PROJECT REPORT

Module 1:

- **We have created a command structure which includes command the data length then actual data pointer and the checksum. We created data pointer so that we can define it on malloc and vary it's size dynamically according to the data received. We are using addition checksum which checks whether data is valid or not. So according to length received we can run loop and receive data.**

```
typedef struct{
Cmds command;
int8_t length;
int8_t *data;
int8_t checksum;
 }CI_Msg;
```

- **For command IDs we have defined various enums**

```
typedef enum{
GREEN_ON,
GREEN_OFF,
BLUE_ON,
BLUE_OFF,
RED_ON,
RED_OFF,
GREEN_BR,
BLUE_BR,
RED_BR,
SYSRESET,
EPOCH

} Cmds;
```

- We have developed a led_control lib which includes led drivers which will control ON/OFF of individual LEDs and also we have written drivers (firmware/bare metal code) for RTC to get timestamp

- Also if command is invalid we are logging "INVALID COMMAND" string and if checksum is wrong we are logging "INVALID CHECK".
- After Decoding commands we are executing those with either if-else ladder or function pointers.
- Using function pointers increases efficiency of the code.
- We are also doing system reset
- Lastly while receiving structures we are using circular buffer so that we can send multiple commands at a time and execute them sequentially
- So many commands packets would be received at the same time and will be stored in buffer and will be executed later one by one.

Code with function pointers:

```
#include"MKL25Z4.h"
#include"uart.h"
#include"circbuf.h"
#include"logger.h"
#include"rtc.h"
#include"led_control.h"
#define SIM_SOPT2_MCGFLLCLK_MASK 0x1000000u

buf_ptr *tx,*rx;
int8_t rec;
int8_t *src;
int8_t index=0;
int8_t count;
int8_t count1 = 4;
int8_t final_value;
int8_t i =0;
int8_t j =0;
int8_t k =0;
int8_t l =0;
int8_t m =0;
int8_t count =0;

CI_Msg *Command = &final_value;
```

```
void sysreset(CI_Msg *Command)
{
 NVIC_SystemReset();
}
int main(void)
{
 tx = malloc(sizeof(buf_ptr));
    tx->Buffer = malloc(sizeof(int8_t) * 100);
    init_buff(tx, 100);
    rx = malloc(sizeof(buf_ptr));
    rx->Buffer = malloc(sizeof(int8_t) * 100);
    init_buff(rx, 100);
    uart_configure();
    rtc_init();
    UART0_C2 |= UART0_C2_RIE_MASK; //Enable receiver interrupt
    typedef void (*func)(CI_Msg *Command);
    func fpointers[] = {red_on, red_off, blue_on ,blue_off , green_on , green_off
,green_brightness,blue_brightness,red_brightness,sysreset};
    while(1)
    {
    if(count > 0)
     {       m = 0;
           Buffer_remove(rx,(int8_t*)&Command->command);
              Buffer_remove(rx,&Command->length);
              k = Command->length;
              for(;k>0;k--)
              {
              Buffer_remove(rx,Command->data);
              Command->data++;
              l++;
              }
              Command->data = Command->data - l;
              l = 0;
              Buffer_remove(rx,&Command->checksum);
              k = Command->length;
              for(;k>0;k--)
```

```
        {
         m = m+*Command->data;
         Command->data++;
         l++;
        }
        Command->data = Command->data - l;
        l = 0;
        m = Command->command + Command->length + m;
        if(Command->checksum == m && Command->command != 10)
        {
        fpointers[Command->command](Command);
        if(Command->command >= 12)
        {
         int8_t arr4[] = " INVALID COMMAND ";
         src = arr4;
         log_string(src ,17 ,tx);
        }
        }
        else if (Command->command == 10)
{
        int8_t arr4[] = " TIME ";
        src = arr4;
        log_string(src ,6 ,tx);
        int8_t arr5[10];
        src = arr5;
        log_integer(src,RTC_TSR,16,tx);

}
        else
        {
         int8_t arr4[] = " INVALID CHECK ";
         src = arr4;
         log_string(src ,15 ,tx);
        }
 UART0_C2 |= UART0_C2_TIE_MASK;
 count =count -1;
```

```
    }
    }

}
void UART0_IRQHandler(void)
{
    __disable_irq();
    if (UART0_S1 & UART_S1_RDRF_MASK){
        rec = UART0_D;
        if(rec == 'e'){
            count++;
        }
        else{
        Buffer_add(rx,&rec);
        }
    }
    else if((UART0_S1 & UART_S1_TDRE_MASK))
    {
     if(tx->Count!=0)
     {
     Buffer_remove(tx,&rec);
     uart_send_byte(rec);
     }
     else
     {
     UART0_C2 &= ~UART0_C2_TIE_MASK;
     }
    }
    __enable_irq();
}
```

