
# EE2028 Microcontroller Programming and Interfacing

# Assignment 2

Friday_PM
Group 10

Huang Zhiwen (A0194530H)
Du Yutong (A0194545W)

# Content

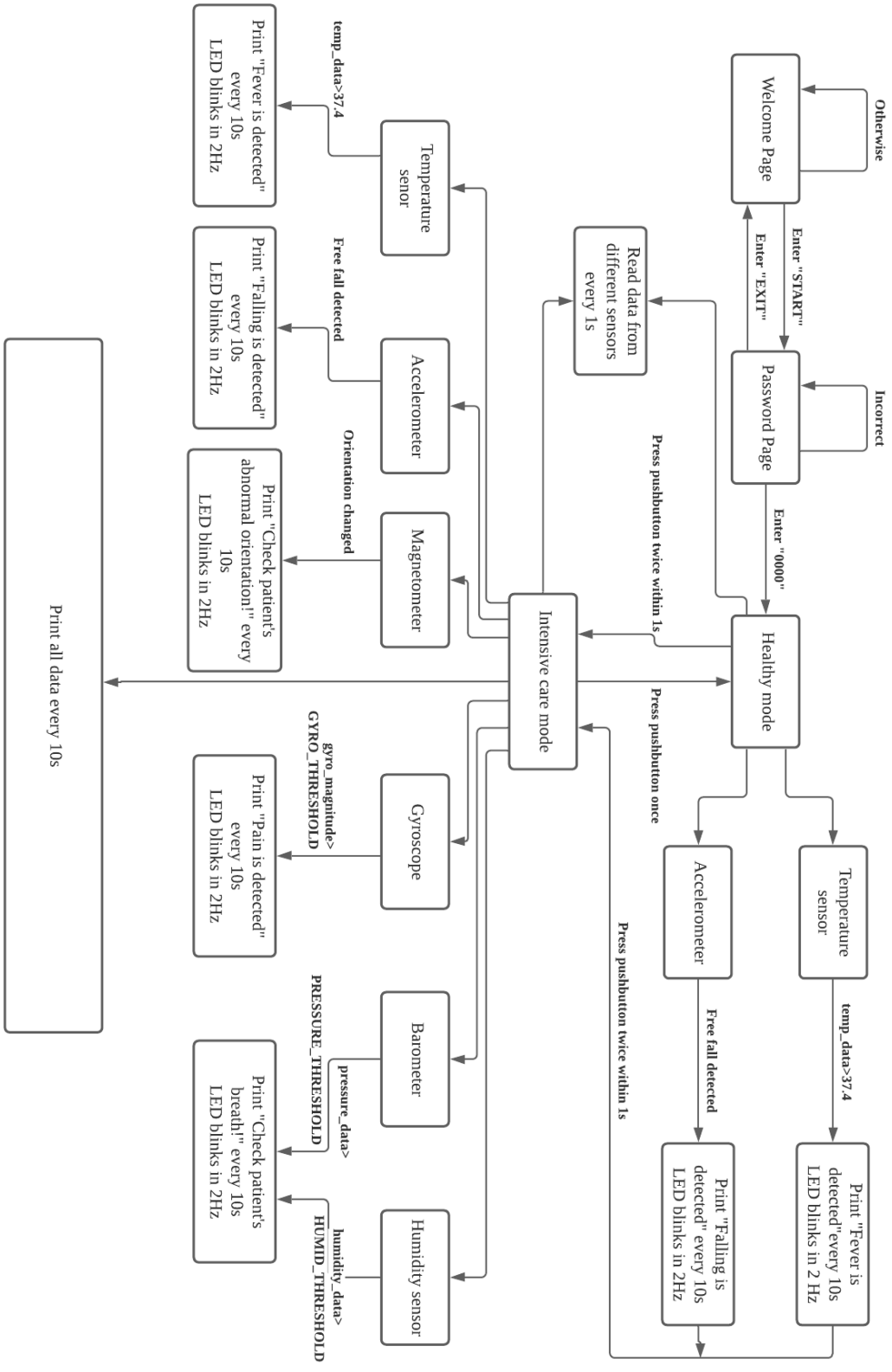
# 1. Introduction

In this project, we designed and implemented a novel monitoring system for medical purpose. Its main target group is COVID patients, especially early COVID patients. We shall call this system as COvid Patient Enhanced MONitoring, or COPEMON. COPEMON is established based on STM32 with those embedded sensors, and it is designed to have two different monitoring mode, including Healthy mode and Intensive Care mode. COPEMON will detect and report the patients' different physical situations according to the mode of the system. Healthy mode is mainly used for early COVID patients, COPEMON can monitor the temperature, and it also has falling warning function. If the patient's temperature is above normal or free fall of the patient is detected, LED on COPEMON will blink, and the warning message will be sent to the computer. Doctors will be warned to check the situation of the patient. If doctors decide to switch the system to Intensive Care mode to have a all-rounded monitor to a certain patient. Besides temperature and free fall detection, the COPEMON will also check the patient's body orientation, lung pressure and humidity, and body movement. COPEMON will send all these data to the computer every 10 seconds. Similarly, if any of the data is beyond the safety threshold, COPEMON will also send warning messages.

# 2. Objectives

- Switch between two different modes.
- Detect and gather all kinds of data through different sensors.
- Print all kinds of data on computer through UART in a certain frequency.
- Send warning messages if the situation of the patient is abnormal.

# 3. Flow chart


Welcome Page
Otherwise
Enter "START"
Enter "EXIT"
Password Page
Incorrect
Enter "0000"
Healthy mode
Read data from different sensors every 1s
Press pushbutton twice within 1s
Intensive care mode
Press pushbutton once
Temperature sensor
temp_data>37.4
