Neo Yu Yao Terence A0164651E, Neo Jia Wen Rachel A0176763R GA:

Table of Contents

[Introduction and objectives 1](#_Toc529643541)

[Flowcharts describing the system design and processes 2](#_Toc529643542)

[Detailed implementation 6](#_Toc529643543)

[Enhancement 6](#_Toc529643544)

[Significant problems encountered and solutions proposed 7](#_Toc529643545)

[Reading of Temperature Sensor causes system lag 7](#_Toc529643546)

[Issues or suggestions 8](#_Toc529643547)

[Conclusion 8](#_Toc529643548)

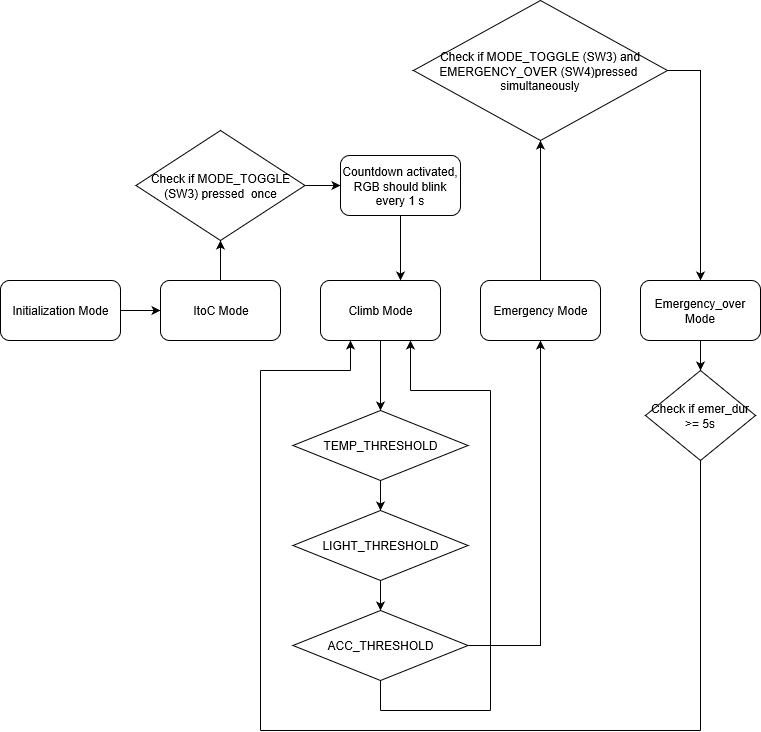
# Introduction and objectives

In this assignment, our group is tasked to implement a fitness tracking system, **FitNUS**. The main purpose of **FitNUS** is to boost daily workouts and make them easier to achieve. **FitNUS** detects acceleration, light and temperature changes. **FitNUS** sends data periodically to a server known as **FiTrackX**. The XBee RF module acts as a low powered wireless communication device that sends collected data to **FiTrackX**.

