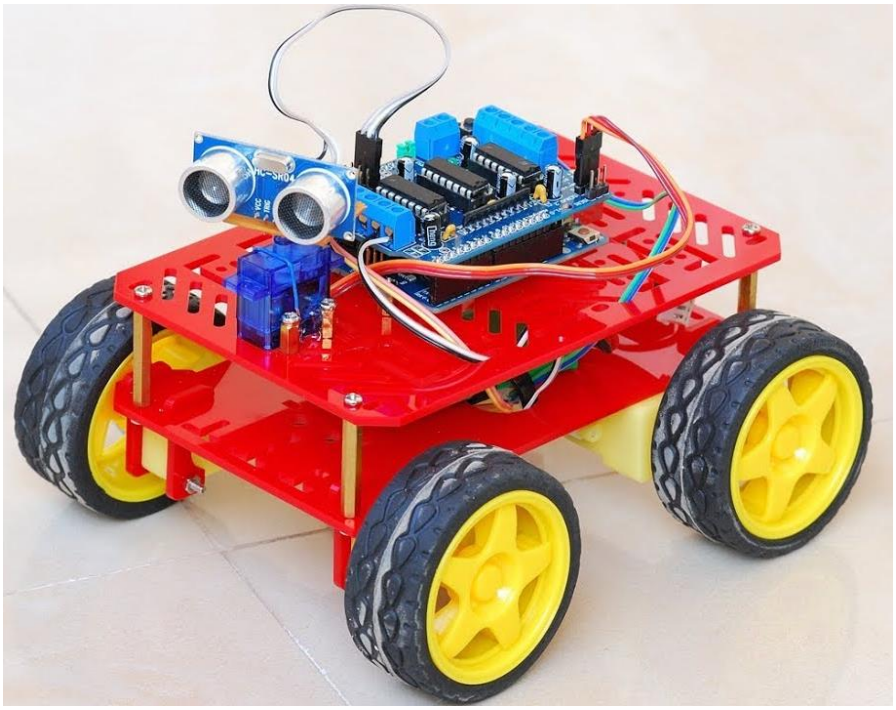
Obstacle avoidance car V1.0



Prepared by:

Mustafa Mohammed Abdou

Table of content:

1 – Description

1.1 Project description

Based on Atmega32 controller we have a 4-wheels robot which aims to avoid any obstacles on its way using an ultrasonic sensor to detect the distance between car and obstacle and takes actions according to the calculated distance.

1.2 Car components:

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
    1. Vcc to 5V in the Board
    2. GND to the ground In the Board
    3. Trig to PB3 (Port B, Pin 3)
    4. Echo to PB2 (Port B, Pin 2)
7. LCD

1.3 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

1. When PBUTTON0 is pressed once, the default rotation will be Left

and the LCD line 2 will be updated

2. When PBUTTON0 is pressed again, the default rotation will be Right

and the LCD line 2 will be updated

3. For each press the default rotation will changed and the LCD line 2 is updated

4. 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 rotate 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

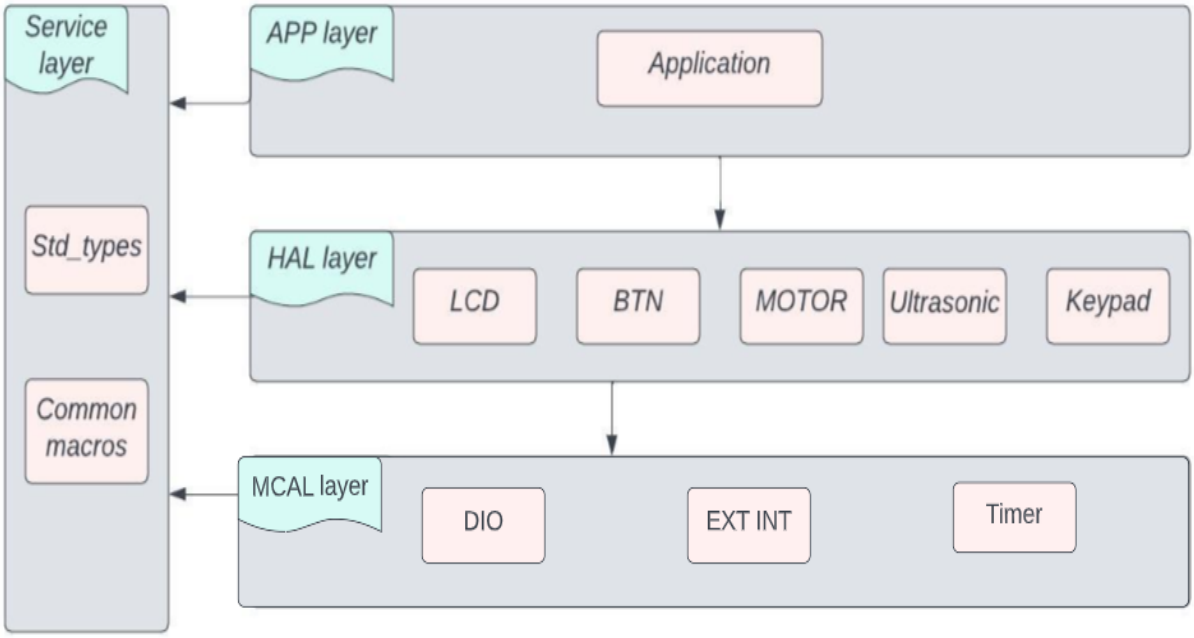
9. Obstacles surrounding the robot (Bonus)

