Air Conditioning System

Team 2

- 1- Mohab Ahmed
- 2- Anas Mahmoud
- 3- Mustafa Mohammed
- 4- Omar Taha

Table of Contents

| 1-Description | 3 |
|---------------------------|----|
| 1.1 Hardware components | 3 |
| 1.2 software Requirements | 3 |
| 2-Layered architecture | 4 |
| 3-System Flow Chart | 5 |
| 4-Schematic Capture | 6 |
| 5-Drivers Description | 7 |
| 5.1 DIO Driver | 7 |
| 5.2 Timer Driver | 7 |
| 5.3 ADC Driver | 7 |
| 5.4 Keypad Driver | 7 |
| 5.5 LCD Driver | 8 |
| 5.6 Application Driver | 8 |
| 6-API's | 9 |
| 6.1 DIO Driver | 9 |
| 6.2 Timer0 Driver | 9 |
| 6.3 Timer2 Driver | 9 |
| 6.4 ADC Driver | 10 |
| 6.5 Keypad Driver | 10 |
| 6.6 LCD Driver | 10 |
| 6.7 APP Driver | 11 |
| 7- API's Flow Chart | 12 |
| 7.1 LCD | 12 |
| 7.2 Keypad | 25 |
| 7.3 ADC | 27 |
| 7.4 buzzer | 29 |
| 7.5 App | 31 |

1-Description

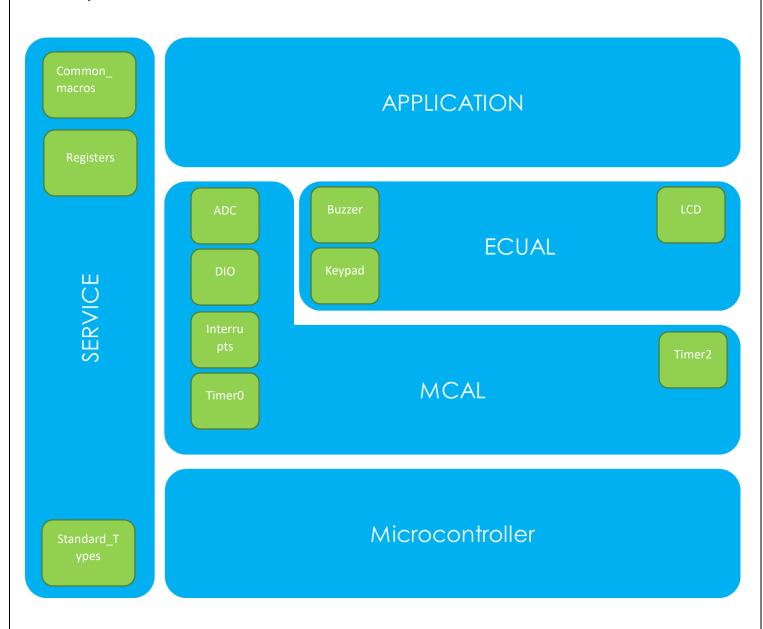
1.1 Hardware components

- 1. LCD (2*16)
- 2. Keypad (3 * 3) (Note: 4 buttons will be used)
- 3. Temperature sensor (LM35)
- 4. Buzzer

