RTOS Mini Project

Smart Home System using RTOS Concepts (Simulated Sensors)

S.No	Name	Registeration Number	Branch
1	Adithya Rao Kalathur	251100610012	CSE
2	Ashirvad B M	251100610024	CSE
3	Harish Kumar D	251100610006	CSE

Objective

To simulate a smart home environment controlled by an embedded RTOS, demonstrating key RTOS features such as:

- Periodic Tasks
- Semaphores (with timeouts)
- Message Boxes (with timeouts)
- Mutex with Priority Inheritance
- Priority Ceiling Protocols (ICPP and OCPP simulations)
- Event Flags and Task Communication

Platform

- **Microcontroller:** LPC2378 (ARM7TDMI-S Core)
- **RTOS:** Keil RTX (CMSIS-RTOS style, via <RTL.h>)
- **IDE:** Keil µVision
- **Peripherals Simulated:**
    - Temperature Sensor
    - Light Sensor
    - Motion Detector

- LCD Display
- LEDs for fan/light indicators

System Description

The project emulates a Smart Home Controller that monitors and reacts to:

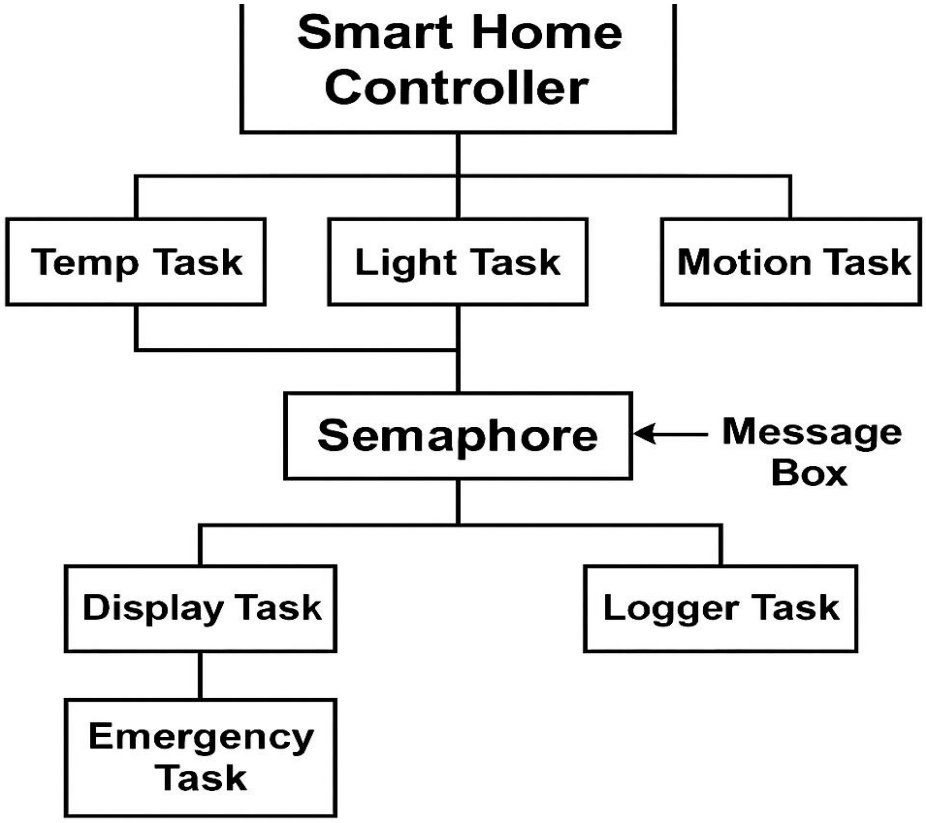
- **Temperature:** Controls fan speed (via LEDs).
- **Light Level:** Adjusts room lighting (via LEDs).
- **Motion:** Overrides light control temporarily for safety/visibility.
- **Clock:** Blinks periodically as a system heartbeat.
- **Logger:** Displays recent system actions.
- **Emergency Monitor:** Alerts in case of overheating.

Each module is implemented as a separate RTOS task, coordinated through synchronization primitives.

Hardware / Software Components Used

Component	Type	Purpose
LPC2378 MCU	Hardware	Main processing core with GPIO for LED/LCD
LEDs (P2.0–P2.7)	Hardware	Indicate system states (fan, light, clock)
LCD (16x2)	Hardware	Display system information and logs
Keil RTX (RTOS)	Software	Manages multitasking and synchronization
Semaphores	Software	Protect shared sensor data
Mutexes	Software	Protect LCD and ceiling manager
Message Box	Software	Queue for log messages
Timers/Delays	Software	Used to simulate real-time behavior

System Architecture



Task Summary

Task Name	Function	Priority	Key RTOS Features Used
temp_task	Simulates temperature & controls fan LEDs	3	Semaphore, ICPP, MessageBox
light_task	Simulates light level & controls lighting LEDs	4	Semaphore, OCPP, Event Flags
motion_task	Simulates motion detection events	2	Event flags
display_task	Displays live readings on LCD	5	Semaphore, Mutex
logger_task	Displays recent log on LCD	6	MessageBox, Mutex
emergency_task	Detects overheating & flashes LEDs	1	Mutex
clock_task	LED heartbeat for system activity	7	Delay (Periodic task)

RTOS Concepts Demonstrated

Periodic Tasks

All tasks in this demo are periodic.
They perform actions, delay for a defined period, and repeat.

