Ejercicio Robot con brazo - Solución

```
// Driver lector
/*
- lector:
    + operaciones:
        * read: devuelve el código de barras leído
    + registros:
        * CODIGO: codigo de barras
*/

int driver_read(char* buffer){
    int codigo = IN(CODIGO)
    copy_to_user(buffer, &codigo, sizeof(codigo));
    return sizeof(codigo);
}
```

```
// Driver brazo
/*
- brazo:
    + operaciones:
        * write: indica al brazo hacia dónde moverse, y qué hacer con la mano
    + registros:
        * POSICION: 0,...,9 posición en la estantería
        * MANO: ABRIR, CERRAR: indica qué debe hacer cuando llegue a la posición
        * BRAZO: ESTIRAR, CONTRAER: indica si debe estirar o contraer el brazo al llegar a la posición
*/
enum mano {ABRIR, CERRAR};
enum brazo {ESTIRAR, CONTRAER};

struct brazo_operacion {
    int posicion;
    enum mano mano;
    enum brazo brazo;
}

int driver_write(char* buffer){
    brazo_operacion p;
    copy_from_user(buffer, &p, sizeof(p));
    OUT(POSICION, p.posicion);
    OUT(MANO, p.mano);
    OUT(BRAZO, p.brazo);
    return 3*sizeof(int);
}
```

```
// Driver movimiento
/*
- movimiento:
    + operaciones:
        * write: indica que debe moverse, retorna cuando haya llegado a destino
    + registros:
        * DEST_X: indica la posición X
        * DEST_Y: indica la posición Y
        * CONTROL: START
    + interrupciones: se levanta la interrupción POS_FIN cuando se llegue a destino.
*/

struct ubicacion {
    int x;
    int y;
}

semaphore sem;
```

```
handler() {
    sem.signal();
}

driver_init() {
    sem_init(sem, 0);
    map_irq(POS_FIN, handler);
}

driver_close() {
    sem_destroy(sem);
    free_irq(POS_FIN);
}

int driver_write(char* buffer) {
    ubicacion u;
    copy_from_user(buffer, &ubicacion, sizeof(ubicacion));

    OUT(DEST_X, u.x);
    OUT(DEST_Y, u.y);
    OUT(CONTROL, START);

    sem.wait(); // Espera a que llegue la interrupción

    return sizeof(ubicacion);
}
// Programa de usuario

struct paquete {
    int x;
    int y;
    int codigo;
}

struct brazo_operacion {
    int posicion;
    enum mano {ABRIR, CERRAR};
    enum brazo {ESTIRAR, CONTRAER};
};

// Devuelve el código del objeto i
int tomar_objeto(int i) {
    op = {posicion=i, mano=ABRIR, brazo=ESTIRAR};
    write(brazo, op, sizeof(op));
    op = {posicion=i, mano=CERRAR, brazo=ESTIRAR};
    write(brazo, op, sizeof(op));
    op = {posicion=i, mano=CERRAR, brazo=CONTRAER};
    write(brazo, op, sizeof(op));

    int codigo;
    read(lector, &codigo, sizeof(codigo));
    return codigo;
}

void dejar_objeto(int i) {
    op = {posicion=i, mano=CERRAR, brazo=ESTIRAR};
    write(brazo, op, sizeof(op));
    op = {posicion=i, mano=ABRIR, brazo=ESTIRAR};
    write(brazo, op, sizeof(op));
    op = {posicion=i, mano=ABRIR, brazo=CONTRAER};
    write(brazo, op, sizeof(op));
```

```
}

void main() {
    int lector = open("/dev/lector");
    int brazo = open("/dev/brazo");
    int movimiento = open("/dev/movimiento");

    while(1) {
        paquete p = siguiente_paquete();
        write(movimiento, {p.x, p.y}, 2*sizeof(int)); // se bloquea hasta que llegue

        // llegó a la estantería
        for(int i=0; i<10; i++) {
            // Tomar el objeto de la estantería y ver el código
            int codigo = tomar_objeto(i);
            if(codigo==p.codigo) {
                break;
            }
            dejar_objeto(i);
        }

        // Al salir del for, el objeto está en la mano del robot
        // Moverse hasta la cinta de entrega, y dejar el paquete
        write(movimiento, {0, 0}, 2*sizeof(int)); // se bloquea hasta que llegue
        dejar_objeto(0);
    }
}
```