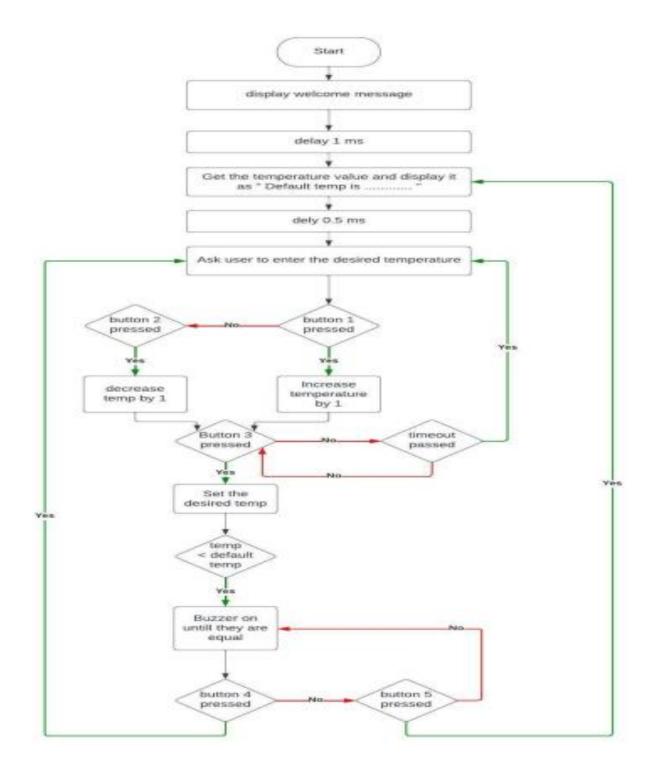
1.2 software Requirements

- 1. The system starts initially by welcoming the user
- 2. Then inform the user about the default temp and ask for desired one
- 3. Display range of temperature starts from min 18 and to max 35 C
- 4. To enter the desired temperature user has access through two buttons button one to increase and button two to decrease above or below the default temperature
- 5. Once button three is pressed the desired temperature will be set
- 6. If the desired temperature is greater than the default one buzzer will be on until both are equal
- 7. If button 4 is pressed at any time that will take the user back to set another desired temperature
- 8. Button 5 will be used to reset the system to start again

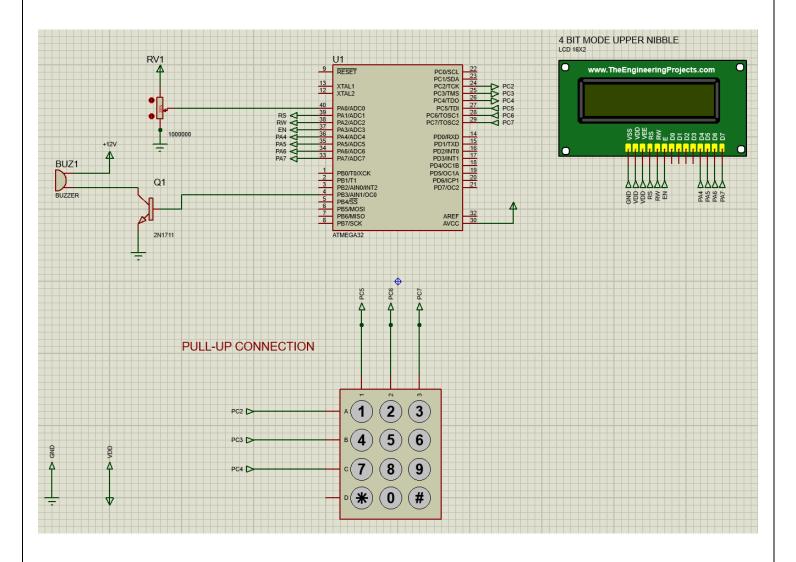
2-Layered architecture



3-System Flow Chart



4-Schematic Capture



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 ADC Driver

Configuration: Consist of 2 API's

Location: MCAL

Function: used to initialize ADC, read the value of ADC

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

5.6 Application Driver

Configuration: Consist of 9 API's

Location: App

Function: combine between the drivers API's to meet the requirement

6-API's

6.1 DIO Driver

```
1- PinDirection_t DIO_setpindir (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pinid);
2- PinValue_t DIO_setpinvalue (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pinid);
3- PinRead_t DIO_readpin (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t* u8_a_val);
4- PinRead_t DIO_togglepin (uint8_t u8_a_portid, uint8_t u8_a_pinid);
```

6.2 Timer0 Driver

```
1- TMR0_init_error TMR0_init(void);
2- TMR0_start_error TMR0_start(void);
3- TMR0_stop_error TMR0_stop(void);
4- TMR0_delay_error TMR0_delayms(uint32_t u32_a_delayms);
5- TMR0_delay_error TMR0_delaymicos(uint32_t u32_a_delaymicros);
```

