Faculty of Engineering
Department of Electrical and Computer Engineering

EE2028
Microcontroller Programming and Interfacing

COvid Patients Enhanced Monitoring

Pojcharapol Leenukiat **A0219984L**
Mehedi Hasan Salim **A0223395A**

Lab B03 (Fri PM) **Group 9**

Table of Content

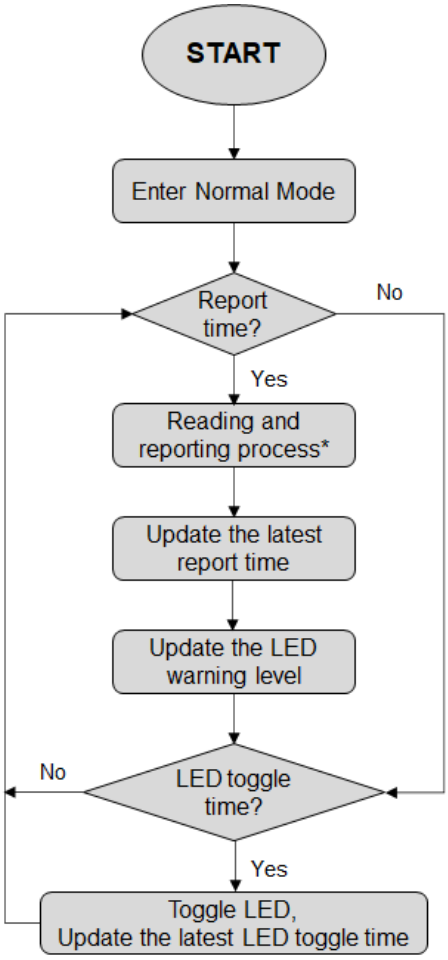
Introduction and Objectives

In this project, we aim to build a system, COPEMON(COvid Patients Enhanced Monitoring), in which the health status of the COVID-19 patients, especially elderly, is monitored. Our system is able to monitor the temperature and breathing of patients, using temperature, humidity and pressure sensors, as well as detecting if the patient has fallen using accelerometer and any usual movement using the gyroscope and magnetometer. This is done by sending warnings when the readings of these sensors do not meet the criteria of the predetermined threshold values.

This microcontroller used in this project is B-L475E-IOT01A, manufactured by STMicroelectronics. It has STM32L475VGT6 as the microprocessors, together with LED and on-board sensors, of which four are used in this project: HTS221 (Temperature sensor and Humidity sensor), LSM6DSL (Accelerometer and Gyroscope), LIS3MDL (Magnetometer), and LPS22HB (Pressure sensor). CHIPACU, the terminal program, is enabled by the Tera Term program, and DEBUG_CONSOLE in STM32CubeIDE is used in the programming process.

