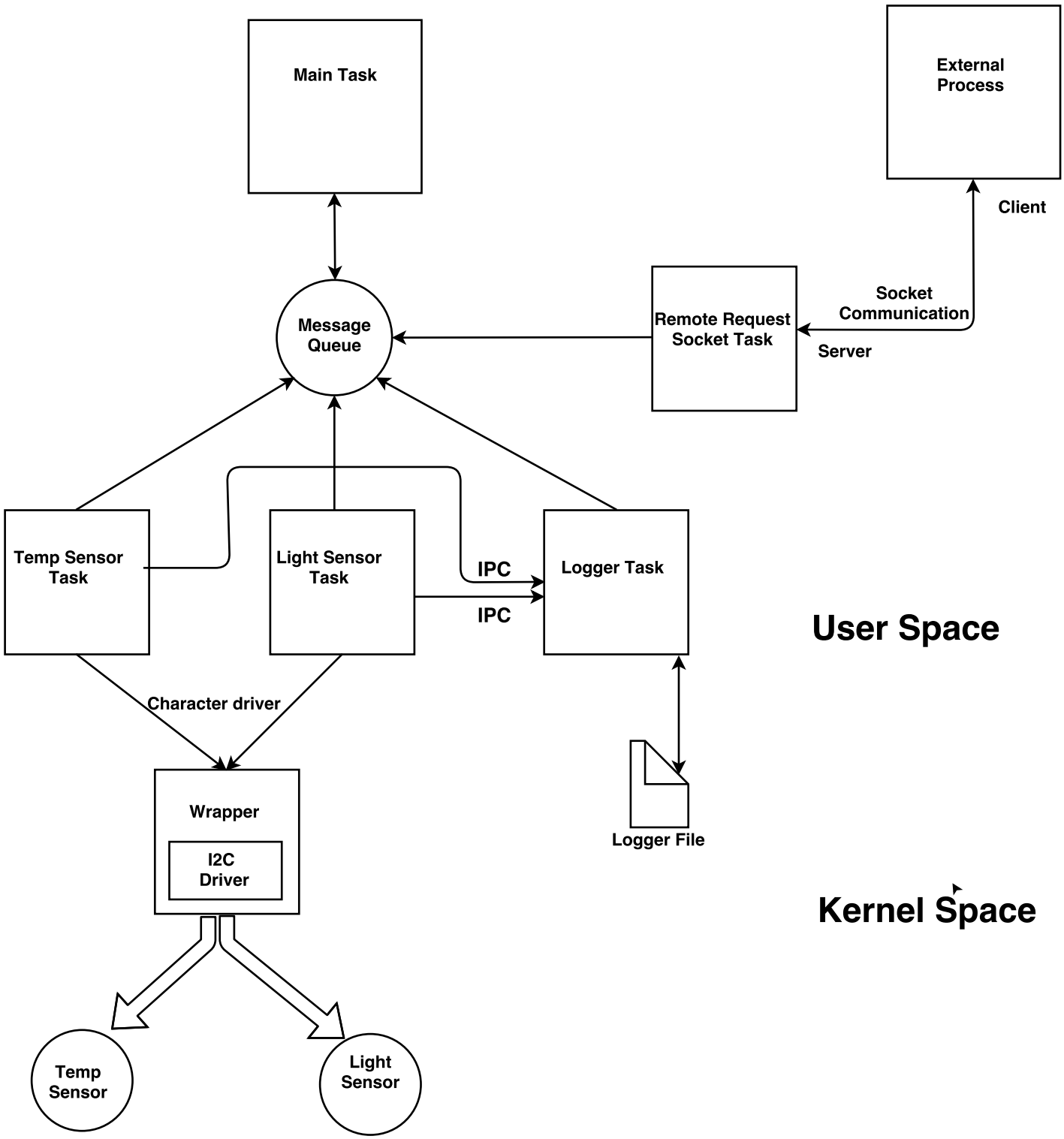
Main Task
External Process
Client
Message Queue
Remote Request Socket Task
Socket Communication
Server
Temp Sensor Task
Light Sensor Task
IPC
IPC
Logger Task
User Space
Character driver
Wrapper
I2C Driver
Logger File
Kernel Space
Temp Sensor
Light Sensor

SOFTWARE ARCHITECTURE DESCRIPTION

USER SPACE :

1. Main Task

Create tasks using pthread_create() i.e task for temp sensor, light sensor, logger and remote request socket task.