6.3 Timer2 Driver

```
1- err_state TIMER2_normalMode(void);
2- err_state TIMER2_initialValue(uint8_t value);
3- err_state TIMER2_perscalerMode(unsigned int prescaler);
4- err_state TIMER2_delay(float f_a_delayInMillis);
5- unsigned int TIMER2_getInitialValue(float f_a_delayInMillis);
```

```
6.4 ADC Driver
1- ADC initstatus ADC Init(void);
2- uint16 t ADC read(void);
6.5 Keypad Driver
1- void KEYPAD init(void);
2- uint8 t KEYPAD getpressedkey(void);
6.6 LCD Driver
1- LCD init error LCD 8 bit init (void);
2- LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);
3- LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);
4- LCD init error LCD 4 bit init(void);
5- LCD sendCommand error LCD 4 bit sendCommand(uint8 t u8 a command);
6- LCD sendChar error LCD 4 bit sendChar(uint8 t u8 a char);
7- LCD_sendString_error LCD_sendString(uint8_t *u8_a_string);
8- void LCD goTo(uint8 t u8 a row, uint8 t u8 a column);
9- void LCD createCustomCharacter(uint8 t *u8 a bitMap,uint8 t u8 a location)
10- LCD init error LCD init(void);
11- LCD_sendCommand_error LCD_sendCommand(uint8 t u8 a command);
12- LCD sendChar error LCD sendChar(uint8 t u8 a char);
13- LCD_sendChar_error LCD_sendFloat(float f_a_number);
14- LCD_sendChar_error LCD_sendInteger(uint16_t u16_a_number);
```

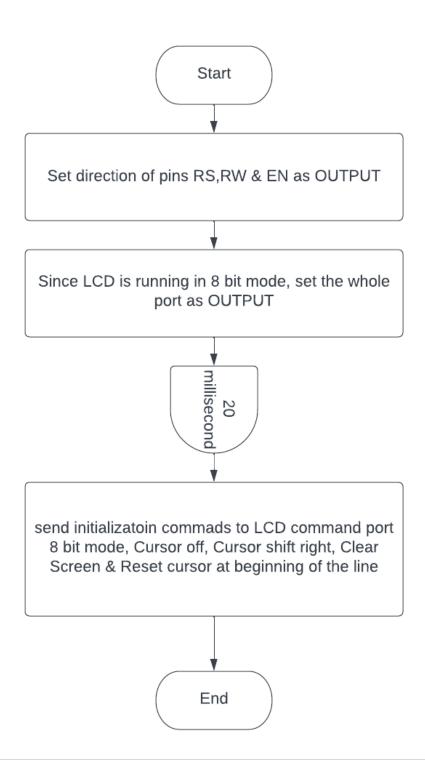
6.7 APP Driver

```
1- APP_initError APP_init(void);
2-void APP_printString(uint8_t u8_a_row, uint8_t u8_a_column, uint8_t *u8_a_str);
3-void APP_printChar(uint8_t u8_a_row, uint8_t u8_a_column, uint8_t u8_a_char);
4-void APP_printInteger(uint8_t u8_a_row, uint8_t u8_a_column, uint16_t u16_a_integer);
5-void APP_setTemp();
6-void APP_decrementBar();
7-void APP_incrementBar();
8-void APP_defaultView(void);
9- void get_current_overflow(void);
```