Print "Fever is detected" every 10s
LED blinks in 2Hz
Accelerometer
Free fall detected
Print "Falling is detected" every 10s
LED blinks in 2Hz
Magnetometer
Orientation changed
Print "Check patient's abnormal orientation!" every 10s
LED blinks in 2Hz
Gyroscope
gyro_magnitude> GYRO_THRESHOLD
Print "Pain is detected" every 10s
LED blinks in 2Hz
Barometer
pressure_data> PRESSURE_THRESHOLD
Print "Check patient's breath!" every 10s
LED blinks in 2Hz
Humidity sensor
humidity_data> HUMID_THRESHOLD
Print all data every 10s
Temperature sensor
temp_data>37.4
Print "Fever is detected"every 10s
LED blinks in 2 Hz
Accelerometer
Free fall detected
Print "Falling is detected" every 10s
LED blinks in 2Hz
Press pushbutton twice within 1s

# 4. Detailed implementation

- Line 9-Line 29:
  Include all library file used in the following code. Define all kinds of thresholds used below to determine the situation of the patient's body condition. Declare functions to configure UART and GPIO.

```
7
8   /* Includes ------------------------------------------------------------------*/
9   #include "main.h"
10  #include <math.h>
11  #include "../../Drivers/BSP/B-L475E-IOT01/stm32l475e_iot01_accelero.h"
12  #include "../../Drivers/BSP/B-L475E-IOT01/stm32l475e_iot01_tsensor.h"
13  #include "../../Drivers/BSP/B-L475E-IOT01/stm32l475e_iot01_gyro.h"
14  #include "../../Drivers/BSP/B-L475E-IOT01/stm32l475e_iot01_magneto.h"
15  #include "../../Drivers/BSP/B-L475E-IOT01/stm32l475e_iot01_psensor.h"
16  #include "../../Drivers/BSP/B-L475E-IOT01/stm32l475e_iot01_hsensor.h"
17  #include "stdio.h"
18  #include "stm32l4xx_it.h"
19  #include "string.h"
20  #define TEMP_THRESHOLD 37.4
21  #define GYRO_THRESHOLD 100
22  #define HUMID_THRESHOLD 70
23  #define PRESSURE_THRESHOLD 1050
24  //extern void initialise_monitor_handles(void); // for semi-hosting support (printf)
25  static void MX_GPIO_Init(void);
26  static void UART1_Init(void);
27  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
28  UART_HandleTypeDef huart1;
29  int buttonpress=0,fall_warning=0;
```

