Digital Lab 4:

Experiment 6:

EEPROM

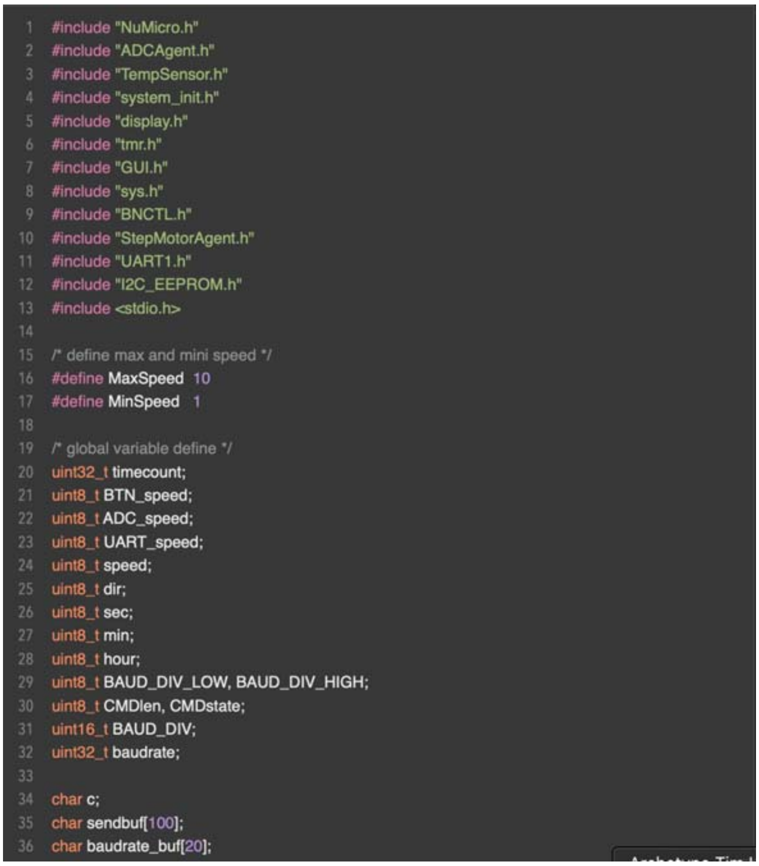
Date: 2024/05/07

Class: 電機三全英班

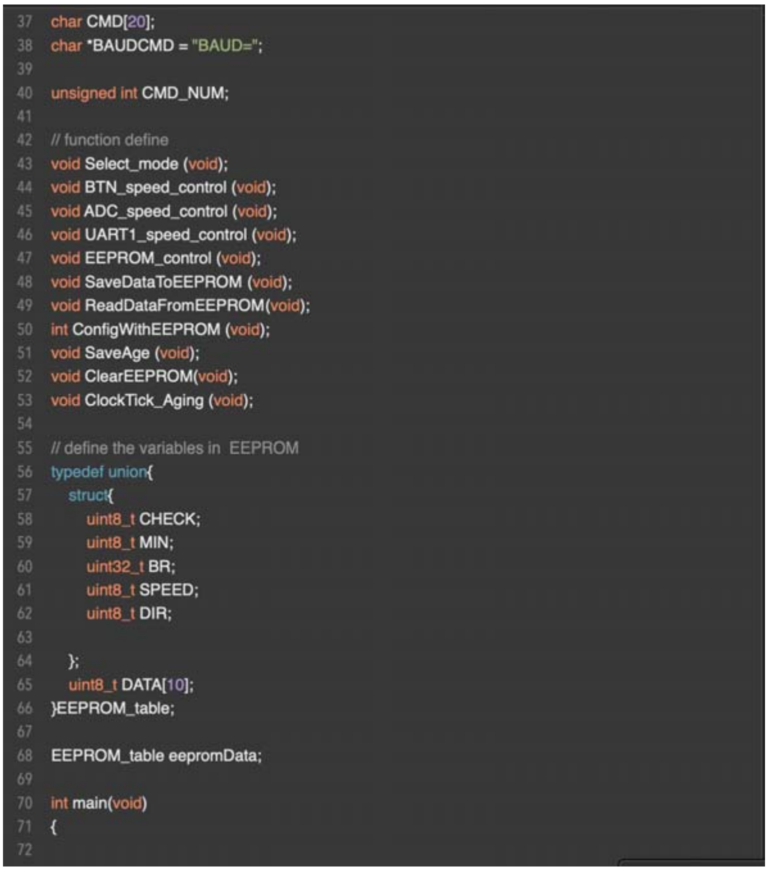
Group: Group 11

Name: B103105006 胡庭翊

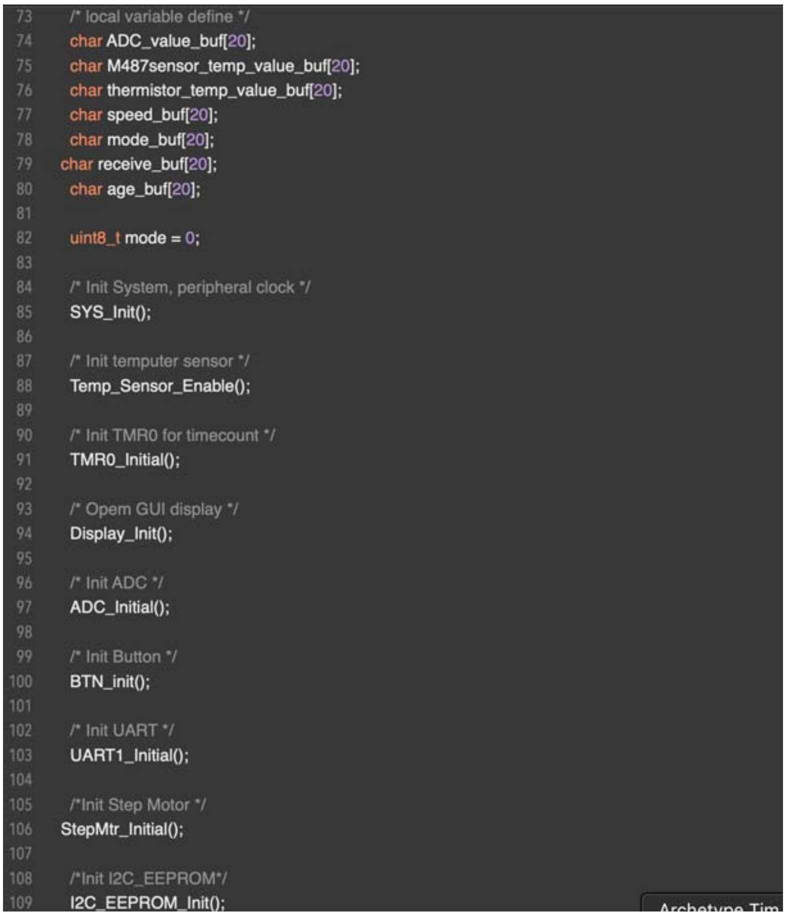
I. Annotated Code



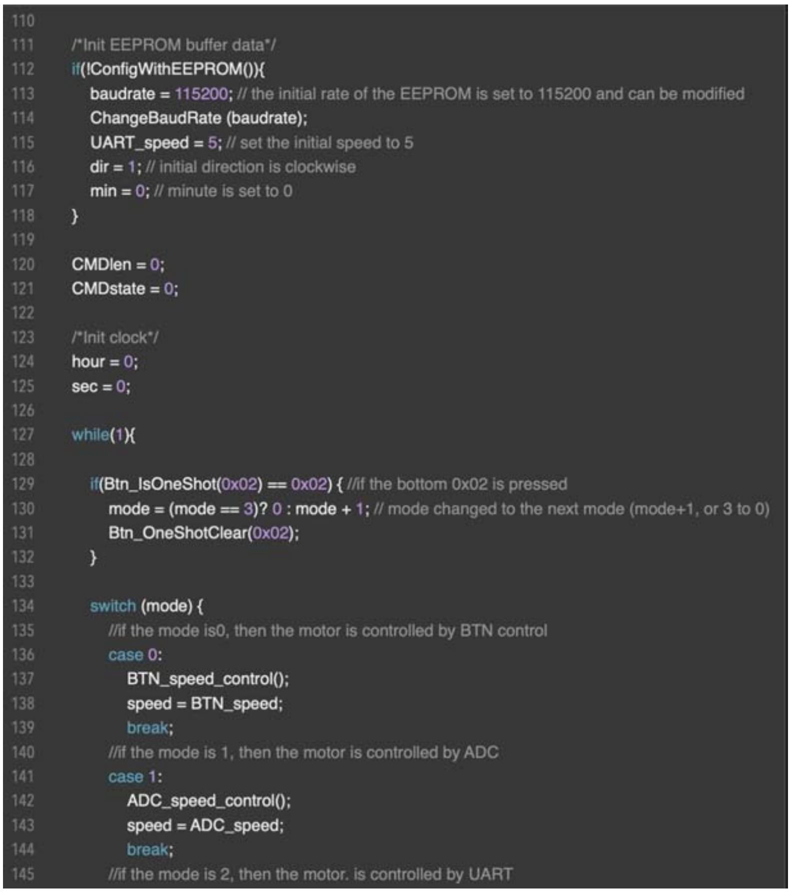
```c
#include "NuMicro.h"
#include "ADCAgent.h"
#include "TempSensor.h"
#include "system_init.h"
#include "display.h"
#include "tmr.h"
#include "GUI.h"
#include "sys.h"
#include "BNCTL.h"
#include "StepMotorAgent.h"
#include "UART1.h"
#include "I2C_EEPROM.h"
#include <stdio.h>

/* define max and mini speed */
#define MaxSpeed  10
#define MinSpeed   1

/* global variable define */
uint32_t timecount;
uint8_t BTN_speed;
uint8_t ADC_speed;
uint8_t UART_speed;
uint8_t speed;
uint8_t dir;
uint8_t sec;
uint8_t min;
uint8_t hour;
uint8_t BAUD_DIV_LOW, BAUD_DIV_HIGH;
uint8_t CMDlen, CMDstate;
uint16_t BAUD_DIV;
uint32_t baudrate;

char c;
char sendbuf[100];
char baudrate_buf[20];
```



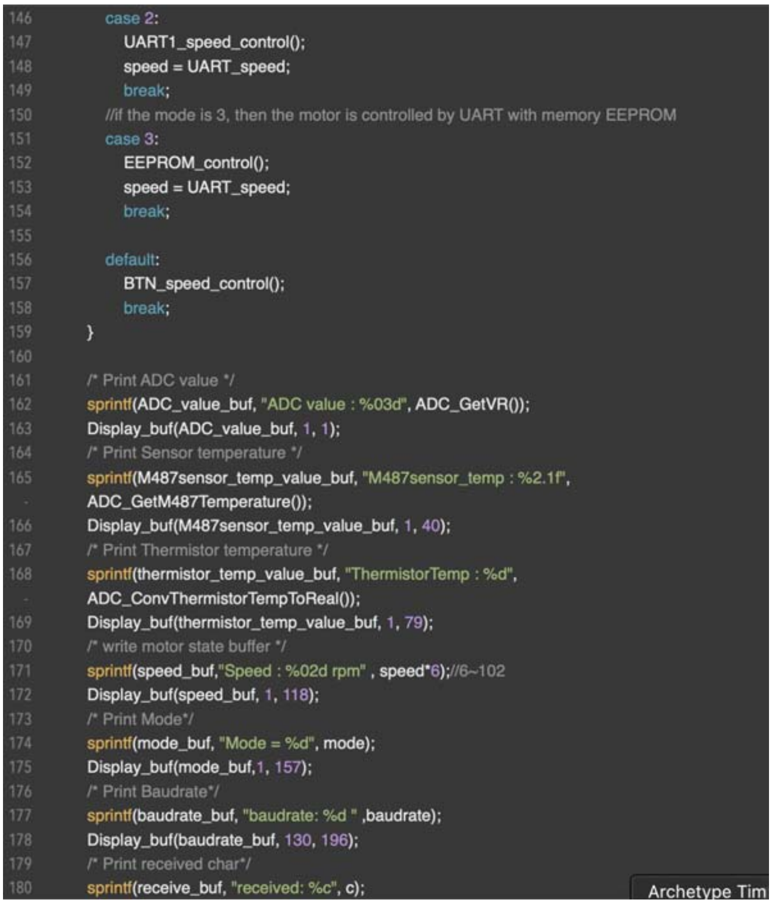
```c
char CMD[20];
char *BAUDCMD = "BAUD=";

unsigned int CMD_NUM;

// function define
void Select_mode (void);
void BTN_speed_control (void);
void ADC_speed_control (void);
void UART1_speed_control (void);
void EEPROM_control (void);
void SaveDataToEEPROM (void);
void ReadDataFromEEPROM(void);
int ConfigWithEEPROM (void);
void SaveAge (void);
void ClearEEPROM(void);
void ClockTick_Aging (void);

// define the variables in  EEPROM
typedef union{
   struct{
      uint8_t CHECK;
      uint8_t MIN;
      uint32_t BR;
      uint8_t SPEED;
      uint8_t DIR;

   };
   uint8_t DATA[10];
}EEPROM_table;

EEPROM_table eepromData;

int main(void)
{

```



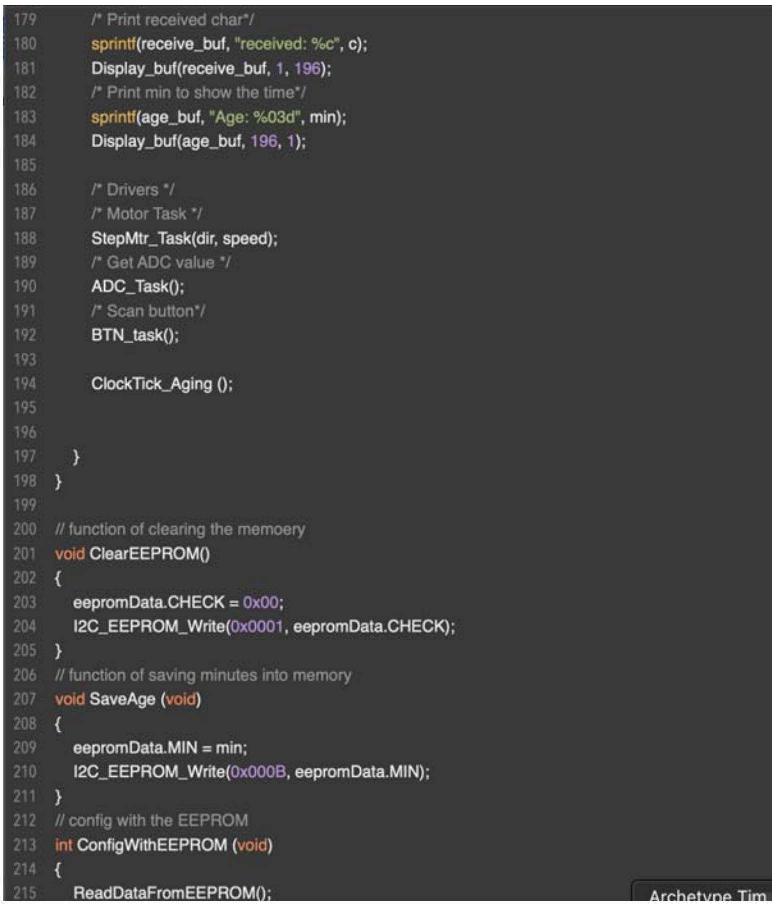
```c
    /* local variable define */
    char ADC_value_buf[20];
