AN INTRODUCTION TO PROGRAMMING
WITH THE TI-Innovator ROVER USING THE TI-inspire CAS CX

-

T3 Symposium 2018, Brussels Belgium
L.A.A. Blomme – T3-Vlaanderen

The TI-innovator Rover is an educational robotic vehicle that can be programmed with the TI-inspire CAS CX (or another TI graphing calculator) to explore topics in mathematics, science, coding and STEM.

Rover is an introduction to coding and robotics. The simple programming language (TI basic) that is built into TI graphing technology makes it easy to program the system, run it and troubleshoot to correct or fine-tune performance.

Why is this new and interesting? Because the TI-innovator Rover *moves*. The Rover adds a physical dimension to the verbal, symbolic and graphic representations. Motion enables students to engage in mathematics from a new perspective. Rover puts mathematics in motion in a way that encourages students to ask, "what if…?" and inspires them to persist in finding solutions to make the vehicle perform the way they want it to.

A rechargeable battery powers Rover's two motors. The vehicle also includes a colour sensor, a distance sensor, an LED display, a holder for a marker to draw paths on paper and a gyroscope to measure heading.

Rover's design allows for access to the TI-innovator Hub input and output ports, so additional sensors and capabilities can be added.

1. THE PROGRAMMING LANGUAGE TI-BASIC

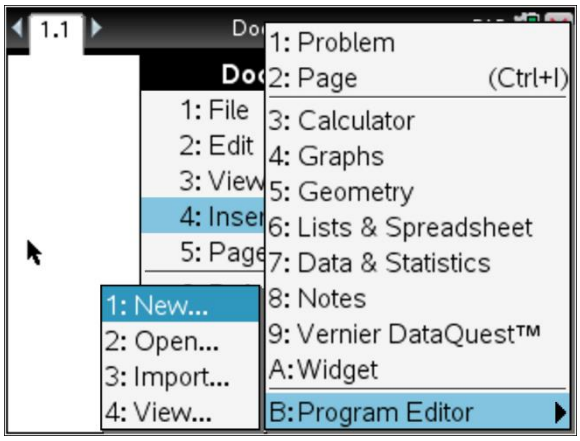
TI-BASIC is the official name of a BASIC-like language built into Texas instruments graphing calculators. For many applications, it is the most convenient way to program any TI calculator, innovator Hub or innovator Rover, since the capability to write programs in TI-BASIC is built-in.
For a complete list of the specific TI-innovator Rover commands, check the online document: TI-innovator Rover Commands Version 1.3 on
https://education.ti.com/html/webhelp/EG_Innovator/EN/content/eg_innovsys/resources/pdf/ti-innovator_rover_commands_en.pdf

In this introduction however, we will only use a limited number of the TI-innovator Rover commands (version 1.3).

• CONNECTING THE ROVER **Send "CONNECT RV "** • DRIVE **Send "RV FORWARD "** **Send "RV BACKWARD "** **Send "RV LEFT "** **Send "RV RIGHT "** **Send "RV TO XY "** • READ SENSORS **Send "RV RANGER "** **Send "RV COLORINPUT "** **Send "RV COLORINPUT.RED "** **Send "RV COLORINPUT.GREEN "** **Send "RV COLORINPUT.BLUE "** • LED COLOUR **Send "RV COLOR "** **Send "RV COLOR.RED "** **Send "RV COLOR.BLUE "** **Send "RV COLOR.GREEN "**	• **WAIT** • RV SETUP **SEND "SET RV.GRID.M/UNIT"**

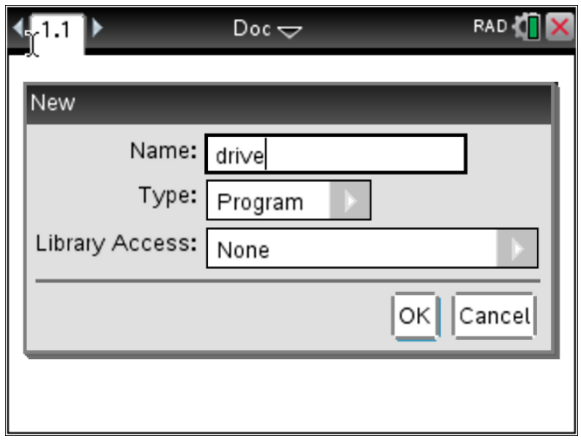
2. GETTING TO KNOW THE PROGRAMMING INTERFACE OF THE TI INSPIRE CAS CX

We will introduce the programming interface, step by step coding a simple program for the Rover.



Defining a program

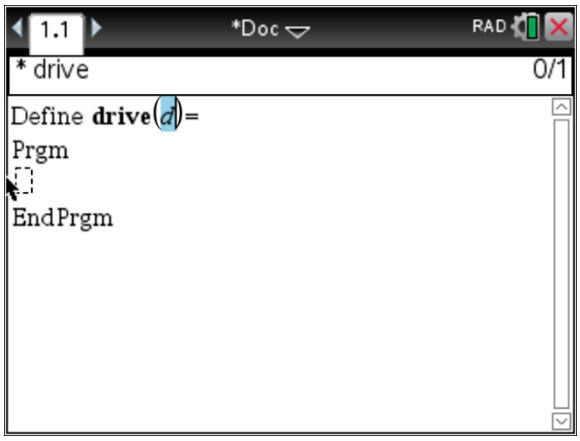
1. **Starting a new Program Editor**
   - On the handheld, press doc and select **Insert > Program Editor > New**.



   - Type a name for the program you are defining: **drive**.
   - Select the **Type** (**Program** or **Function**).
   - Set the **Library Access.**
   - Click **OK**.

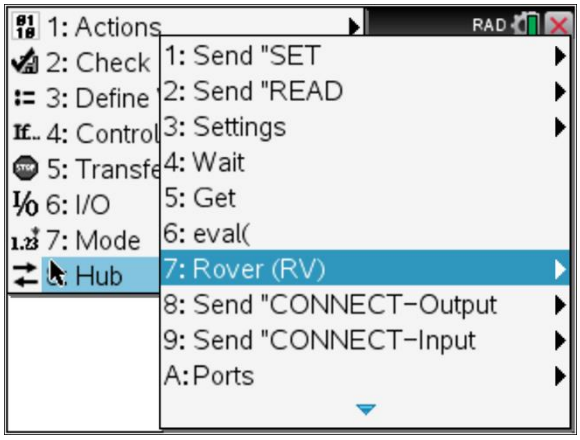
   A new instance of the Program Editor opens, with a template matching the selections you made.

2. Entering Lines

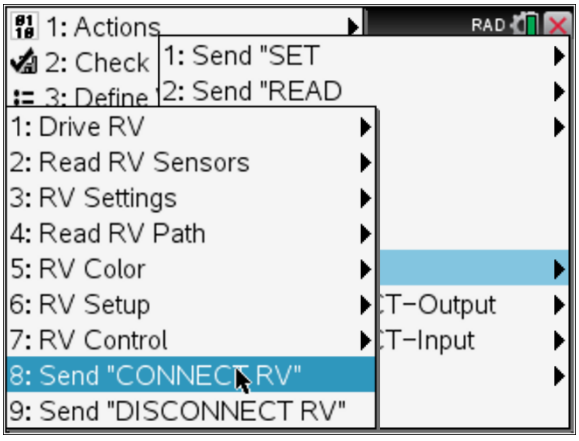


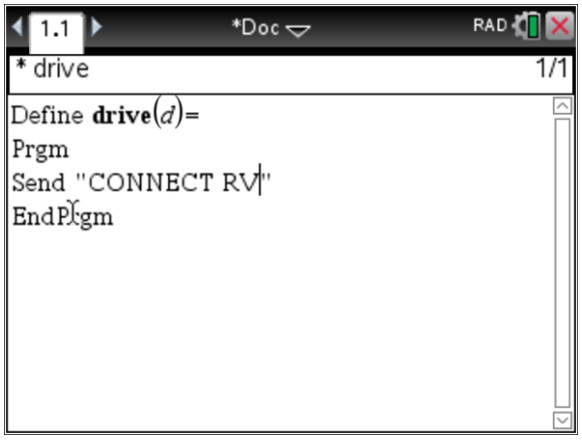
The Program Editor does not execute the commands or evaluate expressions as you type them. They are executed only when you run the program.

- If your program will require the user to supply arguments, type parameter names in the parentheses that follow the name. Separate parameters with a comma.
  Type: **d** (for distance).
- Between the Prgm and EndPrgm lines, type the lines of statements that make up your program.
  - You can either type the names of functions and commands or insert them from the Catalog or Menu.
  - A line can be longer than the width of the screen; if so, you might have to scroll to view the entire statement.
  - After typing each line, press **Enter**. This inserts a new blank line and lets you continue entering another line.
  - Use the ◄, ►, ▲, and ▼ arrow keys to scroll through the function or program for entering or editing commands.
- The first thing you need to do when writing a program for the Rover is to connect the Rover, the Innovator hub and the handheld.
  More info on page 27.

