Digital Lab 4:

Experiment 5:

Communication Interface

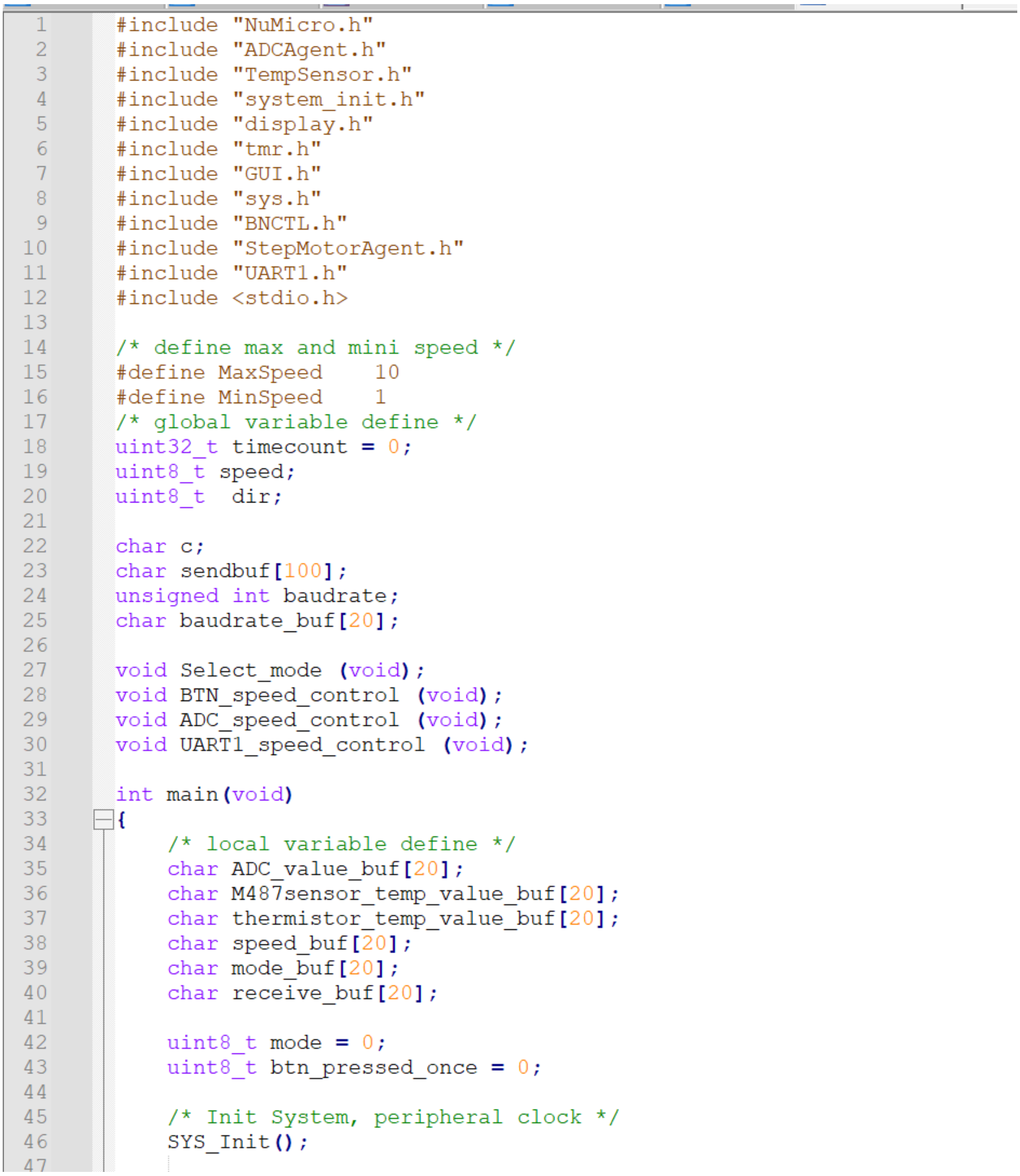
Date: 2024/04/23

Class: 電機三全英班

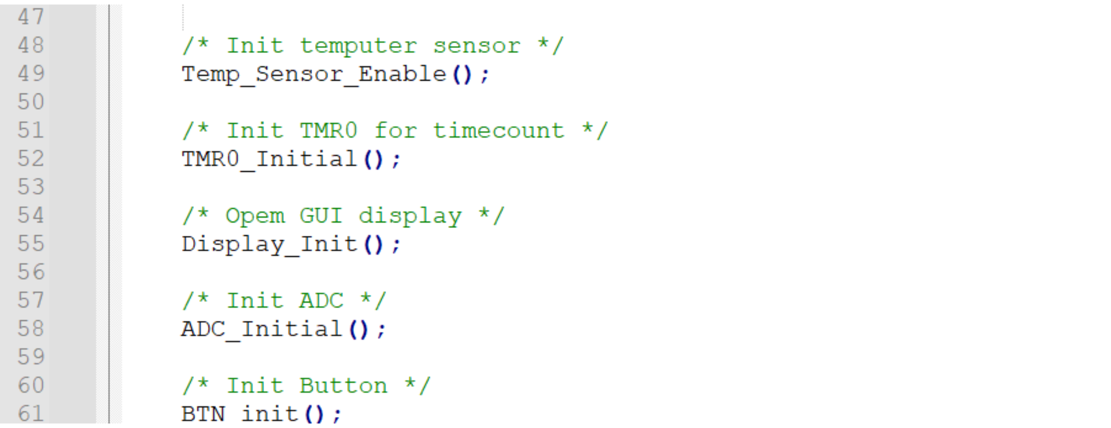
Group: Group 11

Name: B103105006 胡庭翊

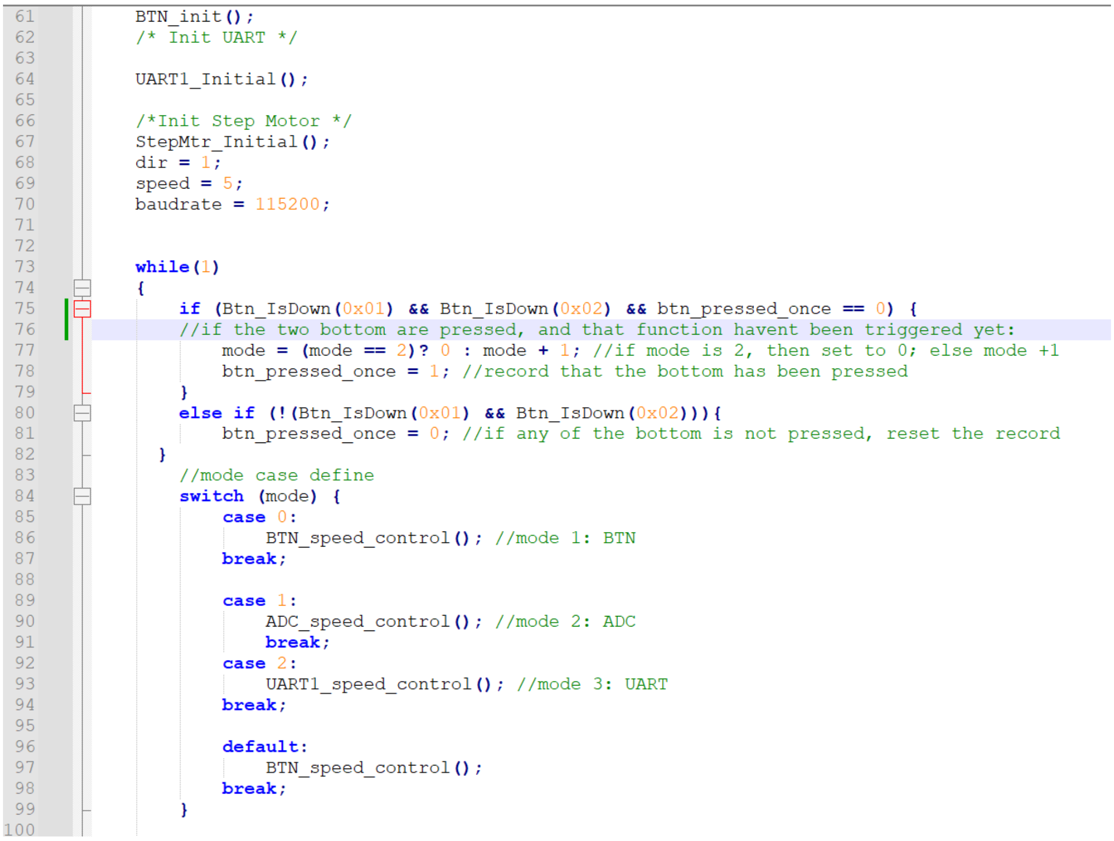
I. Annotated Code



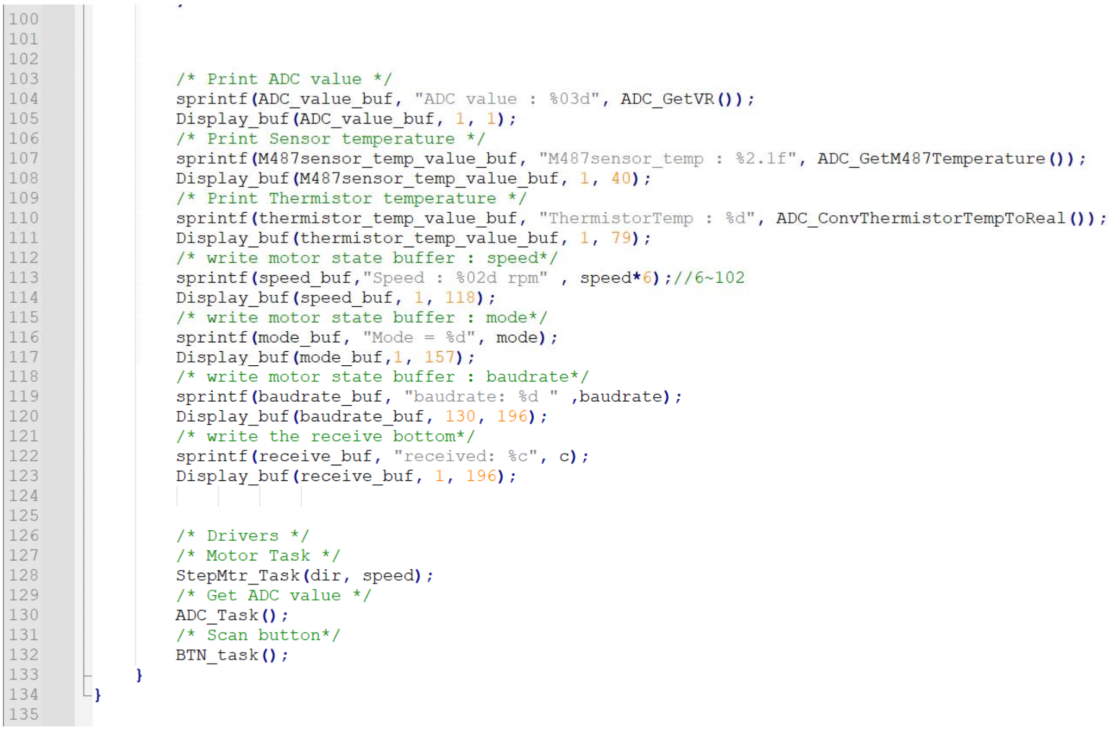
```c
#include "NuMicro.h"
#include "ADCAgent.h"
#include "TempSensor.h"
#include "system_init.h"
#include "display.h"
#include "tmr.h"
#include "GUI.h"
#include "sys.h"
#include "BNCTL.h"
#include "StepMotorAgent.h"
#include "UART1.h"
#include <stdio.h>

/* define max and mini speed */
#define MaxSpeed    10
#define MinSpeed    1
/* global variable define */
uint32_t timecount = 0;
uint8_t speed;
uint8_t  dir;

char c;
char sendbuf[100];
unsigned int baudrate;
char baudrate_buf[20];

void Select_mode (void);
void BTN_speed_control (void);
void ADC_speed_control (void);
void UART1_speed_control (void);

int main(void)
{
    /* 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];

    uint8_t mode = 0;
    uint8_t btn_pressed_once = 0;

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

```



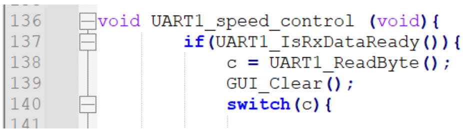
```c

    /* 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();
```