Code with nested if else:

```
#include"MKL25Z4.h"
#include"uart.h"
#include"circbuf.h"
```

```
#include"logger.h"
#include"rtc.h"
#include"led_control.h"
#define SIM_SOPT2_MCGFLLCLK_MASK 0x1000000u

buf_ptr *tx,*rx;
int8_t rec;
int8_t *src;
int8_t index=0;
int8_t count;
int8_t count1 = 4;
int8_t final_value;
int8_t i =0;
int8_t j =0;
int8_t k =0;
int8_t l =0;
int8_t m =0;
int8_t count =0;

CI_Msg *Command = &final_value;



int sysreset(void)
{
 NVIC_SystemReset();
}

int main(void)
{
 tx = malloc(sizeof(buf_ptr));
    tx->Buffer = malloc(sizeof(int8_t) * 100);
    init_buff(tx, 100);
    rx = malloc(sizeof(buf_ptr));
    rx->Buffer = malloc(sizeof(int8_t) * 100);
    init_buff(rx, 100);
    uart_configure();
    rtc_init();
    UART0_C2 |= UART0_C2_RIE_MASK; //Enable receiver interrupt
```

```
while(1)
{
if(count > 0)
 {       m = 0;
       Buffer_remove(rx,(int8_t*)&Command->command);
          Buffer_remove(rx,&Command->length);
          k = Command->length;
          for(;k>0;k--)
          {
          Buffer_remove(rx,Command->data);
          Command->data++;
          l++;
          }
          Command->data = Command->data - l;
          l = 0;
          Buffer_remove(rx,&Command->checksum);
          k = Command->length;
          for(;k>0;k--)
          {
           m = m+*Command->data;
           Command->data++;
           l++;
          }
          Command->data = Command->data - l;
          l = 0;
          m = Command->command + Command->length + m;

          if(Command->checksum == m)
          { if(Command->command == GREEN_ON )
          {
           green_on(Command);
          }
          else if(Command->command == GREEN_OFF)
          {
             green_off(Command);
          }
          else if(Command->command == BLUE_ON)
```