1. If the robot rotated for 360 degrees without finding any distance greater than 20 it will stop

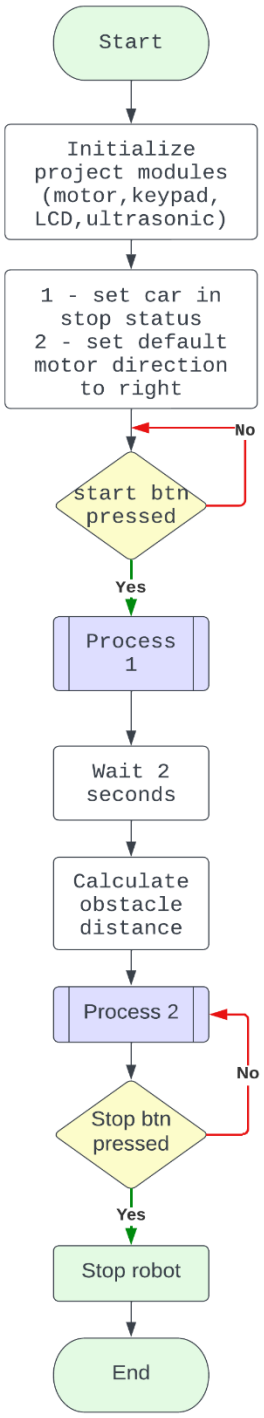
2. LCD data will be updated.

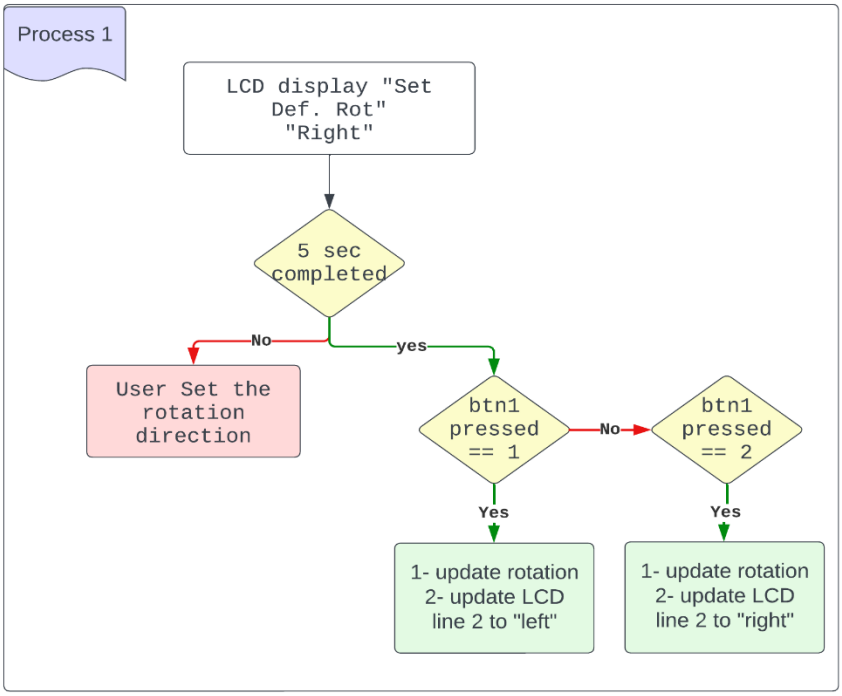
3. The robot will frequently (each 3 seconds) check if any of the obstacles was removed or not and move in the direction of the furthest object

