Advanced Embedded System Development

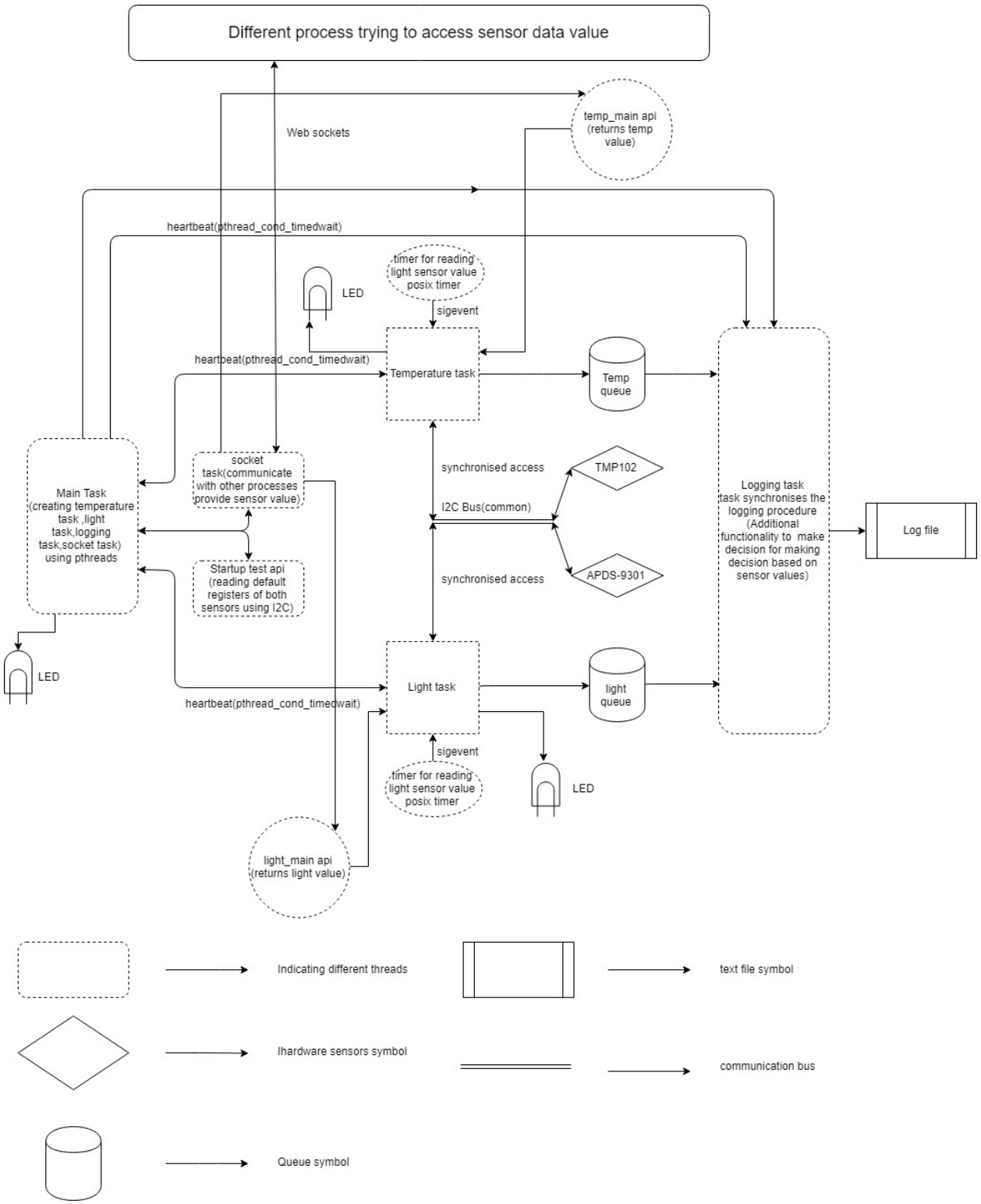
Project 1

MARCH 31, 2019

Yasir Aslam Shah Harsimransingh Bindra

Contents

Updated Software Architecture Diagram



Task Description

Main Task:

This whole project is divided into a parent task and various other children tasks. The main objective of the main task is to implement an asynchronous threading strategy, creating different children tasks for various functionality of the project such as **temperature sensor, light sensor, synchronized logger** and **remote socket request**. The main task is also responsible to ensure that all the tasks are running and are alive at certain regular intervals using **Heartbeat** functionality implementing a request – response mechanism. The main task should also take care of any abrupt halt request by the user and should gracefully exit the program. This main task will also log error information to the console user and indicate any error condition using LEDs available on the BBG (Beagle Bone Green).

