Visual Basic:
LEGO roboten programazioa

(Euskaraz 1.01 bertsioa 2001eko ekaina)

Koldo Olaskoaga
(David Hanley eta Sean Hearnek egindako gidaliburuan oinarritua)

Gidaliburu hau egiteko oinarria David Hanley eta Sean Hearne-k egindako “Lego Mindstorms Programming with Visual Basic”[1] proiektua izan da. Neurri handi batean horren itzulpena da, beraz, niri dagokidan lana itzulpena eta hainbat egokitzapenak izan dira. Kapitulu bakoitzaren hasieran kapituluko edukiak jasotzen saiatu naiz, eta edukien antolaketan hainbat aldaketa egin dut. A eranskina erabat berria da eta eskuliburuan zehar zeuden muntaketa instrukzioak ordezkatzen ditu (MLCAD, L3p eta PovRay aplikazioez egindakoak dira). Testuan ere hainbat aldaketa egin diot (txantiloiak, eranskinean spirit.ocx-ko metodoen erreferentziak...).

[1] Ikus bibliografía.

Edukia

Sarrera

"Lego MindStorms-en programazioa: Visual Basic" gidaliburu honen oinarrizko helburuak bi dira:

- Lego MindStorms-ez egindako robotak Visual Basic erabiliz programatzeko behar diren tresnak eskaintzea.
- Visual Basic programazio hizkuntzaren ikasketan hasteko modu erakargarri bat eskaintzea, hain zuzen, roboten programazioaren bitartekoa.

Gidaliburu honen lehenengo kapituluetan kontrola ordenagailutik egingo dugu, horrela, Visual Basic-en formularioak erabiltzen ikasiko dugu. Hortik aurrera, programak robotetara transferituko ditugu, eta bertan egikarituko ditugu.

Visual Basic-eko seigarren bertsioa (gaztelaniaz) erabili dut, baina bosgarrena ere erabil daiteke. Eranskinean Visual Basic erabiliz programatzeko erabil daitezkeen beste erreminten berri ematen da, hain zuzen, Visual Basic for Aplications eta BrickCommand.

1 Robotics Invention System

1.1. Kapitulu honetako edukiak

Kapitulu honetan LEGOk komertzializatzen duen Robotics Invention System-en ikuspegi orokor bat ematen da, hasiberriak oinarrizkoena ezagutu dezan.

1.2. Lego robotak

LEGO MindStorms kit-a robotak muntatzeko elementu multzo bat da. Egiturak eta transmisio mekanikoak egiteko elementuez gain roboten burmuina izango den RCX adreilu adimentsua eskaintzen du.

1.1 irudia
RCX

RCX mikrokontrolatzaile bat da. Sarrera batetik edo gehiagotik jasotzen dituen datuak prozesatu, datu horien arabera erabakiak hartu eta irteerak kontrolatzen ditu. RCXk hiru sarrera eta hiru irteera ditu. Sarreretan sentsoreak konektatzen dira: argi-sentsoreak, ukitze-sentsoreak, tenperatura sentsoreak... Irteerak motoreak edo argiak izan daitezke. Sentsoreak eta motoreak RCXri konektatzeko LEGOren betiko konektore elektrikoak erabiltzen dira, beraz, ez dugu inolako erremintaren beharrik izango..

1.3. Lego roboten programazioa

Helburu zehatz baterako egiten den robotak egitura egokia eta portaera zehatz batzuk behar ditu. Robotak emango dituen erantzunak programazioaren bitartez definituko ditugu.

LEGO MindStorms robotak programazio hizkuntza grafikoen bitartez programa daitezke: RCXCode (LEGO MindStorms-ekin batera komertzializatutako hizkuntza) eta Robolab (hezkuntzarako bereziki diseinatutako hizkuntza). Proiektu aurreratuagoak egiteko konpiladore ezberdinak erabil daitezke, hala nola, Visual Basic, Visual C++, eta Visual Java++.

Programa ordenagailuan konpilatu ondoren RCXra transmitituko dugu infragorrien bitartez. Hortik aurrera robotak era autonomoan funtzionatuko du.

1.2 irudia
sentsoreak eta motoreak RCXn konektaturik

1.4. Spirit.ocx

LEGO MindStorms edo LEGO Technic CyberMaster-ekin batera datorren CD-ROMa instalatzen denean, automatikoki SPIRIT.OCX activeX kontrola instalatzen da ordenagailuan. Kontrol honek programazio inguru ezberdinetatik RCX kontrolatzeko aukera emango digu, eta ezinbestekoa da RCX Visual Basic-ez programatzeko.

1.3 irudia
Programak ordenagailuan editatu ondoren RCXra transferituko ditugu argi infragorriaz

2 Lehen urratsak Visual Basic-en

2.1. Kapitulu honetako edukiak

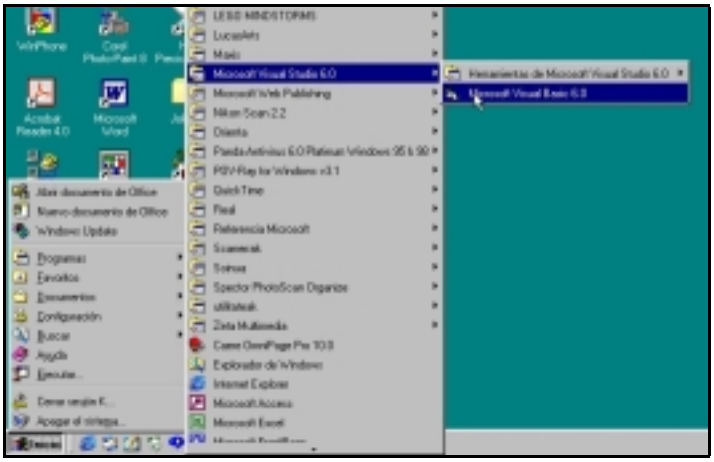
Kapitulu honetan Visual Basic programazio hizkuntzarekin lehen kontaktua izango dugu. Oinarrizko kontzeptuak landuko ditugu, eta ez diogu LEGO adreiluaren programazioari ekingo hurrengo kapitulu arte.

1. Visual Basic-eko oinarrizko kontzeptuekin lehen kontaktua: proiektuak, formularioak eta tresnen koadroa.
2. Propietateen leihoaren oinarrizko esplorazioa.
3. Formularioetan kontrolak nola jarri eta objektuen propietateak aldatzeko prozedura orokorra.
4. Programa egikarigarriak

2.2. Lehen urratsak

Visual Basic zure ordenagailuan instalatu ondoren, hasi zaitezke zure proiektuak egiten. Horretarako, aplikazioa abiarazi beharko duzu, ondoko hau eginez:

- *Hasi* aktibatu
- *Programak* hautatu.
- Bila ezazu *Microsoft Visual Basic 6.0* programa taldea.
- Egizu klik *Visual Basic 6.0* gainean.



2.1 irudia:
Bila ezazu
Visual Basic-eko ikonoa

Pauso hauek guztiak emanda,aurkezpen-leiho baten ondoren, *Nuevo Proyecto* leihoa agertuko da, 2.2 irudian ikusten den eran. Hala gertatuko ez balitz, klik egin *Archivo* menuan eta *Nuevo Proyecto* aukeratu.

Nuevo Proyecto leihoan eskaintzen den aukera kopurua ezberdina izan daiteke Visual Basic-eko bertsio batetik beste batera.

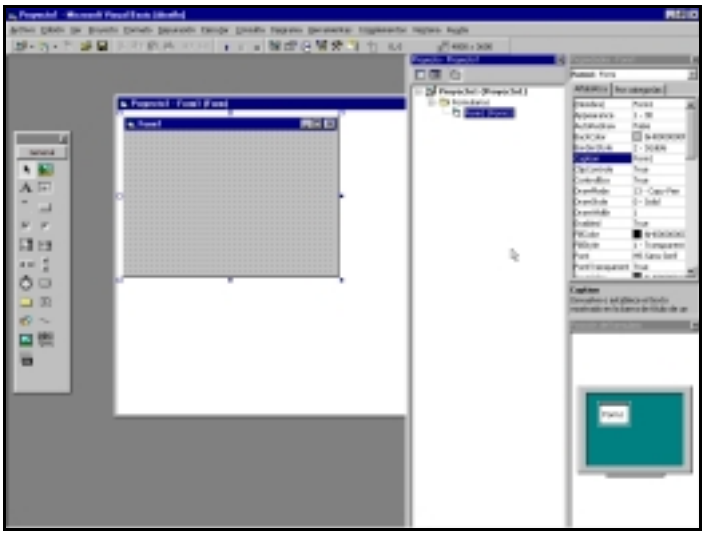
- Hauta ezazu *EXE estándar* aukera proiektu berria sortzeko.

Proiektu berria abiarazi ondoren, mahaigaina 2.3 irudian ikusten den modukoa izango da, edo antzekoa.

Nahiz eta orain arte gauza gutxi egin, proiektua gorde beharko duzu, eta horretarako izen bat eman behar diozu.

2.2 irudia:
Nuevo Proyecto elkarrizketa-leihoan *EXE estándar* aukeratu behar da

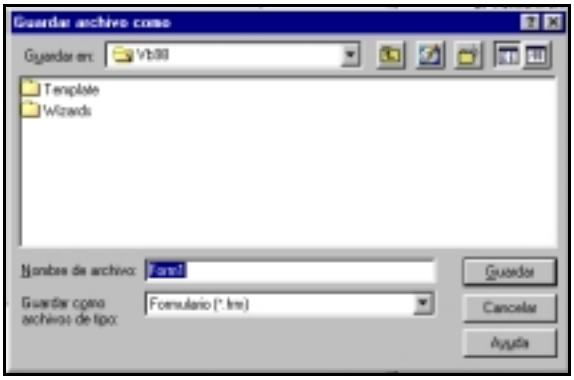
Proiektua gordetzean, bi artxibo sortzen dira: Proiektu artxiboak .VBP luzapena du, eta bertan, Visual Basic-ek proiektua egiteko erabiltzen duen informazioa gordetzen du. Formulario artxiboak .FRM luzapena du, eta formularioari dagokion informazioa gordetzen du.



2.3 irudia:
Visual Basic-eko itxura proiektu berri bat abiatu ondoren

Proiektu bat gorde aurretik, komeni da karpeta berri bat sortzea, bertan proiektuko artxibo guztiak gordetzeko. Beraz, artxiboak gordetzeko ondoko urratsak bete beharko dituzu:

- Hauta ezazu *Archivo* menuko *Guardar Form Como*. Aukera honek aktiboa dagoen formularioa gordeko du.
- Hauta ezazu zure formularioa gorde nahi duzun tokia *Guardar archivo como* elkarrizketa-koadroaren bitartez. Ikastaro honetan zehar sortutako artxibo guztiak C:\VBLego\ direktorioan gorde ditzakezu.



2.4 Irudia
Guardar como elkarrizketa-koadroa. Egin ezazu klik botoian direktorio berria sortzeko.

- Hauta ezazu *Crear nueva carpeta* (2.4 irudia).
- Idatz ezazu direktorio berriaren izena (**kap02**) eta ondoren saka ezazu Return tekla.
- Ondoren, **kap02** direktorioa ireki bere ikonoaren gainean klik-bikoitza eginez.
- Idatz ezazu **Kaixo** *Nombre de archivo* koadroan (Visual Basic-ek .FRM luzapena erantsiko dio gordetzerakoan).
- Egizu klik *Guardar* botoian formularioaren fitxategia gordetzeko.
- Hauta ezazu *Archivo* menuko *Guardar proyecto*. Honen bitartez proiektua bere osotasunean gorde ahal izango duzu.
- Idatz ezazu **Kaixo** *Nombre de achivo* koadroan.
- Egizu klik *Guardar* botoian proiektuaren fitxategia gordetzeko.

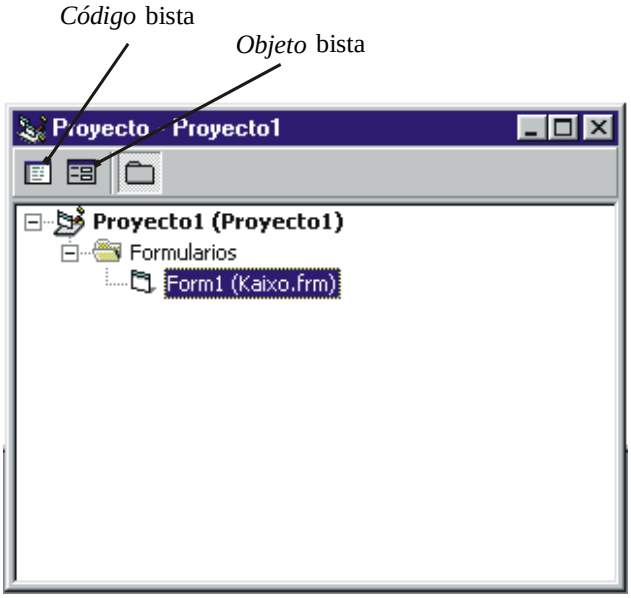
Hemendik aurrera, behin fitxategiei izena emanda, aldaketak gordetzeko nahikoa izango da *Archivo* menuko *Guardar proyecto* hautatzea. Bestela, tresna-barrako *Guardar* ikonoaren bitartez egin dezakezu.

2.3. Proiektuaren esplorazio-leihoa

Orain arte daukaguna Kaixo.vpb proiektua da. Proiektu hau elementu bakar batek osatzen du, hain zuzen, kaixo.frm formularioak. Hala ere, beste zenbait aplikazioek elementu gehiago beharko dute.

Explorador de proyectos leihoak proiektuak dituen artxibo guztien izenak erakusten ditu. Leiho hau agerian ez badago, hauta ezazu *Ver* menuko *Explorador de proyectos.*

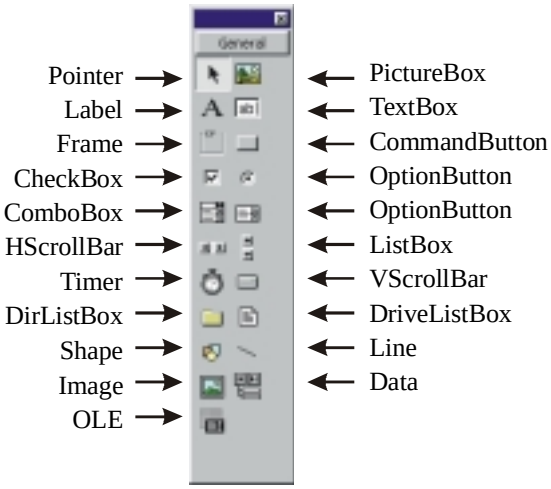
Goiko aldean dituen bi ikonoak aukeratutako objektuaren *Objeto* eta *Código* bistak alternatzeko erabiltzen dira (2.5 irudia).



2.5 irudia
Proiektuaren esplorazio-leihoa

2.4. Tresnen koadroa leihoa

Zure pantailaren ezker aldean tresnen koadroa ikus dezakezu. Bertan Windows-eko kontrol estandarrak jasotzen dira, Windows programa gehienetan agertzen direnak. 2.6 irudiak tresnen koadroa erakusten du. Visual Basic-eko bertsio edo edizioaren arabera, aldaketak egon daitezke tresnen koadroan. Ikusgai ez badago, hauta ezazu *Ver* menuko *Cuadro de herramientas.*



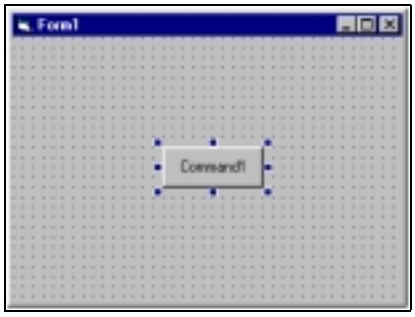
2.6 irudia
Visual Basic-eko tresnen koadroa

2.5. Kontrolen kokatzea formularioan

Orain *CommandButton* kontrol bat jarriko dugu formularioan. Horretarako ondoko urratsak jarraituko ditugu:

- Egizu klik bikoitza tresnen koadroko *CommandButton* ikonoan. Hortik aurrera zure formularioa 2.7 irudian bezala ikusiko duzu.
- Botoi berria aukeratua dagoenean (puntu urdinez inguratua agertzen da) jar ezazu kurtsorea komando botoiaren gainean, eta sakatu eta heldu saguaren ezkerreko botoia. Saguaren ezkerreko botoia sakatuta mantentzen duzun bitartean, mugi ezazu

kurtsorea formularioaren goiko alderantz. Komando botoia nahi den tokian dagoenean, aska ezazu saguaren ezkerreko botoia.