    char M487sensor_temp_value_buf[20];
    char thermistor_temp_value_buf[20];
    char speed_buf[20];
    char mode_buf[20];
   char receive_buf[20];
    char age_buf[20];

    uint8_t mode = 0;

    /* Init System, peripheral clock */
    SYS_Init();

    /* Init temputer sensor */
    Temp_Sensor_Enable();

    /* Init TMR0 for timecount */
    TMR0_Initial();

    /* Opem GUI display */
    Display_Init();

    /* Init ADC */
    ADC_Initial();

    /* Init Button */
    BTN_init();

    /* Init UART */
    UART1_Initial();

    /*Init Step Motor */
   StepMtr_Initial();

    /*Init I2C_EEPROM*/
    I2C_EEPROM_Init();
```



```c

    /*Init EEPROM buffer data*/
    if(!ConfigWithEEPROM()){
        baudrate = 115200; // the initial rate of the EEPROM is set to 115200 and can be modified
        ChangeBaudRate (baudrate);
        UART_speed = 5; // set the initial speed to 5
        dir = 1; // initial direction is clockwise
        min = 0; // minute is set to 0
    }

    CMDlen = 0;
    CMDstate = 0;

    /*Init clock*/
    hour = 0;
    sec = 0;

    while(1){

        if(Btn_IsOneShot(0x02) == 0x02) { //if the bottom 0x02 is pressed
            mode = (mode == 3)? 0 : mode + 1; // mode changed to the next mode (mode+1, or 3 to 0)
            Btn_OneShotClear(0x02);
        }

        switch (mode) {
            //if the mode is0, then the motor is controlled by BTN control
            case 0:
                BTN_speed_control();
                speed = BTN_speed;
                break;
            //if the mode is 1, then the motor is controlled by ADC
            case 1:
                ADC_speed_control();
                speed = ADC_speed;
                break;
            //if the mode is 2, then the motor. is controlled by UART
```