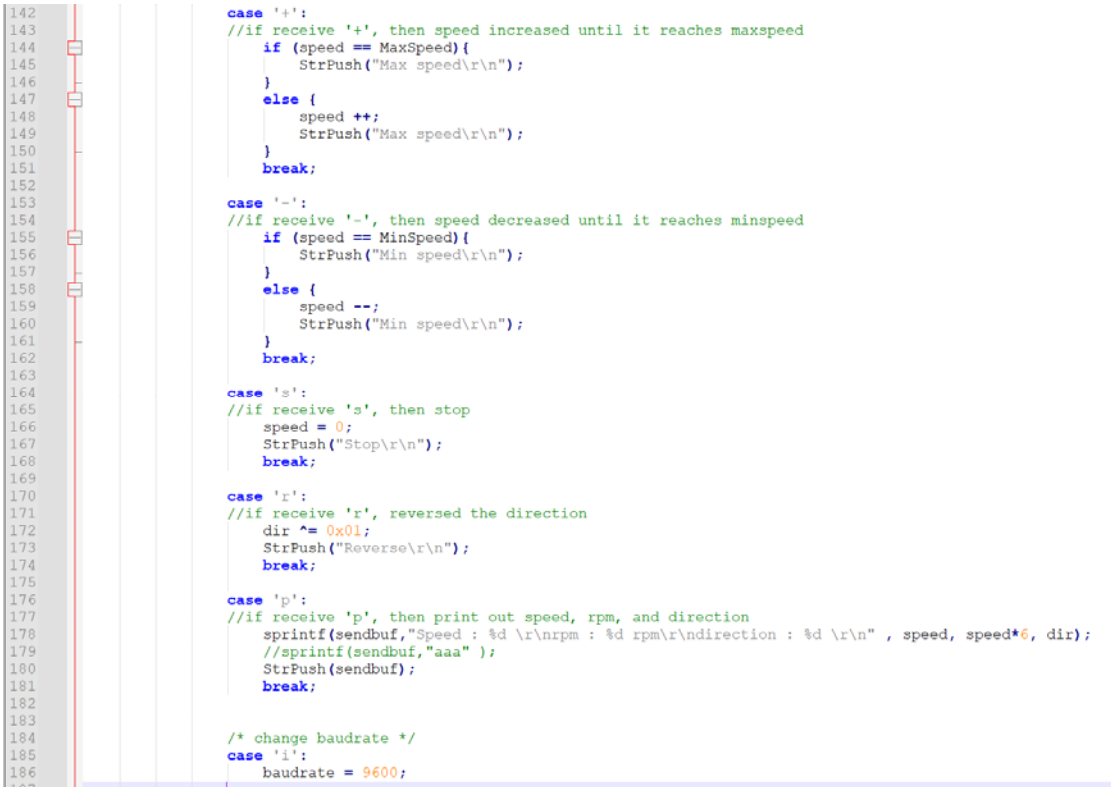
```
    BTN_init();
    /* Init UART */

    UART1_Initial();

    /*Init Step Motor */
    StepMtr_Initial();
    dir = 1;
    speed = 5;
    baudrate = 115200;


    while(1)
    {
        if (Btn_IsDown(0x01) && Btn_IsDown(0x02) && btn_pressed_once == 0) {
        //if the two bottom are pressed, and that function havent been triggered yet:
            mode = (mode == 2)? 0 : mode + 1; //if mode is 2, then set to 0; else mode +1
            btn_pressed_once = 1; //record that the bottom has been pressed
        }
        else if (!(Btn_IsDown(0x01) && Btn_IsDown(0x02))){
            btn_pressed_once = 0; //if any of the bottom is not pressed, reset the record
      }
        //mode case define
        switch (mode) {
            case 0:
                BTN_speed_control(); //mode 1: BTN
            break;

            case 1:
                ADC_speed_control(); //mode 2: ADC
                break;
            case 2:
                UART1_speed_control(); //mode 3: UART
            break;

            default:
                BTN_speed_control();
            break;
        }

```



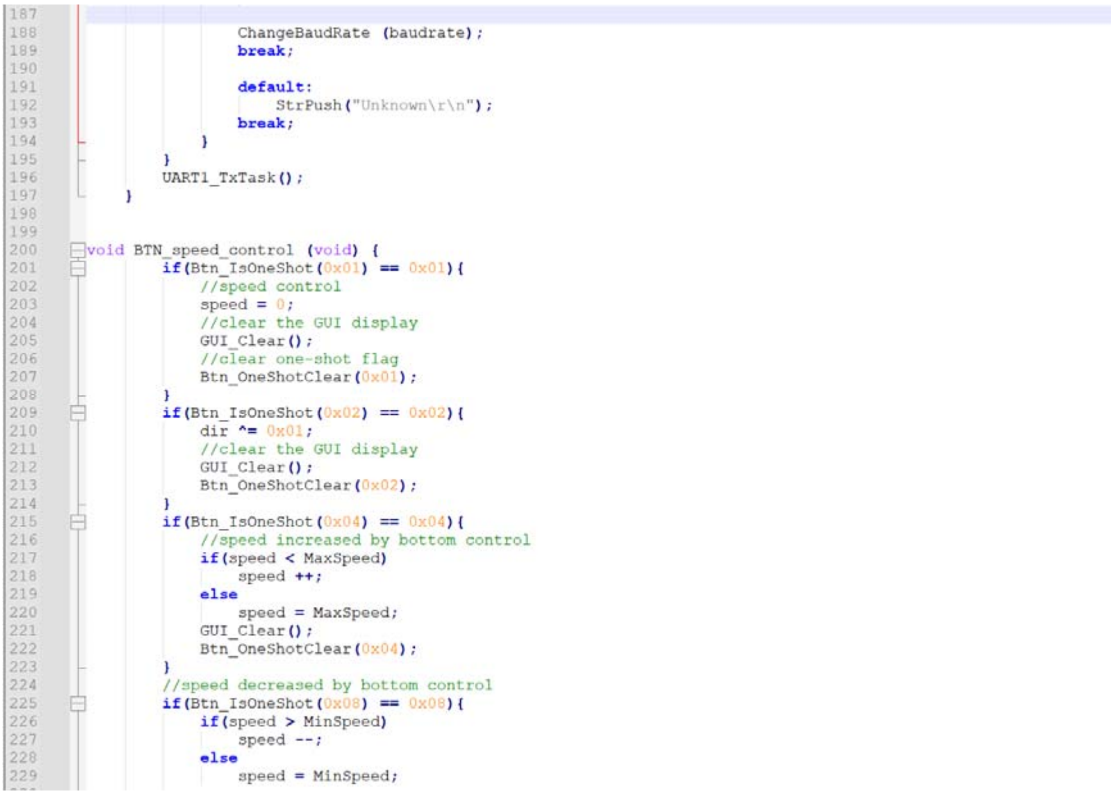
```


        /* 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 : speed*/
        sprintf(speed_buf,"Speed : %02d rpm" , speed*6);//6~102
        Display_buf(speed_buf, 1, 118);
        /* write motor state buffer : mode*/
        sprintf(mode_buf, "Mode = %d", mode);
        Display_buf(mode_buf,1, 157);
        /* write motor state buffer : baudrate*/
        sprintf(baudrate_buf, "baudrate: %d " ,baudrate);
        Display_buf(baudrate_buf, 130, 196);
        /* write the receive bottom*/
        sprintf(receive_buf, "received: %c", c);
        Display_buf(receive_buf, 1, 196);


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

