Ejercicio de planificación
Sistemas de Tiempo Real
Cuarto curso de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

El objetivo de la práctica 5 es programar un planificador cíclico para un simulador robótico simplificado.

Planificador cíclico

Se dispone de un robot como el de la figura 1, dotado de 3 sensores de proximidad y de dos motores. El robot se mueve en un mundo que se diseñará mediante un fichero de texto en el que se indican con "X" las casillas ocupadas por un obstáculo. El fichero tiene unas dimensiones de 50 columnas x 20 filas casillas y se supone circular (si se el robot sale por la derecha vuelve por la izquierda de la misma fila, si sale por arriba aparece por abajo en la misma columna, etc.). El robot parte de la casilla superior izquierda, orientado hacia abajo. El robot puede estar orientado hacia abajo (S), arriba (N), la izquierda (O) o la derecha (E).

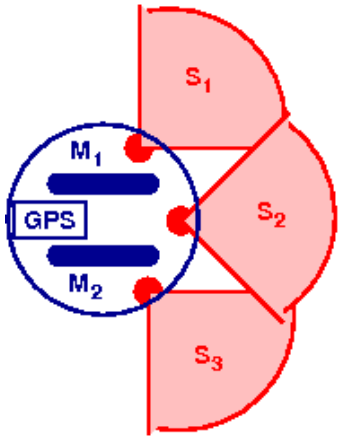

Figura 1: Robot simulado

La función `int leerSensores(t_sensores *sens)` permitirá leer los sensores. Los valores de estos sensores se refrescan a 2 Hz. El sensor devuelve un entero en $[0-5]$ indicando el número de casillas libres en dirección . El tiempo de cómputo necesario para leer este sensor supondremos que es de 10ms.

También disponemos de un sensor GPS que se actualiza a 1Hz y cuyo valor puede consultarse mediante `int leergps(t_posicion &posactual)`. Este sensor devuelve la posición (casilla y orientación) en la que se encuentra actualmente el robot. Inicialmente será la $(0, 0, S)$. El tiempo de cómputo de este sensor se supone de 50ms.

El robot dispone de dos motores que pueden estar encendidos o apagados. Si los dos están encendidos avanza una casilla en la dirección de su orientación actual. Si sólo está encendido el motor izquierdo gira sobre si mismo hacia la derecha y viceversa, si lo está el derecho hacia la izquierda. Cada motor debe recibir una consigna cada 2 segundos. Si no la recibe se apaga. La velocidad de avance si están encendidos los dos motores es de una casilla por segundo, y el giro de un cuarto de vuelta también por segundo. Supondremos que el tiempo de computo del estado de los motores es de 100ms.

La casilla destino del robot se indicará mediante la línea de comandos.

Se pide:

- Calcular una planificación cíclica factible para el conjunto de procesos y programar dicho planificador.
- Implementar el simulador que devuelva los valores adecuados cuando sean leídos desde las tareas del planificador.

Se espera que el programa principal tenga 2 threads: 1 para el planificador cíclico y otro para el simulador del sistema.

A continuación tenéis un fragmento de código para implementar el simulador:

```
// Tamaño del mapa
#define FIL 6
#define COL 10
// Rango de los sensores
#define RANGO 5

// Mapa
char mapa[FIL][COL+1];

// Robot
typedef enum orientacion {N,E,S,O} orientacion;
char simbolo[4]={'^','>','Y','<'};
typedef struct t_posicion {int x,y; orientacion or;} t_posicion;
typedef struct t_sensores {int s1,s2,s3;} t_sensores;
typedef struct t_robot {t_posicion pos; int M1,M2;} t_robot;

// Posición del robot:
//inicialmente (0,0), hacia abajo y con motores encendidos
t_robot robot = {0,0,N,1,1};


// Simulador
void* simulador(void *arg){
  while(1){
```

```
    // El simulador realiza la simulación aproximadamente cada segundo
    // que es ritmo al que hay que actualizar los motores
    usleep(1000000);

    pthread_mutex_lock(&semaforo);

    // Borro el robot
    mapa[robot.pos.x][robot.pos.y]=' ';

    // Calculo la nueva posición en función de motores y orientación
    switch (robot.M1){
    case 1: //motor1 encendido
      if (robot.M2) // motor2 encendido: avanzo
        switch (robot.pos.or) {
        case N: // hacia arriba
          robot.pos.x = (robot.pos.x-1 < 0) ? FIL-1: robot.pos.x-1;
          break;
        case S: // hacia abajo
          robot.pos.x = (robot.pos.x+1>= FIL) ? 0 : robot.pos.x+1;
          break;
        case E: // hacia la derecha
          robot.pos.y = (robot.pos.y+1 >= COL) ? 0 : robot.pos.y+1 ;
          break;
        case O: // hacia la izquierda
          robot.pos.y = (robot.pos.y-1 < 0) ? COL-1 : robot.pos.y-1 ;
        } // switch (robot.or)
      else { //motor2 apagado: giro horario
        robot.pos.or = (robot.pos.or+1)%4;
      }
      break;
    case 0: //motor1 apagado
      if (robot.M2){ // motor2 encendido: giro antihorario
        robot.pos.or = (robot.pos.or-1<0) ? 3: robot.pos.or-1;
      }
      break;
    } // switch (robot.M1)
    // Pongo el robot en el mapa en su posición y con su orientación
    mapa[robot.pos.x][robot.pos.y]=simbolo[robot.pos.or];
    pthread_mutex_unlock(&semaforo);

    pintaMapa(); // Lo vuelvo a pintar
  } // while(1)
}
```