Inverted pendulum implemented on a Lego Mindstorms NXT

Systèmes Temps Réel

INPT/ENSEEIHT - 3SN E+L

November 2020

Plan

Plan

1. Prepare your workspace
2. The Inverted Pendulum: A Bit of Theory
3. Implementation of the inverted pendulum on the Lego NXT robot
4. To do

Your workspace at ENSEEIHT

The steps:

1. Add the Trampoline-OSEK environment to your path:

   > `source /mnt/n7fs/nxt/nxt.sh`

   - Test if it's ok: type `goil --help`
     You should have the list of options of the command `goil`.
   - To avoid doing the manipulation each time, add the previous command at the end of your file `.bashrc`.

Your workspace on your computer

This solution uses the `docker` tool: `https://www.docker.com/`

1. Download the archive containing the Dockerfile and script files
   - `http://moodle-n7.inp-toulouse.fr/mod/resource/view.php?id=59085`
2. Build the image containing Trampoline-OSEK for the NXT:

   ```
   > docker build -t trampoline .
   ```
3. Run the Docker container associated with the created image:
   1. Place yourself in the directory in which you wish to work
   2. Type :

```
> docker run -ti \
--mount src=‘pwd‘,dst=/mnt/shared,type=bind trampoline
```

Note :

The container runs the development tools for Trampoline. The directory `/mn/shared` is connected to the current directory of the host machine. So all files inserted in this directory will be seen in the directory `/mnt/shared`.

Les fichiers fournis

1. Create the working directory of your project:

   `> nxt_create nxtSegway`

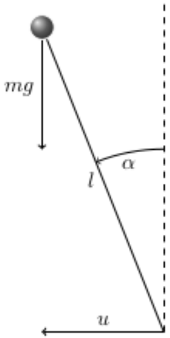
   This command creates the files `nxtSegway.c` and `nxtSegway.oil`.
2. Delete the file `nxtSegway.c`.
3. Download the archive pendulumNXT.tar and unzip it in the working directory nxtSegway:
   - `http://deptr.enseeiht.fr/supports/ermont/support/3TR/penduleNXT.tar`

Files which are present:

- `nxt_config.h`: Lego NXT robot configuration parameters
- `tools.c` and `tools.h`: toolbox
- `nxtSegway.c` : Lego robot control application (to be completed)

Plan

The Inverted Pendulum: A Bit of Theory I



- A bit of physics, model of the pendulum controlled without friction :

$$ml^2\ddot{\alpha}(t) + mlg\sin(\alpha(t)) = u(t)$$

- The state of the system is written $x(t) = (x_1(t), x_2(t)) = (\alpha(t), \dot{\alpha}(t))$
- The resulting differential system is written :

$$\begin{cases} \dot{x_1}(t) & = & x_2(t) \\ \dot{x_2}(t) & = & -\frac{g}{l}\sin(x_1(t)) + \frac{u(t)}{ml^2} \\ x_1(0) & = & \alpha_0 \\ x_2(0) & = & \dot{\alpha}_0 \end{cases}$$

The Inverted Pendulum: A Bit of Theory II

- The system becomes stable and vertical by applying the $u(t)$ control as follows :

$$u(t) = u_e + K(x(t) - x_e)$$

  where $u_e = 0$ and $x_e = 0$ are the equilibrium controls and positions, K is the pole placement matrix.

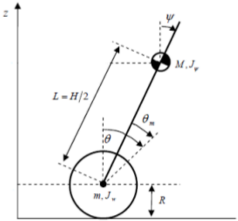
- It is possible to apply a command by adding it to the control :

$$z(t) = Ki \int (x(t) - x_o)dt$$

  where x_o is the objective state.

And for the Lego robot ...

