МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій

Кафедра комп'ютерної інженерії та інформаційних систем

ЛАБОРАТОРНА РОБОТА №5

з дисципліни «Теорія і технології проектування спеціалізованих операційних систем»

ЛРКІ.180102.22.01.02 ПЗ

Галузь знань \_\_\_\_\_\_\_\_\_\_12 – Інформаційні технології\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність \_\_\_\_\_\_\_\_\_123 –Комп’ютерна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Виконав:

студент 1 курсу, групи КІ2м-22-1 Біньковський Я.В.

(Підпис)

Перевірив: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лисенко С.М.

(Підпис)

Хмельницький – 2023 р

Тема:

Проєктування вбудованої RTOS для платформи Lego EV3 на базі ОС Linux.

Модифікація базового ядра Linux для забезпечення функцій RTOS. Дослідження роботи RTOS на платформі Lego EV3.

Завдання:

Простий рівень.

Реалізувати вбудовану RTOS для платформи Lego EV3 на базі ОС Linux Оформити звіт з

виконаної роботи.

Реалізація

1. Включення необхідних заголовочних файлів:

* <stdio.h>: для використання функції printf().
* <stdlib.h>: для використання функції exit().
* <pthread.h>: для використання функцій та типів, пов'язаних з багатопотоковістю.
* <unistd.h>: для використання функції sleep().

1. Визначення макросу NUM\_TASKS, який представляє кількість задач, які будуть виконуватись в RTOS.
2. Визначення структури Task, яка представляє задачу в системі. Вона містить вказівник на функцію, яка виконується як задача, та період, у мілісекундах, який визначає, з якою частотою задача повинна бути виконувана.
3. Визначення трьох задач task1(), task2() та task3(), які просто виводять повідомлення у консоль та затримуються на певний період виконання за допомогою функції sleep().
4. Створення масиву tasks, який представляє задачі в системі. Кожна задача в масиві має вказівник на відповідну функцію задачі та період виконання.
5. Визначення функції scheduler(), яка виконує роль планувальника. У циклі вона послідовно виконує всі задачі з масиву tasks.
6. Перед виконанням кожної задачі перевіряється, чи вказано вказівник на функцію задачі. Після виконання задачі, за допомогою функції usleep() здійснюється затримка, що відповідає періоду задачі.
7. У функції main() створюється змінна schedulerThread для ідентифікатора потоку планувальника.
8. Ініціалізація задач шляхом присвоєння вказівників на функції задач та періоди виконання відповідним елементам масиву tasks.
9. Створення потоку планувальника за допомогою функції pthread\_create(). В потоці буде виконуватись функція scheduler().
10. Очікування завершення потоку планувальника за допомогою функції pthread\_join(). Оскільки планувальник працює у безкінечному циклі, цей крок ніколи не відбудеться. Щоб завершити програму, можна натиснути Ctrl+C.
11. Повернення значення 0 з функції main() для позначення успішного виконання програми.

Це базовий код для простого планувальника задач вбудованої RTOS на базі ОС Linux для платформи Lego EV3. Його можна розширити та оптимізувати для використання конкретних функцій та можливостей платформи Lego EV3.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#define NUM\_TASKS 3

// Структура для представлення задачі

typedef struct {

void (\*taskFunc)(void); // Вказівник на функцію задачі

int period; // Період задачі в мілісекундах

} Task;

// Задачі

void task1(void) {

printf("Виконання Задачі 1\n");

sleep(1);

}

void task2(void) {

printf("Виконання Задачі 2\n");

sleep(2);

}

void task3(void) {

printf("Виконання Задачі 3\n");

sleep(3);

}

Task tasks[NUM\_TASKS];

// Функція планувальника

void\* scheduler(void\* arg) {

int i;

while (1) {

for (i = 0; i < NUM\_TASKS; i++) {

if (tasks[i].taskFunc != NULL) {

tasks[i].taskFunc();

}

usleep(tasks[i].period \* 1000); // Затримка відповідно до періоду задачі

}

}

return NULL;

}

int main() {

pthread\_t schedulerThread;

int i;

// Ініціалізація задач

tasks[0].taskFunc = task1;

tasks[0].period = 1000;

tasks[1].taskFunc = task2;

tasks[1].period = 2000;

tasks[2].taskFunc = task3;

tasks[2].period = 3000;

// Створення та запуск планувальника

pthread\_create(&schedulerThread, NULL, scheduler, NULL);

// Очікування завершення планувальника (в даному випадку ніколи не відбудеться)

pthread\_join(schedulerThread, NULL);

return 0;

}