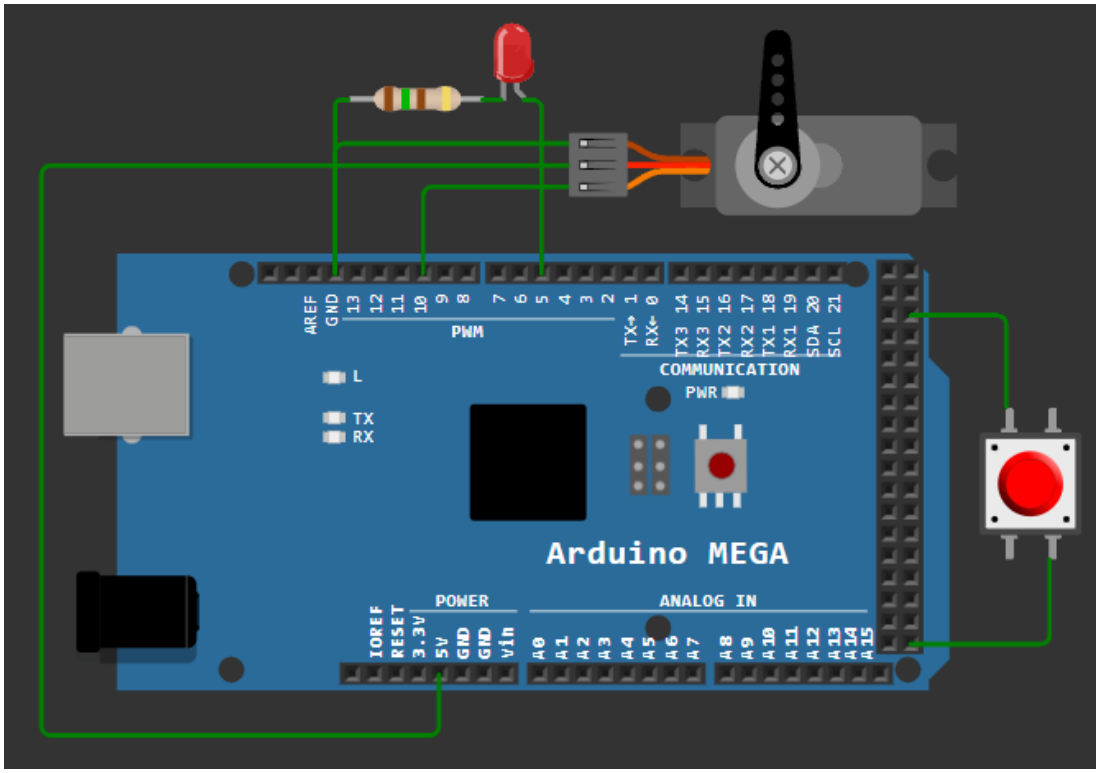
# Railway Crossing

Programul gestioneaza concurent activitatea unei bariere automate care raspunde la sosirea unui tren, controland inchiderea portii, iluminarea LED-ului de traversare si afisarea informatiilor in Serial Monitor.




```
Train detected
Gate closing
Gate closed
Arrival time: 3807ms
Crossing time: 3249ms
Gate opening
Gate opened
```

La inceput, sunt definite constantele pentru pini, intervalele de timp pentru sosire si traversare, precum si durata necesara pentru deschiderea sau inchiderea barierei. Servomecanismul pentru poarta si semafoarele sunt declarate pentru a sincroniza cele doua task-uri: unul care detecteaza sosirea trenului si inchide bariera, si altul care gestioneaza traversarea si deschiderea portii.