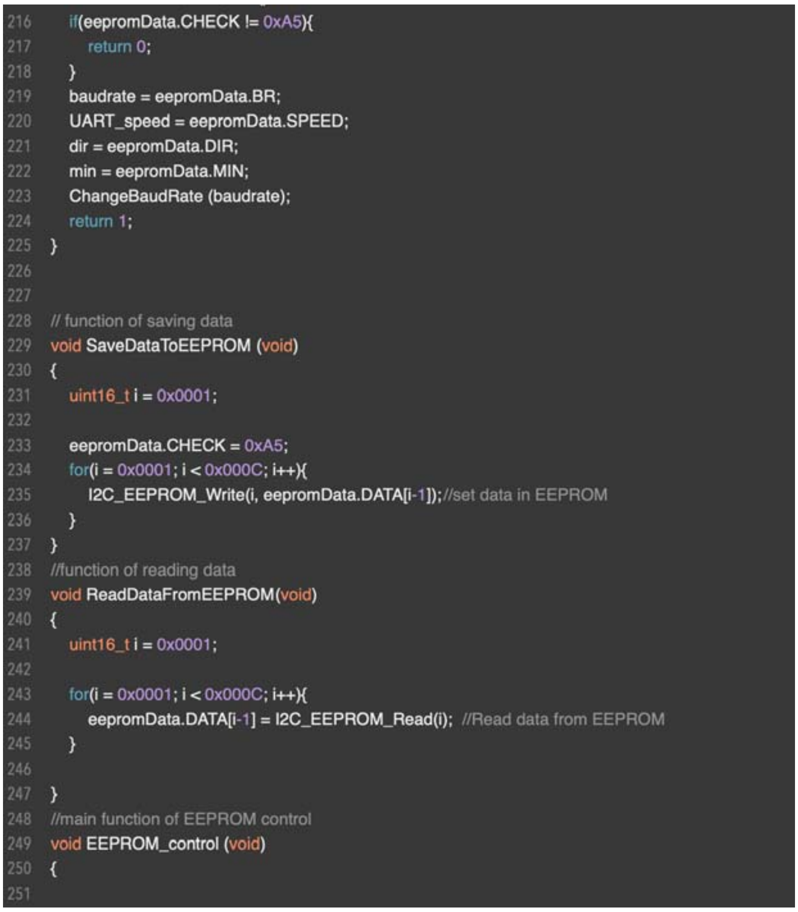


```
            case 2:
                UART1_speed_control();
                speed = UART_speed;
                break;
            //if the mode is 3, then the motor is controlled by UART with memory EEPROM
            case 3:
                EEPROM_control();
                speed = UART_speed;
                break;

            default:
                BTN_speed_control();
                break;
        }

        /* Print ADC value */
        sprintf(ADC_value_buf, "ADC value : %03d", ADC_GetVR());
        Display_buf(ADC_value_buf, 1, 1);
        /* Print Sensor temperature */
        sprintf(M487sensor_temp_value_buf, "M487sensor_temp : %2.1f",
        ADC_GetM487Temperature());
        Display_buf(M487sensor_temp_value_buf, 1, 40);
        /* Print Thermistor temperature */
        sprintf(thermistor_temp_value_buf, "ThermistorTemp : %d",
        ADC_ConvThermistorTempToReal());
        Display_buf(thermistor_temp_value_buf, 1, 79);
        /* write motor state buffer */
        sprintf(speed_buf,"Speed : %02d rpm" , speed*6);//6~102
        Display_buf(speed_buf, 1, 118);
        /* Print Mode*/
        sprintf(mode_buf, "Mode = %d", mode);
        Display_buf(mode_buf,1, 157);
        /* Print Baudrate*/
        sprintf(baudrate_buf, "baudrate: %d " ,baudrate);
        Display_buf(baudrate_buf, 130, 196);
        /* Print received char*/
        sprintf(receive_buf, "received: %c", c);
```

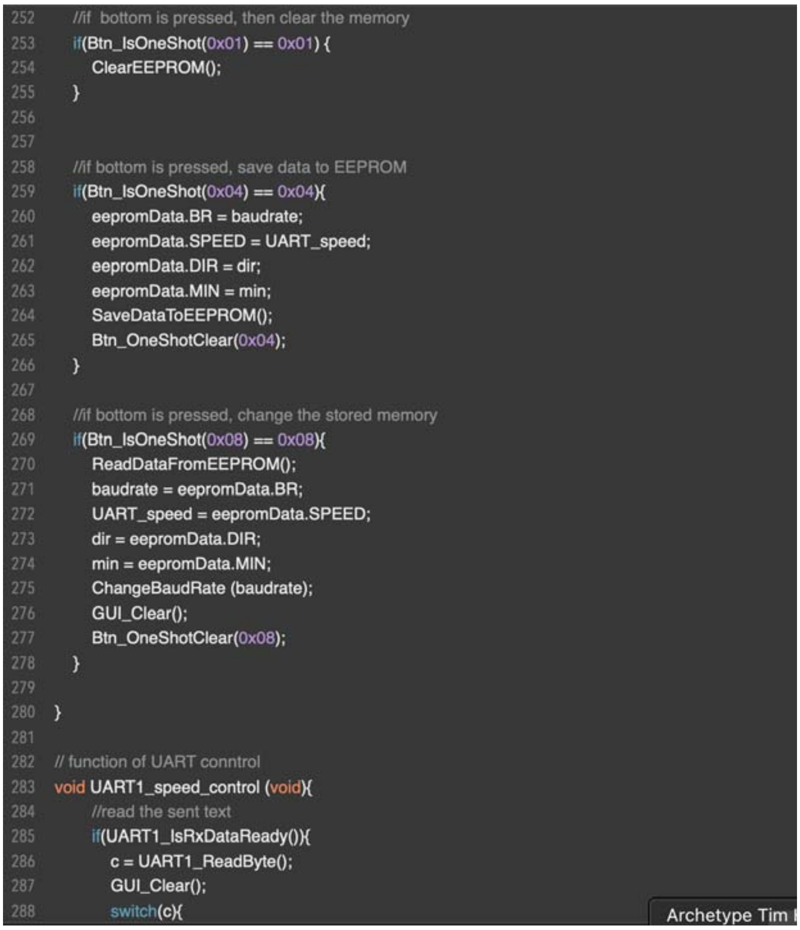


```
        /* Print received char*/
        sprintf(receive_buf, "received: %c", c);
```

