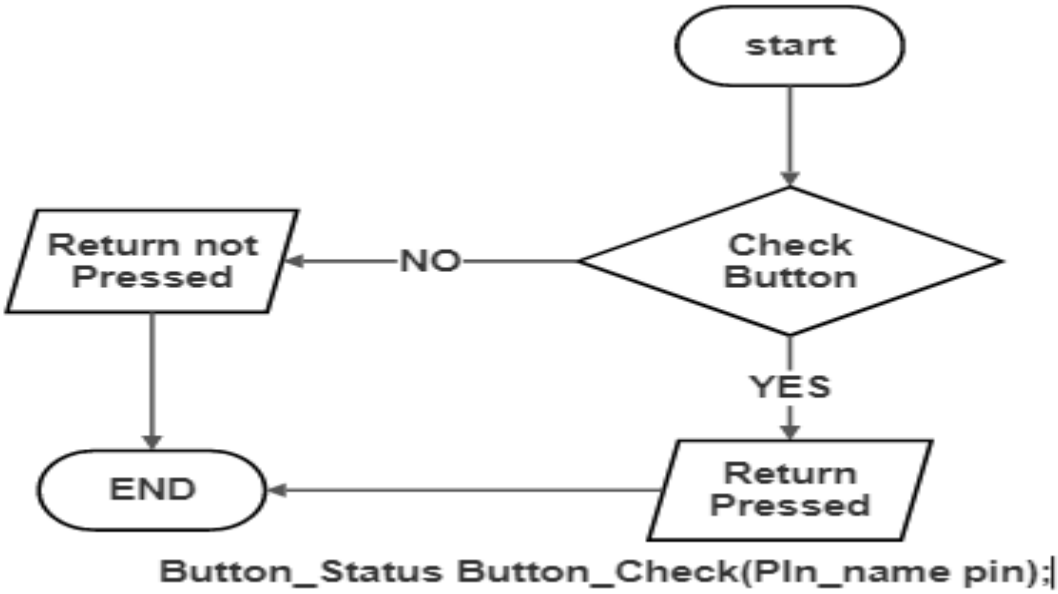


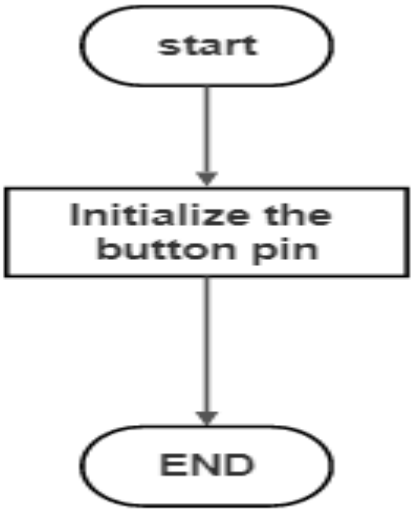
Ultrasonic_Status_en Ultrasonic_Get_Distance(u8* distance);

BUTTON APIs

- Button_Status Button_Check(PIn_name pin);
- void button_init(PIn_name pin);

BUTTON FLOWCHARTS:





void button_init(Pln_name pin);

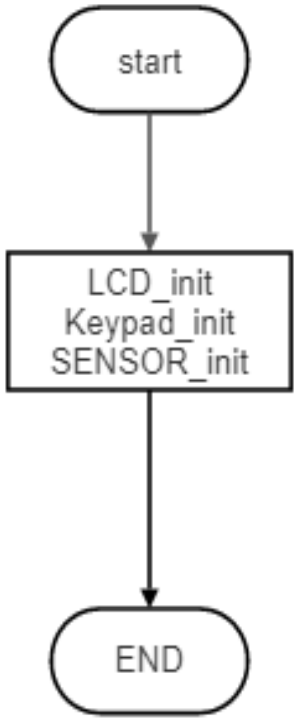
Configuration file:

```
typedef enum{
    NotPressed,
    Pressed
}Button_Status;
```