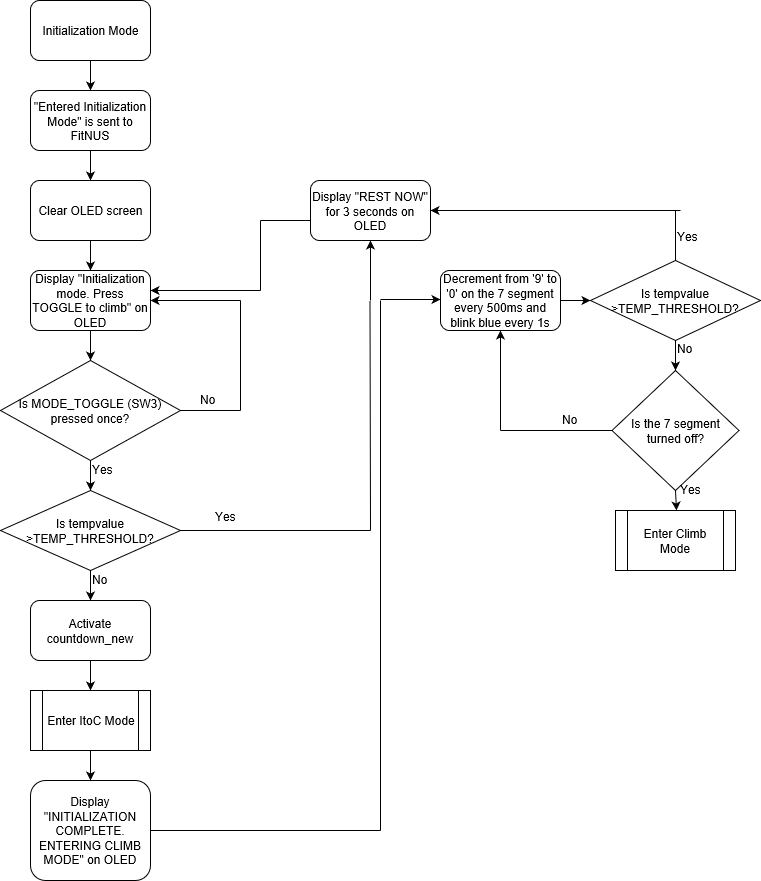
**FitNUS** has 3 modes of operation: Initialization, Climb and Emergency modes, and will transmit to **FiTrackX** is certain conditions are met. We also implemented ItoC and Emergency\_over modes which is later described in our report. Initialization mode would be active when the FitNUS system is first switched on. Climb mode would be active when MODE\_TOGGLE (SW3) is activated. Emergency mode would be active when the user decides to trigger fall detection by slightly shaking the board in Climb mode.