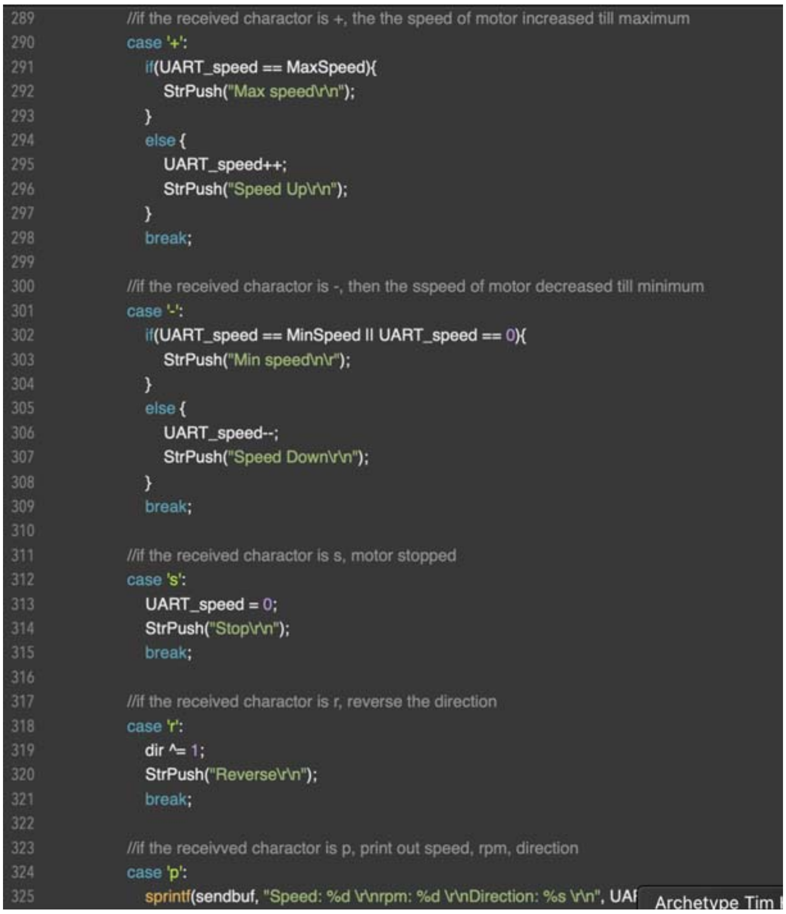
```
        Display_buf(receive_buf, 1, 196);
        /* Print min to show the time*/
        sprintf(age_buf, "Age: %03d", min);
        Display_buf(age_buf, 196, 1);

        /* Drivers */
        /* Motor Task */
        StepMtr_Task(dir, speed);
        /* Get ADC value */
        ADC_Task();
        /* Scan button*/
        BTN_task();

        ClockTick_Aging ();


    }
}

// function of clearing the memoery
void ClearEEPROM()
{
    eepromData.CHECK = 0x00;
    I2C_EEPROM_Write(0x0001, eepromData.CHECK);
}
// function of saving minutes into memory
void SaveAge (void)
{
    eepromData.MIN = min;
    I2C_EEPROM_Write(0x000B, eepromData.MIN);
}
// config with the EEPROM
int ConfigWithEEPROM (void)
{
    ReadDataFromEEPROM();
```



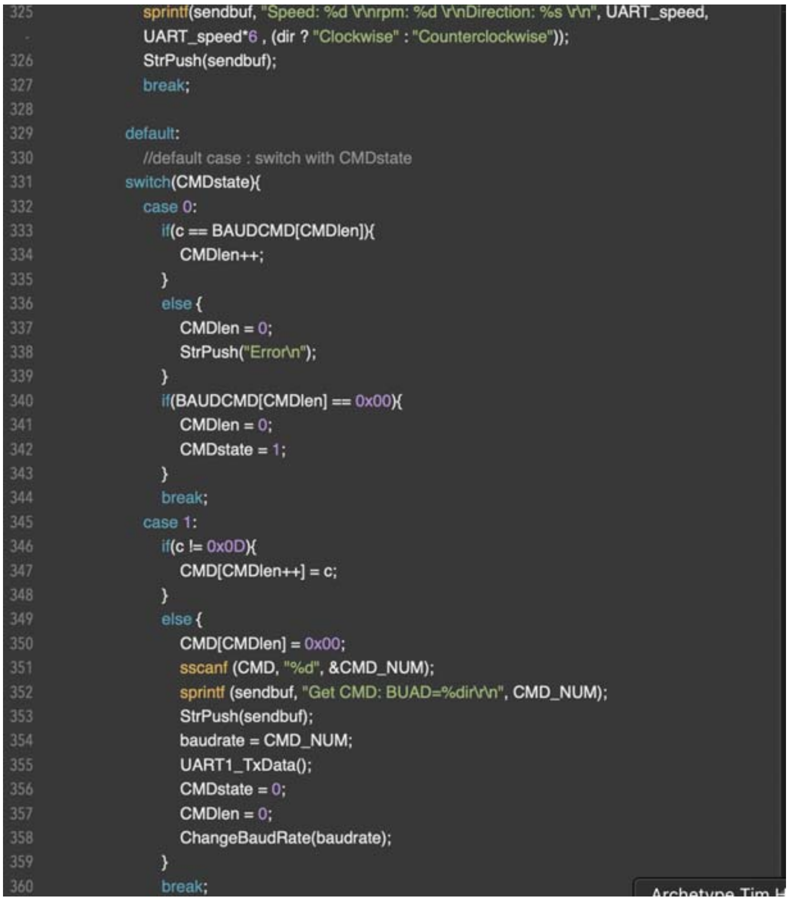
```c
    if(eepromData.CHECK != 0xA5){
        return 0;
    }
    baudrate = eepromData.BR;
    UART_speed = eepromData.SPEED;
    dir = eepromData.DIR;
    min = eepromData.MIN;
    ChangeBaudRate (baudrate);
    return 1;
}


// function of saving data
void SaveDataToEEPROM (void)
{
    uint16_t i = 0x0001;

    eepromData.CHECK = 0xA5;
    for(i = 0x0001; i < 0x000C; i++){
        I2C_EEPROM_Write(i, eepromData.DATA[i-1]);//set data in EEPROM
    }
}
//function of reading data
void ReadDataFromEEPROM(void)
{
    uint16_t i = 0x0001;

    for(i = 0x0001; i < 0x000C; i++){
        eepromData.DATA[i-1] = I2C_EEPROM_Read(i);  //Read data from EEPROM
    }

}
//main function of EEPROM control
void EEPROM_control (void)
{