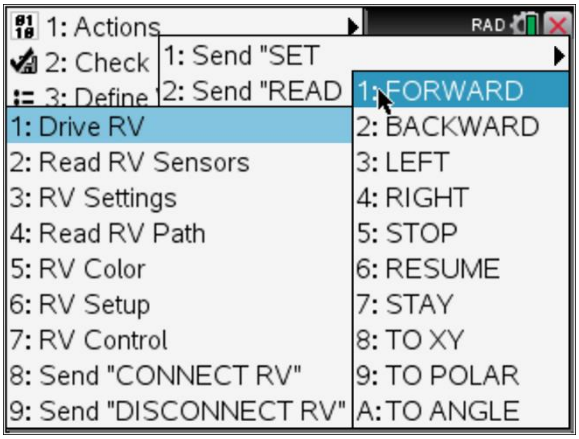


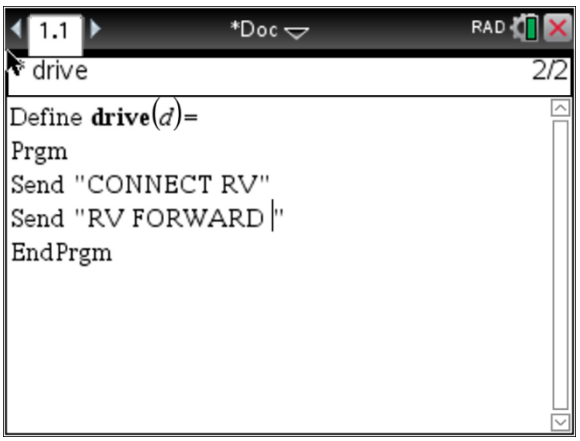
  - On the handheld, click (menu).
  - Click **Hub**
  - Click **Rover (RV)**
  - Click **Send "CONNECT RV"**
  - Press **Enter**.

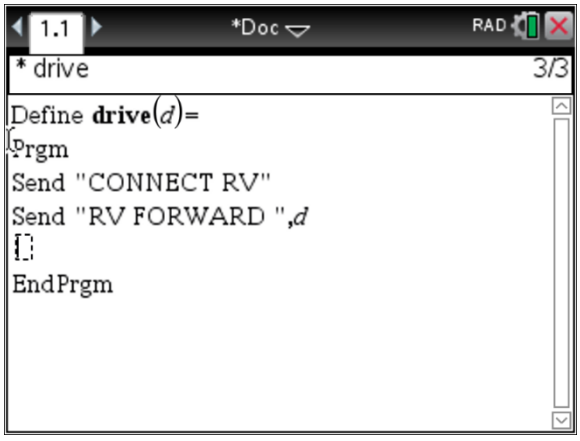




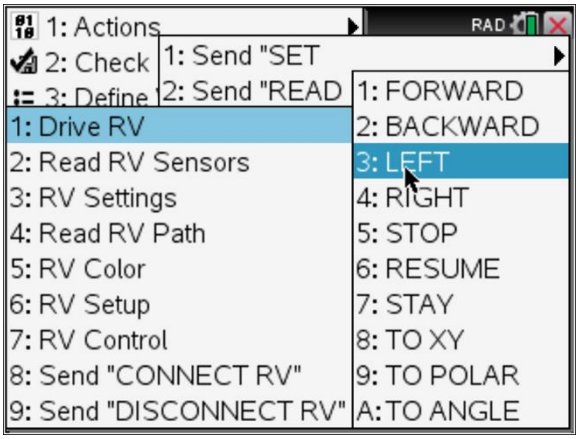
- We want to make the Rover drive a distance **d** forward.
    - On the handheld, click (menu).
    - Click **Hub**
    - Click **Rover (RV)**
    - Click **Drive RV**
    - Click **FORWARD**
    - Type: **,d**
    - Press **Enter**.

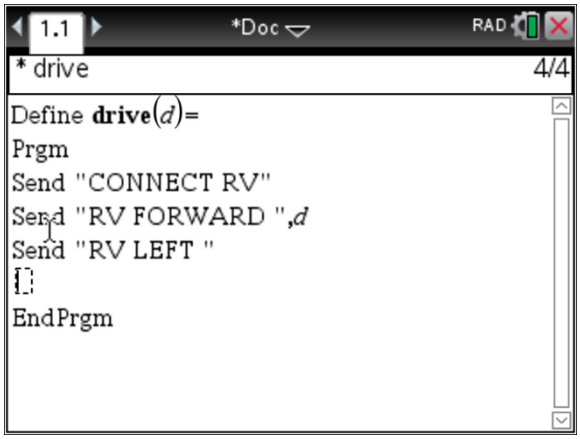






- Next, turn left.
    - On the handheld, click (menu).
    - Click **Hub**
    - Click **Rover (RV)**
    - Click **Drive RV**
    - Click **LEFT**
    - Press **Enter**.

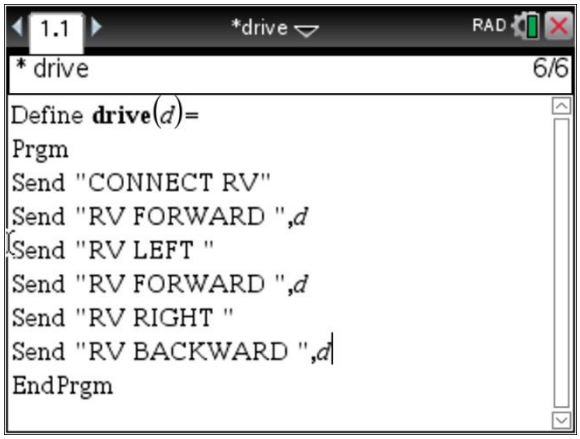




- Again, driving a distance **d** forward.
    - On the handheld, click (menu).

- Click **Hub**
- Click **Rover (RV)**
- Click **Drive RV**
- Click **FORWARD**
- Type: **,d**
- Press **Enter**.

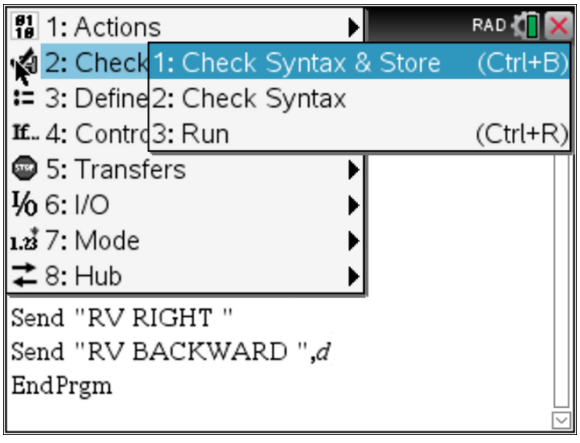
- Next, turn right.
  - On the handheld, click (menu).
  - Click **Hub**
  - Click **Rover (RV)**
  - Click **Drive RV**
  - Click **RIGHT**
  - Press **Enter**.
- Finally, driving a distance **d** backward.
  - On the handheld, click (menu).
  - Click **Hub**
  - Click **Rover (RV)**
  - Click **Drive RV**
  - Click **BACKWARD**
  - Type: **,d**
  - Press **Enter**.

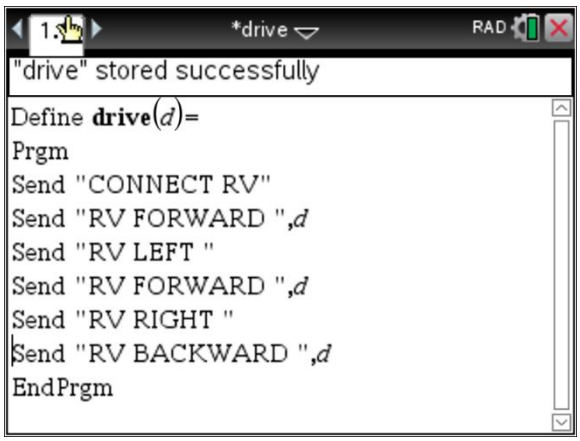


3. Checking Syntax

The Program Editor lets you check the program for correct syntax.

- On the handheld, click (menu).
- Click **Hub**
- Click **Check Syntax & Store**
- Click **Check Syntax & Store** or **Check Syntax**



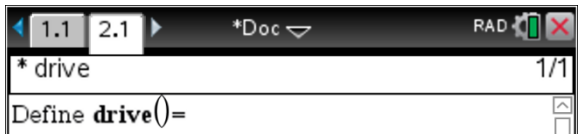


If the syntax checker finds any syntax errors, it displays an error message and tries to position the cursor near the first error so you can correct it.
If no syntax errors are found, the message “Stored successfully“ is displayed in the status line at the top of the Program Editor.

4. Storing the Function or Program

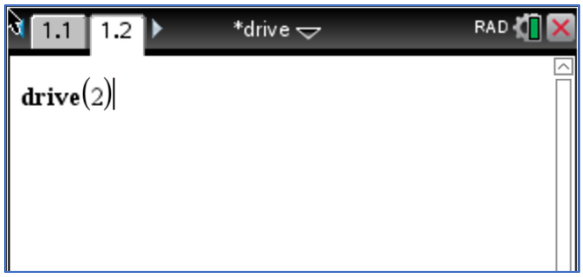
You must store your function or program to make it accessible. The Program Editor automatically checks the syntax before storing.

An asterisk (*) is displayed in the upper left corner of the Program Editor to indicate that the latest updates have not yet been saved.



3. RUNNING THE PROGRAM FOR THE FIRST TIME

- Add ***Calculator page***.
- Type the program name and parenthesis: **drive()**.
- This program requires an argument. Enclose it in the parenthesis. In this case enclose 2: drive(**2**).
- Press **Enter**.



The program runs and the Rover does what the program asks him to do by executing one command after the other.

