Software Architecture:

Task	Function	Signaling	Interfacing with
Main	Creating child threads Collecting and handling errors related to spawned threads		Child threads – Temperature Task, Light Task, Logger Task
Temperature Name: temptask	1.Configure command message to read/write all registers in the sensor 2.Write the pointer register 3.Read/Write Configuration register 4.Configure sensor resolution 5.Configure shutdown modes for sensor 6.Reads temperature from the sensor and converts it into formats of Cel/Kelvin 7.Should respond to queries for temperature 8. Should respond to heartbeat check	Timer for coordinating with Temperature sensor regarding access to I2C driver POSIX mqueue API to send log packets to the logger task	I2C driver and TMP106 Logger Task Timer
Light Sensor Name: lighttask	1.Configure command message to read/write all registers 2.Read/write the Control register 3.Configure integration time in the timing register 4.Enable/Disable the Interrupt Control Register 5.Read the identification register 6. Read sensor LUX data using the ADC registers	Timer for coordinating with Light sensor regarding access to I2C driver POSIX mqueue API to send log packets to the logger task	I2C driver and APDS 9301 Logger task Timer

Synchronized Logger Name: loggertask	1.Query sensor values from lighttask and temptask. 2. Incorporate protection from multiple log sources using control synchronization(mutex) 3. Create a log packet containing timestamp, loglevel, logger source id, log message using a struct log_packet 4. Synchronize querying to all tasks so file isn't overwritten incidentally 5. Logs all information from sensors in a file name taken at the command line from the main task 6. Query tasks when major events occur (Ex: initialization of configuration register) 7.Cannot die unexpectedly. Must close all file handles before exiting thread 8. Should be able to handle both integer/float and string type data	POSIX mqueue API to receive log packets from the temp and light tasks	Main Light Task Temp Task Message API
Decision Thread Name: decisiontask	1.Read from queues for both sensor tasks 2. Obtain threshold values from main 3. Test if queue popped data exceeds threshold values 4. Raises alert if threshold crossed 5, Raises alert when heart beat for either sensor threads fails	POSIX mqueue API to receive log packets from the temp and light tasks Heart beat signal from timer handler	Logger Queue Temp task Light Task

Tasks:

1. Main Name: Main – Responsibility: Spawn all child threads, monitor threads periodically (heartbeat)

2. Temperature

3. Light Sensor

4. Logger

5. Decision Thread

Generic Driver: I2C

Generic API for Message Interface: Posix mqueue

Global User Defined Data Types:

Log Message: Log messages will have a generic structure as they need to adhere to log messages from both sensors and main with data ranging from integers, float and strings. The struct will probably be as below:

typedef struct log{

timestamp time; //records time of arrival of log message

int loglevel; //levels ranging from normal to error /urgent

char logger_source_id[]; //string depicting sender of log(lighttask/main/temptask

float log_message[]; //array of float type that can possibly contain values int and char

}log_t;

Error Handling: We plan to use the errno library or the following enum

Typedef errorlist{

THREAD_CREATE_FAILED,

THREAD_CREATE_SUCCEEDED,

QUEUE_FULL,

QUEUE _EMPTY,

REG_WRITE_FAILED,

REG_WRITE_SUCCEEDED,

Stubs of Functions classified according to tasks:

1. Main
    a. int main(int argc, char* argv[])
    b. Open file handle
    c. error_t Spawn_Logger_thread (FILE* fp)
    d. error_t Spawn_Light_thread(void)
    e. error_t Spawn_Temperature_thread(void)
    f. Monitor logger thread for failure

2. Light Sensor Task
    a. error_t controlreg()
    b. error_t commandreg()
    c. error_t timingreg()
    d. error_t thresholdreg(int threshold high, int threshold low)
    e. error_t interruptreg()
    f. error_t adcchanneldatareg()
    g. float returnlum()
    h. error_t pushLogQueue(log_t packet)
    i. error_t popLogQueue(log_t packet)
    j. void createlogpacket()
    k.
3. Temperature Sensor Task
    a. float converttoKelvin (float rawtemp)
    b. float converttoFarenheit (float rawtemp)
    c. float converttoCelcius (float rawtemp)
    d. error_t pointerreg ()
    e. error_t readTempReg (int regflag)
    f. error_t ConfigureReg (int regflag)
    g. error_t pointerReg (int regflag)
    h. error_t pushtoLogQueue (log_t packet)
    i. error_t popfromLogQueue (log_t packet)
    j. void createlogpacket()
    k. optionally: error_t configure_high_reg()
                   error_t configure_low_reg()
    l. void settimer(timer_property, struct timerval* timer, struct itimerval* anotherpointer)

4. Synchronized Logger Task
    a. void popqueue()
    b. Write to file
    c. Sigwait
    d. Exit gracefuuly