https://github.com/luketschepen/EE2361Final/tree/main

**INTRO- Description of the Library and its overall functions**

**Members: Dhayalan Balasubramanian, Luke Tschepen, Nicholas Kang, Cole Jaeger.**

The Blood Oxygen Sensor project integrates the MAX30102 sensor for pulse oximetry and heart-rate monitoring, alongside the Grove RGB LCD module for versatile display feedback. Physical input is managed by the button_switch.c file, handling interaction with a button switch for toggling display and capturing button events. The algorithm.c file computes heart rate and SpO2 levels using peak detection techniques and precise algorithms on sensor data. The I2C.h file handles essential I2C communication for interfacing with the sensor and other peripherals.

## Separate Descriptions

MAX 30102 Description:
The code implements functionalities for interfacing with the MAX30102 sensor, which is a pulse oximeter and heart-rate sensor. It includes functions for initializing the sensor, configuring its settings, reading sensor data, processing the signals, and displaying the results.

The Grove RGB LCD module is a multifunctional display that integrates an RGB backlight and a character LCD. It is designed to provide a versatile display solution for various electronic projects. This document outlines the functionality and usage of the Grove RGB LCD module within the context of the Blood Oxygen Sensor project.

The button_switch.c file provides functionality for interfacing with a physical button switch in the Blood Oxygen Sensor project. It includes functions for initializing the button, detecting button presses, and determining the state of the button. The button switch is connected to RB7 (Pin 6) of the microcontroller, and its usage involves capturing button press events and toggling the display of heart rate information

The algorithm.c file is a crucial component of the Blood Oxygen Sensor project, dedicated to computing heart rate and SpO2 levels. It implements peak detection techniques on data from IR and red sensors to identify PPG signal peaks for heart rate determination. Additionally, it employs a precise algorithm to derive SpO2 values from the ratio of AC to DC components in the IR and red signals.

The I2C.h file contains functions for initializing and managing I2C communication on PIC24 microcontrollers. It includes functions for initializing I2C parameters, initiating start and stop conditions, writing and reading data over I2C, handling acknowledgments, and performing repeated start conditions.

**Hardware description - what device(s), part numbers, links, etc.**
SNAP: MP LAB SNAP PG164100
https://ww1.microchip.com/downloads/aemDocuments/documents/DEV/ProductDocuments/UserGuides/50002787C.pdf

PIC24: PIC24FJ64GA002
https://ww1.microchip.com/downloads/en/DeviceDoc/39881e.pdf

Sensor: MAX30102
https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf

RGB LCD: Grove-LCD RGB Backlight JHD1313
https://files.seeedstudio.com/wiki/Grove_LCD_RGB_Backlight/res/JHD1313%20FP-RGB-1%201.4.pdf

Button: Square Tactile Pushbutton Switch SPST Momentary -ELSW-TPB-SPST
https://www.ckswitches.com/media/1462/pts125.pdf

**Full documentation- description - what device(s), part numbers, links, etc.**

**MAX 30102 Description:**
The code implements functionalities for interfacing with the MAX30102 sensor, which is a pulse oximeter and heart-rate sensor. It includes functions for initializing the sensor, configuring its settings, reading sensor data, processing the signals, and displaying the results.

Devices and Parts:
SNAP
PIC24
MAX30102 Sensor
RGB LCD display
Button switch

Links to datasheets and part numbers:

SNAP: MP LAB SNAP PG164100
https://ww1.microchip.com/downloads/aemDocuments/documents/DEV/ProductDocuments/UserGuides/50002787C.pdf

PIC24: PIC24FJ64GA002
https://ww1.microchip.com/downloads/en/DeviceDoc/39881e.pdf

Sensor: MAX30102
https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf

RGB LCD: Grove-LCD RGB Backlight JHD1313
https://files.seeedstudio.com/wiki/Grove_LCD_RGB_Backlight/res/JHD1313%20FP-RGB-1%201.4.pdf

Button: Square Tactile Pushbutton Switch SPST Momentary -ELSW-TPB-SPST
https://www.ckswitches.com/media/1462/pts125.pdf

Public Functions:
Initialization and Configuration:
max30102_init(void): Initializes the MAX30102 sensor.
max30102_setup_spo2(): Configures the MAX30102 sensor for SpO2 measurement.
max30102_setup_default(void): Sets up the MAX30102 sensor to the default state.
max30102_setup_display(void): Configures the display for showing SpO2 and heart rate data.
uint32_t max30102_read_partID(); Reads the Part ID(0x15) and was written to ensure the hardware, sensor, and I2C protocol were working.