2.7 irudia
Formularioa bere lehen botoiarekin

2.6. Propietateen leihoa

Propiedades leihoa proiektuko objektuen propietateei balioa emateko erabiltzen da. *Propiedades* leihoa agerian ez badago, hauta ezazu *Ver* menuko *Ventana propiedades*.

Objektu baten propietateek bere itxura eta portaera definitzen dute. Adibidez, formularioa objektu bat da. *Caption* propietateak formulario-leihoko izenburuan agertuko den testua definitzen du. Propietatearen izena zerrendaren ezker aldean agertzen da, eskuinean propietateak une honetan duen balioa ikusten den bitartean.

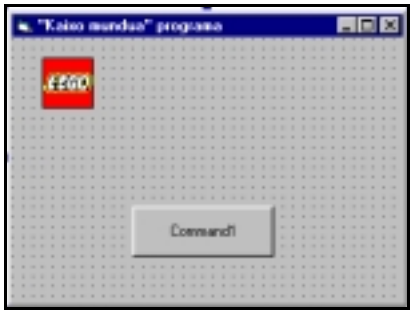
Formularioari **"Kaixo Mundua" programa** izenburua emateko, formularioaren *Caption* propietatearen edukia aldatu behar da.

Egizu klik formularioaren edozein tokitan, komando botoian izan ezik. Orain, *Propiedades* leihoaren izenburuan Propiedades-Form1 irakurri ahal izango dugu (agerian badago) eta formularioaren inguruan karratu txiki urdin batzuk agertuko dira.

2.8 irudia:
Propiedades leihoaren bitartez aukeratutako elementuaren propietateak alda daitezke.

- Egizu klik *Propiedades* leihoan *Caption* hitza duen gelaxkan.
- Besterik egin gabe, idatz ezazu **"Kaixo mundua" programa**.

Hemendik aurrera formularioak 2.9 irudiko itxura izango du.



2.9 irudia:
Orain, formulario-leihoko izenburua adierazkorragoa da.

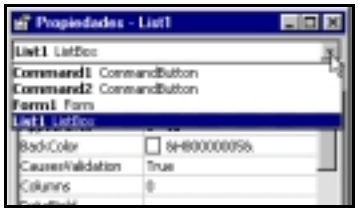
2.6.1. "Nombre" propietatea

Visual Basic-en objektu guztiek izenburua behar dute, hain zuzen, *Nombre* propietatearen bitartez definitzen dena. Formularioaren *Nombre* propietatea begiratzen baduzu bere edukia **Form1** dela ikusiko duzu. Hau da Visual Basic-ek era automatikoan formularioaren *Nombre* propietateari ematen dion balio lehenetsia, baina oso adierazkorra ez denez, hobe da aldatzea erabilgarriagoa izan dadin.

Formulario baten *Nombre* propietatea aldatzeko egizu ondoko hau:

- Egiazta ezazu formularioa aukeratua dagoela.
- Egizu klik *Propiedades* leihoko *Alfabética* etiketan.
- Zerrendako lehenengo propietatea *(Nombre)* da. Parentesi artean dago zerrendaren lehen tokian agertzeko. Egizu klik lehenengo gelaxkan eta idatz ezazu **frmKaixo**.

Nombre propietatea aldatu berria duzu. Lehenengo hiru karaktereek objektua zein kontrol mota den adierazten dute, bere identifikazioa errazteko. Hau ez da beharrezkoa, baina lana errazten du eta kodea ulergarriagoa bilakatzen du.



2.10 irudia

Objektu ezberdinen propietateak ikusteko beste era bat (objektua formularioan aukeratu beharrean) *Propiedades* leihoko goiko aldean dagoen zerrendan aukeratzea da. *Propiedades* leihoak goiko laukian agertzen den objektuaren propietate zerrenda erakusten du. Beste objektu baten propietateak ikusteko, saka ezazu beheraka doan gezia ikonoa eta hauta ezazu nahi duzun objektua.

Formularioan dugun komando-botoia programatik ateratzeko erabiliko dugu. Beraz bere izena adierazgarriagoa izan dadin, bere *Nombre* propietatearen balioa aldatuko dugu:

- Hauta ezazu *Nombre* propietatea, eta emaiozu **cmdIrten** balioa.

Irten botoian "Command1" irakur dezakegu, hori baita bere testu lehenetsia. Testua aldatzeko egizu ondoko hau:

- Hauta ezazu propietateen zerrendan *Caption* propietatea, eta bere testu lehenetsiaren ordez idatz ezazu **&Irten**.

&Irten testuan I-ren aurrean dagoen & karaktereari esker, anpersand izenekoa, I hizkia azpimarratua azalduko da botoiaren testuan. Programa egikaritzean, teklatuko I tekla Alt teklarekin

batera sakatzeak saguaren ezkerreko botoiaz Irten botoiaren gainean klik eginez lortuko dugun ondorio bera izango du.

Arestian formularioaren kasuan aipatu dugun bezala, objektuen izenek hiru karaktereen aurrizkia izango dute. Era honetan zein motako objektuarekin ari garen lanean jakingo dugu. Ariketa honetako formularioaren izena **frmKaixo** da, eta komando botoiarena **cmdIrten**.

2.1 taulan objektu ezberdinen aurrizkien laburpena eskaintzen da.

Aurrizkia	Objektu mota	Adibidea
chk	Ckeck box	chkIrakurtzekoSoilik
cbo	Combo box	cboEuskara
cmd	Command button	cmdIrten
dlg	Common dialog	dlgFitxategiaIreki
frm	Form	frmTransferitu
fra	Frame	fraHizkuntza
gra	Graph	graTenperatura
grd	Grid	grdPrezioak
hsb	Horizontal scroll bar	hsbAbiadura
img	Image	imgIkonoa
lbl	Label	lblLaguntzaMezua
lin	Line	linBertikala
lst	List box	lstSentsorea
mnu	Menu	mnuFitxategiaItxi
pic	Picture	picVGA
shp	Shape	shpKarratua
txt	Text box	txtMotorea
tmr	Timer	tmrPeriodoa
upd	UpDown	updNoranzkoa
vsb	Vertical scroll bar	vsbPotentzia
sld	Slider	sldEskala
tlb	Toolbar	tlbEkintzak
sta	StatusBar	staOrduaEguna

2.1 taula

Hemendik aurrera objektu baten aipamena egiteko *Nombre* propietatea erabiliko dugu.

2.6.2. Irten botoiaren Font propietatearen aldaketa

Edozein objektuan hasieran ikusten den testuak letra-tipo bera du. Batzuetan letra-tipoa aldatu nahi izango dugu formularioaren itxura gure gustura egokitzeko. Horretarako *Font* propietatearen edukia aldatuko dugu.

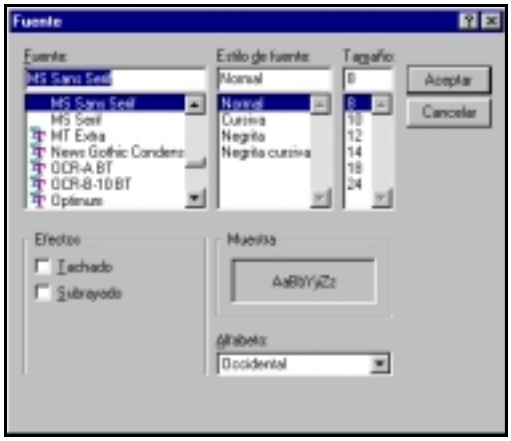
Irten botoiko testuaren letra-tipoa aldatzeko ondoko urratsak jarraituko ditugu:

- Hauta ezazu **cmdIrten** botoia, eta ondoren *Propiedades* leihoan *Font* propietatea.

2.11 irudia:
sortu berriak diren objektuen letra-tipo lehenetsia MS Sans Serif da. *Propiedades* leihoan nahi dituzun aldaketak egin ditzakezu.

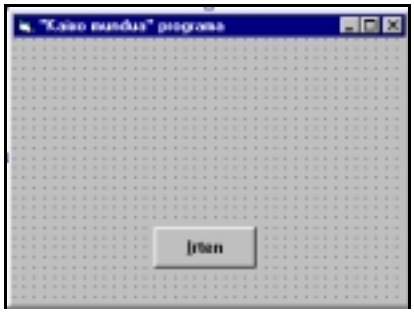
Oraingoz, letra-tipoa MS Sans Serif da, baina Arialekin ordezkatuko dugu:

- Egizu klik hiru puntu dituen ikonoaren gainean (eten-puntuak), *Font* hitzaren eskuin aldean.
- Hauta ezazu Arial letra-tipoa eta 10 puntuko tamaina. Ondoren, egizu klik *Aceptar* botoian.



2.12 irudia:
Fuente elkarrizketa-koadroa**.**

Egindakoaren ondorioz, cmdIrten botoiaren testuaren itxura aldatu da.



2.13 irudia:
komando-botoiaren letra-tipoa aldatu da.

Orain botoi gehiago jarriko dugu formularioan:

- Arestian egindako modu berean, egizu klik-bikoitza tresnen koadroko *CommandButton* ikonoaren gainean.
- Eraman ezazu botoi sortu berria formularioaren ezker aldera.

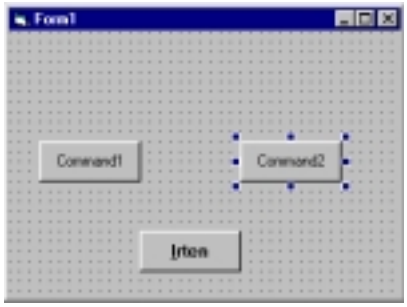
Orain beste botoi bat jarriko dugu, baina beste era batean egingo dugu.

- Egizu klik tresnen koadroko *CommandButton* ikonoaren gainean, eta mugi ezazu kurtsorea formularioraino.
- Mugi ezazu kurtsorea botoiaren erpin bat kokatu nahi duzun tokiraino.
- Saka ezazu saguaren ezkerreko botoia, eta, botoia askatu gabe, arrasta ezazu kurtsorea botoiaren aurkako erpina kokatu nahi duzun tokiraino. Orduan, aska ezazu saguaren botoia.

2.14 irudia:
formularioak botoi berri bat du.

Orain botoiaren tamaina aldatuko dugu:

- Egizu klik *Command1* komando botoiaren gainean. Ondo egin baduzu, botoiaren inguruan helduleku urdinak agertuko dira.
- Jar ezazu kurtsorea beheko erdiko heldulekuaren gainean. Kurtsorearen itxura bi puntadun gezi bilakatuko da.
- Saka ezazu saguaren ezkerreko botoia, eta, askatu gabe, arrasta ezazu beherantz botoia handitzeko.
- Jar ezazu *Command2* komando-botoia formularioan aurreko prozedura bera jarraituz.



2.15 Irudia:
formularioan botoi berria jarri duzu, eta tamaina aldatu diozu.

2.6.3. Botoi berrien propietateen aldaketa

cmdIrten komandoarekin egin dugun moduan, egin berri ditugun botoiei propietate batzuk aldatuko dizkiegu, besteak beste Nombre eta *Caption* propietateak.

- Hauta ezazu *Command1* botoia.
- Emaiozu *Nombre* propietateari **cmdKaixo** balioa.
- Emaiozu *Caption* propietateari **&Kaixo Mundua** balioa.
- Alda ezazu letra-tipoa: formularioko objektu guztietako testuak Arial letra-tipoa 10 puntutako tamainan ager daitezen.
- Egizu gauza bera *Command2* botoiarekin: *Nombre* **cmdEzabatu** eta *Caption* &**Ezabatu** izango dira.

2.16 irudia:
egindako aldaketen ondorioz formularioaren oraingo itxura hau izango da

cmdKaixo botoiko testua bi lerrotan ageri da. Lerro bakar batean nahi baduzu, jarrai itzazu ondoko urrats hauek:

- Hauta ezazu *cmdKaixo* botoia.
- Arrasta ezazu eskuineko erdiko heldulekua botoia luzatzeko.

Bi botoiek (edo hirurek) tamaina berdina edukitzea nahi baduzu, *Formato* menuak eskaintzen dizun aukera erabil dezakezu:

- Hauta ezazu tamaina berdindu nahi dituzun botoi guztiak. Horretarako, klik egin botoien gainean <**Shift**> tekla sakatuta mantentzen duzun bitartean.
- Hauta ezazu *Formato* menuko *Igualar tamaño* ⇒ *Ambos*. Honen ondorioz, botoien tamaina berdina izango da.

Botoiak horizontalki lerrotuta nahi badituzu, hauta itzazu nahi dituzun botoiak eta egizu *Formato* ⇒ *Alinear* ⇒ *Inferior*.

Esperimenta ezazu *Formato* menuak eskaintzen dizkizun aukera ezberdinekin.

2.6.4. Testu-laukia kontrola (TextBox)

Orain objektu mota berri bat jarriko dugu formularioa, testu-laukia hain zuzen. Testu-laukia formularioan kokatu ahal izango dugun koadro bat da, eta, bai programan kodea sartzeko, bai programaren eragiketen ondoriozko emaitzak erakusteko erabiltzen da. *TextBox* elementua *ab* hizkiak bere gainean dituen tresnen koadroko ikonoa da (). Kurtsorea bere gainean jartzen baduzu *TextBox* testua laukizuzen hori baten barruan agertuko da.

- Egizu klik *TextBox* ikonoaren gainean, eta eraman ezazu kurtsorea formularioraino.
- Mugi ezazu kurtsorea *TextBox* objektuaren erpin bat kokatu nahi duzun tokiraino, eta, arrasta ezazu kurtsorea botoiaren aurkako erpina kokatu nahi duzun tokiraino.

Saguaren botoia askatzean, testu-laukia eta bere eduki lehenetsia ikusi ahal izango dituzu.

2.17 irudia :
testu-laukiko konfigurazio lehenetsia.

Testu-laukiaren propietate batzuk aldatuko ditugu:

- Egiazta ezazu testu-laukia hautatua dagoela.
- Alda ezazu *Nombre* propietatea. Eduki berria: **txtKaixo**.
- Ezaba ezazu *Text* propietatearen edukia (bere balio lehenetsia Text1 da), programa egikaritzean hutsik ager dadin.
- *Alignment* (lerrokatzea) propietatearen eduki lehenetsia *0-left Justify* da (ezkerrean lerrokatu), hau da, testua testu-laukiko ezker aldean lerrokatuko da programa egikaritzean. Egiten ari garen programan, testua testu-laukiko erdian lerrokatua nahi dugu, beraz, propietate honen balioa aldatu beharko dugu. *2-Center* balio berria emateko, beherakako gezia sakatzean agertzen den zerrenda erabiliko dugu.

Egin dezakezun beste gauza bat *Multiline* propietateari *True* balioa ematea da. Hau egiten ez baduzu, testuak lerro baten luzera gainditzen duen kasuetan Visual Basic-ek ez du kontuan hartuko *Alignement* propietatea.

- Alda ezazu *Font* propietatearen edukia: Arial 10.

2.7. Programaren egikaritzea

Orain arte egindakoa nola doan ikusteko programa egikari dezakezu.

- Gorde ezazu proiektua *Archivo* menuko *Guardar proyecto* aukeraren bitartez (edo tresna-barrako *Guardar proyecto* ikonoan klik eginez).
- Hauta ezazu *Ejecutar* menuko *Iniciar* (<**F5**> funtzio-tekla sakatuz edo tresna-barrako *Iniciar* botoia sakatuz ere egin daiteke).

Ohartuko zarenez, alferrikakoa dela formularioko botoiak sakatzea. Honen arrazoia botoiei kodea gehitu ez izana da.

- Aplikaziotik ateratzeko, saka ezazu leihoaren goiko aldeko eskuinean dagoen ☒ botoia.