# Flowcharts describing the system design and processes

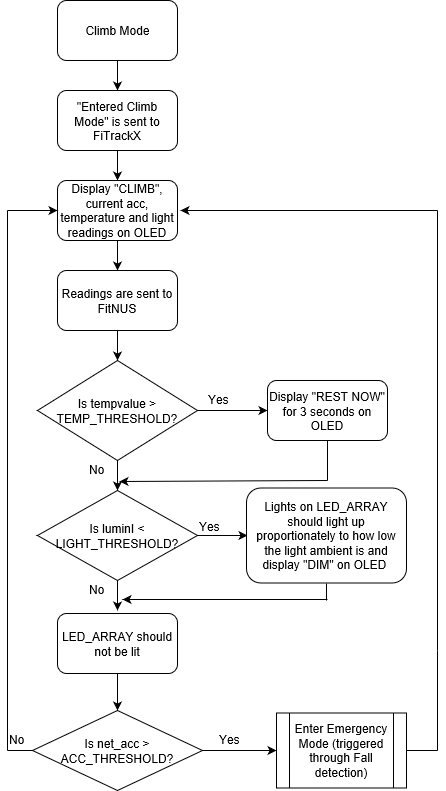
**int main**



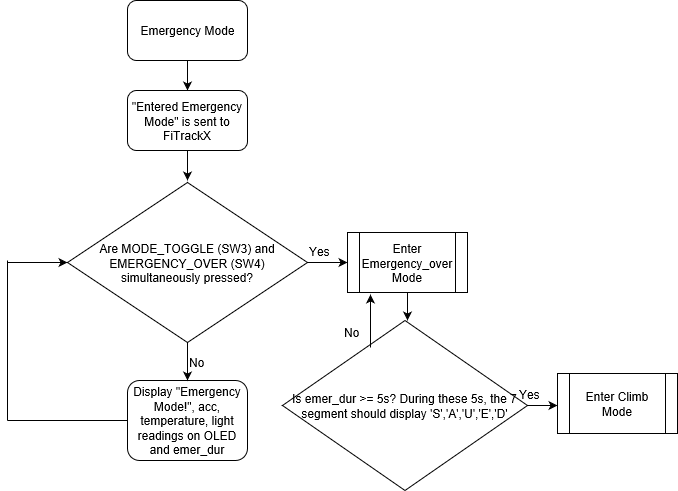
**Initialization Mode**



**Climb Mode**



**Emergency Mode**



# Detailed implementation

|  |  |
| --- | --- |
| **int** **main** (**void**) {  init\_everything();  acc\_read(&x, &y, &z);  xoff = 0-x;  yoff = 0-y;  zoff = 64-z;  moveBar(1, dir);  oled\_clearScreen(*OLED\_COLOR\_BLACK*);  led7seg\_setChar(0xFF, *TRUE*);  **while** (1){  **if** (state == *Initialization*){  do\_Initialization();  }  **if** (state == *ItoC*){  do\_toclimb();  }  **if** (state == *Climb*){  do\_Climb();  }  **if** (state == *Emergency*){  do\_Emergency();  }  **if** (state == *Emergency\_over*){  do\_Emergency\_over();  }  }  } | When the system is first switched ON, all the peripherals will be initialized and the interrupts will be enabled. **FitNUS** will then enter Initialization Mode. The OLED should display “Initialization mode. Press TOGGLE to climb”. Sensors will not be reading any data and no UART transmission should be sent to **FitNUS**.  When SW3 is pressed, **FitNUS** will enter Climb Mode. The OLED should display “CLIMB”. The 7-segment will display the countdown, decrementing from ‘9’ to ‘0’. The sensors will obtain values and store them in variables to be utilised by other functions. The OLED should then display the values obtained.  When fall detection is triggered, **FitNUS** will enter Emergency Mode. The OLED should display “EMERGENCY!”.  In order to make our code less complex, we segmented the codes for the operation modes into several functions outside of the main function and then integrated them back using the conditional while loop. |

|  |  |  |
| --- | --- | --- |
| The function “init\_everything()” initializes all the peripherals required for **FitNUS** to work properly (e.g. i2c, GPIO, uart, OLED, etc). We also segmented the codes for the initialization of the peripherals into several functions outside and integrated them back into the init\_everything() function.  Additionally, when **FitNUS** is first switched ON, it would configure SysTick to generate an interrupt every 1ms.   |  | | --- | | **void** **SysTick\_Handler** (**void**){  msTicks++;  } |   The SysTick handler updates msTicks every 1ms to give a real time reference to the system.  The GPIO interrupts are enabled to activate SW3, light sensor, joystick centre, joystick down, joystick right, joystick up and joystick left respectively. | **static** **void** **init\_everything**(){  init\_i2c();  init\_ssp();  init\_GPIO();  init\_uart();  SysTick\_Config(SystemCoreClock/1000);  temp\_init(&Get\_Time);  pca9532\_init();  joystick\_init();  acc\_init();  oled\_init();  led7seg\_init();  speaker\_init();  rgb\_init();  lightSenIntInit();  LPC\_GPIOINT ->IO0IntEnF |= 1<<4;  LPC\_GPIOINT ->IO2IntEnF |= 1<<5;  LPC\_GPIOINT ->IO0IntEnF |= 1<<17;  LPC\_GPIOINT ->IO0IntEnF |= 1<<15;  LPC\_GPIOINT ->IO0IntEnF |= 1<<16;  LPC\_GPIOINT ->IO2IntEnF |= 1<<3;  LPC\_GPIOINT ->IO2IntEnF |= 1<<4;  NVIC\_EnableIRQ(*EINT3\_IRQn*);} |

