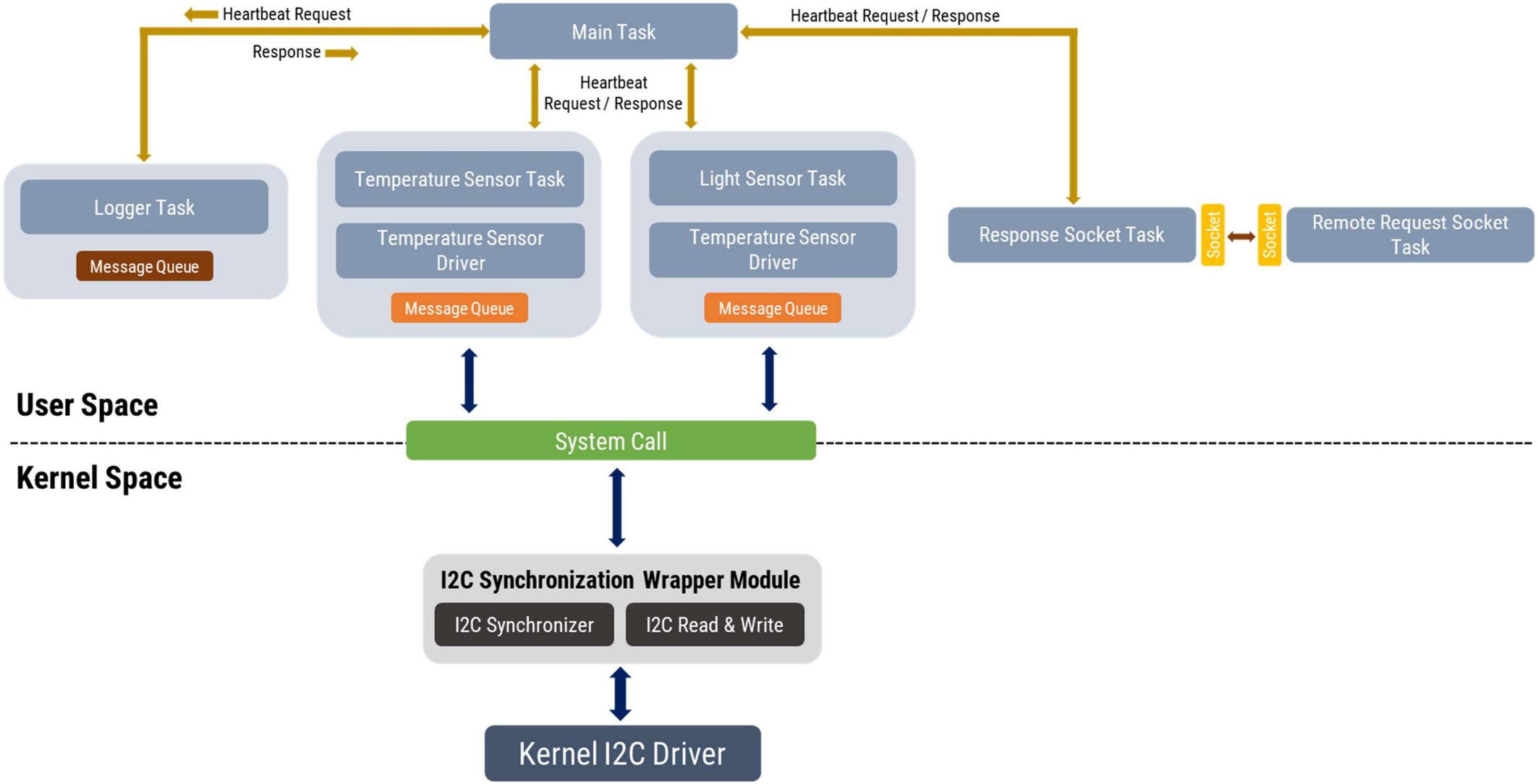
ECEE 5013 – Advanced Practical Embedded Software – Project 1 – Sandeep Raj Kumbargeri & Prithvi Teja Veeravalli

Software Architecture Block Diagram



Software Architecture Description

Main Task

This main task(thread) is like a parent task for the other 4 tasks implemented as part of this project. This task is responsible for create/spawn all the four children tasks(threads) and perform the whole setup. Once it does the whole initial setup, it performs a bunch of system tests which checks if all the tasks, the requisite hardware and drivers are up and running. Once the system tests respond with an affirmative, the main goes to its normal mode of operation, where it continuously monitors whether all the tasks are alive by sending a heartbeat request and checking if the tasks respond with a heartbeat alive response.

