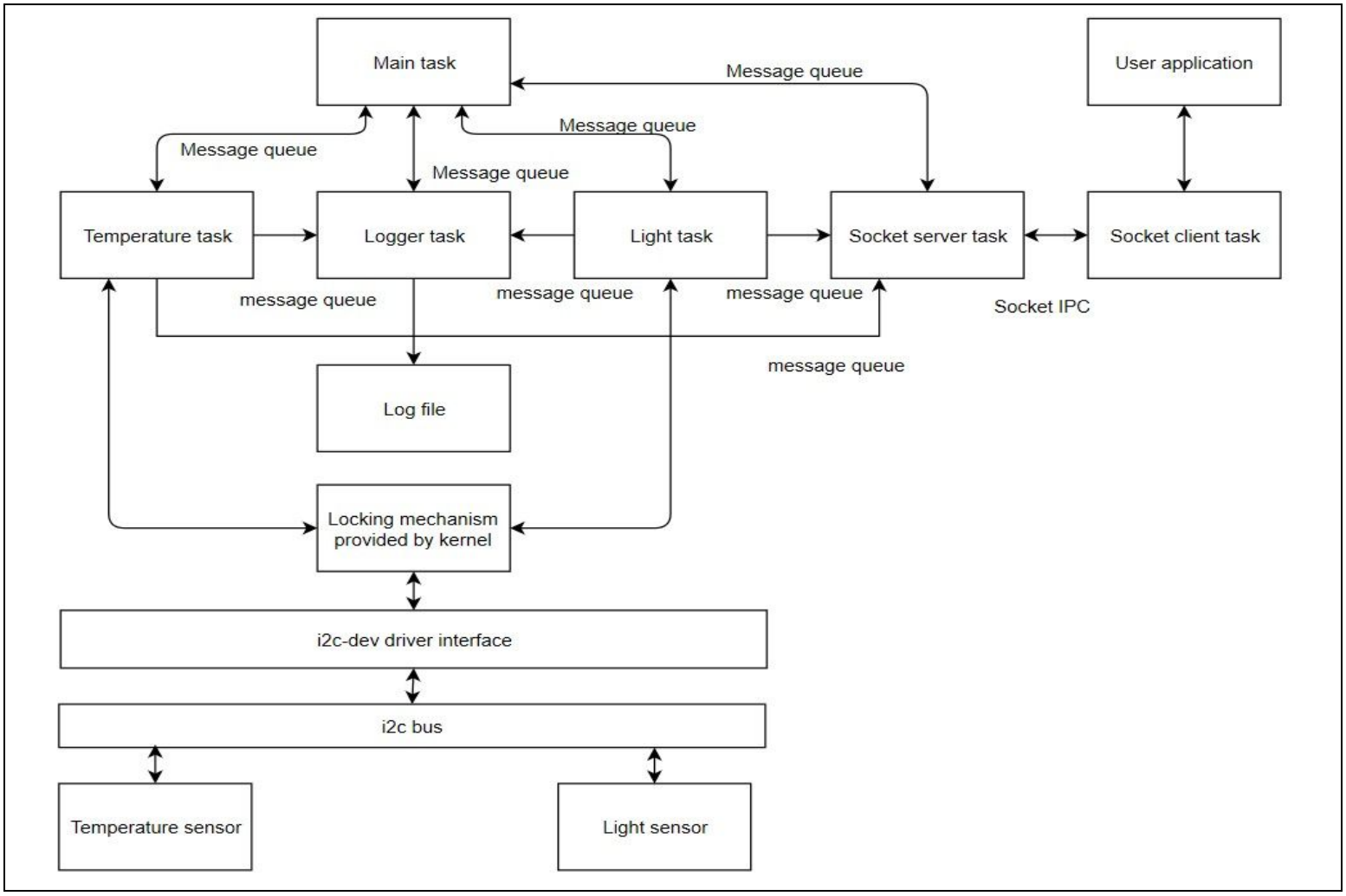
Advanced Embedded Software Development

Puneet Bansal 31st March 2018 Nachiket Kelkar

GITHUB LINK: https://github.com/NachiketKelkar/AESD-Project_1

SYSTEM AND SOFTWARE ARCHITECTURE



Advanced Embedded Software Development

Puneet Bansal 31st March 2018 Nachiket Kelkar

Tasks Description

Main Task –

1. Main task checks if the log file name is provided by the user at runtime. If it is provided it send the information to logger task during initialization else it reports the error to the user that user needs to provide the log file name. initialises the i2c and perform built is self test. It reads the register values from the sensor which are configured at startup. If the test fails (i.e if sensors are not connected) the tasks exists.
2. It then initializes the queues for getting heartbeat messages or request message from socket task, It also initializes the timers which checks if the heartbeat messages are received at appropriate time.
3. Main task then creates 4 tasks namely Temperature Sensor Task, Light Sensor Task, Logger Task and Socket Server Task.
4. Then main task will wait for all four task to complete the initialization and send the init_success message. It waits for limited time and if time is expired the main task exists exiting all other tasks.
5. After successful initialization of all the tasks it then wait for the message from the child tasks. When the update message i.e heartbeat message is received it updates the timeout timer of that task. If no message is received within timeout period the task is marked as dead.
6. If the request message is received from the socket task the main will will check if the task is respective task is alive. If the required task is alive then the message is forwarded to the respective task. If the task is dead the main task replies that the task is dead to the socket.
7. To stop all the tasks when the Ctrl+C is pressed the main will join on the thread to exit. Clean up all the initialization and then safely exits.
8. The temperature task and light task can access the bus at anytime with no need of any locking mechanism since the kernel is responsible for same and it takes care of exclusive access to the shared resource between two tasks.
9. API
    a. lightSensorBIST - It takes the file descriptor of the i2c-dev of light sensor. Checks the id register value. It compares the value with the value from the datasheet and returns if the comparison was successful. This confirms if the i2c is working and also light sensor is present on the bus.
    b. tempSensorBIST - It takes the file descriptor of the temperature sensor i2c. It reads the configuration register value of the temperature sensor which is same upon reset. It is compared to the default value as per the datasheet and returns the success or failure. This confirms if the i2c bus is working and also the temperature sensor is connected and working well.

Advanced Embedded Software Development

Puneet Bansal 31st March 2018 Nachiket Kelkar