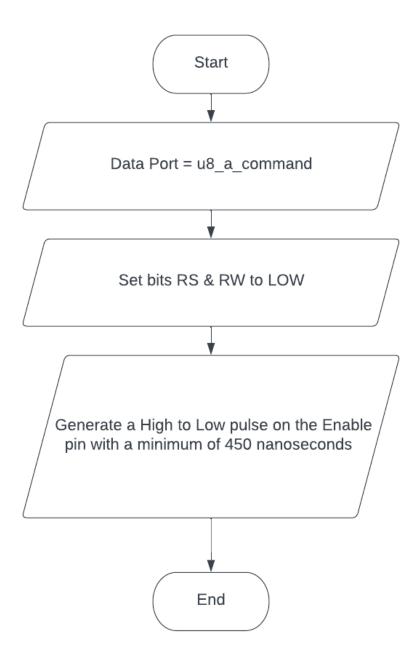
7- API's Flow Chart

7.1 LCD

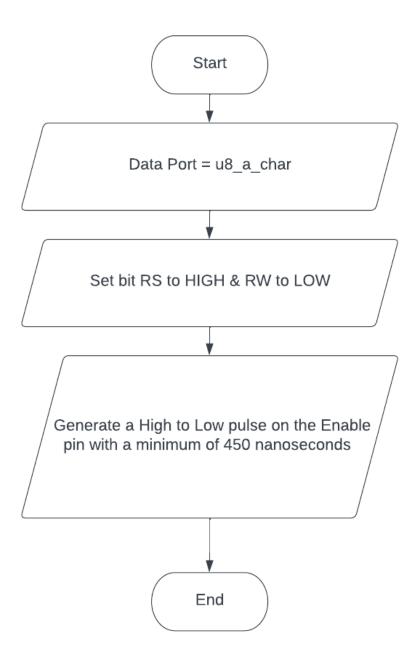
LCD_init_error LCD_8_bit_init(void);



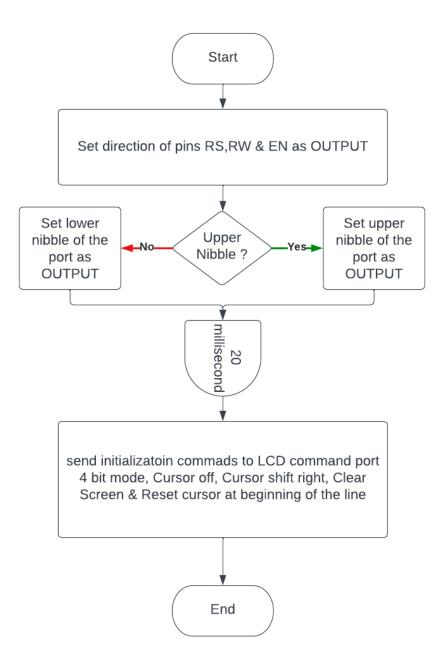
LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);



LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);



LCD_init_error LCD_4_bit_init(void);



LCD_sendCommand_error LCD_4_bit_sendCommand(uint8_t u8_a_command); Start Upper LCD_Data_Port = (LCD_Data_Port & 0xF0) LCD_Data_Port = (LCD_Data_Port & 0x0F) | No-Yes-Nibble? | ((u8_a_command & 0xF0) >> 4) ((u8_a_command & 0xF0)) Set bits RS & RW to LOW Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds **Upper** LCD_Data_Port = (LCD_Data_Port & 0xF0) LCD_Data_Port = (LCD_Data_Port & 0x0F) | Nibble? ((u8_a_command & 0x0F)) ((u8_a_command & 0x0F) << 4) Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds End

LCD_sendChar_error LCD_4_bit_sendChar(uint8_t u8_a_char); Start Upper LCD_Data_Port = (LCD_Data_Port & 0xF0) LCD_Data_Port = (LCD_Data_Port & 0x0F) | Nibble? | ((u8_a_char & 0xF0) >> 4) ((u8_a_char & 0xF0)) Set bits RS to HIGH & RW to LOW Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds **Upper** LCD_Data_Port = (LCD_Data_Port & 0xF0) LCD_Data_Port = (LCD_Data_Port & 0x0F) | No-Nibble? ((u8_a_char & 0x0F)) ((u8_a_char & 0x0F) << 4) Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds End

LCD_sendString_error LCD_sendString(uint8_t *u8_a_string); Start 4 bit mode? Yes-LCD_4_bit_sendChar(u8_a_string[u16_l_charCount]) LCD_8_bit_sendChar(u8_a_string[u16_l_charCount]) u8_a_string[u16_l_charCount!=null Νo End

void LCD_goTo(uint8_t u8_a_row,uint8_t u8_a_column); Start uint8_t[]={Cursor_Reset_Line1,Cursor_Reset_Line2} LCD_8_bit_sendCommand(u8_I_rows[u8_a_row]+u8_a_coloumn) 4 bit mode? LCD_4_bit_sendCommand(u8_I_rows[u8_a_row]+u8_a_coloumn) End **19** | Page

void LCD_createCustomCharacter(uint8_t *u8_a_bitMap,uint8_t u8_a_location); Start u8_I_count=0 4 bit mode? LCD_8_bit_sendCommand(CGRAM + (u8_a_location*8)) LCD_4_bit_sendCommand(CGRAM+(u8_a_location*8)); u8_I_count<8? u8_I_count<8? -No--No-Yes Yes LCD_8_bit_sendChar(u8_a_bitMap[u8_l_count]) End