```



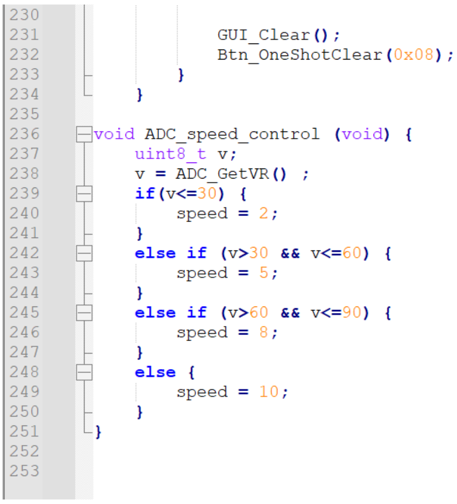
```
void UART1_speed_control (void){
        if(UART1_IsRxDataReady()){
            c = UART1_ReadByte();
            GUI_Clear();
            switch(c){
```



```
                case '+':
                //if receive '+', then speed increased until it reaches maxspeed
                    if (speed == MaxSpeed){
                        StrPush("Max speed\r\n");
                    }
                    else {
                        speed ++;
                        StrPush("Max speed\r\n");
                    }
                    break;

                case '-':
                //if receive '-', then speed decreased until it reaches minspeed
                    if (speed == MinSpeed){
                        StrPush("Min speed\r\n");
                    }
                    else {
                        speed --;
                        StrPush("Min speed\r\n");
                    }
                    break;

                case 's':
                //if receive 's', then stop
                    speed = 0;
                    StrPush("Stop\r\n");
                    break;

                case 'r':
                //if receive 'r', reversed the direction
                    dir ^= 0x01;
                    StrPush("Reverse\r\n");
                    break;

                case 'p':
                //if receive 'p', then print out speed, rpm, and direction
                    sprintf(sendbuf,"Speed : %d \r\nrpm : %d rpm\r\ndirection : %d \r\n" , speed, speed*6, dir);
                    //sprintf(sendbuf,"aaa" );
                    StrPush(sendbuf);
                    break;


                /* change baudrate */
                case 'i':
                    baudrate = 9600;
```