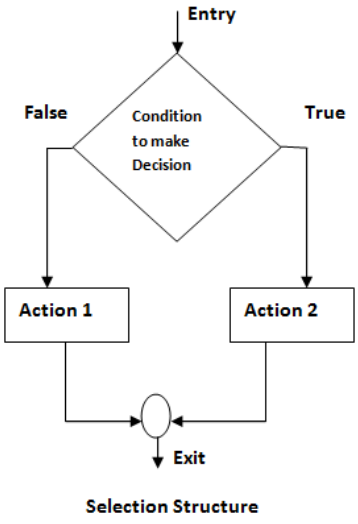
We used the default setting: distance in UNIT (grid units. Default, 1 UNIT = 10 cm).

4. MORE BUILDING BLOCKS FOR CODING

Our first program has a simple list of instructions for the Rover to execute. However, how about making decisions, repeating a set of instructions, etc. ?

1. CONDITIONAL STATEMENTS

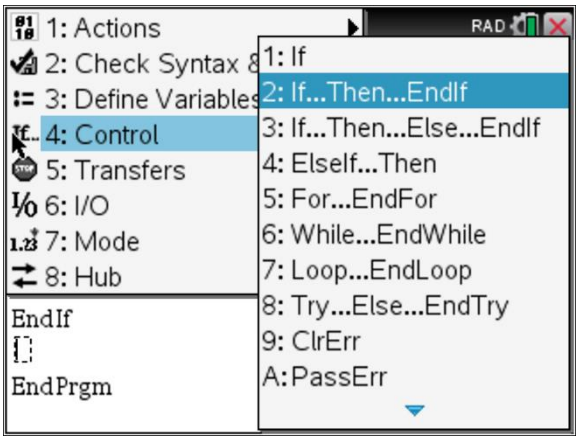
This flowchart illustrates the decision making process.



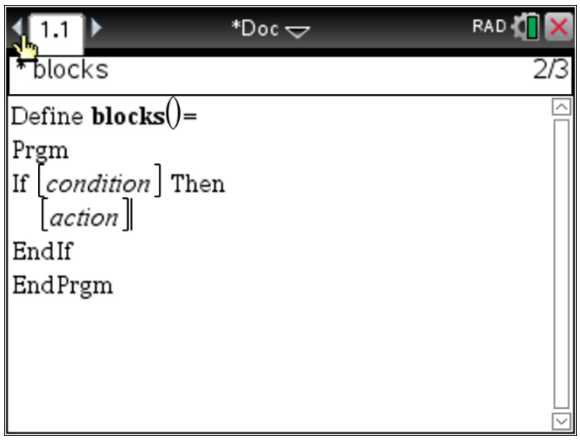
Selection Structure

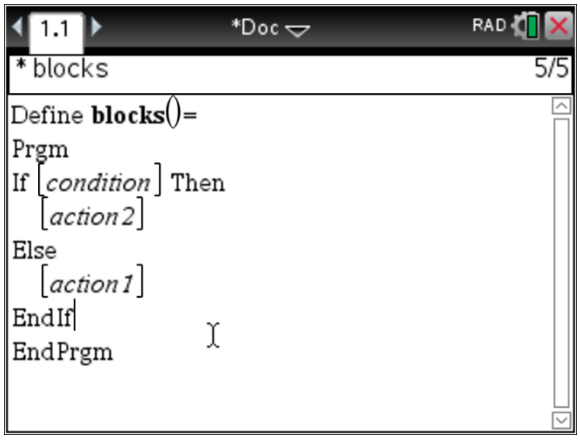
The two most basic conditional statements in the TI-basic programming language are:

- On the handheld, click (menu).
- Click **Control**



- Click **If** or **If…Then…Endif**



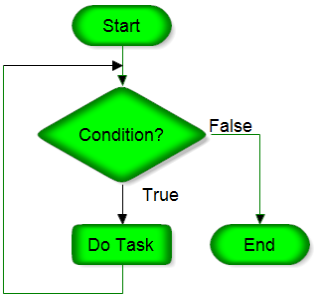


2. LOOPS

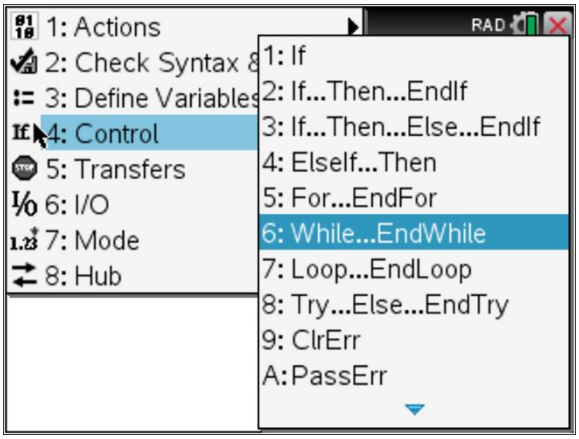
Loops are used to repeat blocks of code. Whenever you want to repeat a block of code, the question is: do you know how many times ? Is it a fixed number of times or does it depend on a certain condition to be met ?

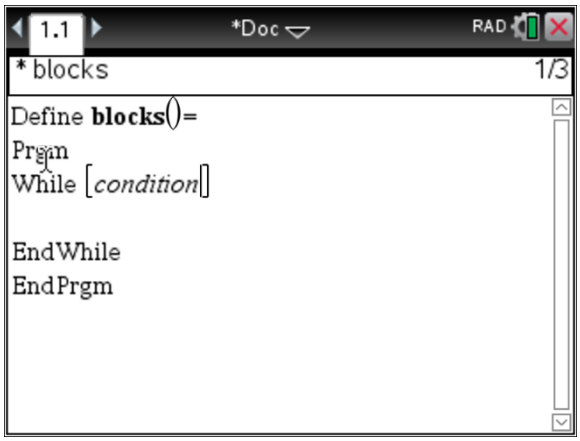
1) WHILE… - LOOP

A WHILE…-loop is used to repeat a block of code as long as some true-or-false condition is met. This condition is checked before entering the loop (if it's false to begin with, the loop is skipped), and checked again every time the loop ends.



- On the handheld, click (menu).
- Click **Control**
- Click **While…Endwhile**



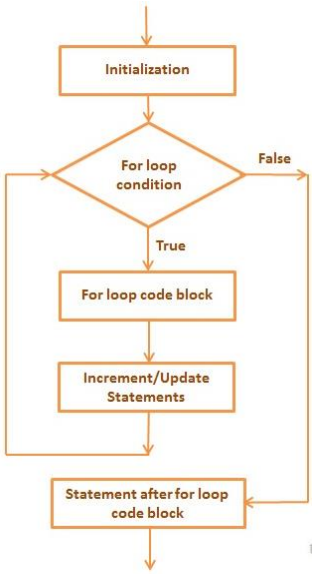


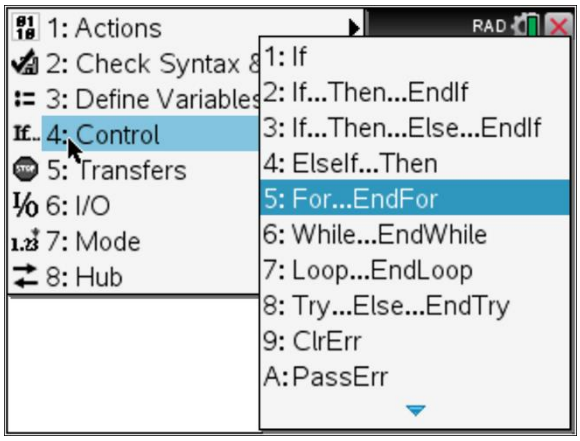
2) FOR… - LOOP

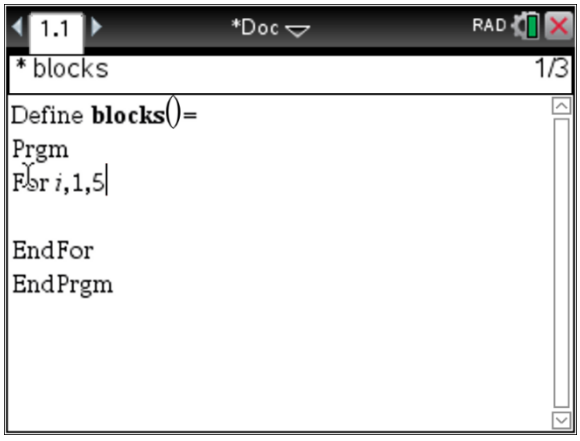
A FOR..-loop is used to repeat the code inside the block some number of times. To specify the number of times to repeat this code, a counter variable is specified, along with a starting value and an ending value. The variable will be set to the

starting value, then increased until it gets to the ending value, running the code inside the block each time.

