

PROJECTSTUDIE

Programmeren van een microprocessor via ActiveX control in Delphi

Studenten:

Barandat Geert
Berckmans Olivier
Rimez Tim

Projectleider: J. Van Calck

Academiejaar 1999-2000

INHOUDSOPGAVE

Voorwoord
Inhoudsopgave

INHOUDSOPGAVE	2
VOORWOORD	5
1. COM OBJECTEN EN ACTIVEX CONTROLS.	6
1.1 Wat is COM.....	6
1.2 Wat is ActiveX?	7
1.3 Waarom ActiveX?.....	7
2. INTRODUCTIE VAN DE LEGO MINDSTORMS KIT.....	8
2.1 De RCX.	8
2.1.1 Algemeen.	8
2.1.2 De batterijen.	8
2.1.3 De besturingsknoppen.	9
2.2 De TOWER.	9
3. CONTROLE VAN DE TOESTAND VAN RCX -HARDWARE.....	10
3.1 Het diagnostics programma.	10
3.1.1 Opstellen van de componenten op de form.	10
3.1.2 Opstellen van de code.	11
4. HET STUREN VAN DE MOTOREN.	14
4.1 Motor&Sensor Test.	14
4.1.1 Opstellen van de componenten op de form.	15
4.1.2 Opstellen van de code.	16
5. HET GEBRUIK VAN DE SENSOREN.....	20
5.1 Configuratie van de druksensor.	20
5.1.1 Opstellen van de componenten op de form.	20
5.1.2 Opstellen van de code.	21
5.2 Configuratie van de lichtsensor.	22
5.2.1 Opstellen van de componenten op de form.	22
5.2.2 Opstellen van de code.	23
6. GEBRUIK VAN HET REGISTER VAN DE RCX.	24
6.1 Bespreking van het register.....	24
6.2 Werken met variabelen.	24

6.2.1	Opstellen van de componenten op de form.	25
6.2.2	Opstellen van de code.	26
7.	BOUWEN VAN AUTONOME ROBOTS.....	27
7.1	Bespreking.	27
7.2	Het downloaden van een programma met configuratie van de druksensor.	27
7.2.1	Opstellen van de componenten op de form.	28
7.2.2	Opstellen van de code.	29
7.3	Het downloaden van een programma met configuratie van de lichtsensor.	30
7.3.1	Opstellen van de componenten op de form.	31
7.3.2	Opstellen van de code.	31
7.4	Het Proximity programma.....	33
7.4.1	Opstellen van de componenten op de form.	33
7.4.2	Opstellen van de code.	33
8.	DE DATALOG.....	35
8.1	Inleiding.	35
8.2	Testen van de datalog.	35
8.2.1	Opstellen van de componenten op de form.	36
8.2.2	Opstellen van de code.	36
8.2.3	Werking van het programma.....	38
8.3	In grafiek brengen van de gegevens in de datalog.....	38
8.3.1	Opstellen van de componenten op de form.	39
8.3.2	Opstellen van de code.	39
8.3.3	Werking van het programma.....	42
9.	BIJLAGE.....	44
9.1	INDEX 1: Standaard programma's van de RCX.	44
•	Programma 1.....	44
•	Programma 2.....	44
•	Programma 3.....	44
•	Programma 4.....	45
•	Programma 5.....	45
9.2	INDEX 2: Properties en Events van de Spirit component.....	46
•	InitComm.....	46
•	CloseComm.....	46
•	UnLockFirmware(UnLockString).....	46
•	DownloadFirmware(FileName).....	47
•	SetWatch(Hours, Min).....	47
•	PBPowerDownTime(Time).....	48
•	PBBATTERY.....	48
•	PBAliveOrNot.....	49
•	PBTurnOff.....	49
•	PBTxPower(Number).....	50
•	TowerAndCableConnected.....	50
•	TowerAlive.....	51
•	On_(MotorList).....	51

•	Off(MotorList)	52
•	SetFwd(MotorList)	52
•	SetRwd(MotorList)	52
•	SetPower(MotorList, Source, Number)	53
•	SetSensorType(Number, Type)	53
•	SetSensorMode(Number, Mode, Slope)	54
•	Poll(Source, Number)	55
•	PlaySystemSound(Number)	56
•	PlayTone(Frequentie, Time)	57
•	SetVar(VarNo, Source, Number)	58
•	Bewerkingen met registerwaarden	58
•	SetEvent(Source, Number, Time)	59
•	BeginOfTask(Number)	60
•	EndOfTask	60
•	DownloadStatus	60
•	DownloadDone	60
•	EndOfTaskNoDownload	61
•	Loop(Source, Number)	61
•	EndLoop	62
•	If_(Source1, Number2, RelOp, Source2, Number2)	62
•	Else_	63
•	EndIf	63
•	Wait(Source, Number)	63
•	SetWatch(Hours,min)	64
•	SendPBMessage(Source,Number)	64
•	SelectPrgm(Number)	64
•	While_(Source1,Number1,RelOp,Source2,Number2)	65
•	EndWhile	65
•	SetDatalog(Size)	66
•	DatalogNext(Source,Number)	66
9.3	INDEX 3: Properties en hun waardenbereik	67
9.4	INDEX 4 : Nuttige internetadressen.	69

Voorwoord

Doel van dit project was om na studie van de ActiveX component, programma's die geschreven zijn in Visual Basic, te schrijven in Delphi. Zodoende we na het schrijven van de diagnostische programma's een eigen robot kunnen ontwerpen en programmeren.

Als ontwerp hadden we voor een heftruck gekozen, en hebben we deze ook gemaakt, maar door problemen met het downloaden in het geheugen van de RCX, hebben we het programmeren van onze robot niet kunnen voltooien. Het is daarom ook niet opgenomen in deze studie.

We hebben getracht om in dit verslag de mogelijkheden die men met de LEGO MINDSTORMS KIT heeft, zo uitgebreid mogelijk weer te geven.

Om dit alles tot stand te kunnen brengen beschikten we over de Legomindstorms kit met drie motoren, verscheidene sensoren en allerlei bouwstenen, de cursus Lego Mindstorms Programming With Visual Basic. Tevens hadden we een computer met Delphi 5, waarmee alle programma's geschreven zijn, ter onzer beschikking.

1. COM objecten en ActiveX controls.

1.1 Wat is COM.

COM staat voor *Component Object Model*, dit is een Microsoft specificatie voor het creëren en implementeren van herbruikbare componenten.

Het is eigenlijk een stuk binaire code die een specifieke functie volbrengt.

Net zoals men in Delphi VCL (*Visual Component Library*) hiërarchie kent, bezitten COM-objecten ook hiërarchie. COM heeft een aantal voordelen tegenover VCL:

- De creatie van COM objecten is onafhankelijk van de programmeertaal, ze kunnen dus in verschillende programmeertalen aangemaakt worden.
- Een COM object kan in vrijwel elke Windows programmeer omgeving gebruikt worden, dus in Delphi, C++Builder, Visual C++, Visual Basic, Visual dBase, Powerbuilder, ...

Er is ook een nadeel: COM objecten zijn een Windows/Intel-specificatie, en kunnen dus enkel in deze omgeving gebruikt worden.

Eén of meerdere COM objecten worden in een DLL (*Dynamic Library Link*) bestand ingekapseld. Deze bestanden bezitten een extensie *.dll* of *.ocx*.

Men verkrijgt toegang tot de functies van het object via de COM interface(s). Alle COM interfaces stammen af van een interface genaamd IUnknown en bezitten door overerving dus ook alle eigenschappen van deze. De interface toont wat het object te bieden heeft en laat ons toe het te gebruiken. Een object kan meerdere interfaces bezitten. Het aanspreken van deze COM interface kan niet direct, we moeten dit doen via de COM class of coclass. Zo een class kan meerdere interfaces bevatten.

COM objecten worden gemaakt met de reden deze objecten te kunnen gebruiken in verschillende programma's en door verschillende personen, en dus niet om op zichzelf te staan. Om het mogelijk te maken dat het COM object door anderen wordt gebruikt, is het noodzakelijk dat ze over informatie betreffende het object beschikken, deze informatie met alle gegevens en eigenschappen van het object bevindt zich in een speciaal bestand, de 'Type Library' met extensie. TLB.

Een COM object creëren:

- 1) Aanmaken van een DLL of ActiveX library om het COM object in op te slaan.
- 2) Het eigenlijke object creëren.
- 3) Met behulp van de Type Library interfaces toevoegen of wegdoen.
- 4) Properties en methoden toevoegen aan het object.
- 5) Compileren en hiermee dus ook het COM object en DLL bouwen.
- 6) Registreren.

1.2 Wat is ActiveX?

Een *ActiveX control*, vroeger *OCX*, is eigenlijk een subset van COM, dus alles wat we over COM objecten gezien hebben is evengoed toepasbaar op ActiveX controls. Het grootste verschil is dat een ActiveX control een *design-time interface* heeft. Ook heeft deze code die het mogelijk maakt om de control te gebruiken op web-pagina's en netwerken.

Om een ActiveX control te gebruiken, importeren we hem eerst naar het componentenpalet en plaatsen hem vandaar op de formdesigner. We kunnen er nu mee werken in design-time.

Een ActiveX control kan op 2 *manieren* aangemaakt worden:

- § vertrekkende van een bestaande VCL-component.
- § vertrekkende van nul door een Active Form te gebruiken.

1.3 Waarom ActiveX?

De ActiveX-technologie laat toe programma's te schrijven die dienst kunnen doen als plug-in voor andere toepassingen. We verduidelijken met een voorbeeld: het is mogelijk een picture viewer aan te maken, die als plug-in kan gebruikt worden voor tekstverwerkers, databases of web browsers. Het voordeel van deze techniek is dat men niet telkens wanneer men een database aanmaakt, de picture viewer moet herschrijven. Het volstaat enkel te verwijzen naar de beschikbare plug-in toepassing.

De communicatie tussen de PC en de RCX steunt op dit principe. Indien men een programma wilt schrijven dat de RCX controleert, hoeft men enkel het RCX-control object aan te roepen. In ons specifieke geval wordt dat object *spirit.ocx* genoemd. Met andere woorden, de *spirit.ocx* zorgt voor een interface tussen het programma en de RCX.

Deze ActiveX-component bezit een aantal eigenschappen en procedures, die toelaten de RCX op een eenvoudige manier te benaderen. Het geheel van procedures, functies en methodes noemt men de *Spirit Code*.

2. Introductie van de LEGO MINDSTORMS KIT.

2.1 De RCX.

2.1.1 Algemeen.



RCX is de afkorting van Robotics Command System en is een programmeerbare LEGO steen.

Dit wil zeggen dat er in de 'brick' een microprocessor aanwezig is, welke dient voor het verwerken van gegevens (ingangswaarden worden door middel van een programma omgezet om alzo een bepaalde uitgangswaarden te geven). Ook is er een inwendig geheugen voor het opslaan van firmware en programma's en een luidspreker.

Uitwendig heeft de RCX nog drie sensor-, dus ingangspoorten, drie uitgangspoorten waaraan men motoren kan koppelen, vier besturingsknoppen, een LCD-venster waarop men de status van de RCX kan aflezen en een infraroodzender, waardoor communicatie mogelijk is tussen de RCX en de computer (evenwel via de tower).

Men moet opletten wanneer men de motoren aan de RCX koppelt daar er vier verschillende mogelijkheden zijn om de kabel vast te zetten op de brick. Wanneer men wil dat motoren in dezelfde zin bewegen na een commando, moet men de kabels op dezelfde wijze op de RCX koppelen. Bij de sensoren heeft de manier van vastzetten geen invloed.

In de RCX zijn vijf programma's opgeslagen die men als test kan gebruiken. Zie index 1 voor verdere gegevens hierover.

2.1.2 De batterijen.

De voeding van de RCX gebeurt met behulp van zes AA-batterijen. Hierbij moet men opletten dat vervanging van de batterijen gebeurt binnen de minuut anders wordt het RCX-geheugen gewist (zowel firmware als gedownloadde programma's).

Om de batterijen te sparen schakelt de RCX zich automatisch uit als hij 15 minuten lang niet wordt gebruikt. Evenwel kan deze tijd programmatisch worden aangepast op een tijd gelegen tussen 1 en 255 minuten of op oneindig zodat hij zich nooit automatisch afzet.

Dit wordt bereikt met de *PBPowerDownTime(Time)* property van de SPIRIT1 component.

2.1.3 De besturingsknoppen.

De knoppen worden gebruikt voor het besturen van de RCX en de bijhorende programma's.

- Met *ON*OFF* schakelt men de RCX aan of uit. Enkel indien de RCX aan staat kan men de andere knoppen bedienen.
- Met *Prgm* kan men een programma kiezen dat bewaard is in één van de vijf programmaslots van de RCX. Het programmanummer wordt rechts in de LCD weergegeven.
- Met *Run* kan men het geselecteerde programma starten en stoppen.
- Met *View* kan men informatie opvragen over sensoren- en motorengegevens.
Zo kan men zijn of een druksensor al-dan-niet is ingedrukt.
Om deze functie te kunnen gebruiken moet het programma minstens één keer zijn uitgevoerd!

2.2 De TOWER.

De TOWER wordt met een kabel aangesloten op een vrije seriële-poort van de computer en zorgt voor een draadloze verbinding tussen de PC en de RCX. Met de TOWER is het aldus mogelijk om programma's vanaf de computer naar de RCX door te sturen.

Men heeft twee mogelijkheden. Men kan de RCX continu laten communiceren met de PC, dit verloopt trager, daar telkens er een ingangsverandering optreedt, de RCX, via de TOWER aan het programma op de PC moet vragen welke actie hij nu moet uitvoeren, waarna het antwoord moet worden terug gezonden. Hierbij moet de PC blijven aanstaan.

Daar deze vertraging zeer ongewenst is, kan men ook het programma volledig downloaden in de RCX, waarbij men dus slechts één keer gebruik maakt van de TOWER en de RCX verder autonoom werkt. De PC mag dus worden afgezet, daar het programma opgeslagen is in één van de vijf programmaslots van de RCX.

De RCX en de TOWER kunnen enkel met elkaar communiceren indien ze elkaar 'zien'. Niettegenstaande 10-12 cm de beste afstand is voor het laden, is communicatie mogelijk tot een maximale afstand van 30 meter.