```



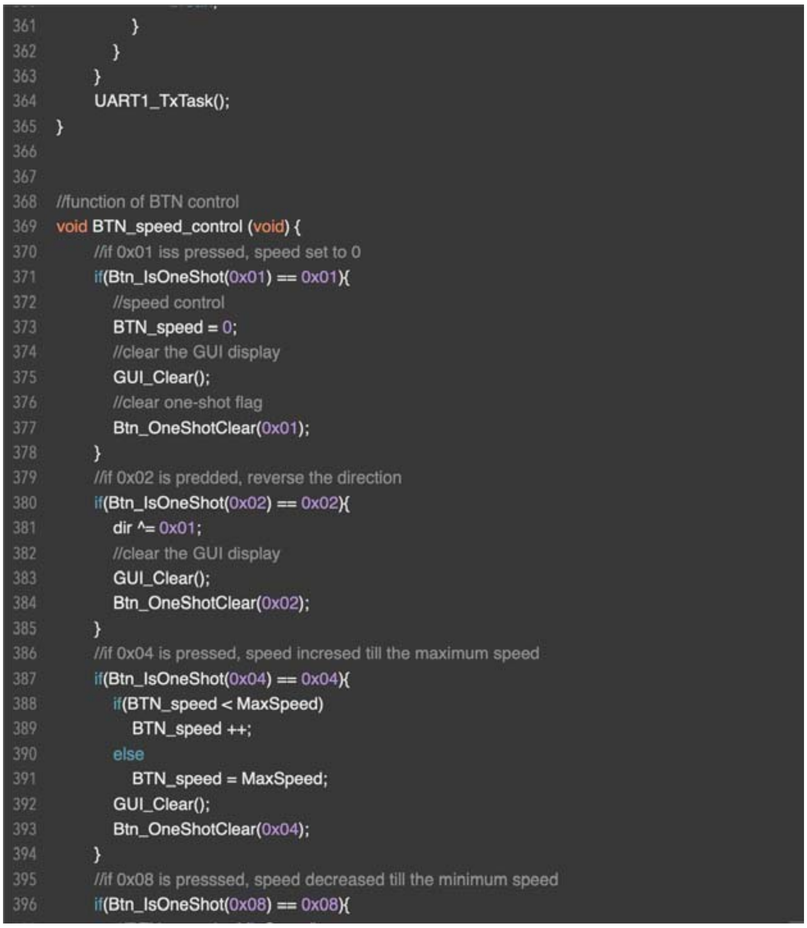
```c
    //if  bottom is pressed, then clear the memory
    if(Btn_IsOneShot(0x01) == 0x01) {
        ClearEEPROM();
    }


    //if bottom is pressed, save data to EEPROM
    if(Btn_IsOneShot(0x04) == 0x04){
        eepromData.BR = baudrate;
        eepromData.SPEED = UART_speed;
        eepromData.DIR = dir;
        eepromData.MIN = min;
        SaveDataToEEPROM();
        Btn_OneShotClear(0x04);
    }

    //if bottom is pressed, change the stored memory
    if(Btn_IsOneShot(0x08) == 0x08){
        ReadDataFromEEPROM();
        baudrate = eepromData.BR;
        UART_speed = eepromData.SPEED;
        dir = eepromData.DIR;
        min = eepromData.MIN;
        ChangeBaudRate (baudrate);
        GUI_Clear();
        Btn_OneShotClear(0x08);
    }

}

// function of UART conntrol
void UART1_speed_control (void){
        //read the sent text
        if(UART1_IsRxDataReady()){
            c = UART1_ReadByte();
            GUI_Clear();
            switch(c){
```



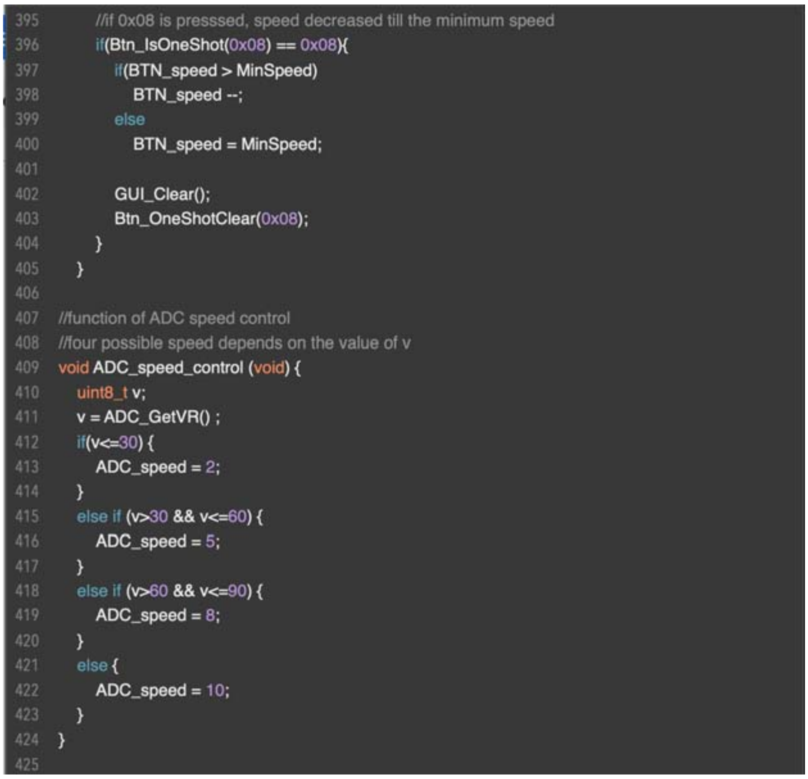
```c
//if the received charactor is +, the the speed of motor increased till maximum
case '+':
    if(UART_speed == MaxSpeed){
        StrPush("Max speed\r\n");
    }
    else {
        UART_speed++;
        StrPush("Speed Up\r\n");
    }
    break;

//if the received charactor is -, then the sspeed of motor decreased till minimum
case '-':
    if(UART_speed == MinSpeed || UART_speed == 0){
        StrPush("Min speed\n\r");
    }
    else {
        UART_speed--;
        StrPush("Speed Down\r\n");
    }
    break;

//if the received charactor is s, motor stopped
case 's':
    UART_speed = 0;
    StrPush("Stop\r\n");
    break;

//if the received charactor is r, reverse the direction
case 'r':
    dir ^= 1;
    StrPush("Reverse\r\n");
    break;

//if the receivved charactor is p, print out speed, rpm, direction
case 'p':
    sprintf(sendbuf, "Speed: %d \r\nrpm: %d \r\nDirection: %s \r\n", UAF
```