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

const int arrivalButton = 25, gatePin = 10, crossingLED = 5,
          arrivalTimeMin = 3000, arrivalTimeMax = 5000,
          crossingTimeMin = 2000, crossingTimeMax = 4000, gateTime = 2000;

Servo gateServo;
SemaphoreHandle_t arrivalSemaphore, crossingSemaphore;
```

“arrivalTask” detecteaza sosirea trenului (apasarea butonului), inchide bariera si genereaza un timp aleatoriu de sosire intre cel minim (cel mai rapid tren) si maxim (cel mai incet tren). Inchiderea portii este realizata treptat pentru a simula miscarea reala a unei bariere, iar utilizarea semaforului binar “arrivalSemaphore” asigura exclusivitate, doar un tren putand trece in acelasi timp.

```
void arrivalTask(void *pvParameters) {
  while (1) {
    xSemaphoreTake(arrivalSemaphore, portMAX_DELAY);
    if (digitalRead(arrivalButton) == LOW) {
      Serial.println("Train detected");
      vTaskDelay(pdMS_TO_TICKS(arrivalTimeMin - gateTime));
      Serial.println("Gate closing");
      for (int angle = 1; angle <= 90; angle++) {
        gateServo.write(angle);
        vTaskDelay(pdMS_TO_TICKS(gateTime / 90));
      }
      Serial.println("Gate closed");
      int arrivalTime = random(arrivalTimeMin, arrivalTimeMax);
      vTaskDelay(pdMS_TO_TICKS(arrivalTime - arrivalTimeMin));
      Serial.print("Arrival time: ");
      Serial.print(arrivalTime);
      Serial.println("ms");
      xSemaphoreGive(crossingSemaphore);
    }
    else {
      xSemaphoreGive(arrivalSemaphore);
    }
    vTaskDelay(pdMS_TO_TICKS(100));
  }
}
```

“crossingTask” gestioneaza traversarea trenului, iluminarea LED-ului de avertizare, si deschiderea portii la final. LED-ul este activat pentru o perioada aleatorie de timp (intre cel minim si maxim), corespunzatoare

duratei de traversare, iar la finalul traversarii, poarta se deschide treptat, utilizand un ciclu invers fata de inchidere. In acest timp, semaforul “crossingSemaphore” sincronizeaza accesul la task, asigurandu-se ca se executa dupa “arrivalTask”.

```
37  void crossingTask(void *pvParameters) {
38    while (1) {
39      xSemaphoreTake(crossingSemaphore, portMAX_DELAY);
40      int crossingTime = random(crossingTimeMin, crossingTimeMax);
41      digitalWrite(crossingLED, HIGH);
42      vTaskDelay(pdMS_TO_TICKS(crossingTime));
43      digitalWrite(crossingLED, LOW);
44      Serial.print("Crossing time: ");
45      Serial.print(crossingTime);
46      Serial.println("ms");
47
48      gateServo.write(0);
49      Serial.println("Gate opening");
50      for (int angle = 89; angle >= 0; angle--) {
51        gateServo.write(angle);
52        vTaskDelay(pdMS_TO_TICKS(gateTime / 90));
53      }
54      Serial.println("Gate opened");
55      xSemaphoreGive(arrivalSemaphore);
56      vTaskDelay(pdMS_TO_TICKS(100));
57    }
58  }
```

In partea de setup, se initializeaza semafoarele si servomecanismul, iar task-urile pentru sosire si traversare sunt initializate cu stack-uri de dimensiuni corespunzatoare si aceeasi prioritate.

```
60  void setup() {
61    Serial.begin(9600);
62    pinMode(arrivalButton, INPUT_PULLUP);
63    pinMode(crossingLED, OUTPUT);
64    gateServo.attach(gatePin);
65    gateServo.write(0);
66
67    arrivalSemaphore = xSemaphoreCreateBinary();
68    crossingSemaphore = xSemaphoreCreateBinary();
69    xSemaphoreGive(arrivalSemaphore);
70
71    xTaskCreate(arrivalTask, "Arrival", 128, NULL, 1, NULL);
72    xTaskCreate(crossingTask, "Crossing", 128, NULL, 1, NULL);
73  }
74
75  void loop() {}
```