Temperature Sensor Tasks –

1. The main task will spawn the temperature thread.
2. The temperature thread initializes the timer to take readings at regular intervals which is set at 100 millisec. It also initializes the gpio pins, mqueue for accepting and sending the messages, assigns the signal handler for timer, assigns signal to time.
3. It then sets the time and starts the timer and send the init_success message to the main task to indicate all the initializations are successful. It sets the high and low temperature threshold as 26 and 30 degrees.
4. When the timer signal occurs it signifies that the temperature measurement procedure should be started. Then read the temperature from the temperature sensor. If the sensor is pulled out the USR0 led is glowed and the thread is exited thus, graceful exit.
5. It then sends the temperature value to the logger and heartbeat message to main signifying that the task is alive.
6. It checks if the measured temperature is to be sent to the user. If yes it converts the temperature to required unit and sends it to the socket task which forwards it to the user.
7. API
    a. temp_i2c_init - This function opens the i2c file for i2c transactions. It then sets the slave address for the transactions according to the parameter.
    b. temp_i2c_write_to_reg - This function takes the file descriptor as parameter which is used to write to a file. It writes the data to the temperature sensor register which is described in parameter. For writing the data to THIGH or TLOW register the data to write should be in Celsius.
    c. temp_i2c_read_from_reg - This function takes the file descriptor as parameter which is used to write to a file. It reads the received i2c data from register passed and return the buffer value in uint16_t format.
    d. read_temperature - This function takes the file descriptor as parameter which is used to write to a file. It formats the data of the register passed in the format of Celsius unit. As configuration register does not contain temperature passing config register address will cause an error.
    e. convert_to_unit - This function takes temperature and unit to convert to and then converts the value in required temperature unit and returns the value.
    f. gpio_init - The function takes the gpio pin number and assigns it as input pin or output pin.
    g. gpio_write_value - The function takes the gpio pin number and outputs the pin high or low.
    h. gpio_read_value - The function takes the gpio pin number and returns the value on the pin.
    i. is_pin_valid - The function takes the gpio pin number and returns if valid pin no is entered.
    j. gpio_interrupt_state - The function takes the gpio pin number and sets the gpio interrupt as rising falling, both or none based on second parameter.

k. gpio_open_value - The function takes the gpio pin number and opens the file and returns the file descriptor.
l. gpio_read_val_with_fd - The function takes the file descriptor of gpio pin and returns the state of the pin whether high or low.