Er zijn twee zendbereiken mogelijk: kort en lang. Dit is van belang wanneer in dezelfde ruimte met meerdere RCX-en wordt gewerkt. In dat geval stelt men het zendbereik in op kort om alzo interferentie tussen de verschillende TOWERS te vermijden.

Deze instelling kan door het knopje op de TOWER op kort of lang in te stellen of programmatisch door aanroepen van de *PBTxPOWER(Number)* property

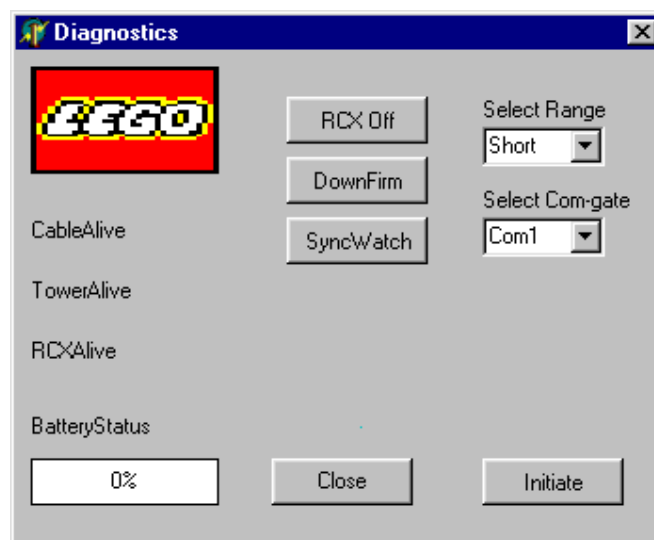
De TOWER moet voorzien worden van een 9-volt batterij

3. Controle van de toestand van RCX -hardware.

3.1 Het diagnostics programma.

Met behulp van dit programma wordt de initiële staat van de RCX weergegeven en kunnen we de COM-Poort en de range instellen, alsook de connecties en de batterijstatus controleren. We kunnen tevens de firmware downloaden en de clock van de brick synchroniseren met die van de PC

3.1.1 Opstellen van de componenten op de form.



Component	Property	waarde
SPIRIT	NAME	Spirit1
FORM	CAPTION NAME	Diagnostics Form1
LABEL	CAPTION NAME	CableAlive Label1
LABEL	CAPTION NAME	TowerAlive Label2
LABEL	CAPTION NAME	RCXAlive Label3
LABEL	CAPTION NAME	Select Range Label4
LABEL	CAPTION NAME	Select Com Label5
LABEL	CAPTION NAME	Wrong ComSetting Label6
LABEL	CAPTION NAME	(laat open) Label7

LABEL	CAPTION NAME	(laat open) LblCableAlive
LABEL	CAPTION NAME	(laat open) LblRcxAlive
LABEL	CAPTION NAME	(laat open) LblTowerAlive
LABEL	CAPTION NAME	BatteryStatus LblBatteryStatus
GAUGE	NAME MINVALUE MAXVALUE	GgBateryStatus 0 10000
COMBOBOX	NAME ITEMS TEXT	CmbSetRange Short / Long Short
COMBOBOX	NAME ITEMS TEXT	CmbSelectCom Com1 ... Com4 Com1
BUTTON	NAME CAPTION	RcxOff RCX Off
BUTTON	NAME CAPTION	DownFirm DownFirm
BUTTON	NAME CAPTION	BtnSyncWatch SyncWatch
BUTTON	NAME CAPTION	BtbnInitiate Initiate
BUTTON	NAME CAPTION	Button1 Close

3.1.2 Opstellen van de code.

```

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if spirit1.InitComm then begin           //indien InitComm aangeroepen was, sluit de verbinding
    spirit1.CloseComm ;                   //met de com-poort
  end;

```

```

procedure TForm1.BtbnInitiateClick(Sender: TObject); //Selecteer de COM-poort waarop
begin                                               //de tower is aangesloten
  if Form1.CmbSelectCom.Text = 'Com1' then spirit1.ComPortNo := 1;
  if Form1.CmbSelectCom.Text = 'Com2' then spirit1.ComPortNo := 2;
  if Form1.CmbSelectCom.Text = 'Com3' then spirit1.ComPortNo := 3;
  if Form1.CmbSelectCom.Text = 'Com4' then spirit1.ComPortNo := 4;

```

```

If Spirit1.InitComm then                          //initialiseer Com verbinding
begin
  Label6.visible := false;

```

```

form1.LblCableAlive.visible := true;
form1.LblTowerAlive.visible := true;
form1.LblRcxAlive.visible := true;
If Form1.CmbSetRange.Text = 'Short' then spirit1.PBTxPower(0) else
spirit1.PBTxPower(1);           //Selecteer de zendrange van de tower

If Spirit1.PBAliveOrNot then      //Controleert of de brick al dan niet kan antwoorden
    Begin
        form1.LblRcxAlive.caption := 'OK';           //alles goed
        form1.GgBaterystatus.Progress := spirit1.PBBattery;

    end
    else
    begin
        form1.LblRcxAlive.caption := 'Failure';       //iets niet in orde
    end;

If spirit1.TowerAndCableConnected then      //kijkt de hardware connecties na
    begin
        form1.LblCableAlive.caption := 'Ok';         //alles goed
    end
    else
    begin
        form1.LblCableAlive.caption := 'Failure';    //iets niet in orde
    end;
end
else
begin
label6.visible := true;
form1.LblCableAlive.visible := false;
form1.LblTowerAlive.visible := false;
form1.LblRcxAlive.visible := false;
end;
If spirit1.TowerAlive then              //Controleert de staat tower
    begin
        form1.LblTowerAlive.caption:= 'OK';          //alles goed
    end
    else
    begin
        form1.LblTowerAlive.caption:= 'Failure';     //iets niet in orde
    end;

end;

procedure TForm1.RcxOffClick(Sender: TObject);      //schakelt de RCX uit
begin
if not(spirit1.PBTurnOff) then          //als hij nog niet uitstaat, zet hem dan uit
spirit1.PBTurnOff
end;

```

```

procedure TForm1.DownFirmClick(Sender: TObject);    //downloaden van de firmware
begin
  spirit1.DownloadFirmware('firm0309.lgo');          //bestandsnaam
  spirit1.UnlockFirmware('Do you byte, when I knock?'); //Unlock key

end;

procedure TForm1.BtnSyncWatchClick(Sender: TObject); //Instellen van de clock
var hour,min,sec,msec:word;                          // van de brick = clock PC
begin
  decodetime(now,hour,min,sec,msec);                 //zet de TdateTime waarde om in uren, min,...
  spirit1.SetWatch(hour,min);                          //synchroniseren tijd brick/PC
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  close;                                              //Sluit programma af
end;

end.

```

Zie index 2 voor meer informatie over de gebruikte properties van de SPIRIT1 component

4. Het sturen van de motoren.

4.1 Motor&Sensor Test.

In het vorige hoofdstuk hebben we RCX als een soort netwerkcomputer. Uiteraard kunnen we veel meer dingen doen met deze LEGO steen.

Allereerst gaan we een eenvoudige robot in elkaar steken en koppelen we twee motoren aan op poorten A en C.

Wij maakten gebruik van de standaard gelijkstroom motoren met eigenschappen: $U = 9\text{ V}$, $I_{\text{nom}} = 100\text{ mA}$

In een volgend stadium koppelen we ook nog eens twee druksensoren aan op poort 1 en 3 en plaatsen de ene aan de voorkant van ons wagentje en de andere aan de achterzijde.

De bedoeling is om met behulp van het programma de robot te laten rijden in de richting die wij verkiezen. Hier werken we zonder sensoren.

In een tweede deel van het programma gaan we bovendien gebruik maken van de twee druksensoren.

Omdat de brick steeds met het programma op de PC moet communiceren maken we gebruik van een timer die op regelmatige intervallen de toestand van de sensoren controleert. Dit heeft uiteraard als nadeel dat er geen ogenblikkelijke reactie van de RCX kan optreden. Met als gevolg dat het wagentje reeds kan verongelukt zijn voordat er een signaal vanuit de PC wordt terug gestuurd om te melden dat de robot bijvoorbeeld moet stoppen. Dit euvel zullen we later kunnen oplossen door het volledige programma in het geheugen van de RCX te downloaden.

In dit programma zal, wanneer de robot aan het rijden is en een druksensor ingedrukt wordt, het wagentje in de andere richting gaan rijden.

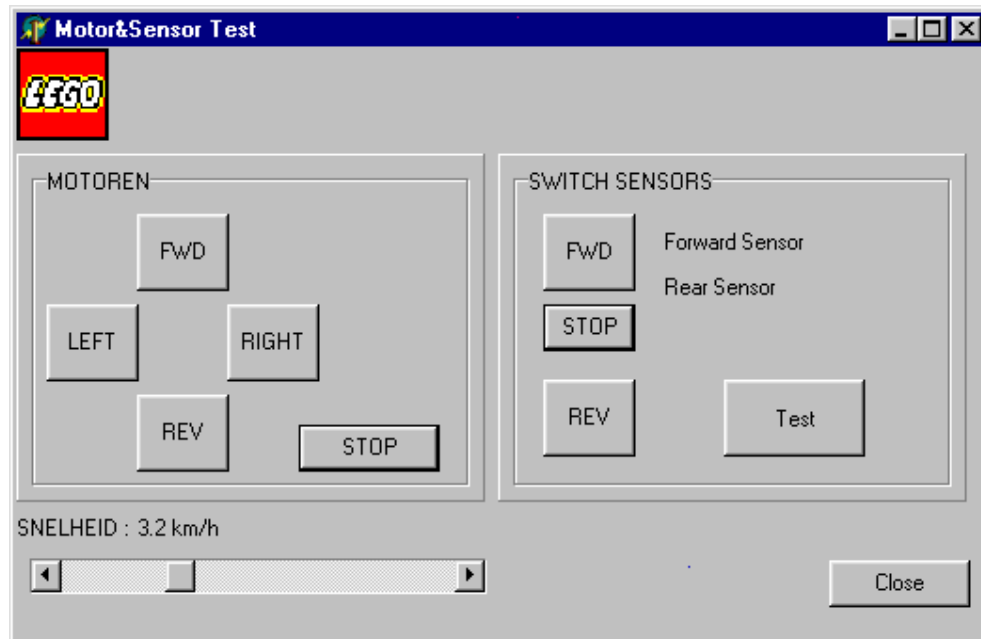
Hiervoor dient men eerst op de knop TEST te drukken om het timerinterval in te schakelen om vervolgens de robot een richting te geven door op FWD of REV te drukken.

Tevens gaan we kunnen zien of er een sensor is ingedrukt daar er naast de captions Forward Sensor en Rear Sensor een 1 of een 0 zal verschijnen wat duidt op het ingedrukt of niet ingedrukt zijn van de sensoren.

Dit doen we door: `spirit1.SetSensorType(0,1) //0: poort 1, 1: sensor is van type switch`
`label1.caption := IntToStr(spirit1.poll(9,0)); //9: geef sensorwaarde,`
`//0: van sensor op poort 1`

Tenslotte gaan ook nog eens de snelheid van het wagentje kunnen instellen door middel van de scrollbar onderaan de form. Hiervoor maken we gebruik van de property *SetPower* van de Spirit component.

4.1.1 Opstellen van de componenten op de form.



Component	Property	waarde
FORM	NAME CAPTION	Form1 Motor&Sensor Test
SPIRIT	NAME	Spirit1
LABEL	NAME CAPTION	Label1 (Leeg laten)
LABEL	NAME CAPTION	Label2 SNELHEID :
LABEL	NAME CAPTION	Label3 3.2 km/h
LABEL	NAME CAPTION	Label4 Forward Sensor
LABEL	NAME CAPTION	Label5 Rear Sensor
LABEL	NAME CAPTION	Label6 (Leeg laten)
TIMER	NAME	Timer1
PANEL	NAME CAPTION	Panel1 Motoren
PANEL	NAME CAPTION	Panel2 Swith Sensors
SCOLLBAR	NAME	Scrollbar1
BUTTON	NAME CAPTION	Button1 FWD
BUTTON	NAME CAPTION	Button2 LEFT

BUTTON	NAME CAPTION	Button3 RIGHT
BUTTON	NAME CAPTION	Button4 REV
BUTTON	NAME CAPTION	Button5 STOP
BUTTON	NAME CAPTION	Button6 FWD
BUTTON	NAME CAPTION	Button7 REV
BUTTON	NAME CAPTION	Button8 STOP
BUTTON	NAME CAPTION	Button9 CLOSE
BITBUTTON	NAME CAPTION	BitBtn1 Test

4.1.2 Opstellen van de code.

```

procedure TForm1.Button1MouseDown(Sender: TObject; Button: TMouseButton; //vooruit
  Shift: TShiftState; X, Y: Integer);
begin
  if scrollbar1.position = 0 then      //indien scrollbar uiterst links:
  begin
    spirit1.SetFwd('02');              //Sluit de motoren aan op A en C en laat ze vooruit draaien
    spirit1.Off('02');                //motoren af
  end
  else                                //anders doe
  begin
    spirit1.On_('02');                 //Sluit de motoren aan op A en C laat ze vooruit draaien
    spirit1.SetFwd('02');              // laat ze vooruit draaien
    spirit1.SetSensorType(0,1);        //0:sensorpoort 1,    1:druksensor
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  spirit1.initcomm;                   //initiëer com-poort 1
  spirit1.setpower('012',0,15);       //012 power van motor op poort A,B en C
  end;                               //0 ,15 verwijst naar de power level, bewaard in variabele 15

procedure TForm1.Button4MouseDown(Sender: TObject; Button: TMouseButton; //achteruit
  Shift: TShiftState; X, Y: Integer);
begin
  if scrollbar1.position = 0 then      // indien op nul, mogen de motoren niet draaien
  begin
    spirit1.SetRwd('02');              //draai achteruit