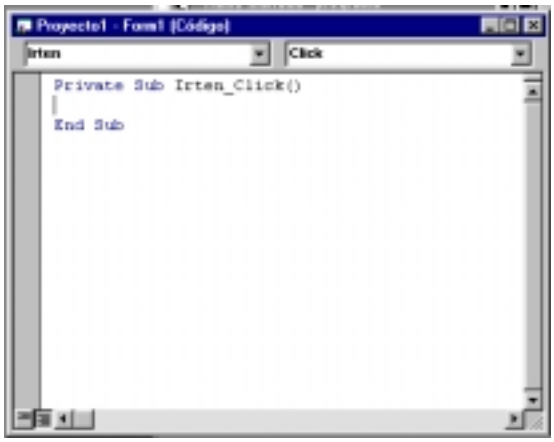
2.8. Objektuei kodea gehitu

Visual Basic gertaerei zuzendutako hizkuntza da. Gertaera bat antzematen duenean, proiektuak gertaera prozedura egokira jotzen du. Gertaera prozedurak ordenagailuari gertaera bati erantzunez zer egin behar duen adierazteko erabiltzen dira.

Gure programan gertaera bat cmdIrten sakatzea izan daiteke. Oraingoz, botoi hori sakatzen dugunean gertaera bat jazotzen da, baina gertaera-prozedurarik lotu ez diogunez, ez dago ondoriorik.

Orain gertaerari kodea lotuko diogu:

- Egizu klik-bikoitza cmdIrten botoiaren gainean. Ondorioz, kode leihoa irekiko da, eta bertan, zure *Sub* prozedurarako eredu bat izango duzu. Eredu honetan zure *Sub* prozeduraren lehen eta azken lerroak aurkituko dituzu.



2.18 irudia
kode leihoan prozeduraren lehen eta azken lerroak ikus daitezke

2.18 irudian ikus daitekeen moduan, goiko ezker aldeko zerrendak (objektuen zerrenda) objektuaren izena erakusten du, eta goiko eskuin aldeko zerrendak “Click“ gertaera (prozeduren zerrenda).

- Saka ezazu behin teklatuko tabuladorea, eta idatz ezazu ondoko instrukzioa:

```
End
```

Código leihoan ondoko hau ikusi ahal izango da:

```
Private Sub cdmIrten_Click()
    End
End Sub
```

Prozedurako instrukzioak koskatzeak bere irakurketa eta interpretazioa errazten ditu.

- Gorde eta egikari ezazu zure proiektua. Tresna-barran bideoko *Play* teklaren itxura duen botoia sakatuz egin dezakezu.
- Saka ezazu Irten botoia (edo <**Alt + I**> teklak batera) eta proiektuaren egikaritzea amaituko da.

Orain, gainerako botoiei gertaera-prozedurak lotuko dizkiegu.

2.8.1. cmdKaixo botoiari kodea gehitu

Hasteko, *cmdKaixo* botoiari gehituko diogu kodea:

- Joan zaitez berriro *Objeto* leihora. Bi eratan egin dezakezu: *Ver* menuko *Objeto* aukeraren bidez, edo *Propiedades* leihoko goiko erdiko ikonoa sakatuz.
- Egizu klik-bikoitza *cmdKaixo* botoian. Berriro agertuko da *Código* leihoa, oraingo honetan *cmdKaixo_Click* Sub prozeduraren ereduarekin.
- Idatz ezazu ondoko hau prozeduraren lehen eta azken lerroen artean:

```
txtKaixo.Text = “Kaixo Mundua”
```

Orain ikusi duzun moduan, txtKaixo ondorengo puntua idatzi bezain laster aukera zerrenda bat irekitzen da. Zerrendakoak dira lanean ari garen objektuarekin erabil ditzakegun aukera guztiak eta bakarrak, kasu honetan testu-laukikoak. Haien artean aukera egiteko hiru bide ezberdin dago: gora eta beherakako geziez aukera eginez eta bukatzeko zuriune-barra sakatuz; saguaz gora eta behera mugitu ondoren nahi den elementuaren gainean klik eginez; edo zeuk hitza idatziz.

Instrukzio honek **Kaixo Mundua** balioa esleitzen dio txtKaixo objektuaren Text propietateari. Testu-katea denez, derrigorrezkoa da komatxoen artean idaztea.

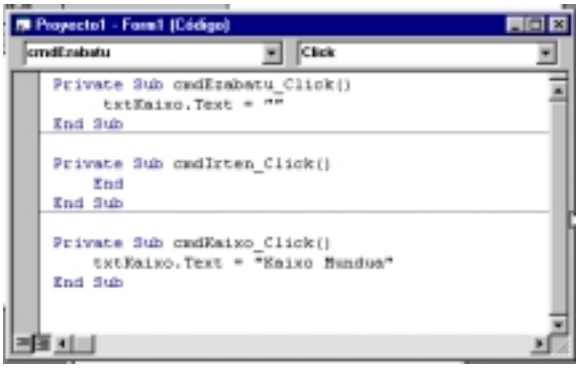
2.8.2. cmdEzabatu botoiari kodea gehitu

Orain **cmdEzabatu** botoiari kodea gehituko diogu. Prozedura aurrekoan erabilitako bera da:

- Joan zaitez berriro *Objeto* leihora.
- Egizu klik-bikoitza *cmdEzabatu* botoian. Berriro agertuko da *Código* leihoa, oraingo honetan *cmdEzabatu_Click* Sub prozeduraren ereduarekin.
- Idatz ezazu ondoko hau prozeduraren lehen eta azken lerroen artean:

```
txtKaixo.Text = ""
```

Instrukzio honek balio nulua esleitzen dio *txtKaixo* objektuaren *Text* propietateari. Honela testu-laukiko edukia ezabatuko du. *Código* leihoak 2.19 irudiko itxura izango du.



2.19 irudia:
Código leihoaren oraingo bista

2.8.3. Proiektuaren egikaritzea

Azkenik Kaixo programa osorik dago eta azken emaitza ikusiko dugu.

- Gorde ezazu zure lana.
- Egikari ezazu proiektua.

2.20 irudia:
zure proiektuak egikaritzean proba itzazu botoi guztiak, era egokian erantzuten duten egiaztatzeko

- Saka ezazu *Kaixo Mundua* botoia, eta "Kaixo Mundua" hitzak testu-laukian agertuko dira.
- Saka ezazu *Ezabatu botoia*, eta testu-laukiko testua ezabatuko da.

Aurrekoa beste era batean egin dezakegu, hain zuzen ere, <**Alt + K**> eta <**Alt + E**> tekla konbinazioak erabiliz.

- Saka ezazu *Irten* botoia programari amaiera emateko (edo <**Alt + I**> tekla konbinazioa).

2.9. Fitxategi egikarigarriak

Orain proiektua dagoen moduan Visual Basic beharrezkoa da egikaritu ahal izateko. Visual Basic-etik kanpo egikaritzeko beste edozein aplikazioren eran ondoko hau egin beharko duzu:

- Hauta ezazu *Archivo* menuan *Generar Kaixo.exe*.
- Jarraian agertuko den elkarrizketa-koadroan fitxategi egikarigarriaren izen lehenetsia Kaixo.exe izango da. Nahi baduzu alda dezakezu.
- Fitxategi egikarigarria gordeko den direktorioa elkarrizketa-koadroaren goiko aldean ikus daiteke. Kapitulu honen hasieran sortutako bera izan beharko luke (kap02). Hala bada, saka ezazu *Aceptar* botoia.

Hemendik aurrera, Kaixo aplikazioa beste edozein aplikazioren moduan egikaritu ahal izango duzu.

2.10. Metodologia

Orain arte proiektuaren kodea objektuz objektu aurkeztu da. Hemendik aurrera, arazoa proposatu ondoren, soluziobide bat aurkeztuko da eta horretarako sortu beharreko objektuak taula baten bitartez emango dira. Ez dira objektuen propietate guztiak aldatu beharko, banakako batzuk baizik, eta horiek bakarrik agertuko dira taulan.

Ondorengo taulan **Kaixo mundua** proiektuaren objektuak eta aldatu behar diren beren propietateak aurkezten dira.

Kontrol mota	**Propietatea**	**Balioa**
Form	Nombre Caption	frmKaixo “Kaixo Mundua” programa
Command Button	Nombre Caption Font	cmdIrten &Irten Arial, Bold, 10
Command Button	Nombre Caption Font	cmdKaixo &Kaixo Mundua Arial, Bold, 10
Command Button	Nombre Caption Font	cmdEzabatu &Ezabatu Arial, Bold, 10
Text Box	Nombre Text Alignment Multiline Caption Font	txtKaixo (hutsik utzi) 2 - Center True (hutsik utzi) Arial, Bold, 10

2.2 taula

Oharra: balioen zutabean parentesi artean agertzen dena instrukzio bat da. Beraz, aurreko taulan balioen zutabean “(hutsik utzi)” agertzen denean propietatearen balio lehenetsia ezabatu behar dela esan nahi du.

3 Sistemaren diagnostikoa

Kapitulu honetan LEGO MindStorms-ekin lehenengo kontaktua izango dugu. Kapitulu honetan ez dugu robotik muntatuko, baina aurrerago baliagarria izango den programa baten bitartez RCXren programazioari ekingo diogu. Programa hau, nahiz eta oso sinplea izan, oso erabilgarria izango da arazoren bat sortzen denean.

3.1. Kapitulu honetako edukiak

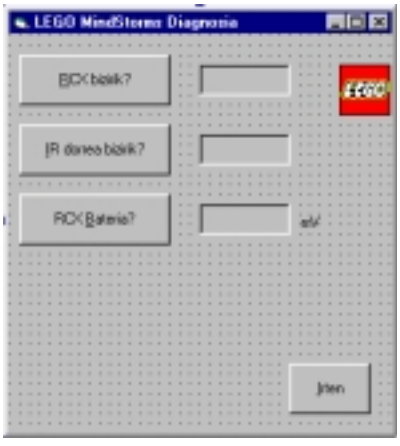
1. Tresnen koadroko kontrolak sartzen jarraitu: Label kontrola.
2. Visual Basic-eko aldagai eta konstanteen erabilpena: sarrera.
3. Kontrol egiturak: sarrera. Visual Basic-eko If ... Else ... Then egitura.
4. Spirit.ocx kontrolarekin lanean hasi: egiaztapenetarako metodoak.

3.2. Proiektu proposamena

Praktika honetako lehenengo partean, RCXren egoera egiaztatzeko programa bat diseinatuko dugu. Ondokoa egiteko gai izango da:

- Infragorrien dorrea konektaturik dagoen egiaztatuko du.
- Infragorrien dorrea RCXrekin komunikatzen den egiaztatuko du.
- RCXren bateria maila neurtuko du.

Ez da ahaztu behar infragorrien dorreak (hemendik aurrera IR dorrea) 9V-eko pila bat erabiltzen duela, eta hau izan daitekeela batzuetan IR dorrea eta RCXren artean gertatzen den komunikazio faltaren arrazoia[2].



3.1 irudia
amaieran formularioak honelako itxura izango du.

[2] RIS 1.0 eta 1.5i dagokien infragorrien dorrea serie atakan konektatzen da, eta, 9V-eko pila bat erabiltzen du. Oraindik merkatuan ez badago ere, RIS 2.0 bertsioarena berriz USB atakan konektatuko omen da.

3.3. Aldagaiak Visual Basic-en

Programa honek eskatzen dituen egiaztapenak egiteko RCX txekeatuko dugu, eta RCXtik jasotzen dugun informazioa formulario baten bitartez ikusi ahal izango da. Informazio hori gordetzeko aldagaiak erabiliko ditugu. Aldagai izena dute beren edukia aldatu daitekeelako. Beharbada matematiketan erabiltzen diren aldagaiak ezagutuko dituzu. Hona hemen adibide bat:

$$x + y = 6$$

Espresio honetan bi aldagai dago: x eta y.

Baina honez gain, Visual Basic-en aldagaiek informazio ez-matematikoa gorde dezakete, adibidez testu-kateak. Aurreko kapituluko kodean ondoko espresioak agertu dira:

```
txtKaixo.Text=""
```

eta

```
txtKaixo.Text="Kaixo Mundua"
```

Bi lerro horietan egin duguna *txtKaixo*.*Text* propietateari "" balioa esleitzea izan da, eta ondoren, aldatu dugu "Kaixo mundua" balioa emanez. *Text* propietatea aldagai baten adibidea da, eta txtKaixo aurrizkiak aldagai hau txtKaixo objektuarena dela adierazten du. Berez, objektu baten propietate guztiak alda daitezkeenez, aldagaiak dira. Gure aldagaiak defini ditzakegu gure programetan erabiltzeko.

Aldagai mota ezberdin ugari dago. Hala ere, esku liburu honetan bi aldagai mota erabiliko dugu soilik: testu-kateak (string) eta zenbakizkoak.

Matematiketan zenbaki mota ezberdinak daudela ikasten da: naturalak, osoak, errealak... Hemen ere zenbakiak gordetzen dituzten aldagai guztiak ez dira berdinak izango: integer (zenbaki osoak: 7, -345, 23...); koma higikorra (2.345, -1.2, 5.00); zenbaki errealak (2, 2 ½, π)... Aldagaia zein motakoa den era azkar batean jakiteko aldagai izenek aurrizki bat izango dute. 3.1 taulan Visual Basic-en erabil daitezkeen aldagai motak zerrendatzen dira.

Datu mota	Aurrizkia	Adibidea
Boolean	bln	blnIndarrean
Byte	byt	bytMezuZbk
Collection object	col	colPolig
Currency	cur	curSarrerak
Date (Time)	dtm	dtmHasiera
Double	dbl	dblDoitasuna
Error	err	errZenb
Integer	int	intKopurua
Long	lng	lngDistantzia
Object	obj	objOraingoa
Single	sng	sngBatazbeste
String	str	strIzena

Datu mota	Aurrizkia	Adibidea
User-defined type	udt	udtEnpleatua
Variant	vnt	vntCheckBatuketa

3.1 taula

3.4. Konstanteak

Hainbat kasuetan aldatzen ez diren balioak erabiltzen dira, adibidez π eta e zenbakiak. Horretarako konstanteak izenekoak ditugu. Konstanteak asko erabiltzen dira matematika eta programazioan. Lego RCXren programazioa aurrez definitutako konstanteak erabiliz erraz daiteke (A_MOTOREA, TENP_2...). Hurrengo kapituluan ikusiko dugu nola egiten den.

3.5. Etiketa-kontrola (label)

Etiketa-kontrola kontrol grafiko bat da. Honi esker aurkeztu ahal izango dugun testua ezin izango du erabiltzaileak era zuzenean aldatu. Hala ere, diseinu fasean etiketa-kontrolaren edukia alda dezakeen kodea idatz daiteke.

Programa berri bat idazteko proiektu berri bat sortu beharko dugu:

- Abiarazi ezazu Visual Basic. *Nuevo Proyecto* leihoa agertzen bada, egizu klik-bikoitza *EXE estándar* ikonoaren gainean. Agertzen ez bada, hauta ezazu *Archivo* menuko *Nuevo Proyecto*.
- Hauta ezazu *Proyecto* menuko *Componentes* (hau bera, kurtsorea tresnen koadroaren gainean dagoenean eskuineko botoia sakatuz, eta ondoren, sortzen den menu berezian *Componentes* hautatuz egin daiteke).
- Bila ezazu “LEGO PbrickControl, OLE Control module” eta marka ezazu bere ondoko laukitxoa. Saka ezazu *Aceptar* botoia, eta honela, tresnen koadroan LEGO osagaia agertuko da.
- Egizu klik-bikoitza LEGO ikonoaren gainean, eta honela LEGO kontrola formularioan agertuko da. Kontrol honen izen lehenetsia spirit1 da, baina nahi izanez gero propietateen leihoan alda daiteke.
- Gorde ezazu proiektua C:\VBLego\Kap03 direktorioan. Horretarako hauta ezazu *Archivo* menuko *Guardar proyecto*. Behin direktorioa sortuta, gorde itzazu proiektua eta formularioa **Diagnostikoa** izenarekin.
- Diseina ezazu frmDiagnostikoak 3.2 taulako datuetan oinarrituz.

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	frmDiagnostikoak LEGO MindStorms-en Diagnostikoak
CommandButton	Nombre Caption ToolTipText	cmdRCXBizirik &RCX: Bizirik? RCXren egoera egiaztatu
CommandButton	Nombre Caption ToolTipText	cmdDorreAktiboa IR &Dorrea ongi? IR dorrea egiaztatu
CommandButton	Nombre Caption ToolTipText	cmdBateria RCXren &Bateria Bateriaren tentsioa
CommandButton	Nombre Caption	cmdIrten &Irten
Label	Nombre Alignement BorderStyle Caption	lblRCXBizirik 2 – Center 1 – Fixed Single (utzi hutsik)
Label	Nombre Alignement BorderStyle Caption	lblDorreaOngi 2 – Center 1 – Fixed Single (utzi hutsik)
Label	Nombre Alignement BorderStyle Caption	lblBateria 2 – Center 1 – Fixed Single (utzi hutsik)

3.2 taula

Formularioa bukatzen duzunean, 3.1 irudiko itxura eduki beharko luke.