Flowcharts

1) Main Program Flowchart



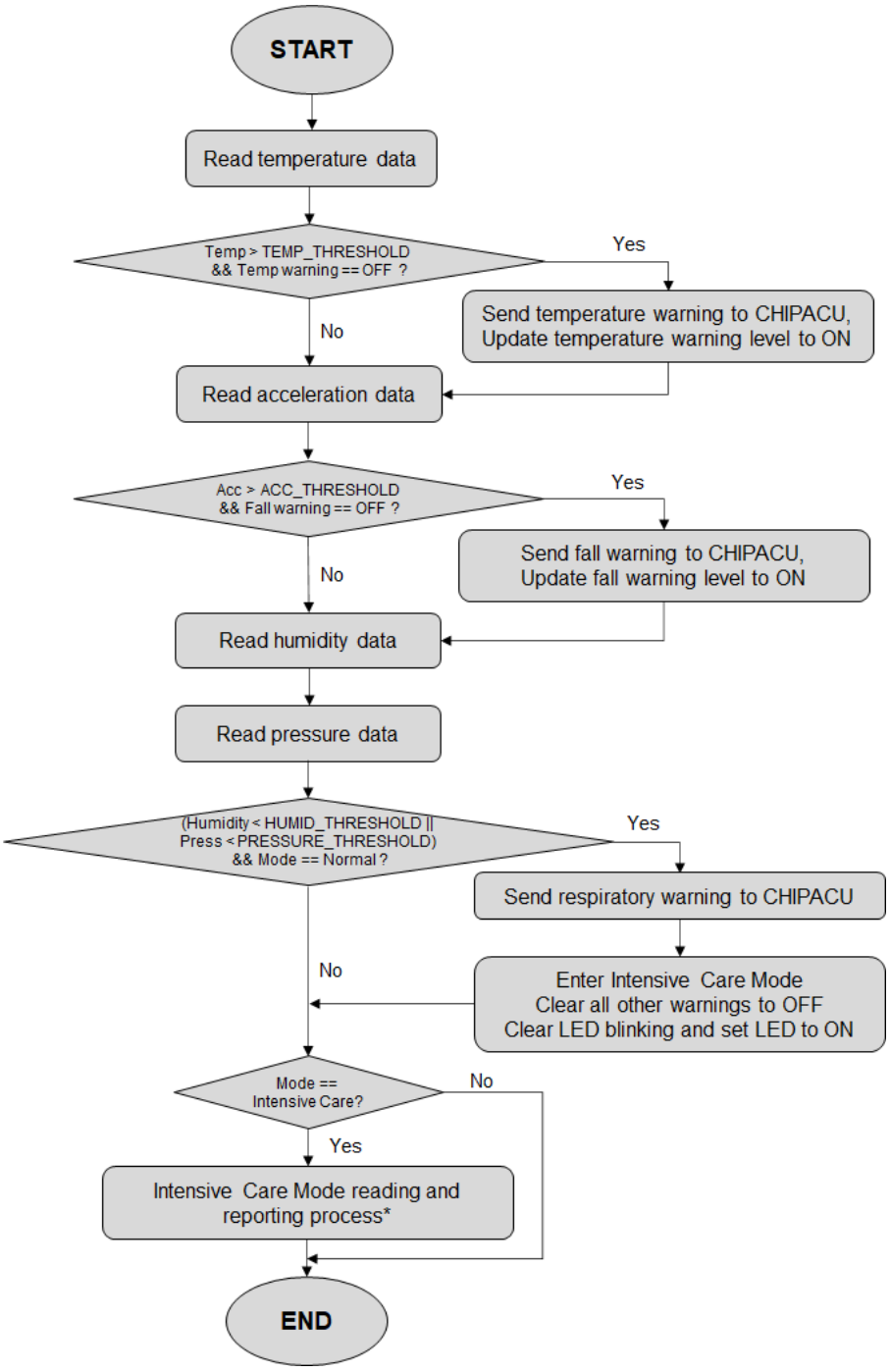
*For **Reading and reporting process**, refer to flowchart 2)

LED warning level is assigned according to the LED blinking frequency:

I. LED warning level 0 when there is no blinking
II. LED warning level 1 (5 Hz) for temperature warning and/or fall warning
III. LED warning level 2 (10 Hz) for pain warning and/or orientation warning