Task	Mechanism
temp_task()	os_dly_wait(200); — every 200 ticks
light_task()	os_dly_wait(50); — faster updates
motion_task()	os_dly_wait(800); — detects motion periodically
clock_task()	os_dly_wait(5/95); — blinking heartbeat
emergency_task()	os_dly_wait(50); — checks overheating regularly

Semaphore with Timeout

Used for protecting shared sensor data between multiple tasks.

```
if (os_sem_wait(&sem_sensors, 50) == OS_R_OK) {
sensor_temp = temp;
os_sem_send(&sem_sensors);
}
```

- **Used In:** temp_task(), light_task(), display_task()
- **Timeout:** 50 ticks (prevents deadlock)
- **Purpose:** Avoids race conditions while updating sensor_temp and sensor_light.

Message Box with Timeout

Used to log system activities asynchronously.

```
os_mbx_send(&msgbox, (void*)log_pool[idx], 40);
os_mbx_wait(&msgbox, &msg, 120);
```

- **Used In:**
    - temp_task() → sends temperature logs
    - light_task() → sends light logs

    - logger_task() → waits and displays logs
- **Timeouts:**
    - Send: 40 ticks
    - Receive: 120 ticks

This allows producer-consumer synchronization between tasks.

Mutex with Priority Inheritance

Protects the LCD display from concurrent access.

```
if (os_mut_wait(&mut_lcd, 100) == OS_R_OK) {
   LCD_puts((U8*)line);
   os_mut_release(&mut_lcd);
}
```

- **Used In:** display_task(), logger_task(), emergency_task()
- **RTOS Feature:** If a low-priority task holds the mutex, a higher-priority waiting task inherits its priority temporarily to prevent priority inversion.

Priority Ceiling – ICPP Simulation

Immediate Ceiling Priority Protocol simulated via icpp_acquire() and icpp_release() functions.

```
while (!icpp_acquire(self)) os_dly_wait(5);
update_temp_leds(level);
icpp_release(self);
```

- **Used In:** temp_task()
- **Concept:** When a task acquires a resource (like LEDs), no other task can preempt it until it releases the resource.
- **Simulation:** Uses global ceiling_owner variable.

Variable	Role
ceiling_owner	Tracks which task holds the resource
mut_ceiling	Protects access to ceiling variables

Priority Ceiling – OCPP Simulation

Original Ceiling Priority Protocol simulated via ocpp_acquire() / ocpp_release().

```
if (ocpp_acquire(self, LIGHT_RESOURCE_CEILING)) {
   update_light_leds(level);
   ocpp_release(self, LIGHT_RESOURCE_CEILING);
}
```

- **Used In:** light_task()
- **Resource Ceiling:** LIGHT_RESOURCE_CEILING = 2
- **Concept:**
    - Task can acquire resource only if its base priority ≤ ceiling.
    - Prevents low-priority tasks from blocking higher ones indirectly.
- **Simulation Variables:**
    - system_ceiling (global ceiling state)
    - task_table[] (records task base priorities)

Tables of Synchronization and Communication

Semaphore Usage

Semaphore	Protected Resource	Used By Tasks	Timeout
sem_sensors	Shared sensors (temp, light, motion)	temp_task, light_task, display_task	50 ticks

Mutex Usage

Mutex	Protected Resource	Tasks Using It	Purpose
mut_lcd	LCD display	display_task, logger_task, emergency_task	Avoid concurrent LCD access
mut_ceiling	ICPP/OCPP state	temp_task, light_task	Protects ceiling manager variables

Message Box

Message Box	Size	Producer Tasks	Consumer Task
msgbox	5	temp_task, light_task	logger_task

Event Flags

Flag	Meaning	Sender	Receiver
EVT_TEMP_UPDATE	Temperature update	temp_task	display_task
EVT_LIGHT_UPDATE	Light update	light_task	display_task
EVT_MOTION	Motion detected	motion_task	light_task, display_task

System Behavior Summary

Condition	Action Taken
Temperature rises above threshold	Fan LEDs increase gradually
Room becomes dark	Light LEDs increase
Motion detected	All LEDs flash (safety override)
Temperature > 45°C	Emergency task triggers LCD alert and flashing LEDs
System idle	Clock LED blinks periodically