Children Tasks:

- **Temperature Sensor Task**
- **Light Sensor Task**
- **Synchronized Logger Task**
- **Remote Request Socket Task**

Temperature Sensor Task:

In this task, we are using **TMP102** Temperature sensor [1] that works with I2C communication providing a resolution of 12 Bits with a wide range of operation ranging from **-40ºC to 125ºC**.The temperature sensor task interfaces with the TMP102 sensor through an I2C function present in the kernel space. The main function of this task includes **writing to the pointer register**, and **configuring the sensor for various modes, sensor resolution and conversion rate**. The temperature data read from the sensor is manipulated to accommodate various formats like **Celsius, Fahrenheit or Kelvin**. Additionally, this task will wake up the sensor after a given timeout, gathering a fresh lot of sensors values.

This task contains other functions too, to configure, read and write various registers that are individually tested in unit test modules.

Light Sensor Task:

This task interfaces with an external light sensor **APDS-9301** using I2C interface. This task again makes use of I2C function from the kernel space to **read and write to the command register** and **configure the sensor for 8- and 16-bit modes**. The output values can be utilized by implementing the conversion formula given in the datasheet **[2]**. This task is also responsible for determining the state as "**light**" or "**dark**" based on the log values from the sensor.

This task contains other functions too, to configure, read and write various registers that are individually tested in unit test modules.

Synchronized Logger Task:

This child task is responsible for accepting various logs from various on-board sources. A synchronization mechanism is implemented for protection from multiple log sources. The logger writes data to the log file, its file name and path. Each log entry must contain the following information as: **Timestamp, log level, logger ID and Log message**. Log entry will include multiple data type entries such as char string, integer or float. The log will be generated at specific intervals from all tasks for different events such as **initialization, data conversion, system failures etc**. This task will also contain an extra functionality in which an alert is generated every time the sensor values cross a certain threshold value for both the temperature as well as light sensor.

Remote Request Socket Task:

This task's responsibility is to accept a socket request from another process/task or the network and make API calls to sensor tasks for the data. The socket allows external programs to make request for the sensor data. This tasks' main objective is to test the functionality of the application by **requesting/injecting API commands** into the internal messaging structure.

Main API calls

Initial_test() - this function calls the temperature test and light test functions to check if there are functional before moving forward with creating the threads.

logging_thread() – The light thread created is linked to this function. This function is used to synchronize the logging process to the log file.

logging_function() – This function is used to log data to a log file specified as one of the parameters.

timer_setup() – It sets up the heartbeat timer parameters based on parameter input. It return the timespec structure

light_function() – The light pthread created is linked to this function. The light message queue communicating with the logger task is created and the timer which periodically logs the light values is enabled in this function. Then the function waits till the exit flag is set to join with the main thread

Light_main() – This function sets up the I2C bus to read from light sensor. Then it reads the instantaneous light value from the sensor and returns it to the calling function using a passed parameter. The function returns the exit status based on success or failure

Light_handler() – This function is called periodically due to the light timer event. This function reads the latest light value and populates the message structure with the light value or ALERT message and sends it to the logger task.

temperature_function() – The temperature pthread created is linked to this function. The temp message queue communicating with the logger task is created and the timer which periodically logs the temperature values is enabled in this function. Then the function waits till the exit flag is set to join with the main thread

temp_main() – This function sets up the I2C bus to read from temperature sensor. Then it reads the instantaneous temperature value from the sensor and returns it to the calling function using a passed parameter. The function returns the exit status based on success or failure

temperature_handler() – This function is called periodically due to the temperature timer event. This function reads the latest temperature value and populates the message structure with the temperature value or ALERT message and sends it to the logger task.

Socket_function() – This socket pthread created is linked to this function. This function handles the connection request from external processes and provide them with requested data.

led_control() – This function is used to handle the BBG user leds and control them according the parameters passed to this function.