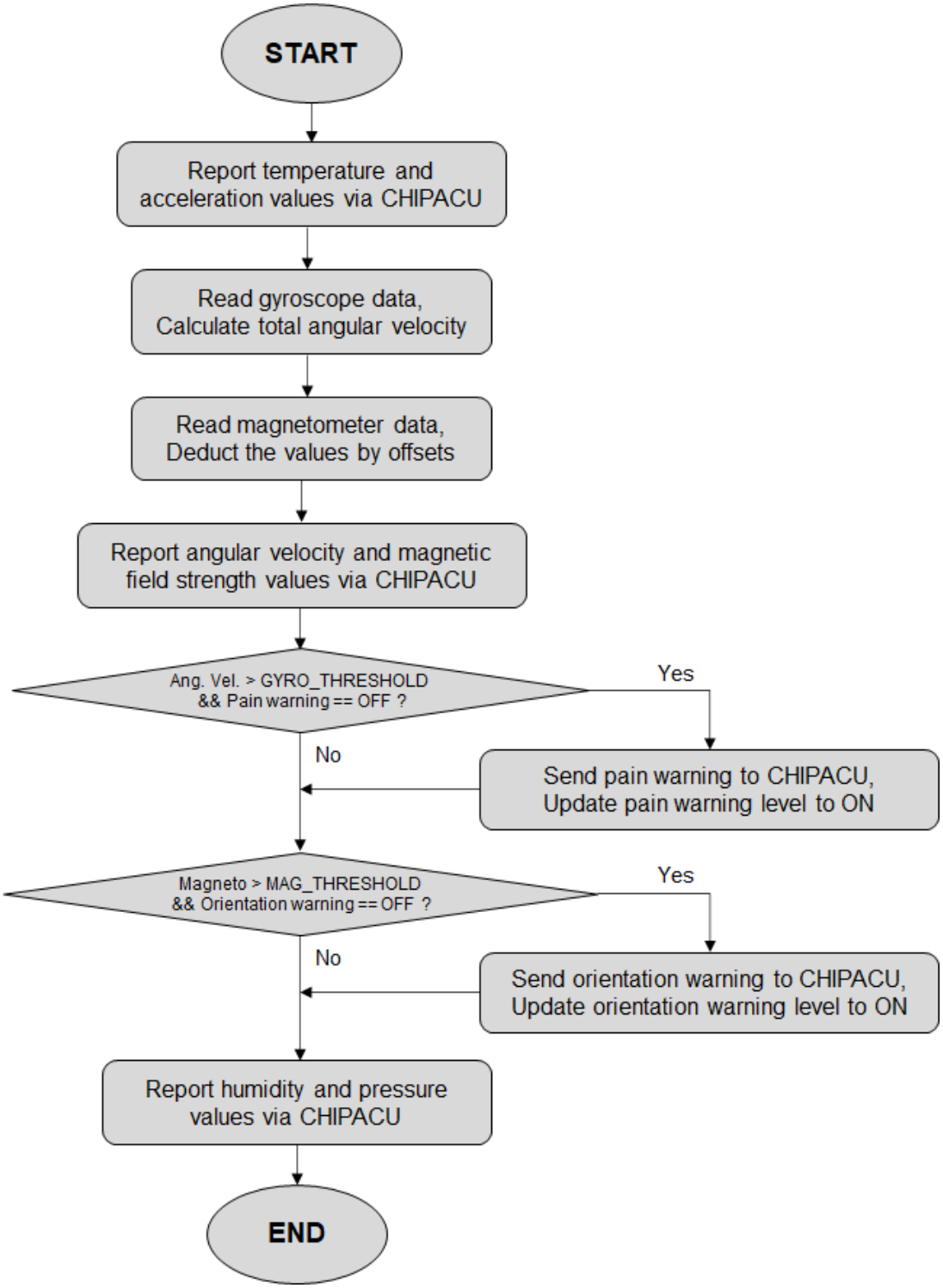
If there are both warnings which are categorized into level 1, and those which are categorized into level 2, the LED warning level will be set to 2.