```



```

spirit1.Off('02');           //zet de motoren uit

end
else
begin
spirit1.on_('02');           //zet de motoren op poort A en C aan
spirit1.SetRwd('02');        //draai achteruit
end;
end;

procedure TForm1.Button4MouseUp(Sender: TObject; Button: TMouseButton; //Stop
  Shift: TShiftState; X, Y: Integer);
begin
spirit1.Off('02');
end;

procedure TForm1.Button1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
spirit1.off('02'); //wanneer de knop wordt losgelaten, stop de robot
end;

procedure TForm1.Button5Click(Sender: TObject); //Stop en zet timerinterval uit
begin
spirit1.off('02');
timer1.enabled :=false;
end;

procedure TForm1.Button2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
Begin           //wanneer er op de knop button2 gedrukt wordt doe è
if scrollbar1.position = 0 then
begin
spirit1.Off('02');           //mag niet rijden indien scrollbar position =0
spirit1.SetFwd('2');
end
else
begin
spirit1.off('02');
spirit1.on_('2');           //zet de rechtse motor uit zodat de robot naar links draait
spirit1.SetFwd('2');
end;
end;
procedure TForm1.Button3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
Begin           //wanneer er op de knop button3 gedrukt wordt doe è
if scrollbar1.position = 0 then
begin
spirit1.SetFwd('0');           //mag niet rijden indien scrollbar position =0

```

```

spirit1.Off('02');
end
else
begin
spirit1.off('02');
spirit1.on_('0');           //zet de linkse motor uit zodat de robot naar rechts draait
spirit1.SetFwd('0');
end;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
spirit1.On_('02');           //zet de motoren in aan modus maar laat ze niet draaien
timer1.enabled:=true;       //timerinterval aan
end;

procedure TForm1.ScrollBar1Change(Sender: TObject); //instellen km/h van de motoren
begin
if scrollbar1.Position = 0 then
begin
spirit1.Off('02') ;
label3.caption:= '0 km/h';
end
else
begin
spirit1.On_('02') ;
spirit1.SetPower('02',2,scrollbar1.position);
label3.caption:=floattostr(scrollbar1.position*1.6) + 'km/h';
end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);      //Timerinterval
begin
// voorste sensor
spirit1.SetSensorType(0,1); //Stel sensor in :0: poort 1, 1: sensor is van het type switch
label1.caption := IntToStr(spirit1.poll(9,0));      //9:geef de toestand weer van,
// achter sensor                                     //0 : sensor op poort 1
spirit1.SetSensorType(2,1);
label6.caption := IntToStr(spirit1.poll(9,2));      //2: sensor op poort 3
If spirit1.poll(9,0) = 1 then                         //wanneer voorste sensor ingedrukt is doe è
begin
    If spirit1.poll(9,2) = 1 then                     //wanneer beide sensoren ingedrukt zijn doe è
    begin
        spirit1.Off('02');           //zet motoren af
    end
else
    // wanneer voorste sensor ingedrukt is doe è
    begin
        spirit1.PlayTone(1000,5); //laat een geluidsignaal horen
        spirit1.PlayTone(500,5);
        spirit1.SetRwd('02');      //laat de robot achteruit gaan
    end
end
end

```

```

    end;
end;

If spirit1.poll(9,2) = 1 then      //wanneer de achterste sensor is ingedrukt doe è
begin
    If spirit1.poll(9,0) = 1 then  //wanneer beide sensoren ingedrukt zijn doe è
    begin
        spirit1.Off('02');        //zet de motoren uit
    end
    else                          // wanneer achterste sensor ingedrukt is doe è

    begin
        spirit1.SetFwd ('02');     //laat de robot vooruit rijden
        spirit1.PlayTone(1000,5);  //laat een geluidsignaal horen
        spirit1.PlayTone(500,5);
    end;
end;
end;

procedure TForm1.Button9Click(Sender: TObject);    //programma afsluiten
begin
    spirit1.off('02');          //zet motoren uit
    timer1.enabled :=false;    //zet timer uit
    spirit1.PBTurnOff;          //zet de brick uit
    spirit1.CloseComm;          //sluit de comm-poort af
    close;                     //sluit form1 en dus programma af
end;

end.

```

5. Het gebruik van de sensoren.

5.1 Configuratie van de druksensor.

In dit programma gebruiken we de property *Poll* om de toestand van de druksensor, welke we op poort A plaatsen, te controleren.

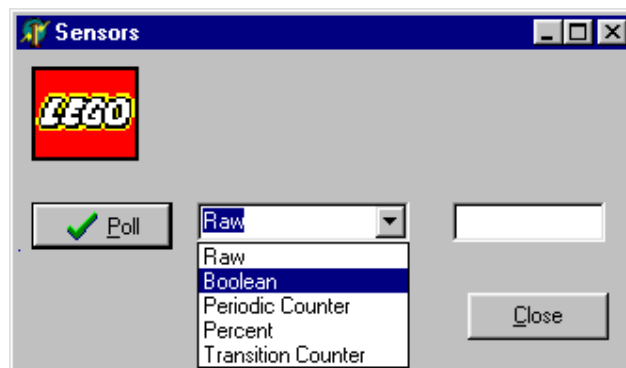
De verschillende meetmogelijkheden zijn :

- Raw
- Boolean
- Periodic Counter
- Percent
- Transition Counter

Voor meer uitleg over deze meetmogelijkheden: zie Bijlage

Het enige wat we doen is op de knop met caption POLL drukken en de druksensor bedienen om also een uitleeswaarde in de editbox te verkrijgen.

5.1.1 Opstellen van de componenten op de form.



Component	Property	waarde
FORM	NAME CAPTION	Form1 Sensors
SPIRIT	NAME	Spirit1
BITBTN	NAME CAPTION	Poll &Poll
BUTTON	NAME CAPTION	Button1 &Close
COMBOBOX	NAME TEXT ITEMS	ComboBox1 Raw Raw Boolean Periodic Counter Percent Transition Counter
EDIT	NAME TEXT	Edit1 (Leeg laten)

5.1.2 Opstellen van de code.

```
procedure TForm1.PollClick(Sender: TObject);    //keuze van weergave sensorresultaat
begin

If combobox1.Text = 'Raw' Then begin
spirit1.SetSensorType(0,1);                    //setsensortype poort 0 (1), soort 1 (Switch)
Edit1.text := inttostr(Spirit1.poll(12,0));    // 12: rawval source:sensorvalue 0: Sensor1
end;

If combobox1.Text = 'Boolean' Then begin
spirit1.SetSensorType(0,1);                    //setsensortype poort 0 (1), soort 1 (Switch)
Edit1.text := inttostr(Spirit1.poll(13,0));    // 13: boolean source:sensorvalue 0: Sensor1
end;

If combobox1.Text = 'Transition Counter' Then begin
spirit1.SetSensorType(0,1);                    //setsensortype poort 0 (1), soort 1 (Switch)
spirit1.SetSensorMode(0,2,0) ;                 // 2:trans count mode
Edit1.text := inttostr(Spirit1.poll(9,0));
end;

If combobox1.Text = 'Periodic Counter' Then begin
spirit1.SetSensorType(0,1);                    //setsensortype poort 0 (1), soort 1 (Switch)
spirit1.SetSensorMode(0,3,0) ;                 // 3:periodic count mode
Edit1.text := inttostr(Spirit1.poll(9,0));
end;

If combobox1.Text = 'Percent' Then begin
spirit1.SetSensorType(0,1);                    //setsensortype poort 0 (1), soort 1 (Switch)
spirit1.SetSensorMode(0,5,0) ;                 // 5:percent mode
Edit1.text := inttostr(Spirit1.poll(9,0));
end;
end;

procedure TForm1.Button1Click(Sender: TObject);    //afsluiten
begin
spirit1.PBTurnOff;
spirit1.CloseComm;
close;
end;

procedure TForm1.FormCreate(Sender: TObject);    //programma starten
begin
spirit1.InitComm
end;
end.
```

5.2 Configuratie van de lichtsensensor.

In het volgende programma maken we gebruik van zowel een licht-(op poort 3) als een druksensor (op poort 1). De bedoeling is om, wanneer de druksensor is ingedrukt, de waarde van de lichtsensensor weer te geven. Hierdoor kunnen we de lichtsensensor over verschillende kleuren laten bewegen en alzo hun RAW-waarde te registreren.

Zo bemerkten volgende waarden voor verschillende kleuren

- Hoe witter het voorwerp, hoe hoger de waarde (van 50 tot in de limiet naar 100)
- Hoe zwarter het voorwerp, hoe lager de waarde (van 29 tot in de limiet naar 0)
- Alle kleurencombinaties hebben aldus een cijferwaarde tussen 29 en 50

Een nadeel van deze manier is dat de afstand lichtsensensor – blad steeds dezelfde moet zijn. Alsook de lichtinval daar deze ook een duidelijke invloed heeft op het verkregen resultaat. Maar de verkregen waarden geven, desalniettemin een goede indicatie van de kleurwaarde die de sensor detecteert.

Om de resultaten weer te geven maken we gebruik van de timer-component om op regelmatige intervallen een waarde te verkrijgen:

```
procedure TForm1.Timer1Timer(Sender: TObject);  
begin  
  If spirit1.Poll(9,0)=1 then begin  
    edit1.text := inttostr(Spirit1.poll(9,2));  
  end;
```

5.2.1 Opstellen van de componenten op de form.



Component	Property	waarde
FORM	NAME CAPTION	Form1 Sensors
SPIRIT	NAME	Spirit1
MEMO	NAME LINES	Memo1 Hou de druksensor ingedrukt en beweeg de lichtsensoren over verschillende kleuren
EDIT	NAME TEXT	Edit1 (laat leeg)
SHAPE	BRUSH/COLOR	ClBlack
TIMER	NAME INTERVAL	Timer1 1
BUTTON	NAME CAPTION	Button1 Close
LABEL	NAME CAPTION	Label1 Light Sensor

5.2.2 Opstellen van de code.

```

procedure TForm1.Button1Click(Sender: TObject);           //Afsluiten
begin
  spirit1.CloseComm;
  close;
end;

procedure TForm1.FormCreate(Sender: TObject);             //Bij programma starten
begin
  Spirit1.InitComm;
  Spirit1.SetSensorType(0,1);                             //poort 1, switch
  Spirit1.SetSensorType(2,3);                             // poort 3 light  %
  Spirit1.SetSensorMode(3,0,0);                          // % naar raw
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  If spirit1.Poll(9,0)=1 then begin
    edit1.text := inttostr(Spirit1.poll(9,2));           //geef uitgangswaarde weer
    Shape1.Brush.Color := clgreen;                     //maak shape groen
  End
  else
    Shape1.Brush.Color := clblack;
  end;
end;

```

6. Gebruik van het register van de RCX.

6.1 Bespreking van het register.

In de RCX zijn er 32 globale, interne variabelen of registers, deze kunnen waarden kunnen stockeren tussen de -32768 en $+32767$.

Er zijn verschillende methoden om deze registerwaarden te manipuleren om er alsoo verschillende wiskundige handelingen op uit te voeren.

Zo kunnen we in het programma de waarde opvragen die in een register gestockeerd is, en kunnen we bovendien deze waarde vervangen door een andere. Hierbij moeten we opletten dat de nieuwe waarde tussen de twee grenswaarden ligt, anders zal er een fout worden weergegeven. Tevens wordt er een fout gegeven indien er indien het nummer van het register groter is dan 31.

6.2 Werken met variabelen.

In dit programma gaan we enkel de waarde van het register wijzigen om zijn eigenschappen aan te tonen. Hierbij hebben we twee weergavemogelijkheden van de interne variabelen.

* Een eerste mogelijkheid is om één per één het registernummer (tussen 0 en 31) in te geven in de Editbox *txtPollVar* waarna de waarde gestockeerd in deze variabele weergegeven wordt in de Editbox *txtPollVall*.

* Een tweede mogelijkheid is de *Poll All* functie te gebruiken. Hierbij zal het programma alle registers afdraan en de waarden gestockeerd in de registers alsook het registernummer weergegeven in de Memobox.

Om de waarde in een register te veranderen kijken we eerst na welke waarde in dit register gestockeerd is (zie hierboven). Vervolgens zetten we het cijfer dat het aan te passen register aanduidt in de editbox *txtSetVar* en de gewenste waarde in de editbox *txtSetVall*, waarna we op de knop met caption *Set Variable* klikken.

Wanneer we nu opnieuw controleren welke waarde het register bevat, zien we dat er de nieuwe waarde in gestockeerd staat, waarmee we de waarde in het register van de RCX hebben aangepast.

Tenslotte kunnen we met de property *SetEvent* en de event *VariableChange* het programma laten weergegeven dat er een registerwaarde is gewijzigd alsook de waarde waarin hij in gewijzigd is.

Zie Index 2 voor meer informatie hierover.