The function calls/interfaces implemented as part of this are:

- create_childern_tasks():- This function is used to spawn all four of the children tasks and perform the initial setup.
- system_tests():- This function is a wrapper function for all the system tests and it checks if all the requisite hardware and drivers are functional.
- cleanup_handler():- This callback handler gets called when a request to stop this application is made. This task is responsible for joining on all the 4 children threads in a systematic manner.
- heartbeat_request():- This function is used to send a heartbeat request to a task. Which basically provides a mechanism for checking if a task is alive.

Temperature Sensor Task

This task is used to interact with the temperature sensor through the temperature sensor driver which makes use of I2C Synchronization Wrapper in the kernel.

The temperature sensor driver has the following interfaces or functions.

- write_pointer_register():- This function is used to write the pointer register.
- read_tLow_register():- This function is used to read the contents of the tlow register and return it.
- read_tHigh_register():- This function is used to read the contents of the thigh register and return it.
- read_temperature():- This function is used to read the temperature readings and return it. There is an option to select units in whwich the temperature should be returned.
- shutdown_temp_sensor():- This function is used to shutdown the temperature sensor by setting the configuration register of the sensor appropriately.
- set_temp_resolution():- This function is used to set the resolution of the temperature sensor by setting the configuration register of the sensor appropriately.
- set_conversion_rate():- This function is used to set the conversion rate of temperature sensor by setting the configuration register of the sensor appropriately.
- read_fault_bits():- This function is used to read thee fault bits.
- read_conversion_rate():- This function is used to read the conversion rate of the temperature sensor.
- heartbeat_alive_respond():- This function is used to respond to the heartbeat request from the main task with an alive response.

Light Sensor Task

This task is used to interact with the light sensor through the light sensor driver which makes use of I2C Synchronization Wrapper in the kernel.

The light sensor driver has the following interfaces or functions

- write_command_register():- This function is used to write to the command register.
- write_control_register():- This function is used to write the control register.
- set_integration_time():- This function is used to set integration time of the light sensor.
- set_gain():- This function is used to set the gain of the light sensor.
- read_timing_register():- This function is used to read the timing register of the light sensor.
- enable_interrupts():- This function is used to enable interrupts from the light sensor by setting the interrupt control register appropriately.
- disable_interrupts():- This function is used to disable interrupts from the light sensor by setting the interrupt control register appropriately.
- read_interrupt_threshold():- This function is used to read the interrupt threshold.
- write_interrupt_threshold():- This function is used to change the interrupt threshold by writing the interrupt threshold.
- read_sensor_lux():- This function is used to read sensor lux by reading the ADC registers.
- heartbeat_alive_respond():- This function is used to respond to the heartbeat request from the main task with an alive response.

Synchronizing Logger Task

This function is used to receive logs from various sources and this task should be responsible for writing all of the logs to a common log file whose location is given at the time of executing the program image. The synchronization is achieved through a logger queue.

The driver task makes use of the following interfaces/functions:-

- check_logger_queue():- This function is used to check if the logger queue is empty. It returns true if the logger queue is empty.
- write_logs():- This function is used to log the log information which is first in the logger queue.
- heartbeat_alive_respond():- This function is used to respond to the heartbeat request from the main task with an alive response.

Remote Request Socket Task

This task accepts requests for light or temperature readings from light and temperature sensor from an external process. This accepts such requests from an external process with the help of unix sockets. The unix sockets keeps listening for connections from an external process. Once a connection with an external process is made, it is ready to accept requests from the external process.

This task accepts requests for light or temperature readings from light and temperature sensor from an external process. This accepts such requests from an external process with the help of unix sockets. The unix sockets keeps listening for connections from an external process. Once a connection with an external process is made, it is ready to accept requests from the external process.

I2C Synchronization Wrapper Module

This kernel module is used to wrap around the default I2C driver module provided as part of the kernel. This module makes sure to provide exclusive access to I2C bus to a particular task as there are 2 tasks trying to read and write to the I2C connected sensors which uses the same I2C bus. I am thinking about implementing this using a spinlock as a kernel synchronization device.