2) Sensor Reading and Reporting Flowchart

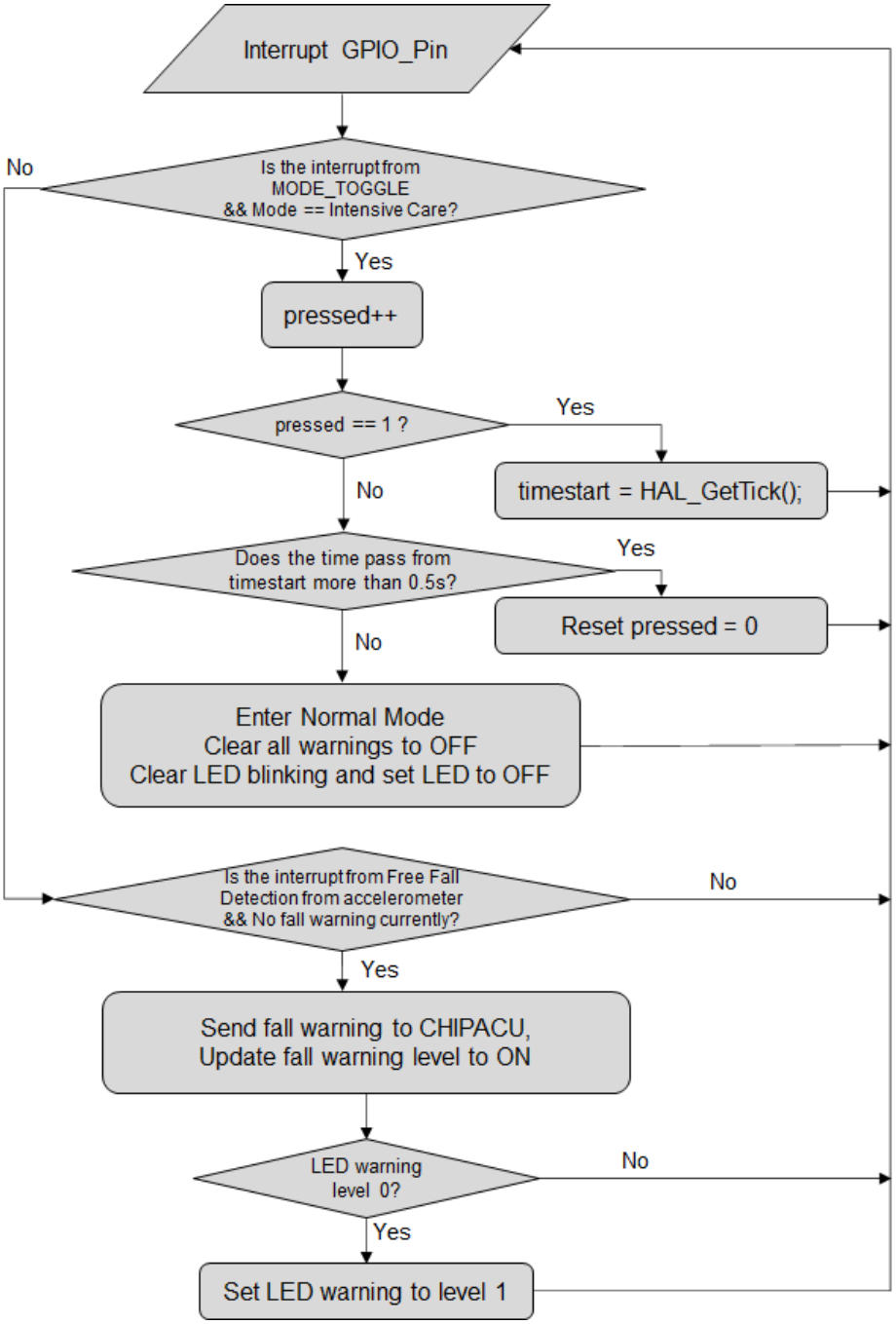


*For **Intensive Care Mode reading and reporting process**, refer to flowchart 3)

3) Intensive Care Mode Reading and Reporting Flowchart



4) Interrupt Flowchart



Detailed Implementation

After planning the program flow as shown in the flowcharts above, we started the coding from the most essential structure of the program - Respiratory Monitoring, as it would be the key for switching the mode from Normal to Intensive Care. Next, MODE_TOGGLE interrupt is implemented to complete the cycle switching the mode form Intensive Care back to Normal.

When the structure was ready, we started adding the readings of each sensor to the position planned in the flowchart. The units of the readings could be found in the respective sensor datasheet or the sensor file in the BSP folder, and a few lines of codes are written to convert them into the required units. During this process, DEBUG_CONSOLE is used to observe and verify the values.