6.2.1 Opstellen van de componenten op de form.

Component	Property	waarde
SPIRIT	NAME	Spirit1
FORM	NAME CAPTION	Form1 Variable Manipulation
EDIT	NAME TEKST	txtSetVar (Leeg Laten)
EDIT	NAME TEKST	txtSetVall (Leeg Laten)
EDIT	NAME TEKST	txtPollVar (Leeg Laten)
EDIT	NAME TEKST	txtPollVall (Leeg Laten)
MEMO	NAME ITEMS SCROLLBARS	Memo1 (Leeg Laten) ssVertical
BUTTON	NAME CAPTION	cmdSet Set Variable
BUTTON	NAME CAPTION	CmdPoll Poll Variable
BUTTON	NAME CAPTION	Button2 Poll All
BUTTON	NAME CAPTION	cmdAutoPoll A&uto Poll
BUTTON	NAME CAPTION	Button1 exit
LABEL	NAME CAPTION	lblSet To
LABEL	NAME CAPTION	lblPoll Gives

6.2.2 Opstellen van de code.

```
procedure TForm1.Button1Click(Sender: TObject);           //programma afsluiten
begin
  Spirit1.CloseComm;                                     //com-poort afsluiten
close;
end;

procedure TForm1.CmdPollClick(Sender: TObject);
begin
  if (strtoint(form1.txtpollvar.text)>0) and (strtoint(form1.txtpollvar.text)<31) Then
  begin          //registernummer moet gelegen zijn tussen 0 en 31
    form1.txtPollVall.Text := inttostr(Spirit1.Poll(0,strtoint(txtpollvar.text))); //0:Poll register
    if form1.txtSetVall.Text = form1.txtpollvall.Text then //strtoint(txtpollvar.text) adres Reg
    begin
      messagedlg('Alles OK',mtinformation,[mbok],0); //wanneer beide waarden gelijk
    end
  else
    messagedlg('somethings wrong!!',mterror,[mbok],0); // wanneer beide waarden verschillend
  end ;
  if (strtoint(form1.txtpollvar.text)<0) Or (strtoint(form1.txtpollvar.text)>31) Then begi
  messagedlg('somethings wrong!!',mterror,[mbok],0);    // registernummer moet
  end;                                                    //gelegen zijn tussen 0 en 31
end;

procedure TForm1.cmdSetClick(Sender: TObject);
begin
  Spirit1.SetVar(strtoint(txtsetvar.text),2,strtoint(txtsetvall.text));
end; // strtoint(txtsetvar.text): Registernr., 2:source, strtoint(txtsetvall.text) :nieuwe waarde

procedure TForm1.FormCreate(Sender: TObject);           //programma starten
begin
  Spirit1.InitComm;
end;

procedure TForm1.Button2Click(Sender: TObject);         //POLL ALL
var i:integer;    //variabele
begin
  for i:=0 to 31 do //doorloop het ganse register en geef hun waarden weer
    memo1.lines.add(inttostr(i) + ' => ' +inttostr(spirit1.poll(0,i)));
  end;
  procedure TForm1.cmdAutoPollClick(Sender: TObject); //automatisch register 6 weergeven
  begin          //bij verandering
    Spirit1.Setevent(0,6,200); //0: source =cte, 6:registernr, 200: tijd in ms
  end;

  procedure TForm1.Spirit1VariableChange(Sender: TObject; Number, Value: Smallint);
  begin          //Geef bericht wanneer een register gewijzigd is
    messagedlg('Variable ' + txtpollvar.Text + ' has changed',mtinformation,[mbok],0);
  end;
```

7. Bouwen van autonome robots.

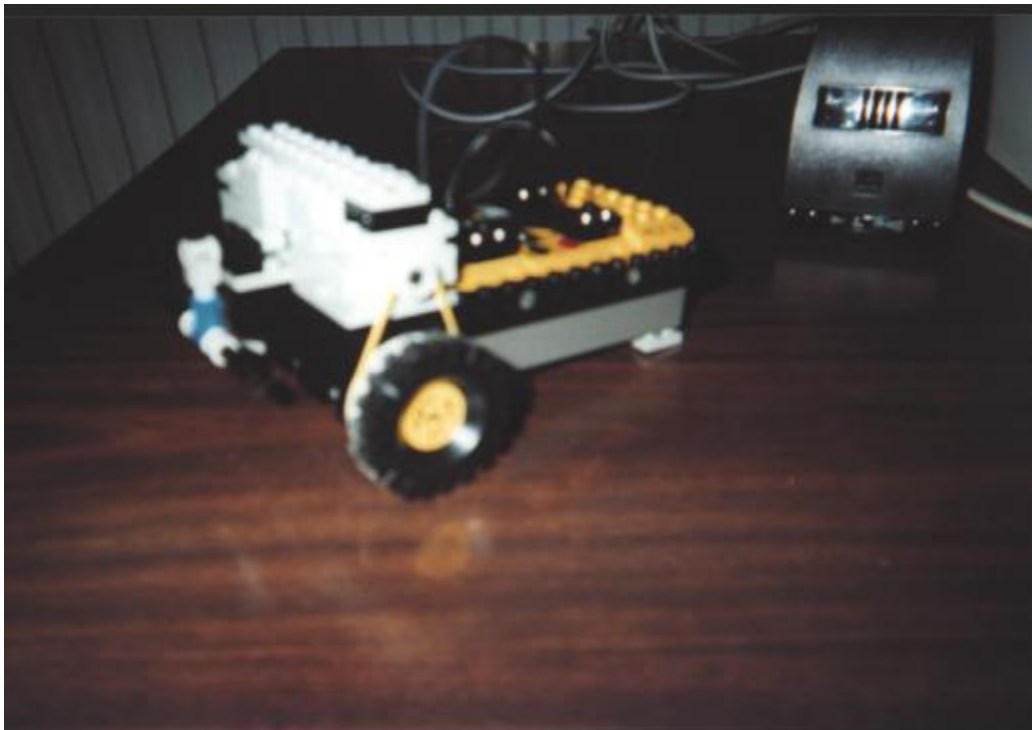
7.1 Bespreking.

Alle voorgaande programma's waren gebaseerd op '*real time*' controle van de RCX door de computer. Nadeel hiervan was dat er steeds een vertragingstijd optrad wanneer er een signaalverandering van de sensoren was, daar de RCX steeds aan het programma op de PC moet vragen welke actie hij, als gevolg van de ingangsverandering moet uitvoeren.

Dit wil dus ook zeggen dat de computer en de TOWER steeds moest aanstaan opdat communicatie mogelijk zou zijn.

Om deze problemen te vermijden, kunnen we gebruik maken van de '*autonome*' controle van de RCX. Dit houdt in dat het volledige programma opgeslagen wordt in het geheugen van de RCX zonder dat er nog tussenkomst van de computer vereist is omdat er nu direct en intern besluiten kunnen worden gemaakt bij eventuele ingangsveranderingen.

7.2 Het downloaden van een programma met configuratie van de druksensor.



7.2.1 Opstellen van de componenten op de form.

Om dit programma te kunnen gebruiken dient men een motor te koppelen op poort A en één op poort C, alsook een druksensor op poort 1

Wanneer men op de knop Download klikt, zal het bijhorende programma gedeelte worden gedownload in de RCX in programmaslot 3.

Wanneer het groene lichtje van de tower uit is, is het downloaden voltooid en kunnen we door op de knop Run (op de RCX) te drukken het programma doen werken.

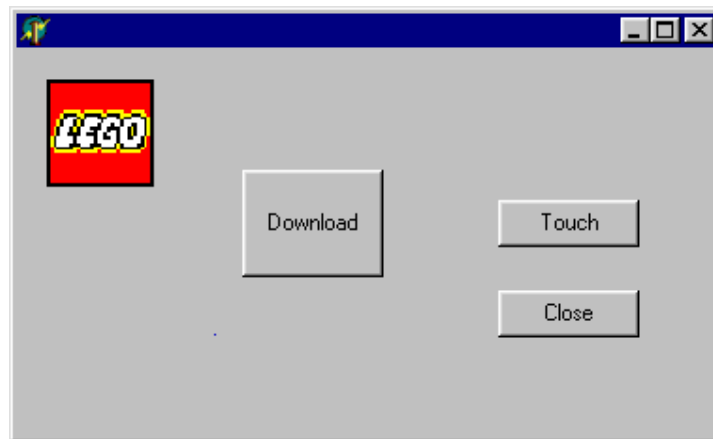
Hierdoor zullen de twee motoren beginnen te draaien en na twee seconden stoppen. Dit is uiteraard een zeer eenvoudig programma, maar ze geeft zeer goed de mogelijkheden en voordelen aan van een programma te downloaden naar de RCX.

Uiteraard dient de code hiervoor aangepast worden, zo moeten we duidelijk maken dat we het programma wensen over te brengen naar de brick.

Een ander programma verkrijgen we door op de knop met caption 'Touch' te klikken. Hier zal opnieuw het programma worden overgezet maar nu in slot 2 van de RCX.

Indien we nu op run drukken (Het LCD-scherm gaat automatisch op programma 2 staan), zal de RCX vooruit bewegen totdat de voorste sensor is ingedrukt.

Hierdoor zal hij achteruit bewegen gedurende 1 seconde waarna hij zich gedurende 2 seconden zal draaien, om vervolgens terug vooruit te rijden.



Component	Property	Waarde
SPIRIT	NAME	Spirit1
FORM	NAME	Form1
	CAPTION	Download Programma
BUTTON	NAME	Button1
	CAPTION	Download
BUTTON	NAME	Button2
	CAPTION	Close
BUTTON	NAME	Button3
	CAPTION	Touch

7.2.2 Opstellen van de code.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Spirit1.CloseComm;           //Afsluiten programma
  close;
end;

procedure TForm1.Button1Click(Sender: TObject);      //klikken op download knop
begin
  Spirit1.SelectPrgm(2);       //Kiezen voor programmaslot 3 (van 5)
  spirit1.BeginOfTask(1);       //initialiseert het downloaden van de code
  spirit1.SetPower('02',0,15);  //stelt het vermogen aan de as van de motor in
  spirit1.Setfwd ('02');        //stelt de draaizin van de motoren op poort A en C in
  spirit1.On_('02');            //laat de motoren draaien
  spirit1.Wait(2,200);          //2 :source: cte 200: 2 seconden
  spirit1.Off('02');            //na 2 sec zet de motoren op poort A en C uit
  spirit1.EndOfTask;            //stop met download
end;

procedure TForm1.FormCreate(Sender: TObject);        //opstarten programma(op pc)
begin
  spirit1.InitComm;
end;

procedure TForm1.Spirit1DownloadDone(Sender: TObject; ErrorCode DownloadNo:
Smallint);
begin
  If errorcode = 0 then i      //indien geen downloadproblemen
  begin
    Spirit1.PlaySystemSound(3);
    Messagedlg('Download Succesful',MTInformation,[Mbok],0);
  end
  ELSE                          //indien downloadproblemen
  begin
    Messagedlg('Download Failed',MTError,[Mbok],0);
  end;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Spirit1.SelectPrgm(1);       //stokeer in pgr 2
  spirit1.BeginOfTask(2);       //initialiseert het downloaden van de code
  Spirit1.SetSensorType(0,1);   // 0 =poort 1, 1=switch
  Spirit1.SetPower('02',2,7);   //02:motor, 2:cte, 7:vermogen
  Spirit1.Loop(2,0);            //2: cte, 0: loop forever
  Spirit1.If_(9,0,2,2,1);       //9:cte, 0: druksensor, 2: = ,2: cte, : als sensor in is doe:
  Spirit1.SetRwd('02');        //indien sensor ingedrukt : achteruit rijden en draaien
  Spirit1.Wait(2,100);
  Spirit1.Off('2');            //zet motor op poort C uit
```

```

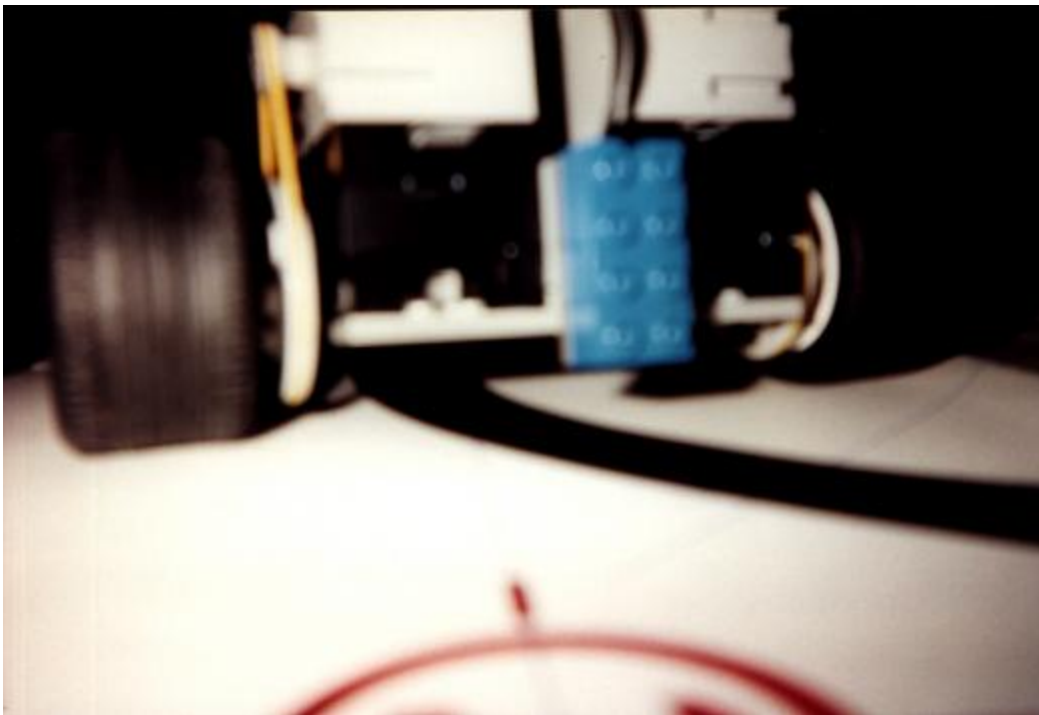
Spirit1.Wait(2,100);
Spirit1.Off('0');           //zet motor op poort A uit

Spirit1.Else_ ;             // wanneer sensor niet is ingedrukt doe:
Spirit1.SetFwd('02');       //stelt de draaizin van de motoren op poort A en C in
Spirit1.On_('02');          //laat RCX vooruit rijden

Spirit1.EndIf;              //sluit de .if ... .else structuur af
Spirit1.EndLoop;            //sluit de loop af
Spirit1.EndOfTask;          //stop met download
end;

```

7.3 Het downloaden van een programma met configuratie van de lichtsensor.



In dit programma maken we gebruik van dezelfde robot als in het voorgaande programma, maar in plaats van een druksensor gebruiken we nu een lichtsensor die naar de grond is gericht en hier twee centimeter bovenstaat. Deze sensor wordt aangesloten op poort 3.

De bedoeling van dit programma is de robot een ellipsvormige, zwart gekleurde baan te laten volgen. Uiteraard zal de robot steeds rechtdoor willen gaan, maar de lichtsensor controleert of de RCX zich nog wel degelijk boven de zwarte lijn bevindt.

Is dit niet zo, dan zal één van de motoren worden uitgeschakeld, waardoor de RCX zal draaien, totdat hij weer boven de zwarte lijn komt, waarna hij weer rechtdoor beweegt, waarbij hij zichzelf continu controleert en eventueel aanpast.

Voordat we het programma kunnen downloaden in de RCX dienen we een klein programmaatje te schrijven opdat de RCX zou weten wanneer hij zich boven de zwarte lijn bevindt en wanneer hij hiervan is afgeweken.

Dit kan eenvoudig daar grosso modo elke kleur kan worden waargenomen (zie paragraaf 6.2) en een cijferwaarde heeft. Zo heeft zwart, 29 als cijfervoorstelling.

Dus door deze waarde te gebruiken in een while_ ... endwhile statement, kan men aan de RCX duidelijk maken welke acties hij moet ondernemen wanneer hij al-dan-niet op de zwarte band staat.

7.3.1 Opstellen van de componenten op de form.



Component	Property	Waarde
SPIRIT	NAME	Spirit1
FORM	NAME CAPTION	Form1
BUTTON	NAME CAPTION	Button1 Set Up sensor
BUTTON	NAME CAPTION	Button2 Download Program

7.3.2 Opstellen van de code.

