## 2.3 Straight-line robots

### *2.3.1 Introduction*

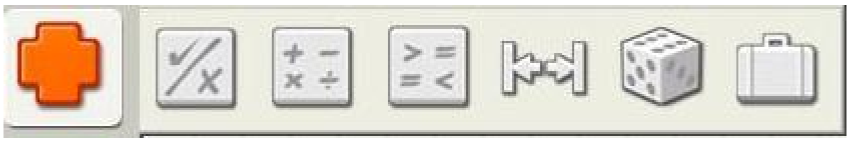
In this section, we will introduce the reader into the "designing-building-programming robots" loop process necessary to address the problem of "Robotics as learning object". We need to acquire all the skills necessary for designing, constructing and programming robots. From a methodological point of view we suggest that the proposed activities in the study of "Robotics as learning object" should be based on the constructivist educational methodology explained in other sections of this book. The format is, therefore, a progressive sequence of problems that can synthesize the constructivist path towards the "robot designing-building-programming" process. In this section, we start with the simplest possible robot i.e. a robot moving along a straight line.

The main objectives (competences or skills to be acquired) behind the problems presented here are:

- To design and build "good" straight line robots
- To know how the actuators (motors) work
- To know how to write, download and execute a NXT-G program
- To know how to create and manage variables and blocks (necessary for abstractions from NXT-G in order to construct primitives oriented to the user and/or to the problem)

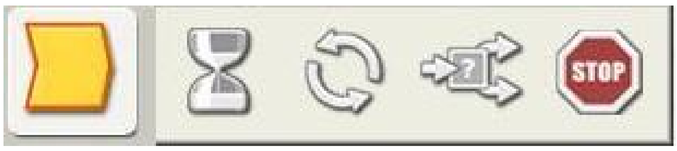
### *2.3.2 Necessary features from NXT-language*

Using *arithmetic & logic operators*: Data group blocks



This group of icons allows the user to make any arithmetic or logic operation. The use of variables (explained later on) is also available here.

Using *Control Structures:* The Flow group is represented by three main different icons.

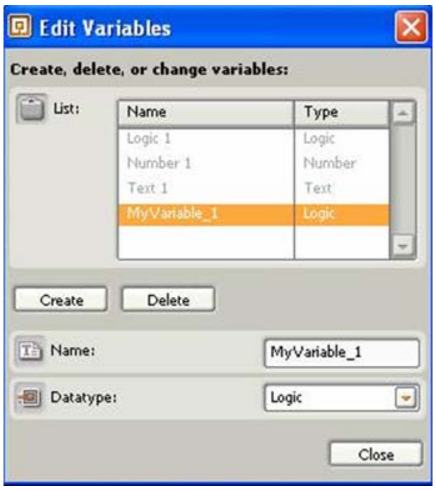


*Wait:* Waiting for an amount of time or waiting for a specific sensor event

*Loop:* Loop control structure (inclusive of either time or sensor events)

*Switch:* This block allows the use of *either If* statements or Case statements.

Creating and using *Variables:* To use a variable with LEGO Mindstorms, we need first to create it; to do so, we define the type, we write the name and then we click the “create” button.


**Edit Variables**

**Create, delete, or change variables:**

List:

| Name | Type |
|---|---|
| Logic 1 | Logic |
| Number 1 | Number |
| Text 1 | Text |
| MyVariable_1 | Logic |