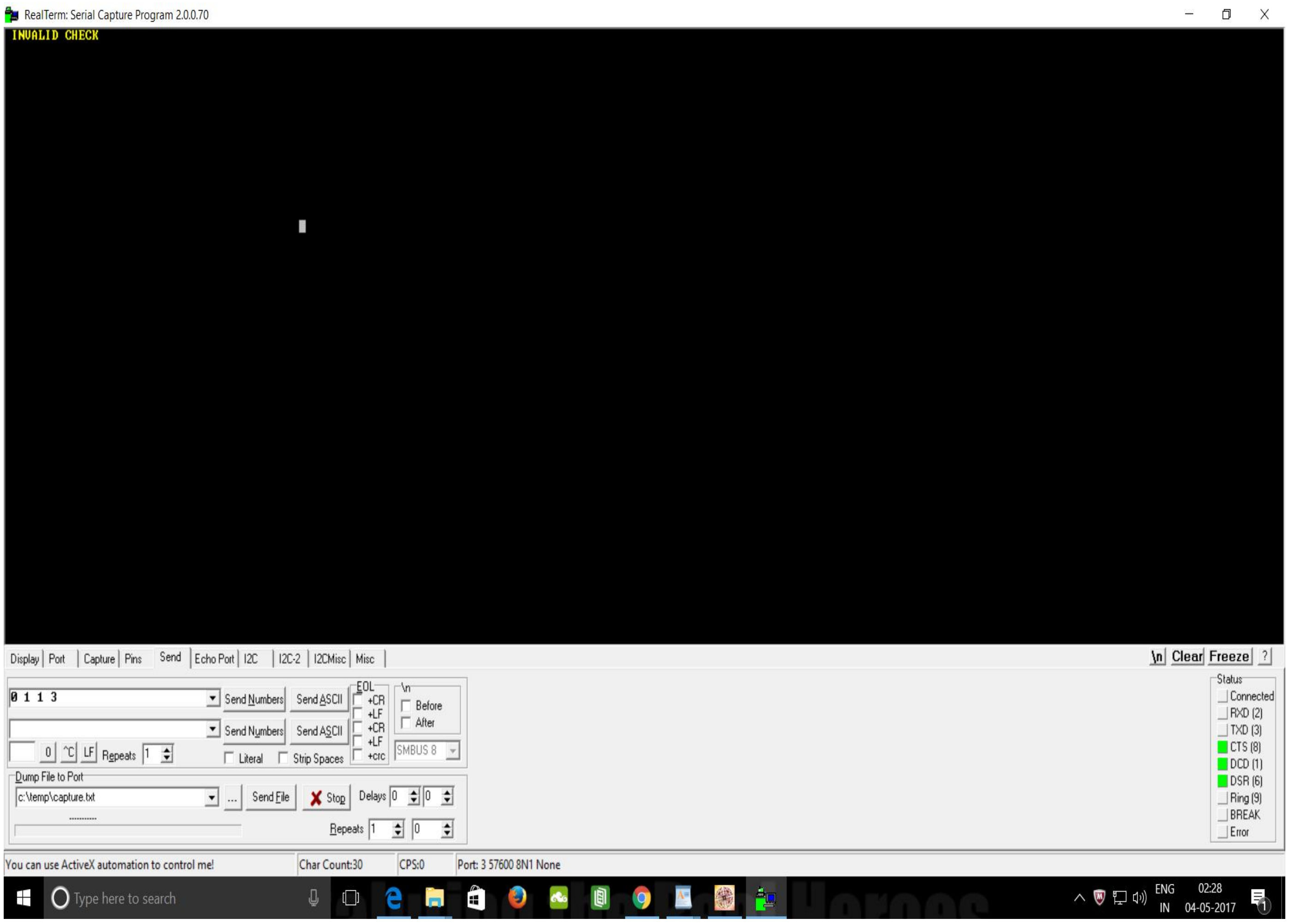
```
{
   blue_on(Command);
}
else if(Command->command == BLUE_OFF)
{
   blue_off(Command);
}
else if(Command->command == RED_ON)
{
   red_on(Command);
}
else if(Command->command == RED_OFF)
{
   red_off(Command);
}
else if(Command->command == GREEN_BR)
{
 green_brightness(Command);
}
else if(Command->command == BLUE_BR)
{
 blue_brightness(Command);
}
else if(Command->command == RED_BR)
{
 red_brightness(Command);
}
else if(Command->command == SYSRESET)
{
 sysreset();
}
else if(Command->command == EPOCH)
{   int8_t arr4[] = " TIME ";
 src = arr4;
 log_string(src ,6 ,tx);
 int8_t arr5[10];
```

```
              src = arr5;
              log_integer(src,RTC_TSR,16,tx);

             }
     else
             {
              int8_t arr4[] = " INVALID COMMAND ";
              src = arr4;
              log_string(src ,17 ,tx);
             }
             }
             else
             {
              int8_t arr4[] = " INVALID CHECK ";
              src = arr4;
              log_string(src ,15 ,tx);
             }
      UART0_C2 |= UART0_C2_TIE_MASK;
      count =count -1;


    }
    }

}

void UART0_IRQHandler(void)
{
   __disable_irq();
   if (UART0_S1 & UART_S1_RDRF_MASK){
      rec = UART0_D;
      if(rec == 'e'){
         count++;
      }
      else{
      Buffer_add(rx,&rec);
      }
   }
```

```
    else if((UART0_S1 & UART_S1_TDRE_MASK))
    {
     if(tx->Count!=0)
     {
     Buffer_remove(tx,&rec);
     uart_send_byte(rec);
     }
     else
     {
     UART0_C2 &= ~UART0_C2_TIE_MASK;
     }
    }
    __enable_irq();
}
```

Screenshots for Timestamp Invalid Command and Invalid Checksum

Invalid Checksum:



RealTerm: Serial Capture Program 2.0.0.70

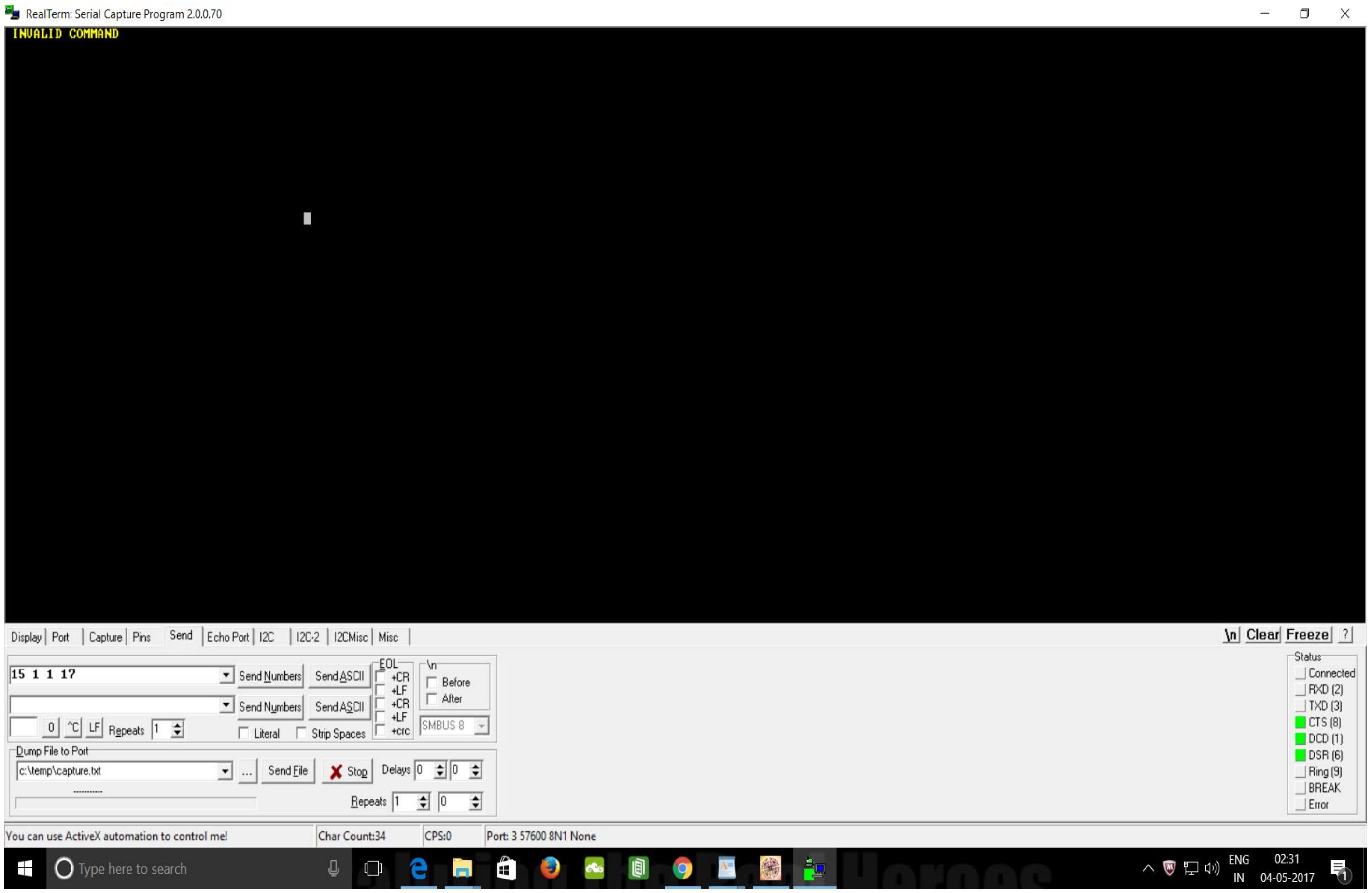
```
INVALID CHECK
```

Send: 0 1 1 3

Dump File to Port: c:\temp\capture.txt

You can use ActiveX automation to control me! | Char Count:30 | CPS:0 | Port: 3 57600 8N1 None

Invalid Command :