Data Acquisition and Processing:
max30102_process_signals(): Processes the sensor signals to calculate SpO2 and heart rate.
max30102_readTemp(): Reads the die temperature from the sensor.
calculateMovingAverage(uint32_t *buffer, int size): Calculates the moving average of the data buffer.

Buffer Management:
max30102_RED_putBuff(long int data): Inserts data into the Red LED buffer.
max30102_IR_putBuff(long int data): Inserts data into the Infrared LED buffer.
max30102_RED_getBuff(): Retrieves data from the Red LED buffer.
max30102_IR_getBuff(): Retrieves data from the Infrared LED buffer.

Control and Configuration:

Functions to configure various parameters such as LED mode, ADC range, sample rate, pulse width, FIFO settings, etc.

Utility:
max30102_shutdown(void): Shuts down the sensor.
softReset(void): Performs a soft reset of the sensor.

Arguments and Outputs:
Many functions take parameters to configure specific settings on the sensor, such as LED mode, sample rate, pulse width, etc.
The functions for inserting and retrieving data from the buffers (max30102_RED_putBuff, max30102_IR_putBuff, max30102_RED_getBuff, max30102_IR_getBuff) accept and return data values.
Functions for processing signals (max30102_process_signals()) calculate SpO2 and heart rate and store the results in global variables.
Display functions (max30102_setup_default, max30102_setup_display) output data to an RGB LCD display.

**Grove RGB LCD Documentation**

Description:

The Grove RGB LCD module is a multifunctional display that integrates an RGB backlight and a character LCD. It is designed to provide a versatile display solution for various electronic projects. This document outlines the functionality and usage of the Grove RGB LCD module within the context of the Blood Oxygen Sensor project.

Devices and Parts:

SNAP
PIC24 microcontroller
MAX30102 Sensor
RGB LCD display
Button switch
Links:

Public Functions:

// Function prototypes

```
void delay_ms(unsigned int ms); // Delays execution for a specified number of milliseconds.
void rgb_cmd(char Package); // Sends a command to control RGB LEDs.
void setReg(unsigned char reg, unsigned char dat); // Sets a register with specified data.
void setRGB(unsigned char r, unsigned char g, unsigned char b); // Sets RGB LED color with specified red, green, and blue values.
void rgb_clr(); // Clears RGB LED color.
void rgb_home(); // Sets RGB LEDs to default state.
void noDisplay(); // Turns off display.
void display(); // Turns on display.
void blinkLED(void); // Blinks an LED.
void noBlinkLED(); // Stops LED blinking.
void setColorAll(); // Sets color for all LEDs.
void setColorWhite(); // Sets color to white for all LEDs.
void setColorRed(); // Sets color to red for all LEDs.
void setCursor(uint8_t x, uint8_t y); // Sets cursor position on a display.
void grovergb_init(void); // Initializes Grove RGB module.
void printStr(const char s[]); // Prints a string on a display.
void printChar(char myChar); // Prints a character on a display.
```

Initialization and Configuration:
grovergb_init(): Initializes the Grove RGB LCD module to be compatible with the PIC24 microcontroller using I2C communication.

Data Acquisition and Processing:
Not applicable in this document. (Related to the blood oxygen sensor functionality)

Buffer Management:
Not applicable in this document.

Control and Configuration:
setColorAll(): Turns off all colors on the display.
setColorWhite(): Sets the screen color to white.
setColorRed(): Sets the screen color to red.
blinkLED(): Flashes the specified LED registers.
noBlinkLED(): Stops the flashing of the LED registers.

Utility:
printStr(const char s[]): Prints a string to the display.
printChar(char myChar): Prints a character to the display.

Arguments and Outputs:
grovergb_init(): No arguments or outputs.
setColorAll(), setColorWhite(), setColorRed(), blinkLED(), noBlinkLED(): No arguments or outputs.
printStr(const char s[]): Takes a string s as input and prints it on the display.
printChar(char myChar): Takes a character myChar as input and prints it on the display.

## Basic usage example- bare minimum to test hardware functionality

In order to test the functionality of the hardware the max30102_read_ partID() function can be called. After calling this function the sensors will be configured and running and will display the part ID on the LCD.

