Lab#6: Mini assembly line

Leonardo Fusser (1946995)

Objectives:

- Create a thread to automize a mini assembly line.
- Work in teams.

Material: see previous lab.

Warning: to avoid mechanical damage when testing, only soft object can be manipulated by the robot.

To hand in: see previous lab.

File system structure:

See the following hierarchy. You must populate the files in **orange.**

- **multithread_mover4_v6\pdcurses_test\mover4_v6**
  - can.c
  - adc.c
  - task_controller.c
  - log
  - state
  - ncurses_init.c
  - **sol_mover4**
    - build
    - mover4.c
    - taskKeybd.c
    - **taskAuto.c**
    - taskDisplay.c
  - **header**
    - can.h
    - adc.h
    - kinematic.h
    - **public.h**
    - task_controller.h
    - ncurses_init.h
  - **demo**
    - exec
  - **excel_sim**
    - animation_v2
    - export

Requirements:

In a team of 2 students, you must setup a mini assembly line.

A vial stands on a box. A detector must detect the presence or absence of the vial.

One robot must grab the vial (if present) and move it on another box.

The second robot must detect the presence of the vial and move it back to the first box.

Safety measures:

For safety reason, one student must always stay close to the emergency stop switch.

Hint: you must give the robot a trajectory, so it does not go straight from one point to another and hit boxes.

Lab Work

ADC reading

The distance sensor is connected to the ADC at pin 5.

To read the sensor you must use the following C function:

```
int readADC(unsigned int pin)
```

Example of use:

```
int sensor;
sensor = readADC(5); // reads the sensor at pin 5
```

TaskAuto thread:

You must create a thread named `taskAuto`.

When in auto mode, it generates a specific algorithm for the mini assembly line inside an infinite loop.

To set the SP angles, you must use the `set_sp_angles` or `set_sp_angle()` APIs.

To set the speed, you must use `set_speed_max()` API.

Example to set the speed of all 4 joints:

```
set_speed_max(SPEED_MED, SPEED_MAX, SPEED_MAX, SPEED_SLOW);
```

Your other threads must be modified so when the ‘n’ key is pressed, the system runs the auto thread and when the ‘j’ key is pressed, the system stops running the auto thread.

~~When in auto threads all jog keys must be disabled.~~

Also, switching from one mode to another should be almost immediate: response time should be less than one second.

Also, the menu should be modified accordingly:

Main menu display area **Line 0 to Line 9**

```
*************************************************************************
* Use the following keys to jog the joints or grip                      *
*                                                                       *
*    q         w         e        r        t                           *
*  Joint1    Joint2    Joint3   Joint4   Gripper                        *
*    a         s         d        f        g                           *
*                                                                       *
*  Use also the following keys:                                         *
*  Exit: x - Jog mode: j - Auto mode: n                                 *
*************************************************************************
```

Note: make sure the task gives slack time to the system.

Test the thread for the mini assembly line.

When you try to return to jog mode, is it responsive or is there a latency? In other words, when you press jog, can you jog right away, or do you need to wait?

- When the user selects to return back to jog mode (manual jogging), the system is very responsive and the user can manually jog the robot right away. There is almost no delay when switching back to jog mode. Furthermore, there is also no delay switching from jog mode to auto mode.

  This is because the auto task does not hog the system. Steps were taken to ensure that the system has sufficient slack time to keep the system responsive. Steps were also taken to ensure that the auto task cannot be created multiple times if the user was to press the “n” key multiple times.

Give a demo to the teacher.

At the end of the demo, you must erase all your source files from BBB:

mover4.c, taskKeybd.c, taskDisplay.c, taskAuto.c and public.h

Annex

APIs

```
Set the speed of the joints.
The first parameter is the base, the second parameter is the shoulder, the third
is the elbow and the last parameter is the wrist.
The list of possible speeds are:SPEED_MAX, SPEED_MED and SPEED_SLOW.
void set_speed_max(double base, double shld, double elbow, double wrist);
```