Test API calls

Conversion Test

```
**********TEST CONVERSION**********
Test I2C Init:Pass!
Pass!

Test Temperature in Celsius:Temp:25.875000C  Pass!

Test Temperature in Kelvin:Temp:78.574997F  Pass!

Test Temperature in Farenheit:Temp:299.375000K  Pass!
**********END**********#
```

Screenshot:1A

Test_I2C_init_Conversion

This function checks the I2C path as well as I2C bus allocation and returns PASS on successfull allocation.The file parameter provides the path to the I2C.

Test_Read_Temperature_Celsius

This test function reads raw temperature from the register and converts the raw temperature to Celsius

Test_Read_Temperature_Kelvin

This test function reads raw temperature from the register and converts the raw temperature to Kelvin

Test_Read_Temperature_Farenheit

This test function reads raw temperature from the register and converts the raw temperature to Farenheit

My_Assert_Not_Equal:

This function compares the return type of all the test functions with the expected value. Returns PASS upon value not equal to EXIT_FAILURE and FAIL, if equal to EXIT_FAILURE

Main_Conversion:

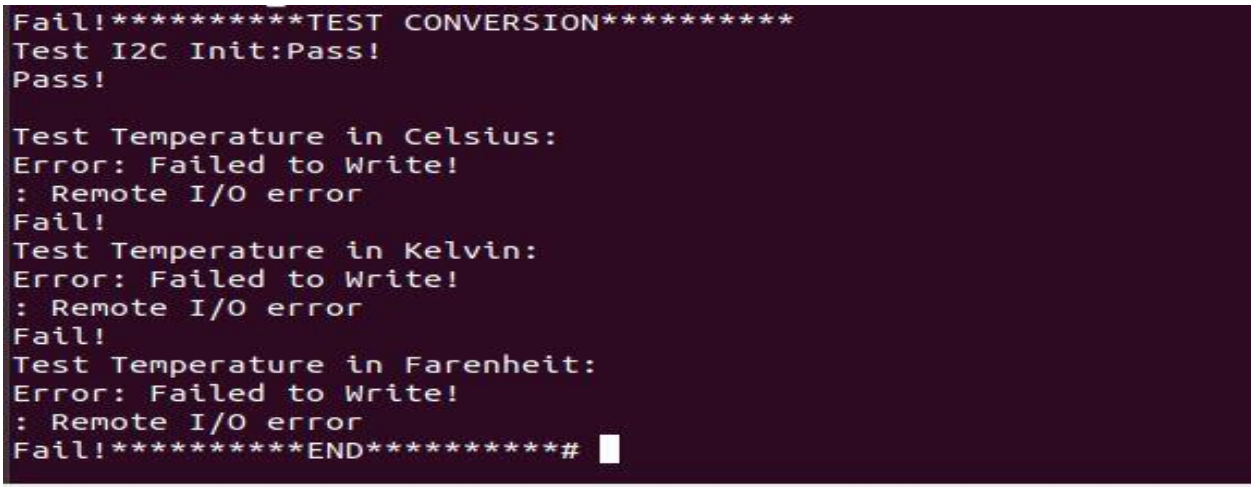
This function calls all the above test functions.

PASS CONDITION:

Various functions are tested and upon succesffull exit, PASS is returned as shown in the screenshot 1A.

FAIL CONDITION:

Various functions are tested and in case of any failure to execute any of the functional procedure, a FAIL status is returned. For example, in cases like SDA or SCL pin not connected, all the test modules involving I2C bus allocation, I2C read or write returns a failure as shown in screenshot 1B.



Screenshot:1B

Light Test

Screenshot:2A

The Unit test for Light Sensor APDS 9301 tests the sensor for various functionality of the sensor. In this project, the following unit test are ran as follows:

Test_I2C_init_Light:

This function checks the I2C path as well as I2C bus allocation and returns PASS on successfull allocation.The file parameter provides the path to the I2C.

Test_Turn_on_light_sensor:

This test function cheks if the write to the control register is successful and returns a PASS on successful powering up. In case of SDA or SCL pin not connectied to the BBGreen, the function fails and returns a FAIL

Test_Read_sensor_ID:

This test function reads the sensor ID, and returns PASS on succesffull read.