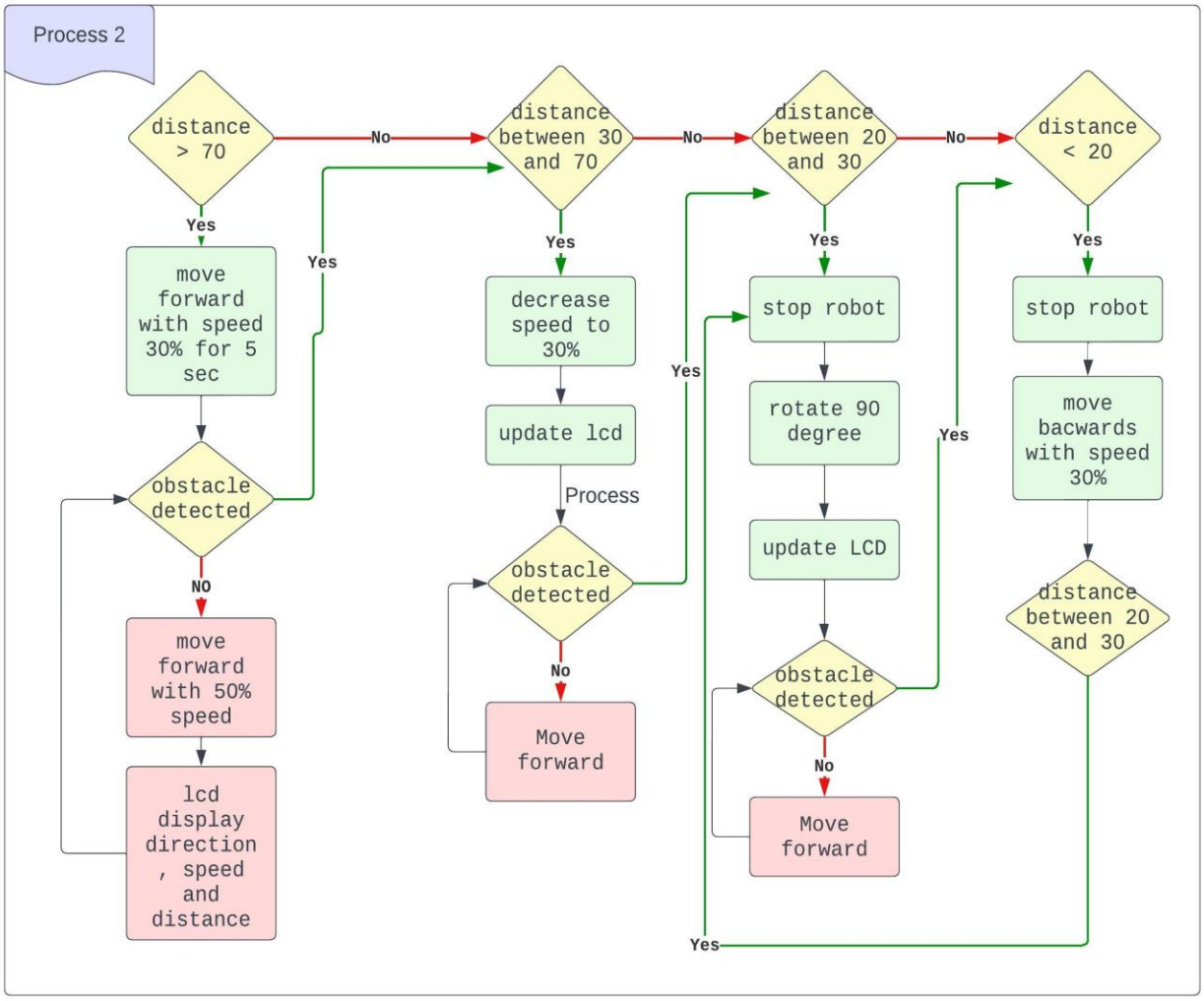
2 – Layered architecture



3- System flow chart






Process 2
distance > 70
No
distance between 30 and 70
No
distance between 20 and 30
No
distance < 20
Yes
move forward with speed 30% for 5 sec
obstacle detected
Yes
NO
move forward with 50% speed
lcd display direction , speed and distance
Yes
decrease speed to 30%
update lcd
Process
obstacle detected
Yes
No
Move forward
Yes
stop robot
rotate 90 degree
update LCD
obstacle detected
Yes
No
Move forward
Yes
stop robot
move bacwards with speed 30%
distance between 20 and 30
Yes