Example LCD Output

Smart Home System

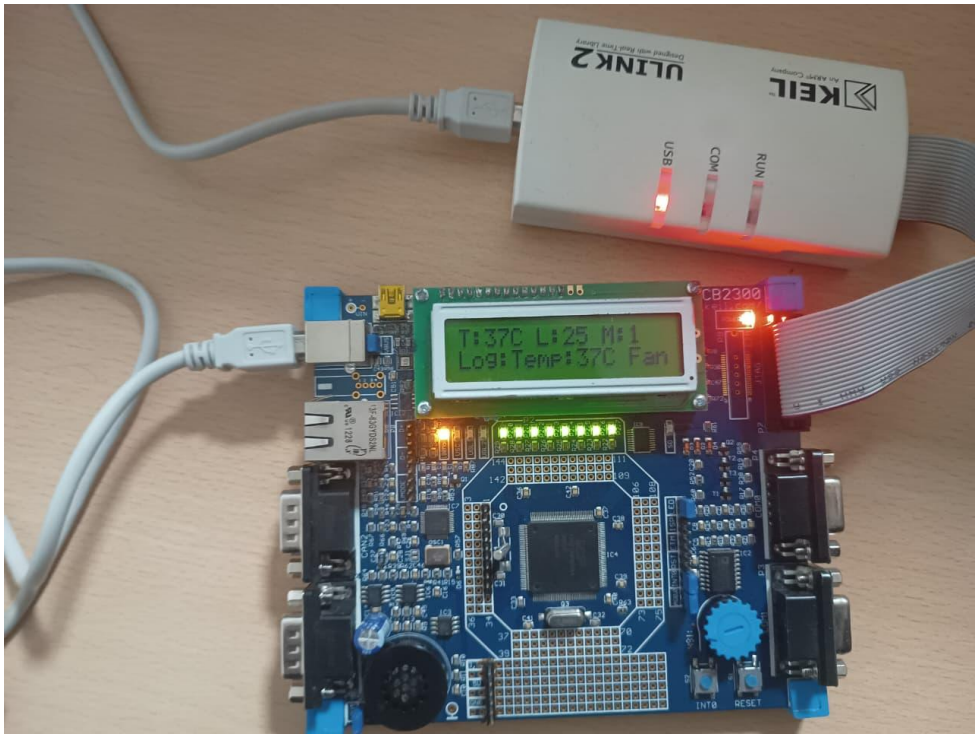
T:30C L:45 M:1

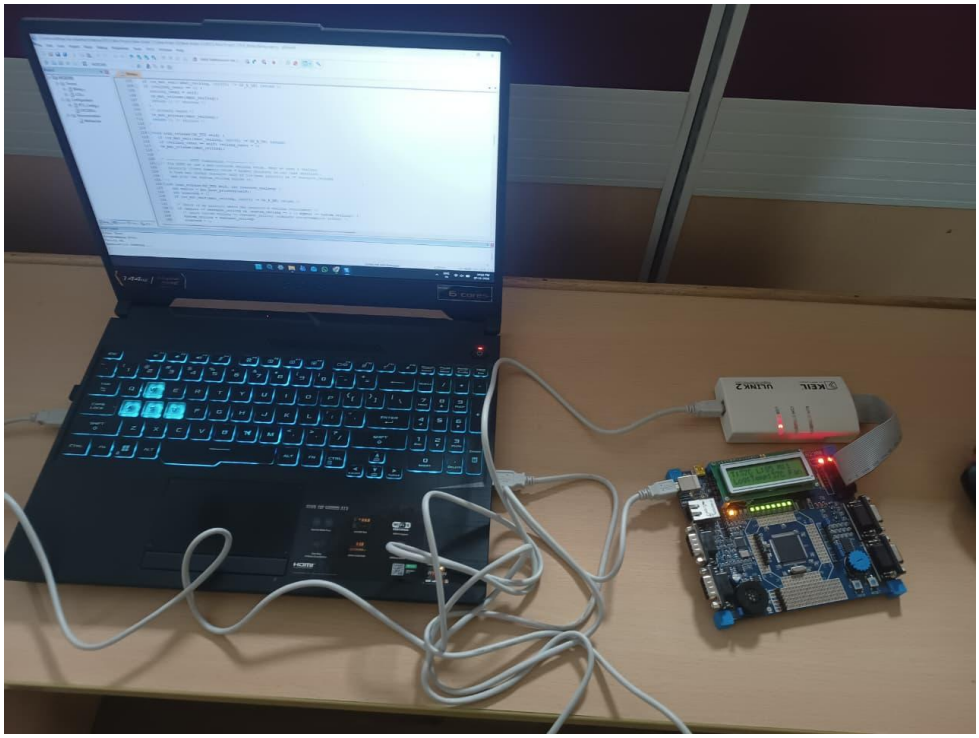
Log:Temp:30C Fan:2

Possible Enhancements

- Add real sensor interface via ADC instead of simulation.
- Implement true priority ceiling protocol using actual os_tsk_prio() API calls.
- Add UART logging or SD card data storage.
- Introduce network control (e.g., MQTT or Wi-Fi).

Result Outputs:





Conclusion

This project effectively demonstrates multiple RTOS synchronization mechanisms in a cohesive smart home simulation.
It showcases:

- Controlled concurrency
- Safe inter-task communication
- Prevention of race conditions
- Handling of real-time constraints

It provides a practical educational platform for understanding how advanced scheduling protocols like ICPP and OCPP can be implemented and observed in embedded systems.