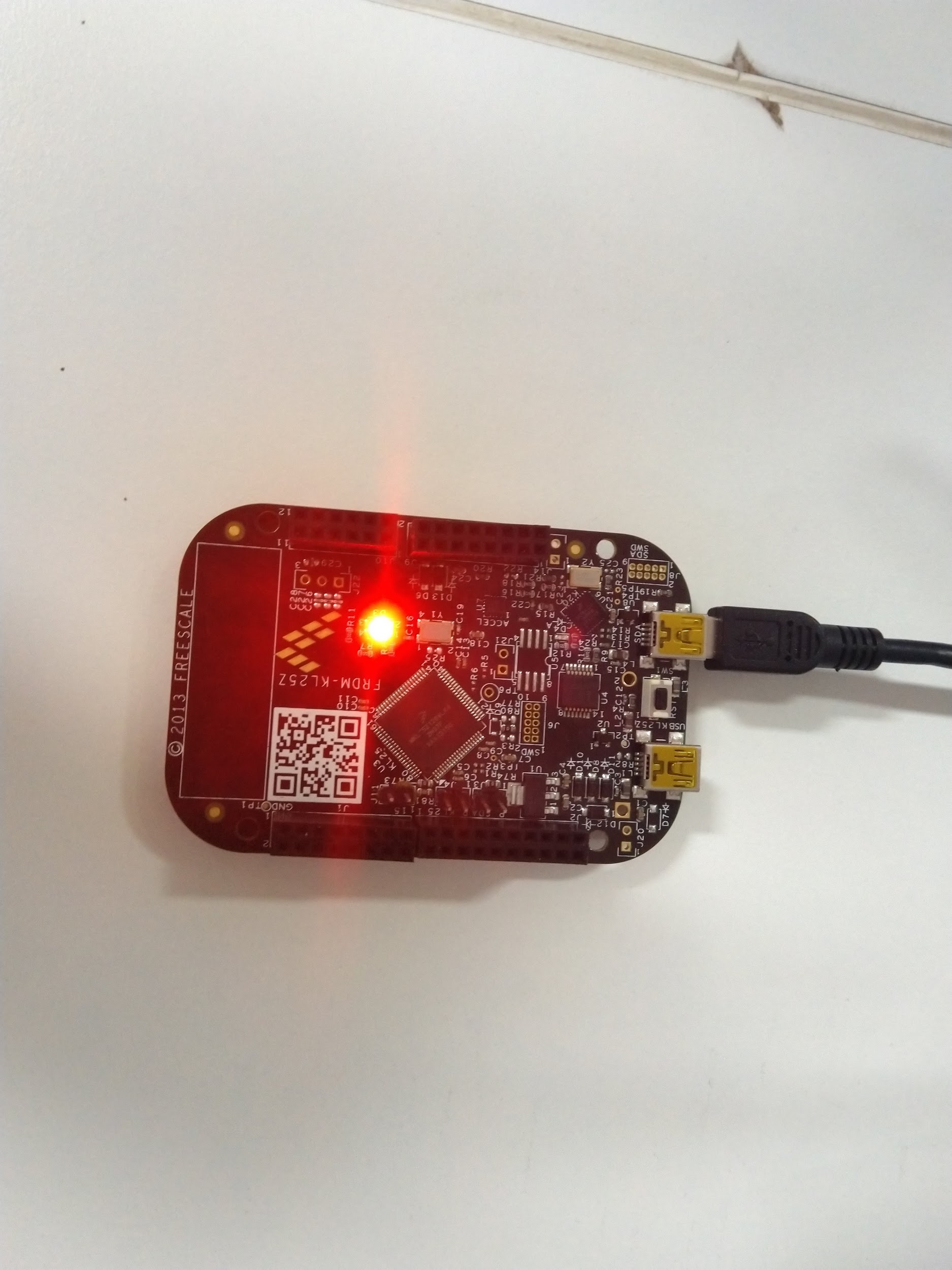
**Lab Report**

****

**Embedded Systems**

**Himanshu Rai**

**111601032 6th September, 2019**

**Objective**

To light up RGB lights on FRDM-KL25Z board, with LED colour determined by user input.