- On the handheld, click (menu).
- Click **Control**
- Click **For…Endfor**

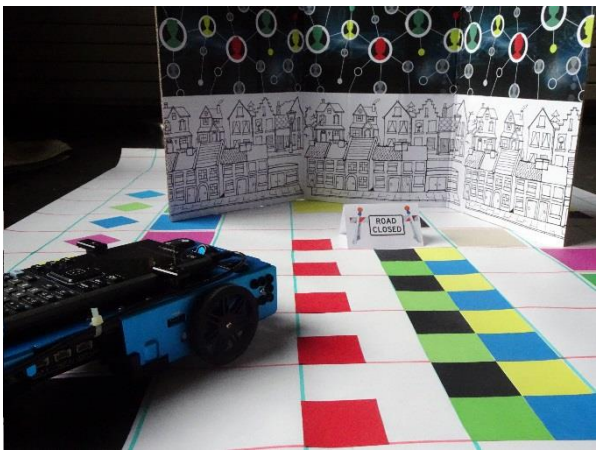




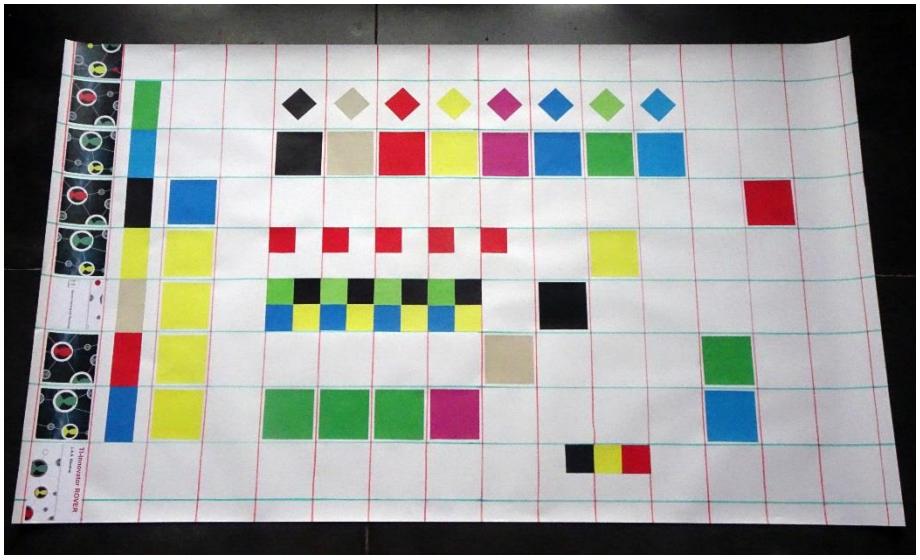


5. ROVER GRIDS

To make the programming of the Rover even more exciting and challenging, I have created 4 different 1.5m x 0.9m Rover grids to inspire and enrich the problem set. It makes coding a lot more fun.



6. ROVER AND THE MONDRIAN GRID



	Project	File Name .TNS	Level	Sensors		LED
				Distance	Colour	
1.	Rover_S()	ROVER	★	-	-	-
2.	Rover_S2(size)	ROVER	★★	-	-	-
3.	Rover_M()	ROVER	★	-	-	-
4.	Square(side)	POLYGON	★	-	-	-
5.	Rectangle(l,w)	POLYGON	★	-	-	-
6.	Polygon(n,side)	POLYGON	★★	-	-	-
7.	Distance()	DISTANCE	★	X	-	-
8.	Distance2()	DISTANCE	★★	X	X	-
9.	To_obst(dis)	DISTANCE	★	X	-	-
10.	Lap(s)	DISTANCE	★	X	-	-
11.	Colours(dis)	COLOUR	★	X	X	-
12.	Countcol(u,dis)	COLOUR	★	X	X	-
13.	Find_colour(co,dis)	COLOUR	★★★	X	X	X
14.	Car(x,y)	CARS	★★	-	-	-
15.	Route(x,y)	CARS	★★★★	X	-	X

1. Rover_S()

The Rover drives an S-shaped path using a fixed distance of 2 units for each length.

- 1 unit = 10 cm (default value)

```
rover_s
Define rover_s()=
Prgm
Send "CONNECT RV"
DispAt 1,"drive S"
Send "RV FORWARD 2 "
Send "RV LEFT "
Send "RV FORWARD 2 "
Send "RV LEFT "
Send "RV FORWARD 2"
Send "RV RIGHT "
Send "RV FORWARD 2"
Send "RV RIGHT "
Send "RV FORWARD 2"
EndPrgm
```

2. Rover_S2(size)

The Rover drives a S-shaped path using a variable to increase or decrease the size. (parameter: *size* in meters).

- RV FORWARD SPEED
  Use the SPEED 0.15 M/S option at least once instead of the default DISTANCE.
- Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec.

```
rover_s2
Define rover_s2(size)=
Prgm
Send "CONNECT RV"
DispAt 1,"variable S"
Send "RV FORWARD ",size," M"
Send "RV LEFT "
Send "RV FORWARD ",size," M"
Send "RV LEFT "
Send "RV FORWARD ",size," M"
Send "RV RIGHT "
Send "RV FORWARD SPEED .15 M/S ",size/0.15
Send "RV RIGHT 90 DEGREES"
Send "RV FORWARD ",size," M"
EndPrgm
```

3. Rover_M()

The Rover drives a M-shaped path using the coordinates in the XY plane (in units).

- RV TO XY X,Y
  Moves Rover from the current grid location to the specified grid location.
  At the start of the program execution the location is (0,0) with Rover facing the positive X-axis.

```
rover_m
Define rover_m()=
Prgm
Send "CONNECT RV"
Send "RV TO XY 1 −1"
Send "RV TO XY 1 4"
Send "RV TO XY 3 2"
Send "RV TO XY 5 4"
Send "RV TO XY 5 −1"
EndPrgm
```

4. square(side)

The Rover drives a square with a variable (parameter: *side*) side of *side* meter.

- FOR …ENDFOR
  Use the symmetry of the square.

```
square
Define square(side)=
Prgm
Send "CONNECT RV"
For i,1,4
  Send "RV FORWARD ",side," UNIT"
  Send "RV LEFT "
EndFor
EndPrgm
```

5. rectangle(l,w)

The Rover drives an rectangle with a variable length (parameter: l) and width (parameter: w) in units.

- FOR …ENDFOR
  Use the symmetry of the rectangle.

```
rectangle
Define rectangle(l,w)=
Prgm
Send "CONNECT RV"
For i,1,2
  Send "RV FORWARD ",l," UNIT"
  Send "RV RIGHT "
  Send "RV FORWARD ",w," UNIT"
  Send "RV RIGHT "
EndFor
EndPrgm
```

6. polygon(n,side)

The Rover drives a n-polygon with a variable side (parameters: n and side in meters).

- FOR …ENDFOR
  Use the symmetry of the polygon.

```
polygon
Define polygon(n,side)=
Prgm
Local i
DispAt 1,n,"−gon"
Send "CONNECT RV"
For i,1,n
  Send "RV FORWARD ",side," M"
  Send "RV RIGHT ",360/n," DEGREES"
EndFor
EndPrgm
```

- Can the students work out the formula for the angle?

7. distance()

The vehicle does not move at all. Put an obstacle in front of the Rover and move the obstacle. The Rover displays on the handheld the changing distance to the obstacle in meters. The program ends when this distance is less than 10 cm.

```
distance                                    8/8
Define distance()=
Prgm
Send "CONNECT RV"
Send "READ RV.RANGER"
Get d
 While d>0.1
  DispAt 1,d
  Send "READ RV.RANGER"
  Get d
EndWhile
EndPrgm
```

- READ RV RANGER
  Get d
  With the ultrasonic distance sensor, the Rover measures the distance in meters. 10 m means no obstacle was detected.

8. Distance2()

The vehicle does not move at all. Put an obstacle in front of the Rover and move the obstacle. The LED colour changes according to the distance to the obstacle.

- Orange: > 80 cm
- Red: 60-80 cm
- Blue: 40-60 cm
- Green: 20-40 cm

The Rover displays on the handheld the changing distance to the obstacle and the matching colour. The program ends when this distance is less than 10 cm.

```
distance2                                   9/26
Define distance2()=
Prgm
Send "CONNECT RV"
Send "READ RV.RANGER"
Get d
DispAt 1,d
 While d>0.1
  Send "READ RV.RANGER"
  Get d
  If d<0.2 Then
    Send "SET RV.COLOR.GREEN 255"
    DispAt 1,d," = green"
  Else
    If d<0.4 Then
      Send "SET RV.COLOR.BLUE 255"
      DispAt 1,d," = blue"
    Else
      If d<0.8 Then
        Send "SET RV.COLOR.RED 255"
        DispAt 1,d," = red"
      Else
        Send "SET RV.COLOR 200 100 0"
        DispAt 1,d," = orange"
      EndIf
    EndIf
  EndIf
EndWhile
Send "SET RV.COLOR 0 0 0"
EndPrgm
```

- SET RV.COLOR.GREEN gg
  Set the green colour to be displayed on the Rover's RGB led.
- SET RV.COLOR.BLUE bb
- SET RV.COLOR.RED rr
- SET RV.COLOR rr gg bb
  Set the RGB colour to be displayed on the Rover's RGB led.

9. To_obst(dis)

The Rover measures the distance to the obstacle and starts moving in a straight line towards it. All the time displaying the actual distance. The program ends when the distance to the obstacle is less than (parameter: *dis*) *dis* meter.

```
to_obst
Define to_obst(dis)=
Prgm
Send "CONNECT RV"
DispAt 2,"Put an obstacle in front of Rover"
Wait 5
Send "READ RV.RANGER"
Get d
DispAt 2,"Original distance =",d," m"
While d>dis
  DispAt 3,"Actual distance =",round(d,2)," m"
  Send "RV FORWARD TIME .25 S"
  Wait 1
  Send "READ RV.RANGER"
  Get d
EndWhile
DispAt 4,"Obstacle found!"
EndPrgm
```

- WHILE …ENDWHILE
  The number of steps forward onwards the obstacle is unknown as it depends on the distance.
- WAIT
  The forward movement has to be completed before the distance is measured again.