4- Module APIs

4.1 MCAL APIs

4.1.1 DIO APIs

```
/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PIN   */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinDirection_t                                       */
/*****************************************************************/
PinDirection_t DIO_setpindir(uint8_t u8_a_portid , uint8_t u8_a_pinid , uint8_t u8_a_pindir);

/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PORT  */
/* INPUT   : PORT , DIRECTION                                     */
/* RETURNS : PinDirection_t                                       */
/*****************************************************************/
PinDirection_t DIO_setportdir(uint8_t u8_a_portid , uint8_t u8_a_portdir);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE VALUE OF SPECIFIC PIN       */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinValue_t                                           */
/*****************************************************************/
PinValue_t DIO_setpinvalue(uint8_t u8_a_portid , uint8_t u8_a_pinid , uint8_t u8_a_pinval);

/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE VALUE OF SPECIFIC PORT      */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinValue_t                                           */
/*****************************************************************/
PinValue_t DIO_setportvalue(uint8_t u8_a_portid , uint8_t u8_a_portval);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO GET THE VALUE OF SPECIFIC PIN       */
/* INPUT   : PORTID , PINID , POINTER TO SET THE VALUE IN IT      */
/* RETURNS : PinRead_t                                            */
/*****************************************************************/
PinRead_t DIO_readpin(uint8_t u8_a_portid , uint8_t u8_a_pinid , uint8_t* u8_a_val);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO TOGGLE SPECIFIC PIN                 */
/* INPUT   : PORTID , PINID                                       */
/* RETURNS : PinRead_t                                            */
/*****************************************************************/
PinRead_t DIO_togglepin(uint8_t u8_a_portid , uint8_t u8_a_pinid );
```

4.1.2 Timer APIs

```c
typedef enum { VALID_INIT  , NOT_VALID_INIT} TMR0_init_error ;
/*************************************************************/
/** FUNCTION TO INITIALIZE TMR0 WITH SOME CONFIGURATIONS     */
/** ARGUMENTS  : VOID                                        */
/** RETURNS    : TMR0_init                                   */
/*************************************************************/
TMR0_init_error TMR0_init(void);


typedef enum {VALID_START , NOT_VALID_START } TMR0_start_error;
/*****************************************************************************/
/** FUNCTION TO LET TIMER 0 START WORK BY ASSIGN PRESCALLER OR CLOCK SOURCE   */
/** ARGUMENTS  : VOID                                                        */
/** RETURNS    : TMR0_start                                                  */
/*****************************************************************************/
TMR0_start_error TMR0_start(void);



typedef enum {VALID_STOP , NOT_VALID_STOP } TMR0_stop_error;
/*****************************************************************************/
/** FUNCTION TO STOP TIMER 0                                                 */
/** ARGUMENTS  : VOID                                                        */
/** RETURNS    : TMR0_stop                                                   */
/*****************************************************************************/
TMR0_stop_error TMR0_stop(void);


typedef enum {VALID_DELAY , NOT_VALID_DELAY } TMR0_delay_error ;
/*****************************************************************************/
/** FUNCTION TO SET DELAY USING TIMER 0                                      */
/** ARGUMENTS  : TAKES DELAY IN ms                                           */
/** RETURNS    : TMR0_delay                                                  */
/*****************************************************************************/
TMR0_delay_error TMR0_delayms(uint32_t u32_a_delayms);
```

4.1.3 Interrupt APIs

```
/*******************************************************/
/** FUNCTION TO SET THE GLOBAL INTERRUPT ENABLE FLAG    */
/** ARGUMENTS   : VOID                                  */
/** RETURNS     : VOID                                  */
/*******************************************************/
void SET_GLOBALINTERRUPT(void);

/*******************************************************/
/** FUNCTION TO INITIALIZE INT0                         */
/** ARGUMENTS   : VOID                                  */
/** RETURNS     : VOID                                  */
/*******************************************************/
void INT0_init(void);

/*******************************************************/
/** FUNCTION TO INITIALIZE INT1                         */
/** ARGUMENTS   : VOID                                  */
/** RETURNS     : VOID                                  */
/*******************************************************/
void INT1_init(void);

/*******************************************************/
/** FUNCTION TO INITIALIZE INT2                         */
/** ARGUMENTS   : VOID                                  */
/** RETURNS     : VOID                                  */
/*******************************************************/
void INT2_init(void);
```