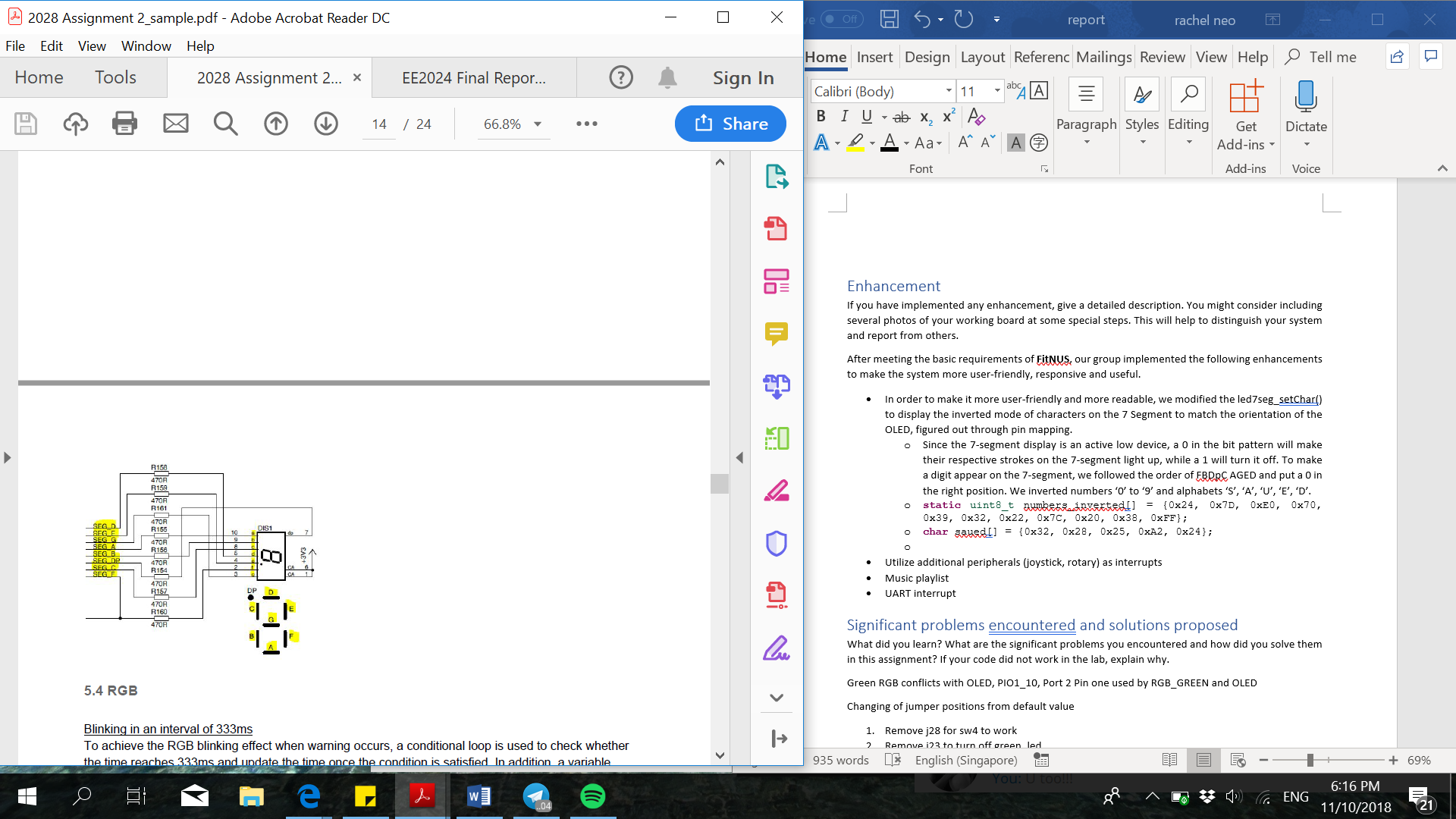
# 

# Enhancement

If you have implemented any enhancement, give a detailed description. You might consider including several photos of your working board at some special steps. This will help to distinguish your system and report from others.

After meeting the basic requirements of **FitNUS**, our group implemented the following enhancements to make the system more user-friendly, responsive and useful.