Test_Light_Sensor:

This function calls Read Data to read from channel one and two and returns PASS upon processing the raw temperature values

Test_Read_Data:

This function reads the 16 bit ADC channels ny writing the ADC low, high values to the sensor regsiter and returns PASS as test result

Test_Write_Interrupt:

This function writes threshold Low Low, threshold Low High, threshold High Low, threshold High High values to the sensor and returns PASS

Test_Read_Interrupt:

This function reads the threshold values from the sensor and returns PASS on succesffull read

Test_Status:

This function reads the Lux values from the sensor and compares the Lux values with the threshold values to define the state as Light or Dark

main_light:

The function: This function calls all the above test function.

PASS CONDITION:

Various functions are tested and upon succesffull exit, PASS is returned as shown in the screenshot 2A.

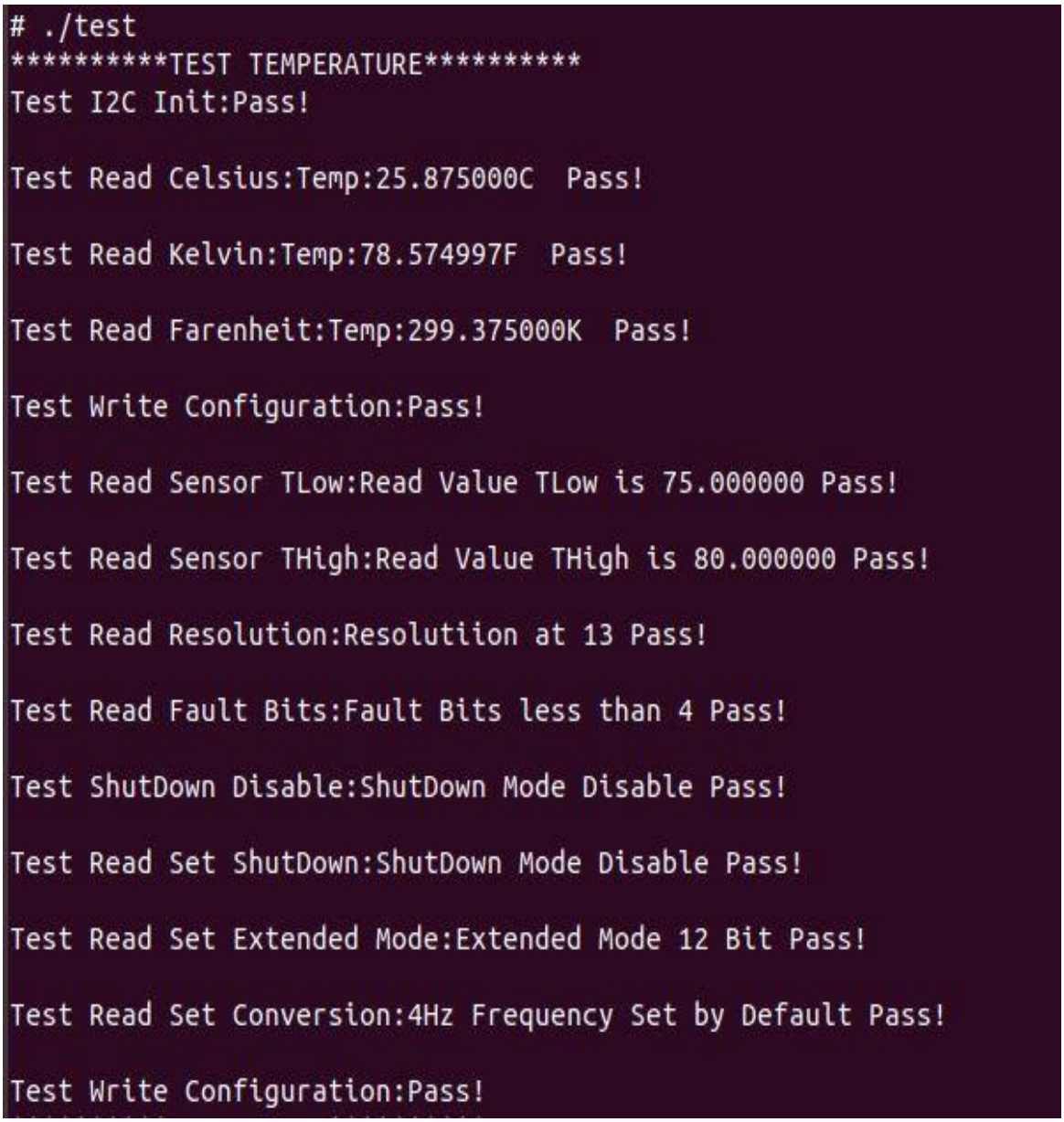
FAIL CONDITION:

Various functions are tested and in case of any failure to execute any of the functional procedure, a FAIL status is returned. For example, in cases like SDA or SCL pin not connected, all the test modules involving I2C bus allocation, I2C read or write returns a failure as shown in screenshot2B.

```
Fail!**********TEST LIGHT**********
Test I2C Init:Pass!

Test Turn Sensor ON:
Error: Sensor Initialization Failed!
Fail!
Test Check Power Up:
Error: Sensor Initialization Failed!

Error: Sensor Initialization Failed!
Fail!
Test Read Sensor ID:
Error: Sensor Initialization Failed!

Error: Sensor_ID Write Failed!
Fail!
Test Light Sensor:
Error: Sensor Initialization Failed!

Error: SensorII Reading Failed!

Error: SensorII Reading Failed!
Pass!

Test Status:
Status: DarkPass!

Test Write Interrupt:
Error: Sensor Initialization Failed!

Error: Sensor_ID Write Failed!
Fail!
Test Read Interrupt:
Error: Sensor Initialization Failed!

Error: Sensor_ID 2Write Failed!
```

Screenshot:2B

Temperature Test



Screenshot:3A

Test_I2C_init_Temp:

This function checks the I2C path as well as I2C bus allocation and returns PASS on successfull allocation.The file parameter provides the path to the I2C.

Test_Read_Sensor_Celsius:

This test function reads raw temperature from the register and converts the raw temperature to Celsius

***Write_pointer_Reg*:**

This function allows writing to the pointer register of the sensor TMP102

Test_Read_Sensor_Kelvin:

This test function reads raw temperature from the register and converts the raw temperature to Kelvin

Test_Read_Sensor_Farenheit:

This test function reads raw temperature from the register and converts the raw temperature to Fahrenheit

Test_Read_TLow:

This test function reads the threshold low value from the sensor register TMP 102, it is set to 75 by default

Test_Read_Thigh:

This test function reads the threshold high value from the sensor register TMP 102 , it is set to 80 by default

Test_Read_Resolution:

This function checks if configuring the resolution is succesfull or not. A 12 bit or a 13 bit resolution can be set

Test_Read_Fault_Bits:

This test function is used to configure or read Fault bits

Test_Clear_Shutdown:

This test function is used to disable Shutdown by disabling the SD bit

Test_Set_Shutdown:

This test function is used to enable Shutdown by enabling the SD bit

Test_Set_EM:

This test function is used to set and clearthe extended mode.Extended mode allows temperature read more than 128C

Test_Set_Conversion:

This test function sets the conversion frequency of the sensor.By defualt 4Hz is set

Test_Write_Configuration:

This regsiter is used to write to the configuration register

Main_temp:

This function call all the test functions for the temperarture

PASS CONDITION:

Various functions are tested and upon succesffull exit, PASS is returned as shown in the screenshot 3A.

FAIL CONDITION:

Various functions are tested and in case of any failure to execute any of the functional procedure, a FAIL status is returned. For example, in cases like SDA or SCL pin not connected, all the test modules involving I2C bus allocation, I2C read or write returns a failure as shown in screenshot 3B.

```
# ./test
**********TEST TEMPERATURE**********
Test I2C Init:Pass!

Test Read Celsius:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Kelvin:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Farenheit:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Write Configuration:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Sensor TLow:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Sensor THigh:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Resolution:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Fault Bits:
Error: Failed to Write!
: Remote I/O error
Fail!
Test ShutDown Disable:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Set ShutDown:
Error: Failed to Write to Pointer!
: Remote I/O error
```

```
Test Read Sensor THigh:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Resolution:
Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Fault Bits:
Error: Failed to Write!
: Remote I/O error
Fail!
Test ShutDown Disable:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Set ShutDown:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Set Extended Mode:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write!
: Remote I/O error
Fail!
Test Read Set Conversion:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write_Convert!
: Remote I/O error
Fail!
Test Write Configuration:
Error: Failed to Write to Pointer!
: Remote I/O error

Error: Failed to Write!
: Remote I/O error
```

Screenshot:3B

References

- http://www.ti.com/lit/ds/symlink/tmp102.pdf
- https://www.broadcom.com/products/optical-sensors/ambient-light-photo-sensors/apds-9301
- http://www.ti.com/lit/ds/symlink/tmp102.pdf
- https://www.sparkfun.com/products/13314
- https://www.broadcom.com/products/optical-sensors/ambient-light-photo-sensors/apds-9301
- https://learn.sparkfun.com/tutorials/apds-9301-sensor-hookup-guide/all
- http://cunit.sourceforge.net/doc/writing_tests.html
- AESD-Project-1 reference guide.