4.2 HAL APIs

4.2.1 LCD APIs

```
/************************************************************/
/** FUNCTION TO INITIALIZE LCD                              */
/** ARGUMENTS    : VOID                                     */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_init(void);

/************************************************************/
/** FUNCTION TO SEND COMMAND TO LCD                         */
/** ARGUMENTS    : COMMAND                                  */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_sendcmd(uint8_t u8_a_cmd);

/************************************************************/
/** FUNCTION TO DISPLAY CHARACTER ON LCD                    */
/** ARGUMENTS    : CHARACTER                                */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_writechar(uint8_t u8_a_chr);

/************************************************************/
/** FUNCTION TO DISPLAY STRING ON LCD                       */
/** ARGUMENTS    : CHARACTER                                */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_writestr(uint8_t* u8_s_str);

/************************************************************/
/** FUNCTION TO JUMP TO SPECIFIC POSITION ON LCD            */
/** ARGUMENTS    : ROW , COLUMN (POSITION)                  */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status  LCD_goto(uint8_t u8_a_row , uint8_t u8_a_column);
```

```
/************************************************************/
/** FUNCTION TO CLEAR THE LCD                               */
/** ARGUMENTS    : ROW , COLUMN (POSITION)                  */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_clear(void);


/************************************************************/
/** FUNCTION TO WRITE INT ON THE LCD                        */
/** ARGUMENTS    : ROW , COLUMN (POSITION)                  */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_writeint(sint32_t s32_a_num);


/************************************************************/
/** FUNCTION TO WRITE CUSOMIZED CHARACTER ON THE LCD        */
/** ARGUMENTS    : PATTERN , OCATION IN CGRAM               */
/** RETURNS      : ERROR STATUS                             */
/************************************************************/
LCD_status LCD_writecustomchar(uint8_t * u8_a_pattern , uint8_t u8_a_location);
```

4.2.2 Keypad APIs

```
/*****************************************************************************/
/** @brief  Keypad initialization                                            */
/** @param  void                                                             */
/** @return KEYPAD_initSuccess                                               */
/** @return KEYPAD_initFail                                                  */
/*****************************************************************************/
KEYPAD_initError KEYPAD_init(void);

/*****************************************************************************/
/** @brief Reads keypad strokes                                              */
/** @param *u8_a_value refrence to store the read value in                   */
/** @return KEYPAD_readSuccess                                               */
/** @return KEYPAD_readFail                                                  */
/*****************************************************************************/
KEYPAD_readError KEYPAD_read(uint8_t *u8_a_value);
```

5-Drivers Description

5.1 DIO Driver

Configuration: Consist of 4 API's
Location: MCAL
Function: used to set pin direction (input or output), pin value (high or low) or read a value from a pin or toggle a pin

5.2 Timer Driver

Configuration: Consist of 5 API's
Location: MCAL
Function: used to set a time delay

5.3 Interrupt Driver

Configuration: Consist of 4 API's
Location: MCAL
Function: used to initialize external interrupts

5.4 Keypad Driver

Configuration: Consist of 2 API's
Location: HAL
Function: used to initialize the keypad, get pressed key