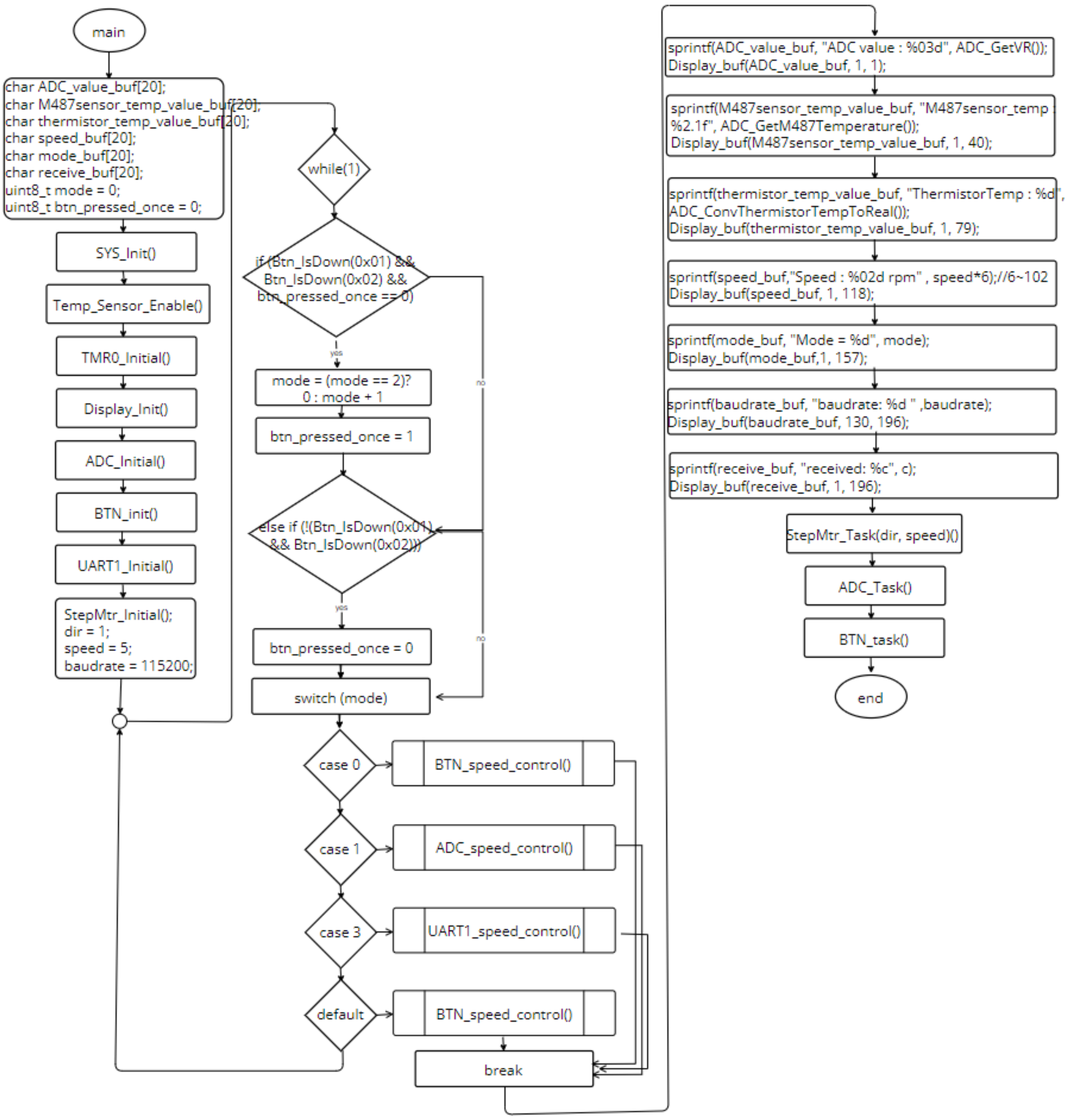


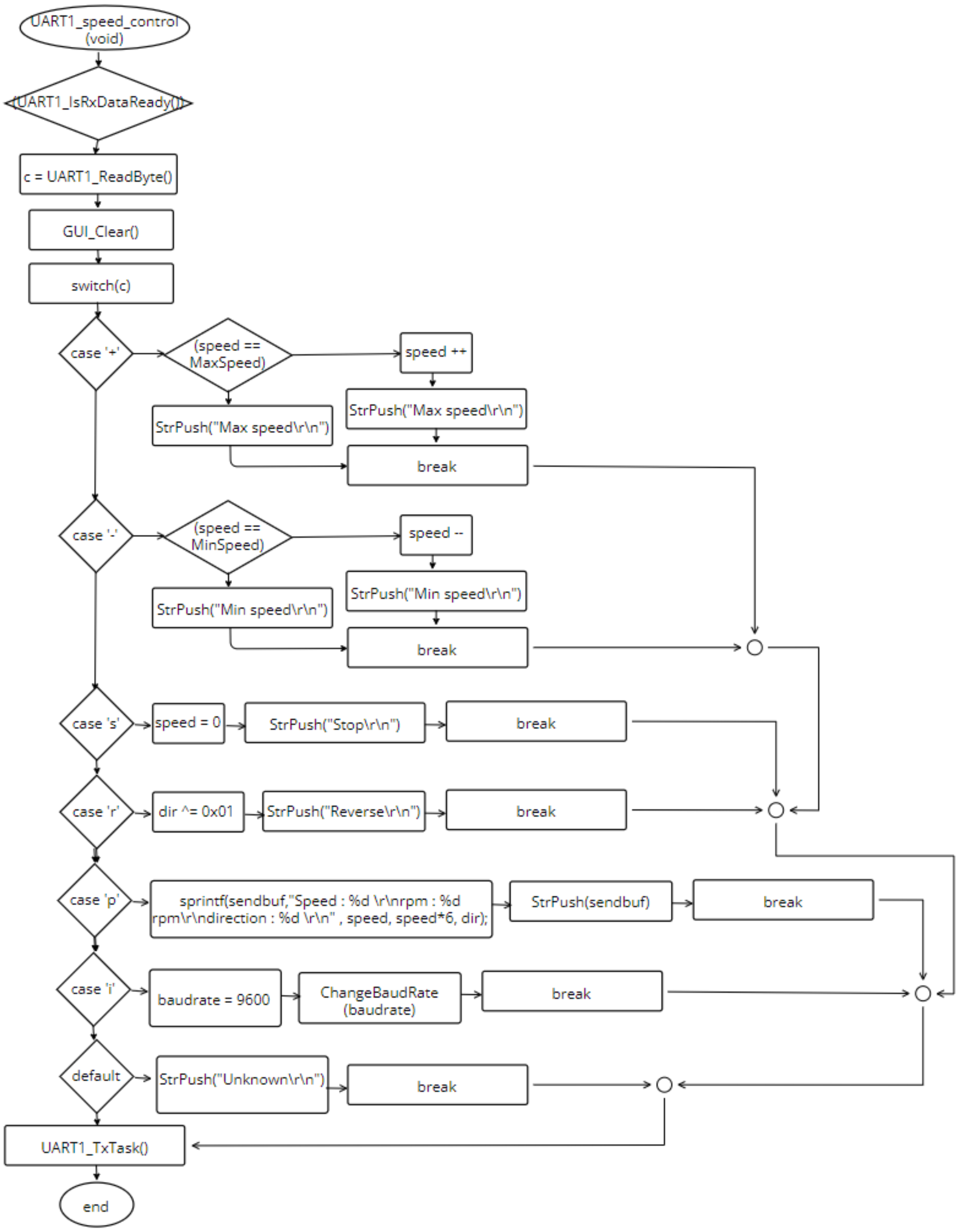
```
                ChangeBaudRate (baudrate);
                break;

                default:
                    StrPush("Unknown\r\n");
                break;
            }
        }
        UART1_TxTask();
    }


void BTN_speed_control (void) {
        if(Btn_IsOneShot(0x01) == 0x01){
            //speed control
            speed = 0;
            //clear the GUI display
            GUI_Clear();
            //clear one-shot flag
            Btn_OneShotClear(0x01);
        }
        if(Btn_IsOneShot(0x02) == 0x02){
            dir ^= 0x01;
            //clear the GUI display
            GUI_Clear();
            Btn_OneShotClear(0x02);
        }
        if(Btn_IsOneShot(0x04) == 0x04){
            //speed increased by bottom control
            if(speed < MaxSpeed)
                speed ++;
            else
                speed = MaxSpeed;
            GUI_Clear();
            Btn_OneShotClear(0x04);
        }
        //speed decreased by bottom control
        if(Btn_IsOneShot(0x08) == 0x08){
            if(speed > MinSpeed)
                speed --;
            else
                speed = MinSpeed;
```