10. lap(s)

As the Rover is driving, the vehicle follows the perimeter of the box of the grid at a distance of 20 cm, to finish one complete lap. The Rover should end exactly at its starting location.

- FOR …ENDFOR
  The number of steps is 4.
- Use list to store the distances travelled.

```
lap
Define lap(s)=
Prgm
m:={0,0,0,0}
If s="l" or s="r" Then
  Send "CONNECT RV"
  For i,1,4
    Send "READ RV.RANGER"
    Get d
    DispAt 2,d
    m[i]:=d-0.2
    Send "RV FORWARD ",d-0.2," M"
    If s="l" Then
      Send "RV LEFT "
    Else
      Send "RV RIGHT "
  EndIf
  Wait 7
  EndFor
  Send "READ RV.RANGER"
  Get d
  Send "RV FORWARD ",m[3]-m[1]," M"
  DispAt 2,"finished the lap!",m
Else
  DispAt 2,"error parameter:only l or r"
EndIf
EndPrgm
```

11. colours(dis)

The Rover detects the colour and starts moving in a straight line towards the obstacle. All the time displaying the actual colour. The program ends when the distance to the obstacle is less than (parameter: *dis*) *dis* meter.

- WHILE …ENDWHILE
  The number of steps forward onwards the obstacle is unknown as it depends on the distance.
- RV FORWARD TIME n S
  Default speed is 0.2 m/sec.
- Use a list to store the colours.

```
colours
Define colours(dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 2,"Put an obstacle in front of Rover"
Wait 3
Send "READ RV.RANGER"
Get d
While d>dis
   Send "READ RV.COLORINPUT"
   Get c
   DispAt 3,"colour =",c," - ",col[c]
   Send "RV FORWARD TIME 0.5 S"
   Wait 2
   Send "READ RV.RANGER"
   Get d
EndWhile
EndPrgm
```

12. countcol(u,dis)

The Rover detects the colour and starts moving in a straight line in steps of one unit towards the obstacle, while displaying the actual colour found. The vehicle also displays and counts how many times each colour is detected.
The program ends when the distance to the obstacle is less than (parameter: *dis*) *dis* meter)
The first parameter u (in meter) is the new unit to be used.

- RV.GRID.M/UNIT
  Sets the new unit (in meters)
- RV FORWARD
  Default is unit.
- Use a list to store the colours and how many times a colour is found.

```
countcol
Define countcol(u,dis)=
Prgm
Local u,d,c,col,nrs,shortcol
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
shortcol:={red,gre,blucya,mag,yel,bla,whi,gry}
nrs:={0,0,0,0,0,0,0,0,0}
Send "CONNECT RV"
Send "SET RV.GRID.M/UNIT ",u
DispAt 2,"Put an obstacle in front of Rover"
Wait 3
Send "READ RV.RANGER"
Get d
DispAt 1,shortcol
While d>dis
  Send "READ RV.COLORINPUT"
  Get c
  nrs[c]:=nrs[c]+1
  DispAt 2,nrs
  DispAt 3,"colour =",c," - ",col[c]
  Send "RV FORWARD 1 "
  Wait 3
  Send "READ RV.RANGER"
  Get d
EndWhile
EndPrgm
```

13. Find_colour(co,dis)

The first parameter *co* sets the colour to be found.
The Rover checks the colours while moving in a straight line in small steps towards the obstacle.
The vehicle also reads the colour sensors to match the RGB led to the colour detected.
The program ends when the distance to the obstacle is less than (parameter: *dis*) *dis* meter) or when the colour *co* is found.

- READ RV.COLORINPUT.RED
- READ RV.COLORINPUT.GREEN
- READ RV.COLORINPUT.BLUE
- SET RV.COLOR
- Use a list to store the colours.

```
find_colour
Define find_colour(co,dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 2,"Put an obstacle in front of Rover"
Wait 3
Send "READ RV.RANGER"
Get d
found:=false
While d>dis and not found
  Send "READ RV.COLORINPUT"
  Get c
  If c=co Then
    DispAt 2,"found colour =",c,col[c]
    found:=true
  Else
    DispAt 2,"colour =",c," - ",col[c]
    Send "RV FORWARD TIME .15 S"
    Wait 1
    Send "READ RV.RANGER"
    Get d
  EndIf
  Send "READ RV.COLORINPUT.RED"
  Get rr
  Send "READ RV.COLORINPUT.GREEN"
  Get gg
  Send "READ RV.COLORINPUT.BLUE"
  Get bb
  Send "SET RV.COLOR ",rr,gg,bb
EndWhile
If not found Then
  DispAt 2,"colour ",co,col[co]," not found"
EndIf
Wait 4
Send "SET RV.COLOR 0 0 0"
EndPrgm
```

14. Car(x,y)

In this program the vehicle drives to the point P(x,y) in the XY plane. Both coordinates are given as parameters of the function.
The Rover moves first along the x-axis.

- GOTOXY
  Can find out the difference between this program and the GOTOXY command?

```
car
Define car(x,y)=
Prgm
Local x,y
Send "CONNECT RV"
If x>0 Then
  Send "RV FORWARD ",x· 0.1," M"
Else
  Send "RV BACKWARD ",-x· 0.1," M"
EndIf
Send "RV LEFT "
If y>0 Then
  Send "RV FORWARD ",y· 0.1," M"
  Else
   Send "RV BACKWARD ",-y· 0.1," M"
EndIf
EndPrgm
```

15. Route(x,y)

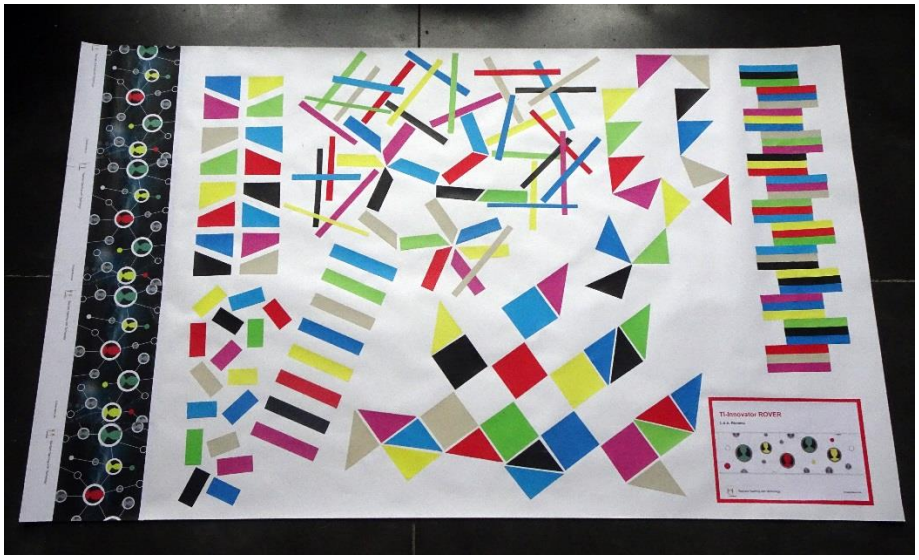
Route(x,y) is a more advanced version of the Car(x,y) program. Both coordinates of the point of destination are given as parameters of the function.
In this case the two main roads, the x-axis and/or the y-axis can be blocked because of road works.
The Rover always starts to drive along the x-axis and then continues parallel to the y-axis.
If there are no roadblocks on the way, the vehicle reaches its point of destination.
If x-axis road is blocked along the way, Rover goes for the alternative strategy: moving first along the y-axis and then continuing parallel to the x-axis to get to the destination.
If both roads x-axis and y-axis are blocked, the vehicle does not find a route to the destination and displays a message.
A blocked x-axis road is also indicated by a blinking RGB led on the vehicle and a blocked y-axis road by a blinking blue RGB led.

- Road blocked
  This is not a problem for the Rover if the vehicle does not have to pass by this point.

```
route
Define route(x,y)=
Prgm
Local x,y,blockx,blocky
x:=x· 0.1
y:=y· 0.1
blockx:=false
blocky:=false
If x>0 Then
  Send "READ RV.RANGER"
  Get d
  Wait 1
  If x<d Then
    Send "RV FORWARD DISTANCE ",x," M"
    If y>0 Then
      Send "RV LEFT "
      Send "RV FORWARD DISTANCE ",y," M"
    Else
      If y<0 Then
        Send "RV RIGHT "
        Send "RV FORWARD DISTANCE ",-y," M"
      EndIf
    EndIf
  Else
    blockx:=true
    DispAt 1,"positive x-axis blocked"
    Send "SET RV.COLOR.RED 150 BLINK 9 3"
  EndIf
Else
  If x<0 Then
    Send "RV LEFT 180 DEGREES"
    Wait 3
    Send "READ RV.RANGER"
    Get d
    Wait 1
    If -x<d Then
      Send "RV FORWARD ",-x," M"
      If y>0 Then
        Send "RV RIGHT "
        Send "RV FORWARD ",y," M"
      Else
        Send "RV LEFT "
        Send "RV FORWARD ",-y," M"
      EndIf
  Else
```

```
    blockx:=true
    DispAt 1,"negative x−axis blocked"
    Send "SET RV.COLOR.RED 150 BLINK 9 3"
  EndIf
  EndIf
EndIf
If blockx and y≠0 Then
  If x<0 Then
    Send "RV RIGHT 180 DEGREES"
    Wait 2
  EndIf
  If y>0 Then
    Send "RV LEFT "
    Wait 2
    Send "READ RV.RANGER"
    Get d
    Wait 1
    If y<d Then
      Send "RV FORWARD ",y," M"
      If x>0 Then
        Send "RV RIGHT "
        Send "RV FORWARD ",x," M"
      Else
        Send "RV LEFT "
        Send "RV FORWARD ",-x," M"
      EndIf
    Else
      blocky:=true
      DispAt 2,"positive y−axis blocked"
      Send "SET RV.COLOR.BLUE 150 BLINK 9 3"
    EndIf
  Else
    Send "RV RIGHT "
    Wait 2
    Send "READ RV.RANGER"
    Get d
    Wait 1
    If -y<d Then
      Send "RV FORWARD ",-y," M"
      If x>0 Then
        Send "RV LEFT "
        Send "RV FORWARD ",x," M"
      Else
        Send "RV RIGHT "
        Send "RV FORWARD ",-x," M"
      EndIf
    Else
      blocky:=true
      DispAt 2,"negative y−axis blocked"
      Send "SET RV.COLOR.BLUE 150 BLINK 9 3"
    EndIf
  EndIf
EndIf
If blockx and blocky Then
  DispAt 3,"alternative route needed"
Else
  DispAt 3,"arrived at destination"
EndIf
EndPrgm
```

