**BÁO CÁO TUẦN 10 ĐỒ ÁN MÔN HỌC**

Tên đồ án: “Ứng dụng các mô hình máy học vào bài toán phân loại hoạt động của người dùng, trên các thiết bị đeo tay theo dõi sức khỏe.”

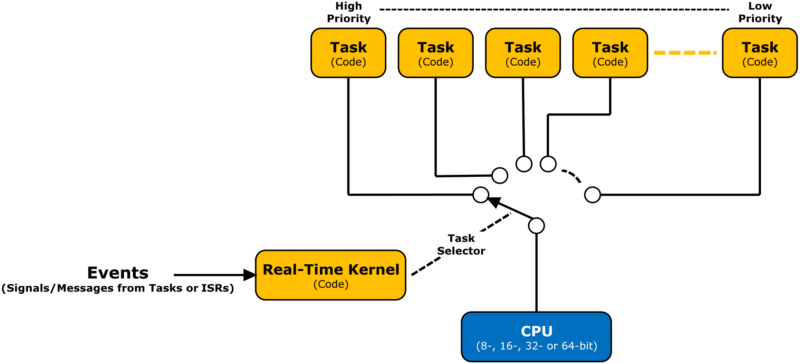
Thành viên thực hiện:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | MSSV | ĐIỆN THOẠI | EMAIL |
| 1 | Nguyễn Ngọc Minh | 19520165 | 0585115056 | 19520165@gm.uit.edu.vn |

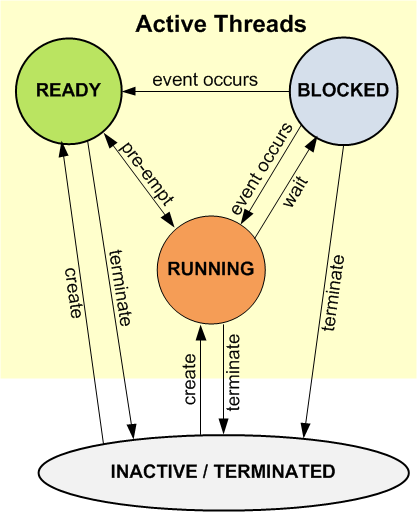
1. Nội dụng thực hiện:

Tối ưu hóa hệ thống.

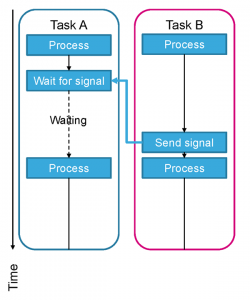
1. Báo cáo quá trình thực hiện:
   1. Tìm hiểu Hệ điều hành RTOS:

* Một hệ thống nhúng lớn phải thực thi nhiều tác vụ thì cần thiết có một hệ điều hành để lập lịch cũng như phân vùng, sử dụng bộ nhớ hiệu quả.
* Đối với STM32F4 (arm cortex M4) platform có tới 5 loại hệ điều hành khác nhau:
  + μClinux
  + μC / OS-II
  + eCos
  + FreeRTOS
  + Hệ điều hành Dujiangyan
* Trong đó, FreeRTOS nổi tiếng với mã nguồn mở, miễn phí, dễ sử dụng.
* RTOS là viết tắt của cụm từ Real-time operating system hay hệ điều hành thời gian thực thường được nhúng trong các dòng vi điều khiển dùng để điều khiển thiết bị một cách nhanh chóng và đa nhiệm (multi tasking)
* Các ứng dụng cần RTOS
  + Ứng dụng nhiều trạng thái máy (States Machine)
  + Ứng dụng lớn
  + Ứng dụng liên quan tới các tác vụ xử lý nhanh, xử lý ảnh, âm thanh.
* Lợi ích khi sử dụng OS nói chung và RTOS nói riêng:
  + Chia sẻ tài nguyên một cách đơn giản: cung cấp cơ chế để phân chia các yêu cầu về bộ nhớ và ngoại vi của MCU
  + Dễ debug và phát triển: Mọi người trong nhóm có thể làm việc một cách độc lập, các lập trình viên thì có thể tránh được các tương tác với ngắt, timer, với phần cứng (cái này mình không khuyến khích lắm vì hiểu được phần cứng vẫn sẽ tốt hơn nhiều)
  + Tăng tính linh động và dễ dàng bảo trì: thông qua API của RTOS,…
* Vì thế, sử dụng RTOS trong đồ án giúp tối ưu cũng như dễ mở rộng và nâng cấp.
  1. Nguyên lý RTOS, Tìm hiểu API và cách sử dụng:
* RTOS là một phân đoạn hoặc một phần của chương trình, trong đó nó giải quyết việc điều phối các task, lập lịch và phân mức ưu tiên cho task, nắm bắt các thông điệp gửi đi từ task.

