Robot

Programul gestioneaza concurent activitatea a patru roti ale unui robot, fiecare cu viteza si directia sa si momentul de timp al masurarii, pentru a afisa datele in Serial Monitor.

La inceput, am definit o constanta pentru numarul de roti si o structura “Roata” care contine campurile mentionate si o metoda pentru generarea aleatorie a vitezei intre -10 si 10, iar directia este calculata pe baza semnului vitezei. Apoi, am declarat un mutex care va fi utilizat pentru a sincroniza accesul la datele rotilor.

```
#include <Arduino_FreeRTOS.h>
#include <semphr.h>

const int nr_roti = 4;
struct Roata {
  float viteza = 0;
  int directie = 0;
  long timpMasurare = 0;
  void generareViteza() {
    viteza = random(-1000, 1001) / 100.0;
    directie = (viteza != 0) ? (viteza / abs(viteza)) : 0;
  }
} roti[nr_roti];

SemaphoreHandle_t mutex;
```

“TaskRoti” genereaza periodic viteza si directia pentru fiecare roata. Utilizarea mutexului asigura ca accesul la structura roti este exclusiv, prevenind conflictele de date.

```
void TaskRoti(void *pvParameters) {
  int id = (int)pvParameters;
  while (1) {
    if (xSemaphoreTake(mutex, portMAX_DELAY) == pdTRUE) {
      roti[id].generareViteza();
      roti[id].timpMasurare = millis();
      xSemaphoreGive(mutex);
    }
    vTaskDelay(pdMS_TO_TICKS(1000));
  }
}
```

Acest task afiseaza datele curente pentru fiecare roata la fiecare 3 secunde. Din nou, se utilizeaza mutexul pentru a evita conflictele.

```
29  void TaskAfisare(void *pvParameters) {
30    while (1) {
31      if (xSemaphoreTake(mutex, portMAX_DELAY) == pdTRUE) {
32        for (int i = 0; i < nr_roti; i++) {
33          Serial.print("Roata ");
34          Serial.print(i);
35          Serial.print(": Viteza: ");
36          Serial.print(roti[i].viteza);
37          Serial.print(", Directie: ");
38          Serial.print(roti[i].directie);
39          Serial.print(", Timp: ");
40          Serial.print(roti[i].timpMasurare);
41          Serial.println("ms");
42        }
43        Serial.println();
44        xSemaphoreGive(mutex);
45      }
46      vTaskDelay(pdMS_TO_TICKS(3000));
47    }
48  }
```

```
Roata 0: Viteza: -3.04, Directie: -1, Timp: 2976ms
Roata 1: Viteza: 9.40, Directie: 1, Timp: 2976ms
Roata 2: Viteza: -3.88, Directie: -1, Timp: 2976ms
Roata 3: Viteza: -9.74, Directie: -1, Timp: 2977ms
```

In final, in partea de setup se creeaza mutexul si task-uri pentru fiecare roata, avand prioritate 2, si un task pentru afisare cu prioritate 1.

```
50  void setup() {
51    Serial.begin(9600);
52    mutex = xSemaphoreCreateMutex();
53
54    for (int i = 0; i < nr_roti; i++) {
55      xTaskCreate(TaskRoti, "TaskRoti", 128, (void *)i, 2, NULL);
56    }
57    xTaskCreate(TaskAfisare, "TaskAfisare", 128, NULL, 1, NULL);
58  }
59
60  void loop() {}
```