- Idatz ezazu cmdIrten_Clic() prozedurari dagokion kodea (prozedura hau idazteko egizu klik-bikoitza *Objeto* bistako *Irten* botoiaren gainean). Idatz ezazu prozedura honen aurrean Option Explicit lerroa.

```
 'Aldagai guztien deklarazioa beharrezkoa da

Option Explicit

Private Sub kmdIrten_Click()
    PBrickCtrl.CloseComm ' komunikazio ataka ixten du
    End
End Sub
```

- Ondoren, idatz ezazu Form_Load() prozedurari dagokion kodea (prozedura hau idazteko egizu klik-bikoitza formularioan kontrolik ez duen edozein tokitan).

```
Private Sub Form_Load()
    PBrickCtrl.InitComm ' komunikazio ataka irekitzen du
End Sub
```

Azter dezagun kodea urratsez urrats.

Azalpenak: kodearen lehenengo lerroa azalpen bat da. Azalpen bat apostrofo bat (') aurrean duen edozein testu lerro da. Apostrofoaren ondoren nahi den guztia idatz daiteke, testu hori

ez baita kontuan hartuko programaren konpilazioan. Visual Basic-eko editorean berde kolorez agertuko den testu honi esker programa ulergarriagoa izango da, batez ere, hirugarren pertsonek irakurtzen dutenean.

Option Explicit: instrukzio hau jartzen badugu, erabili nahi diren aldagai guztiak aurrez deklaratu behar dira. Hau oso mesedegarria da, batez ere aldagai baten izena idaztean akats bat egiten badugu Visual Basic-ek ez baitu aldagai berritzat hartuko, baizik eta idazketa akatsatzat.

Komunikazio ataka ireki: Visual Basic-ek RCXrekin komunikatu ahal izateko, ordenagailuko komunikazioetarako serie ataka hasieratu behar da. Hau *PBrickCtrl.InitComm* aginduaren bitartez egiten da. Lana errazteko programaren abiatzearekin batera egitea komeni da. Horretarako, agindua *Private Sub Form_Load* gertaera-prozeduran jarri behar da. Prozedura honi dagokion eredua lortzeko, formularioan kontrolik ez duen edozein tokitan klik-bikoitza egin behar da.

Komunikazio ataka itxi: Behin RCXrekin egin beharreko komunikazioak bukatuta, komunikazio ataka itxi behar da, badaezpada beste aplikazioren batek behar duen. Honetarako *PBrickCtrl.CloseComm* instrukzioa erabiliko dugu. Ez zaigu interesatzen RCXrekin komunikazioak bukatu aurretik instrukzio hau egikaritzea. Beraz, egokiena cmdIrten_Click() prozedurari lotzea da, programaren amaieran egikari dadin.

Jarrai dezagun orain proiektuarekin:

- Gorde ezazu proiektua *Archivo* menuko *Guardar proyecto* aukeratuz..
- Egikari ezazu programa tresna-barrako *Iniciar* (Play) botoia sakatuz.
- Saka ezazu *Irten* botoia eta programa amaituko da.

Programak, formularioa kargatzen denean, *InitComm* prozedura deitzen du, eta, *Irten* botoia sakatzean *CloseComm* prozedura. Orain bi sistema agindu hauen artean IR dorrea eta RCXren artean interakzioa abiaraziko duen kodea idatziko dugu.

3.6. Erabakiak hartzea

Erabakiak hartzeko instrukzioek kodeak eskaintzen dituen aukeren artean hautatzeko bidea irekitzen du, programa egikaritzean sortzen diren egoerei era egokian erantzuteko. Erabakiak hartzeko aukera zabaltzeko If … Then ... Else kontrol-egitura erabiltzen da.

3.6.1. If ... Then ... Else kontrol-egitura

Kontrol-egitura honek hiru zati ditu:

- **If**-ek erabakiaren oinarria izango den baldintzari sarrera ematen dio.
- Baldintza betetzen denean egin behar dena **Then**-en ondoren dator.
- **Else**-k baldintza betetzen ez denean egin beharrekoa zehazten du (aukerakoa da).

Idatz dezagun orain RCXtik hainbat informazio eskuratzeko kodea:

- Idatz ezazu programaren gainerako kodea cmdBateria_Click() prozeduratik hasita:

```
Private Sub cmdBateria_Click()
    lblBateria.Caption=Str(PBrickCtrl.PBBattery)
End Sub
```

- Orain beste hau idatzi::

```
Private Sub cmdRCXBizirik_Click()
    If PBrickCtrl.PBAliveOrNot Then
```

```
            lblRCXBizirik.Caption=”Bai”
      Else
            lblRCXBizirik.Caption=”Ez”
      End If
End Sub
```

- Eta bukatzeko beste hau:

```
Private Sub cmdDorreaAktiboa_Click()
    If PBrickCtrl.TowerAlive Then
            lblDorreaOngi.Caption=”Bai”
      Else
            lblDorreaOngi.Caption=”Ez”
      End If
End Sub
```

cmdBateria_Click(): prozedura honen bitartez RCXren pilek ematen duten tentsioa jakin dezakegu. Kodean, hasteko bateriaren uneko tentsioa irakurtzen da (milivoltetan), ondoren, zenbaki bat den balio hau testu-kate bihurtzen da *Str*() funtzioa erabiliz, eta bukatzeko, *lblBateria* etiketaren *Caption* propietateari testu kate hori esleitzen dio.

cmdRCXBizirik_Click(): prozedura honetan Visual Basic-eko If ... Then ... Else kontrol egitura erabili dugu. Honela, RCXk ematen digun erantzunaren arabera erabaki egokia hartu ahal izango dugu (kasu honetan, formularioan zein informazioa erakutsi behar den izango da erabakia). RCX piztuta badago, eta IR dorrea berarekin komunikatzeko gai bada, emaitzaren label-ak “Bai” erakutsiko du. Hala ez bada, “Ez”. Kontrol-egitura honen bukaerak esplizitua izan behar du, eta horretarako *End If* instrukzioa erabiltzen da, subrutinen kasuan *End Sub* erabiltzen den antzera.

cmdDorreAktiboa_Click(): prozedura honek IR dorrea aktibo dagoen egiaztatzen du. Dorrearen hardwarea behar bezala badago, eta bere bateria behar den tentsioa ematen badu, baiezkoa ikusiko da dagokion label kontrolean. Arazoren bat badago “Ez” irakurriko dugu.

Spirit.ocx metodoak: prozedura hauetan hiru metodo berri erabili ditugu. Lehenengoak, *PBrickCtrl.PBBattery*, zenbakizko balio bat itzultzen du, hain zuzen, RCXko baterien tentsioa milivoltetan. Beste biek berriz, balio logiko bat itzultzen dute, hau da, egiazkoa ala faltsua, horregatik egitura-kontrolean zuzenean erabil ditzakegu erabakiak baldintzatzeko.

Jarrai dezagun proiektuarekin:

- Gorde ezazu proiektua.
- Egikari ezazu proiektua.
- RCX piztuta IR dorretik gertu dagoela saka itzazu egiaztapenak egiteko ditugun hiru botoiak.
- Orain, deskonekta ezazu RCX, eta ondoren, saka ezazu *RCX Bizirik?* botoia (RCX itzalita badago, ez ezazu saka “RCX Bateria” botoia, errore bat sortuko baitu).

Baterien tentsioa milivoltetan neurtzen du, eta RCXk bateria berriak dituenean irakurketak 9000mV-en ingurukoa izan beharko luke. Balioa etengabe jaisten da, beraz, behar ez denean itzaltzea gomendatzen da. IR dorrearen irismena egiaztatzeko urrundu pixkanaka-pixkanaka RCX *RCX Bizirik?* botoia sakatzen duzun bitartean.

Bi RCX edo gehiago dugunean, transmisioetan interferentziak sor daitezke. Hau ekiditeko, programaren bitartez RCXren transmisioen potentzia alda dezakegu. Gela batean bat baino gehiago dugunean irismen laburrerako konfiguratu beharko genituzke.

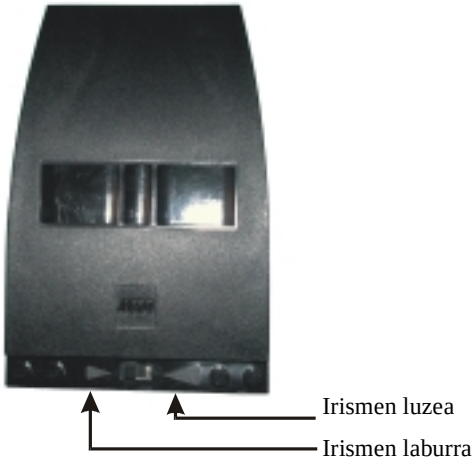
- Erants iezazkiozu 3.3 taulako elementuak formularioari eta ondoren ematen den kodea.

Kontrol mota	Propietatea	Balioa
CommandButton	Nombre Caption ToolTipText	cmdIRLaburra IR &Laburra Komunikazioak irismen laburrean
CommandButton	Nombre Caption ToolTipText	cmdIRLuzea IR L&uzea Komunikazioak irismen luzean
Label	Nombre BorderStyle Caption	lblIrismena 1 – Fixed Single (hutsik utzi)

3.3 taula

```
Private Sub cmdIRLaburra_Click()
      PBrickCtrl.PBTxPower IRISMEN_LABURRA
      LblIrismena= ”RCX irismen laburrean konfiguratua”
End Sub

Private Sub cmdIRLuzea_Click()
      PBrickCtrl.PBTxPower IRISMEN_LUZEA
      LblIrismena = ”RCX irismen luzean konfiguratua”
End Sub
```



3.2 irudia
IR dorrearen konfigurazioa

Hemen RCXren transmisio potentzia aldatzen ari gara. IR dorrearena eskuz alda daiteke dorreak aurreko aldean duen etengailuaz.

- Gorde ezazu proiektua.
- Egikari ezazu programa.
- Saka ezazu irismen laburrerako botoia.

- Jar ezazu RCX dorretik distantzia ezberdinetara (baina ezkutatu gabe), eta sala ezazu *RCX: Bizirik?* botoia distantzia bakoitzean. Honela, irismen laburrean konfiguratua dagoenean irismena norainokoa den jakin ahal izango duzu.
- Saka ezazu irismen luzerako botoia.
- Errepika ezazu aurreko prozedura irismen luzean konfiguratua dagoenean irismena norainokoa den jakiteko.

Programa honetatik atera aurretik, saka ezazu beste programetarako interesatzen zaizun irismenaren botoia.

3.7. Erabilgarritasunean hobekuntzak

Programa honen erabilgarritasuna zabalagoa izan dadin, bi aukera berri gehituko dizkiogu: lehenengoaren bitartez RCXren barneko ordua finkatu ahal izango dugu (ordenagailuko ordu berdina), eta bigarrenak ordenagailutik RCX itzaltzeko aukera emango digu.

- Jar itzazu ondoko kontrolak formularioa:

Kontrol mota	Propietatea	Balioa
CommandButton	Nombre Caption ToolTipText	cmdSetOrdua &Orduan jar RCXn oraingo ordua jartzen du
CommandButton	Nombre Caption ToolTipText	cmdRCXOff Itzali R&CX RCX itzaltzen du

3.4 taula

- Eta jarraian, idatz ezazu kodea:

```
Private Sub cmdRCXOff_Click()
      PBrickCtrl.PBTurnOff
End Sub

Private Sub cmdSetOrdua_Click()
      PBrickCtrl.PBSetWatch Hour(Now),Minute(Now)
End Sub
```

RCX itzaltzeko kodea oso sinplea da. *PBTurnOff* metodoa egikarituz RCX itzaltzen baita.

Bigarren prozedura ez da horren erraza. RCXri ordenagailuko ordu berdina jartzeko eman beharreko lehenengo urratsa ordenagailuko ordua jakitea izango da. Horretarako Visual Basic-eko *Now* funtzioa erabiliko dugu. Funtzio honek sistemaren ordua eta data ematen ditu, baina guk ordu eta minutuen datuak behar ditugu soilik, eta horretarako *Hour* eta *Minute* funtzioak erabiliko ditugu. Honela, SetWatch metodoari oraingo ordu (0 eta 23 artean) eta minutuaren (0 eta 59 artean) balioak pasatzen zaizkio.

3.8. Ariketa

Praktikaren lehen zatian RCXtik informazioa jasotzen dugu. Baina horretarako hiru botoi sakatu behar izan ditugu. Egin itzazu behar diren aldaketak informazio hori guztia botoi bat sakatuz eskuratzeko.

RCX ez badago (itzalita edo urruti dagoenean gertatzen da) bateriaren tentsioaren balioa irakurtzen saiatzen denean errore bat gertatzen da. Idatz ezazu beharrezkoa den kodea errore hau ekiditeko.

4 Zure lehenengo robota

4.1. Kapitulu honetako edukiak

1. Proiektu ezberdinetan errepikatzen diren prozedurak ez errepiklatzeko, Visual Basic-eko txantiloiak nola egiten diren ikasi.
2. Robot mugikorrak: Ordenagailu bidezko kontrola formularioen bitartez.
3. Moduluen erabilpena: RCXdatuak.bas moduluaren erabilpena programak irakurterrazak bilakatzeko.
4. Tresnen koadroa: Desplazamendu-barrak, aukeraketa-botoiak eta frameak.
5. RCXk motoreak kontrolatzeko eskaintzen dituen metodoak.

4.2. Robota

Orain arte ez dugu robot mugikorrik egin. LEGO kit-ak dituen motoreak erabiliz mugitzeko gaitasuna emango diegu gure asmakizunei. Motoreak RCXri konektatzeko konektoreak LEGO adreiluen modukoak dituzten kableak erabiliko ditugu.

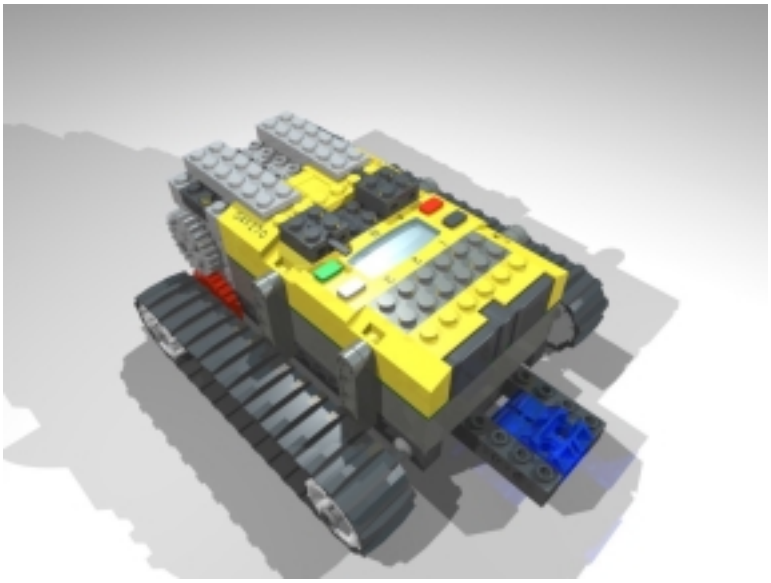
4.1 irudia
Motorea

4.2 irudia
Konexio kablea

RCXk motoreak konektatzeko hiru irteera ditu. Konektore beltz hauen ondoan motoreak izendatzen dituzten A, B eta C hizkiak ikusi daitezke. Konektorea jartzen duzun moduaren arabera motorearen biraketa noranzkoa erlojuarena ala horren kontrakoa izango da.

4.3 irudia
RCXko konexioak: goiko grisak sentsoreak konektatzeko eta beheko beltzak motoreak konektatzeko

Kapitulu honetan erabiliko dugun robota 4.4 irudian ikus daitekeena da. Hau muntatzeko egiteko jarrai itzazu A eranskinean ematen diren instrukzioak.