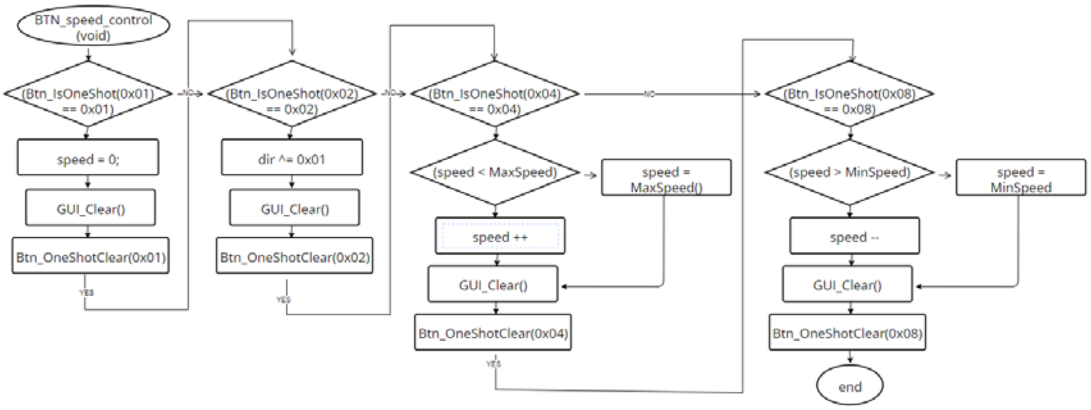


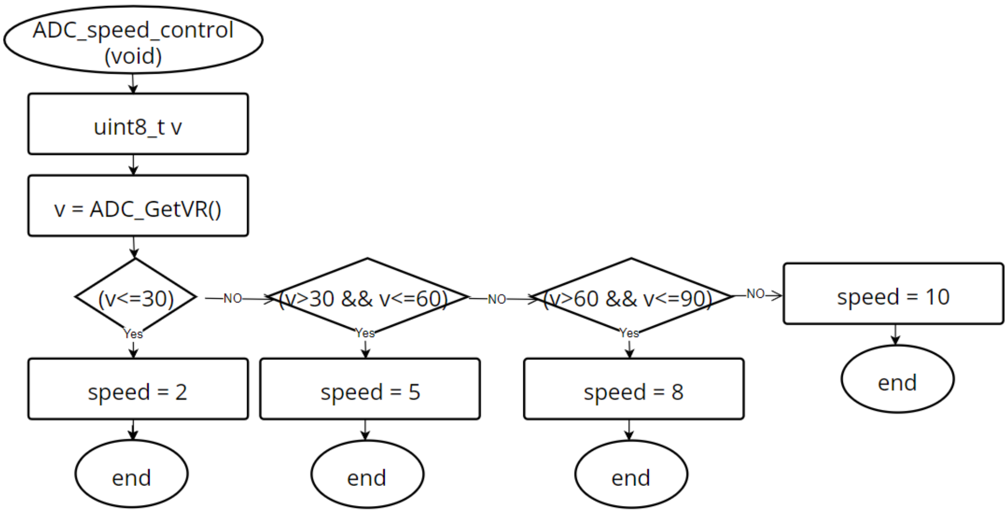
```
            GUI_Clear();
            Btn_OneShotClear(0x08);
        }
    }

void ADC_speed_control (void) {
    uint8_t v;
    v = ADC_GetVR() ;
    if(v<=30) {
        speed = 2;
    }
    else if (v>30 && v<=60) {
        speed = 5;
    }
    else if (v>60 && v<=90) {
        speed = 8;
    }
    else {
        speed = 10;
    }
}
```