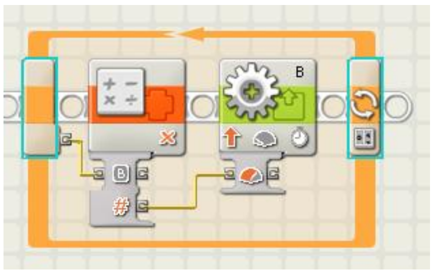
Create Delete

Name: MyVariable_1

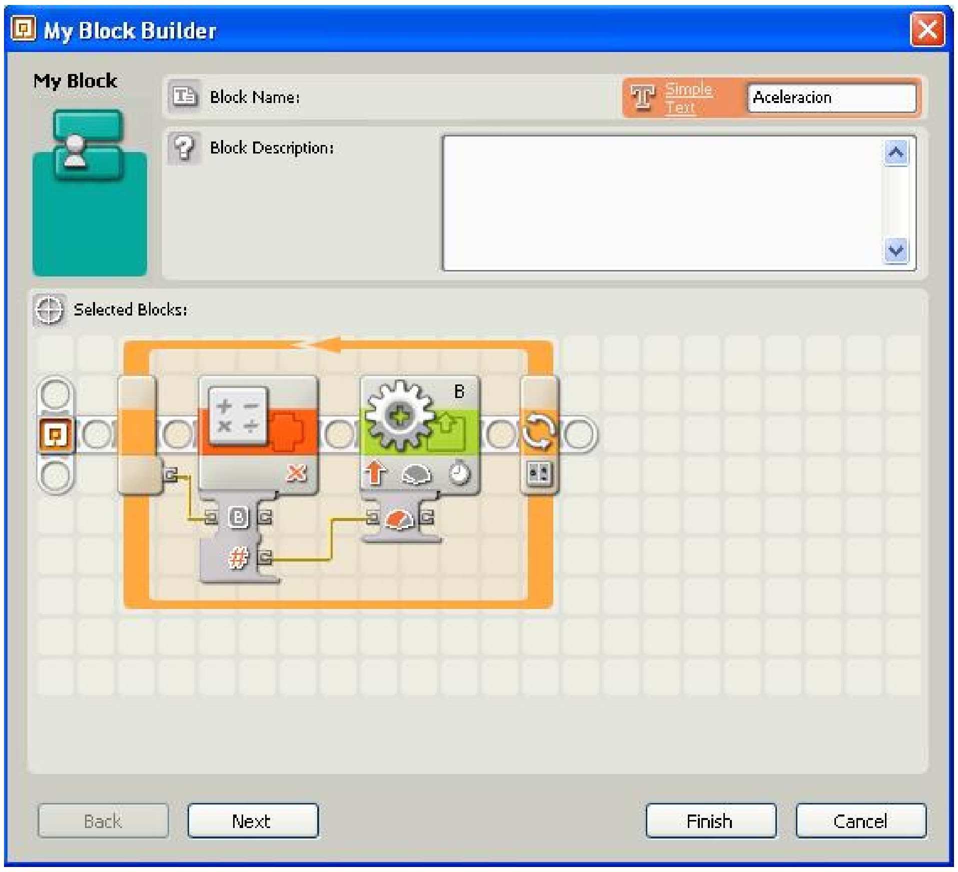
Datatype: Logic

Close

Creating and using *User Blocks:* User blocks are for the user (robot programmer) the means whereby to define procedures; from a didactical point of view, this can, also, be seen as a tool for abstraction on the part of the user (as we will explain it later on). Let us see how we can create a user block with an example. We have written a program that tells to our robot to go forward with a constant acceleration. The following NXT-G program has a loop where a variable is increasing at every step that produces a constant acceleration:



If we wish to transform this program into a new User Block: We select the entire program, and then we click “*New My Block*”: we follow the instructions giving a name and assigning an icon to the new block. These “created blocks” are, in fact, procedures and can be defined both without and with parameters or arguments. Here is another example where a new block is defined with one input parameter and one output value (it is a block to convert numerical values into text values for displaying purposes):



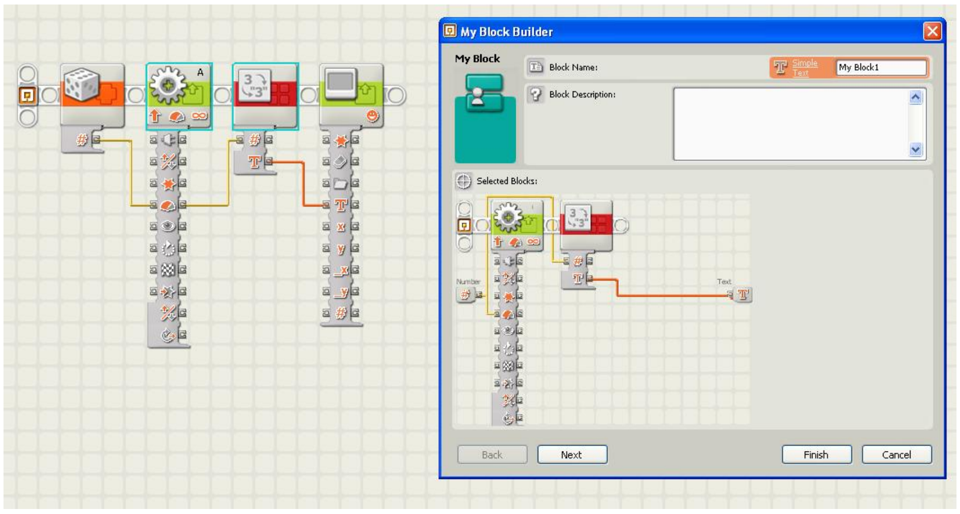
My Block Builder

My Block

Block Name: Aceleracion

Block Description:

Selected Blocks:



My Block Builder

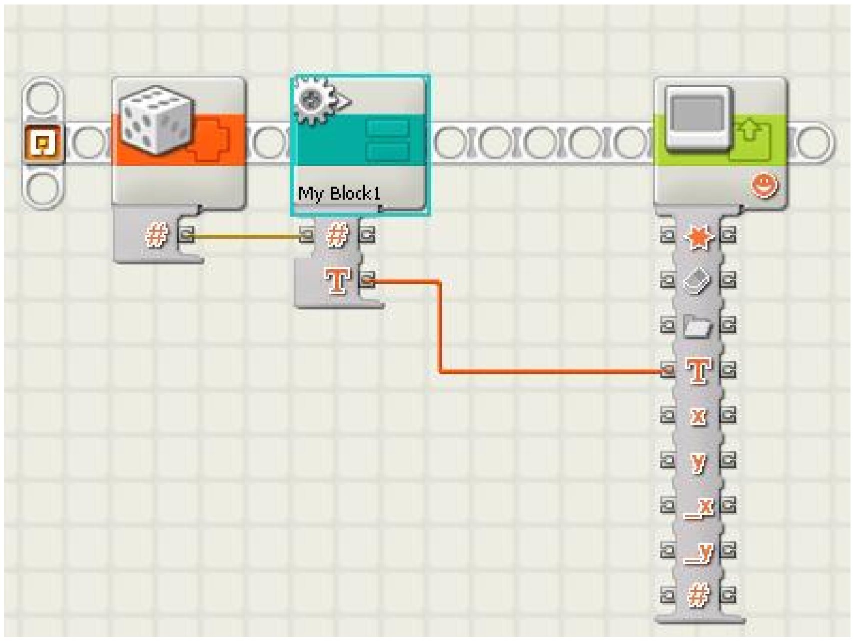
My Block

Block Name: My Block1

Block Description:

Selected Blocks:

The new block can be used within a NXT-G program.



The problem 0 will focus on designing and building robots. The problems 1 to 6 will focus on the use of the *Move* blocks; the main interest is in understanding the relation between the 2 main parameters, i.e. power and duration (degrees/rotations or time) of this block and the speed (v), time (t) and displacement (x) of the robot (for example in cm/s and in cm):

### *2.3.3 Problem 0: How to assemble "good" straight line robots*

*a. Objectives/ Aims:* Pupils should be able to

- Assemble simple straight line robots (with one or two Motors) using the educational Lego Mindstorms kits.
- "Copy" some of the models shown as examples and designed with LEGO DIGITAL DESIGNER
- Compare and asses the different robots built in the classroom by others from other projects. Criteria such as stiffness, stability, simplicity and aesthetics should be applied.

*b. Previous knowledge required:*

- Basic techniques/skills to assemble LEGO Mindstorms pieces:
  - Girders / beams assembly in parallel with pegs/ dowels
  - Girders/ beams cross assembly

  - Axles /axis of symmetry, friction rings, wheels
- Basic mechanical rules about:
  - Rigidity/ stiffness
  - Stability
  - Friction and adherence
- Simplicity and aesthetics criteria
- Basic user knowledge of LEGO DIGITAL DESIGNER software (http://ldd.lego.com)

Some basic skills are required to assemble the pieces and a previous recognition of the different kinds of pieces is, likewise, required. An initial step could be to take a robot, dismantle it and classify its different pieces.

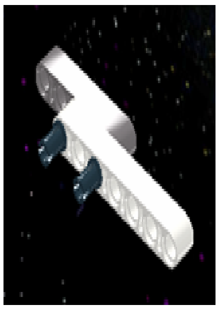
c. The teacher formulates a problem: *Are students able to build a simple straight line robot after observing the examples of the LEGO DIGITAL DESIGNER (LDD) application?* Conditions: a simple, stiff stable robot with one or two motors.

d. Students will build their own robots with the teacher's help. Most students will have their first contact with Lego Mindstorms kit, so the teacher will have to teach the basic assembly skills:

- Longitudinal assembly with pegs / dowels
- Girders cross assembly
- Axles assembly with rings and wheels

Next, students will build a complete robot, while teachers should stimulate them to build their personal robot, to obtain a certain variety and, then, analyse the different criteria of rigidity, stability etc.

Students can build robots as shown in the next images.

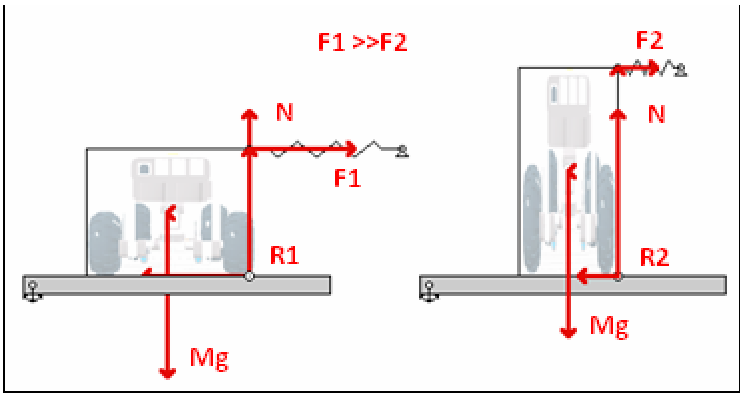


e. generalisations of local solution

Teachers will propose to students to test the quality of the robots, taking into account different aspects. The activity consists of testing the robots and discussing in groups what characteristics an ideal robot should have.