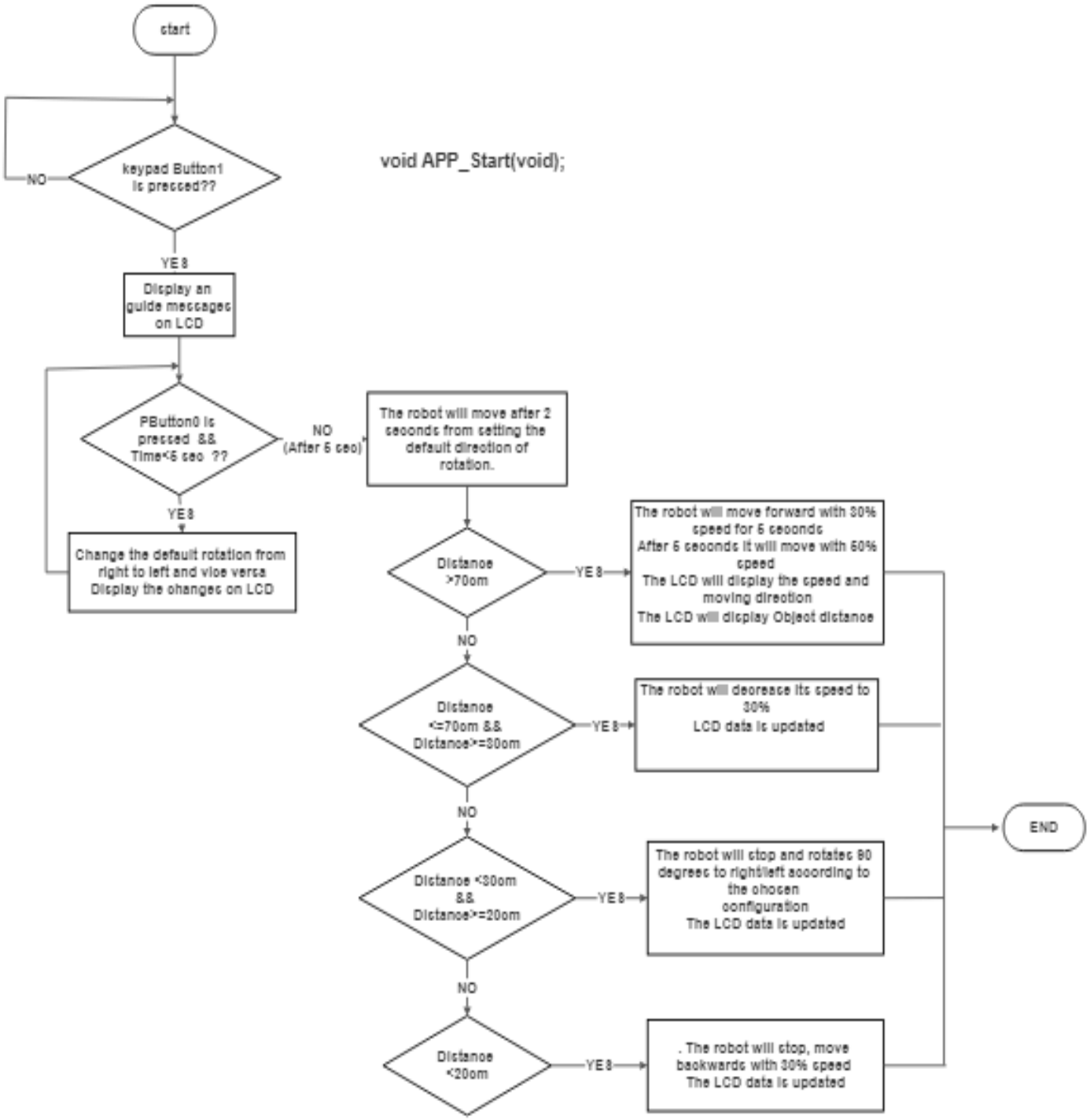
APPLICATION APIs:

- void APP_Init(void);
- void APP_Start(void);

APPLICATION FLOWCHARTS:



void APP_Init(void);


start
keypad Button1 is precced??
NO
YE S
Display an guide meccagec on LCD
PButton0 is prescsed && Time<5 sec ??
NO (After 5 sec)
YE S
Change the default rotation from right to left and vice versa Display the ohanges on LCD
The robot will move after 2 seconds from setting the default direction of rotation.
void APP_Start(void);
Distance >70cm
YE S
The robot will move forward with 30% speed for 5 seconds
After 5 seconds it will move with 60% speed
The LCD will display the speed and moving direction
The LCD will display Object distance
NO
Distance <=70cm && Distance>=30cm
YE S
The robot will decrease its speed to 30%
LCD data is updated
NO
Distance <30cm && Distance>=20cm
YE S
The robot will stop and rotates 90 degrees to right/left according to the chosen configuration
The LCD data is updated
NO
Distance <20cm
YE S
. The robot will stop, move backwards with 30% speed
The LCD data is updated
END