- Line 30- Line 57:
  Initial all kinds of variables used below. Initial peripherals.
  mode=0: Healthy mode
  mode=1: Intensive Care mode
  mode=3: Password page
  mode=4: Welcome page  (There is no mode=2)
  lying_time: used as clock for magnet_warning
  sample_time: used as clock for reading data
  press_time: used to determine the interval of two pushes
  blink_time: used as clock for LED blinking
  current_time: used as clock for other warnings
  press: how many times you press the button
  temp_warning, gyro_warning, magnet_warning, humid_warning, pressure_warning: flags for warnings
  blink: flag for LED blinking

```
30  int main(void)
31  {
32      int mode=4;//0 for healthy mode and 1 for intensive care mode
33      int seconds_count = 0,enter1=0,enter2=0;
34      int current_time=0,lying_time=0,sample_time=0,press_time,blink_time=0,press=0;//press: how many times you press the button
35      int temp_warning=0,gyro_warning=0,magnet_warning=0,humid_warning=0,pressure_warning=0,firstdetect=0;
36
37      char line_print[64];
38      char buffer[64];
39
40      HAL_Init();
41      UART1_Init();
42
43      /* Peripheral initializations using BSP functions */
44      BSP_ACCELERO_Init();
45      BSP_TSENSOR_Init();
46      BSP_GYRO_Init();
47      BSP_MAGNETO_Init();
48      BSP_HSENSOR_Init();
49      BSP_PSENSOR_Init();
50
51      MX_GPIO_Init();
52      HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
53      SENSOR_IO_Write(0xD4,0x58,0x80);//enable free fall interrupt
54      SENSOR_IO_Write(0xD4,0x5E,0x90);//configure in
55      int blink=0;
56      char password[]="0000",wrong[]="Password incorrect.(Enter EXIT to quit)\n\r",exitmode1[]="EXIT",exitmode2[]="exit",exitmode3[]="Exit";
57      char exitmode[]="Exit Successfully\n\r";
```

- Line 58- Line 108:

Initial all variables and arrays used to record data from sensors. Record and process raw data from sensors. Record the patient's initial body orientation (maget_data).

```
{
    int len=0;
    char data;
    char fall[]="Falling is detected\n\r";
    char breath[]="Check patient's breath!\n\r";
    char healthymode[]="Entering Healthy Mode\r\n";
    char intensivemode[]="Entering Intensive Care Mode\r\n";
    float accel_data[3];
    float gyro_data[3];
    float maget_data[3];
    float gyro_magnitude;
    float oringinal_orientation[3];
    char fever[]="Fever is detected\n\r";
    char pain[]="Pain is detected\n\r";
    char orientation[]="Check patient's abnormal orientation!\r\n";
    float temp_data,humidity_data,pressure_data;
    int16_t accel_data_i16[3] = { 0 };          // array to store the x, y and z readings.
    int16_t maget_data_i16[3]={0};
    if(HAL_GetTick()-sample_time>1000)
    {
    BSP_ACCELERO_AccGetXYZ(accel_data_i16);     // read accelerometer
    BSP_GYRO_GetXYZ(gyro_data);
    BSP_MAGNETO_GetXYZ(maget_data_i16);

    //gyro data
    gyro_data[0]=(float)gyro_data[0]/1000.0f;
    gyro_data[1]=(float)gyro_data[1]/1000.0f;
    gyro_data[2]=(float)gyro_data[2]/1000.0f;
    gyro_magnitude=(float)sqrt(gyro_data[0]*gyro_data[0]+gyro_data[1]*gyro_data[1]+gyro_data[2]*gyro_data[2]);
    //accel data
    accel_data[0] = (float)accel_data_i16[0] / 100.0f;
    accel_data[1] = (float)accel_data_i16[1] / 100.0f;
    accel_data[2] = (float)accel_data_i16[2] / 100.0f;
    //maget data
    maget_data[0] = (float)maget_data_i16[0]/1000.0f;
    maget_data[1] = (float)maget_data_i16[1]/1000.0f;
    maget_data[2] = (float)maget_data_i16[2]/1000.0f;
```

```
    if(firstdetect==0)
    {
        oringinal_orientation[0]=maget_data[0];
        oringinal_orientation[1]=maget_data[1];
        oringinal_orientation[2]=maget_data[2];
        firstdetect=1;
    }
    temp_data = BSP_TSENSOR_ReadTemp();         // read temperature sensor
    humidity_data=BSP_HSENSOR_ReadHumidity();   // humid
    pressure_data=BSP_PSENSOR_ReadPressure();   // pressure

    sample_time=HAL_GetTick();
    }
```