In the next step, we started to implement the warning system (warning messages and LED warning), and set threshold values for each sensor from a trial-and-error process. When the system was working well, we replaced printing the values and the warning messages on DEBUG_CONSOLE with sending them via serial communication - UART to CHIPACU. (Tera Term)

Temperature Sensor (HTS221):
We used BSP_TSENSOR_ReadTemp() to get the temperature reading, and given that the temperature was in degree celsius, we did not do any conversion, and set a TEMP_THRESHOLD at 37.5 degree celsius [1] as the temperature above 37.5 is considered as having fever.

Accelerometer (LSM6DSL):
We used BSP_ACCELERO_AccGetXYZ() to get the X, Y and Z direction acceleration values and since the values were in mg(milli g-force), we divided the values by 1000 to get units in g, as per requirement. Then we set the ACC_THRESHOLD to 0.5 as rotating the board is not able to give values which are lower than 0.5 in all directions and only a fall can ensure the X, Y and Z values are all below 0.5. This is done so as the patient might move but not necessarily have fallen, and hence a false warning will not be sent.

Gyroscope (LSM6DSL):
We used BSP_GYRO_GetXYZ() to get angular velocity in the X, Y and Z directions in milli degrees per second(mdps). We divided the values by 1000 to convert to dps as it is easier to work with for average person and as we were supposed to have a combinational value, we calculated combinational angular_velocity according to the equation below:

$$angular\ velocity = \sqrt{gyro\_data[0]^2 + gyro\_data[1]^2 + gyro\_data[2]^2}$$

We set the GYRO_THRESHOLD as 2.2 because angular_velocity is 2 (+-0.1) at rest, and any slightest movement can be harmful at intensive care mode, so we set the value just slightly above 2.1 but also to account for slight changes due to the sensor errors that might change at times, we did not set it to 2.15.

Magnetometer (LSM6DSL):
We used BSP_MAGNETO_GetXYZ() to get the magnetometer values in X, Y and Z directions in mGauss. As such, we divided the values by 1000.0f to convert to Gauss so it is easier to work with. We set the MAG_THRESHOLD to -0.15 because the value was -0.15 at rest, however it needs to be calibrated based on the environment given that it is not constant everywhere.

Humidity Sensor(HTS221):
We used BSP_HSENSOR_ReadHumidity() to get the humidity reading in %. We set the HUMID_THRESHOLD to 65 as exhaled human breath has humidity of 65-91%[2].

Pressure Sensor(LPS22HB):
We used BSP_PSENSOR_ReadPressure() to get pressure in hectopascal(hPa). We set the PRESSURE_THRESHOLD to 1015hPa assuming that the pressure at rest is the default reading of 1008+-1 hPa, and that maximum possible pressure in the lungs of the patient is 1020 hPa.

Finally, an additional interrupt from the accelerometer (LSM6DSL) was implemented to detect free fall, as we had observed that the free fall event is usually short, and rarely happens during the sensor reading time (every 10 seconds). There are four registers in the peripheral to be written in order to enable an appropriate interrupt: [3]

1. TAP_CFG = 0x80

Table 183. TAP_CFG register

INTERRUPTS_ENABLE	INACT_EN1	INACT_EN0	SLOPE_FDS	TAP_X_EN	TAP_Y_EN	TAP_Z_EN	LIR

INTERRUPTS_ENABLE = 1 to enable the interrupt from LSM6DSL
LIR = 0 to turn off the latched mode - the interrupt signal is automatically reset when the interrupt condition is no longer valid or after a certain amount of time.

2. WAKE_UP_DUR = 0x00

Table 192. WAKE_UP_DUR register

FF_DUR5	WAKE_DUR1	WAKE_DUR0	TIMER_HR	SLEEP_DUR3	SLEEP_DUR2	SLEEP_DUR1	SLEEP_DUR0

FF_DUR5 = 0

3. FREE_FALL = 0x0F

Table 194. FREE_FALL register

FF_DUR4	FF_DUR3	FF_DUR2	FF_DUR1	FF_DUR0	FF_THS2	FF_THS1	FF_THS0

FF_DUR[4:0] = 00001b to minimize free fall duration, FF_DUR[5:0] = 000001b
FF_THS[2:0] = 111b to maximize free fall threshold, resulting in a very sensitive sensor

