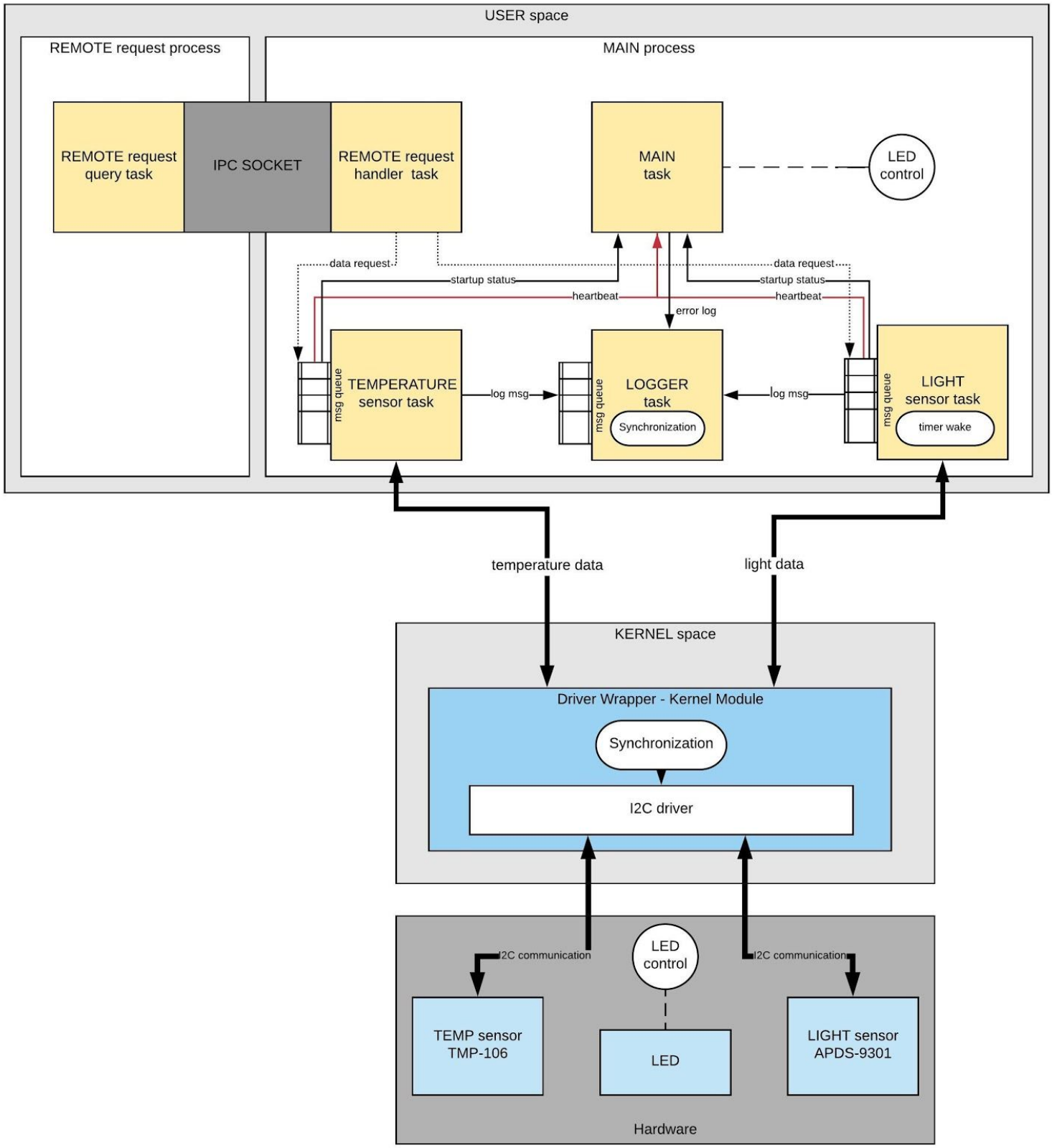
Software Architecture - Project 1

Advanced Practical Embedded Software Development

Praveen Gnanasekaran
Vijoy Sunil Kumar

Date: 02.15.2018

Top level Software Architecture view



Documentation on needed software structures, tasks names, and task responsibilities will need to be provided.

There will be 5 **tasks** in total in the **MAIN process**

- Main task
- Temperature sensor task
- Light sensor task
- Logger task
- Remote request handler task

And **1 task** in the **REMOTE process**

- Remote request query task

Software Structures and typedefs
The project will include the following:

Typedefs

```
typedef enum{
        TEMPERATURE,
        LIGHT
}task_src_t;

typedef enum{                   //used in log message packet- error log or normal log
        NORMAL,
        ERROR
}log_level_t;

typedef enum{                   //used in heartbeat message packet
        TEMP_ALIVE,
        LIGHT_ALIVE,
        TEMP_DEAD,
        LIGHT_DEAD
}task_status_t;

typedef enum{
        NIGHT,
        DAY
}light_format_t;
```

```
typedef enum{
        CELCIUS,
        KELVIN,
        FAHRENHEIT
}temperature_format_t;
```