- Line 109- Line 142: First part of Healthy mode.
  Mode toggle: press_time will record the last time of pressing the pushbutton. If the current time of pressing (HAL_GetTick) minus press_time is smaller than 1000 (1s), mode will be changed to 1 (Intensive Care mode), and all flags will be initialed. Otherwise, press_time will be updated to the current time of pressing the pushbutton.
  enter1: Making sure the "Entering Healthy Mode" will only be printed once.

```
109         if(mode==0)
110             {
111             if(enter1==0)
112             {
113             HAL_UART_Transmit(&huart1, (uint8_t*)healthymode, strlen(healthymode),0xFFFF);
114             enter1=1;
115             }
116             //mode toggle
117             if(buttonpress==1&&press==0)
118                 {
119                     press_time=HAL_GetTick();
120                     press=1;
121                     buttonpress=0;
122                 }
123                 else if(buttonpress==1&&press==1)
124                 {
125                     if(HAL_GetTick()-press_time<1000)
126                     {
127                         mode=1;
128                         press=0;
129                         buttonpress=0;
130                         enter2=0;
131                         blink=0;
132                         temp_warning=0;
133                         fall_warning=0;
134                         gyro_warning=0;
135                         magnet_warning=0;
136                         humid_warning=0;
137                         pressure_warning=0;
138                     }
139                     else {
140                         press_time=HAL_GetTick();
141                         buttonpress=0;}
142                 }
```

- Line 143- Line 163: Second part of Healthy mode
  If the body temperature is abnormal or free fall is detected, the LED flag will be updated (LED blinks in 2Hz), and warning messages will be printed every 10 seconds by checking the difference between the current time (HAL_GetTick) and current_time (the time of last print).

```
143             //temperature&&free_fall
144                 if(HAL_GetTick()-current_time>=10000)
145                 {
146                     if(temp_data>TEMP_THRESHOLD)
147                     {
148                        temp_warning=1;
149 
150                     }
151                     if(temp_warning==1)
152                     {
153                         HAL_UART_Transmit(&huart1, (uint8_t*)fever, strlen(fever),0xFFFF);
154                         blink=1;
155                     }
156 
157                     if(fall_warning==1)
158                     {
159                         HAL_UART_Transmit(&huart1, (uint8_t*)fall, strlen(fall),0xFFFF);
160                         blink=1;
161                     }
162                     current_time=HAL_GetTick();
163                 }
```

- Line 167- Line 188: First part of Intensive Care mode
  enter2: Making sure “Entering Intensive Care Mode” will only be printed once.
  Mode toggle: If the pushbutton is pressed, all flags will be initialed to 0, and LED will be reseted.

```
167         if(mode==1)
168         {
169             if(enter2==0)
170             {
171             HAL_UART_Transmit(&huart1, (uint8_t*)intensivemode, strlen(intensivemode),0xFFFF);
172 
173             enter2=1;
174             }
175             //mode_toggle
176             if(buttonpress==1)
177                {
178                     mode=0;
179                     buttonpress=0;
180                     enter1=0;
181                     blink=0;
182                     temp_warning=0;
183                     fall_warning=0;
184                     gyro_warning=0;
185                     magnet_warning=0;
186                     humid_warning=0;
187                     pressure_warning=0;
188                     HAL_GPIO_WritePin(GPIOB, LED2_Pin, GPIO_PIN_RESET);
```

- Line 190- Line 232: Second part of Intensive Care mode
  If any of the sensor data is beyond the safety threshold, the system will update the warning flags. Subsequently, the system will print warning message on computer every 10 seconds by checking the difference between the current time (HAL_GetTick) and current_time (the time of last print).

```
190             //warning message
191             if(HAL_GetTick()-current_time>=10000)
192             {
193               /*temp sensor*/
194                if(temp_data>TEMP_THRESHOLD)
195                    temp_warning=1;
196                if(temp_warning==1)
197                {
198                    HAL_UART_Transmit(&huart1, (uint8_t*)fever, strlen(fever),0xFFFF);
199                    blink=1;
200                }
201                /*accel sensor*/
202
203                if(fall_warning==1)
204                {
205                    HAL_UART_Transmit(&huart1, (uint8_t*)fall, strlen(fall),0xFFFF);
206                    blink=1;
207                }
208                /*gyro sensor*/
209               if(gyro_magnitude>GYRO_THRESHOLD)
210                {
211                  gyro_warning=1;
212                }
213             if(gyro_warning==1)
214               {
215                HAL_UART_Transmit(&huart1, (uint8_t*)pain, strlen(pain),0xFFFF);
216                blink=1;
217               }
218            /*humidity and pressure sensor*/
219            if(humidity_data<HUMID_THRESHOLD)
220            {
221                humid_warning=1;
222            }
223            if(pressure_data>PRESSURE_THRESHOLD)
224            {
225                pressure_warning=1;
226            }
```

```
227             if(humid_warning==1||pressure_warning==1)
228             {
229                HAL_UART_Transmit(&huart1,(uint8_t*)breath,strlen(breath),0xFFFF);
230                blink=1;
231             }
232
```