**Theory**

**UART Communication (Universal asynchronous receiver/transmitter)**

In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs.

We will use UART0 ports on the FRDM board to read user input based on which we can decide the colour with which we can light the LED. The following steps are required to configure the UART0 port so that we can use it to read the inputs.

* By default, UART modules are disabled to save power. So, we should first enable the clocks to the ports corresponding to UART0 to prevent hardware fault. The *system clocking gate control register 4 (SIM\_SCGC4)* gates the clocks to UART ports. We can enable the clock for UART0 by setting the 10th bit in *SIM\_SCGC4* control register.
* Next we need to choose from the different available clock sources. *SOPT2* contains the controls for selecting many of the module clock source options on this device. The clock for UART0 can be configured by modifying bit positions 26th and 27th.
  + 00 - Clock disabled
  + 01 - MCGFLLCLK clock or MCGPLLCLK/2 clock
  + 10 - OSCERCLK clock
  + 11 - MCGIRCLK clock

We have to choose the second option, so we will configure the bits to have 01 at the mentioned bit positions.

* Next we need to set up the *SBR* and *OSR* (over sampling ratio). These two are related by the following equation

SBR = clock frequence / (OSR \* baud rate)

For us, baud rate = 57600Hz, clock frequency = 22MHz, and we will have *OSR* of 16. So, SBR will be 24.

We can set *OSR* through *UART0\_C4* register using bits 0 to 4. For setting *SBR* we need to modify two registers - *UART0\_BDL* which corresponds to *SBR[7:0]* and *UART0\_BDH* whose first 5 bits corresponds to *SBR[12:8]* .

* Next we will activate *port A* by setting 9th bit in *SIM\_SCGC5* register and then configure *port A1* to be used as UART\_Tx by putting 010 at bit positions 10 to 8 in PCR for *port A1*.
* Finally activate the receiver of *UART0* by setting *UART0\_C2* register as 4.

We can check whether we have received some input by checking 5th bit in *UART0\_S1* register. If this is set it means the receive buffer is full and we can see the input in *UART0\_D* register.

**Lighting up the LEDs**

The following tables give the correspondence between the LEDs and port pins associated with them

|  |  |
| --- | --- |
| **LED** | **GPIO PIN** |
| Red | PortB18 |
| Green | PortB19 |
| Blue | PortD1 |

First we will have to follow the following procedure to configure the ports/pins corresponding to the LEDs, before using them

* By default, GPIO modules are disabled to save power. So, we should first enable the clocks to the ports corresponding to the LEDs to prevent hardware fault. The *system clocking gate control register 5 (SIM\_SCGC5)* gates the clocks to GPIO ports. We can enable the clock for these ports by setting the respective bits in *SIM\_SCGC5* control registers - which are bit 10 (for port B - red and green LEDs) and bit 12 (for port D - blue LED).
* Next configure the pins for GPIO . For this we must write 001 in the corresponding *program control registers (PCR)* at bit positions 10 to 8.
* Now set the pins as output pins by setting the corresponding pin position in the *port data direction register (PDDR)* of the corresponding ports.

After setting up the pins, we can turn the LEDs on and off by setting the corresponding bits in *port clear output register (PCOR)* and *port toggle output register (PTOR).*

**Procedure**

According to the steps mentioned above, it would suffice to write a function for activating *UART0 port* and three other functions for each LEDs - one to initialise the ports and the other two to switch the LEDs on and off.