Some examples follow:

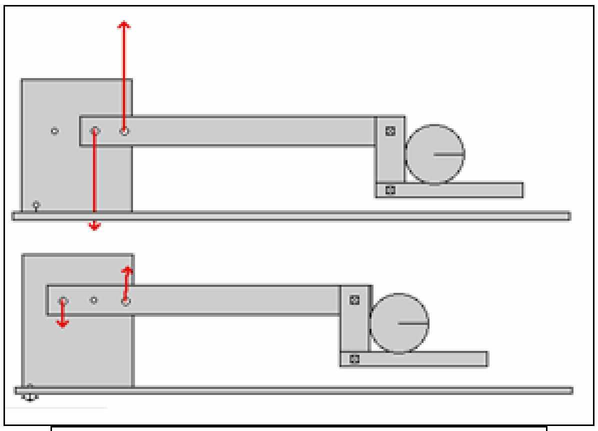
*Stability*: Every car will be pulled with a dynamometer from the height it corresponds to the top of it and, then, the force necessary to knock it over will be observed. The next image represents forces F1 and F2.

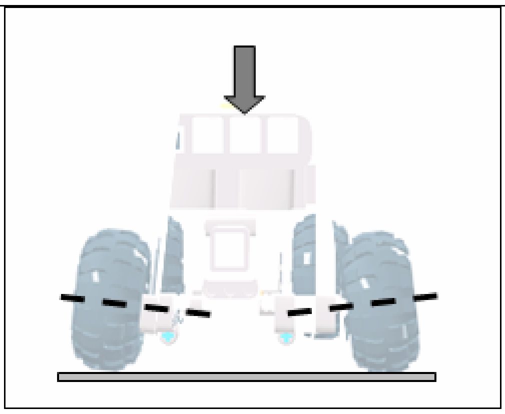


Both robots are of the same weight but of different dimensions (robot 1 is short and wide and robot 2 is high and narrow). It can be observed that both robots are of the same weigh, but robot 1 needs more force than robot 2 to be knocked down.

*Rigidity / stiffness:* the higher the tension of the pieces (axles/axes and pegs/ dowels), the less the rigidity of the robot. This is due to external forces like gravity and weights to be lifted by the robots.

The following image shows the different tensions in dowels/pegs assembling an arm, when they are exerted in contiguous or alternate holes. In the first case the tension is higher.





It is enough for rigidity testing to apply an external force on the robot (pushing down with a finger, for instance) and to observe if the structure gives way or gets loose.

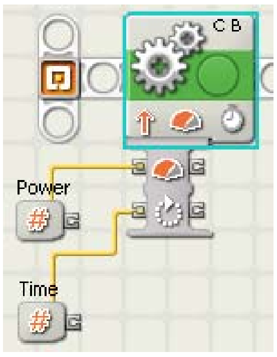
### 2.3.4 Problem 1: How to teach a simple robot to do a forward movement for a given interval t of time with a given Power P?

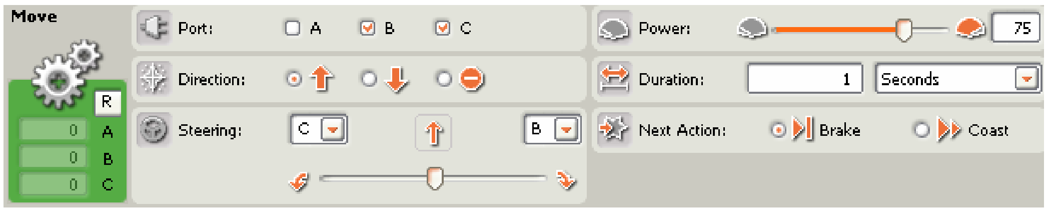
*a. Aims:* The pupils should learn how to write and use a procedure (my block) that allows the robot to make a lineal displacement for a given time t and with a given power P.

*b. Previous knowledge required:*

- To know how to build a simple robot.
- To know the basic features of NXT-G: to write a program and to download it to the robot.