- Line 233- Line 246: Third part of Intensive Care mode
  The system will always print all kinds of data on computer every 10 seconds by checking the difference between the current time (HAL_GetTick) and current_time (the time of last warning-printing, also the time of last data-printing).
  memset: Reset the line_print to reuse this string.

```
233             sprintf(line_print,"%.3d_TEMP_%.2f_ACC_%.2f_%.2f_%.2f \r\n",seconds_count,temp_data,accel_data[0]/9.8, accel_data[1]/9.8, accel_data[2]/9.8);
234             HAL_UART_Transmit(&huart1,(uint8_t*)line_print , strlen(line_print),0xFFFF);
235             seconds_count++;
236             memset(line_print,0,strlen(line_print));
237             sprintf(line_print,"%.3d GYRO %.1f MAGNETO %.1f %.1f %.1f \r\n",seconds_count,gyro_magnitude,maget_data[0], maget_data[1], maget_data[2]);
238             HAL_UART_Transmit(&huart1,(uint8_t*)line_print,strlen(line_print),0xFFFF);
239             seconds_count++;
240             memset(line_print,0,strlen(line_print));
241             sprintf(line_print,"%.3d HUMIDITY %.1f and BARO %.2f \n\r",seconds_count,humidity_data,pressure_data);
242             HAL_UART_Transmit(&huart1,(uint8_t*)line_print,strlen(line_print),0xFFFF);
243             seconds_count++;
244             memset(line_print,0,strlen(line_print));
245
246               current_time=HAL_GetTick();
```

- Line 248- Line 269: Fourth part of Intensive Care mode
  If the system finds the patient's body orientation is changed (magnet_warning is updated), it will then print "Check patient's abnormal orientation!" on the computer immediately by checking the difference between the current time (HAL_GetTick) and lying_time (the time of last print of magnet_warning).

```
248                 if(HAL_GetTick()-lying_time>10000)
249                 {
250                     if((maget_data[0]-oringinal_orientation[0])>0.3||(oringinal_orientation[0]-maget_data[0])>0.3)
251                     {
252                         magnet_warning=1;
253                     }
254                     if((maget_data[1]-oringinal_orientation[1])>0.3||(oringinal_orientation[1]-maget_data[1])>0.3)
255                     {
256                         magnet_warning=1;
257                     }
258                     if((maget_data[2]-oringinal_orientation[2])>0.3||(oringinal_orientation[2]-maget_data[2])>0.3)
259                     {
260                         magnet_warning=1;
261                     }
262                     if(magnet_warning==1)
263                     {
264                         HAL_UART_Transmit(&huart1, (uint8_t*)orientation, strlen(orientation),0xFFFF);
265                         blink=1;
266                     }
267                     lying_time=HAL_GetTick();
268                 }
269         }
```

- Line 270- Line 328: Enhancement: Password Page and Welcome Page
- Line 329- Line 338:
  If the flag for LED blinking (blink) is 1, the LED will blink in 2Hz by checking the difference between the current time (HAL_GetTick) and blink_time (the time of last TogglePin, or the time of last blinking).

```
329             if(blink==1)
330             {
331                 if(HAL_GetTick()-blink_time>500)
332                 {
333                     HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_14);
334                     blink_time=HAL_GetTick();
335                 }
336             }
337
338 }
```

- Line 342- Line 407:
  UART, peripherals GPIO configuration.

```
342 static void MX_GPIO_Init(void)
343     {
344       GPIO_InitTypeDef GPIO_InitStruct = {0};
345       /* GPIO Ports Clock Enable */
346       __HAL_RCC_GPIOB_CLK_ENABLE();
347       __HAL_RCC_GPIOC_CLK_ENABLE();
348       __HAL_RCC_GPIOD_CLK_ENABLE();
349       /*Configure GPIO pin Output Level */
350       //pushbutton
351       HAL_GPIO_WritePin(GPIOC,BUTTON_EXTI13_Pin, GPIO_PIN_RESET);
352       GPIO_InitStruct.Pin = BUTTON_EXTI13_Pin;
353       GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
354       GPIO_InitStruct.Pull = GPIO_PULLUP;
355       HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
356       //freefall
357       HAL_GPIO_WritePin(GPIOD,LSM6DSL_INT1_EXTI11_Pin, GPIO_PIN_RESET);
358       GPIO_InitStruct.Pin = LSM6DSL_INT1_EXTI11_Pin;
359       GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
360       GPIO_InitStruct.Pull = GPIO_PULLUP;
361       HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
362       //led
363       HAL_GPIO_WritePin(GPIOB, LED2_Pin, GPIO_PIN_RESET);
364       GPIO_InitStruct.Pin = LED2_Pin;
365       GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
366       GPIO_InitStruct.Pull = GPIO_NOPULL;
367       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
368       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
369     }
370
```

