**BÁO CÁO TUẦN 11 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.”

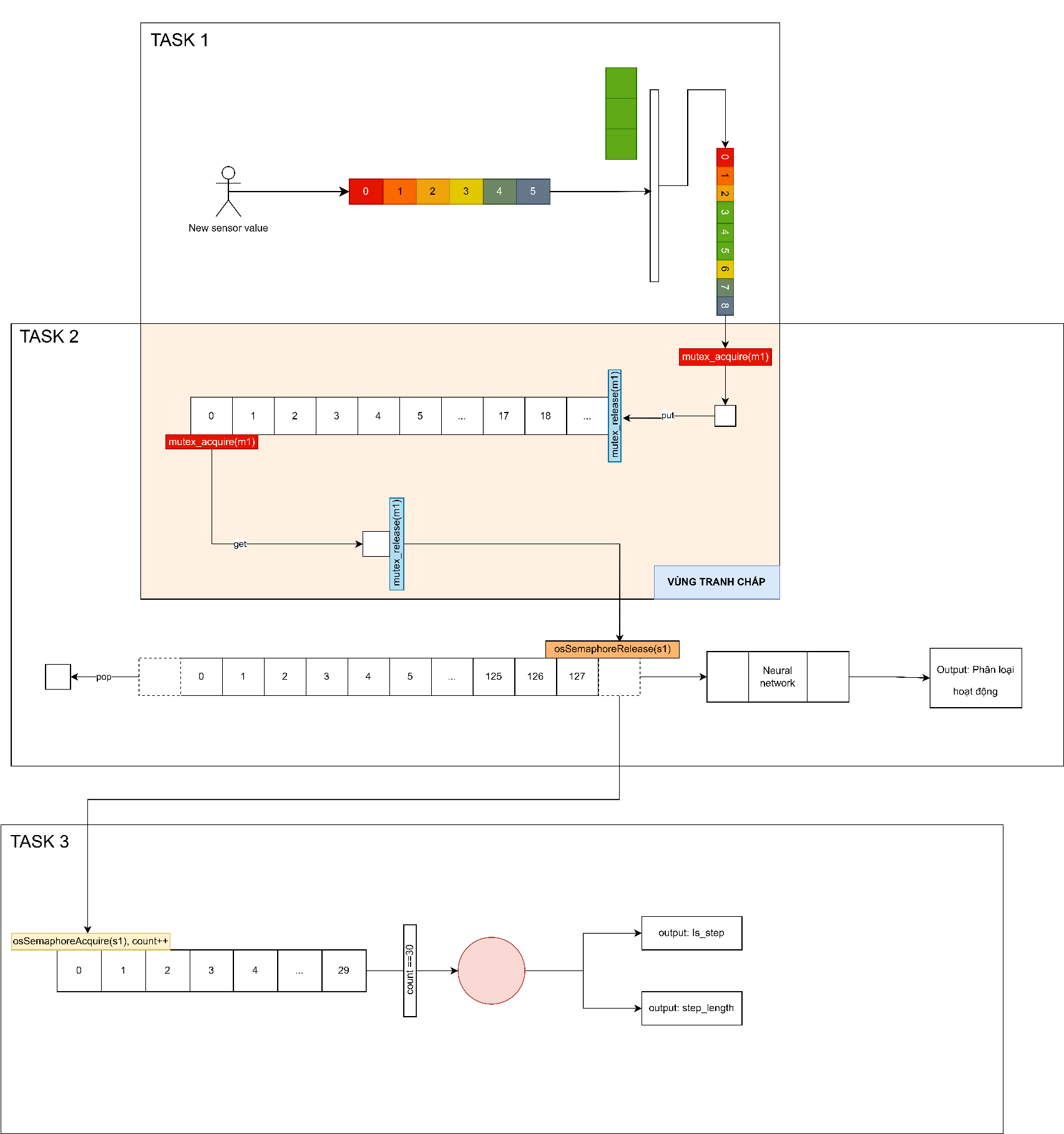
Thành viên thực hiện:

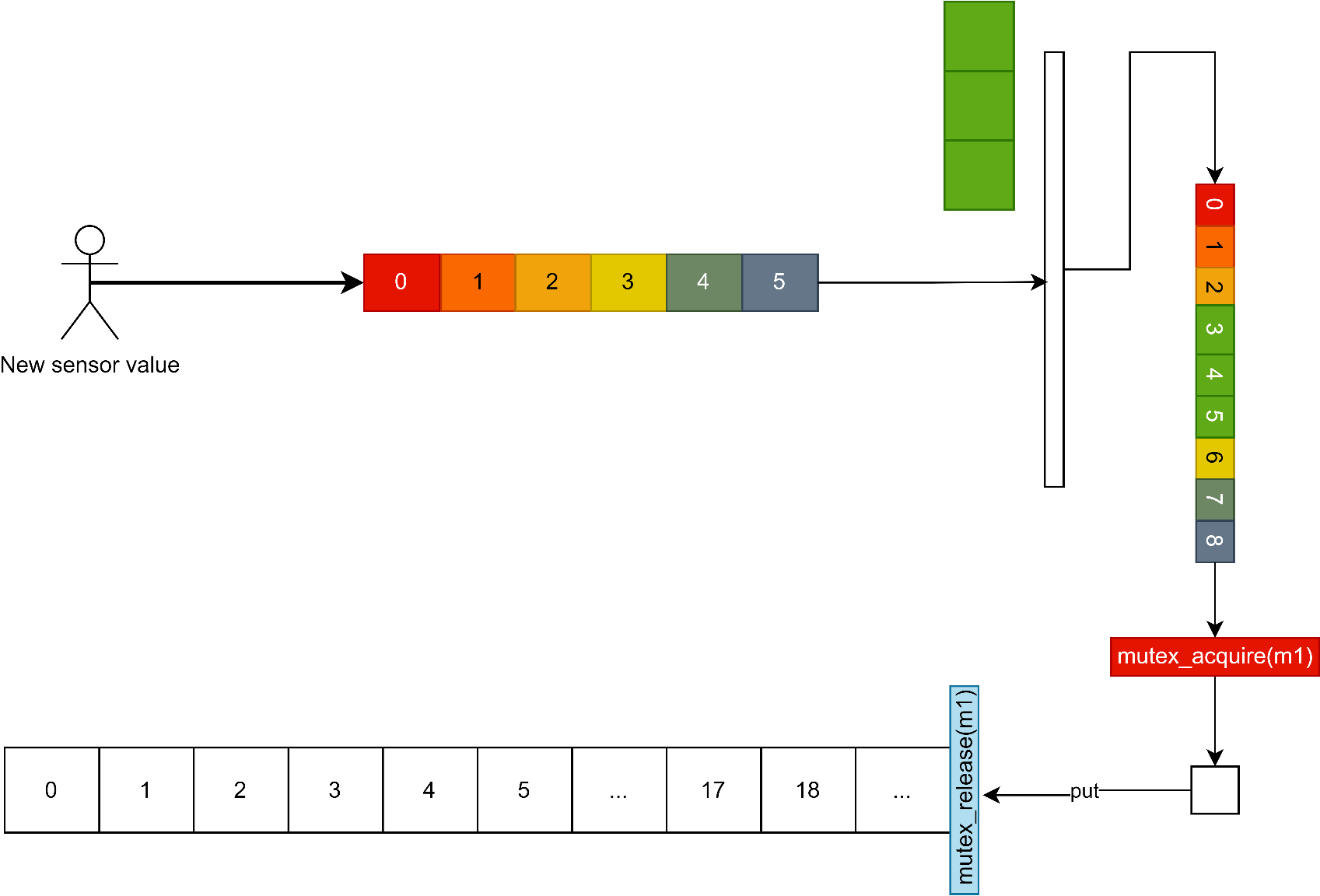
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | MSSV | ĐIỆN THOẠI | EMAIL |
| 1 | Nguyễn Ngọc Minh | 19520165 | 0585115056 | 19520165@gm.uit.edu.vn |

1. Nội dụng thực hiện:

Tối ưu hóa hệ thống.

