PROJECT 1 REPORT

-BY MONISH NENE

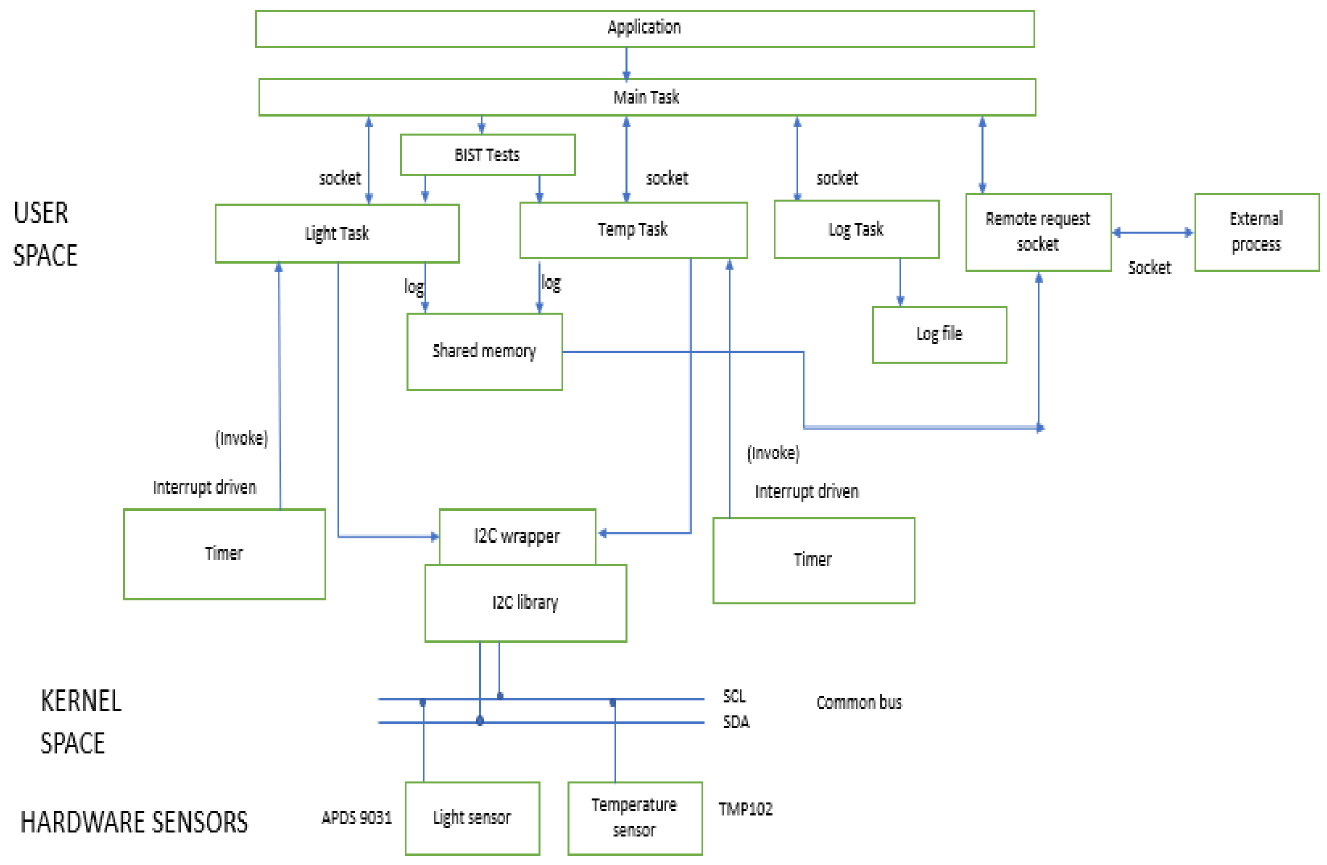
AND

SANIKA DONGRE

INDEX

Description	Page Number
Software Block Diagram and Architecture	**3**
Project Description	**4**
Function and Threads Description	**5**
Testing – Unit and BIST	**6**
API (code functions)	**7-8**

System and Software Architecture Diagram:



Documentation:

<u>**Project Description**</u>:

The project involves interfacing two off-board sensors – APDS 9301 light and TMP102 temperature sensors that are connected to the same I2C bus. The data from the sensors is logged on to a common log file. The tasks are connected to the logger by sockets and the logger uses shared memory to store the light and temperature data. The main task (project.c) uses asynchronous thread strategy using posix threads and the main task and creates four child threads:

1) Light thread
2) Temperature thread
3) Logger thread
4) Socket thread

Tasks and their Responsibilities:

1) Main Task: (project.elf):

   It creates the above mentioned four child threads – light, temperature, logger and socket. It creates logger and socket tasks in a loop and temperature and the light register tasks are run every instance and exited. The heartbeat feature is added to perform request/response notifications for checking if the tasks are alive. The main task (project.c) monitors if all the tasks are alive or not and the corresponding message is then logged into log file.

   The main responsibility of the main task is to exit gracefully. A condition flag is used which can be switched and then the threads are made to exit thus freeing the resources. The unlinking of semaphores is done while exiting.

   The BBG USR LED task is a part of the main task that uses the GPIO library to turn on/off the LEDS on the Beagle bone to indicate error message or alert condition. If the temperature threshold is above 15 degrees C then as an alert condition BBG LED is turned on.

