

COMP105P Lab Tasks 2

Tasks

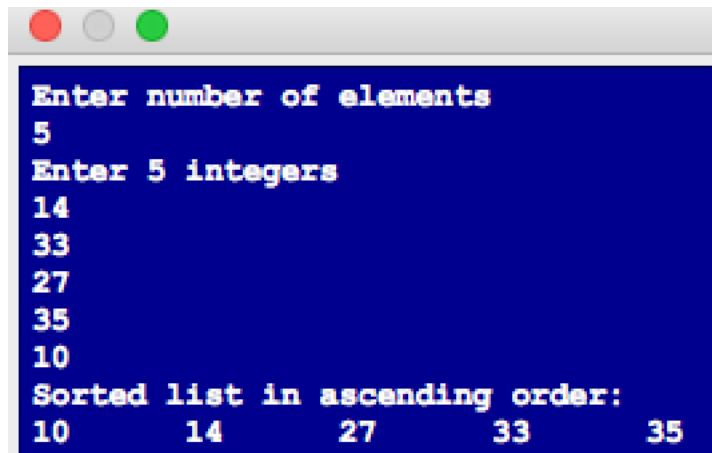
A sorting algorithm is an algorithm that puts elements of a list in a certain order (e.g. ascending, descending). There are several kinds of sorting algorithms: bubble sort, merge sort, selection sort, etc. In this lab, we will study some of the sorting algorithms using the Propeller C microcontroller (i.e. the robot's brain).

Preparation: Study the bubble sort algorithm: There are many resources on the web. The following is an example: https://en.wikipedia.org/wiki/Bubble_sort

Note: For tasks 2.1 and 2.2, the robot 3-position switch should be set to position 1 (no need for the servos).

- **Task 2.1.** A user should be able to enter a list of random numbers using the SimpleIDE terminal. The robot will then sort this list in ascending order using the bubble sort algorithm (see figure 1).

Note: your code should be able to accept any number of input elements (integers).



```
Enter number of elements
5
Enter 5 integers
14
33
27
35
10
Sorted list in ascending order:
10      14      27      33      35
```

Figure 1

- **Task 2.2.** Modify your code written for task 2.1 in order to display the content of the list of numbers in each iteration (intermediary steps of the algorithm). See example in figure 2. You don't need to use colors, just output the content of the list of elements for each iteration of the algorithm.
- **Task 2.3.** You will be using a setting similar to figure 3. The starting position for the robot is shown in figure 3. The robot will order the input list of numbers using the bubble sort algorithm. To mark a swap, get the robot to move forward and stop at the middle of the two numbers to swap and blink one of the LEDs on its board (figure 4 and 5). You should manually swap the two numbers (numbered sheets of paper). The robot will continue till the list is sorted. To mark a finish, get the robot to move to the middle of the list and turn 90° right (see figure 6).

Note: For task 2.3, the 3-position switch of the robot should be set to position 2 after loading the code to EEPROM as the servos are needed for this task.