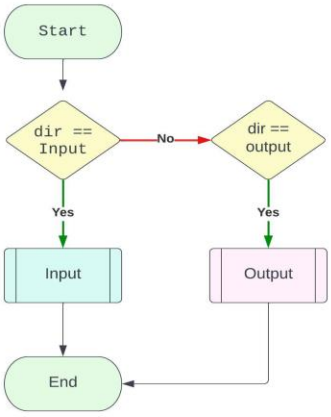
5.5 LCD Driver

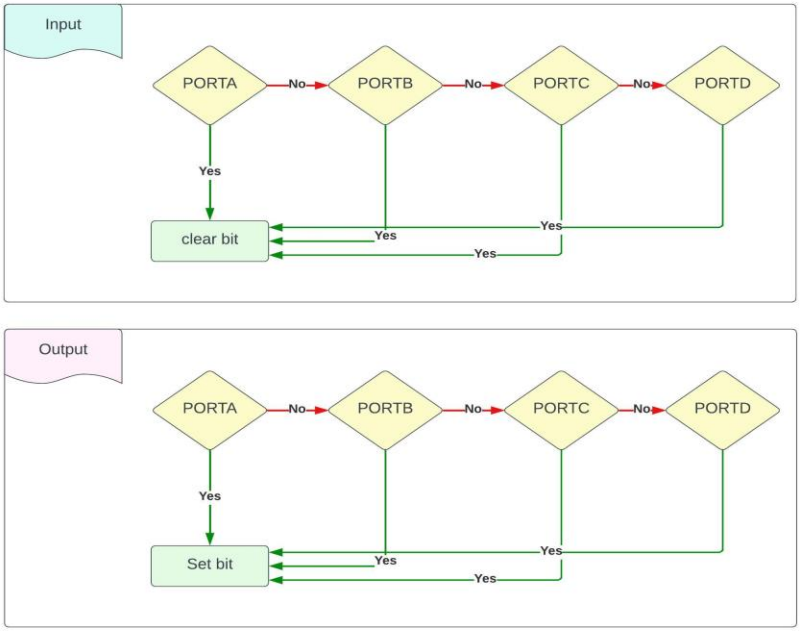
Configuration: Consist of 14 API's
Location: HAL
Function: used to initialize the LCD, send command to LCD & display character or string to LCD & jump to specific position on LCD & to clear the LCD & to wright integer or float number on the LCD

6 Low level design

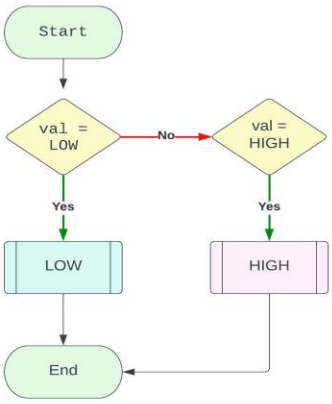
6.1 DIO APIs Flow charts

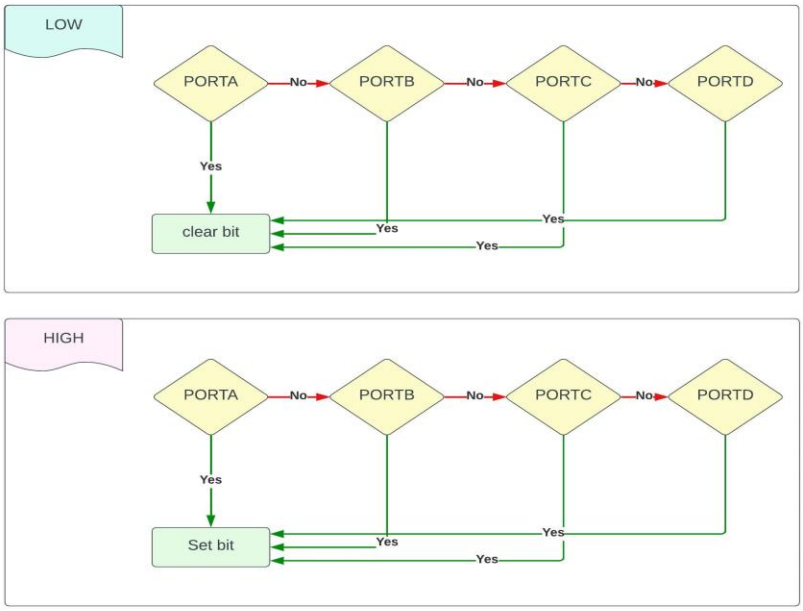
6.1.1 PinDirection_t DIO_setpindir (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pindir);





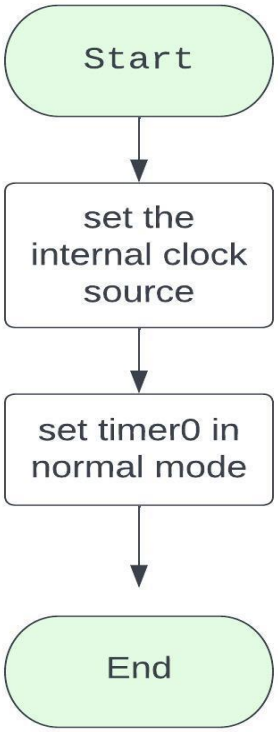
6.1.2 PinValue_t DIO_setpinvalue (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pinval);