You can also display the raw readings from the sensor using the following configuration.

```
int main(void) {
   setup();  // Perform all initial setup tasks

   while(1) {
         rgb_clr();
         getRead();
         printf("IR Reading: %u\n ", max30102_IR_getBuff());
        printf("RED Reading: %u\n", max30102_RED_getBuff());


   }
   return 0;
}
```

## Advanced usage example - covering all the functions and feature

An advanced usage example could be displaying either heart rate or Sp02 and controlling the display with the press of a button similar to what we demo'd.

```
void max30102_process_signals(){
     getRead();
     // Read 100 samples into the buffers
     for (int i = 0; i < 100; i++) {
             irBuffer[i] = max30102_IR_getBuff();    // Get one IR sample
            redBuffer[i] =  max30102_RED_getBuff();  // Get one Red sample
     }
```

```
    // Process the collected data
    maxim_heart_rate_and_oxygen_saturation(irBuffer, 100, redBuffer, &spo2, &validSpo2,
&heartRate, &validHeartRate);
    if (spo2 > 85) {
      // Update heart rate buffer with latest value
      for (int i = 0; i < MOVING_AVG_SIZE_HR - 1; i++) {
        heartRateBuffer[i] = heartRateBuffer[i + 1];
      }
      if (heartRate > 0){
        heartRateBuffer[MOVING_AVG_SIZE_HR - 1] = heartRate;


      }


      // Update SpO2 buffer with latest value
      for (int i = 0; i < MOVING_AVG_SIZE_SPO2 - 1; i++) {
        spo2Buffer[i] = spo2Buffer[i + 1];
      }
      if (spo2 > 0){
        spo2Buffer[MOVING_AVG_SIZE_SPO2 - 1] = spo2;


      }


      // Calculate moving average of heart rate
      heartRateAvg = calculateMovingAverage(heartRateBuffer, MOVING_AVG_SIZE_HR);
      spo2Avg = calculateMovingAverage(spo2Buffer, MOVING_AVG_SIZE_SPO2);


      max30102_setup_display();


    }
    else {


      max30102_setup_default();
    }
}
```

**int main(void) {**

```
  setup();  // Perform all initial setup tasks
  while(1) {
        rgb_clr(); //clear screen
        max30102_process_signals(); //process readings and update display

  }
  return 0;
}
```

If you wanted to build upon this even further you could add the following. An even more advanced usage example could be implementing a power saving mode, By detecting how long it has been since the previous valid reading. If it has been over two minutes then go into “power saving mode” and the sensor would stop reading. The button could then be used to turn the sensor back on and continue reading. Another possible thing to add would be to specify your age and then corresponding “zones'' would be set. Then based on the readings for heart rate and SpO2, we’d display either a red yellow or green backlight for how “healthy” the values are.

```
int main(void) {
   setup();  // Perform all initial setup tasks
   readForAge();//function to take in age and set zones. Uses button to take age.
   turnOnPowerSaving();//function to turn on power saving mode


while(1) {
          powerSaveCheck();//check for readings and if sensor should stay on/off
          rgb_clr();
          max30102_process_signals()
           }
   return 0;
```

}

## References used:

Basic Background Video - https://www.youtube.com/watch?v=V5UvNVQsUsY

## Sensor:

Guide to using Sensor -
https://www.instructables.com/Guide-to-Using-MAX30102-Heart-Rate-and-Oxygen-Sens/

OxSensor Example Library
https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library/blob/master/src/MAX30105.h

Oxsensor Arduino Tutorial
https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/

Data Sheet
https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf.

## LCD:

Grove LCD wiki
https://wiki.seeedstudio.com/Grove-LCD_RGB_Backlight/

Grove LCD datasheet
https://files.seeedstudio.com/wiki/Grove_LCD_RGB_Backlight/res/JHD1313%20FP-RGB-1%201.4.pdf

Other LCD datasheet
https://files.seeedstudio.com/wiki/Grove-LCD_RGB_Backlight/Grove-LCD_RGB_Backlight_V5.0_Datasheet.pdf

Other Other LCD datasheet
https://files.seeedstudio.com/wiki/Grove_LCD_RGB_Backlight/res/PCA9633.pdf

## PIC 24 Data Sheet:

**https://ww1.microchip.com/downloads/en/DeviceDoc/39881e.pdf**

**PIC 24 I2C:**

**https://ww1.microchip.com/downloads/en/devicedoc/70000195f.pdf**