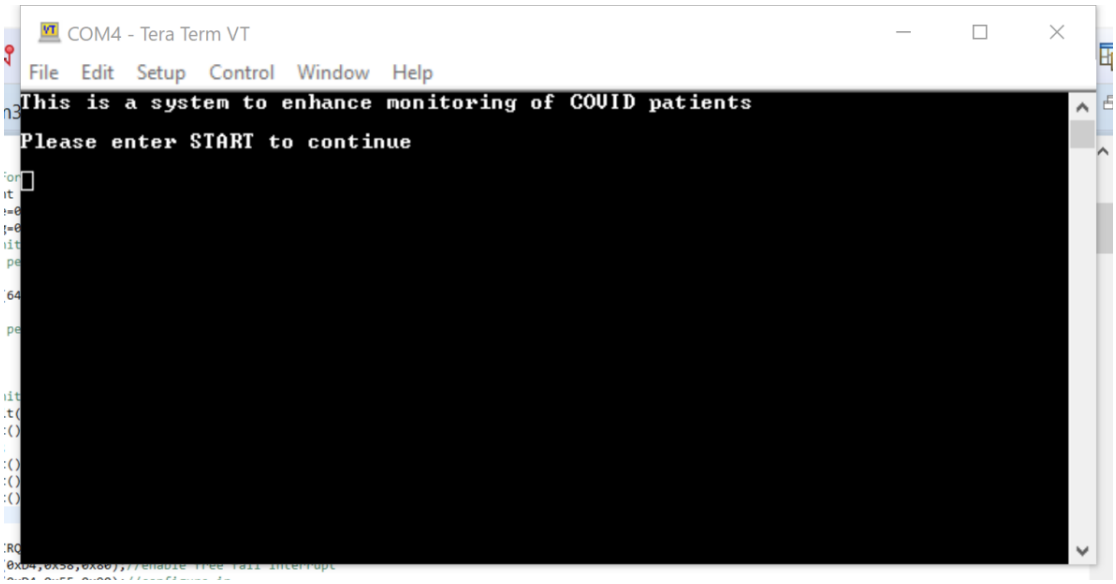
```
371 static void UART1_Init(void)
372 {
373     /* Pin configuration for UART. BSP_COM_Init() can do this automatically */
374     __HAL_RCC_GPIOB_CLK_ENABLE();
375     GPIO_InitTypeDef GPIO_InitStruct = {0};
376     GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
377     GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_6;
378     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
379     GPIO_InitStruct.Pull = GPIO_NOPULL;
380     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
381     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
382
383     /* Configuring UART1 */
384     huart1.Instance = USART1;
385     huart1.Init.BaudRate = 115200;
386     huart1.Init.WordLength = UART_WORDLENGTH_8B;
387     huart1.Init.StopBits = UART_STOPBITS_1;
388     huart1.Init.Parity = UART_PARITY_NONE;
389     huart1.Init.Mode = UART_MODE_TX_RX;
390     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
391     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
392     huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
393     huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
394     if (HAL_UART_Init(&huart1) != HAL_OK)
395     {
396       while(1);
397     }
398
399 }
```

```
400 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
401     {
402
403             if(GPIO_Pin==GPIO_PIN_13)//mode_toggle
404                buttonpress=1;
405             if(GPIO_Pin==GPIO_PIN_11)//Free fall
406                 fall_warning=1;
407     }
```