Receive heartbeat from all the children task which says whether it is alive or not. This heartbeat comes into message queue.

Running status: check timer value running in all threads to be greater than zero;

Use pthread_join() to safely terminate the threads.

Use send_signal() function which sends signal to children tasks to terminate itself. If that does not succed even with some timeout, use pthread_kill(); to kill children tasks.

Use error_condition() to log any error information to the user through logger, also through LEDs.

2. Temperature Sensor Task

Send heartbeat message to main task using heartbeat_temp() function on regular intervals achieved with posix timer nanosleep() function.

One write_temp(register, value); function with two parameters. Register to which the value must be written and the Value itself. Return NULL on any error.

One read_temp(register); function with register from which value must be read as parameter. Return the value read or NULL on error.

Processing the data using process_data(raw_data); function taking raw data from the sensor as parameter to represent it in different units and to meet required part.

Use posix timers to read data on regular intervals. Specifically timer_settime() function.

Using queues IPC is established so that it can respond to an external request.

All the above-mentioned functionalities will have access to a queue through which it can pass logging data to logger task. Any errors or temperature data with timestamp, log level and thread ID will be passed from the function log_temp() to the Logger queue.

When a terminate signal is received from the main task, it should call pthread_exit() to terminate itself.

3. Light Sensor Task

Send heartbeat message to main task using heartbeat_light() function on regular intervals achieved with posix timer nanosleep() function.

One write_light(mode, register, value_high, value_low); function with four parameters. Mode to specify whether its byte or word operation, Register to which the value must be written, higher byte value and lower byte value. Return NULL on any error.

One read_light(mode, register); function with two parameters. Mode to specify whether it's a byte or word operation and register from which value must be read. Return the value read or NULL on error.

Processing the data using process_data(raw_data); function taking raw data from the sensor as parameter to calculate the luminosity and to say whether it is day/night.

Use posix timers to read data on regular intervals. Specifically timer_settime() function.

Using queues IPC is established so that it can respond to an external request.

All the above-mentioned functionalities will have access to a queue through which it can pass logging data to logger task. Any errors or light data with timestamp, log level and thread ID will be passed from the function log_light() to the Logger queue. Also log unexpected change, if any.

When a terminate signal is received from the main task, it should call pthread_exit() to terminate itself.

4. Logger Task

Send heartbeat message to main task using heartbeat_logger() function on regular intervals achieved with posix timer nanosleep() function.

This task takes logging data from different tasks using Logger queue and writes it to a file. Mutex lock must be used while reading the queue in-order to maintain data integrity. The log file path can be obtained as argument to the task. Check if file exists or not using access() or stat() functions and delete the file, if exists, using unlink(filename) function. A structure struct_log should be allocated memory dynamically using malloc. Struct_log contains timestamp, log level, thread ID and message.

Open the file given by the user using fopen() and print all the data present in the queue to file using fprintf(); function.

When a terminate signal is received from the main task, it should call pthread_exit() to terminate itself.

5. Remote Request Socket Task

Send heartbeat message to main task using heartbeat_socket() function on regular intervals achieved with posix timer nanosleep() function.

This task should create the socket using socket(domain, type, protocol); with domain, type and protocol as parameter. Bind it to specific address(port) using bind(socket, *addr, addr_length); taking socket, address and address length as parameter. Make sure that the port is open. Now, it accepts request from the external process and receives the data. This data is decoded and communicated to either light sensor task or temperature sensor task using message queue. Takes back the data from temp/light sensor and sends it to external process through same socket. Data can be exchanged using read() and write() functions.

When a terminate signal is received from the main task, it should call pthread_exit() to terminate itself.

Startup/System Tests

Making sure that the light and temperature tasks are up and running using heartbeat and using logger. Make sure that you are getting heartbeat from all the threads. Both of these functionalities are implemented in their respective tasks and hence no extra thing has to be done.

Kernel Space:

I2C Wrapper

This wrapper provides interface between user space applications to the and underlying i2c driver using /dev interface. In order to write to an I2C sensor the function write(file, source, length); is used. The function takes file pointer from /dev, source to be written to I2C device and length of the data as parameter. To read from the sensor read(file, destination, length); is used. The function parameters are file from /dev, destination where the received data has to be stored and the length of the data. This wrapper is made as kernel module and can be installed using insmod command.

This wrapper is accessed using character drivers i.e i2c file in /dev folder.