7. ROVER AND THE EUCLID GRID



	Project	File Name .TNS	Level	Sensors		LED
				Distance	Colour	
1.	Colours(dis)	COLOUR	★	X	X	-
2.	Countcol(u,dis)	COLOUR	★	X	X	-
3.	Find_colour(co,dis)	COLOUR	★★★	X	X	X
4.	Bl_poly(n,side)	LIGHT	★★	-	-	X
5.	Max_poly(n,side)	LIGHT	★★★	-	-	X
6.	Col_circle(co,r,dis)	COLOURGAMES	★★★★	X	X	X
7.	Col_walk(dis)	COLOURGAMES	★★★★	X	X	X
8.	Randwalk(steps,dis)	COLOURGAMES	★★★★	X	X	X

1. **Colours(dis)**
   The program has already been developed for the Mondrian grid. This grid provides interesting new colour schemes. Maybe the program needs some fine-tuning: smaller step size, etc.?

2. **Countcol(u,dis)**
   The program has already been developed for the Mondrian grid. This grid provides interesting new colour schemes. Maybe the program needs some fine-tuning: smaller step size, etc.?

3. **Find_colour(co,dis)**
   The program has already been developed for the Mondrian grid. This grid provides interesting new colour schemes. Maybe the program needs some fine-tuning: smaller step size, etc.?

4. Bl_poly(n,side)

The Rover drives a n-polygon with a variable side (parameters: *n* and *side* in meters).
Every side is divided in 10 steps. At every location on the path, the vehicle checks for the RGB colours (red, green and blue). If the colour is found, the RGB led of the Rover starts blinking in the corresponding colour.

- FOR …ENDFOR
  Use the symmetry of the polygon and the 10 steps.

```
bl_poly
Define bl_poly(n,side)=
Prgm
Local i
DispAt 1,n,"-gon / blinking R G B"
Send "CONNECT RV"
For i,1,n
  For k,1,10
    Send "READ RV.COLORINPUT"
    Get c
    If c=1 Then
      Send "SET RV.COLOR.RED 150 BLINK 4 1 S"
      Wait 1
    Else
      If c=2 Then
        Send "SET RV.COLOR.GREEN 150 BLINK 5 1 S"
        Wait 1
      Else
        If c=3 Then
          Send "SET RV.COLOR.BLUE 150 BLINK 6 1 S"
          Wait 1
        EndIf
      EndIf
    EndIf
    Send "RV FORWARD ",side· 0.1," M"
    Wait 1.5
  EndFor
  Send "RV RIGHT ",360/n," DEGREES"
  Wait 2
EndFor
EndPrgm
```

5. Bl_poly(n,side)

The Rover drives a n-polygon with a variable side (parameters: *n* and *side* in meters).
Every side is divided in 20 steps. At every location on the path, the vehicle checks for the RGB colours (red, green and blue).
The RGB led of the Rover shows at every location the most popular colour so far.

- FOR …ENDFOR
  Use the symmetry of the polygon and the 20 steps.

```
max_poly
Define max_poly(n,side)=
Prgm
Local i,m,k,rgb
DispAt 1,n,"-gon /  looking for R G B"
Send "CONNECT RV"
rgb:={0,0,0}
m:=1
For i,1,n
  For k,1,20
    Send "READ RV.COLORINPUT"
    Get c
    If c=1 or c=2 or c=3 Then
      rgb[c]:=rgb[c]+1
    EndIf
    DispAt 2,rgb
    If sum(rgb)>0 Then
      If rgb[1]≥rgb[2] Then
        m:=1
      Else
        m:=2
      EndIf
```

```
    If rgb[3]>rgb[m] Then
      m:=3
    EndIf
    If m=1 Then
      Send "SET RV.COLOR.RED 150 1 1 S"
    Else
      If m=2 Then
        Send "SET RV.COLOR.GREEN 150 1 1 S"
      Else
        Send "SET RV.COLOR.BLUE 150 1 1 S"
      EndIf
    EndIf
  EndIf
  Send "RV FORWARD ",side·0.05," M"
  Wait 1
 EndFor
 Send "RV RIGHT ",360/n," DEGREES"
 Wait 2
EndFor
EndPrgm
```

6. Col_circle(co,r,dis)

The Rover tries to find a spot with a variable colour *co* (parameters: *r* = radius and *dis* in meters and *co* = colour).
The search area of the vehicle tries is a circle with radius r. It moves a maximum distance r from the centre of the circle in steps of 3 cm. If the colour has not been detected it goes back to the centre and turns 30 degrees and starts again. This process is repeated until the colour is found or the circle is completely covered. Getting within a distance dis of an obstacle ends the program immediately.
The RGB led of the Rover shows at every location the corresponding colour found.

```
col_circle
Define col_circle(co,r,dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 2,"Put an obstacle in front of Rover"
Wait 3
Send "READ RV.RANGER"
Get d
found:=false
ang:=0
DispAt 2,"searching for the colour ",col[co]
While d>dis and not found and ang<360
 st:=0
 While st·0.03<r and not found and d>dis
  Send "READ RV.COLORINPUT"
  Get c
  If c=co Then
    DispAt 4,"found colour =",c,col[c]
    found:=true
  Else
    DispAt 3,"colour =",c," - ",col[c]
    Send "RV FORWARD DISTANCE 0.03 M"
    st:=st+1
    Wait 1
    Send "READ RV.RANGER"
    Get d
  EndIf
  Send "READ RV.COLORINPUT.RED"
  Get rr
  Send "READ RV.COLORINPUT.GREEN"
  Get gg
  Send "READ RV.COLORINPUT.BLUE"
  Get bb
  Send "SET RV.COLOR ",rr,gg,bb
  Wait 0.5
 EndWhile
 If not found Then
  Send "RV BACKWARD DISTANCE",0.03·st," M"
  ang:=ang+30
  Send "RV RIGHT 30 DEGREES"
  Wait st/2
 EndIf
EndWhile
If not found Then
  DispAt 4,"colour ",co,col[co]," not found"
EndIf
Wait 4
Send "SET RV.COLOR 0 0 0"
EndPrgm
```

7. Col_walk(dis)

The purpose of the game is that the Rover makes as many steps (in any direction) as possible before it is stopped by an obstacle within distance *dis* (parameter) meter.
All relevant information concerning the game is displayed in real-time.
Rules of the game:

- ✓ *Rover moves forward in steps of 2 cm until it finds another colour, except white or grey.*
- ✓ *Rover hits black: game lost.*
- ✓ *Rover hits grey: 2 steps lost.*
- ✓ *Rover hits red, green, blue, cyan, magenta or yellow: turn right 45 degrees times the corresponding colour number.*

```
col_walk
Define col_walk(dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
st:=0
old:=0
Send "READ RV.RANGER"
Get d
lost:=false
While d>dis and not lost
  Send "READ RV.COLORINPUT"
  Get c
  DispAt 3,"colour =",c," - ",col[c]
  DispAt 5," "
  If c≠old Then
  If c=7 Then
     lost:=true
     DispAt 5,"You lost, hitting BLACK!"
  Else
    If c≤6 Then
      Send "RV RIGHT ",c·45," DEGREES"
      DispAt 5,"Still alive…"
    Else
      If c=9 Then
        st:=st-2
        DispAt 5,"You lost 2 steps, hitting GRAY."
      EndIf
    EndIf
  EndIf
  EndIf
  Send "RV FORWARD DISTANCE .02 M"
  old:=c
  st:=st+1
  DispAt 4,"random steps =",st
  Wait 1
  Send "READ RV.RANGER"
  Get d
EndWhile
If not lost Then
  DispAt 5,"Stopped by an obstacle"
EndIf
EndPrgm
```

8. randwalk(steps,dis)