RealTerm: Serial Capture Program 2.0.0.70

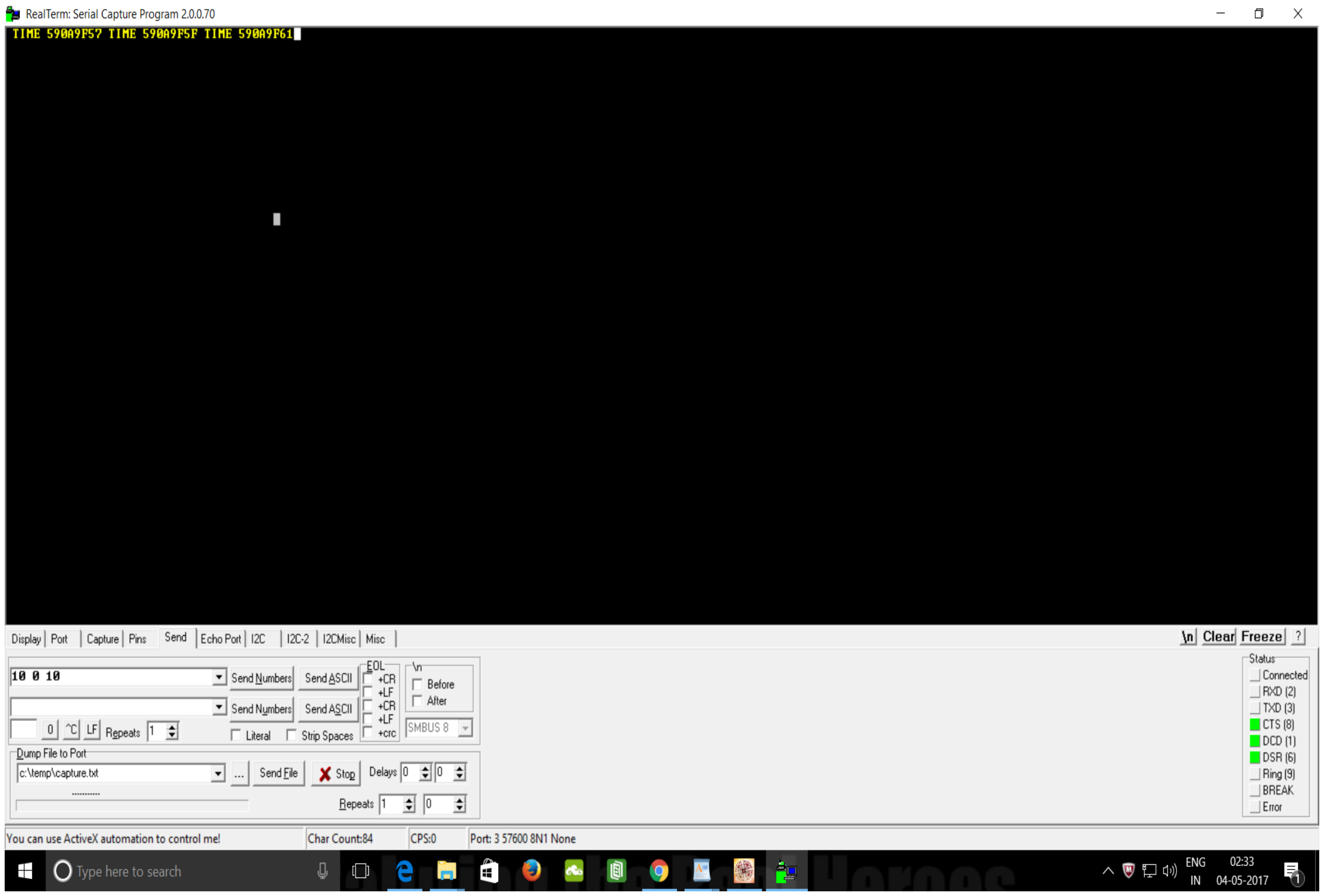
```
INVALID COMMAND
```

Send: 15 1 1 17

Dump File to Port: c:\temp\capture.txt

You can use ActiveX automation to control me! | Char Count:34 | CPS:0 | Port: 3 57600 8N1 None

TIMESTAMP and Command of Variable Length:



```
TIME 590A9F57 TIME 590A9F5F TIME 590A9F61
```