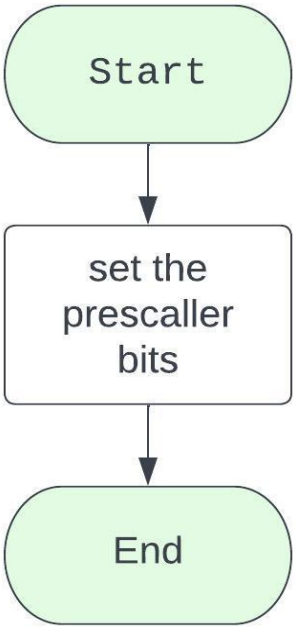


6.2 Timer0 APIs flowchart

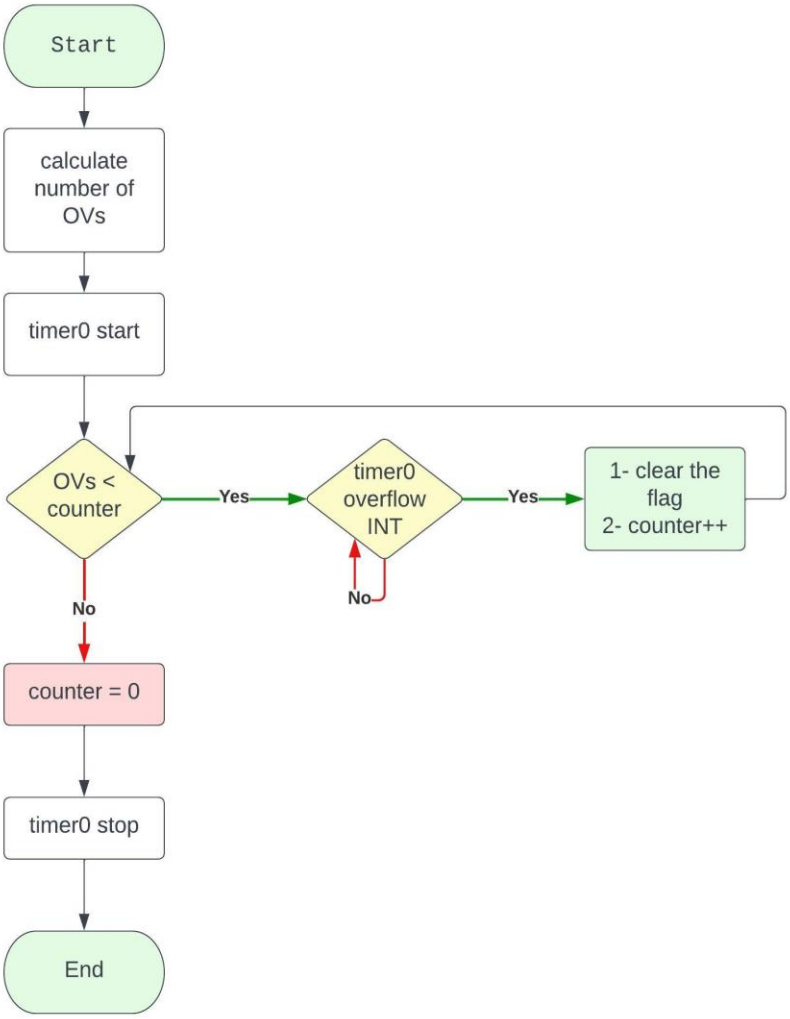
6.2.1 TMR0_init_error TMR0_init(void);