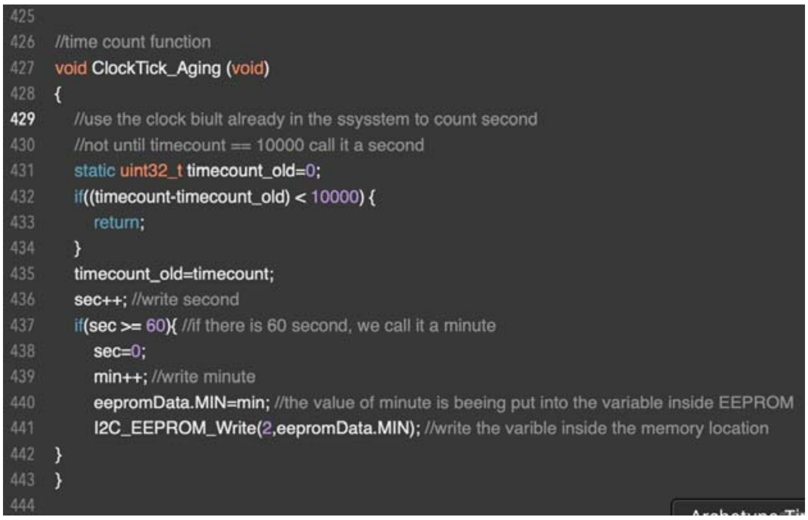
```c
    sprintf(sendbuf, "Speed: %d \r\nrpm: %d \r\nDirection: %s \r\n", UART_speed,
    UART_speed*6 , (dir ? "Clockwise" : "Counterclockwise"));
    StrPush(sendbuf);
    break;

default:
    //default case : switch with CMDstate
switch(CMDstate){
    case 0:
        if(c == BAUDCMD[CMDlen]){
            CMDlen++;
        }
        else {
            CMDlen = 0;
            StrPush("Error\n");
        }
        if(BAUDCMD[CMDlen] == 0x00){
            CMDlen = 0;
            CMDstate = 1;
        }
        break;
    case 1:
        if(c != 0x0D){
            CMD[CMDlen++] = c;
        }
        else {
            CMD[CMDlen] = 0x00;
            sscanf (CMD, "%d", &CMD_NUM);
            sprintf (sendbuf, "Get CMD: BUAD=%dir\r\n", CMD_NUM);
            StrPush(sendbuf);
            baudrate = CMD_NUM;
            UART1_TxData();
            CMDstate = 0;
            CMDlen = 0;
            ChangeBaudRate(baudrate);
        }
        break;
```



```
        }
      }
    }
    UART1_TxTask();
}


//function of BTN control
void BTN_speed_control (void) {
    //if 0x01 iss pressed, speed set to 0
    if(Btn_IsOneShot(0x01) == 0x01){
      //speed control
      BTN_speed = 0;
      //clear the GUI display
      GUI_Clear();
      //clear one-shot flag
      Btn_OneShotClear(0x01);
    }
    //if 0x02 is predded, reverse the direction
    if(Btn_IsOneShot(0x02) == 0x02){
      dir ^= 0x01;
      //clear the GUI display
      GUI_Clear();
      Btn_OneShotClear(0x02);
    }
    //if 0x04 is pressed, speed incresed till the maximum speed
    if(Btn_IsOneShot(0x04) == 0x04){
      if(BTN_speed < MaxSpeed)
        BTN_speed ++;
      else
        BTN_speed = MaxSpeed;
      GUI_Clear();
      Btn_OneShotClear(0x04);
    }
    //if 0x08 is presssed, speed decreased till the minimum speed
    if(Btn_IsOneShot(0x08) == 0x08){
```

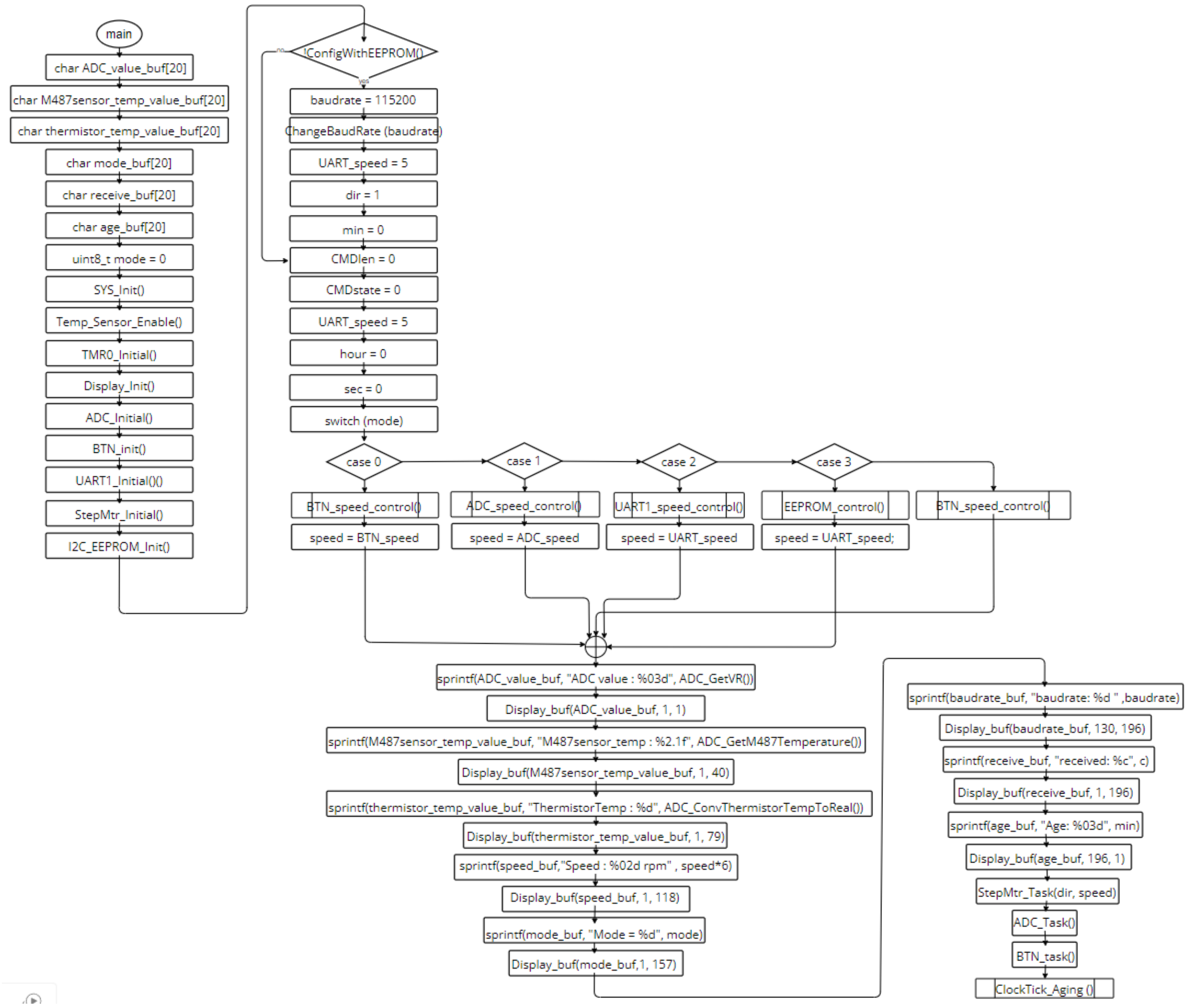


```
    //if 0x08 is presssed, speed decreased till the minimum speed
    if(Btn_IsOneShot(0x08) == 0x08){
      if(BTN_speed > MinSpeed)
        BTN_speed --;
      else
        BTN_speed = MinSpeed;

      GUI_Clear();
      Btn_OneShotClear(0x08);
    }
  }

//function of ADC speed control
//four possible speed depends on the value of v
void ADC_speed_control (void) {
  uint8_t v;
  v = ADC_GetVR() ;
  if(v<=30) {
    ADC_speed = 2;
  }
  else if (v>30 && v<=60) {
    ADC_speed = 5;
  }
  else if (v>60 && v<=90) {
    ADC_speed = 8;
  }
  else {
    ADC_speed = 10;
  }
}
```

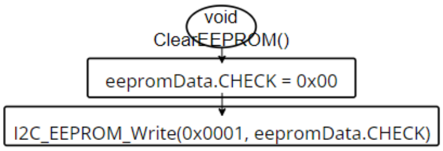


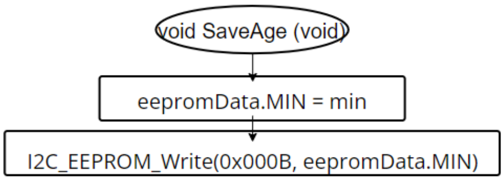
```
//time count function
void ClockTick_Aging (void)
{
    //use the clock biult already in the ssysstem to count second
    //not until timecount == 10000 call it a second
    static uint32_t timecount_old=0;
    if((timecount-timecount_old) < 10000) {
        return;
    }
    timecount_old=timecount;
    sec++; //write second
    if(sec >= 60){ //if there is 60 second, we call it a minute
        sec=0;
        min++; //write minute
        eepromData.MIN=min; //the value of minute is beeing put into the variable inside EEPROM
        I2C_EEPROM_Write(2,eepromData.MIN); //write the varible inside the memory location
}
}
```