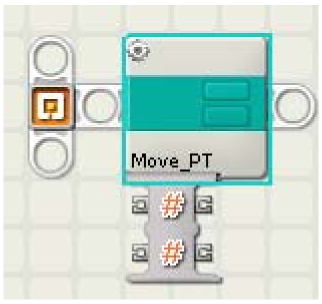
c. Constructing the procedure *Move_PT (power, time):*

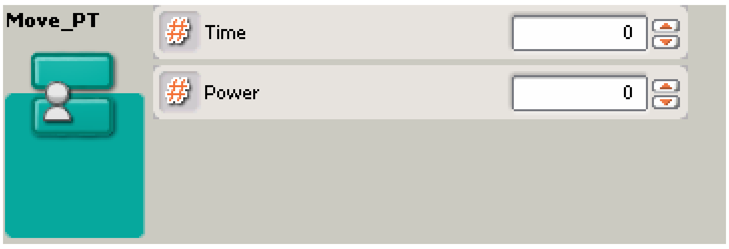



**Move**

R
0 A
0 B
0 C

Port: A [✓ B] [✓ C]
Direction: [✓ ↑] ↓ ⊖
Steering: C ↑ B
Power: 75
Duration: 1 Seconds
Next Action: [✓ Brake] Coast




**Move_PT**

| | |
|---|---|
| Time | 0 |
| Power | 0 |

We construct one user block with 2 parameters: time and power. This means that we have one block with only 2 parameters, easier to use than the normal "Motor block". With the new block we only have to define the power and the time (in seconds).

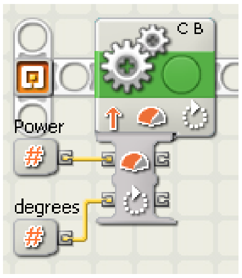
### *2.3.5 Problem 2: How to teach a simple robot to make a forward movement with a given power for a given angular distance φ (of the wheels)*

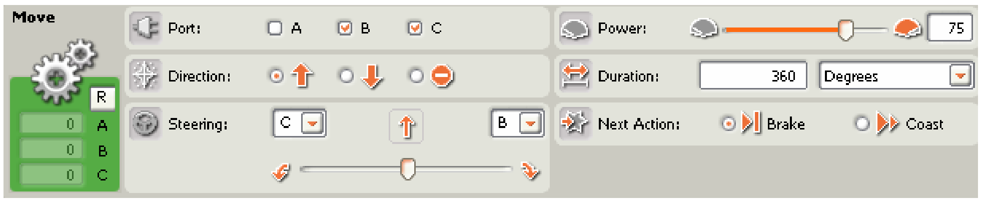
*a. Aims:* The pupils should be able to write and use a procedure to make a lineal displacement of the robot for a given value of *φ* degrees (of the wheels) of the Motor and a given Power P.

*b. Previous knowledge required:*

- Problem 1
- To know how to build a simple robot.
- To know the basic features of NXT-G: to write a program, to download it to the robot.

c. The procedure *Move_PD (power, degrees)*:




Move

Port: A [✓ B] [✓ C]

Direction: [✓ forward] backward stop

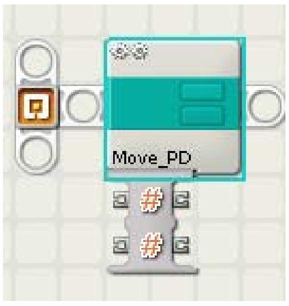
Steering: C B

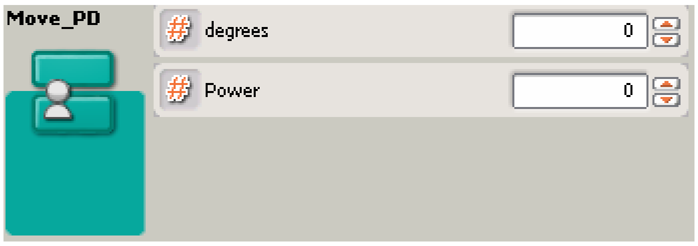
Power: 75

Duration: 360 Degrees

Next Action: [✓ Brake] Coast

We construct one user block with 2 parameters: angle and power. This means that we have one block with only 2 parameters, easier to use than the normal “Motor block”. With the new block we have only to define the power and the number of degrees (it means how long we rotate the wheels/motors of the robot); the result is to move forward (or backward) a given distance depending on the given angle.




Move_PD

degrees 0

Power 0

### *2.3.6 Problem 3: How to teach a simple robot to make a forward movement movement with a given power for a given distance X?*

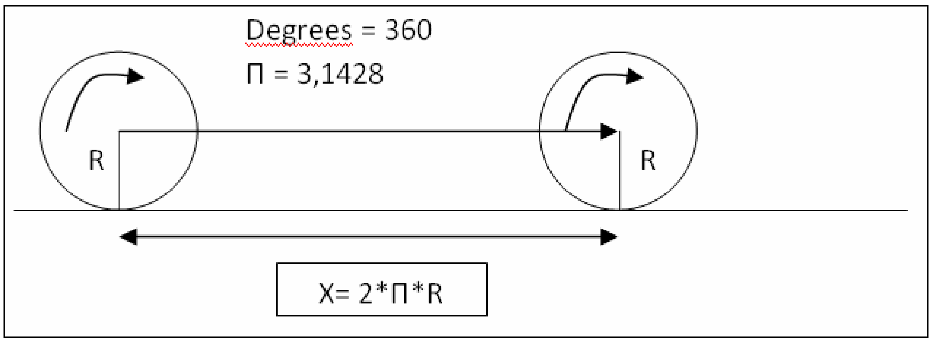
a. *Aims:* The pupils should

- learn how to create a procedure to convert a value X of distance in cm into the equivalent parameter "degrees" of the Motor block (angular distance)