1. Báo cáo quá trình thực hiện:
   1. Tổng quan hệ thống:

* Bao gồm 3 tác vụ chính:
  + Task 1: Tác vụ lấy data từ cảm biến và xử lý, sau đó cho vào hàng đợi dữ liệu chung (message queue).
  + Task 2: Tác vụ lấy toàn bộ data từ hàng đợi chung message queue, đưa toàn bộ vào hàng đợi data\_in trước khi chạy neural lấy kết quả dự đoán.
  + Task 3: Tác vụ detect bước chân và đo khoảng cách. Chỉ chạy khi có 30 dữ liệu mới.
* Cơ chế giao tiếp task: sử dụng message queue để trao đổi dữ liệu (task 1 & 2)
* Cơ chế đồng bộ:
  + Bảo vệ vùng tranh chấp (message queue): Mutex giữa task 1 & 2
  + Yêu cầu 30 dữ liệu mới để chạy task 3: semaphore. Producer (task2) & Consumer (task3)
  1. Mô hình hệ thống:
  2. Mô tả chi tiết:

Task1: Thu thập data từ sensor

* Lấy data từ sensor sau đó xử lý, yêu cầu đi vào vùng trang chấp, put dữ liệu vào message queue, thoát khỏi vùng tranh chấp.

**void** **StartTask1**(**void** \*argument) {

**for** (;;) {

MPU\_Read\_Data(); //đọc dữ liệu

MutexAcquire(&m1); //yêu cầu vào vùng tranh chấp

osMessageQueuePut(ValueQueueHandle, &mpudata, NULL, 0); //đưa data vào vùng nhớ chia sẻ

MutexRelease(&Neural\_permitHandle); //thoát khỏi vùng tranh chấp

osDelay(10); //delay

}

}

Ảnh có chứa bàn

Mô tả được tạo tự độngTask 2: Dự đoán loại hoạt động của người dùng

* Yêu cầu đi vào vùng chanh chấp, get dữ liệu từ massage queue, (1), cho vào hàng đợi dữ liệu của neural, chạy neural.
* (1) Thoát khỏi vùng tranh chấp, tăng semaphore lên 1 đơn vị.

**void** **StartTask02**(**void** \*argument) {

**for** (;;) {

MutexAcquire(&Neural\_permitHandle); // yêu cầu vào vùng tranh chấp

uint32\_t count = osMessageQueueGetCount(ValueQueueHandle); //đếm lương data trong vùng nhớ chia sẻ

**memmove**(in\_data, in\_data + count, (128 - count)); //dịch mảng

**for** (**int** i = 128 - count; i < 128; i++) {

osMessageQueueGet(ValueQueueHandle, &data, NULL, 1);//Lấy data trong vùng nhớ chia sẻ