4.4 irudia
erabiliko dugun robot oinarria

4.3. Txantiloiak

Kapitulu bakoitzean proiektu berri bat ari gara sortzen, eta ikusiko dugun bezala, eman beharreko hainbat urrats behin eta berriro errepikatzen dira, adibidez, spirit.ocx osagaia tresnen koadroan eta formularioan jartzea. Bestelako aplikazio informatiko ugaritan egiten den moduan, hemen ere txantiloiak erabil ditzakegu, hasierako konfigurazio-lana behin eta berriro ez errepikatzeko.

Prestatuko dugun txantiloi honen bitartez bi abantaila izango ditugu:

- Spirit.ocx osagaia hasieratik erabilgarri izango dugu.
- Spirit.ocx-ko kontrolek erabiltzen dituzten argumentuak ulergarriagoak izango dira.

Txantiloi hau ahal den sinpleena izango da, baina erabiltzaile bakoitzak behar duen guztia gehi diezaioke. Adibidez, hasiera eta irteerako prozedurak beti berdinak badira txantiloian sar daitezke lana errazteko.

Txantiloian beharrezkoa izango dugunez, *RCXdatuak* modulua zer den ikusiko dugu.

4.3.1. RCXdatuak modulua

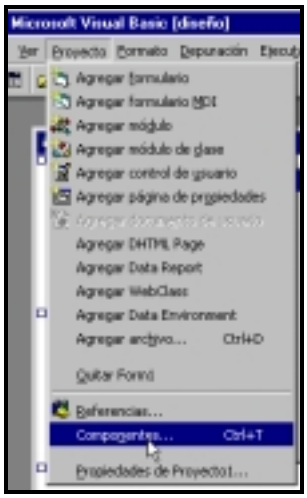
Spirit.ocx-ko kontrolen argumentuak bat ere adierazkorrak ez diren zenbakiak izaten dira. Lana errazteko konstanteak erabiliko ditugu funtzioarekin zerikusia duten izenak emanez. Batzuk komunak izango dira edozein programarako, eta RCXdatuak moduluan gordeko ditugu. Beste batzuk berriz, programa bakoitzean izendatuko ditugu. B eranskinean eskuliburu honekin batera eskaintzen den RCXdatuak.bas modulu fitxategiaren edukia eskaintzen da, hori izango baita hemendik aurrera erabiliko dena.

Hala ere, zure gustutara egoki dezakezu.

4.3.2. Prozedura

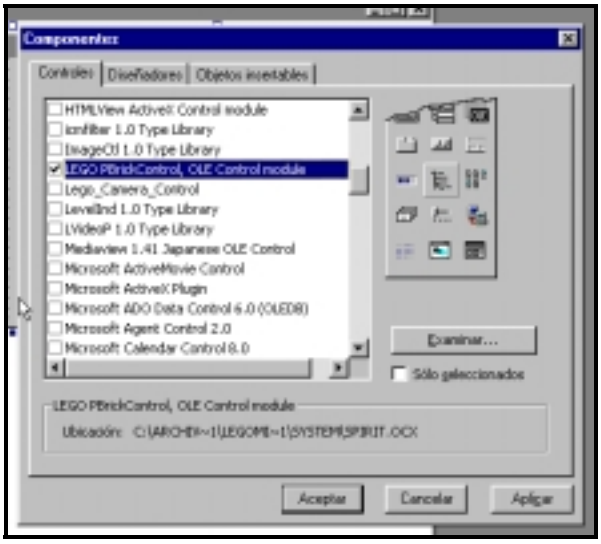
Ikus dezagun orain txantiloia sortzeko prozedura:

- Abiaraz ezazu Visual Basic. *Nuevo Proyecto* leihoa agertzen bada, egizu klik--bikoitza *EXE estándar* ikonoaren gainean. Agertzen ez bada, hauta ezazu *Archivo* menuko *Nuevo Proyecto*.
- Hauta ezazu *Proyecto* menuko *Componentes* (hau bera, kurtsorea tresnen koadroaren gainean dagoenean eskuineko botoia sakatuz, eta ondoren, sortzen den menu berezian *Componentes* hautatuz egin daiteke).



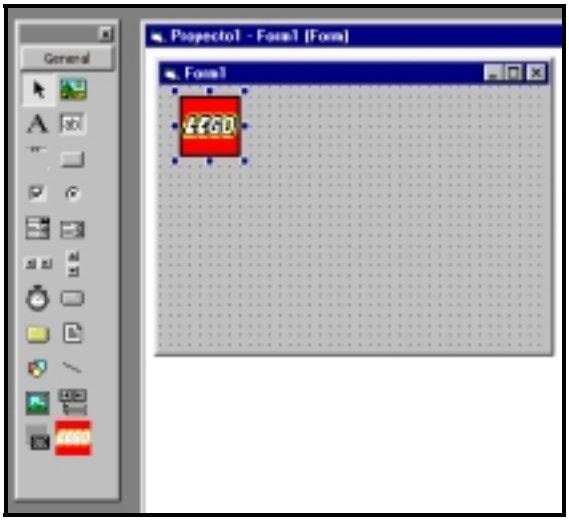
4.5 irudia
Hauta ezazu *Proyecto* menuan *Componentes* (Ctrl+T).

- Bila ezazu “LEGO PbrickControl, OLE Control module” eta marka ezazu bere ondoko laukitxoa. Saka ezazu *Aceptar* botoia, eta honela, tresnen koadroan LEGO osagaia agertuko da. Bertan agertzen ez bada, erabil ezazu *Examinar* osagai hau bilatzeko.



4.6 irudia
Bila ezazu LEGOren ActiveX kontrola osagaien zerrendan.

- Egizu klik-bikoitza LEGO ikonoaren gainean, eta honela LEGO kontrola formularioan agertuko da. Eman iezazkiozu nahi dituzun neurriak eta koka ezazu formularioaren izkina batean.



4.7 irudia
Jar ezazu LEGO kontrola formularioan.

Objektua aukeratzen baduzu, bere propietate batzuk aldatu ahal izango dituzu. LEGOk gomendatzen du spirit.ocx kontrolari **PbrickCtrl** izena ematea. Eskuliburu honetan izen hori bera erabiltzen da, baina zure esku dago beste bat erabiltzea. Objektu honi izena aurreko kapituluan bezala aldatuko diogu:

- LEGO objektua aukeratua dagoela, egin ezazu klik (Nombre) propietatearen gainean, eta, idatz ezazu **PbrickCtrl**.

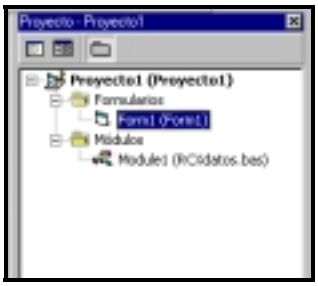
Gainerako propietateen balio lehenetsia egokia da RCX 1.0 eta 1.5-ekin lan egiteko (COM1 komunikazio serie ataka erabiltzen du). CyberMaster erabiltzeko aldaketa batzuk egin beharko dituzu.

Orain RCXdatuak.bas modulua gehituko diogu:

- Hauta ezazu *Proyecto* menuan *Agregar módulo*.

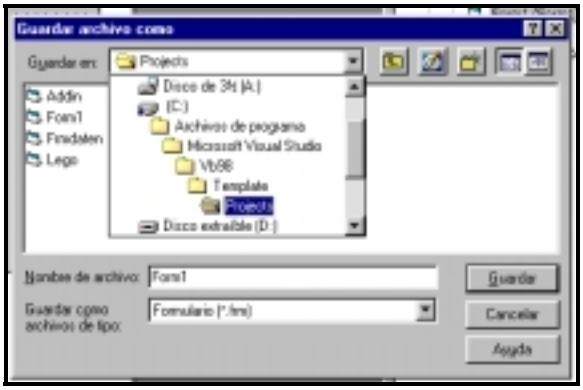
- Egizu klik *Existente*-n, eta bila ezazu RCXdatuak.bas fitxategia (eskuliburu honekin batera eskuratutako fitxategia).
- Aurkitzen duzunean, hauta ezazu, eta ondoren, saka ezazu *Abrir* botoia.

Proiektu leihoa 4.6 irudian ikusten den moduan agertuko da (*Proyecto* leihoan karpetak badituzu, egizu klik beren gainean barruan dutena ikusteko).



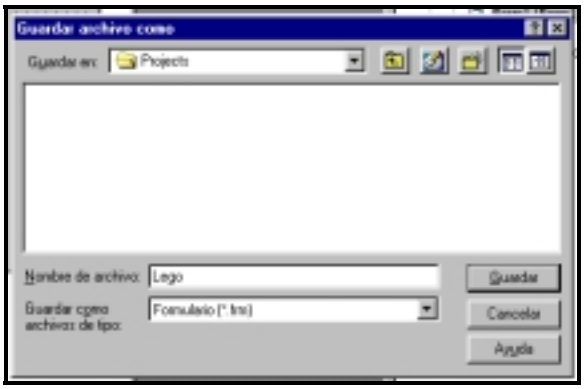
4.8 irudia
Proyecto leihoa

- Hauta ezazu *Proyecto* leihoan Form1.
- Hauta ezazu *Archivo* menuan *Guardar Form1 Como*.
- Bila ezazu C:\Archivos de programa\Microsoft Visual Studio\Vb98\Template\Projects[3] direktorioa.



4.9 irudia
Bila ezazu Visual Basic-eko *Projects* direktorioa

- Eman iezaiozu **Lego** izena formularioari, eta, egin ezazu klik *Guardar* botoian..



4.10 irudia
Gorde ezazu formularioa **Lego** izenarekin.

[3] Visual Basic-eko bertsio edo edizioaren arabera ezberdintasunk egon daitezke *Projects* direktorioaren bidean.

- Hauta ezazu *Archivo* menuan *Guardar proyecto como*, eta idatz ezazu **Lego** testu-laukian.
- Egin ezazu klik *Guardar* botoian.
- Egin ezazu klik *Proyecto* leihoan ikus dezakezun *Modulo1* izeneko objektuan
- Eman iezaiozu *RCXdatuak* izena.
- Saka ezazu *Guardar* botoia.
- Hauta ezazu *Archivo* menuan *Salir* Visual Basic-etik ateratzeko.
- Abiaraz ezazu Visual Basic berriro.

Orain, berriro Visual Basic abiatzean, 4.10 irudiko elkarrizketa-koadroa agertuko da. bertan Lego ikono berria ikusi ahal izango da.

4.11 irudia
Orain, *Nuevo proyecto* elkarrizketa-koadroak Lego txantiloia izango du.

- Elkarrizketa-koadroa agertzen ez bada, hauta ezazu *Archivo* menuan *Nuevo proyecto*.

Orain, zure proiektuak abiarazteko Lego ikonoa erabil dezakezu. Nahi baduzu, aukera gehiago gehi diezazkiokezu txantiloiari, adibidez, programen transferentzian sor daitezkeen erroreak kontrolatzen dituena.

4.4. Lan proposamena

Kapitulu honetan robot mugikor bat ordenagailutik kontrolatzeko programa egingo dugu. Programa hau erabiliz, ibilgailua bide ezberdinetatik gidatu ahal izango dugu, aurrera, atzera, ezkerrera edo eskuinera mugimenduak eginez.

Kapitulu honen bigarren partean kontrol gaitasun handiagoa emango diogu, eta motoreak ematen duen potentzia ere programatik kontrolatu ahal izango dugu.

4.5. Programa

Aurreko kapituluan RCXrekin komunikatzeko Spirit.ocx kontrola nola erabili ikasi duzu. Hemendik aurrera, Spirit.ocx kontrolak eskaintzen dituen gainerako aukerak erabiltzen hasiko gara.

Programa berria sortzeko jarrai itzazu ondoko urratsak:

- Abiaraz ezazu Visual Basic. *Nuevo Proyecto* agertzen bada, egin ezazu klik-bikoitza *Lego* ikonoaren gainean. Agertzen ez bada, hauta ezazu *Archivo* menuan *Nuevo Proyecto*.
- Egiazta ezazu proiektu berriko *Form1* leihoa aukeratua dagoela eta hauta ezazu Archivo menuan *Guardar Form1 Como*.
- *Guardar Como* elkarrizketa-koadroaren bitartez, bila ezazu C:\VBLEGO\ direktorioa.
- Egizu klik *Crear Nueva Carpeta* botoian, eta eman iezaiozu Kap04 izena direktorio berriari.
- Ireki ezazu direktorio berria.
- Emaiozu **Urrutiko kontrola** izena formularioari, eta saka ezazu *Guardar* botoia.
- Hauta ezazu *Archivo* menuan *Guardar proyecto como*.
- Gordeko den lehen fitxategia .bas luzapena duena izango da (modulua). Idatz ezazu **Urrutikoa** izena eta saka ezazu *Guardar* botoia.
- Ondoren, .vbp fitxategiaren izena eskatuko dizu. Emaiozu honi ere **Urrutikoa** izena.
- Diseina ezazu frmUrrutikoa formularioa 4.1 taula datuak erabiliz. Bukatzen duzunean, emaitzak 4.12 irudian ikus daitekeenaren itxura eduki beharko luke.

4.12 irudia
proiektu honen lehenengo urratsa honelako formulario bat egitea izango da.

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	frmUrrutikoa Urrutiko kontrola
CommandButton	Nombre Caption ToolTipText	cmdAurrera &Aurrera Aurrerako mugimendua
CommandButton	Nombre Caption ToolTipText	cmdAtzera A&tzera Atzerako mugimendua
CommandButton	Nombre Caption ToolTipText	cmdEzkerra &Ezkerra Eskerrera biratu
CommandButton	Nombre Caption ToolTipText	cmdEskuina &Eskuina Eskuinera biratu
CommandButton	Nombre Caption ToolTipText	cmdIrten &Irten Programa amaiera

4.1 taula

Orain arte botoiekin erabili dugun gertaera mota bakarra *Click* izan da (hau da, botoi baten gainean klik egitean zerbait gertatzen zen). Proiektu honetan, robota alde batera edo bestera mugitzeko, ez dugu botoien gainean klik egingo. *Aurrera* botoia sakatzean, ibilgailuak aurrera egingo du, eta askatzean, gelditu egingo da. Hau bera gertatuko da gainerako botoiekin.

Click ez diren gertaerak erabiltzeak aldaketa bat dakar edizioan. Aurreko prozedurak editatzeko klik bikoitza egiten genuen komando-botoiaren gainean, eta ondorioz, *Código* leihoan eredu bat agertzen zen prozeduraren lehen eta azken lerroekin. Baina metodo hau ez da erabilgarria *Click* ez diren gertaerekin, eta programa honetan MouseDown (saguaren ezkerreko botoia sakatu) eta MouseUp (askatu) gertaerak inplementatu behar ditugu.

Orain *cmdAurrera*_MouseDown gertaera-prozedurari dagokion kodea idatziko dugu:

- Egin ezazu klik-bikoitza formularioko *cmdAurrera* botoiaren gainean.

Orain, gure aurrean *Código* bista izango dugu, aurreko kapituluan bezala. Leihoaren goiko aldeko ezkerreko zerrendan (objektuen zerrenda) cmdAurrera eta eskuinekoan (prozeduren zerrenda) *Click* irakurri ahal izango da.

- Egin ezazu klik eskuineko koadroko beherakako geziaren gainean, eta hauta ezazu *MouseDown* aukera.

Gertaera honi dagokion eredua sortu da.

- Ezaba ezazu cmdAurrera_Click()-en eredua.
- Orain, idatz ezazu ondorengo kodea sortu berria den prozeduraren ereduan:

```
Private Sub cmdAurrera_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
      PBrickCtrl.SetFwd A_MOTOREA + C_MOTOREA
      PBrickCtrl.On A_MOTOREA + C_MOTOREA 'Aurrera egiten du
End Sub
```

Aurreko kodearen lehenengo lerroaren bukaeran beheko marratxoa erabiltzen da lerroari amaiera emateko. Hala ere, hori ez da kode lerroaren amaiera. Beheko marratxoa karaktereak

jakinarazten dio Visual Basic-i kode lerroa ez dela oraindik amaitu, eta hurrengo lerroan jarraitzen duela. Hau erabilgarria da kode lerro luzeak idatzi behar direnean.