Figure 2

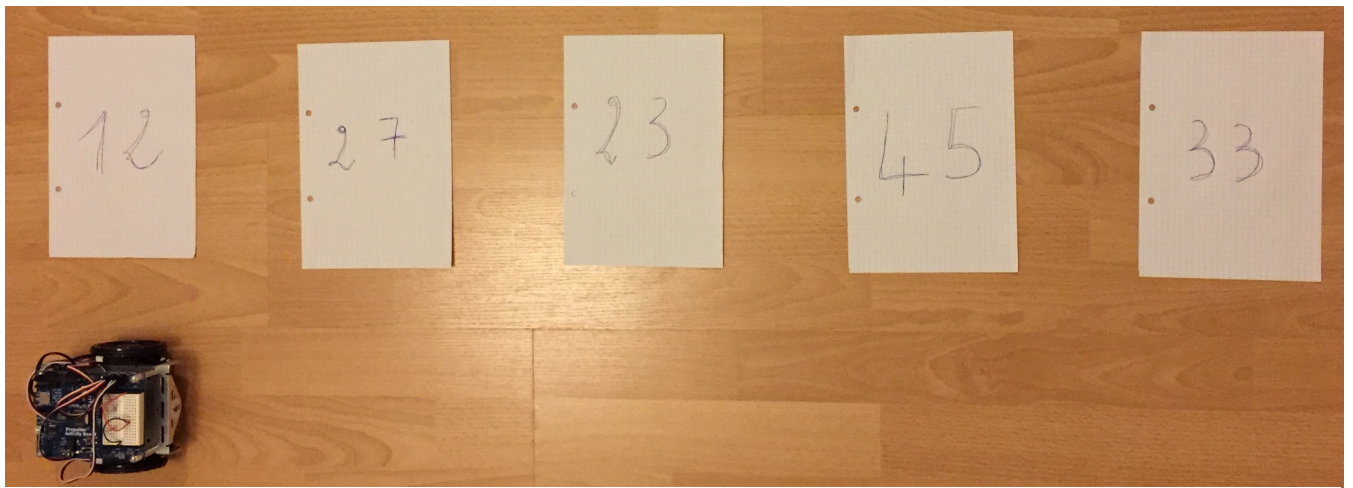


Figure 3

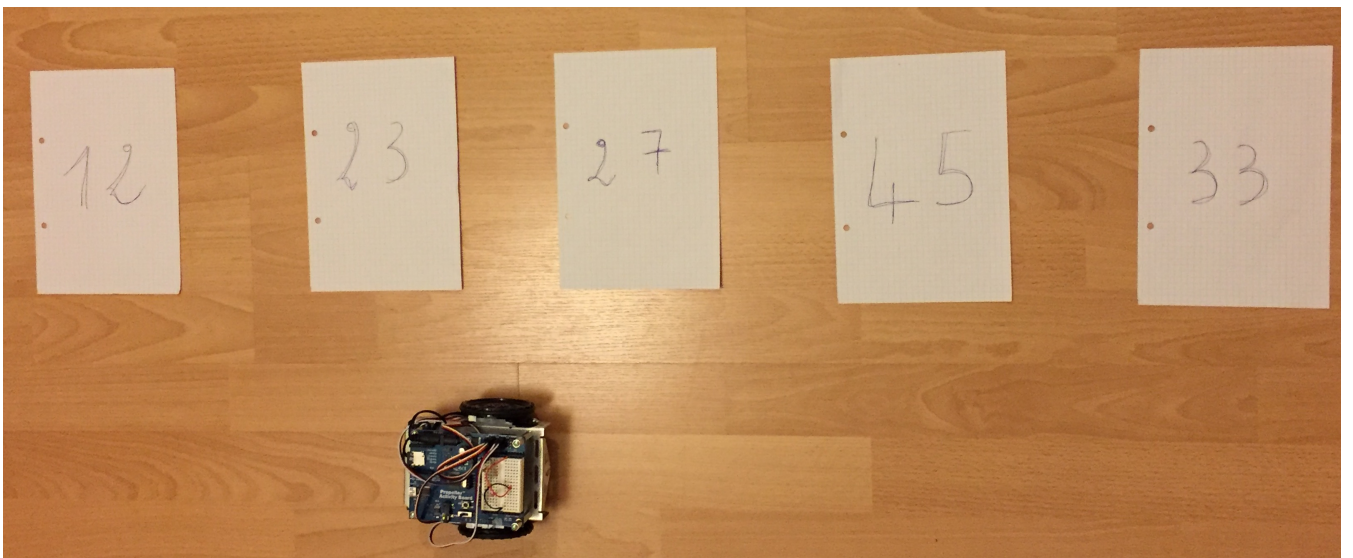