Hình 1: Điều phối CPU cho các Task

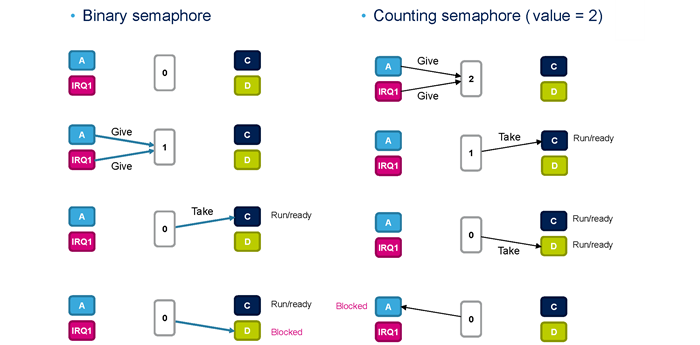
* **Task** là một đoạn chương trình thực thi một hoặc nhiều vấn đề gì đó, được Kernel quản lý.
* Các Task state:
  + RUNNING: đang thực thi
  + READY: sẵn sàng để thực hiện
  + WAITING: chờ sự kiện
  + BLOCKED: không được kích hoạt
* Cơ chế Đồng bộ:

Hình 2: Task state và mối liên hệ giữa chúng.

* + Signal Events

osEventFlagsSet(); //Task tạo signal

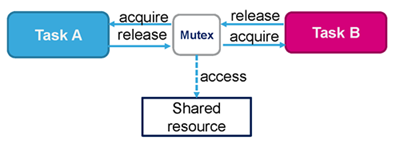
osEventFlagsWait(); // Task chờ signal

* + Semaphores – Truy xuất tài nguyên liên tục từ các task khác nhau

[osSemaphoreNew](https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__SemaphoreMgmt.html#ga2a39806ace781a0008a4374ca701b14a)(); //Tạo Semaphore

[osSemaphoreAcquire](https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__SemaphoreMgmt.html#ga7e94c8b242a0c81f2cc79ec22895c87b)();// Task Yêu cầu tài nguyên

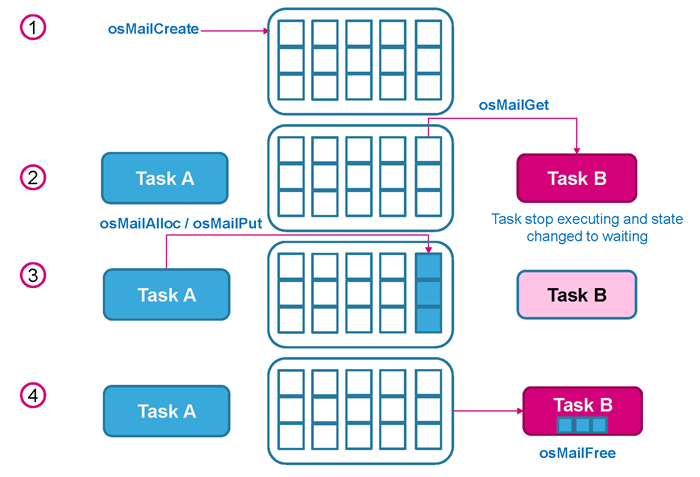
[osSemaphoreRelease](https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__SemaphoreMgmt.html#ga0abcee1b5449d7a6928fb9248c690bb6)();// Task Tạo thêm tài nguyên

* + Mutex – Đồng bộ hóa truy cập tài nguyên sử dụng Mutual Exclusion

[osMutexNew](https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__MutexMgmt.html#gab90920022ab944296821368ef6bb52f8)(); //Tạo mới mutex

[osMutexAcquire](https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__MutexMgmt.html#gabc54686ea0fc281823b1763422d2a924)(); //Yêu cầu truy cập vào vùng tranh chấp

[osMutexRelease](https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__MutexMgmt.html#gaea629705703580ff58776bf73c8db915)(); //Rời khỏi vùng tranh chấp

* Cơ chế giao tiếp:
  + Message queue – Trao đổi tin nhắn giữa các task trong hoạt động giống như FIFO
  + Mail queue – Trao đổi dữ liệu giữa các task sử dụng hằng đợi của khối bộ nhớ
  1. Công việc tuần kế tiếp:
* Sử dụng RTOS cho project