```

procedure TForm1.Button1Click(Sender: TObject);
//Dit deel van het programma zorgt ervoor dat men de zwartwaarde kan aflezen op het LCD
//scherm van de RCX en dit door op de knop view te drukken en het pijltje bij sensor3 te
//brengen.
begin
spirit1.SelectPrgm(3);           //programma slot 4
spirit1.SetSensorMode (2,0,0);   //sensorpoort3,raw,slope:absoluut
spirit1.SetSensorType (2,3);     // sensorpoort3,licht sensor
spirit1.SetPower('02',2,7);     // '02'motoren op poorten A en C,2 powermode, 7 power
spirit1.SetVar(5,2,2);          //5:varno,2:source, 2:number
end;

```

```

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
spirit1.CloseComm;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
spirit1.InitComm;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
spirit1.SelectPrgm(3);
spirit1.BeginOfTask(1);
    spirit1.SetVar(5,2,5);
    spirit1.SetVar(6,2,29);           //6:varno,2:const,29:zwartwaarde, gestockkeerd
                                     // in register 6 (zie later)

    spirit1.SetSensorType(2,3);
    spirit1.SetPower('02',2,6);
    spirit1.On_('02');
    spirit1.Loop(2,0);               //loop forever
        spirit1.While_(9,2,0,0,6);
                                     //wanneer de sensorwaarde(9)van sensor op poort 3(2) groter is
                                     //(0) dan de waarde gestockkeerd in register(0) 6 (6) waar 29
                                     //instaat doe dan:

        spirit1.Off('2');           // zet motor op poort C uit waardoor de robot
                                     // zal draaien
        spirit1.Wait(0,5);           //wacht gedurende de tijdswaarde, gestockkeerd
                                     //in register 5 voordat het programma verder
                                     //gaat
        spirit1.EndWhile;           //else statement in DELPHI (anders doe: )
        spirit1.On_('2');           //motor c wordt terug aangezet, de robot rijdt
                                     //terug vooruit
        spirit1.EndLoop;           //einde van loop
        spirit1.EndOfTask;
end;

```


7.4 Het Proximity programma.

Buiten kleuren onderscheiden kunnen we de lichtsensoren ook gebruiken als een nabijheidsensor om te detecteren of er al dan niet een obstakel in de robots baan ligt. Dit is mogelijk omdat we op regelmatige intervallen, de RCX infra-roodsignalen kunnen laten uitzenden.

De lichtsensoren kan nu schommelingen in deze uitgezonden infrarood signalen vaststellen en interpreteren, waardoor de RCX uiteindelijk weet of er eventueel een voorwerp/muur in zijn baan ligt, om vervolgens de opgelegde handelingen uit te voeren

7.4.1 Opstellen van de componenten op de form.



Component	Property	Waarde
SPIRIT	NAME	Spirit1
FORM	NAME	Form1
	CAPTION	Download Programma
BUTTON	NAME	Button1
	CAPTION	Download

7.4.2 Opstellen van de code.

```
procedure TForm1.Button1Click(Sender: TObject);
const                                     //declaren van constanten
Last = 10;                               //stelt het register in
Fluct = 11;                             //stelt de gevoeligheid in hoe hoger, hoe gevoeliger
begin
spirit1.SelectPrgm(3);
spirit1.BeginOfTask(1);                  //opdracht 1
spirit1.SetVar(fluct,2,100);
```

```

spirit1.starttask(1);
spirit1.StartTask(2);
spirit1.EndOfTask;

spirit1.BeginOfTask(1);
    spirit1.Loop(2,0);           //loop forever
    spirit1.SendPBMessage(2,0);
    spirit1.wait(2,1);          //elke 10ms wordt een radar signaal uitgestuurd
    spirit1.Endloop;
spirit1.EndOfTask;

spirit1.BeginOfTask(2);        //opdracht 2
spirit1.SetSensorType(2,3);    //light;
spirit1.SetSensorMode(2,0,0);  //raw
spirit1.SetFwd('02');
spirit1.On_('02');
spirit1.Loop(2,0);
    Spirit1.SetVar(last,9,2);
    Spirit1.SumVar(last,0,fluct); //store result
    Spirit1.If_(9,2,0,0,last);
        spirit1.SetRwd('02');
        spirit1.Wait(2,100);
        spirit1.Off('2');
        spirit1.Wait(2,100);
        spirit1.off('0');
    spirit1.Else_;
    spirit1.SetFwd('02');
    spirit1.On_('02');
    Spirit1.EndIf;
Spirit1.EndLoop;
Spirit1.EndOfTask;

end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    Spirit1.InitComm;

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    spirit1.CloseComm;
end;

```

8. De datalog.

8.1 Inleiding.

De *datalog* laat ons toe gegevens van timers, variabelen en sensoren op te slaan. Om de datalog te kunnen gebruiken moeten we initiëel aangeven hoeveel elementen we wensen te registreren. Dit kan op een eenvoudige manier gebeuren door gebruik te maken van de functie *SetDatalog(Size)*. Elk element neemt in het geheugen drie bytes in.

Naast deze *SetDatalog(Size)* zijn er nog andere vaak voorkomende functies, we geven een overzicht:

- § *DatalogNext(Source, Number)* : deze functie laat toe op elk moment in het programma een waarde op te slaan in de datalog. De variabelen *Source* en *Number* kunnen de volgende waarden aannemen (zie tabel1).

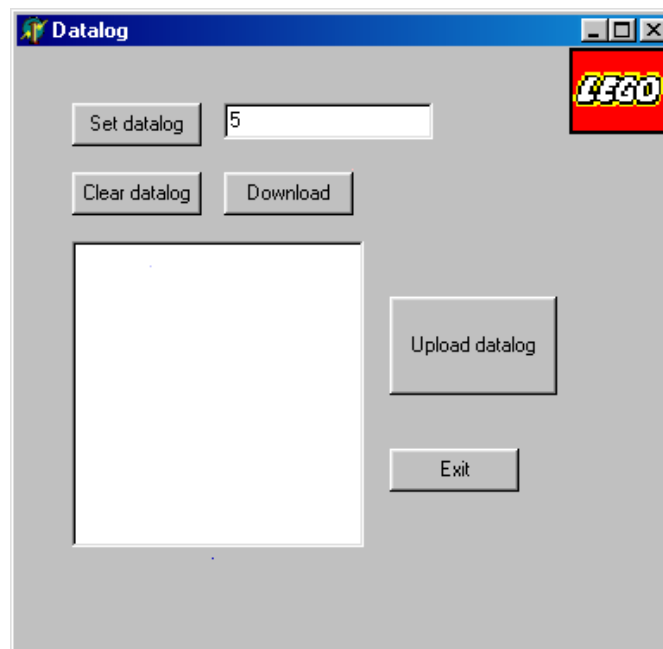
Source	Number
0 VAR	0-31
1 TIMER	0-3 Timer_1, Timer_2, Timer_3, Timer_4
9 SENVAL	0-2 Sensor_1, Sensor_2, Sensor_3
14 WATCH	0

Tabel 1

- § *UploadDatalog(From,Size)* : opvragen van de in de datalog opgeslagen gegevens

8.2 Testen van de datalog.

Aan de hand van een programma tonen we hoe de datalog kan benaderd worden.



8.2.1 Opstellen van de componenten op de form.

Component	Property	waarde
FORM	NAME CAPTION	Form1 Datalog
SPIRIT	NAME	Spirit1
BUTTON	NAME CAPTION	cmdSetDLSize Set Datalog
BUTTON	NAME CAPTION	cmdClearDL Clear Datalog
BUTTON	NAME CAPTION	cmdUploadDL Upload Datalog
BUTTON	NAME CAPTION	cmdDownload Download Program
EDITBOX	NAME CAPTION	Edit1 5
LABEL	NAME CAPTION	lblDatalog (Leeg laten)
MEMO	NAME CAPTION	Memo1 (leeg laten)
BUTTON	NAME CAPTION	cmdExit Exit

8.2.2 Opstellen van de code.

procedure TForm1.CmdClearDLClick(Sender: TObject);

begin

spirit1.SetDatalog(0); *//Clear Datalog*

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

spirit1.SelectPrgm(3); *//Selects program 4*

spirit1.BeginOfTask(1); *//Initialises a task download*

spirit1.SetSensorType(1,3); *//Input 1 setup for a Light Sensor*

spirit1.SetVar(10,2,1234); *//Setvar(VarNo,Source,Number) VarNo=variable number to be set
//Source=adreses the type and the source of the new value=Number*

spirit1.Loop(2,3);

 spirit1.DatalogNext(1,3); *//The PBrick increments its internal datalog pointer/datalog
sensor 3*

 spirit1.Wait(2,100);

spirit1.EndLoop ;

spirit1.DatalogNext(9,1);

spirit1.datalognext(0,10);

 spirit1.DatalogNext(1,3);

spirit1.EndOfTask; *//Ends the download sequence*

```

end;
procedure TForm1.cmdexitClick(Sender: TObject);
begin
spirit1.CloseComm; //Closes the COM-port
end;

procedure TForm1.CMDSetDLSizeClick(Sender: TObject);
begin
if spirit1.SetDatalog(strtoint(form1.edit1.text)) then //This procedure sets the datalog size, if
//there is not enough memory, the statement spirit1.SetDatalog(strtoint(form1.edit1.text)) fails
//and a message appears
lbdatalog.caption := 'datalog size set to '+edit1.text
else
lbdatalog.caption := 'Not enough memory available'
end;

procedure TForm1.cmdUploaddlClick(Sender: TObject);
var
arr :variant;
iCounter : integer;
begin //Returns the low bound of a dimension in a Variant array
arr := spirit1.UploadDatalog(0,strtoint(edit1.text)+1);
for icounter := vararraylowbound(arr,2) to vararrayhighbound(arr,2) do
begin
memo1.Lines.Add(' type: ' + inttostr(arr[0,icounter])+ ' No. ' + inttostr(arr[1,icounter])+ ' value '+inttostr(arr[2,icounter]));
end;
inc(icounter);
end;
//We 'd like to start from the first element in the datalog, and continue until the end of the
//datalog is reached. To return these values, we can interpret the datalog as a two
//dimensional array. If the array is a valid array, the lower bound is found and the
//upperbound of the array is found. Then for each element between these two values there is
//an entry (table 2).

```

<i>Type</i>		<i>Number</i>	<i>Readings</i>
0	VAR	0-31	<i>Readings returned</i>
1	TIMER	0-3	
9	SEINVAL	0-2	
14	WATCH	0	

Table 2

```

procedure TForm1.FormCreate(Sender: TObject);
begin
spirit1.InitComm; //Initialises the COM-port
end;

```

8.2.3 Werking van het programma.

We testen nu het programma. We sluiten een lichtsensor aan op de derde ingang. We vullen een waarde in het tekstvak in. We starten het programma. Wanneer het programma is gestopt klikken we op *Upload Datalog*. Er verschijnen zeven lijnen. De eerste regel geeft de grootte van de datalog aan. Deze bedraagt minstens 1 daar de grootte op zichzelf deel uitmaakt van de datalog. De tweede, derde en vierde regel geven informatie over de timers, de volgende regel over de sensor. De gegevens van de andere variabelen worden uitgelezen in de twee laatste regels.

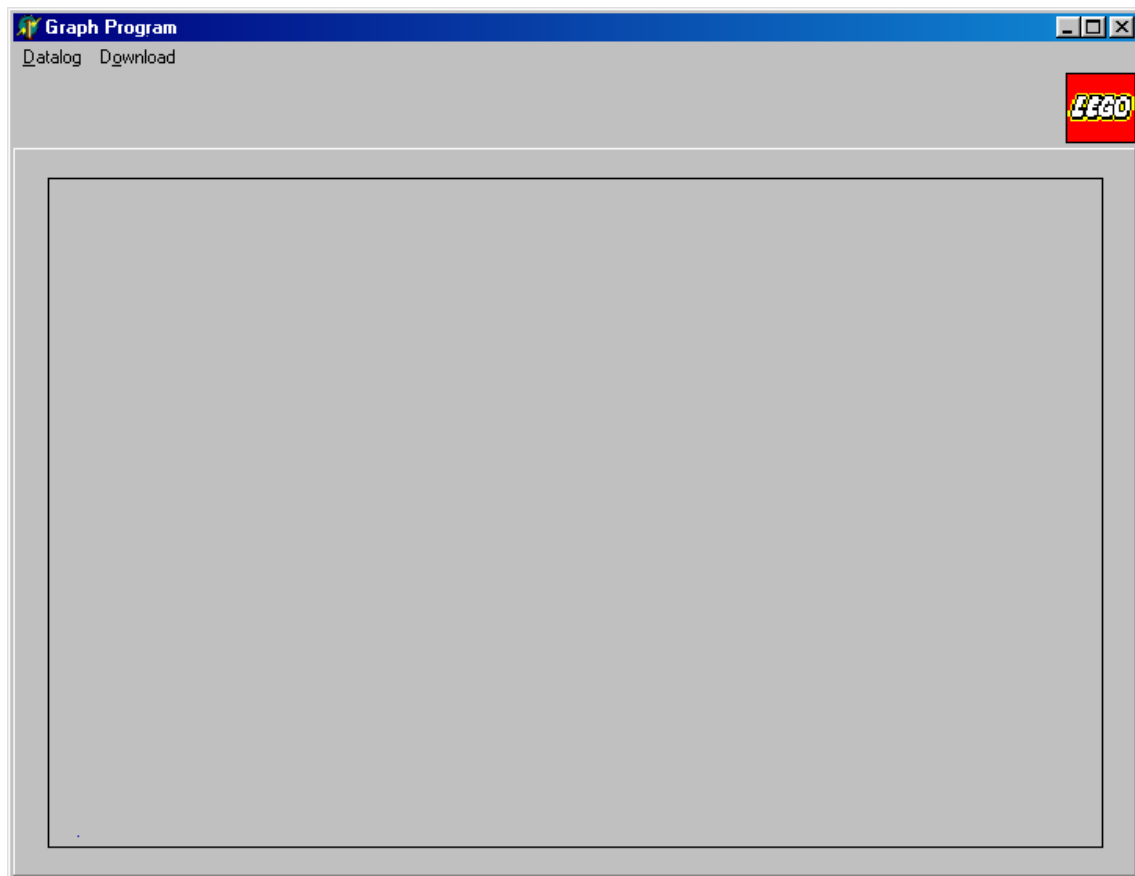
Om alle gegevens, opgeslagen in de datalog te verwijderen klikt men op *Clear Datalog*.

Opmerking: Wanneer men op *Set Datalog* klikt verschijnt er op het scherm van de RCX een cirkel. Tijdens het uitvoeren van het programma zal dit cirkeltje per kwadrant opgevuld worden. Na het verwijderen van alle gegevens uit de datalog verdwijnt dit cirkeltje.

8.3 In grafiek brengen van de gegevens in de datalog

We hebben reeds vermeld dat met behulp van de functie *UploadDatalog* de geregistreerde gegevens in de datalog kunnen weergegeven worden. Het is nu mogelijk deze waarden grafisch voor te stellen.

In het volgende geven we een verklaring van het programma.



8.3.1 Opstellen van de componenten op de form.

Component	Property	waarde
FORM	NAME CAPTION	Form1 Graph Program
SPIRIT	NAME	Spirit1
MAIN MENU	NAME	MainMenu1
MENU-ITEM	NAME CAPTION	Datalog1 &Datalog
MENU-ITEM	NAME CAPTION	SetDatalog1 &Set Datalog
MENU-ITEM	NAME CAPTION	UploadDatalog1 &Upload Datalog
MENU-ITEM	NAME CAPTION	ClearDatalog1 &Clear Datalog
MENU-ITEM	NAME CAPTION	Exit1 E&xit
MENU-ITEM	NAME CAPTION	Five1 &Five
MENU-ITEM	NAME CAPTION	Ten1 &Ten
MENU-ITEM	NAME CAPTION	Fifty1 &Fifty
MENU-ITEM	NAME CAPTION	Onehundred1 &One Hundred
MENU-ITEM	NAME CAPTION	FiveHundred1 Fi&ve Hundred
MENU-ITEM	NAME CAPTION	Download1 D&ownload
MENU-ITEM	NAME CAPTION	ProximityProgram1 &Proximity Program
TCHART	NAME	Chart1

8.3.2 Opstellen van de code.