- be able to create a user block *Move_PX (power,x)***,** where the distance is given in cm.

b. To do so, we could use the already constructed block *Move_PD(power, degrees)*. We need to make the following transformation: OPERATION (x)→degrees, transforming angular distance into linear distance.

We should know that there is a relation between the radius R of the wheels/motors, the angular distance (rotation in degrees of the wheels/motor) and the linear distance (X). In the following figure we see that for 360 degrees of rotation, the linear distance X is given by the expression: 2*Π*R.



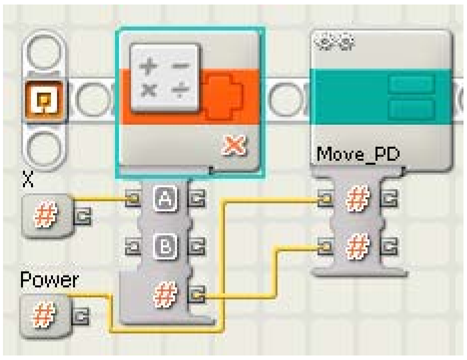
This means that we have the following relation: Angle/x = 360/2*Π*R and then we can deduce that Angle = (360*X)/( 2*Π*R). The relation between angle and x (angular and linear distance, what we need to find) is: angle = (360/(2*Π*R)) X or angle = K*X where K = 360/2*Π*R = 14 (for R=4,08). Then we can make the following explorations:

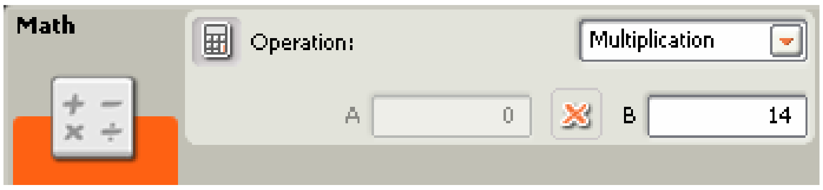
| rotation (360 degrees) | experimental distance | theor distance | Radius= | 4,08 |
|---|---|---|---|---|
| 0 | 0,0 | 0,0 | | |
| 1 | 26,0 | 25,6 | | |
| 2 | 52,0 | 51,3 | | |
| 3 | 77,3 | 76,9 | | |
| 4 | 102,5 | 102,5 | | |
| 5 | 128,7 | 128,2 | | |
| 6 | 153,8 | 153,8 | | |

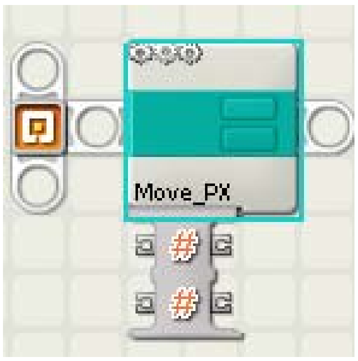
*c. Previous Knowledge required:*

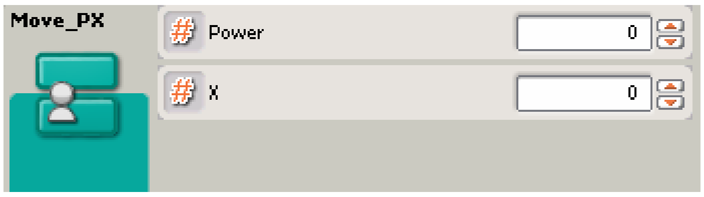
- Problems 1, 2 and 3
- Experimental knowledge: X = 2*π*R* degrees/360
- Programming previous knowledge

d. The procedure *Move_PX (power, X)*









### 2.3.7 *Problem 4: How to teach a simple robot to make a forward movement with a given speed V for a given interval t of time?*