* In order to make it more user-friendly and more readable, we modified the led7seg\_setChar() to display the inverted mode of characters on the 7 Segment to match the orientation of the OLED, figured out through pin mapping.
  + Since the 7-segment display is an active low device, a 0 in the bit pattern will make their respective strokes on the 7-segment light up, while a 1 will turn it off. To make a digit appear on the 7-segment, we followed the order of FBDpC AGED and put a 0 in the right position. We inverted numbers ‘0’ to ‘9’ and alphabets ‘S’, ‘A’, ‘U’, ‘E’, ‘D’.
  + **static** uint8\_t numbers\_inverted[] = {0x24, 0x7D, 0xE0, 0x70, 0x39, 0x32, 0x22, 0x7C, 0x20, 0x38, 0xFF};
  + **char** saued[] = {0x32, 0x28, 0x25, 0xA2, 0x24};



* Music playlist
* UART interrupt

# Significant problems encountered and solutions proposed

What did you learn? What are the significant problems you encountered and how did you solve them in this assignment? If your code did not work in the lab, explain why.

## Reading of Temperature Sensor causes system lag

The temp\_read() function provided in the temperature sensor library is a blocking function, containing lines such as shown below that waits for GET\_TEMP\_STATE to change before the function proceeds to the next instruction.

**while**(GET\_TEMP\_STATE == state);

Furthermore, there is a for loop that loops for up to 340 times when both pins U7-TSI0 and U7-TSI1 are set to 0. The purpose of this, we presume, is to do a smoothing on the temperature reading across the set time interval to minimise noise. As our programme is not reading the sensor in real time, but rather at a fixed time interval set by sensor\_refresh\_ticks, smoothing in sensor readings are not necessary for us. We will however, take the average temperature reading over 10 periods to get a more reliable reading. We have hence written our own temp read function using an interrupt.

**void** **EINT3\_IRQHandler**(**void**){

// Temperature sensor

**if** ((LPC\_GPIOINT ->IO0IntStatR>>2) & 0x1){

LPC\_GPIOINT ->IO0IntClr = 1<<2; //clear the interrupt

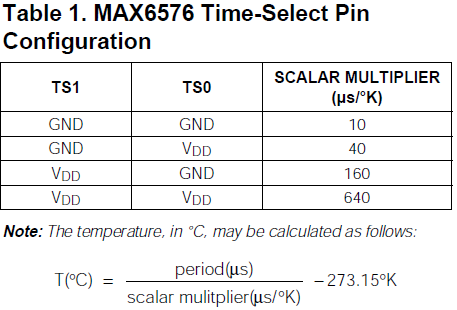
temp\_periods++;

**if** (temp\_periods == 1) t1 = Get\_Time();

**else** **if** (temp\_periods == 10) t2 = Get\_Time();

**else** **if** (temp\_periods == 20) temp\_periods = 0;

}

By this formula given in the datasheet, we can define our function as shown.

// Replacement for slow temp\_read() driver function

int32\_t **fast\_temp\_read** (**void**){

//10T(C) = (period (us) / scalar\_div10) - 2731 K

**return** **abs**(t2-t1)\*1000/(10\*TEMP\_SCALAR\_DIV10)-2731;

}

## Green RGB conflicts with OLED

PIO1\_10, Port 2 Pin one used by RGB\_GREEN and OLED

Changing of jumper positions from default value

1. Remove j28 for sw4 to work
2. Remove j23 to turn off green\_led

## Blue RGB conflicts with Speaker

# Issues or suggestions

These feedbacks, whether positive or negative, will not affect your marks in any way, but will make the report more complete.

As this is the first hardware programming project we did on LPC, there were many times when we got stuck, baffled by lines of codes that seems to work but did not. Fortunately, we were able to readily consult the various teachers, lab staff, and graduate assistances who are not only very knowledgeable, spotting our errors instantly, but also extremely patient when explaining the concepts to us. And we are very grateful for you all!

# Conclusion