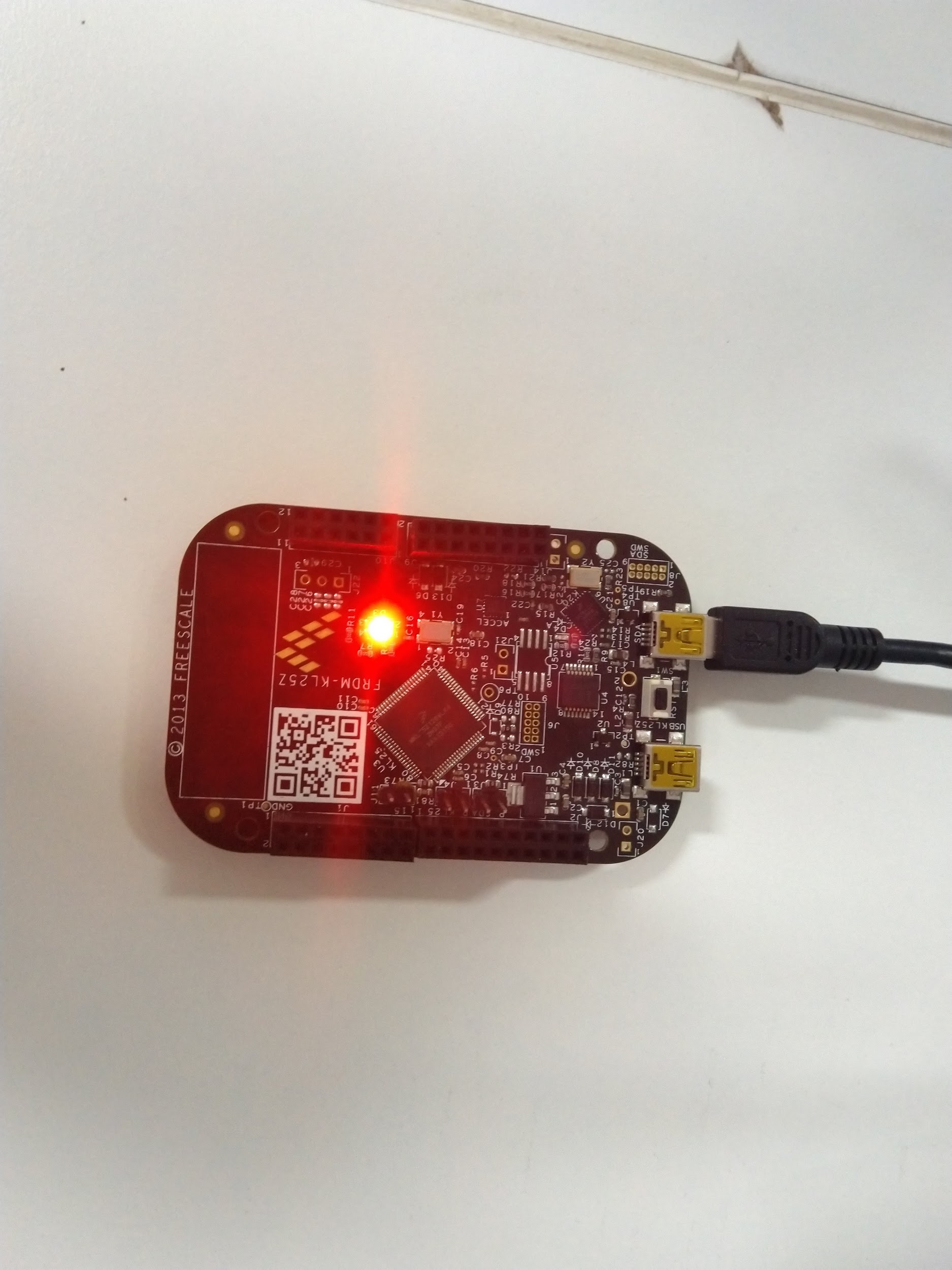
The initialisation functions for each LEDs and UART0 need to be called only once, at the beginning of the program. Also it is safe to call off function for each LED at the beginning.

Next in a loop we wait for user input and then light up the LED depending on it. We keep some feasible delay and then turn off the LED and then again wait for next input.

The source code is at the end of the report.

**Results**

The LEDs worked as desired depending upon the user input.