- Orain, aukera ezazu *MouseUp* prozeduren zerrendan eta idatz ezazu ondoko kodea:

```
Private Sub KmdAurrera_MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.Off A_MOTOREA + C_MOTOREA
End Sub
```

- Metodo berdina erabiliz, idatz ezazu ondorengo kode hau:

```
Option Explicit

Private Sub Form_Load()
    PBrickCtrl.InitComm 'Ordenagailuko komunikazio ataka hasieratzen du
    PBrickCtrl.SetPower A_MOTOREA + C_MOTOREA, KONST, 2
End Sub

Private Sub cmdEzkerra_MouseDown(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.SetFwd C_MOTOREA
    PBrickCtrl.On C_MOTOREA
End Sub

Private Sub cmdEzkerra_MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.Off C_MOTOREA
End Sub

Private Sub cmdIrten_Click()
    PBrickCtrl.CloseComm
    End
End Sub
```

4.5.1. Kodearen deskribapena

Aurreko kapituluan ikusi genuen moduan, programa abiatzean Form_Load prozedura egikaritzen da. Programa honetan bi gauza egiteko erabiliko dugu: batetik komunikazio ataka irekitzeko (InitComm), eta bestetik, motoreen potentzia finkatzeko.

Motoreen potentzia: motoreen potentzia finkatzeko erabili dugun kode lerroa ondoko hau izan da:

PBrickCtrl.SetPower A_MOTOREA + C_MOTOREA, KONST, 2

Motore baten potentzia 0 eta 7 bitartekoa izan daiteke: 7 da azkarrena eta 0 motelena. Potentzia 0 mailan finkatzea ez du esan nahi motorea geldituko denik.. Kasu honetan A eta C motoreen potentzia maila 2 izango da, eta konstante baten bidez finkatzen da. Honek ez du eragin handirik izango motorearen abiaduran, baina bai eman dezakeen potentzian. Robota marruskadura handiko toki baten gainetik dabilenean, potentziarik handiena eman beharko zaio.

cmdAurrera botoia sakatzean robotak aurrera egingo du. Botoia sakatuta dagoenean, ondoko gertaera-prozedura egikarituko da:

```
Private Sub cmdAurrera_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
      PBrickCtrl.SetFwd A_MOTOREA + C_MOTOREA
      PBrickCtrl.On A_MOTOREA + C_MOTOREA 'Aurrera egiten du
End Sub
```

Bi motoreak aurrera mugitzeko konfiguratu ditu (SetFwd), eta ondoren martxan jarri ditu (On).

Botoia askatzean beste hau egikarituko da:

```
Private Sub cmdAurrera _MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.Off A_MOTOREA + C_MOTOREA
End Sub
```

eta motoreak gelditukо dira.

Ezker aldera biratzeko kodea oso antzekoa da. Kasu honetan eskuineko motorea jarriko dugu martxan, eta horrela ezkerrera biratzea lortuko dugu. SetFwd metodoari esker motorea aurrera mugituko da. Erabil daitezkeen beste metodoak hauek dira:

.SetRwd -motorea atzera mugituko da.

.AlterDir -aipatzen den motorearen noranzkoa aldatzen du.

4.5.2. Ariketa

Atzera joateko eta eskuin aldera biratzeko kodea falta da. Falta den kodea idazteko, kopia ezazu orain arte egindakoa, eta egin itzazu beharrezkoak diren aldaketak.

4.6. Irudien erabilpena komando-botoietan

Orain arte ikusi dugu nola aldatzen den botoi batean agertzen den testua, *Caption* propietatearen edukia aldatuz. Nahi izanez gero, testuaren ordez irudi bat jar dezakegu, eta hau oso lagungarria da hainbat programetan. Hori egiteko, jarrai itzazu ondoko urratsak:

- Hauta ezazu aldatu nahi duzun komando-botoia.
- Ezaba ezazu *Caption* propietatearen edukia (zerbait badu).
- Alda ezazu *Style* propietatea: **1 – Graphical**.
- Bila ezazu erabili nahi duzun fitxategi grafikoa *Picture* propietatearen bitartez.

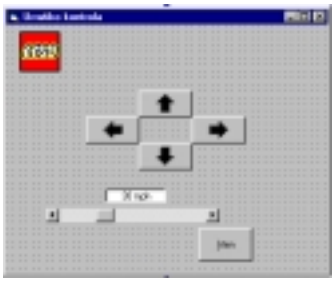
Ariketa honetan, Microsoft Visual Studio/Common/Graphics/Icons/Arrows direktorioan dauden irudiak erabili dira. Hala ere, baliteke zure ordenagailuan ez egotea, hasierako instalazioaren araberakoa baita.

4.7. Hobekuntzak robotaren kontrolean

Orain, gure programaren ahalmenak areagotuko ditugu. Aurreko programan, Form_Load() gertaera-prozeduran motoreen potentzia maila finkatu dugu, eta programan zehar ez dugu aldaketarik egin. Jarraian egingo ditugun aldaketei esker, edozein unetan motoreen potentzia maila aldatu ahal izango dugu. Erarik egokiena, desplazamendu-barra horizontal bat erabiltzea da. *HscrollBar* ikonoa da (izena kurtsorea gainean jartzean agertzen da). Testu lauki batean abiaduraren balioa erakutsiko du (ez da benetako abiadura izango).

Aurreko Urrutiko Kontrola programako formularioan, desplazamendu-barra jarriko dugu:

- Aukera ezazu *Barra de Desplazamiento Horizontal* () kontrola tresnen koadroan, eta jar ezazu kurtsorea formularioan (kurtsoreak gurutze baten itxura hartuko du). Saka ezazu saguaren ezkerreko botoia eta arrasta ezazu sagua nahi den tamainako lauki-zuzena egin arte. Aska ezazu botoia eta kontrola ikusi ahal izango da. Desplazamendu-barraren tamaina aldatu nahi baduzu, kontrola hautatu ondoren, tira ezazu inguruko puntu urdin batetik.



4.13 irudia
formularioa desplazamendu-barra eta testu laukiarekin.

Kontrol mota	Propietatea	Balioa
HscrollBar (desplazamendu-barra horizontala)	Nombre Max Min LargeChange SmallChange Value	hsbAbiadura 7 0 1 1 2
Text Box	Nombre Value Alignment	txtAbiadura 20 km/h Center

4.2 taula

- Egin ezazu klik-bikoitza desplazamendu-barraren gainean.

```
Private Sub hsbAbiadura_Change()
    PBrickCtrl.SetPower A_MOTOREA + C_MOTOREA, KONST, hsbAbiadura.Value
    TxtAbiadura.Text = Str(hsbAbiadura.Value * 10 + 10) + "km/h"
End Sub
```

4.7.1. Kodearen deskribapena

Desplazamendu-barra horizontalean balioak 0 eta 7 bitartekoak dira (gutxiena 0-gehiena 7). Une bakoitzeko konfigurazioa *hsbVelocidad.Value* propietatean gordeko dugu. Ondoko instrukzioaren bitartez, motoreen potentzia maila finkatuko dugu (hsbAbiadura.Value), hain zuzen, desplazamendu-barraren oraingo balioan.

```
PBrickCtrl.SetPower MOTOR_A + MOTOR_C, KONST, hsbAbiadura.Value
```

Prozeduraren hurrengo lerroan, abiaduraren balioa kalkulatuko dugu.

```
TxtVelocidad.Text = Str(hsbAbiadura.Value * 10 + 10) + "km/h"
```

Kalkulu hau ez da erreala, baina praktikatzeko baliagarria izan daiteke. *hsbVelocidad.Value* bider 10 egingo dugu abiadura balioa kalkulatzeko. Gehi 10 egingo dugu 0 potentzia mailan geldirik ez dagoelako. Testu-laukiaren Text propietatearen balioak testu-katea izan behar du, beraz, aurreko eragiketen emaitza den zenbakia testu-kate bihurtu beharko dugu. Horretarako Str() funtzioa erabiltzen da. Behin zenbakia testu-kate bihurtuta, “km/h” testu-katea gehituko diogu. Ohar zaitez testu datuak komatxoen artean idazten direla.

4.8. Hobekuntza gehiago

Gure programa ondo dabil, baina motoreak beti modu berean konektatu beharko ditugu (ezkerrekoa A-n eta eskuinekoa C-n). Orain egingo ditugun aldaketen ondorioz, programa

malguagoa izango da, eta erabiltzaileak motore bakoitza non dagoen konektatuta zehaztu ahal izango du. Horretarako bi kontrol mota berri erabiliko dugu: aukeraketa-botoiak eta frameak.

4.8.1. Aukeraketa-botoiak (OptionButton)

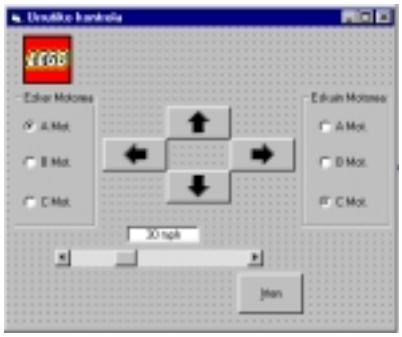
OptionButton ( )kontrolak bai ala ez izan daitekeen aukera erakusten du. Formulariotan, elkarren artean harremana duten aukeraketa-botoiak erabiltzen dira, eta ezin izaten da bat baino gehiago batera aukeratu. Hala ere, batzuetan aukeraketa-botoi talde bat baino gehiago behar da formulario batean. Hau *Frame* kontrolak erabiliz egin daiteke, honela, programak aukeraketa-botoi talde ezberdinak bereiztu ahal izango ditu.

4.8.2. Frame kontrola

Frame kontrolak kontrolen taldekatzeak identifikatzen ditu. Frameak formularioa funtzioen arabera zati dezake, adibidez, aukeraketa-botoi taldeak bereizteko, hemen egingo dugun bezala.

- Kontrolak taldekatzeko, has zaitez *Frame* kontrola marrazten ("xy" goiko aldeko ezkerreko izkinan duen ikonoa), eta ondoren, marraz itzazu kontrolak *Frame*-ren barruan.
- Ez ezazu ahaz inolako aukeraketa-botoi jarri aurretik *Framea* marraztu behar duzula. Jar itzazu ezkerreko motorearen aukeraketa botoiak ezkerreko *Frame*-an, eta eskuinekoarenak, eskuineko *Frame*-an.

Oharra: formulario bateko kontrol batzuk aukeratzeko, aukera itzazu saguaz <**Ctrl**> tekla sakatuta duzun bitartean. Ondoren, *Propiedades* leihora jo dezakezu eta guztien propietateak berdindu, adibidez, letra-tipoa edo kolorea.



4.14 irudia
formularioa bukatua

4.8.3. Kodea

Kontrol mota	Propietatea	Balioa
Frame	Nombre Caption	fraEzkerra Ezkerreko Motorea
Option Button	Nombre Caption Value	OptEzkerA A Motorea True

Option Button	Nombre Caption	OptEzkerB B Motorea
Option Button	Nombre Caption	OptEzkerC C Motorea
Frame	Nombre Caption	fra Eskuin Eskuineko Motorea
Option Button	Nombre Caption	OptEskuinA A Motorea
Option Button	Nombre Caption	OptEskuinB B Motorea
Option Button	Nombre Caption Value	OptEskuinC C Motorea True

4.3 taula

Jarraian kode osoa aurkezten da:

```
Option Explicit
Dim strEzkerMotorea, strEskuinMotorea As String

Private Sub Form_Load()
    PBrickCtrl.InitComm ‘Ordenagailuaren komunikazio ataka hasieratzen du
    strEzkerMotorea = A_MOTOREA     ‘A motorea hasierako aukera ezkerrean
    strEskuinMotorea = C_MOTOREA    ‘C motorea hasierako aukera eskuinean
    PBrickCtrl.SetPower strEzkerMotorea + strEskuinMotorea, 2, 2
End Sub

Private Sub cmdAurrera_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    PBrickCtrl.SetFwd strEzkerMotorea + strEskuinMotorea
    PBrickCtrl.On strEzkerMotorea + strEskuinMotorea
End Sub

Private Sub cmdAurrera _MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.Off strEzkerMotorea + strEskuinMotorea
End Sub

Private Sub cmdAtzera_MouseDown(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.SetRwd strEzkerMotorea + strEskuinMotorea
    PBrickCtrl.On strEzkerMotorea + strEskuinMotorea
End Sub

Private Sub cmdAtzera_MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.Off strEzkerMotorea + strEskuinMotorea
End Sub

Private Sub cmdEskuina_MouseDown(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.SetFwd strEzkerMotorea
    PBrickCtrl.On strEzkerMotorea
End Sub

Private Sub cmdEskuina_MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
```

```
    PBrickCtrl.Off strEzkerMotorea
End Sub

Private Sub cmdEzkerra_MouseDown(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.SetFwd strEskuinMotorea
    PBrickCtrl.On strEskuinMotorea
End Sub

Private Sub cmdEzkerra_MouseUp(Button As Integer, Shift As Integer, X _
As Single, Y As Single)
    PBrickCtrl.Off strEskuinMotorea
End Sub

Private Sub hsbAbiadura_Change()
    PBrickCtrl.SetPower strEzkerMotorea + strEskuinMotorea, 2,_
hsbAbiadura.Value
    txtAbiadura = Str(hsbAbiadura.Value * 10 + 10) + "km/h"
End Sub

Private Sub OptEskuinA _Click()
    strEskuinMotorea = A_MOTOREA          'A Motorea
End Sub

Private Sub OptEskuinB_Click()
    strEskuinMotorea = B_MOTOREA          'B Motorea
End Sub

Private Sub OptEskuinC_Click()
    strEskuinMotorea = C_MOTOREA          'C Motorea
End Sub

Private Sub optEzkerA_Click()
    strEzkerMotorea = A_MOTOREA           'A Motorea
End Sub
Private Sub optEzkerB_Click()
    strEzkerMotorea = B_MOTOREA           'B Motorea
End Sub
Private Sub optEzkerC_Click()
    strEzkerMotorea = C_MOTOREA           'C Motorea
End Sub

Private Sub cmdIrten_Click()
    PBrickCtrl.CloseComm
    End
End Sub
```

4.8.4. Kodearen deskribapena

Programak lehenengo lerroetan testu-kateak (string) gordeko dituzten bi aldagai hasieratzen ditu:

```
Option Explicit
Dim strEzkerMotorea, strEskuinMotorea As String
```

Form_Load gertaera-prozedurak konexioen balio lehenetsiak definitzen ditu, A_MOTOREA *strEzkerMotorea*-ri eta C_MOTOREA *strEskuinMotorea* aldagaiari esleituz. 4.3 taula begiratzen baduzu, ohartuko zara *optEzkerA* aukeraketa-botoiaren balioa True (egiazkoa) dela programaren hasieran. Honela, aukeratua izango den motorea eta formularioak adieraziko diguna bat etorriko dira. Gauza bera gertatzen da eskuineko motorearekin (balio lehenetsia *optEskuinC* da).

Aurreko kodean A_MOTOREA eta C_MOTOREA erabili ditugu. Kasu honetan, konexioa aldatzeko aukera eman nahi izan dugunez, *strEzkerMotorea* eta *strEskuinMotorea* aldagaiak erabili ditugu.

Ondoko gertaera-prozedura optEskuinA aukeraketa-botoian klik egitean egikaritzen da.

```
Private Sub optEskuinA_Click()
    strEskuinMotorea = A_MOTOREA ‘A motorea
End Sub
```

Honen ondorioz “0” (A_MOTOREA konstanteak ordezkatzen duena) balioa esleitzen zaio *strEskuinMotorea* aldagaiari, eta, A irteeran konektaturiko motorea ezkerreko motore moduan konfiguratuko da.

4.8.5. Programaren egikaritzea

- Gorde eta egikari ezazu programa.
- Konekta itzazu motoreak irteera ezberdinetan, eta hauta itzazu irteerak aukeraketa--botoien bitartez berriz konfiguratzeko.
- Gida ezazu robota programaren kontrolen bitartez.

Desplazamendu-barraren mugimenduak ezusteko ondorioak dakartza barra arrastatzen denean alboetako gezia erabili ordez (testu-laukiko edukia ez da aldatzen saguko botoia askatu arte). Hau ekiditeko, idatz ezazu ondoko kodea:

- Hauta ezazu *hbsAbiadura Código* leihoko goiko *Objeto* zerrendan.
- Hauta ezazu *Scroll Procedimientos* zerrendan.

Aurrekoaren ondorioz prozedura idazteko eredua agertuko da:

```
Private Sub hsbAbiadura_Scroll()
    PBrickCtrl.SetPower strEzkerMotorea + strEskuinMotorea, 2,_
hsbAbiadura.Value
    txtAbiadura = Str(hsbAbiadura.Value * 10 + 10) + "Km/h"
End Sub
```

5 Sentsoreen erabilera

5.1. Kapitulu honetako edukiak

Kapitulu honetan roboten zentzumen-aparatuaz arituko gara, baina motore-aparatuarekin harremanetan jarri gabe. Hau da, sentsoreak konfiguratu ondoren beren irakurketak eskuratuko ditugu, baina formulario batean aurkeztu baino ez dugu egingo.