Send: 10 0 10

Repeats 1

Dump File to Port: c:\temp\capture.txt

Delays 0 0

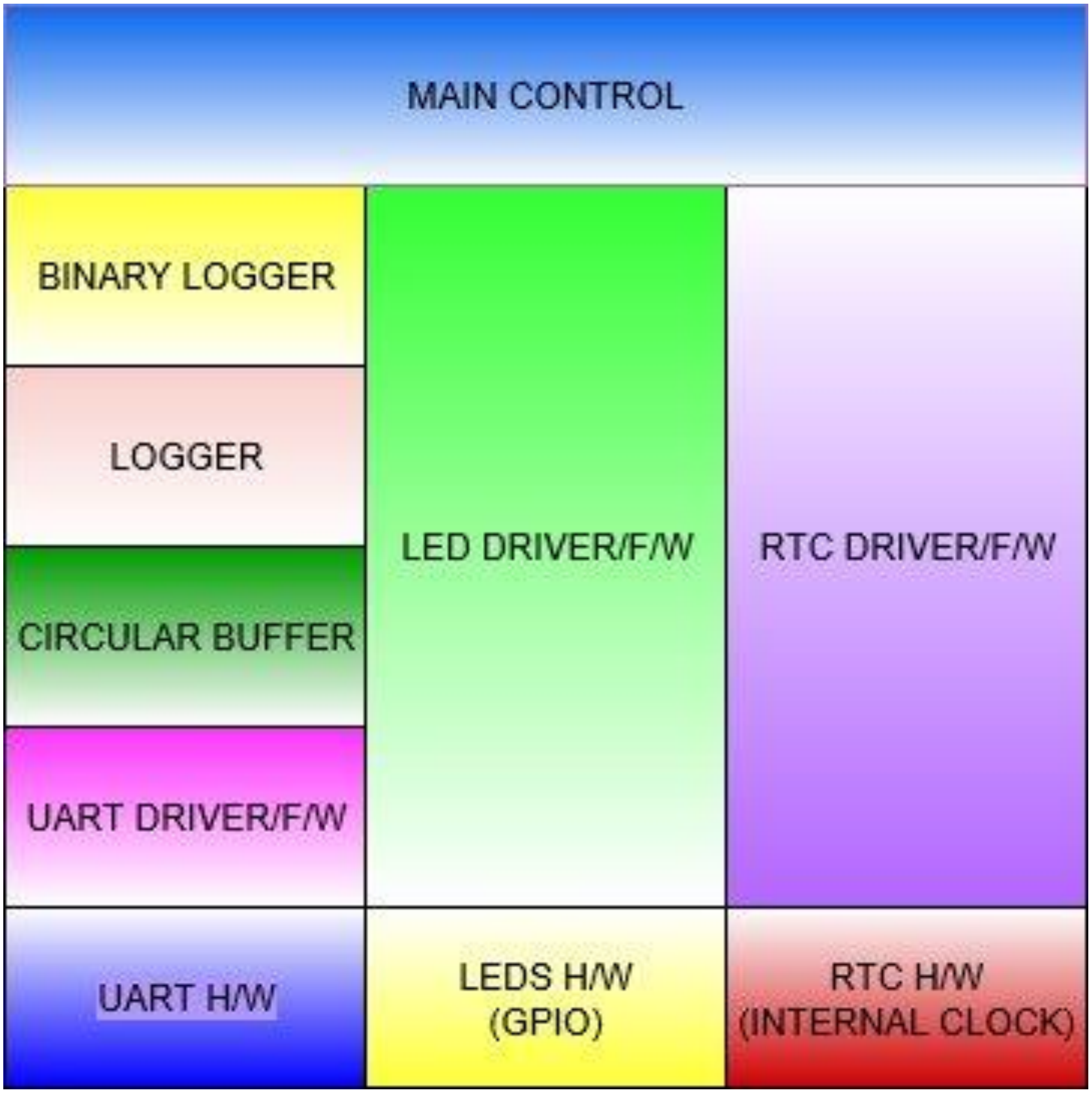
Repeats 1 0

Status: 🟩 CTS (8) 🟩 DCD (1) 🟩 DSR (6)

You can use ActiveX automation to control me! | Char Count:84 | CPS:0 | Port: 3 57600 8N1 None