2) Light sensor task: (light_read.c)

   The light thread continuously checks the lux value output of the light sensor connected to the Beagle bone via I2C interface. This file contains the get luminosity function which returns the luminosity value.

The registers – control, interrupt control, interrupt threshold, id register, power register, timing register (setting gain and other timing register tests) , checking the power off condition by sending the value 0 to the control reg (80 address) and data registers are read and write and that code is present in the file – light_task_registers.c

3) Temperature task: (temperature_read.c):

   The temperature thread continuously checks the temperature value output of the temperature sensor connected to the Beagle bone via I2C interface. This file contains the get_temperature() function that returns the temperature value in Celsius which is then converted into Kelvin and Fahrenheit and logged into the logger file.

   The code to write and read the registers – pointer, config, temp low, temp high, setting shut down mode and Interrupt mode is present in the file – temp_task_registers.c

4) Socket server task:
   The thread is used for making a remote server that is on continuously and that can be used for remote requesting the information and data from the sensors. It checks if the data is being received or not (sensors not connected or data is faulty). The socket server task is a TCP server. It is remotely connected by using the IP address (10.0.0.152).

   The server is being blocked on connection waiting for the client to get connected.
   The client blocks on the connection state and forking is done to keep the process running in background.

5) Logging Task: The thread is used for logging data to a txt file. It has the timestamp, log level –
   Info or error and the source ID and the message corresponding to the log. The data from light and temperature sensors is being logged in the log file continuously.

A backup of log files are being created after checking recursively if the file exists previously or not and if it doesn't exist then it renames the latest logfile and creates a backup of the previous file.

Testing:

- BIST test: The Built in Self tests:
  - For Light Sensor:
    1) I2c file write is checked using i2c_file function.
       If that fails it indicates that there is no I/O i.e. the light sensor is not connected.
       If it is successful it indicates that the light sensor is connected and initialized.
    2) The I2C read/write has been tested by writing to the timing register to check that the I2C works
    3) The threads are up and functioning is checked
    4) And the BIST are completed and this information is logged to the log file that the start up tests are completed and the normal processing can begin.
  - For Temperature sensor:
    1) I2c file write is checked using i2c_file function.
       If that fails it indicates that there is no I/O i.e. the light sensor is not connected.
       If it is successful it indicates that the light sensor is connected and initialized.
    2) The I2C read/write has been tested by writing to the configuration register -This indicates that the I2C is working and the BIST test for I2C works.
    3) The threads are up and functioning is checked
    4) And the BIST are completed and this information is logged to the log file that the start up tests are completed and the normal processing can begin.

- Unit test:
    1) These Tests are performed on individual codes to check functionality before integration.
    2) Unit test on light sensor to check that all its registers can be read and written. An enum that returns success or failure based on the
       Data that can be written to read from registers is used to indicate if the unit tests failed or passed.
    3) Unit test on temperature sensor to check that all its registers can be read and written. An enum that returns success or failure based on the
       Data that can be written to read from registers is used to indicate if the unit tests failed or passed.
    4) Unit test for logger
    5) Unit test for socket to check if a dummy value of temperature can be sent to the client from server.

<u>API functions</u>:

Functions common for light and temperature sensor:

i2c_write(): It will take the file descriptor and the address of the register as the parameter and write to the register of the temperature and light sensor respectively.

- i2c_read(): It will take the file descriptor and the address of the register as a parameter and read the respective sensor.
- i2c_file(): to open the file in (/dev/i2c-2) folder and to perform ioctl taking into account the respective sensor's slave address.

Temperature task:

- In temperature_run(): run temperature measurement function
- get_temperature(): Reads data from i2c temperature and returns output in Celsius
- temperature init: to initialize the temp measurement
- temperature_read(): Read data from temperature sensor and log it.
- In temptest.c for unit test:
  register_read(): used for reading a register. This API is used for reading the register
  register_write(): used for writing a register
- get_temperature(): get the latest temperature value and convert into required temperature units.

Light task:

- light_init(): To initialize semaphores and shared memory
- get_luminosity(): get the latest luminosity/lux value
- day_night() to check if it is day time or night time
- light_read(): This function is used to read luminosity from sensor and log it

Error handling (done via the main task):

- led_on(): To turn on the leds
- led_off(): To turn off the leds
- led_toggle(): to toggle the leds

- main(): To toggle the leds in a

Remote request task:

- server.c : handle request from client

Log synchronization task:

- logger_init(): initialize resources required for logger
- logger(): o continuously run and log data
- log_creator(): To log the data to the log file. It takes the Log level : error, info as the first parameter and the string to be printed as the second parameter.

Main task:

- temperature_run():To check if the tasks/processes are alive and return error if any process/thread has stopped.
- Light_run(): to run light measurement function
- Logger run(): run logger functions
- Server_run(): run server functions
- Logfile_Setup(): backup of old file and create a new one
- Temperature_thread(): create and join temp thread
- Light_thread(): create and join light thread
- Logger_thread(): create and join logger thread
- Server_thread(): create and join server thread
- System_end(): exit smoothly when signint is sent to main process
- Heartbeat(): check all threads running properly and alive
- Join threads(): to join threads
- Timer_init(): to initialize the timer
- Kill_server(): to kill thread with usr2
- Kill_logger(): to kill thread with usr1
- System_init(): to initialize all resources of the project

User and Kernel Space:

- The main task and BIST tests are included in the USER SPACE.
- The I2C wrapper are used for provide access to the I2C bus exclusively for two different sensor tasks and bbgled are included in the KERNEL SPACE.