1. Sentsoreen erabilera: sentsore motak eta sentsore moduak
2. Ukitze-sentsorea eta argi-sentsorea
3. Sentsoreen irakurketen aurkezpena formularioen bitartez.
4. Tresnen koadroa: temporizadorea (Timer), irudi geometrikoak (Shape), eta zerrendak (Combo-box eta List-Box) kontrolak.

5.2. Zentzumen-aparatua

Irteeretan konektaturik dauden motoreak kontrolatzeaz gain, RCXk informazioa jaso dezake sarreretan konektatutako sentsoreetatik. Sentsore ezberdinak konekta daitezke RCXn, batzuk aktiboak (funtzionatzeko energia behar dutenak, argi sentsorea bezala), eta besteak pasiboak (adibidez, ukitze-sentsoreak). LEGO MindStorms oinarrizko kit-ak bi ukitze-sentsore eta argi-sentsore bat dauzka. “Ultimate Accesory Kit”-ak angelu-sentsore bat dakar, eta bezeroentzat LEGOk duen zerbitzuaren bitartez[4] tenperatura sentsorea eskura daiteke. Hortik aurrera, erabiltzaileak bere sentsoreak egin ditzake.

Motoreak konektatzean konektorea jartzen den orientazioak bere ondorioak ditu, sentsoreen kasuan, ordea, berdin da nola konektatzen diren.

Sentsoreak erabiltzeko lehen urratsa sentsoreak konfiguratzea da. Horretarako hauxe definitu behar dugu: zein sentsore mota ari gara erabiltzen, eta zein formatuan nahi ditugu egiten dituen irakurketak.

5.1 irudia: argi-sentsorea ukitze-sentsorea kable konektorea

[4] DACTAren banatzailearen bitartez ere eskura daitezke sentsoreak.

5.2.1. Ukitze-sentsoreak

Konfigura dezagun orain ukitze-sentsorea.

Programa berria sortzeko, proiektu berri bat behar duzu:

- Abiaraz ezazu Visual Basic. *Nuevo Proyecto* agertzen bada, egin ezazu klik-bikoitza *Lego* ikonoaren gainean. Agertzen ez bada, hauta ezazu *Archivo* menuan *Nuevo Proyecto*.
- Aurreko kapituluan egin duzun moduan, gorde itzazu fitxategi guztiak **Sentsoreak** izenarekin. Zure proiektua gordetzeko direktorioa C:\VBLEGO\kap05 izango da.
- Diseina ezazu *frmSentsoreak* formularioa 5.1 taulako datuak erabiliz.

5.1 irudia
hasteko, formulario sinple hau egingo duzu

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	frmSentsoreak Sentsoreak
CommandButton	Nombre Caption	cmdPoll &Poll
CommandButton	Nombre Caption	cmdIrten &Irten
Cuadro de texto	Nombre Alignment Caption	txtPoll 2 – Center (Utzi hutsik)

5.1 taula

- Idatz ezazu ondoko kodea:

```
'Aldagai guztiak deklaratu behar dira
Option Explicit

Private Sub cmdIrten_Click()
    PBrickCtrl.CloseComm
    End
End Sub

Private Sub cmdPoll_Click()
    ‘ 1 irteera ukitze-sentsore eran konfiguratzen du
    PBrickCtrl.SetSensorType SENTSOREA_1, UKITZE_SENTS
    ‘ testu-laukiari 1 sentsorearen irakurketa ematen dio.
    txtPoll.Text = PBrickCtrl.Poll(SENBALIO, SENTSOREA_1)
End Sub
```

```
Private Sub Form_Load()
    PBrickCtrl.InitComm
End Sub
```

5.2.2. Kodearen deskribapena

cmdPoll_Click() gertaera-prozeduraren lehenengo lerroak 1 sarreran konektatuta dagoen sentsore mota definitzen du. Kasu honetan ukitze-sentsore bat izango da.

```
PBrickCtrl.SetSensorType SENTSOREA _1, UKITZE_SENTS
```

Ondoko taulan sentsore mota zerrenda ematen da, beren zenbaki-balioa eta dagozkien konstanteekin batera:

Sentsore mota	Konstantea	Zenbakia
Bat ere ez	EZ_MOTA	0
Ukitze-senstorea	UKITZE_SENTS	1
Tenperatura	TEMPER_SENTS	2
Argi-sestsorea	ARGI_SENTS	3
Angelua	ANGELU_SENTS	4

5.2 taula

Sentsorea konfiguratu ondoren, bere irakurketa eskuratu ahal izango dugu. Ondoko instrukzio lerroak 1 izeneko sarreran konektatuta dagoen sentsorearen irakurketa txtPoll testu--laukian jarriko du (balio booleara, hau da, egiazkoa “1” ala faltsua “0”):

```
txtPoll.Text = PBrickCtrl.Poll(SENBALIO, SENTSOREA_1)
```

Poll metodoak RCXtik informazio mota ezberdinak eskuratzeko erabil dezakegu. Lehen parametroak eskuratu nahi dugun datu mota adierazten du (SENBALIO: sentsore baten irakurketa), eta bigarrenak, kasu honetan, zein sentsorearen irakurketa nahi dugun. Bigarren parametroaren balioa datuaren jatorriaren araberakoa da, adibidez, datuaren jatorria aldagai bat bada, zenbakia 0 eta 31 artekoa izango da). 5.3 taulan balio hauek jasotzen dira.

Datu Iturria	Konstantea	Zenbakia	Deskribapena
0	ALD	0-31	Aldagaia 0 – 31
1	TENPOR	0-3	Tenporizadorea 0 – 3
2	KONST	-	-
3	MOTEST	0, 1, 2	Motorearen egoera. Informazioa paketatua dago: Bit 7: ON/OFF 1/0 Bit 6: Balazta/Float 1/0 Bit 5: Irteera zbk. HiBit Bit 4: Irteera zbk. LoBit Bit 3: Noranzkoa CW/CCW[5] 1/0 Bit 2: Potentzia maila: bit esanguratsuena Bit 1: Potentzia maila: Bit 0: Potentzia maila: bit gutxien esanguratsua
4	RAN	-	-
8	KEYS	-	Programa zenbakia. Zein programa dagoen orain aukeratua.
9	SENBALIO	0, 1, 2	Sentsorearen irakurketa sarreran neurtutako balioa. Sentsore moduaren araberakoa da.
10	SENMOTA	0, 1, 2	Sentsore mota. Sarrera zein sentsore motarako dagoen konfiguratua adierazten du.
11	SENMODU	0, 1, 2	Sentsore modua. Sarrera zein sentsore moduan dagoen konfiguratua adierazten du.
12	SENRAW	0, 1, 2	Raw sentsorea, hau da, sarreran neurtutako balioaren analogoa.
13	BOOL	0, 1, 2	Sentsore Boolearra. Sarrerako balio boolearra ematen du.
14	ERLOJU	0	Erlojua. Zbk osoa. MSB = orduak y LSB = minutuak.
15	PBMENS	0	RCXn gordetako PBMezua ematen du.

5.3 taula

5.2.3. Programaren egikaritzea

- Gorde ezazu programa.
- Konekta ezazu ukitze-sentsore bat 1 sarreran.
- Piztu ezazu RCX.
- Saka ezazu Poll botoia, eta “0” zenbakia testu-laukian agertuko da.
- Egin ezazu presioa ukitze-sentsorean, eta askatu gabe, saka ezazu berriro Poll: testu-laukian “1” zenbakia agertuko da.

[5] CW: biraketa erlojuaren orratzen noranzkoan. CCW: biraketa erlojuaren orratzen noranzkoaren aurkakoan.

5.3. Sentsore moduak

Sentsorearen irakurketak jasotzen ditugun modua alda daiteke. SetSensorMode metodoak jakinarazten dio RCXri datuak nola eskuratu nahi ditugun. Metodo honen sintaxia SetSensorMode(Number, Mode,Slope) da.

Number (zenbakia) konfigurazioa zein sentsoreari dagokion adierazten du.

Mode (modua) ondorengo taulako zenbakia (edo konstantea) zutabean ematen den balioa.

Zenbakia	Konstantea	Sentsore modua	Deskribapena
0	RAW_MODUA	Raw	Tratatu gabeko datu analogikoa (0 – 1023)
1	BOOL_MODUA	Boolearra	Egiazkoa ala faltsua
2	TRANS_KONT_MODUA	Transizioa	Transizioak zenbatzen ditu (Bai positiboak, bai negatiboak)
3	PERIOD_KONT_MODUA	Kontadore periodikoa	Periodo osoak kontatzen ditu bakarrik (transizio positibo bat + negatibo bat edo alderantziz)
4	PORTZ_MODUA	Portzentajezkoa	Sentsorearen irakurketa eskala osoaren ehunekoan ematen da.
5	CELSIUS_MODUA	Celsius	Tenperatura Celsius unitatetan
6	FAHRENHEIT_MODUA	Fahrenheit	Temperatura Fahrenheit unitatetan
7	ANGELU_MODUA	Angelua	Sarrerako datua angelu zatikitan kontatzen da.

5.4 taula

Slope modu boolearra aukeratzen denean erabiltzen da, bestela “0” balioa eman dakioke. Modu boolearra aukeratu ezkero, Slope-k egiazkoa ala faltsua den erabakitzeko modua adierazten du. Beste aldetik, sarreretan gertatzen diren aldaketen aurrean kontadoreek duten erantzunean eragina izango du.

0: Neurketa absolutua (eskala osoaren %45-etik behera = egiazkoa, eta eskala osoaren %55etik gora = faltsua). Adibidez, sakatutako ukitze-sentsoreak (tentsio baxuko neurria) TRUE (Egiazkoa) balioa ematen du.

1-31: Neurketa dinamikoa. Zenbakiak *slope* dinamikoaren tamaina adierazten du. Adibidez, zenbat *bit-count*-eko aldaketa behar den bi irakurketen artean egoera boolearrean aldaketa eragiteko.

SetSensorType metodoak automatikoki aldatzen du ukitze-sentsorearen modua boolearrera, eta argi-sentsorearena portzentajezkora. Beraz, bestelako modu bat interesatzen bazaigu, SetSensorType metodoa SetSensorMode metodoaren aurretik jarri behar da, bestela alperrikakoa izango da SetSensorMode metodoa erabiltzea.

5.3.1. ComboBox eta ListBox

Programaren bitartez, RCXri sentsoreen irakurketak zein modutan eskuratu nahi ditugun adierazteko aukera izan nahi badugu, *Combobox* eta *ListBox* kontrolak erabil ditzakegu.

Bi kontrol hauei esker, erabiltzaileari aukera zerrenda bat eskaintzen diogu, berak interesekoa duena hauta dezan. Bien arteko diferentziak txikiak dira:

- *ComboBox* batean testua idatz daiteke egikaritzen den artean.
- Biek estilo ezberdina dute, *Listbox*-ek ezin du beheraka zabaltzen den zerrenda bat eduki, eta *ComboBox*-ek bai. Kasu ezberdinetan erabil daitezke.

5.3 irudia:
ezkerrean *ComboBox* eta eskuinean *ListBox* kontrolak

Has gaitezen programan behar diren aldaketak egiten:

- Egizu klik tresnen koadroko *ComboBox* ( )kontrolean.
- Eman iezaiozu **cboModua** izena.
- *ComboBox*-en balioak *List* propietatearen bitartez jartzen dira. Egin ezazu klik *List* propietatearen gainean, eta ondoren, bere eskuineko gelaxkako geziaren gainean.
- Idatz ezazu **Raw**.
- Saka ezazu <**Ctrl.+Enter**> tekla konbinazioa kurtsorea hurrengo lerrora eramateko.
- Idatz ezazu **Boolearra**.
- Bukatzeko, saka ezazu <**Enter**> tekla, edo egin ezazu klik zerrendatik kanpoko edozein tokitan.

5.3 irudia
estilo aldaketa *Propiedades* leihoan

- Alda ezazu estiloa (*Style*): 2 – Dropdown List.

5.3.2. Kodea

```
Option Explicit
Dim iModua As Integer

Private Sub cmdIrten_Click()
    PBrickCtrl.CloseComm
    End
End Sub

Private Sub kmdPoll_Click()
    ‘Begiratu zein den nahi den modua
    If cboModua.ListIndex = 0 Then
        iModua = RAW_MODUA
    ElseIf cboModua.ListIndex = 1 Then
        iModua = BOOL_MODUA
    EndIf
    '1 sarrerako sentsorea ukitze-sentsore moduan konfiguratzen du
    PBrickCtrl.SetSensorType SENTSOREA_1, UKITZE_SENTS
    ‘Irakurketa era boolearrean hartzen du
    PBrickCtrl.SetSensorMode SENTSOREA_1, iModua, 0
    ‘Sentsorearen irakurketa testu-laukira
    txtPoll.Text = PBrickCtrl.Poll(SENBALIO, SENTSOREA_1)
End Sub

Private Sub Form_Load()
    PBrickCtrl.InitComm ‘Ordenagailuko komunikazio ataka hasieratzen du
    cmbModua.Text = cmbModua.List(0) ‘Lehenengo elementua erakusten du
End Sub
```

5.3.3. Kodearen deskribapena

Kodearen hasieran Integer (zenbaki osoa) motako aldagai bat hasieratzen da: iModua. Aldagai honetan *ComboBox*-en aukeratutako moduaren balioa gordeko dugu.

```
‘Begiratu zein den nahi den modua
If cboModua.ListIndex = 0 Then
    iModua = RAW_MODUA
ElseIf cboModua.ListIndex = 1 Then
    iModua = BOOL_MODUA
EndIf
```

ComboBox-en lehen aukeraren balioa 0 da, hurrengoa 1, eta abar. *ListIndex* propietateak *ComboBox*-en aukeratutako balioa du. Balioa 0 bada, iModua aldagaiari RAW_MODUA esleituko zaio, eta balioa 1 bada, BOOL_MODUA esleituko zaio.

```
PBrickCtrl.SetSensorMode SENTSOREA_1, iModua, 0
```

Instrukzio honen bitartez, sentsorea iModua-n gordetako balioari dagokion moduan konfiguratuko da.

Bestetik, hasieran aukera lehenetsia erakutsi behar denez, Form_Load gertaera-prozeduran balio hori definitu beharko dugu. Horretarako ondoko instrukzioa erabiliko dugu

```
cboModua.Text = cboModua.List(0)    ' Lehen elementua erakusten du
```

Egikari dezagun programa emaitza nahi dugun modukoa ote den egiaztatzeko.

- Gorde ezazu proiektua.
- Egikari ezazu proiektua.

- Hauta ezazu lehen aukera, eta egin ezazu presioa ukitze-sentsorea; ondoren, egizu gauza bera bigarren aukeraz. Gogora itzazu irakurketen balioak.

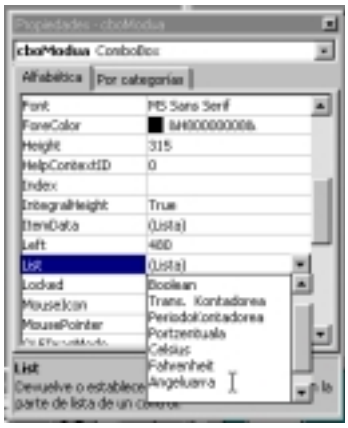
5.3.4. Kodea biribildu

Ariketa honetan idatzi duzun kodearen zati handi bat ez zen beharrezkoa. Berehala ohartuko zara *ComboBox*-en indize balioei eta modu ezberdinen zenbakizko balioak arretaz begiratzen badituzu. Hau hala izango da moduak taulako ordenan sartu badituzu.

- Gehi iezazkiozu ondoko balioak *ComboBox*-en *List* propietateari arestian egin den modu berean:

  Trantsizio kontadorea
  Periodo kontadorea
  Portzentajezkoa
  Celsius
  Fahrenheit
  Angeluarra

List-ak 5.4 irudian erakusten den itxura bera eduki beharko luke (Oharra: Raw sarrera bertan dago, baina bistatik kanpo).



5.4 irudia
Zure zerrenda hemen ikusten diren elementu berberak eduki beharko lituzke.