*a. Aims:* The pupils should learn

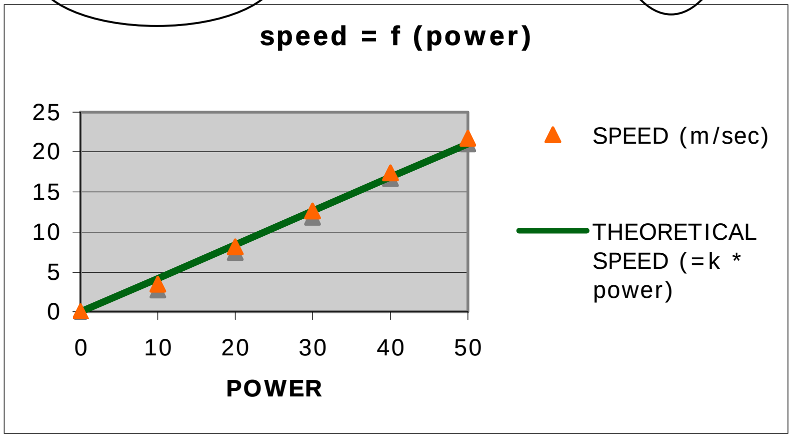
- To create a procedure to convert a value V, cm/s, into the equivalent of Power P (as it is defined in the Motor block).
- To create a block "*Move_VT (V, time)"*, where the velocity is given in cm/sec and time in seconds.

We can use the already constructed block *Move_PT (power, time)* and for that we need to make the following transformation: OPERATION V → power (transforming velocity into power values)

b. We do several experiments where we measure the distance travelled for 10 seconds with different power values. Doing so, we obtain the experimental speed. We observe in the following data table and graph that there is a relation (K) between Power and Speed:

V = 0,42*Power and Power = V/0,42 or Power = 2,38 * V

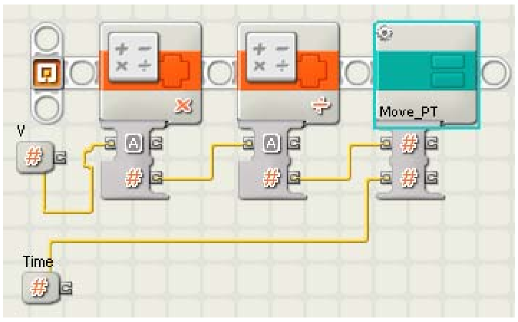
| POWER (%) | DISTANCE (cm) | TIME (S) | SPEED (cm/sec) | THEORETICAL SPEED (=k * Power) |
|---|---|---|---|---|
| 0 | 0 | 10 | 0 | 0 |
| 10 | 35 | 10 | 3,5 | 4,2 |
| 20 | 82 | 10 | 8,2 | 8,4 |
| 30 | 126 | 10 | 12,6 | 12,6 |
| 40 | 174 | 10 | 17,4 | 16,8 |
| 50 | 217 | 10 | 21,7 | 21 |
| | | | | |
| **k=** | **0,42** | | | |

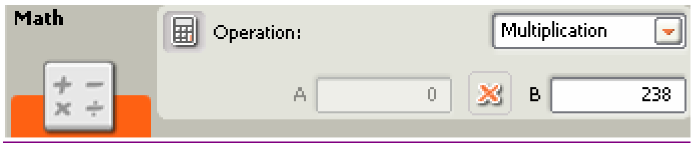


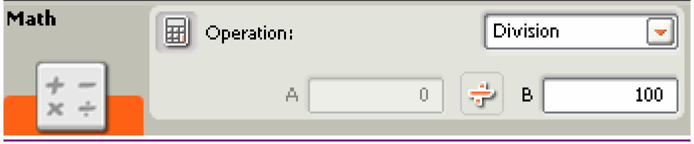
*c. Previous Knowledge required:*

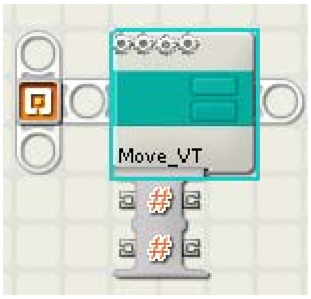
- Problems 1 and 2
- Experimental knowledge: V = 0,42 * Power
- Procedure *Move_PT(power, time)*

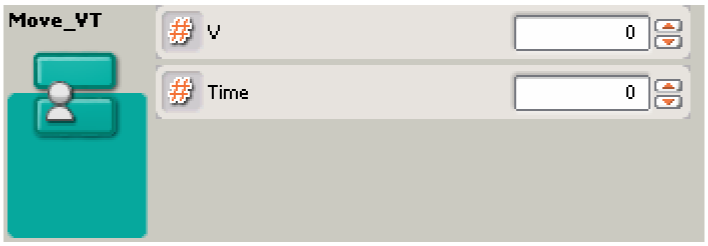
d. So the procedure *"Move_VT (v, time)"* is the following:




Math
Operation: Multiplication
A 0 B 238

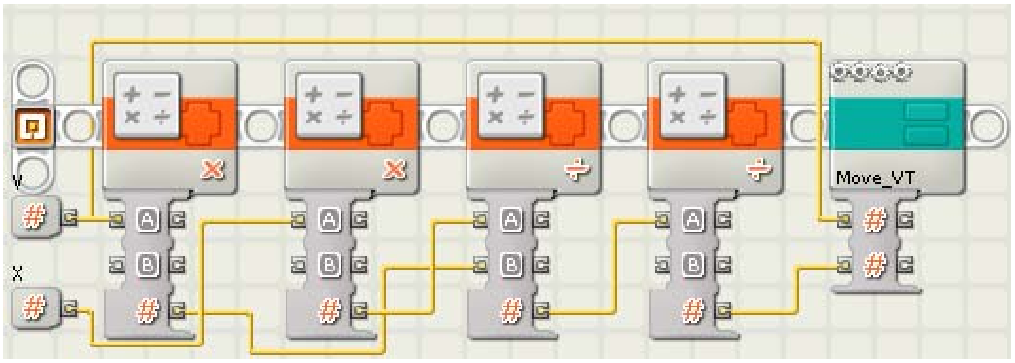

Math
Operation: Division
A 0 B 100

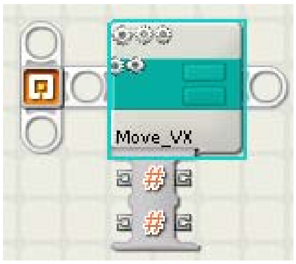


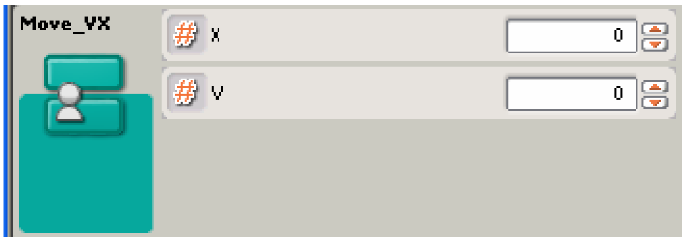

Move_VT
V 0
Time 0

### *2.3.8 Problems 5 and 6: How to teach a simple robot to make a forward movement with a given speed for a given distance X? How to teach a simple robot to make a forward movement for a given distance X during a given time t?*

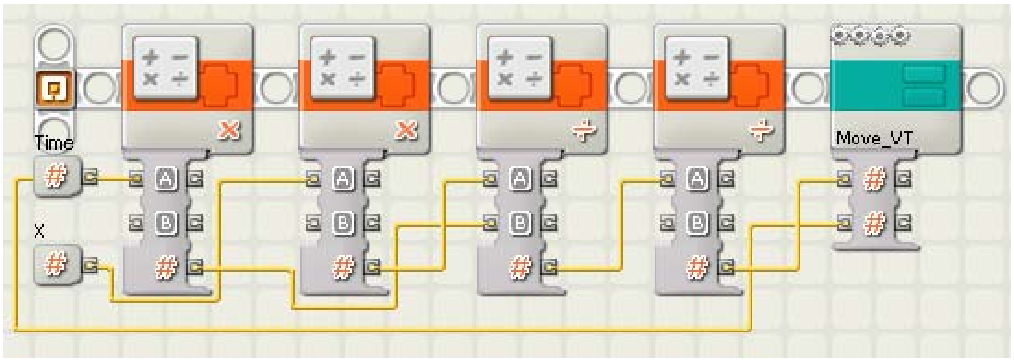
After all we have already done it is straightforward to "construct" the procedures *Move_VX (v, x)* and *Move_XT (x, time)* using the known relation V= X/T and the already developed procedure *Move_VT (v, time)*.

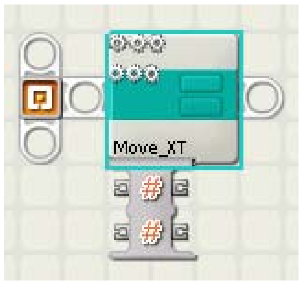


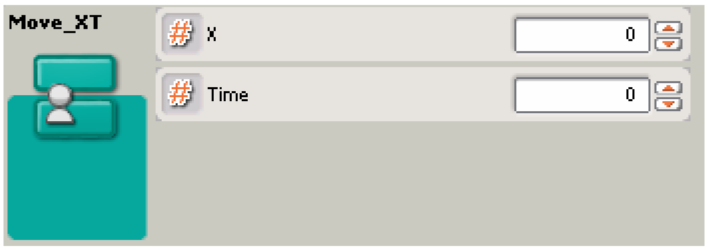




| | |
|---|---|
| X | 0 |
| V | 0 |







We have now completed all the possible combinations of V, X and T to address the motion for a "straight line robot" constructing different procedures or blocks that realise these possibilities and solve all the sequence of relevant problems.

After solving these problems, we can discuss the following further questions (mostly to introduce the use of loops with the blocks):

- How to make a "long" forward displacement as a sequence of short forward displacements?
- How to make different successive short displacements for a given time and continuously longer every time?