Structures

```
typedef struct startup{              //sent from children threads to main after startup
        task_src_t source;
        task_status_t status;
}startup_packet_t;

typedef struct logmsg{
        uint32_t timestamp;          //HH:MM:SS:mm
        log_level_t level;           //Error log or normal operation log
        task_src_t source;           //TEMPERATURE/LIGHT
        uint32_t payload;            //temperature data or light data or error data
}msg_packet_t;

typedef struct heartbeat{
        uint32_t timestamp;          //HH:MM:SS:mm
        task_src_t source;           //TEMPERATURE/LIGHT
        task_status_t status;        //Alive or Dead
}beat_t;

typedef struct remote_req{
        task_src_t  source;          //make requests to temp or light
        uint32_t reg_addr;           //address to read data or config register etc.
}remote_t;
```

Task responsibilities will be as follows:

- **Main task**
  Create children tasks, clean-up on exit, LED control on error, ensure children threads are running - heart beat, receive startup tests results from children tasks.
- **Temperature sensor task**
  Configure sensor, read and write registers on sensor-like Tlow THigh, configuration and data , timer for periodic wake up and measurement of data, heartbeat to main, setup message queue, startup tests to check hardware is working, receive request from remote task
- **Light sensor task**
  Configure sensor, read and write to registers- like command, control , timing, interrupts, threshold and configuration registers, timer for periodic wake up and measurement of data, heartbeat to main, setup message queue, startup tests to check hardware is working, receive request from remote task.
- **Logger task**
  Collect log information from the other threads (temperature, light) such as timestamp, log level (severity of log like alarm or debug logs), log ID to indicate which task files the log and the log message, Collect error log from main, setup message queue.
- **Remote request query task**
  The requesting query task will open socket based on port number and wait for the handler task to accept it so that data transfer can occur between sockets, send data requests to MAIN process.
- **Remote request handler task**
  create sockets from other processes, receive data requests from REMOTE request process and send data requests to temperature and light tasks.

API functions

Temperature task will contain APIs like

startup_packet_t check_startup_temp(void);	check hardware status - sensor, I2C
uint8_t send_startup_temp(**startup_packet_t** startup_result);	returns success/error in sending startup status to main
void set_register_temp(**uint32_t** reg_addr, **uint8_t** value);	Write to registers
uint8_t get_register_temp(**uint8_t** reg_addr);	Read from register
uint32_t format_convert(**temperature_format_t** format);	returns transformed data
void init_timer_temp(**uint8_t** period);	Configure timer
uint8_t log_data_temp(**msg_packet_t** temp_packet);	returns success/error in sending packet to logger task
uint8_t send_beat_temp(**beat_t** temp_beat);	returns success/error in sending heartbeat to main
uint32_t check_temp_req(**void**);	Check request from remote request handler task, gets reg address to read.

For Light sensor task,

startup_packet_t check_startup_light(void);	check hardware status - sensor, I2C
uint8_t send_startup_light(**startup_packet_t** startup_result);	returns success/error in sending startup status to main
void set_register_light(**uint32_t** reg_addr, **uint8_t** value);	Write to registers
uint8_t get_register_light(**uint32_t** reg_addr);	Read from register
uint32_t format_convert**(light_format_t** format);	returns transformed data - day or night
void init_timer_light(**uint8_t** period);	Configure timer
uint8_t log_data_light(**msg_packet_t** light_packet);	returns success/error in sending packet to logger task

uint8_t send_beat_light(**beat_t** light_beat);	returns success/error in sending heartbeat to main
uint32_t check_light_req(**void**);	Check request from remote request handler task, gets reg address to read.

For MAIN task,

uint8_t spawn_children(void);	sets thread attributes, creates threads Return status
void get_heartbeat(**beat_t** beat);	check heart beat from children and writes to logger task on error
void get_startup(**startup_packet_t** startup);	check startup status from temperature, light tasks and writes to logger task on error
uint8_t log_errort(**msg_packet_t** error_package);	send error to logger task Return status
void do_led(**uint8_t** led_port);	Controls led on error - start up/heartbeat error
uint8_t destroy_all_children(void);	Close children threads - gracefully Return status

For LOGGER task,

uint8_t write_log(**msg_packet_t** packet);	Logs message packet from temperature task / light task / error message from main Return status

For REMOTE request handler task,

void send_temp_req(**uint32_t** reg_addr);	Send request to temperature task, with reg address to read
void send_light_req(**uint32_t** reg_addr);	Send request to light task, with reg address to read
uint8_t socket_join();	Join socket from REMOTE process.

For REMOTE request query task,

uint8_t socket_connect();	Create socket from REMOTE request process to MAIN process. Return status

Other important concepts will be what is user vs. kernel space software. Indicate if something is a kernel module or a system call.

KERNEL SPACE	**USER SPACE**
Kernel module as wrapper for I2C driver	MAIN task
	TEMPERATURE sensor task
	LIGHT sensor task
	LOGGER task
	REMOTE request handler task
	REMOTE request query task