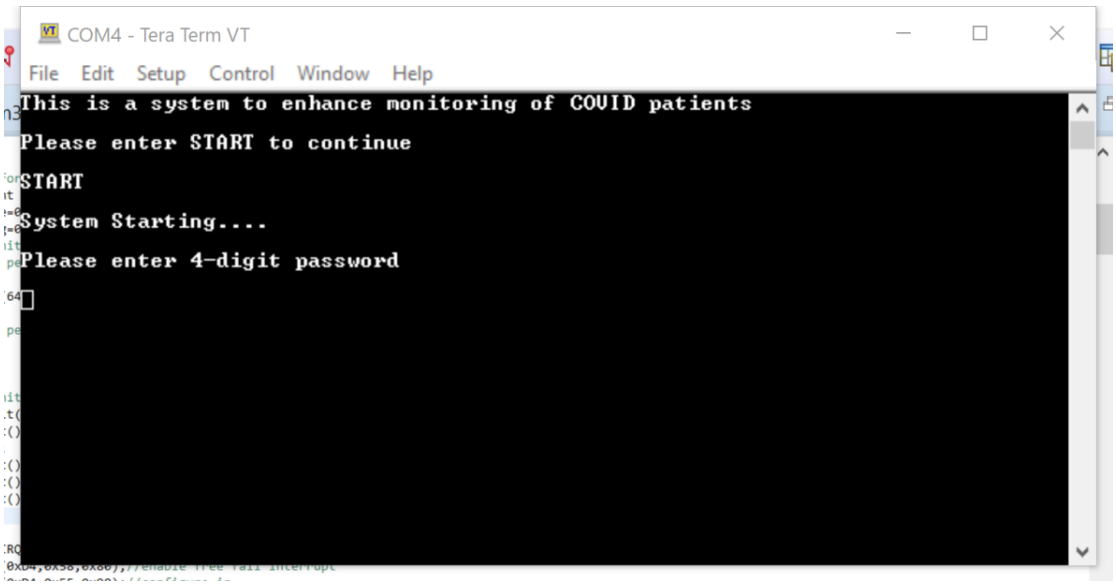
# 5. Enhancement

This enhancement consists of two parts, including Welcome Page, and Password Page. The Welcome Page requires user to type in “START” to start the whole system. This function can prevent users from mistouch. After typing “START”, the system will then switch to Password Page. Users are required to type in the correct password (0000) to start the Healthy Mode. Otherwise, this system will not be switched. Also, users could type in “EXIT” or “Exit” or “exit” to exit from password page and enter welcome page. This function can ensure the system not be used by other unauthorized users.

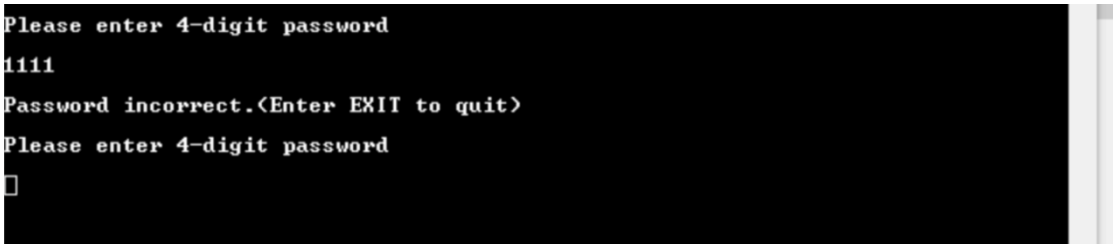
- Enhancement demonstration:
  A. Welcome Page