Project staff

Harsimransingh Bindra	Yasir Aslam Shah
Graduate student Embedded systems engineering University of Colorado, Boulder Harsimransingh.bindra@colorado.edu	Graduate student Embedded systems engineering University of Colorado, Boulder Yasir.shah@colorado.edu

Appendix

Negative case when sensor is disconnected

```
# ./cross-main test.log

Test I2C Init:Pass!

Test Read Celsius:Fail!
Temperature Sensor disconnected!

Destroy all
```

Positive case when all start up tests are passed

```
# ./cross-main test.log

Test I2C Init:Pass!

Test Read Celsius:Pass!

Test Read Kelvin:Pass!

Test Read Farenheit:Pass!

Test Write Configuration:Pass!

Test Read Sensor TLow:Read Value TLow is 75.875000 Pass!

Test Read Sensor THigh:Read Value THigh is 80.000000 Pass!

Test Read Resolution:Resolution at 13 Pass!

Test Read Fault Bits:Fault Bits less than 4 Pass!

Test ShutDown Disable:ShutDown Mode Disable Pass!

Test Read Set ShutDown:ShutDown Mode Disable Pass!

Test Read Set Extended Mode:Extended Mode 12 Bit Pass!

Test Read Set Conversion:4Hz Frequency Set by Default Pass!

Test Write Configuration:Pass!

Temperature Sensor Successfully connected

Test I2C Init:Pass!

Test Turn Sensor ON:Pass!

Test Check Power Up:Pass!

Test Read Sensor ID:Pass!

Test Light Sensor:Pass!

Test Status:Pass!

Test Write Interrupt:Pass!
```

Values read periodically

```
temperature message queue created successfully
Successful in creating message queue
Successful in creating temp message queue
light message queue created successfully

Waiting To Recieve Command
Temperature value:25.56C        State:cool
Temperature value:25.56C        State:cool
Light value is 7.38L            State:dark

Temperature value:25.56C        State:cool
Temperature value:25.56C        State:cool
Light value is 7.41L            State:dark

^C
SIGNAL HANDLER CAUGHT

REACHED HERE

Destroy all
```

Part of log file indicating successful login

```
[Timestamp: 15755 seconds]
Thread:temperature
Log level:DATA
sensor_value:25.56C
state:cool
Timer event temperature handler

[Timestamp: 15756 seconds]
Thread:temperature
Log level:DATA
sensor_value:25.56C
state:cool
Timer event temperature handler

[Timestamp: 15756 seconds]
Thread:light
Log level:DATA
sensor_value:7.41L
state:dark
Timer event light handler
- test.log 87/87 100%
```

Part of log file indicating ALERT messages

```
[Timestamp: 15629 seconds]
Thread:temperature
Log level:ALERT
Temperature sensor not functional
```

Part of log file indicating ALERT message is sensor is plugged out while the code is running

```
[Timestamp: 16285 seconds]
Thread:temperature
Log level:ALERT
Timer event temperature handler
```

Console message when the sensor is disconnected and connected back again

```
Temperature value:25.50C        State:cool
Temperature value:25.50C        State:cool
Light value is 4.73L            State:dark

Temperature value:25.50C        State:cool

Temperature Sensor disconnected!

light Sensor Disconnected

Light Sensor disconnected

temperature Sensor disconnected

Light Sensor disconnected

temperature Sensor disconnected

Light Sensor disconnected

temperature Sensor disconnected

Light Sensor disconnected

temperature Sensor disconnected
Temperature value:25.50C        State:cool
Light value is 5.41L            State:dark

Temperature value:25.50C        State:cool
Temperature value:25.50C        State:cool
Light value is 5.97L            State:dark
```

External request