\*(in\_data + i) = data; //đưa data mới vào

osSemaphoreRelease(s1); //Thêm 1 tài nguyên

}

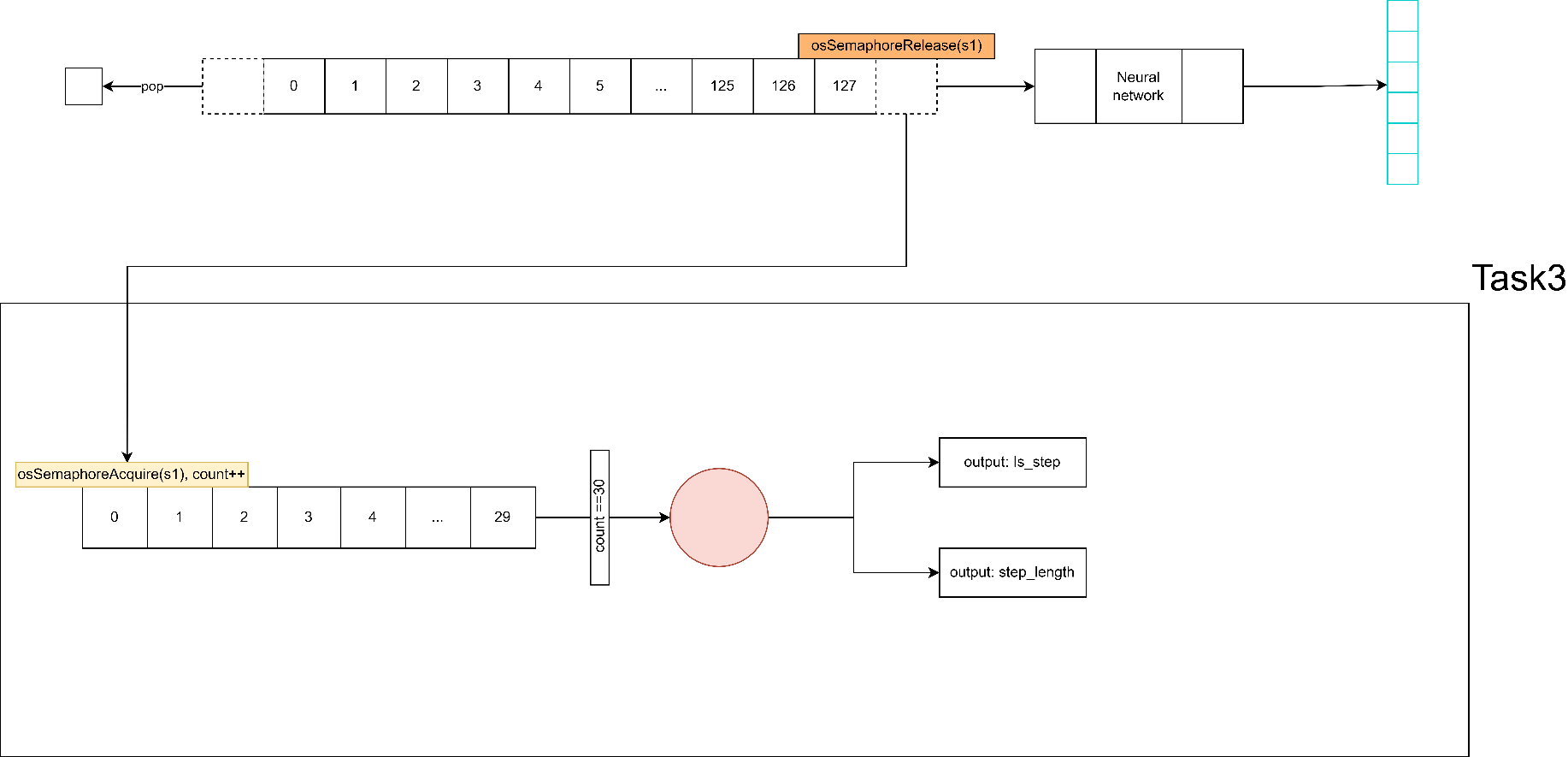
MutexRelease(&Neural\_permitHandle); //Thoát khỏi vùng tranh chấp

nbatch = ai\_har\_run(har\_model, &ai\_input[0], &ai\_output[0]); //chạy neural

osDelay(500); //delay

}

}

Task 3: Phát hiện bước chân và tính độ dài

* Yêu cầu sử dụng 1 tài nguyên từ semaphore, đưa data từ hàng đợi dữ liệu ở task 2 vào, nếu lấy được 30 tài nguyên thì thực hiện giải thuật.

**void** **StartTask03**(**void** \*argument) {

**for** (;;) {

**for**(**int** i=0; i<30; i++){ //Yêu cầu 30 tài nguyên

osSemaphoreAcquire(ValueReadyHandle);

}

**for** (**int** i = 0; i < 30; i++) {

window\_data[i] = MPU\_Mag(in\_data[i]);

**if** (max\_mag < window\_data[i]) max\_mag = window\_data[i];

}

**if** (max\_mag > MAG\_THERSHOLD) {

step\_count++;

step\_length += calculateStepLength(window\_data);

}

}

}

1. Công việc tuần kế tiếp:

* Chạy thực tế, kiểm tra, đánh giá.