LCD_init_error LCD_init(void); Start 4 bit mode? End **21** | Page

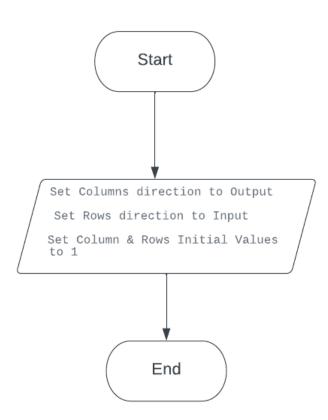
LCD_sendCommand_error LCD_sendCommand(uint8_t u8_a_command); Start 4 bit mode? LCD_8_bit_sendCommand(u8_a_command) -Yes► LCD_4_bit_sendCommand(u8_a_command) End **22 |** Page

LCD_sendChar_error LCD_sendChar(uint8_t u8_a_char); Start 4 bit mode? LCD_4_bit_sendChar(u8_a_char) -Yes**⊳** End **23** | Page

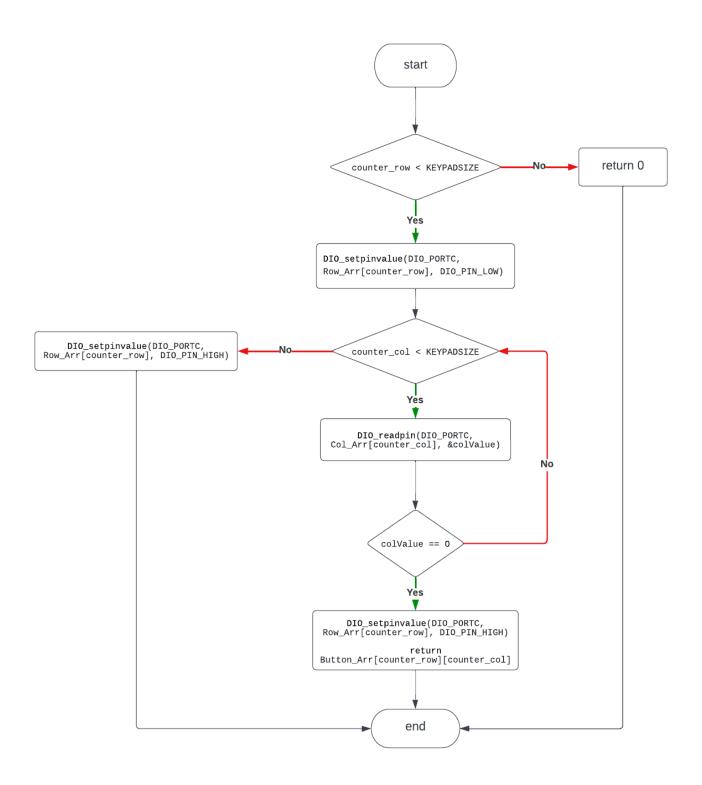
sendChar_error LCD_sendInteger(uint16_t u16_a_number); Start uint16_t u_l_toString[10]; uint8_t i =0; $uint8_t j = 0;$ LCD_sendChar('0'); -u16_a_number==0? No j>0 (u16_a_number>0)-Noj=i Yes u_l_toString[i]=(u16_a_number % 10) +48; u16_a_number/=10; Νo End

7.2 Keypad

void KEYPAD_init(void)

