**Advanced Embedded Software Development (AESD) – ECEN 5013 – Project 1**

**Author -Souvik De, Devansh Mittal**

**Multi-threaded BeagleBone Linux application having sensor interface, logger and sockets.**

This project is a multi-threaded C application which interacts with a temperature and light sensor simultaneously and continuously logs the values in a log file.

The main task spawns 4 threads:

* Temperature sensor thread which is responsible for interacting with the TMP102 temperature sensor over MT-safe I2C
* Light sensor thread which is responsible for interacting with the APDs9301 light sensor over MT-safe I2C
* Logger thread which is responsible for logging sensor values, errors and sensor status.
* Remote thread which runs a TCP server and accepts connections from clients with various requests like temperature value and lux value

Sanity Checks:

* Start up tests for each sensor
* Regular monitoring and logging of status of each thread (heartbeat check).
* Unit Tests.

If any of the 2 sensors fail or are stuck for more than a timeout, the main task gracefully closes that thread. If the logger thread fails, the main task closes all the threads and gracefully exits the program. Any failure is indicated by an LED on the BeagleBone.

Functions and their descriptions:

**i2c.h**

init\_i2c: This function initializes the I2C bus

write\_i2c: This function writes 8-bit data to the I2C Bus

write\_i2c16: This function writes 16-bit data to the I2C Bus

read\_i2c16: This function reads 16-bit data from the I2C Bus

read\_i2c8: This function reads 8-bit data from the I2C Bus

close\_i2c: This function concludes the I2C Communication.

write\_i2c16\_config: This function writes 24-bit data to the I2C Bus to write to the configuration register of the temperature sensor.

**led.h**

led: This function switches the LED on and off

**lightSensor.h**

startup\_test: This function checks the identification register as part of a startup test and confirms if the light sensor is working properly.

power\_up: This function powers up the light sensor.

read\_control\_register: This function reads control register of the light sensor.

set\_timing\_register: This function is used to set the integration time and gain of light sensor.

read\_timing\_register: This function reads timing register of the light sensor.

enable\_interrupt: This function enables interrupts in the interrupt control register of the light sensor.

disable\_interrupt: This function disable interrupts of the light sensor.

set\_interrupt\_threshold: This function sets the high and low interrupt threshold of the light sensor.

read\_interrupt\_threshold: This function reads the high and low interrupt threshold of the light sensor.

read\_visible\_light: This function reads channel 0 of light sensor.

read\_IR\_light: This function reads channel 1 of light sensor.

cal\_lumen: This function calculates actual luminosity based on channel 0 and channel 1 lux values of light sensor.

change: This function checks if there is any change in light state from the previous state.

state: This function determines current light state based on threshold value.

read\_LightSensor: This function calls functions to read channel 0 and 1, calculate lumen, determine current light state, and if there was a change from previous state and stores it in the allocated structure.

**logger.h**

write\_log: This function logs events, errors, and heartbeat notifications generated by the program

**message.h**

init\_MessageQueues: This function initializes all the message queues

dest\_MessageQueues: This function destroys (closes and unlinks) all the message queues

open\_MessageQueue: This function opens a message queue identified by its parameters

send\_Message: This function is used to send messages through a queue identified by the first parameter

recv\_Message: This function is used to receive messages through a queue identified by the first parameter

CloseUnlinkQueue: This function closes and unlinks a message queue identified by the first parameter

**remoteTask.h**

init\_socket: This function initializes the socket at the server side

send\_data: This function sends data through a socket from server to client

read\_data: This function reads data sent from client to server via a socket

**temperature.h**

read\_temperature: This function reads the temperature register of the temperature sensor

cal\_temp: This function calculates actual temperature in celsius, fahrenheit, and kelvin based on digital temperature values read from the temperature register

read\_Tlow: This function reads the lower temperature limit from the tlow register of the temperature sensor.

read\_Thigh: This function reads the higher temperature limit from the thigh register of the temperature sensor

read\_configuration\_reg: This function reads the configuration register of the temperature sensor

set\_shutdown: This function sets the shutdown mode in the configuration register of the temperature register

disable\_shutdown: This function disables the shutdown mode in the configuration register of the temperature register

read\_fault: This function reads the fault bits from the configuration register of the temperature register

read\_em: This function reads the extended mode bit in the configuration register of the temperature register.

write\_em: This function sets/disables the extended mode in the configuration register of the temperature register

read\_conversion\_rate: This function reads the conversion rate bits in the configuration register of the temperature register

set\_conversion\_rate: This function sets the conversion rate bits in the configuration register of the temperature register

write\_fault: This function sets the fault bits in the configuration register of the temperature register

**clientprocess.h**

init\_socket: This function initializes the socket at the client side

send\_data: This function sends data through a socket from client to server

read\_data: This function reads data sent from server to client via a socket

**main.c**

kill\_signal\_handler: Signal handler for signal SIGINT aka Ctrl-C

getSensorData: Common call back function for temperature and light timer

timer\_init: This function initializes a unique timer on demand for every requesting thread

temperature: Temperature thread callback function

light: Light thread callback function

logger: Logger thread callback function

remote: Remote task thread callback function

check\_heartbeat: This function registers heart beats from different threads and sends relevant data to the logger.

setup\_signalhandler: This function initialize a signal handler to kill any particular thread or the process itself.

Unit Tests

**Light Sensor Task**

Start-up Test: Read ID register to verify light sensor is functional

Power-up Test: Read Control register to verify that light sensor has powered up

Timing Register Test: Set random integration and gain values and check if it is writing to the register by reading it.

Lumen Calculation Test: With random channel 0 and channel 1 lux values, the lumen calculation formula is tested to see if we are getting correct luminosity.

**Temperature Sensor Task**

Temperature Calculation Test: With random digital temperature values, it is tested whether the correct temperature values in Celsius, Fahrenheit and Kelvin are obtained.

Tlow Register Test: The Tlow register is read to test if the default value of 75 Celsius is obtained.

Thigh Register Test: The Thigh register is read to test if the default value of 80 Celsius is obtained.

Configuration Register Test: The configuration register is read to test if the default value of 0x60A0 is obtained.

**Logger Task**

File Creation Test: The function which creates the file is called to test if the file is created without any errors.

Start-up Tests

Light Sensor Task: The ID register is read and then checked if it correctly equals 0x50.

Temperature Sensor Task: The configuration register is read and checked if it correctly equals the default value of 0x60A0.

Logger Task: The start-up test constitutes the logger thread spawning and a file being created.