|  |
| --- |
| Operating Systems CS6605 |
| Application Proposal |
| Deliverable 2 |
|  |
| Eduardo Gutarra Velez. |
| 3/1/2010 |

# Description of the user application

To demonstrate that the operating system is functional a user application will be written in order to validate it as well as to demonstrate its features. The user application proposed involves the robot advancing and playing a particular sound pattern while it is not being hit by light of a given intensity. If the robot is hit by light it will stop and start beeping to the amount of intensity it is being hit. Once the robot is no longer hit with light, it will resume its regular behavior of advancing and playing sound patterns. The following figure illustrates the main idea of this user application (Figure 1).

Figure

# Explanation of how the application will utilize the operating system.

## Periodic Processes

There will be five periodic processes kept in this application. The first periodic process will control the motors by periodically passing some voltage to the motors making them advance. Another periodic process is kept in order to check whether the robot is close to an obstacle or not. If it finds it is close to an obstacle it will determine whether the obstacle is to its left or to its right and change direction accordingly. The high level pseudo code for the obstacle detection process (Algorithm 1) and for controlling the robot’s motors (Algorithm 2) are listed below:

Algorithm 1. Obstacle detecting process

robot emits infrared

if (robot detects obstacle) then

if (obstacle is to the left) then

**OS\_Write**(MotorProcessFifo, LEFT)

endif

if (obstacle is to the right) then

**OS\_Write**(MotorProcessFifo, RIGHT)

endif

endif

Algorithm 2.Process for controlling robot’s motors.

**OS\_Read**(MotorProcessFifo, &value) *// Read value from message queue*

if (value == LEFT) then

move robot back

send signal to move right wheel

else if (value == RIGHT) then

move robot back

send signal to move left wheel

else *// Assumes the process has encountered no obstacles.*

send signals to move both wheels

endif

Finally, the other three periodic processes will print in the LCD three different messages in text. To make the messages readable without mixing them with one another through preemption, only one process will be allowed to write to the LCD at a time. This will be coordinated through semaphores.

## Sporadic Processes

The sporadic processes will produce a sequence of characters that represent different frequencies of beeps in order to create a sound pattern. This pattern will then be communicated to a device process so that it can reproduce it as a sequence of sounds in the piezo buzzer.

## Device Processes

One of the device processes in the application will play the patterns that are communicated by the sporadic processes which are left in its message queue. Because more than one sporadic process may attempt to write to that queue at the same time, a semaphore will be implemented to coordinate access to the device process which transforms the sequence of characters into actual beeps of different frequencies.

Another device process will also check at certain periods of time whether it has detected light of a given intensity. If it does, it will preempt all the other processes and wait until the last sporadic process’s message is done playing. Once that is done, the robot will start beeping with a frequency that is proportional to the intensity of light it is getting. If no light is detected by this process, it will yield execution to the other processes. The high-level pseudo-code for this process is listed below:

algorithm 3. lIGHT DETECTION PROCESS

while (true)

if (light is detected) then

robot beeps to the intensity of light.

else

**OS\_Yield**()

endif

## Idle Process

The robot’s sporadic processes eventually finish producing their sound pattern and terminate. When the robot is done running the sporadic processes, it will run the idle process. The idle process will make the robot stay silent for a short period of time and then it will start creating new sporadic processes that continue producing sequences of characters to produce sound patterns.

## Interprocess communication

Whenever the robot needs to change direction, the periodic process that detects where there is an obstacle, must communicate a flag to the motor controlling process. This flag will be used by the motor controlling process to determine what motors it needs to move in order to perform the required change in direction.

Interprocess communication also exists between the sporadic processes and the device process that uses the piezo buzzer to produce the sound patterns. Each sporadic process will try to write its sound pattern in the device process’s message queue. In order to prevent a race condition between the sporadic processes, a semaphore will be used to coordinate which sporadic process gets to write in the device process message queue at a time.

## Semaphores.

Semaphore primitives will be implemented in each of the sporadic processes in order to coordinate access to the message queue of the piezo buzzer device process. Each sporadic process will act like a producer process and the piezo buzzer device process will act like a single consumer process. An algorithm to solve the common producers and consumer’s problem will be implemented so that the sporadic process can synchronize with the device process to produce sound patterns effectively. The pseudo code for this scenario is listed as follows:

algorithm 4. producer and consumer synchronization through semaphores

Semaphore fillCount = 0

Semaphore emptyCount = FIFOSIZE

sporadic **process**() {

while (true) {

character = **produceCharacter**()

**OS\_Wait**(emptyCount)

**OS\_Write**(character)

**OS\_Signal**(fillCount)

}

}

device buzzer **process**() {

while (true) {

**OS\_Wait**(fillCount)

**OS\_Read**(character)

**OS\_Signal**(emptyCount)

reproduce beep with character

}

}

Additionally if there is more than one sporadic process, only one process should be allowed to write in the piezo buzzer device process’s queue, and therefore a mutex semaphore must be kept so that only one process can write its pattern in the message queue at a time.

Finally a semaphore for the LCD will be used to coordinate the three periodic processes that write messages to the LCD. The pseudo code for one such process is shown below:

Periodic Process X ()

{

**OS\_Wait**(LCD Semaphore);

write message;

**OS\_Signal**(LCD Semaphore)

}