  B. Password Page

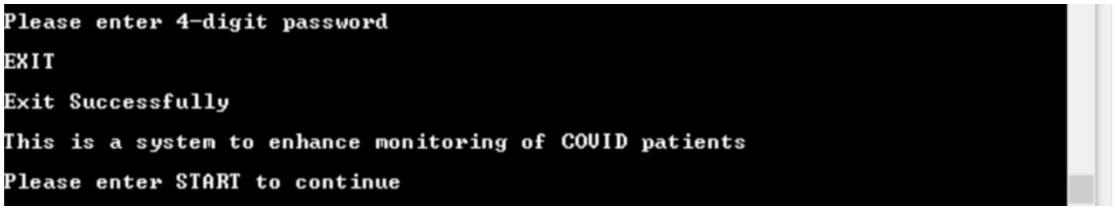



C. Wrong password




D. Correct password


E. Exit from password Page




- Enhancement code

A. Line 270- Line 301: The system will verify the correctness of the entered password. The system will compare the string user typed in with “0000”, “EXIT”, “Exit”, and “exit” to determine the system’s next mode.
mode=3: Password Page
memset: clear the content of buffer to reuse this variable.
“\r”: move cursor to the first letter of this line
“\n”: move cursor to next line
buffer: the string user typed in

```
270         if(mode==3)
271         {
272               char enterpassword[]="Please enter 4-digit password\n\r";
273               HAL_UART_Transmit(&huart1, (uint8_t*)enterpassword, strlen(enterpassword),0xFFFF);
274               memset(buffer,0,strlen(buffer));
275               len=0;
276                do
277                {
278                    HAL_UART_Receive(&huart1,(uint8_t*)&data,1,0xFFFF);
279                    if(data!='\r'&& data!='\n')
280                    {
281                        len++;
282                        buffer[len-1]=data;
283                    }
284                }
285                 while((len<=64)&&(data!='\r'));
286                 buffer[len]='\0';
287
288              if(strcmp(exitmode1,buffer)==0||strcmp(exitmode2,buffer)==0||strcmp(exitmode3,buffer)==0)
289               {
290
291                   mode=4;
292                   HAL_UART_Transmit(&huart1, (uint8_t*)exitmode, strlen(exitmode),0xFFFF);
293               }
294                   else
295                   { if(strcmp(password,buffer)==0)
296                           {mode=0;}
297                       else
298                           HAL_UART_Transmit(&huart1,(uint8_t*)wrong,strlen(wrong),0xFFFF);
299
300                   }
301         }
```

B. Line 305- Line 328: Users should type “START” to start the monitoring system. The system will compare the string user typed in with “START” to determine the system’s next mode.
mode=4: Welcome Page

```
305     if(mode==4)
306     {
307       char mode_start[]="This is a system to enhance monitoring of COVID patients\n\rPlease enter START to continue\n\r";
308       char start[]="System Starting....\n\r";
309       HAL_UART_Transmit(&huart1,(uint8_t*)mode_start,strlen(mode_start),0xFFFF);
310       memset(buffer,0,strlen(buffer));
311       len=0;
312       do
313        {
314            HAL_UART_Receive(&huart1,(uint8_t*)&data,1,0xFFFF);
315            if(data!='\r'&& data!='\n')
316            {
317                len++;
318                buffer[len-1]=data;
319            }
320        }
321         while((len<=64)&&(data!='\r'));
322         buffer[len]='\0';
323       if(strcmp("START",buffer)==0)
324       {
325          HAL_UART_Transmit(&huart1,(uint8_t*)start,strlen(start),0xFFFF);
326          mode=3;
327       }
328     }
```

# 6. Problems and solutions

We encountered three major problems in this assignment.

- Problem1: How can we decide the switch of the mode by checking the button is pressed twice within 1 second?
  Solution1: Use press_time to record the first time the button is pressed. Check the difference of the two pressing times. If the difference of two pressing time is within 1000 (1s), mode switched. If the difference time is beyond 1000, the press_time will be updated to second time of button-pressing. And the system waits for another button press. In other words, press_time will record the last time of button press. And system will check the difference between the current press time and last press time (press_time).
- Problem2: Entering difference strings needs numerous arrays to read them. How to simplify this process?
  Solution2: memset function. This function can clear the array, so that we can reuse one array for several times.
- Problem3: Printing “Please enter 4-digit password” will also include “\r” and “\n” to place

the cursor to the beginning of the next line. However, “\r” and “\n” will also be recorded in the reading array (it is called “data” in our code), which will affect the strcmp function. How can we get rid of it?
Solution3: Judgement statement. If the element in the reading array (“data”) is “\r” or “\n”, the code will skip this element. Except “\r” and “\n”, other elements will be transferred to another new array (it is called “buffer” in our code). Subsequently, the new array (“buffer”) will participate in strcmp function.

# 7. Issue and suggestions

This project is suggested to do further improvement on Internet connectivity. Future designers may work more on Wi-Fi connection and database establishment. COPEMON could be further developed as a node to a bigger IoT monitoring system. In addition, shared database of this IoT monitoring system is also preferred. Medical practitioners could acquire, analyze, and summarize patient’s body situation characteristics from this database.

# 8. Conclusion

In this assignment, in order to monitor COVID patients’ health situation, we finished to design a preliminary two-mode monitoring system called COPEMON by using STM32. Health data detection, analysis, and printing are achieved by implementing different kinds of interfacing with embedded system. Hope everyone takes great care in this pandemic era, and none of my beloved professors, tutors, and friends use my COPEMON.