4. MD1_CFG = 0x10

Table 197. MD1_CFG register

INT1_INACT_STATE	INT1_SINGLE_TAP	INT1_WU	INT1_FF	INT1_DOUBLE_TAP	INT1_6D	INT1_TILT	INT1_TIMER

INT1_FF = 1 to connect the free fall interrupt to INT1 pin of LSM6DSL

Enhancement: Password Mode

If Password Mode is enabled, the user needs to enter the password in order to return to Normal Mode from Intensive Care Mode after MODE_TOGGLE is pressed twice. This would ensure a second layer of protection for patients, as only doctors and nurses who take care of the patient will know the password, and COPEMON will not return to Normal Mode when MODE_TOGGLE is pressed accidentally.

At the beginning when COPEMON is turned on, the user is asked whether they wish to use Password Mode. The user is able to type their input via CHIPACU, and the input will be passed to the main program through UART serial communication. In the case that the user wishes to use Password Mode, they will be able to choose their own password whose length is not more than 64 characters. The entered password is transferred to the main program using the same method, and the password is confirmed when Enter is pressed.

When MODE_TOGGLE is pressed twice, the user is asked to enter the password via terminal program, and the input is sent back to the main program using the same method of serial communication. If the user enters a wrong password, they will be able to try new input again three times, and if they are not able to enter the correct password, COPEMON will return to Intensive Care Mode reporting.

UART is a full-duplex serial communication. In the main requirement of the project, the message is sent to print on CHIPACU using the function HAL_UART_Transmit, while the Password Mode requires the communication from the terminal program back to the main program, which is enabled by HAL_UART_Receive. Working with strings in C, e.g. string storage, string comparison, also needs complex codes, therefore, pointers and string.h library are used to optimize the code in this part of the program.

Problems and Solutions

1. **Problem:** Due to the magnetic property of the USB ST-LINK port, the magnetic field strength obtained from the magnetometer is not accurate.

   Solution: Offsets are deducted from readings of the magnetometer.
   We determined the offset in X-axis by measuring value once, flipping the board to invert the X-axis, and measuring the value again when the X-axis is flipped. The difference between these two values is twice the earth's magnetic field strength in X-axis, and the average value of them is the magnetic field strength from the environment in X-axis. (ST-LINK port, nearby laptop, etc.) Therefore, we take the average value as the offset in X-axis. The same method applies to Y-axis and Z-axis.

2. **Problem:** As we needed to think of a common threshold for the accelerometer, we had difficulty in ensuring that the value we set is not due to the rotation of the board.

   Solution: We had to do trial-and-error to find a value, 0.5 as it was not possible for acceleration to be below 0.5g without a fall. Also, as the free fall event is usually short, the sensor interrupt enabled by LSM6DSL is used to detect even slight fall of the board.

Conclusion

The project explores microcontroller programming and interfacing, as well as system planning. Six sensors from four peripherals on B-L475E-IOT01A are studied along with their communication with the microprocessor via I2C protocol. UART serial communication enables the program to send and receive message messages with the terminal program on the computer. Interrupts from USER_BUTTON and LSM6DSL are also implemented to create a more interactive program. Finally, the program is edited and Password Mode is added to enhance the user experience.

Reference

[1] COVID-19 Symptoms, 2021. [Online] Available: https://safetravel.ica.gov.sg/health/covid19-symptoms [Accessed Nov. 12, 2021]
[2] Measurement of temperature and relative humidity in exhaled breath. [Online] Available: https://www.researchgate.net/publication/337064921_Measurement_of_temperature_and_relative_humidity_in_exhaled_breath [Accessed Nov. 12, 2021]
[3] *LSM6DSL: always-on 3D accelerometer and 3D gyroscope*.STMicroelectronics, 2017. [Online] Available: https://www.st.com/resource/en/datasheet/lsm6dsl.pdf