Team - Project 1

Project Leader

Alexander Fosdick	**Nov 6, 2017**
NAME	**DATE**

Project team member

Vihanga Bare	**Nov 6, 2017**
NAME	**DATE**

Project team member

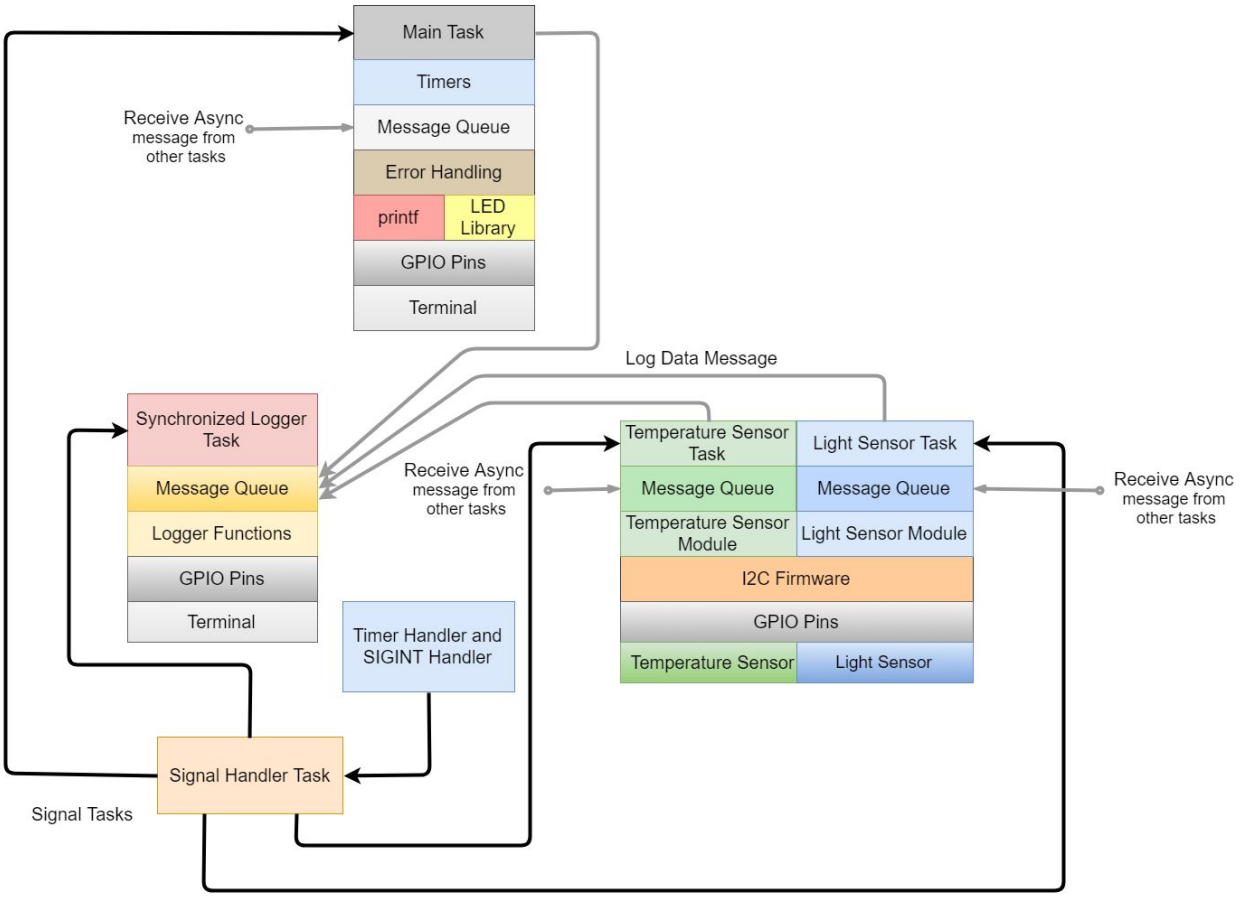
Virag Gada	**Nov 6, 2017**
NAME	**DATE**

Software Architecture:

Components:

1. **BeagleBone Green**
2. **Temperature Sensor (I2C module) - TMP 106**
3. **Light Sensor(I2C module) - APDS 9301**
4. **Linux OS**

Software Architecture Diagram:



Processes/Threads:

1. **Main Task -**
    - **Create child tasks namely TempThread, LighThread, LoggerThread, SigHand Thread.**
    - **Make sure above child tasks are alive and running.**
    - **Log errors and indicate errors through USR LED.**
    - **Create Timer to send periodic updates from other tasks to main and from periodic data logging from tasks to logger thread.**
    - **Log a Thread is dead message in case any thread is killed.**

2. **TempThread - Interact with Temperature sensor TMP 102 using I2C library for beaglebone and perform the following actions-**
    - **Configure command to read/write to the registers**
    - **Write to the Pointer register**
    - **Read/Write to the config register**
    - **Read the sensor temperature data register**
    - **Configure the sensor to go in and out of shutdown**
    - **Configure the sensor resolution**
    - **Provide the temperature values in °C, °K and °F**
    - **Use timer to periodically gather temperature data**
    - **Send heartbeat message to main task.**

3. **Light Thread - Interact with Light sensor APDS 9301 using I2C library for beaglebone and perform the following actions -**
    - **Configure a command message to read/write all registers**
    - **Read/Write the Control Register**
    - **Configure the Integration time in the Timing register**
    - **Enable and disable the Interrupt Control Register**
    - **Read the Identification Register**
    - **Read Sensor LUX data using the ADC registers**
    - **Report current light state - dark or light based on LUX value**
    - **Log change in state**
    - **Use timer to periodically gather light data**

- **Send heartbeat message to main task.**

4. **<u>Logger Thread</u> - This task receives messages from different threads by using ms_receive() logs the messages into a log file.**

5. **<u>Sighand Thread</u> - A common signal handler which waits on a signal from different events throughout the program like, Timer Expire Event, Asynchronous request event, and Signal Interrupt event. This signal handler, sets value for an atomic flag, sends out a condition broadcast to all the threads who are waiting on these events. On the thread end, the thread waits for a condition signal from this signal handler, and decides which event action to honour based checking the value of atomic flag set.**
6. **Timer struct itimerval used to create an interval timer.**
7. **Threads communicate with each other using the inter process communication mechanism called message queues. We are using POSIX message queue API, ms_send() and ms_receive to communicate messages between threads. Each message is of type logger_struct, we has details like, payload, requestID (what kind of message), sourceID(who sent the message) etc.**
8. **We have used mutex locks to synchronise this communication.**
9. **Each task has it own queue that it uses to send as well as receive messages from other threads/task.**

<u>Overview of Structures used-</u>

```
typedef enum i2c_states
{
  SUCCESS,
  ERROR_READ,
  ERROR_WRITE,
  ERROR_OPEN,
  ERROR_ADDRESS,
```

```
  ERROR_VALUE,
  NULL_POINTER,
}i2c_state;

typedef enum loglevel
{
        INFO,
        WARNING,
        ALERT,
        HEART_BEAT,
        INITIALIZATION
}LogLevel;

typedef enum{
   GET_TEMP_C,
   GET_TEMP_K,
   GET_TEMP_F,
   GET_LUX,
   GET_LIGHT_STATE,
   LOG_DATA,
   HEARTBEAT,
   DECIDE,
   SYSTEM_SHUTDOWN
}reqCmds;

typedef enum{
  MAIN_TASK,
  TEMP_TASK,
  LIGHT_TASK,
  LOGGER_TASK
}Sources;
```

```
typedef struct logger
{
        Sources sourceId;
        reqCmds requestID;
        LogLevel level;
        time_t timestamp;
        char payload[100];
}LogMsg;
```

Routines highlighted -

```
i2c_state setupI2CDevice (uint32_t * file, char * bus , uint8_t devAddr);
i2c_state writeI2CByte (uint32_t * file, uint8_t * data);
i2c_state readI2CByte (uint32_t * file, uint8_t * data);
i2c_state writeI2CWord (uint32_t * file, uint8_t * data);
i2c_state readI2CWord (uint32_t * file, uint8_t * data);
i2c_state readI2CDWord (uint32_t * file, uint8_t * data);

void *TempThread(void *);
void *LightThread(void *);
void *LoggerThread(void *);
void *SighandThread(void *);
void create_interval_timer(float timerval);
void create_timer(float timer_val, timer_t *timer_id);
void initialize_queue(char * qName, mqd_t *msgHandle);
void sighandler_sigint(int signum);
void sighandler_sigusr1(int signum);

gpioState pinSet(char * devAddr);
gpioState pinReset(char * devAddr);

mq_send (queue,(const char*)&loggerstruct, sizeof(LogMsg), 1)
mq_receive (queue,(const char*)&loggerstruct, sizeof(LogMsg), 1)
```