- Lego robot model



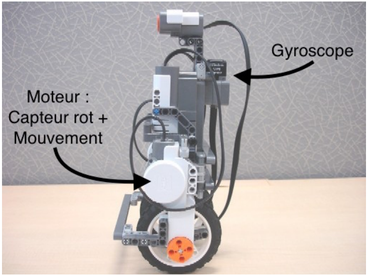
- System state variable

$$x(t) = (\theta, \psi, \dot{\theta}, \dot{\psi})$$

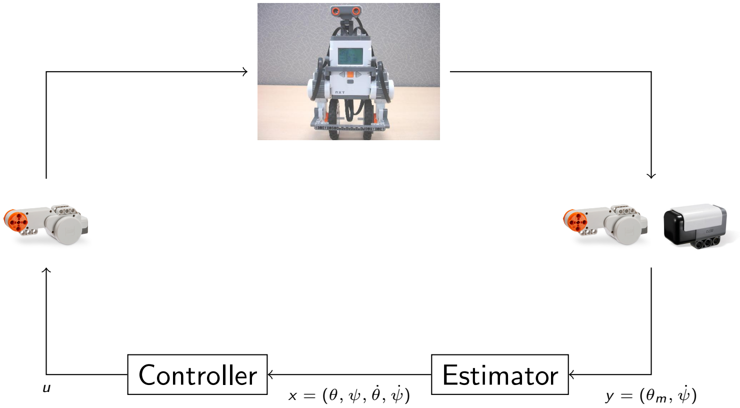
Plan

Implementation of the inverted pendulum on the Lego NXT robot

- Objectives: understand the implementation of the inverted pendulum on the physical system: the NXT robot.
- The Lego Mindstorm NXT robot in inverted pendulum configuration :



The Inverted Pendulum System



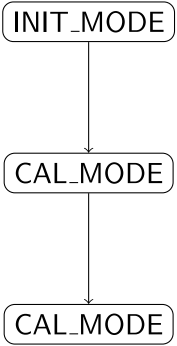
- y : Observed state of the pendulum
- x : State of the pendulum
- u : Control applied to engines

The tools at your disposal

- OSEK real-time system (Trampoline)
  - performs tasks on a periodic basis
    - the tasks are executed again after a predefined period of time
  - Here, 2 tasks to perform: pendule (to be completed), affichage (provided)
- The toolbox
  - The affichage task displays the values of θ and ψ provided by the robot's motors and gyroscope respectively, as well as the current state of the robot (value of x)
  - The functions :
    - `float getGyro(int gyro_offset)` : output the value of $\dot{\psi}$.
    - `float getMotorAngle()` : outputs the rotation of the motors (i.e. θ_m)
    - `void nxt_motors_set_command(float u)` : applies the command to the motors of the Lego NXT robot
    - `float delta_t()` : provides the step of discretization

The Pendule Task

- The role of this task is to control the vertical position of the Lego robot.
- 3 states :
  - INIT_MODE: initializes the system (reset variables, . . .)
  - CAL_MODE :
    - calibrates the gyroscope in the vertical position
    - the robot emits a sound when performed
  - CONTROLE_MODE :
    - mode of operation of the inverted pendulum
    - that's the part to complete



Plan

Step 1: Implementation of the application

1. Create the function `estimator` that provides the current state of the system $x = (\theta, \psi, \dot{\theta}, \dot{\psi})$ from the observed state $y = (\theta_m, \dot{\psi})$ and the discretization step, where $\theta = \theta_m + \psi$
2. Create the function `controller` that calculates the u command to be applied to the system based on the current state of the system.
3. Complete the CONTROL status of the task controlling the robot motion.

Integration/discrete derivative (Euler method)

- Integration : $f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i)f'(x_i)$
- Derivative: $f'(x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$

where $x_{i+1} - x_i$ is the discretization step.

Values of K and Ki

K=(0.6700,19.9053,1.0747,1.9614)
Ki=0.2534

Step 2: OSEK configuration and compilation

1. Modify the file `nxtSegway.oil` in order to configure the tasks `pendulum` (period = 4 ms) and `display`. (period = 500 ms) The system timer has a period of 1 ms.
2. Compile and execute the program on the Lego robot.
   - Check that there is no Makefile already created, otherwise delete it:
     - `make clean`
     - `rm Makefile`
   - Generate a Makefile from the file `.oil` present in the archive : `nxt_goil nxtSegway.oil`
   - Compilation : `make`
   - Downloading on the robot : `nxt_send nxtSegway_exe.rxe`

Step 3: Obstacle detection

1. Write a new task that allows the acquisition of a distance using the ultrasonic sensor and provides a command to the robot making it move backwards. The period of this new task is 40 ms.
2. Modify the file `nxtSegway.oil` accordingly.
3. Test the new application on the Lego robot

- The ultrasonic sensor must be initialized `void ecrobot_init_sonar_sensor(int Input_Port)`
- Acquisition of the distance by the ultrasonic sensor: `int ecrobot_get_sonar_sensor(int Input_Port)`
- Port to which the sensor is connected: `PORT_SONAR`

Step 4: Initializing the system using a task

1. Write a new task that initializes the system and launches the task `pendulum` by sending an event
2. Add in the file `nxtSegway.oil` the configuration of the event
3. Test the new application on the Lego robot

Step 5: System Initialization via Hook Routines

- The StartupHook routine allows you to perform operations when OSEK is started.

1. Write the code of the StartupHook routine that initializes the system.
   `void StartupHook(void)`
2. Enable the use of the StartupHook routine in the file `nxtSegway.oil`.
3. Test the new application on the Lego robot