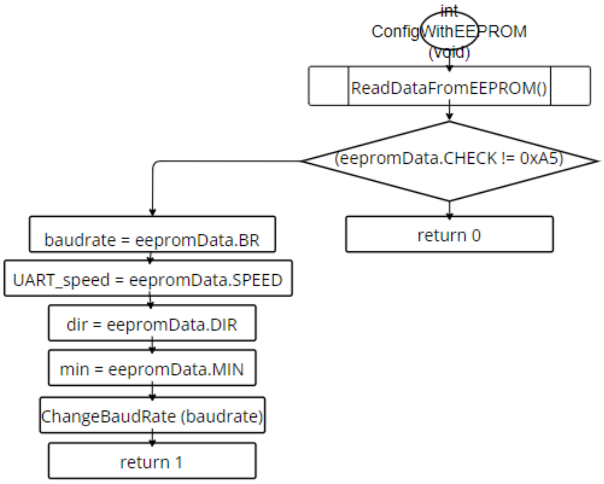
II. Program Flow

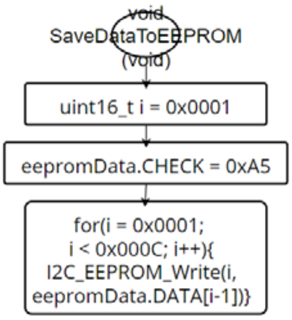


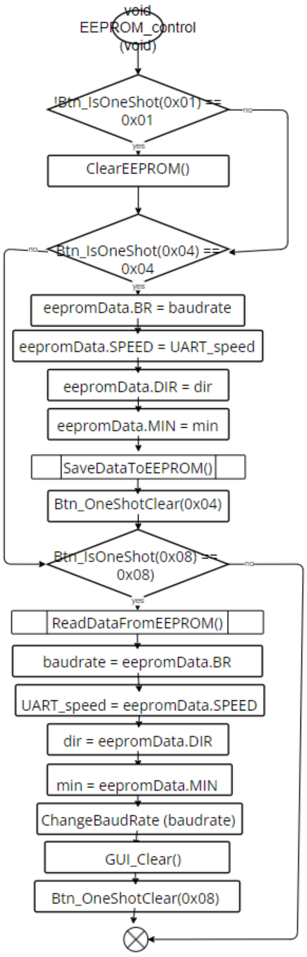
main
char ADC_value_buf[20]
char M487sensor_temp_value_buf[20]
char thermistor_temp_value_buf[20]
char mode_buf[20]
char receive_buf[20]
char age_buf[20]
uint8_t mode = 0
SYS_Init()
Temp_Sensor_Enable()
TMR0_Initial()
Display_Init()
ADC_Initial()
BTN_init()
UART1_Initial()()
StepMtr_Initial()
I2C_EEPROM_Init()
!ConfigWithEEPROM()
no
yes
baudrate = 115200
ChangeBaudRate (baudrate)
UART_speed = 5
dir = 1
min = 0
CMDlen = 0
CMDstate = 0
UART_speed = 5
hour = 0
sec = 0
switch (mode)
case 0
case 1
case 2
case 3
BTN_speed_control()
ADC_speed_control()
UART1_speed_control()
EEPROM_control()
BTN_speed_control()
speed = BTN_speed
speed = ADC_speed
speed = UART_speed
speed = UART_speed;
sprintf(ADC_value_buf, "ADC value : %03d", ADC_GetVR())
Display_buf(ADC_value_buf, 1, 1)
sprintf(M487sensor_temp_value_buf, "M487sensor_temp : %2.1f", ADC_GetM487Temperature())
Display_buf(M487sensor_temp_value_buf, 1, 40)
sprintf(thermistor_temp_value_buf, "ThermistorTemp : %d", ADC_ConvThermistorTempToReal())
Display_buf(thermistor_temp_value_buf, 1, 79)
sprintf(speed_buf,"Speed : %02d rpm" , speed*6)
Display_buf(speed_buf, 1, 118)
sprintf(mode_buf, "Mode = %d", mode)
Display_buf(mode_buf,1, 157)
sprintf(baudrate_buf, "baudrate: %d " ,baudrate)
Display_buf(baudrate_buf, 130, 196)
sprintf(receive_buf, "received: %c", c)
Display_buf(receive_buf, 1, 196)
sprintf(age_buf, "Age: %03d", min)
Display_buf(age_buf, 196, 1)
StepMtr_Task(dir, speed)
ADC_Task()
BTN_task()
ClockTick_Aging ()

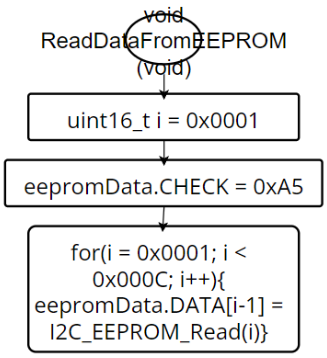

void ClearEEPROM()
eepromData.CHECK = 0x00
I2C_EEPROM_Write(0x0001, eepromData.CHECK)

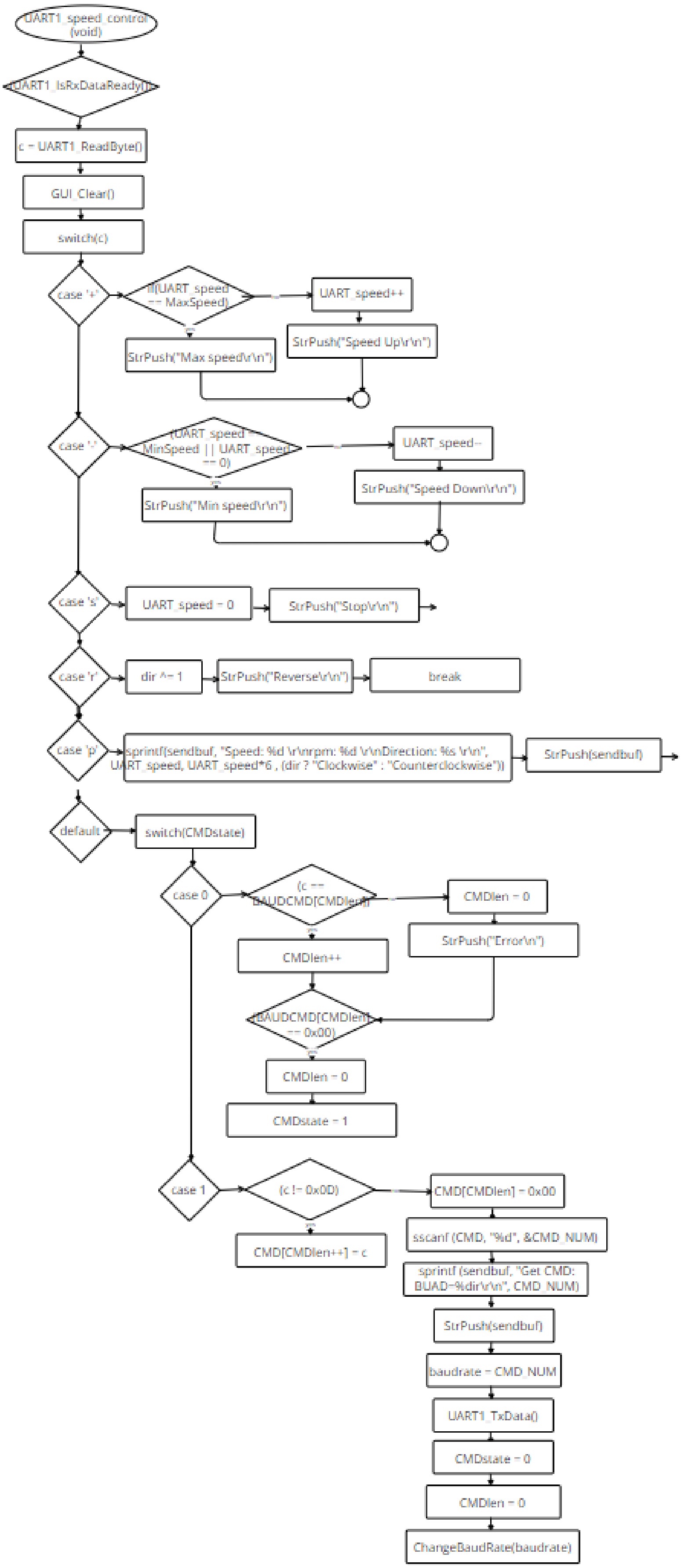

void SaveAge (void)
eepromData.MIN = min
I2C_EEPROM_Write(0x000B, eepromData.MIN)