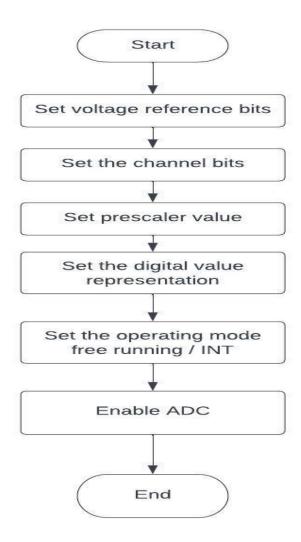


uint8_t KEYPAD_getpressedkey(void)

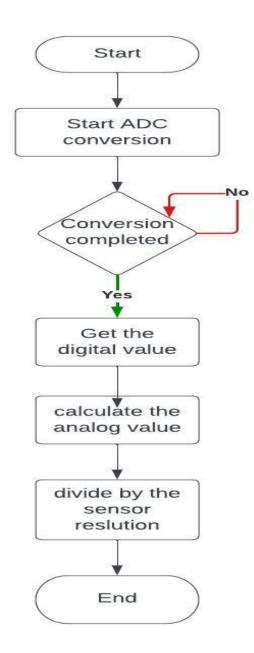


7.3 ADC

ADC_initstatus ADC_init(void);

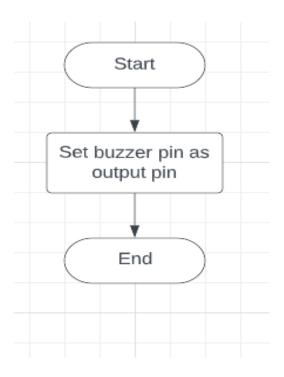


uint32_t ADC_read(void);

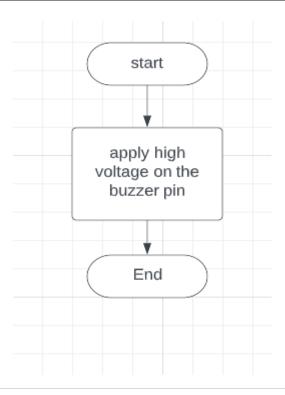


7.4 buzzer

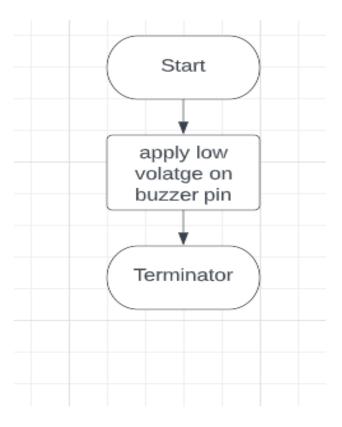
void BUZZ_init();



void BUZZ_on();

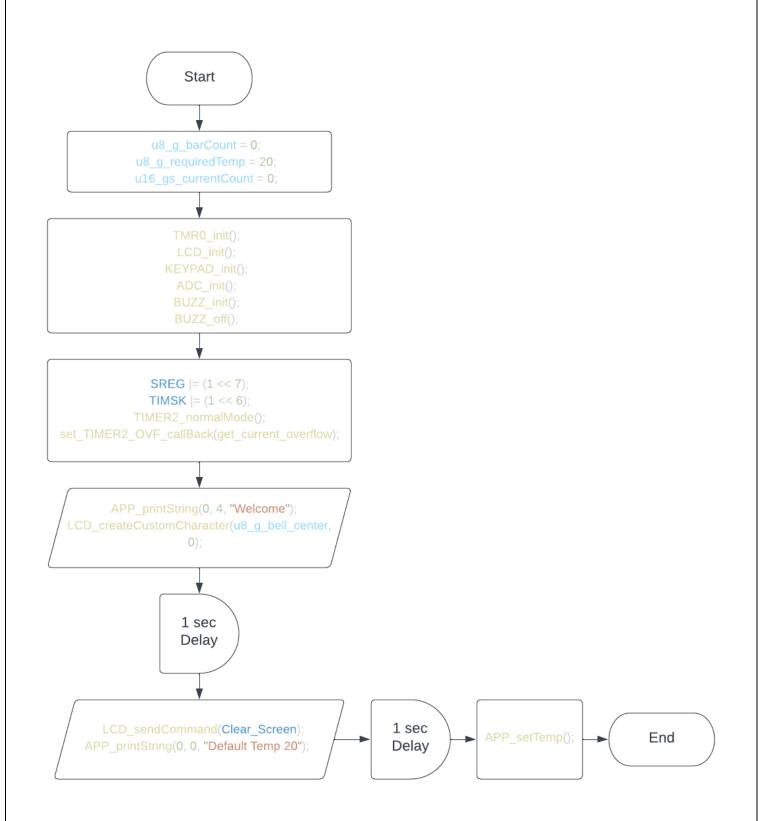


void BUZZ_off();