02:33 04-05-2017

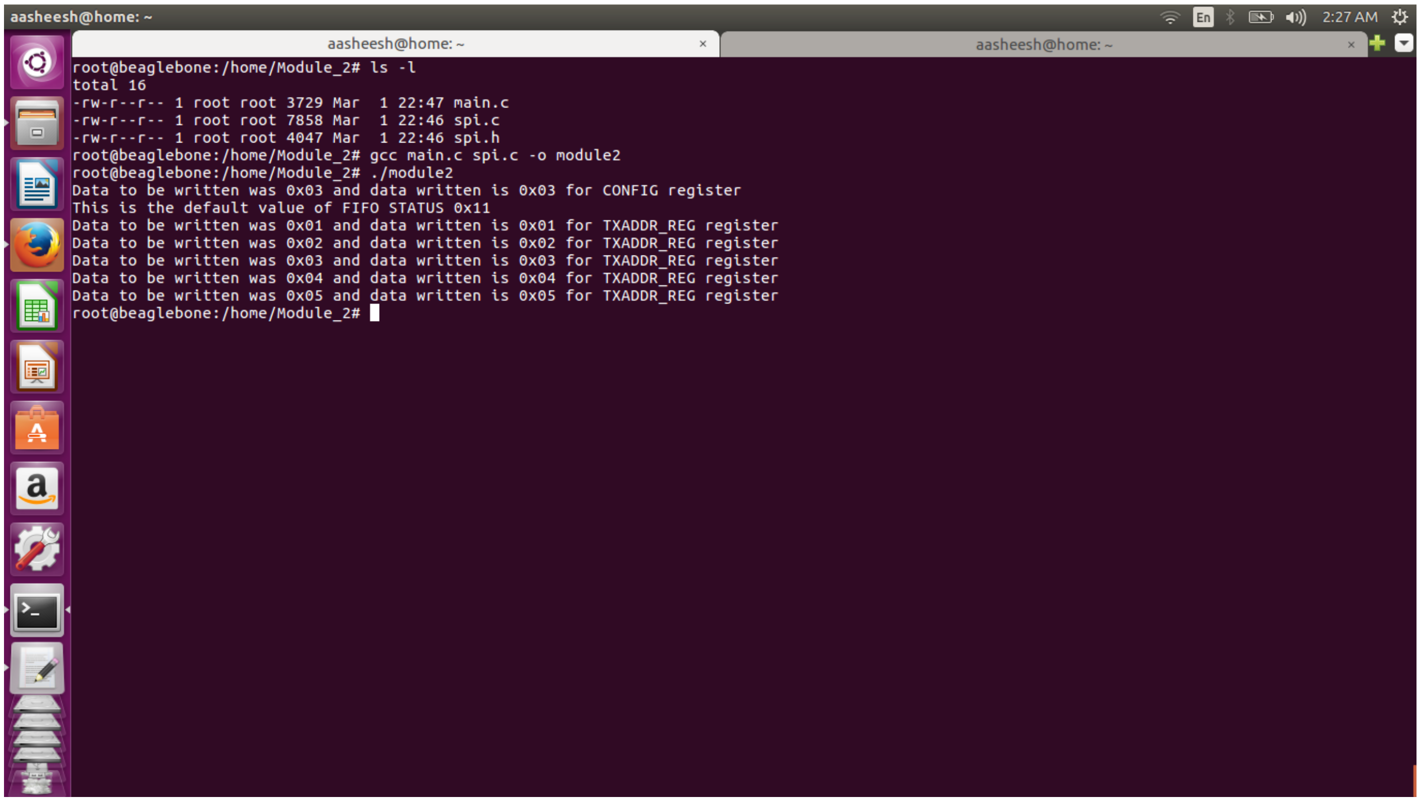
Software/Hardware Block Diagram :



- So here we have hardware layers at the bottom which represents the actual hardware
- Above that we have firmware layer which is bare-metal code. For LED and RTC it is extended till main control as we are using them directly in main without using any wrapper functions .
- In case of uart we are using logger and for logger binary logger as a wrapper.

Module 2:

- Here we are integrating the spi driver and communicating with nrf module through spi.
- We are creating a file descriptor to which we write config GPIO and spidev
- We are using unistd write(); read(); functions for spi read and write.
- A demonstration of read and write of CONFIG FIFO STATUS and TX ADDR has shown below.



```
root@beaglebone:/home/Module_2# ls -l
total 16
-rw-r--r-- 1 root root 3729 Mar  1 22:47 main.c
-rw-r--r-- 1 root root 7858 Mar  1 22:46 spi.c
-rw-r--r-- 1 root root 4047 Mar  1 22:46 spi.h
root@beaglebone:/home/Module_2# gcc main.c spi.c -o module2
root@beaglebone:/home/Module_2# ./module2
Data to be written was 0x03 and data written is 0x03 for CONFIG register
This is the default value of FIFO STATUS 0x11
Data to be written was 0x01 and data written is 0x01 for TXADDR_REG register
Data to be written was 0x02 and data written is 0x02 for TXADDR_REG register
Data to be written was 0x03 and data written is 0x03 for TXADDR_REG register
Data to be written was 0x04 and data written is 0x04 for TXADDR_REG register
Data to be written was 0x05 and data written is 0x05 for TXADDR_REG register
root@beaglebone:/home/Module_2# 
```