1) CODE - :

```
#include <MKL25Z4.h> //required header file
// I2C start, stop, and acknowledge macros
#define I2C_M_START     (I2C0->C1 |= I2C_C1_MST_MASK)
#define I2C_M_STOP      (I2C0->C1 &= ~I2C_C1_MST_MASK)
#define I2C_M_RSTART    (I2C0->C1 |= I2C_C1_RSTA_MASK)
#define NACK            (I2C0->C1 |= I2C_C1_TXAK_MASK)
#define ACK             (I2C0->C1 &= ~I2C_C1_TXAK_MASK)

// I2C transmitter and receiver mode macros
#define I2C_TRAN        (I2C0->C1 |= I2C_C1_TX_MASK)
#define I2C_REC         (I2C0->C1 &= ~I2C_C1_TX_MASK)

// I2C wait for acknowledge macro
#define I2C_WAIT        while ((I2C0->S & I2C_S_IICIF_MASK) == 0) {} \
                        I2C0->S |= I2C_S_IICIF_MASK;
#define BUSY_ACK        while (I2C0->S & 0x01)

// Function prototypes
void i2c_init(void);
void i2c_start(void);
void i2c_read_setup(uint8_t dev, uint8_t address);
uint8_t i2c_repeated_read(uint8_t isLastRead);
uint8_t i2c_read_byte(uint8_t dev, uint8_t address);
void i2c_write_byte(uint8_t dev, uint8_t address, uint8_t data);

// Initialize I2C
void i2c_init(void) {
   SIM->SCGC4 |= SIM_SCGC4_I2C0_MASK;
   SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;
   PORTE->PCR[24] |= PORT_PCR_MUX(5);
```

```
    PORTE->PCR[25] |= PORT_PCR_MUX(5);

    I2C0->F = (I2C_F_ICR(0x12) | I2C_F_MULT(0));
    I2C0->C1 |= I2C_C1_IICEN_MASK;
}

// Start function
void i2c_start(void) {
    I2C_TRAN;
    I2C_M_START;
}

// Single byte read
uint8_t i2c_read_byte(uint8_t dev, uint8_t address) {
    uint8_t data;
    i2c_start();

    I2C0->D = dev;
    I2C_WAIT
    I2C0->D = address;
    I2C_WAIT

    I2C_M_RSTART;
    I2C0->D = (dev | 0x01);
    I2C_WAIT
    I2C_REC;
    NACK;

    data = I2C0->D;
    I2C_WAIT
    I2C_M_STOP;
```

```
    data = I2C0->D;
    return data;
}

// To be used together to perform a multi-byte read
void i2c_read_setup(uint8_t device, uint8_t address) {
    i2c_start();
    I2C0->D = device;
    I2C_WAIT
    I2C0->D = address;
    I2C_WAIT
    I2C_M_RSTART;
    I2C0
    ->D = (device | 0x01);
    I2C_WAIT
    I2C_REC;
}

uint8_t i2c_repeated_read(uint8_t isLastRead) {
    uint8_t data;
    if (!isLastRead) {
        NACK;
    } else {
        ACK;
    }
    data = I2C0->D;
    I2C_WAIT
    if (!isLastRead) {
        I2C_M_STOP;
    }
    data = I2C0->D;
```

```
    return data;
}

void i2c_write_byte(uint8_t dev, uint8_t address, uint8_t data) {
    i2c_start();
    I2C0->D = dev;

    I2C_WAIT
    I2C0->D = address;

    I2C_WAIT
    I2C0->D = data;

    I2C_WAIT
    I2C_M_STOP;
}
```

CODE -2 :

```
#include <MKL25Z4.h>
#include "Accelerometer.c"

#define EXTERNAL_LED_PORT       PTB     // Port for the external LEDs
#define EXTERNAL_LED_PIN_RED    0x09    // Pin for the external red LED
#define EXTERNAL_LED_PIN_GREEN  0x0B    // Pin for the external green LED
#define EXTERNAL_LED_PIN_BLUE   0x0A   // Pin for the external blue LED
#define EXTERNAL_LED_PIN_GND    0x08    // Pin for the external GND connection

void update_external_led(uint8_t red, uint8_t green, uint8_t blue) {
```

```
    uint32_t led_state = 0;

    if (red > 0x10) {
        led_state |= EXTERNAL_LED_PIN_RED;    // Set external red LED pin if red is active
    }

    if (green > 0x10) {
        led_state |= EXTERNAL_LED_PIN_GREEN;  // Set external green LED pin if green is active
    }

    if (blue > 0x10) {
        led_state |= EXTERNAL_LED_PIN_BLUE;   // Set external blue LED pin if blue is active
    }

    EXTERNAL_LED_PORT->PSOR = EXTERNAL_LED_PIN_RED | EXTERNAL_LED_PIN_GREEN
|   EXTERNAL_LED_PIN_BLUE | EXTERNAL_LED_PIN_GND;  // Turn off all LEDs and GND connection

    EXTERNAL_LED_PORT->PCOR = led_state;  // Turn on LEDs based on the state
}


void LED_init(void) {

SIM->SCGC5 |= ((1 << SIM_SCGC5_PORTB_SHIFT) | (1 << SIM_SCGC5_PORTD_SHIFT));
        //enables clock gate control for Port B and Port D -Enable VDD power
SIM->SOPT1 |= SIM_SOPT1_USBVSTBY_MASK;
// This enables VDD (USB 3.3V) power during the standby mode
SIM->SOPT1 &= ~SIM_SOPT1_USBREGEN_MASK;
        // This disables the USB regulator enable feature by clearing the USBREGEN
PMC->REGSC = 0x10u;                                                                    //
configures the regulator status and control register (REGSC) of the Power Management
```

```
PORTB->PCR[18] = 0x100;
PTB->PDDR |= 0x40000;

PORTB->PCR[19] = 0x100;
PTB->PDDR |= 0x80000;

PORTD->PCR[1] = 0x100;
PTD->PDDR |= 0x2;
}

void delay_ms(uint32_t ms) {
volatile uint32_t i;
for (i = 0; i < ms * 10000; i++) {}
}

int main(void) {
   LED_init();
   i2c_init();

   i2c_write_byte(0x3A, 0x2A, 0x00);
  elay_ms(100);

   i2c_write_byte(0x3A, 0x2A, 0x01);              //turning on from standby mode to active mode
   delay_ms(100);

   while (1) {
      uint8_t accel_x = i2c_read_byte(0x3A, 0x01);
      if (accel_x == 0xFF) {}
      delay_ms(10);

      uint8_t accel_y = i2c_read_byte(0x3A, 0x03);
```

```
        if (accel_y == 0xFF) {}
        delay_ms(10);

        uint8_t accel_z = i2c_read_byte(0x3A, 0x05);
        if (accel_z == 0xFF) {}
        delay_ms(10);

        update_external_led(accel_x, accel_y, accel_z);
    }
}
```