```
procedure TForm1.Eit1Click(Sender: TObject);  
begin  
spirit1.CloseComm; //Closes the COM-port  
close;  
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);  
begin  
spirit1.InitComm; //Initialises the COM-port  
end;
```

```

procedure TForm1.Five1Click(Sender: TObject);
begin
spirit1.SetDatalog(5); //Initialises a datalog of 5 elements
end;
procedure TForm1.Ten1Click(Sender: TObject);
begin
spirit1.SetDatalog(10); //Initialises a datalog of 10 elements
end;

procedure TForm1.Fifty1Click(Sender: TObject);
begin
spirit1.SetDatalog(50); //Initialises a datalog of 50 elements
end;

procedure TForm1.Onehundred1Click(Sender: TObject);
begin
spirit1.SetDatalog(100); //Initialises a datalog of 100 elements
end;

procedure TForm1.FiveHundred1Click(Sender: TObject);
begin
spirit1.SetDatalog(500); //Initialises a datalog of 500 elements
end;

procedure TForm1.ProximityProgram1Click(Sender: TObject);
const lastreading=10; //Defining constants
      fluctuation=11;
begin
with spirit1 do begin
SelectPrgm(3); //Selects program 4
BeginOfTask(1); //Initialises a task download
SetVar(fluctuation,2,100); //setvar(VarNo,Source,Number) VarNo=variable number to be set
                        //Source=addresses the type and the source of the new value=Number
StartTask(1); //Starts and runs task 1
StartTask(2);
EndOfTask; //Ends the download sequence
BeginOfTask(1);
Loop(2,0); //Loop forever
SendPBMMessage(2,0); //The Pbrick sends a message on the IR-communication channel
Wait(2,5); //Waits for 50 milliseconds
EndLoop; //End of Loop
EndOfTask;
BeginOfTask(2);
SetSensorType(2,3); //Input 2 setup for a Light Sensor
SetSensorMode(2,0,0); //Sets the Sensormode of input 2 to raw data, measurement is
// absolute
SetFwd('02'); //Sets the direction of the motors on input 1 and 3 to forward
On_('02'); //Starts the motors
StartTask(3);
Loop(2,0);

```



```

SetVar(lastreading,9,2);
SumVar(lastreading,0,fluctuation);
If_(9,2,0,0,lastreading);
SetRwd('02'); //Sets the direction of the motors on input 1 and 3 to reverse direction
Wait(2,100);
Off('2'); //Stops motor on input 3
Wait(2,100);
SetFwd('02');
On_('2');
EndIf;
EndLoop;
EndOfTask;
BeginOfTask(3);
Loop(2,100);
DatalogNext(9,2); //The PBrick increments its internal datalog pointer/datalog sensor 3
wait(2,10);
EndLoop;
Off('02');
StopAllTasks; //Stops all the tasks
EndOfTask;
end;
end;

```

```

procedure TForm1.UploadDatalog1Click(Sender: TObject);
var itime:integer;
    i,icounter:integer;
    arr:variant;
    ix,iupper,ilower:integer;
    iminx,imaxx,iminy,imaxy:integer;
begin
arr:=spirit1.UploadDatalog(0,1);
iupper:=arr[2,0];
iminx:=0;imaxx:=iupper;
iminy:=500;imaxy:=850;
ix:=0;
series1.clear;
chart1.LeftAxis.Maximum:=imaxy;
chart1.LeftAxis.Minimum:=iminy;
chart1.BottomAxis.Maximum:=iupper;
chart1.BottomAxis.Minimum:=iminx;
itime:=iupper div 50;
for icounter:=0 to itime do
    ilower:=icounter * 50;
if iupper<=50 then
    arr:=spirit1.UploadDatalog(ilower,iupper) else
    arr:=spirit1.UploadDatalog(ilower,50);
iupper:=iupper-50;
if varisarray(arr) then
    for i:=vararraylowbound(arr,2) to vararrayhighbound(arr,2) do
        begin

```

```

        ix:=ix+1;
        series1.addxy (ix,arr[2,i],"clteecolor");
    end;
inc(i);
inc(icounter);
end;

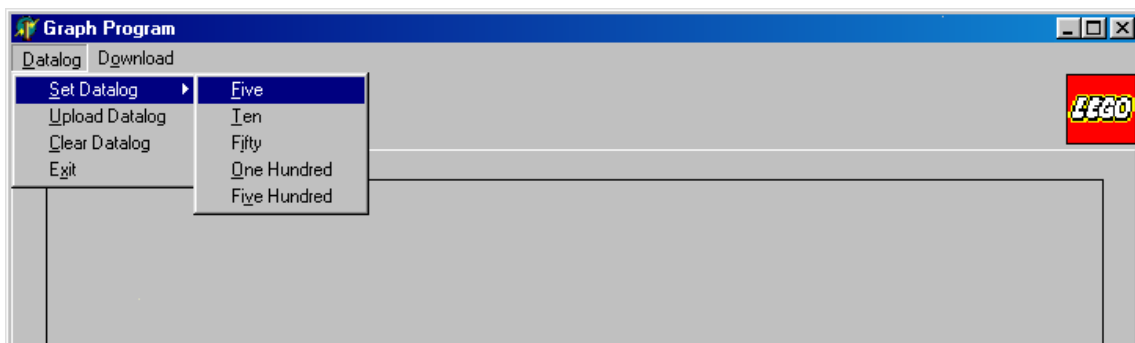
procedure TForm1.ClearDatalog1Click(Sender: TObject);
begin
    spirit1.SetDatalog(0); //Clear Datalog
end;

end.

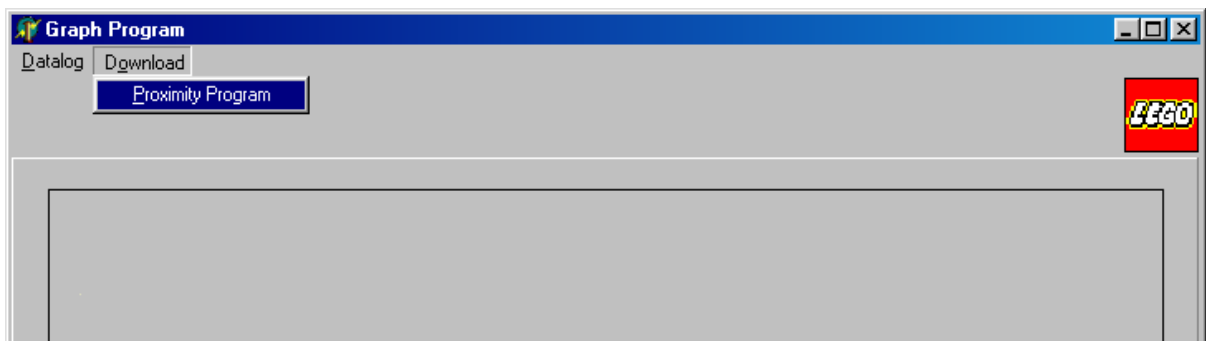
```

8.3.3 Werking van het programma.

We testen nu het programma. We selecteren van het *Datalog-menu*, *Set Datalog* en stellen de grootte van de datalog in op 5.

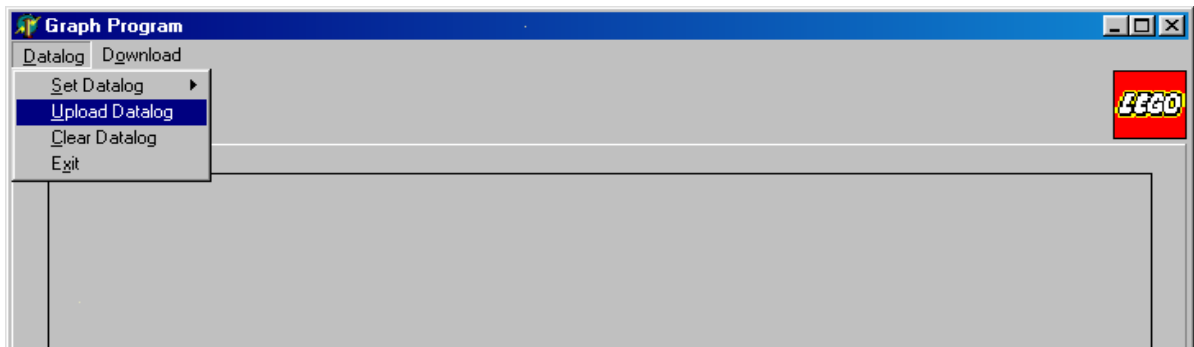


Vervolgens laden we het *Proximity Program* in de RCX.



We drukken op de Run-knop van de RCX en wachten tot het programma is afgelopen.

Dan selecteren we in het Datalog-menu *Upload Datalog*.



Om alle gegevens uit de datalog te verwijderen, klikt men *Clear Datalog* in het Datalog-menu.

9. Bijlage

9.1 INDEX 1: Standaard programma's van de RCX.

- **Programma 1.**

Doel:

De robot een geluidsignaal laten weergeven en laten vooruit rijden.

Configuratie:

Twee motoren, de ene gekoppeld op poort A, de andere op poort C.

- **Programma 2.**

Doel:

Een besturing maken met twee druksensoren en twee motoren. De robot rijdt vooruit indien men de ene sensor indruk, achteruit wanneer men de tweede indrukt.

Configuratie:

De motoren aansluiten op poorten A en C. De druksensoren op poorten 1 en 3
De sensor op poort 1 stuurt de motor op poort A.

- **Programma 3.**

Doel:

Een besturing maken met een lichtsensor en twee motoren. De robot rijdt vooruit totdat de lichtsensor een verandering van lichtintensiteit waarneemt, waarna hij stopt.

Hiermee kan men de RCX bijvoorbeeld een muur laten waarnemen voordat hij deze raakt.

Configuratie:

De lichtsensor die op poort 2 is aangesloten, bestuurt de motoren aangesloten op poorten A en C.

Indien een lichtintensiteit wordt waargenomen tussen de 51% en 100%, rijdt de robot. Indien een lager percentage wordt waargenomen, dan stop de RCX.

- **Programma 4.**

Doel:

De robot vijf keer vooruit en achteruit laten gaan gedurende een willekeurige tijd.

Configuratie:

Er moeten twee motoren gekoppeld worden op poorten A en C.

- **Programma 5.**

Doel:

De robot vooruit laten rijden tot hij een versperring tegenkomt, waarna hij van richting verandert.

Configuratie:

Er worden twee motoren gekoppeld op poorten A en C dewelke bestuurd worden door een druksensor, aangesloten op poort 1.

Wanneer de druksensor is ingedrukt, zal de draairichting van de motoren gedurende één seconde worden omgekeerd (*Wait*) waarna de motor op poort C zal worden uitgezet, en de motor op poort A zal terug in de andere richting draaien, waardoor de robot zal draaien. Na een halve seconde zal ook de andere motor in dezelfde zin draaien, waardoor de robot zich weer vooruit beweegt.

9.2 INDEX 2: Properties en Events van de Spirit component.

- **InitComm**

InitComm initialiseert de computers seriële communicatie poort. Er wordt geen informatie uitgewisseld tussen de brick en de PC

De communicatiepoort COM1 ... COM4 kan worden geselecteerd met behulp van de property *ComPortNo*.

Dit commando moet als eerste worden aangeroepen.

DEEL	BESCHRIJVING
RETOURWAARDE	indien de initialisatie lukt is de retour waarde TRUE Anders is de retour waarde FALSE

Voorbeeld:

```
If Spirit1.InitComm Then           //Wanneer de property ComPortNo waarde 1 heeft
    Label1.Caption := 'OK'         //wordt de communicatie geïnitialeerd met Comm-
Else                               //poort 1.
    Label1.Caption := 'Failure';   //Indien Failure is de kapbel van de Tower op een ver-
End;                              //keerde poort aangesloten.
```

- **CloseComm**

CloseComm sluit de seriële poort zodat andere andere applicaties de poort kunnen gebruiken.

- **UnLockFirmware(UnLockString)**

Dit commando wordt gebruikt om NA *DownLoadFirmware* de RCX te ontsluiten.

Voorbeeld:

```
Label1.Caption := Spirit1. UnLockFirmware('Do you byte, when I knock?')
//enkel deze string is mogelijk als unlock key
```

- **DownloadFirmware(FileName)**

Dit commando download de firmware in de microprocessor en wist tegelijkertijd alle in de RCX opgeslagen programma's.

Het downloaden kan een paar minuten duren.

DEEL	BESCHRIJVING
FILENAME	Bevat zowel directory- als bestandsnaam van de firmware
RETOURWAARDE	Indien de start van downloaden lukt, is de retourwaarde TRUE Anders is ze FALSE
	De DownloadStatus property bevat informatie over de hoeveelheid downgeloadde bytes, de geschatte downloadtijd en het downloadtype.
	Het resultaat van de firmwaredownload kan worden getoond m.b.v. de DownloadDone property. 0: Alles in orde 1: Downloaden mislukt

Voorbeeld:

```
Spirit1.DownloadFirmware('Firm0309.LGO'); //download de firmware naar de brick
                                           //vanuit bestand Firm0309.LGO
```

- **SetWatch(Hours, Min)**

Stelt de RCX' Software klok in en i van het 24 uur type (0 tot 23:59)

DEEL	BESCHRIJVING
HOURS	Stelt de uren in (0 tot 23)
MIN	Stelt de minuten in (0 tot 59)
RETOUR WAARDE	Indien de functie lukt is de retour waarde TRUE Anders is ze FALSE

Voorbeeld:

```
Spirit1.SetWatch (13,22) //zet de interne klok op 13:22
```

- **PBPowerDownTime(Time)**

Met deze property kan men de tijd waarna de RCX zichzelf automatisch uitschakelt instellen. De default waarde is 15 minuten. D.w.z dat indien er gedurende 15 minuten geen enkel commando gegeven wordt aan de RCX of op geen van de vier besturingsknoppen wordt gedrukt, de RCX zichzelf uitschakelt.

DEEL	BESCHRIJVING
TIME	waarde van 0 tot 255 0 ⤵ niet automatisch uitschakelen 1 – 255 ⤵ aantal minuten voor automatisch af te sluiten
RETOUR WAARDE	Indien de functie lukt is de retour waarde TRUE Anders is ze FALSE

Voorbeeld:

Spirit1.PowerDownTime (120) //Na 2 uur sluit de RCX zichzelf af

- **PBBATTERY**

Controleert de batterij in de RCX. De functie retourneert het voltage van de batterij. De waarde is een gemiddelde gemeten over 30 seconden.
 Het opstarten van een motor beïnvloedt deze waarde niet.

DEEL	BESCHRIJVING
RETOUR WAARDE	Indien de functie lukt is de retour waarde een cijfer die het batterijvoltage weergeven in milliVolt. Anders is de waarde 0. Indien de RCX niet bereikbaar is, zal een error worden weergegeven.

Voorbeeld:

Label1.caption := Spirit1.PBBattery

- **PBAliveOrNot**

Deze property wordt gebruikt om op een snelle manier te controleren of de RCX in de mogelijkheid is om te antwoorden.

Zowel de kabel, de tower als de RCX worden hierdoor getest.

DEEL	BESCHRIJVING
RETOUR WAARDE	<p>Booleaans: Indien de functie lukt is de retourwaarde True Anders is de waarde False.</p> <p>Indien er geen antwoord van de RCX verkregen wordt binnen de TimeOut periode, wordt de retour waarde op False gezet.</p>

Voorbeeld:

```
If Spirit1.PBAliveOrNot Then
    Label1.Caption :=('OK')
Else
    Label1.Caption := ('Unable to communicate with RCX');
End;
```

- **PBTurnOff**

The RCX stopt alles en sluit zichzelf af.

DEEL	BESCHRIJVING
RETOUR WAARDE	<p>Indien de functie lukt is de retourwaarde TRUE Anders is de waarde FALSE.</p>

Voorbeeld:

```
If Spirit1. PBTurnOff Then
    Label1.Caption :=('RCX is turned off')
Else
    Label1.Caption := ('Unknown Status');
End;
```

- **PBTxPower(Number)**

Stelt de Infra-Rood transmitter sterkte in op korte afstand of lange afstand.

DEEL	BESCHRIJVING
NUMBER	0 : Korte afstand modus 1 : Lange afstand modus
RETOUR WAARDE	Indien de functie lukt is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

```
If Spirit1. PBTxPower(1) Then
    Label1.Caption := ('Tower is ingesteld voor Lange afstand')
Else
    Label1.Caption := ('Tower is ingesteld voor Korte afstand');
End;
```

- **TowerAndCableConnected**

Wordt gebruikt om te detecteren of de kabel en tower correct zijn aangesloten.

DEEL	BESCHRIJVING
RETOUR WAARDE	Indien de functie lukt is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

```
• If Spirit1. TowerAndCableConnected Then
    Label1.Caption := ('OK')
Else
    Label1.Caption := ('HARDWARE FOUT');
End;
```

- **TowerAlive**

Wordt gebruikt om de status van de tower (zowel hardware als batterij) te controleren.

Belangrijk: Moet steeds na de *TowerAndCableConnected* property komen.

DEEL	BESCHRIJVING
------	--------------

RETOUR WAARDE	Indien de tower 'alive' is, is de retourwaarde TRUE Anders is de waarde FALSE.
---------------	---

Voorbeeld:

```

• If Spirit1.TowerAlive Then
    Label1.Caption := ('OK')
Else
    Label1.Caption := ('Er is een fout');
End;
```

- **On_(MotorList)**

Geeft voeding door naar de motoren die tussen de haakjes staan. De motoren worden door dit commando enkel startklaar gemaakt, draaisnelheid, richting,... worden niet ingesteld met deze property, de motor blijft stilstaan.

DEEL	BESCHRIJVING
------	--------------

MOTORLIST	String: Duidt de poorten aan van de motoren die moeten worden aangezet. Motorlist kan dus enkel de cijfers 0, 1 of 2 bevatten, wat duidt op de motoren (resp.) op poort A, B en C
-----------	---

RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.
---------------	--

Voorbeeld:

```
Spirit1.On_('02')           //motoren op poort 1 en 3 zijn in AAN-toestand gezet
```

- **Off(MotorList)**

Dit commando stopt alle motoren die in de motorlist staan simultaan.

Alle andere instellingen van de motoren (draaisnelheid, richting,...) blijven behouden

DEEL	BESCHRIJVING
MOTORLIST	String: Duidt de poorten aan van de motoren die moeten worden aangezet. Motorlist kan dus enkel de cijfers 0, 1 of 2 bevatten, wat duidt op de motoren (resp.) op poort A, B en C
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

Spirit1.Off('02') //motoren op poort 1 en 3 zijn in UIT-toestand gezet

- **SetFwd(MotorList)**

Dit commando laat alle motoren die in de motorlist staan simultaan in dezelfde richting draaien. Alle andere instellingen van de motoren (draaisnelheid, Aan/Uit,...) blijven behouden

DEEL	BESCHRIJVING
MOTORLIST	String: Duidt de poorten aan van de motoren die moeten draaien. Motorlist kan dus enkel de cijfers 0, 1 of 2 bevatten, wat duidt op de motoren (resp.) op poort A, B en C
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

Spirit1.Setfwd('2') //motor op poort 3 draait in de richting Forward

- **SetRwd(MotorList)**

Zelfde als *SetFwd*, enkel de draaizin wordt gewijzigd. Hier de omgekeerde zin als bij *SetFwd*.

- **SetPower(MotorList, Source, Number)**

Definieert het vermogen aan de as van de motoren geselecteerd in de motorlist. Alle andere instellingen van de motoren zoals Aan/Uit, richting,... wordt niet geraakt.

De verandering in nuttig vermogen aan de assen van de verschillende motoren gebeurt simultaan.

DEEL	BESCHRIJVING
MOTORLIST	String: Duidt de poorten aan van de motoren worden waarvan men het vermogen wenst aan te passen. Motorlist kan dus enkel de cijfers 0, 1 of 2 bevatten, wat duidt op de motoren (resp.) op poort A, B en C
SOURCE, NUMBER	Cijfer verwijst naar de power level, bewaard in variabele gestoeerd in NUMBER(van 0 tot 31)
RETOUR WAARDE	Boolean Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

```
Spirit1.SetPower('02',0,15) // 02 : motoren gekoppeld op poort 1 en 3
                             //0,15 verwijst naar de power level, bewaard in variabele 15
```

- **SetSensorType(Number, Type)**

Deze property wordt gebruikt om het soort gebruikte sensor op een poort aan de RCX duidelijk te maken.

DEEL	BESCHRIJVING
NUMBER	Cijfer: Duidt de poorten aan van de sensoren die men gebruikt en. kan dus enkel de cijfers 0, 1 of 2 bevatten, wat duidt op de sensoren (resp.) op poort 1, 2 en 3
TYPE	Cijfer Specificeert het sensortype: 1: Druk 3 Licht
RETOUR WAARDE	Boolean Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **SetSensorMode(Number, Mode, Slope)**

Met deze property kan men de manier van meten instellen.

DEEL	BESCHRIJVING
NUMBER	Cijfer: Duidt de poorten aan van de sensoren die men gebruikt en. kan dus enkel de cijfers 0, 1 of 2 bevatten, wat duidt op de sensoren (resp.) op poort 1, 2 en 3
MODE	<p>De sensor mode</p> <p>0: RAW Analoge data (0 tot 1023)</p> <p>1: BOOLEAN Waar/Niet-Waar</p> <p>2: TRANSITION COUNTER Telt de toestandsveranderingen van de sensoruitgang</p> <p>3 PERIODIC COUNTER Telt ganse periodes. Dus de teller gaat één hoger indien de sensor hoog en laag, of omgekeerd is geweest.</p> <p>4 PERCENT Geeft de waarde weer in %</p>
SLOPE	0: Absolute meting, d.w.z. onder 45% van de schaal is TRUE, Boven 55% van de schaal is FALSE
RETOUR WAARDE	Boolean Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

```
Spirit1.SetSensorType (1,1); //1:De sensor op poort 2 is (1):een druksensor
Spirit1.SetSensorMode(1,1,0) //De sensordata moet booleaans en absoluut worden
                             //weergegeven
```

- **Poll(Source, Number)**

Deze property wordt gebruikt om informatie te verkrijgen over de RCX in het algemeen (zowel over sensoren, motoren,...)

DEEL	BESCHRIJVING
------	--------------

SOURCE, NUMBER Duidt op hetgeen waarover men informatie wenst te verkrijgen
Zie tabel index 3

SOURCE	NUMBER	BESPREKING
0	0-31	InterneVariabele = register
1	0-3	Timer
3	0,1,2	Motor status
8	-	RCX Programma nummer
9	0,1,2	Sensor waarde
10	0,1,2	Sensor type
11	0,1,2	Sensor mode
12	0,1,2	RCX Sensor Raw
13	0,1,2	RCX Sensor Boolean

RETOUR WAARDE Geeft de waarde weer achter dewelke men vraagt.
Indien de RCX onbereikbaar is zal er een fout worden gegeven.

Voorbeeld:

```
Label1.caption := IntToStr(spirit1.poll(9,0));      //9: geef sensorwaarde,  
                                                    //0: van sensor op poort 1
```

- **PlaySystemSound(Number)**

Hiermee kan men de RCX één van de zes voorgedefiniëerde muziek patronen laten afspelen

DEEL	BESCHRIJVING
------	--------------

NUMBER Cijfer Geeft aan welk muziekje men wens te horen.
(Zie tabel.)

NUMBER	SOUND	DOEL
0	'Key Click'	Standaard gebruikt wanneer er op een knop gedrukt wordt
1	'Beep Beep'	Standaard gebruikt als 'acknowledged'
2	'Afnemende frequentie'	Standaard gebruikt op het einde van een succesvolle 'download'
3	'Toenemende frequentie'	Standaard gebruikt op het einde van een succesvolle 'upload'
4	'Buhh'	Error geluid
5	'Snelle Toenemende frequentie'	3x 🎉 geeft Hoera geluid

RETOUR WAARDE Boolean Indien functie lukt, is de retourwaarde TRUE
Anders is de waarde FALSE.

Voorbeeld:

```
Spirit1.PlaySystemSound(4);
```


- **PlayTone(Frequentie, Time)**

Deze property wordt gebruikt om via de luidsprekers van de RCX een muziekstukje te laten horen die in code gecomponeerd is.

DEEL	BESCHRIJVING
------	--------------

FREQUENCY Cijfer Stelt de frequentie in van de noot (0-2000)
Om muzikale stukjes te schrijven: zie tabel.

		OCTAAF							
NOOT		1	2	3	4	5	6	7	8
La	A	28	55	110	220	440	880	1760	3520
La#		29	58	117	233	466	932	1865	3729
Si	B	31	62	123	247	494	988	1976	3951
Do	C	33	65	131	262	523	1047	2093	4186
Do #		35	69	139	277	554	1109	2217	
Re	D	37	73	147	294	587	1175	2349	
Re#		39	78	156	311	622	1245	2489	
Mi	E	41	82	165	330	659	1319	2637	
Fa	F	44	87	175	349	698	1397	2794	
Fa#		46	92	185	370	740	1480	2960	
Sol	G	49	98	196	392	784	1568	3136	
Sol#		52	104	208	415	831	1661	3322	

TIME Cijfer De tijd dat de toon gespeeld wordt, in stappen van
10 ms $1 < \text{TIME} < 255$

RETOUR WAARDE Boolean Indien functie lukt, is de retourwaarde TRUE
Anders is de waarde FALSE.

Voorbeeld:

Spirit1.Playtone(2000,100) //laat een geluid horen van 2000Hz gedurende 1 seconde

- **SetVar(VarNo, Source, Number)**

Deze property verandert de waarde gestockeerd in register(VarNo) tot de waarde Number

DEEL	BESCHRIJVING
VARNO	cijfer Het nummer van het register waarvan men de waarde wilt wijzigen. Overige zie tabel index 3
SOURCE	constante 2 Overige zie tabel index 3
NUMBER	cijfer Nieuwe waarde die in het register moet worden opgeslagen. Moet liggen tussen de -32768 en +32767
RETOUR WAARDE	Boolean Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

Spirit1.SetVar(16,2,33) //wijzigt de waarde in register 16 in 33

- **Bewerkingen met registerwaarden .**

Hiervoor kan men de properties:

- * SumVar(VarNo,Source,Number) Telt de waarde van het register aangegeven door Source en Number op bij het registernummer aangegeven door VarNo en slaat de nieuwe waarde hier in op.
- *.SubVar(VarNo,Source,Number) Zelfde werkwijze enkel vermindert men registerwaarden.
- * DivVar(VarNo,Source,Number) Deelt de waarde in register (VarNo) door de waarde gestockeerd in het register geadresseerd door Source en Number). Het resultaat wordt steeds afgerond naar beneden naar het dichtst bijzijnde integer-waarde. Delen door nul resulteert in het opslaan van de waarde 0 in het register.
- * MulVar(VarNo,Source,Number) Vermenigvuldigt registerwaarden de waarde wordt steeds afgerond zodat het ligt in het 16-bit integer interval [-32768-32767].

Property	Source		
	Var 0	Const 2	VarNo
SumVar(VarNo,Source,Number)	0-31	-32768 tot 32767	0-31
SubVar(VarNo,Source,Number)			
DivVar(VarNo,Source,Number)			
MulVar(VarNo,Source,Number)			

Voorbeeld:

Spirit1.MulVar(2,2,8); //vermenigvuldigd de waarde in register 2 met de constante 8

- **SetEvent(Source, Number, Time)**

Met deze property is de autopolling van de registers mogelijk, om de ingestelde tijd (Time) zal er een controle worden uitgevoerd op het register, waarbij er automatisch gecontroleerd wordt of een waarde in het register is gewijzigd waarna een signaal wordt terug gestuurd.

DEEL	BESCHRIJVING
SOURCE	constante: 0
NUMBER	cijfer: regeisternummer
TIME	cijfer Stelt het stijdsinterval in ms in, waartussen de polls gebeuren.[0...10000ms]

Voorbeeld:

Spirit1.SetEvent(0,6,300); //6: register nr. 6, 300: elke 300ms een poll doen

- **BeginOfTask(Number)**

Start de download sequentie. Alle commando's hierop volgend totaan *EndOfTask* of *EndOfTaskNoDownload* zullen gebufferd worden in de ActiveX control. De eigenlijke compilatie en downloaden van de code begint wanneer het commando *EndOfTask* of *EndOfTaskNoDownload* is bereikt.

Wanneer het downloaden is voltooid zal men dit kunnen weergeven met het *DownloadDone*-event.

DEEL	BESCHRIJVING
NUMBER	zie tabel index2
RETOUR WAARDE	Boolean Indien de start van het downloaden lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **EndOfTask**

Deel van de *BeginOfTask* ... *EndOfTask* sequentie.

Dient om het einde van het te downloaden programma duidelijk te maken.

- **DownloadStatus**

Hiermee krijgt men informatie over de grootte van het gedownloadde programma, geschatte download tijd en task nummer.

- **DownloadDone**

Deze event wordt verkregen wanneer het downloaden gedaan is.

- **EndOfTaskNoDownload**

Deel van de *BeginOfTask ... EndOfTaskNoDownload* sequentie en wordt gebruikt om informatie te winnen over een te downloaden opdracht of over de nodige RAM-ruimte in de RCX die het te downloaden programma nodig heeft.

DEEL	BESCHRIJVING
RETOUR WAARDE	Boolean Indien de start van het downloaden lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

```
Spirit1.BeginOfTask(1);           //initialiseer download naar task 1
....
Spirit1.EndOfTaskNoDownload; // stop de pseudo-download en verkrijg de bock-grootte
                               //enz. in de DownloadStatusEvent
```

- **Loop(Source, Number)**

Deel van de *Loop ... EndLoop* structuur en duidt het begin van de loop aan. Code, geschreven tussen *Loop* en *EndLoop* worden zoveel herhaald als Number met als uitzondering het *Loop(2,0)* welke duidt op een oneindige loop.

DEEL	BESCHRIJVING
SOURCE	constante 2 voor oneindige loops (=FOREVER) cijfer voor eindige loops
NUMBER	constante 0 voor oneindige loop Cijfer voor oneindige loops zie tabel index 3 = aantal keer dat de loop moet doorlopen worden
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

```
Spirit1.Loop(2,0);           //Loop Forever
...
Spirit1.EndLoop;
```

- **EndLoop**

Deel van de *Loop ... EndLoop* structuur en duidt het einde van de loop aan.

DEEL	BESCHRIJVING
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **If_(Source1, Number2, RelOp, Source2, Number2)**

Deel van de *If_...[Else_]...EndIF* structuur en duidt het begin hiervan aan. Komt overeen met de gewone if statement in pascal en wordt uitgevoerd indien de voorwaarde WAAR is anders wordt overgegaan naar de code achter *else_*.

DEEL	BESCHRIJVING										
SOURCE1, NUMBER1	Zijn beide vergelijkwaarden(1) zie tabel index 3										
RelOp	Relatie operator tussen de twee vergelijkwaarden (1) en(2) <table> <tr> <td>Operator</td><td>Constante equivalente</td></tr> <tr> <td>></td><td>0</td></tr> <tr> <td><</td><td>1</td></tr> <tr> <td>=</td><td>2</td></tr> <tr> <td><></td><td>3</td></tr> </table>	Operator	Constante equivalente	>	0	<	1	=	2	<>	3
Operator	Constante equivalente										
>	0										
<	1										
=	2										
<>	3										
SOURCE2,NUMBER2	Zijn beide vergelijkwaarden(2) zie tabel index 3										
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.										

Voorbeeld:

```

Spirit1.If_(0,5,1,2,10)    //indien variabele 5 < constante 10 doe dan:
...
Spirit1.else               // anders doe
...
Spirit1.EndIf

```

- **Else_**

Deel van de If_...[Else_]...EndIF structuur en duidt het 'midden' hiervan aan. Komt overeen met Else in de gewone Pascal.

DEEL	BESCHRIJVING
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **EndIf**

Deel van de If_...[Else_]...EndIF structuur en duidt het 'einde' hiervan aan.

DEEL	BESCHRIJVING
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **Wait(Source, Number)**

Dit commando wordt gebruikt om het programma te pauzeren gedurende Number ms.

DEEL	BESCHRIJVING
SOURCE, NUMBER	zie tabel index 3 Meestal source: 2 en number/100: # seconden
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

Spirit1.wait(2,1000) // pauzeert het programma voor 10 seconden

- **SetWatch(Hours,min)**

Deze property stelt de RCX interne klok in. De klok is van het 24h type.

DEEL	BESCHRIJVING
HOURS	Stelt de uren in (0-23)
MIN	Stelt de minuten in (0-59)
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

Voorbeeld:

Spirit1.setwatch(11,45) // stelt de tijd in op 11h45

- **SendPBMessage(Source,Number)**

Deze property laat de brick een bericht op het IR-communicatie kanaal verrichten. Hiermee is het mogelijk om 2 bricks met elkaar te laten communiceren, zonder enige verbinding met de PC. Dit commando wordt tevens gebruikt wanneer men een lichtsensor als proximity-sensor wenst te gebruiken.

DEEL	BESCHRIJVING
SOURCE, NUMBER	zie tabel index 3
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **SelectPrgm(Number)**

Hiermee stelt men het actieve programmaslot in.

DEEL	BESCHRIJVING
NUMBER	0-4 : 0 komt overeen met programma 1 zie tabel index 3
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **While_(Source1,Number1,RelOp,Source2,Number2)**

Het *While_* commando is deel van de *While_ ... EndWhile* structuur.

De code tussen de *While_ ... EndWhile* statement wordt uitgevoerd zolang aan de voorwaarde, gesteld door de *While_(Source1,Number1,RelOp,Source2,Number2)*-parameter, wordt voldaan.

Dit komt volledig overeen met de reserved words *While (statement) do begin ... end;* in de gewone pascal-taal.

DEEL	BESCHRIJVING										
SOURCE1, NUMBER1	Zijn beide vergelijkwaarden(1) zie tabel index 3										
RelOp	Relatie operator tussen de twee vergelijkwaarden (1) en(2)										
	<table> <tr> <th>Operator</th><th>Constante equivalente</th></tr> <tr> <td>></td><td>0</td></tr> <tr> <td><</td><td>1</td></tr> <tr> <td>=</td><td>2</td></tr> <tr> <td>< ></td><td>3</td></tr> </table>	Operator	Constante equivalente	>	0	<	1	=	2	< >	3
Operator	Constante equivalente										
>	0										
<	1										
=	2										
< >	3										
SOURCE2,NUMBER2	Zijn beide vergelijkwaarden(2) zie tabel index 3										
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.										

Voorbeeld:

```

Spirit1.While_(0,5,1,2,10)    //zolang de waarde, gestockeerd in variabele
                               // 5 < constante waarde 10 doe dan:
...
Spirit1.EndWhile              //indien niet meer aan de voorwaarde wordt voldaan:
                               //ga verder met de rest van de code

```

- **EndWhile**

Deel van de de *While_ ... EndWhile* structuur en duidt het 'einde' hiervan aan.

DEEL	BESCHRIJVING
RETOUR WAARDE	Indien functie lukt, is de retourwaarde TRUE Anders is de waarde FALSE.

- **SetDatalog(Size)**

De datalog laat ons toe gegevens van timers, variabelen en sensoren op te slaan. Om de datalog te kunnen gebruiken moeten we initiëel aangeven hoeveel elementen we wensen te registreren. Dit kan op een eenvoudige manier gebeuren door gebruik te maken van de functie *SetDatalog(Size)*. Elk element neemt in het geheugen drie bytes in.

Door het aanroepen van deze property, wist men het ingestelde datalog.

DEEL	BESCHRIJVING
------	--------------

RETOUR WAARDE Indien functie lukt, is de retourwaarde TRUE
Anders is de waarde FALSE.

Voorbeeld:

Spirit1.SetDatalog(50); //initialiseerd het datalog met een grootte van 50 elementen.

- **DatalogNext(Source,Number)**

Deze functie laat toe op elk moment in het programma een waarde op te slaan in de datalog.

DEEL	BESCHRIJVING
------	--------------

SOURCE, NUMBER

Te bewaren waarde	Source	Number
Var 0-31	0	0-31
Timer 0-3	1	0-3
Input(Sensor waarde) 0-2	9	0-2
Watch	14	0

RETOUR WAARDE Indien functie lukt, is de retourwaarde TRUE
Anders is de waarde FALSE.

Voorbeeld:

Spirit1.DatalogNext(9,1) *//Datalog sensor1*

9.3 INDEX 3: Properties en hun waardenbereik

PROPERTY	SOURCE								MototList	VarNo	RelOp > 0, < 1 = 2, <> 3	Time
	Var	Timer	Const	Motor Status	Random No	Sensor Value	Sensor Type	Sensor Mode				
	0	1	2	3	4	9	10	11				
On(MotorList)à Off(MotorList) SetFwd(MotorList) SetRwd(MotorList) SetPower(MotorList,Source,Number)	0-31	-	0-7	-	-	-	-	-	0,1,2	-	--	-
SetEvent(Source,Number,Time)	0	-	-	-	-	-	-	-	-	-	-	0-10000 ms
Poll(Source,Number)	0-31	0-3	-	0,1,2	0,1,2	0,1,2	0,1,2	0,1,2	-	-	-	-
SetVar(VarNo,Source,Number)	0-31	0-3	-32768 +32767	0,1,2	0,1,2	0,1,2	0,1,2	0,1,2	-	-	0-31	-
Loop(Source,Number)	0-31	-	0-255	-	-	-	-	-	-	-	-	-
If_(Src1,Nbr1,Relop,Src2,Nbr2); While_(Src1,Nbr1,Relop,Src2,Nbr2);	0-31	0-3	-32768 +32767	0,1,2	0,1,2	0,1,2	0,1,2	0,1,2	0	-	-	-
Wait(Source,Number)	0-31	-	1-32767	-	-	-	-	-	-	-	-	-

	Number	Time	Freq	Type 0:geen 1:Druk 3:licht	Mode 0:Raw 1 :Boolean 2 :Trans,counter 3:Period Counter 4:Percent	Slope 0 : Absolute 1-31 Dynamisch	Size	Hours	Min
PROPERTY									
BeginOfTask(Number)	0-9	-		-	-	-			
SetSensorType(Number,Type)	0,1,2	-		-	0-4	0-31			
ClearSensorValue(Number)	0,1,2	-		-	0-4	0-31			
SetSensorMode(Number,mode,slope)	0,1,2	-		-	0-4	0-31			
PlayTone(Frequency,Time)	0-5	1-255 (10ms)		1-20000	-	-			
PlaySystemSound(Number)	0-5	1-255 (10ms)		1-20000	-	-			
PBPowerDownTime(Time)	-	1- 255(min) 0=forever		-	-	-			
SelectPrgm(Number)	0-4	-		-	-	-			
SetWatch(Hours,Minutes)	-	-		-		-		0-23	0-59
SetDatalog(Size)	-	-		-	-	-	0:delete log area 1:available memory		
PBTxPower(Number)	0-1	-		-	-	-			

9.4 INDEX 4 : Nuttige internetadressen.

- <http://www.legomindstorms.com>
- <http://www.legowordshop.com>
- <http://www.legomindstorms.com/sdk/>
- <http://www.graphics.stanford.edu/~kekoa/rcx/>
- <http://www.viktoria.informatics.gu.se/~peter/lego/>
- <http://www.homepages.svc.fcj.hvu.nl/brok//legomind/robo/>
- <http://www.crynwr.com/lego-robotics/>
- <http://www.enteract.com/~dbaum/nqc/index.html>
- <http://www.oreilly.com/catalog/lmstorms/resources/index.html>
- <http://www.lego.com/be/>