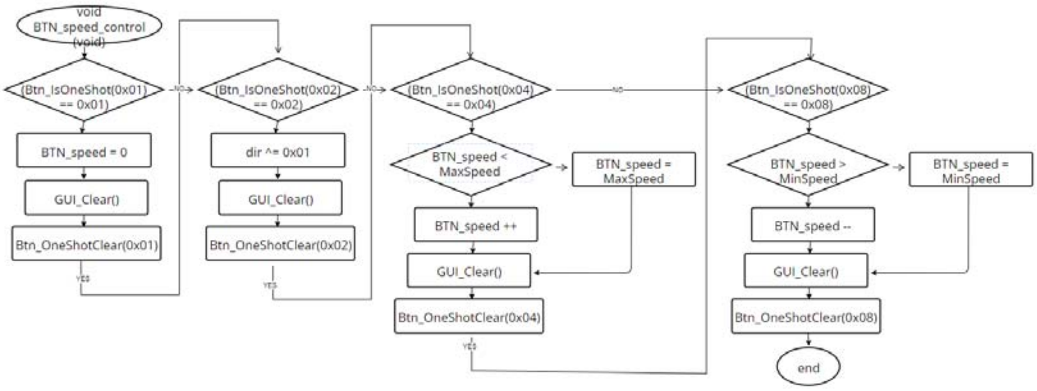

int ConfigWithEEPROM (void)
ReadDataFromEEPROM()
(eepromData.CHECK != 0xA5)
baudrate = eepromData.BR
UART_speed = eepromData.SPEED
dir = eepromData.DIR
min = eepromData.MIN
ChangeBaudRate (baudrate)
return 1
return 0

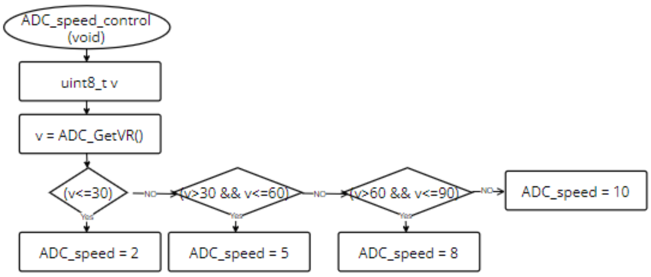

void SaveDataToEEPROM (void)
uint16_t i = 0x0001
eepromData.CHECK = 0xA5
for(i = 0x0001; i < 0x000C; i++){ I2C_EEPROM_Write(i, eepromData.DATA[i-1])}

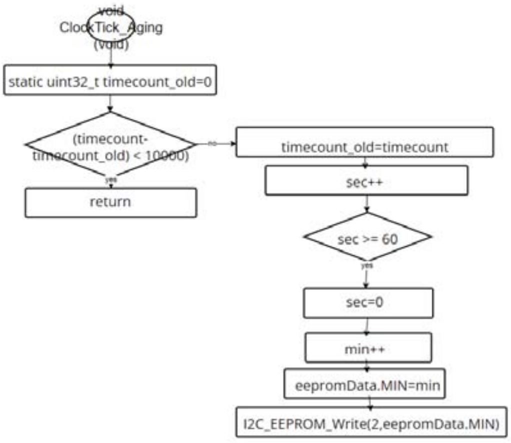

void EEPROM_control (void)
!Btn_IsOneShot(0x01) == 0x01
no
yes
ClearEEPROM()
Btn_IsOneShot(0x04) == 0x04
no
yes
eepromData.BR = baudrate
eepromData.SPEED = UART_speed
eepromData.DIR = dir
eepromData.MIN = min
SaveDataToEEPROM()
Btn_OneShotClear(0x04)
Btn_IsOneShot(0x08) == 0x08)
no
yes
ReadDataFromEEPROM()
baudrate = eepromData.BR
UART_speed = eepromData.SPEED
dir = eepromData.DIR
min = eepromData.MIN
ChangeBaudRate (baudrate)
GUI_Clear()
Btn_OneShotClear(0x08)


void ReadDataFromEEPROM (void)
uint16_t i = 0x0001
eepromData.CHECK = 0xA5
for(i = 0x0001; i < 0x000C; i++){ eepromData.DATA[i-1] = I2C_EEPROM_Read(i)}


UART1_speed_control (void)
(UART1_IsRxDataReady())
c = UART1_ReadByte()
GUI_Clear()
switch(c)
case '+'
if(UART_speed == MaxSpeed)
UART_speed++
StrPush("Speed Up\r\n")
StrPush("Max speed\r\n")
case '-'
(UART_speed = MinSpeed || UART_speed == 0)
UART_speed--
StrPush("Speed Down\r\n")
StrPush("Min speed\r\n")
case 's'
UART_speed = 0
StrPush("Stop\r\n")
case 'r'
dir ^= 1
StrPush("Reverse\r\n")
break
case 'p'
sprintf(sendbuf, "Speed: %d \r\nrpm: %d \r\nDirection: %s \r\n", UART_speed, UART_speed*6 , (dir ? "Clockwise" : "Counterclockwise"))
StrPush(sendbuf)
default
switch(CMDstate)
case 0
(c == BAUDCMD[CMDlen])
CMDlen = 0
StrPush("Error\n")
CMDlen++
(BAUDCMD[CMDlen] == 0x00)
CMDlen = 0
CMDstate = 1
case 1
(c != 0x0D)
CMD[CMDlen] = 0x00
sscanf (CMD, "%d", &CMD_NUM)
sprintf (sendbuf, "Get CMD: BUAD=%d\r\n", CMD_NUM)
StrPush(sendbuf)
baudrate = CMD_NUM
UART1_TxData()
CMDstate = 0
CMDlen = 0
ChangeBaudRate(baudrate)
CMD[CMDlen++] = c






void ClockTick_Aging (void)
static uint32_t timecount_old=0
(timecount-timecount_old) < 10000)
no
yes
return
timecount_old=timecount
sec++
sec >= 60
yes
sec=0
min++
eepromData.MIN=min
I2C_EEPROM_Write(2,eepromData.MIN)

III. Thoughts

This electrical engineering experiment has provided a rich learning experience and valuable insights. We employed C language, stepper motors, UART communication interface, RealTerm software, and EEPROM to construct a versatile stepper motor system capable of mode switching. Building upon our previous experiments, where we became adept at using C language to control stepper motors and display relevant information on the board, this experiment introduced a new element: EEPROM.

Unlike previous iterations, this time we integrated EEPROM into the system to access time data. This crucial addition ensures that even in the event of power loss, the system's configurations and information remain intact within the EEPROM, thereby guaranteeing system stability and reliability.

Throughout this experiment, I gained significant knowledge and skills:

I familiarized myself with the operation principles of the I2C communication interface and learned how to utilize it for data transmission in practical applications.

I mastered the operations of EEPROM, including reading and writing data, and grasped its importance in embedded programming for preserving system configurations.

I learned how to utilize EEPROM effectively in embedded programming to store system configurations, thereby enhancing system reliability and stability.

Through this experiment, our stepper motor system has become more refined, flexible, and reliable. It represents a significant exploration in the field of electrical engineering and lays a solid foundation for our future learning and research endeavors.