Figure 4

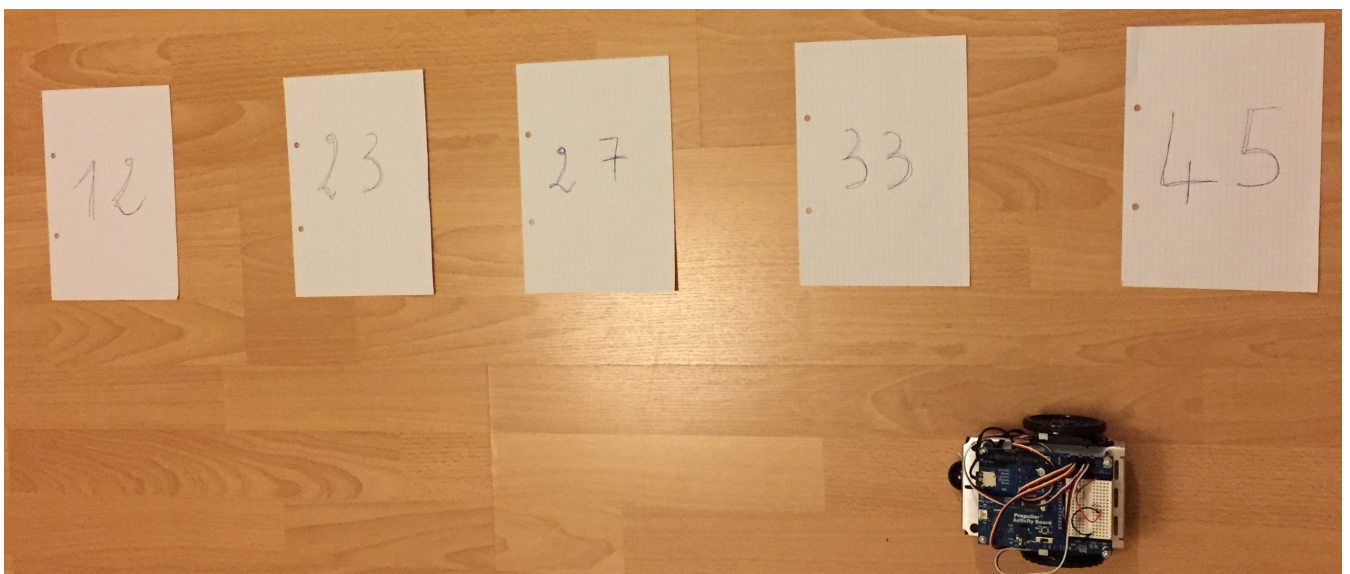


Figure 5

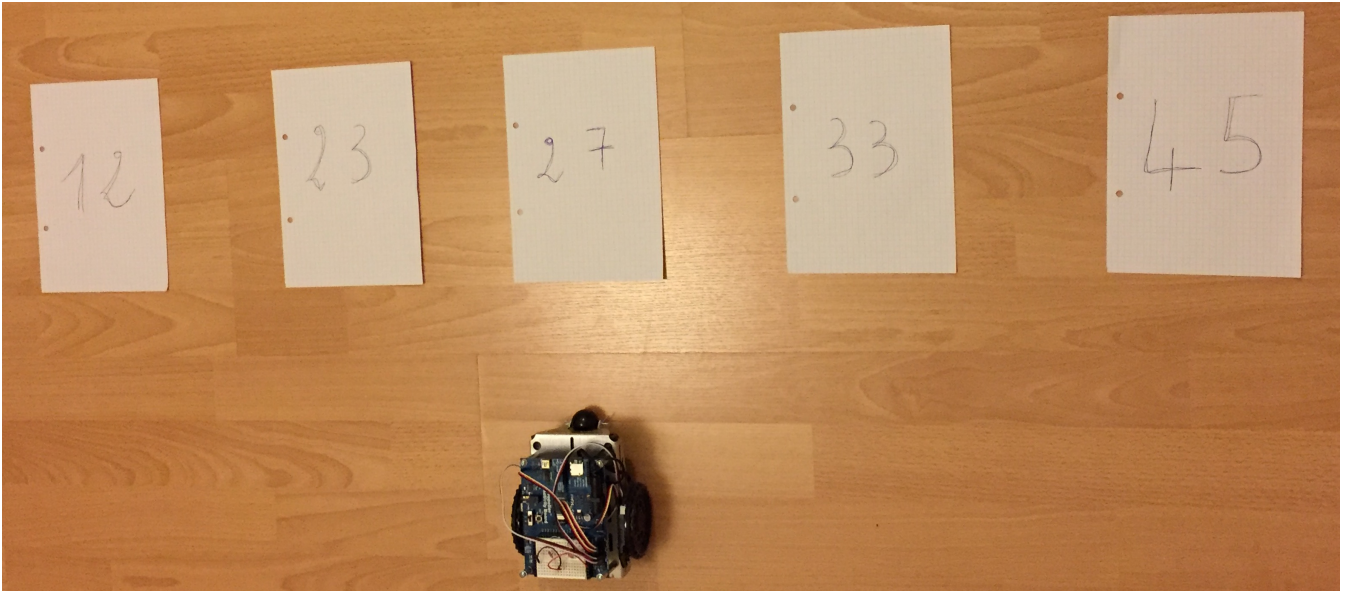


Figure 6