6.2.2 TMR0_start_error TMR0_start(void);

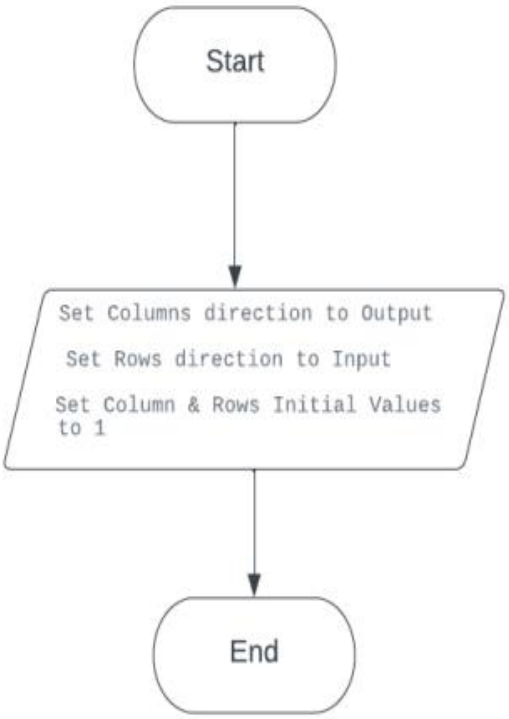


6.2.3 TMR0_delay_error TMR0_delayms(uint32_t u32_a_delayms);

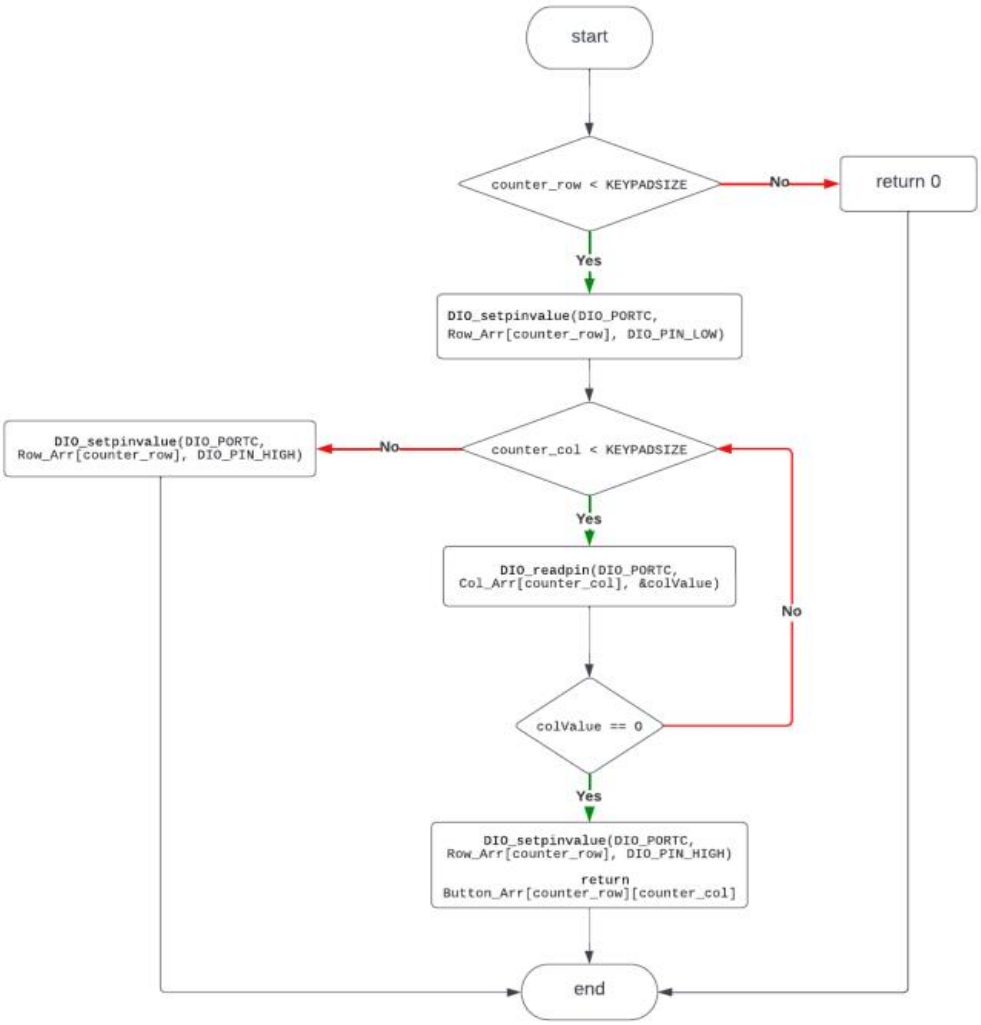


6.3 Keypad APIs flowchart

6.3.1 void KEYPAD_init(void) ;



6.3.2 uint8_t KEYPAD_getpressedkey(void) ;



6.4 Interrupt APIs flowchart

6.4.1 void INT0_init(void);