Light Sensor Task --

The light sensor routine is primarily responsible for taking up Lux value from the APDS-9301 Ambient Light Photo Sensor every 100 ms and log it to the Log File.The light sensor routine has its own message queue and a structure. If any task wants to communicate to the light task, they have to populate the lightStruct and send it to the light message queue.

Other responsibilities of Light Sensor Task include:

•Sending initialization successful message to the main task:

The task initializes all the message queues used for inter-process communication between the various tasks. Along with this a posix timer is set up to provide an interrupt in every 100 ms. When the message queues and the timers are successfully initialized an init_success message is sent to the main task (via message queue)indication successful initialization of the task.

•Sending heart beat messages to the main task throughout the span of program indicating that the task is alive:
A timer is set up in the task to provide a signal in every 100 ms. So at every 100 ms , the task sends a signal to the main task indicating that the task is alive. The task basically populates a “mainStruct” with the following fields

.source=”light”

.status=”success”

.messageType=”update”

•Receive requests from main task to send light sensor readings to the socket task:
The main sends request messages to the light task, requesting the task to send Lux values to the socket server. The light task continuously reads it’s message queue and if it get’s any message with source “main” and message type = “request” it sends the updated lux value to socket task.

•Read the LUX value via I2C and specify based on the threshold, whether the vicinity is bright or dark.

Advanced Embedded Software Development

Puneet Bansal 31st March 2018 Nachiket Kelkar

API's Used:

- **mqueue_init() :** wrapper around mq_open function. Sets the attributes of the queue and opens the queue with the specified parameters.
- **myi2cInit()**: Opens the /dev/i2c-2 file and designates the slave.
- **lightSensorWrite():** Writes the specified number of bytes(len) of data (specified in parameters) to the register (specified in parameter)
- **luxCalc():** reads from adc channel 0 and adc channel 1 using lightSensorRead function. Does the necessary computations to calculate LUX and returns the lux value in float to the user.
- **lightSensorRead():** Function to read from the specified i2c registers using i2c.The register address from where data is to be read, is first written via myi2cwrite function, which is basically a wrapper to write to the file specified by the filedescriptor. After this a read operation of the required number of bytes is performed via myi2cread function. The value received is written on the buffer and returned.
- **myi2cRead() :** wrapper around the read system call to read the designated number of bytes from the file. When the length read from file is equal to the length specified by the user, it indicates success. NULL is returned on failure.
- **myi2cWrite():** wrapper around the write system call to write the designated number of bytes to the file. When the length written to file is equal to the length specified by the user, it indicates success and returns 0 . 0 is returned on failure.

Logger Task –

1. The temperature sensor task, Light sensor task and socket task will send the data through the message queue (IPC).
2. The Logger task will synchronize the process and write the data in to a log file specified as command line parameter.

API's used:

- **logToFile():** Takes the structure with the data as a parameter. Prints this data to the file along with the timestamp, source of the message, loglevel,value and unit.The source can be light sensor, temperature sensor, maintask and sockettask.
  Three log levels are defined.

  1)DEBUG: to print general debug information

  2)INFO: printing the temp and light sensor values

  3)ALERT: important error messages.

Advanced Embedded Software Development

Puneet Bansal 31st March 2018 Nachiket Kelkar

Socket Server Task –

1. This task will act as a socket server and wait for a communication from the client.
2. On receipt of a request from a client it will either it communicate the request to the main task. The main task would check if the sensor is alive or not and send a request to temp/light task. The temperature sensor task or the light sensor task will report the temp/light via message queue to the socket and the socket then send these values to client. (Socket IPC).

API's used:

- All standard Linux and C api's are used for this task.

User Application (Client Task)–

1. The user application is a menu driven program with options to get the temperature sensor readings or the light sensor readings.
2. These are the options available to the client.
    - Get temp in Celsius
    - Get temp in Kelvin
    - Get temp in Fahrenheit
    - Get Lux value
3. It accomplishes this by acting as a socket client (Socket IPC) and connecting to the server IP address.

API - Existing C api's used.

Advanced Embedded Software Development

Puneet Bansal 31st March 2018 Nachiket Kelkar

EXISTING LIBRARIES

1. pthread.h - For task design
2. semaphore.h - For semaphore
3. mqueue.h - For message queues IPC
4. mraa/i2c.h - For i2c communication with the sensors
5. sys/socket.h - For creating server and client sockets
6. linux/timer.h - For handling timers
7. signal.h - For signal handling