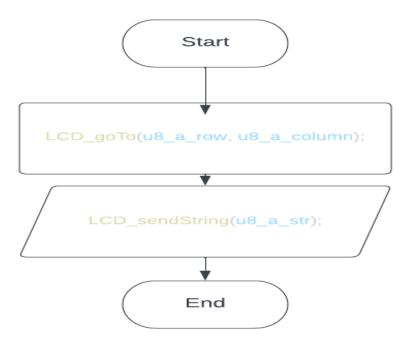


7.5 App

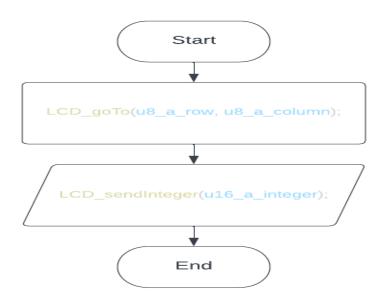
initError APP_init(void);



void APP_printString(uint8_t u8_a_row, uint8_t u8_a_column, uint8_t *u8_a_str);

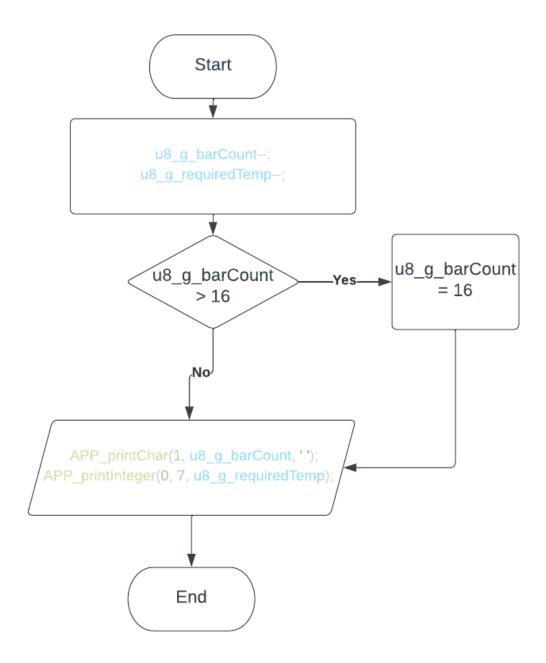


printInteger(uint8_t u8_a_row, uint8_t u8_a_column, uint16_t u16_a_integer);



void APP_incrementBar(); Start APP_printChar(1, u8_g_barCount, '|'); u8_g_barCount++; u8_g_requiredTemp++; u8_g_barCount APP_printInteger(0, 7, u8_g_barCount = 16 u8_g_requiredTemp); > 16 Nο End

void APP_decrementBar();



void APP_setTemp(); Start . Yes uint16_t u16_l_ovf; uint8_t keypadValue; == '1' == '1' LCD_sendCommand(Clear_Screen); APP_printString(0, 0, "Set Temp") 0.5 sec Delay ¥ Yes _CD_sendCommand(Clear_Screen); APP_printString(0, 0, "Min=18"); (keypadValue != '3') && u8_g_requiredTemp); APP_printString(0, 0, "Temp set to APP_printString(0, 10, "Max=35") TIMER2_perscalerMode(TIMER_OFF_); > "); TCNT2 = 0; APP_printInteger(0, 14, TIMER2_perscalerMode(PRESCALER_MODE) 18) 1 Sec Delay Yes u8_g_barCount++; End ${\color{red} LCD_sendCommand(Clear_Screen);}$

void APP_defaultView(void);