II. Program Flow


main
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];
uint8_t mode = 0;
uint8_t btn_pressed_once = 0;
SYS_Init()
Temp_Sensor_Enable()
TMR0_Initial()
Display_Init()
ADC_Initial()
BTN_init()
UART1_Initial()
StepMtr_Initial();
dir = 1;
speed = 5;
baudrate = 115200;
while(1)
if (Btn_IsDown(0x01) && Btn_IsDown(0x02) && btn_pressed_once == 0)
yes
no
mode = (mode == 2)? 0 : mode + 1
btn_pressed_once = 1
else if (!(Btn_IsDown(0x01) && Btn_IsDown(0x02)))
yes
no
btn_pressed_once = 0
switch (mode)
case 0
BTN_speed_control()
case 1
ADC_speed_control()
case 3
UART1_speed_control()
default
BTN_speed_control()
break
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);//6~102
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);
StepMtr_Task(dir, speed)()
ADC_Task()
BTN_task()
end


UART1_speed_control (void)
(UART1_IsRxDataReady())
c = UART1_ReadByte()
GUI_Clear()
switch(c)
case '+'
(speed == MaxSpeed)
speed ++
StrPush("Max speed\r\n")
StrPush("Max speed\r\n")
break
case '-'
(speed == MinSpeed)
speed --
StrPush("Min speed\r\n")
StrPush("Min speed\r\n")
break
case 's'
speed = 0
StrPush("Stop\r\n")
break
case 'r'
dir ^= 0x01
StrPush("Reverse\r\n")
break
case 'p'
sprintf(sendbuf,"Speed : %d \r\nrpm : %d rpm\r\ndirection : %d \r\n" , speed, speed*6, dir);
StrPush(sendbuf)
break
case 'i'
baudrate = 9600
ChangeBaudRate (baudrate)
break
default
StrPush("Unknown\r\n")
break
UART1_TxTask()
end


BTN_speed_control (void)
(Btn_IsOneShot(0x01) == 0x01)
NO
speed = 0;
GUI_Clear()
Btn_OneShotClear(0x01)
YES
(Btn_IsOneShot(0x02) == 0x02)
NO
dir ^= 0x01
GUI_Clear()
Btn_OneShotClear(0x02)
YES
(Btn_IsOneShot(0x04) == 0x04)
NO
(speed < MaxSpeed)
speed = MaxSpeed()
speed ++
GUI_Clear()
Btn_OneShotClear(0x04)
YES
(Btn_IsOneShot(0x08) == 0x08)
(speed > MinSpeed)
speed = MinSpeed
speed --
GUI_Clear()
Btn_OneShotClear(0x08)
end



III. Thoughts

In this experiment, we delved into the realm of Communication Interface by employing C language, Stepper Motor, UART communication interface, and RealTerm Software. Having previously gained experience in controlling stepper motors using C and displaying information such as speed and direction on the board, this experiment built upon our prior knowledge. Utilizing the same circuit board we assembled in the previous experiments, we aimed to establish remote transmission control via UART and enable remote control of the stepper motor by inputting commands from a computer.

This experiment provided invaluable insights into designing communication programs in embedded programming. We learned how to establish communication channels between embedded systems and external devices, facilitating remote control and data exchange. By leveraging UART communication interface and RealTerm Software, we successfully implemented remote transmission control, enabling us to manipulate the stepper motor's actions through commands input from the computer.

Through this experiment, I realized the importance of systematic design in embedded programming, especially in communication interfaces. Each iteration of experiments contributed to the refinement and completeness of our stepper motor program. It's fascinating to witness the evolution of our stepper motor program, from its initial stages to its current state of robustness and versatility.

Overall, this experiment not only expanded our understanding of communication interfaces in embedded systems but also underscored the significance of iterative learning in engineering. It's gratifying to see how our efforts and learning experiences have contributed to the enhancement of our skills and the refinement of our projects.