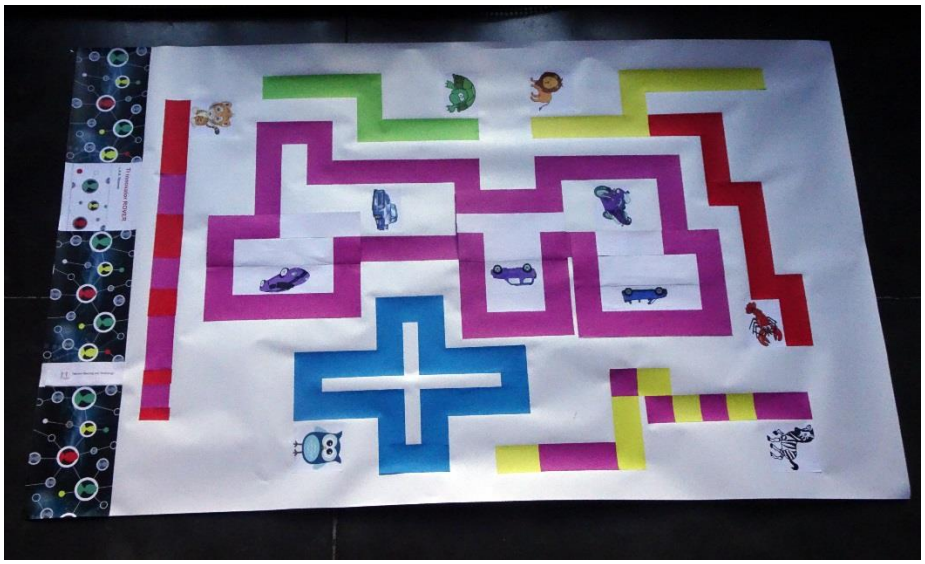
The purpose of the game is that the Rover tries to collect all colours before it is stopped by an obstacle within distance *dis* (parameter) meter.
All relevant information concerning the game is displayed in real-time.
Rules of the game:

- ✓ *Rover moves a number of times (parameter steps) forward, each step 2 cm.*
- ✓ *Next, a random turning angle between -90 and 90 degrees is chosen.*
- ✓ *After every step the vehicle checks the colour at the current location to eliminate it from the list.*

```
randwalk
Define randwalk(steps,dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
missing:={red,gre,blu,cya,mag,yel,bla,whi,gry}
found:={0,0,0,0,0,0,0,0,0}
RandSeed 19550429
nr:=0
Send "CONNECT RV"
Send "READ RV.RANGER"
Get d
While d>dis and product(found)=0
  Send "READ RV.COLORINPUT"
  Get c
  Wait 0.5
  DispAt 4,"recent colour =",c," - ",col[c]
  found[c]:=found[c]+1
  missing[c]:=_
  DispAt 1,missing
  Send "RV FORWARD TIME .2 S"
  If mod(nr,steps)=1 Then
    ang:=randInt(-90,90)
    Send "RV RIGHT ",ang," DEGREES"
```

```
    DispAt 3,"turning ",ang,"degrees"
    Wait 4
  Else
    Wait 2
  EndIf
  nr:=nr+1
  Send "READ RV.RANGER"
  Get d
EndWhile
DispAt 5,nr-1," steps"
If product(found)>0 Then
    DispAt 6,"You won. All colours found."
Else
    DispAt 6,"You lost. Obstacle found."
EndIf
EndPrgm
```

8. ROVER AND THE RACING GRID



	Project	File Name .TNS	Level	Sensors		LED
				Distance	Colour	
1.	Turtle(dis)	PATH	★★	X	X	-
2.	Lion(dis)	PATH	★★	X	X	-
3.	Lobster(first,dis)	PATH	★★★	X	X	-
4.	Owl(first,dis)	PATH	★★	X	X	-
5.	Cars(dis)	PATH	★★★	X	X	-
6.	Tiger(dis)	-	★★	X	X	-
7.	Zebra(dis)	-	★★★	X	X	-

1. Turtle(dis)

The Rover is placed on the green path and follows it. The vehicle has to make exactly two turns: first 90 degrees right and the second time 90 degrees left.

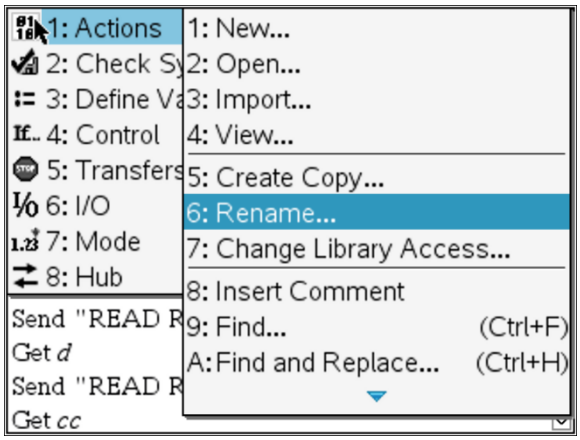
- The Rover moves slowly to check the colours and distance at every step.
- The turtle keeps moving forward until it leaves the yellow path. That means there is a turn to be made.
- A small correction was needed before making the turn: moving backward only 0.5 cm

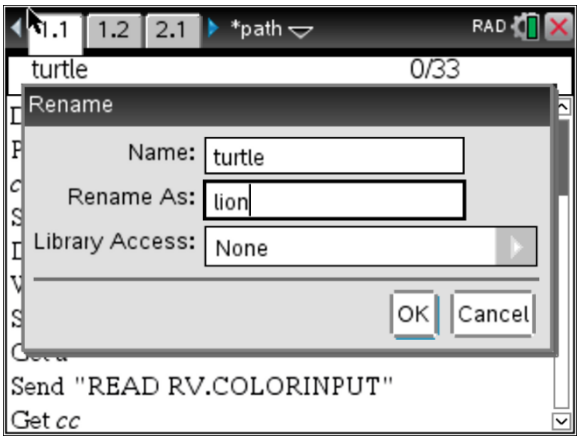
```
turtle
Define turtle(dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 1,"Put an obstacle at the end of the path"
Wait 3
Send "READ RV.RANGER"
Get d
Send "READ RV.COLORINPUT"
Get cc
DispAt 3,"following ",col[cc]," path"
c:=cc
turn:=0
While d>dis
  Send "RV FORWARD .01 M"
  Wait 1
  Send "READ RV.COLORINPUT"
  Get c
  If c≠cc Then
    Send "RV BACKWARD 0.005 M"
    If turn=0 Then
      Send "RV RIGHT "
    Else
      Send "RV LEFT "
    EndIf
    Send "RV FORWARD 0.02 M"
    Wait 1.5
    turn:=turn+1
  EndIf
  Send "READ RV.COLORINPUT"
  Get c
  Send "READ RV.RANGER"
  Get d
EndWhile
DispAt 2,"Found end of path"
EndPrgm
```

2. lion(dis)

The Rover is placed on the yellow path and follows it. The vehicle has to make exactly two turns: first 90 degrees left and the second time 90 degrees right.

- The Rover moves slowly to check the colours and distance at every step.
- The lion keeps moving forward until it leaves the yellow path. That means there is a turn to be made.
- A small correction was needed before making the turn: moving backward only 0.5 cm
- Since this program is very similar to the turtle() you could use a copy and change the code slightly.
- To change the name of the program:
  - On the handheld, click (menu).
  - Click **Actions**
  - Click **Rename**
  - Type a new name: lion
  - Press **Enter**






```
lion
Define lion(dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 1,"Put an obstacle at the end of the path"
Wait 3
Send "READ RV.RANGER"
Get d
Send "READ RV.COLORINPUT"
Get cc
DispAt 3,"following ",col[cc]," path"
c:=cc
turn:=0
While d>dis
  Send "RV FORWARD .01 M"
  Wait 1
  Send "READ RV.COLORINPUT"
  Get c
  If c≠cc Then
    Send "RV BACKWARD 0.005 M"
    If turn=1 Then
      Send "RV RIGHT "
    Else
      Send "RV LEFT "
    EndIf
    Send "RV FORWARD 0.02 M"
    Wait 1.5
    turn:=turn+1
  EndIf
  Send "READ RV.COLORINPUT"
  Get c
  Send "READ RV.RANGER"
  Get d
EndWhile
DispAt 2,"Found end of path"
EndPrgm
```

3. lobster(first,dis)

There are two new challenges in this program compared to the turtle().
Depending on which end of the red part the Rover starts, the first turn to be made can be left or right.
The total number of turns to be made before reaching the obstacle at the end is unknown.

- The parameter first can be “l” or “r”, indicating the type of the first turn to be made.
- No other values can be accepted for this parameter when launching the program. An error message should be displayed and the program halts immediately.

```
lobster
Define lobster(first,dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 1,"Put an obstacle at the end of the path"
Wait 3
Send "READ RV.RANGER"
Get d
Send "READ RV.COLORINPUT"
Get cc
DispAt 3,"following ",col[cc]," path"
c:=cc
err:=false
If first="r" Then
  turn:=1
Else
  If first="l" Then
    turn:=0
  Else
    err:=true
  EndIf
EndIf
```

4. owl(first,dis)

The lobster() program should also work perfectly for this path.
Depending on the location on the blue path and the chosen direction, the first turn to be made can be left or right.
Without on obstacle on the path the lobster() program would run forever.
With an obstacle this can be easily arranged.

- If you write a custom made version of owl() make sure to use the symmetry of the path.

```
While d>dis and not err
  Send "RV FORWARD .01 M"
  Wait 1
  Send "READ RV.COLORINPUT"
  Get c
  If c≠cc Then
  Send "RV BACKWARD 0.005 M"
    If mod(turn,2)=1 Then
      Send "RV RIGHT "
    Else
      Send "RV LEFT "
    EndIf
    Send "RV FORWARD 0.02 M"
    Wait 1.5
    turn:=turn+1
  EndIf
  Send "READ RV.COLORINPUT"
  Get c
  Send "READ RV.RANGER"
   DispAt 2,"error parameter: l or r"
Else
   DispAt 2,"Found end of path"
EndIf
EndPrgm
```

5. cars(dis)

Cars() is the general solution to the path problem. It does everything the other programs, turtle(), lion() or lobster() do and more.
No need to specify the type of the first turn. Left and right turns can follow each other in any sequence possible.