**Source Code**

#include<MKL25Z4.h>

void UART0\_init() {

// enabling UART0 by setting the corresponding bit in SCGC4 register

SIM->SCGC4 |= 1 << 10;

// enable the clock at UART0 from FLL or PLL/2 by putting 01 at bit

// positions 27 and 26 in SOPT2 registers

SIM->SOPT2 |= 1 << 26;

SIM->SOPT2 &= ~(1 << 27);

/\*

SBR = (clock frequency = 22000000) / (OSR \* (baud rate = 57600))

clock frequency and baud rate are fixed so we need only OSR or SBR

SBR is of 13 bits and we set its value using the following registers -

SBR[7:0] = BDL[7:0] and SBR[12:8] = BDH[4:0]

\*/

// disable all the operation modes in UART0

UART0->C2 = 0x00;

// SBR setting for baud rate of 57600 (and OSR = 16)

UART0->BDH = 0x00;

UART0->BDL = 0x18;

UART0->C4 = 0x0F; // for an OSR of 16

UART0->C1 = 0x00; // no parity

// activate the receiver of UART0

UART0->C2 = 0x04;

// activating port A

SIM->SCGC5 |= 1 << 9;

// configuring PTA1 as UART\_Tx by putting 010 at bit positions 10 to 8 in PCR for PORT A1

/\*

Note that we can also do this using alternative 4 for PORTE21

But here we are using alternative 1 for PORTA1 because it is connected to USB port

whereas PORTE20 is not

\*/

PORTA->PCR[1] &= ~(7 << 8);

PORTA->PCR[1] |= 1 << 9;

}

void delay(int n) {

while(n--);

}

void led\_red\_init() {

SIM->SCGC5 |= 1 << 10; // will activate the port B

// set the pin as gpio

PORTB->PCR[18] &= 0xFFFFF8FF; // set 8th, 9th and 10th bit of PCR[18] to 0

PORTB->PCR[18] |= 1 << 8; // set 8th bit of PCR[18] to 1

// set the port as output port

PTB->PDDR |= 1 << 18; // set the 18th bit of PDDR to 1 for output

}

void led\_red\_on() {

// clear the 18th bit of PDOR register

PTB->PCOR |= 1 << 18; // set the 18th bit of PCOR to 1

}

void led\_red\_off() {

// toggle the 18th bit of PDOR register

PTB->PTOR |= 1 << 18;

}

void led\_green\_init() {

SIM->SCGC5 |= 1 << 10; // will activate the port B

// set the pin as gpio

PORTB->PCR[19] &= 0xFFFFF8FF; // set 8th, 9th and 10th bit of PCR[19] to 0

PORTB->PCR[19] |= 1 << 8; // set 8th bit of PCR[19] to 1

// set the port as output port

PTB->PDDR |= 1 << 19; // set the 19th bit of PDDR to 1 for output

}

void led\_green\_on() {

// clear the 18th bit of PDOR register

PTB->PCOR |= 1 << 19; // set the 19th bit of PCOR to 1

}

void led\_green\_off() {

// toggle the 18th bit of PDOR register

PTB->PTOR |= 1 << 19;

}

void led\_blue\_init() {

SIM->SCGC5 |= 1 << 12; // will activate the port D

// set the pin as gpio

PORTD->PCR[1] &= 0xFFFFF8FF; // set 8th, 9th and 10th bit of PCR[1] to 0

PORTD->PCR[1] |= 1 << 8; // set 8th bit of PCR[1] to 1

// set the port as output port

PTD->PDDR |= 1 << 1; // set the 1th bit of PDDR to 1 for output

}

void led\_blue\_on() {

// clear the 18th bit of PDOR register

PTD->PCOR |= 1 << 1; // set the 1th bit of PCOR to 1

}

void led\_blue\_off() {

// toggle the 18th bit of PDOR register

PTD->PTOR |= 1 << 1;

}

int main() {

SystemCoreClockUpdate(); // updating the clock from PLL

UART0\_init();

led\_red\_init();

led\_green\_init();

led\_blue\_init();

led\_red\_off();

led\_green\_off();

led\_blue\_off();

while(1) {

while(!(UART0->S1 & (1 << 5))); // wait while transmit data register is empty

char ch = UART0->D;

if(ch == 'A') led\_red\_on(), delay(1000000), led\_red\_off();

else if(ch == 'B') led\_green\_on(), delay(1000000), led\_green\_off();

else led\_blue\_on(), delay(1000000), led\_blue\_off();

}

}