- The Rover always starts with a left turn. Then it checks to see if the path goes on into that direction. If not the vehicle turns around 180 degrees to continue in the opposite direction.

```
  Send "RV LEFT "
  Send "RV FORWARD 0.02 M"
  Wait 2
  Send "READ RV.COLORINPUT"
  Get c
  If c≠cc Then
    Send "RV BACKWARD 0.06 M"
    Send "RV RIGHT 180 DEGREES"
    r:=r+1
    DispAt 4,"turning right: ",r," times"
    Wait 1.5
  Else
    l:=l+1
    DispAt 5,"turning left: ",l," times"
  EndIf
  Send "READ RV.RANGER"
  Get d
 EndIf
EndWhile
DispAt 6,"Found end of path"
EndPrgm
```

```
cars
Define cars(dis)=
Prgm
col:={red,green,blue,cyan,magenta,yellow,black,white,gray}
Send "CONNECT RV"
DispAt 1,"Put an obstacle at the end of the path"
Wait 3
Send "READ RV.RANGER"
Get d
Send "READ RV.COLORINPUT"
Get cc
DispAt 3,"following ",col[cc]," path"
c:=cc
l:=0
r:=0
While d>dis
  Send "RV FORWARD .01 M"
  Wait 1.5
  Send "READ RV.RANGER"
  Get d
  Send "READ RV.COLORINPUT"
  Get c
  If c≠cc Then
```

- The program keeps track of the number of left and right turns the Rover makes and displays these results in real time.

6. **tiger(dis)**

   This program follows a straight path marked with two different colours. The program ends when an obstacle is within reach or the colour changes again at the end of the path.
   - The Rover should detect automatically the two colours used.
   - There are no turns in this program.

7. **zebra(dis)**

   This program takes tiger() to the next level by adding an unknown number of left and right turns in any order.

9. ROVER AND THE FLOWER CITY GRID

	Project	File Name .TNS	Level	Sensors		LED
				Distance	Colour	
1.	Vetdonkey()	CITY	★	-	-	-
2.	Vet() Vetsheep() Vetfarm()	CITY	★★	-	-	-
3.	Drone_lake(dis)	CITY	★★	X	X	X

The grid represents a complete city with the usual buildings, houses and shops and also a few surprises. All roads are drawn in black. All points of interest have a unique number of reference. Most of them have only one access point but some have more than one.

1. **Vetdonkey()**

Rover drives the veterinarian from her clinic to the donkey.

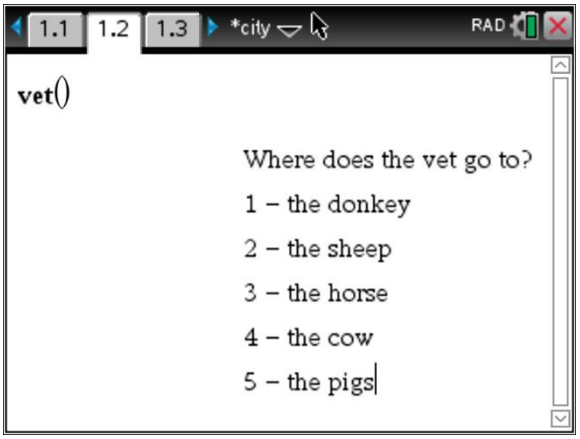
- The program starts with a list of possible destinations. Type the corresponding number to execute the selected route.

```
vetdonkey
Define vetdonkey()=
Prgm
Send "RV FORWARD 2"
Send "RV RIGHT 45 DEGREES"
Send "RV FORWARD 2.82"
Send "RV LEFT 135 DEGREES"
Send "RV FORWARD 3"
Send "RV LEFT"
Send "RV FORWARD .5"
Send "RV LEFT"
EndPrgm
```

2. **Vet()**

Rover drives the veterinarian from her clinic to the animals.
This is a nice opportunity to introduce the coding technique where a program uses another already existing program.

- The program starts with a list of possible destinations. Type the corresponding number to execute the selected route.




```
vet
Define vet()=
Prgm
opt:=""
Send "CONNECT RV"
DispAt 1,"Where does the vet go to?"
DispAt 2,"1 – the donkey"
DispAt 3,"2 – the sheep"
DispAt 4,"3 – the horse"
DispAt 5,"4 – the cow"
DispAt 6,"5 – the pigs"
opt:=getKey(1)
choice:=true
If opt="1" Then
  vetdonkey()
ElseIf opt="2" Then
  vetsheep()
ElseIf opt="3" Then
  vetfarm()
  Send "RV RIGHT "
  Send "RV FORWARD .5"
  Send "RV RIGHT "
  Send "RV FORWARD 1.5"
  Send "RV LEFT "
 ElseIf opt="4" Then
  vetfarm()
  Send "RV RIGHT "
  Send "RV FORWARD .5"
  Send "RV RIGHT "
  Send "RV FORWARD .5"
  Send "RV LEFT "
ElseIf opt="5" Then
  vetfarm()
  Send "RV RIGHT "
  Send "RV FORWARD .5"
  Send "RV LEFT "
  Send "RV FORWARD .5"
  Send "RV RIGHT "
Else
  DispAt 7,"foute keuze"
EndIf
EndPrgm
```

- If another key is pressed, an error message is displayed and the vehicle does not leave the clinic.
- Since the route **vetdonkey()** already exists, we want to use it in the vet() program.
- A problem can have more than one page and more than one program. Only programs inside the same problem are able to communicate (without touching the library).
- Code **vetsheep()** on a new page in the same problem.
- Some animals (the pigs, horse and cow) have a big part of the route in common. Write a separate program **vetfarm()** for this part on another page.

```
vetfarm
Define vetfarm()=
Prgm
Send "RV FORWARD 2"
Send "RV LEFT "
Send "RV FORWARD 2"
Send "RV LEFT 45 "
Send "RV FORWARD 2.82 "
Send "RV LEFT "
Send "RV FORWARD 1.41 "
Send "RV RIGHT 135 "
Send "RV FORWARD 4 "
Send "RV RIGHT 45 "
Send "RV FORWARD 1.41 "
Send "RV RIGHT 45 "
Send "RV FORWARD .5 "
Send "RV LEFT "
EndPrgm
```

```
vetsheep
Define vetsheep()=
Prgm
Send "RV FORWARD 2"
Send "RV LEFT "
Send "RV FORWARD 2"
Send "RV RIGHT 45 "
Send "RV FORWARD .7"
EndPrgm
```

3. Drone_lake(dis)

The Rover checks the colours while moving in a Z-shaped path in towards the lake or an obstacle.
The vehicle also reads the colour sensors to match the RGB led to the colour detected.
The program ends when the distance to the obstacle is less than (parameter: dis) *dis* meter) or when the lake is found.

- To be sure that the drone arrived at the lake at least two consecutive colour readings need to be blue.
- Design the Z-shape carefully.

```
drone_lake
Define drone_lake(dis)=
Prgm
Local c,co,c2,st
Send "CONNECT RV"
Send "READ RV.RANGER"
Get d
Wait 1
found:=false
co:=3
st:=0
While d>dis and not found
  Send "READ RV.COLORINPUT"
  Get c
  If c=co Then
    Send "RV FORWARD 0.02 M"
    Send "READ RV.COLORINPUT"
    Wait 3
    Get c2
    If c2=co Then
      DispAt 2,"drone arrived at the lake!"
      found:=true
    EndIf
  Else
    Send "RV FORWARD 0.015 M"
    Wait 1.5
    Send "READ RV.RANGER"
    Get d
    st:=st+1
    If st=5 Then
      Send "RV LEFT 60"
      Wait 1.5
    ElseIf mod(st,10)=0 Then
      Send "RV RIGHT 120"
      Wait 1.5
    ElseIf mod(st,10)=5 Then
      Send "RV LEFT 120"
      Wait 1.5
    EndIf
  EndIf
  Send "READ RV.COLORINPUT.RED"
  Get rr
  Send "READ RV.COLORINPUT.GREEN"
  Get gg
  Send "READ RV.COLORINPUT.BLUE"
  Get bb
  Send "SET RV.COLOR ",rr,gg,bb· 1.5
EndWhile
If not found Then
  DispAt 2,"drone lost, hitting an obstacle"
EndIf
Wait 4
Send "SET RV.COLOR 0 0 0"
EndPrgm
```

10. SOME INTERESTING LINKS to get the ROVER ready

- **Updating the TI Innovator HUB**

  To download the INNOVATOR HUB update program:
  https://education.ti.com/en/software/details/en/EFD1D3762FE941FAA21E774D8520AEF0/TI-Innovator_Hub_Update_SW

  To update SKETCH on the INNOVATOR HUB :
  https://education.ti.com/af/software/details/en/6DFDA25B4E12425F928D208950E7D78D/TI-Innovator_Sketch?sc_lang=nl-BE

   How to do the update : https://youtu.be/XXVCN9uAjiE

- **Connecting the TI innovator HUB with the TI innovator ROVER**

   link: https://youtu.be/WrZAhkWI-5E