- Egin itzazu kodean aldaketa hauek:

```
Option Explicit

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub

Private Sub cmdPoll_Click()
    '1 sarrerako sentsorea ukitze-sentsore moduan definitzen du
      PBrickCtrl.SetSensorType SENTSOREA_1, UKITZE_SENTS
    ‘Irakurketa era boolearrean hartzen du
      PBrickCtrl.SetSensorMode SENTSOREA_1, cboModua.ListIndex, 0
    ‘Sentsorearen irakurketa testu laukira
      txtPoll.Text = PBrickCtrl.Poll(SENBALIO, SENTSOREA_1)
End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm ‘Ordenagailuko komunikazio ataka hasieratzen du
```

```
    cboModu.Text = cboMode.List(0) 'Lehenengo elementua erakusten du
End Sub
```

- Gorde ezazu berriz proiektua.
- Egikari ezazu proiektua.
- Egin ezazu klik Poll botoian.

5.5 irudia
Proiektua egikaritzen baduzu, sentsore ezberdinen irakurketak eskuratu ahal izango dituzu.

Sentsore mota eta moduak era ezberdinetan konbina daitezke. Hala ere, Angeluar, Celsius eta Fahrenheit moduak ez dira erabilgarriak ukitze sentsorearekin.

5.4. Argi sentsorea

Programaren egikaritzean sentsore mota hautatzeko aukera izatea oso interesgarria izan daiteke (orain arte sentsore modua aukeratu ahal izan dugu soilik), programan aldaketak egin gabe sentsore mota eta modu konbinaketa ezberdinekin esperimentatu ahal izango dugulako.

- Jar ezazu beste ComboBox bat zure formularioan.

Kontrol mota	Propietatea	Balioa
ComboBox	Nombre List Style	CboMota Bat ere ez Ukitze-S. Tenperatura Argia Angeluarra 2 – Drpdown List

5.5 taula

Oraingo honetan, kodea idaztean, erabil ezazu *cboMota.ListIndex* sentsore mota konfiguratzeko, eta lehen aukera (bat ere ez) izan dadila aukera lehenetsia programa abiaraztean.

- Gorde eta egikari ezazu programa berriro.
- Behin sentsorea konektatuta, alda itzazu sentsore modua eta mota, eta ondoren, eskura ezazu bere irakurketa.

5.5. Adreiluen antolatzailea

Proiektu berri honen helburua bi kolore ezberdinetako LEGO adreiluak bereiztuko dituen “Adreiluen antolatzailea” izango da. Hau kolore ezberdinek islatzen duten argi intentsitate ezberdinari esker egin ahal izango dugu. Lehen fase batean kolore bakar bateko adreiluak islatzen duen argiaren intentsitatea neurtuko dugu, eta bigarrenean programa osatuko duzu.

Adreiluen antolatzailea muntatzeko instrukzioak A eranskinean aurki ditzakezu.

Hurrengo lerroetan programa honen abiapuntua deskribatzen da, baina zure esku geldituko da antolatzailea bukatzea.

- Hauta ezazu *Archivo* menuko *Nuevo proyecto*.
- Hauta ezazu LEGO ikonoa *Nuevo proyecto* leihoan, eta egin ezazu klik *Aceptar* botoian.
- *Form1* leihoa aukeratua dagoela, hauta ezazu *Archivo* menuko *Guarda Form1*.
- Agertuko den elkarrizketa-koadroan hauta ezazu C:\VBLEGO\Kap05 direktorioa formularioa gordetzeko.
- Eman iezaiozu **Antolatzailea** izena, eta saka ezazu *Guardar* botoia.
- Hauta ezazu *Archivo* menuan *Guardar proyecto.*
- Lehen, .bas fitxategia gordeko da. Eman iezaiozu **Antolatzailea** izena (formularioa gordetako toki berean gordeko da).
- Jarraian, Visual Basic-ek .vbp fitxategiaren izena eskatuko dizu. Idatz ezazu **Antolatzailea** izena eta saka ezazu *Guardar* botoia.

5.5.1. Denbora kontrola (Timer)

Komando botoi bat sakatzen den bakoitzean, lotua duen gertaera-prozedura egikaritzen da. Zure nahia ekintza zehatz bat era automatikoan periodikoki gertatzea bada, *Timer* kontrola erabil dezakezu. *Timer* kontrolari esker, prozedura bat aurrez zehaztutako denbora-tarteka egikarituko da. *Interval* propietatea denbora-tarteen luzera definitzen du, eta bere balioa 0 eta 65535 izan daiteke (unitatea 1= milisegundu 1). *Timer* kontrola ez da ikusten programa egikaritzen denean, bakarrik diseinu fasean ikus daiteke.

5.5.2. Shape kontrola

Shape irudi geometriko hauek marrazteko erabil daiteke:

- Laukizuzenak
- Karratuak
- Zirkuluak
- Obaloak
- Erpinak biribilduak dituzten karratuak
- Erpinak biribilduak dituzten laukizuzenak

Shape () kontrolaren *Shape* propietatearen balioa aldatuz nahi dugun irudi geometrikoa marraztu ahal izango dugu. Proiektu honetan, antolatzaileak ikusten duen kolorea adierazteko erabiliko dugu. Hasieran, beltzez ikusiko da, baina antolatzaileak adreilu berde[6] bat ikusten duenean, formularioan marraztutako laukizuzena berde kolorez ikusiko da.

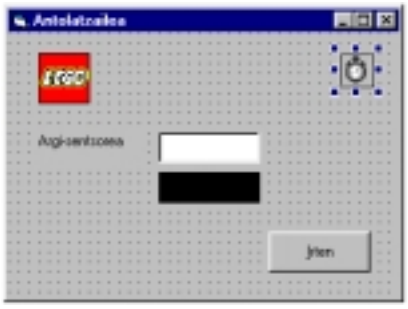
- Egin ezazu formularioa 5.6 taulako datuak erabiliz.

[6] Berez ez du kolorea antzemango. Beste kolore batekoa sartzen badugu erantzun bera izango du, baina programa osorik dagoenean bi koloreen artean bereizketak egin ahal izango ditu (tranparik egiten ez diogun bitartean).

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	FrmAntolatzailea Adreiluen antolatzailea
CommandButton	Nombre Caption	CmdIrten &Irten
Timer	Nombre Enabled Interval	TmrBegira True 1000
TextBox	Name Text	TxtBegira (hutsik utzi)
Label	Nombre Caption	LblBegira Argi-sentsorea
Shape	Nombre BorderStyle FillStyle	ShpAdreilua 0 – Transparent 0 - Solid

5.6 taula

Bukatutako formularioak 5.6 irudiko itxura izango du.



5.6 irudia
Hauek dira zure formularioak eduki behar dituen osagaiak.

- Idatz ezazu ondorengo kodea:

```
Option Explicit

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub

Private Sub Form_Load()
      With PBrickCtrl
            .InitComm ‘Ordenagailuaren komunikazio ataka hasieratzen du
            .SetSensorType SENTSOREA_1, UKITZE_SENTS ' 1 sentsorea
            ukitze-sentsorea
            .SetSensorType SENTSOREA_3, ARGI_SENTS ' 3 sentsorea argi-
            -sentsorea
            .SetSensorMode SENTSOREA_3, RAW_MODUA, 0 ' Modua
            portzentualetik Raw-era aldatzen du
      End With
End Sub
```

- Itzul zaitez *Objeto* bistara, egin ezazu klik-bikoitza formularioan jarritako *Timer* kontrolaren gainean eta idatz ezazu ondoko kodea:

```
Private Sub tmrBegira_Timer()
      If PBrickCtrl.Poll(SENBALIO, SENTSOREA_1) = 1 Then
```

```
            TxtBegira = PBrickCtrl.Poll(SENBALIO, SENTSOREA_3)
            ShpAdreilua.FillColor = QBColor(2) ' berdea
      Else
            ShpAdreilua.FillColor = QBColor(0) ' beltza
      End If
End Sub
```

5.5.3. Antolatzailea programaren egikaritzea

- Gorde ezazu proiektua.
- Egikari ezazu proiektua.

Hasieran irudi geometrikoa beltza da. Ukitze-sentsorean presioa egitean testu-laukian argi--sentsorearen Raw irakurketa ikusiko duzu, eta irudiaren kolorea berde bihurtuko da. Ukitze--sentsorea askatzean irudiaren kolorea berriro beltz bihurtuko da, eta testu-laukian eskuratutako azken balioa geldituko da.

5.5.4. Programaren deskribapena

Programa abiarazten denean, 1 sarreran konektatuta dagoen sentsorea ukitze-sentsore bezala konfiguratzen da, eta 3 sarreran dagoena, argi-sentsore bezala.

Arreta berezia eskatzen duen lerroa Form_Load prozedurako lehenengoa da, lana erraztuko digun instrukzioa baita.

```
With PBrickCtrl
```

Instrukzio honi esker, *PbrickCtrl* kontrolari dagozkion metodoak idazteko orduan ez dugu aldi guztietan aurrean *PbrickCtrl* idatzi beharko, prozedura honetan bertan ikus daitekeen moduan. Kontrolaren beharra bukatzen denean, *End With* idatzi beharko da.

tmrBegiraTimer: prozedura hau etengage egikarituko da segundu bateko denbora-tartea utziaz (1000 ms). Lehenengo kode lerroak ukitze sentsorea presionatuta dagoen egiaztatuko du.

```
If PBrickCtrl.Poll(SENBALIO, SENTSOREA_1) = 1 Then
```

Kontaktua badago (hau da, berdin 1 bada) argi-sentsorearen irakurketa *txtBegira* testu-laukiari esleituko zaio, eta laukizuzenaren kolorea berde bihurtuko da. Honetarako, *shpAdreilua* kontrolaren *FillColor* propietatearen edukia aldatuko dugu.

```
txtBegira = PBrickCtrl.Poll(SENBALIO, SENTSOREA_3)
shpAdreilua.FillColor = QBColor(2) ' berdea
```

Ukitze-sentsorea askatzean, laukia berriz belztuko da.

QBColor(2) funtzioak 5.7 taulak adierazten duen kolorea itzultzen du.

Zenbakia	Kolorea	Zenbakia	Kolorea
0	Beltza	8	Grisa
1	Urdina	9	Urdin argia
2	Berdea	10	Berde argia
3	Cyan	11	Cyan argia
4	Gorria	12	Gorri argia
5	Magenta	13	Magenta argia
6	Horia	14	Hori argia
7	Zuria	15	Zuri distiratsua

5.7 taula

5.5.5. Ariketa

Egin itzazu Antolatzailea programan behar diren aldaketak kolore urdin eta berdeko adreiluak bereizteko gai izan dadin. *Shape* kontrolak argi-sentsorearen azpian dagoen LEGO adreiluaren kolorea erakutsi beharko du.

Gogoan izan argi-sentsorearen irakurketak kolorearen araberakoak direla.

Aldaketa guztiak tmrBegira_Timer prozeduran egin behar dira. Hori egiteko ondoko eskema izan daiteke:

```
Private Sub tmrPoll_Timer()
      ' sentsorearen irakurketa gordetzeko aldagaia. Integer (zenbaki
      osoa) modukoa
      Dim iArgiRaw As Integer

      If PBrickCtrl.Poll(SENBALIO, SENTSOREA_1) = 1 Then
            iArgiRaw = PBrickCtrl.Poll(SENBALIO, SENTSOREA_3)
            txtBegira = iArgiRaw
            ' Idatz ezazu hemen koloreei dagokien kodea
      End If
End Sub
```

If ... Then ... Else kontrol egitura bat beste baten barruan jar dezakezu. *iArgiaRaw* aldagaia tmrBegira_Timer prozeduraren barruan hasieratu denez, bertan bakarrik erabil daiteke. Programako prozedura guztietan erabili ahal izateko, programaren hasieran hasieratu behar da, aldagai global moduan.

6 Aldagaiak RCXn

6.1. Kapitulu honetako edukiak

Orain arte, Visual Basic-eko aldagai batzuk erabili ditugu, beren balioa ordenagailuan bertan gordetzen dutenak. Kapitulu honetan RCXko aldagaietan informazioa nola gorde ikasiko duzu.

1. RCXko aldagaien erabilpena.
2. Zenbakiak eta testu-kateak.
3. Errore mezuak.

6.2. RCXko aldagaien ezaugarriak

RCXk 32 aldagai global erabil ditzake. Aldagai hauek –32786 eta 32786 arteko balio osoak gorde ditzakete. Aldagaien balioak modu ezberdinetan alda daitezke, besteak beste, batuketak, kenketak, biderketak eta zatiketak egiten. Kasu guztietan emaitzek –32786 eta 32786 arteko balio osoa izan beharko dute, eta zatiketaren kasuan, zatidura zenbaki oso gertuenera borobilduko da. Aldagaien balioak ordenagailutik irakur daitezke Poll metodoaren erabiliz. Aldagaien izenak 0 eta 31 arteko balioak dira.

6.3. Proiektu proposamena

Kapitulu honetan egingo dugun proiektuak RCXko aldagaien balioak irakurri eta aldatzeko aukera emango digu. Hau formularioak eskainiko dizkigun aukeren bitartez egingo dugu.

Bigarren urrats batean, datuak sartzean sor daitezkeen erroreak ekiditeko aldaketa batzuk egingo ditugu.

- Hauta ezazu *Archivo* menuko *Nuevo proyecto*.
- Hauta ezazu *Lego* ikonoa *Nuevo proyecto* leihoan, eta saka ezazu *Aceptar* botoia.
- Aurreko kapituluetan bezala, gorde itzazu fitxategi berriak. Eman iezaiezu **Aldagaiak** izena eta gorde itzazu C:\VBLego\Kap06 direktorio berrian.
- Erabil itzazu 6.1 taulako datuak formulario berria egiteko.

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	FrmAldagaiak Aldagaiekin jolasean
CommandButton	Nombre Caption Font	cmdSet &Balioa eman Times New Roman 10
CommandButton	Nombre Caption Font	cmdPoll B&egiratu Times New Roman 10
CommandButton	Nombre Caption Font	cmdIrten &Irten Times New Roman 10
TextBox	Nombre Alignement Font Text	TxtSetAld 2 – Center Times New Roman 10 (Hutsik utzi)
TextBox	Nombre Alignement Font Text	TxtSetBalioa 2 – Center Times New Roman 10 (Hutsik utzi)
TextBox	Nombre Alignement Font Text	TxtPollAld 2 – Center Times New Roman 10 (Hutsik utzi)
TextBox	Nombre Alignement Font Text	TxtPollBalioa 2 – Center Times New Roman 10 (Hutsik utzi)
Label	Nombre Alignement Caption Font	LblSet 2 – Center aldagaiari Times New Roman 10
Label	Nombre Alignement Caption Font	LblPoll 2 – Center aldagaiaren balioa Times New Roman 10

6.1 taula

6.1 irudia:
formularioak itxura hau izango du

- Idatz ezazu kode hau:

```
'Aldagai guztiak hasieratu behar dira
Option Explicit

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub

Private Sub kmdPoll_Click()
      'Aldagaiaren balioa irakurri
      txtPollBalioa.Text= PBrickCtrl.Poll(ALD,Val(txtPollAld))
End Sub

Private Sub kmdSet_Click()
      'Aldagaiaren balioa finkatu
      PBrickCtrl.SetVar Val(txtSetAld),KONST,Val(txtSetBalioa)
End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm
End Sub
```

- Gorde ezazu proiektua.
- Egikari ezazu proiektua.
- Piztu ezazu RCX.
- Eskura ezazu 15 izeneko aldagaiaren balioa.
- Eslei iezaiozu 15 izeneko aldagaiari 3333 balioa.
- Eskura ezazu berriro 15 izeneko aldagaiaren balioa.

Ikus dezakegu orain aldagaiaren balioa 3333 dela.

6.3.1. Kodearen deskribapena

Aldagaiei balioa esleitzeko SetVar instrukzioa erabiliko dugu.

```
PBrickCtrl.SetVar Val(txtSetAld),KONST,Val(txtSetBalioa)
```

txtSetAld testu laukiaren edukia testu-kate bat da (beste edozein testu-laukian gertatzen den bezala), baina *SetVar* metodoak zenbakizko balioa behar du. Horregatik, testu-katea zenbakizko balio bihurtu behar dugu. Hau *Val()* funtzioaren bitartez egiten da. Funtzio honen osagarria *Str()* da, hain zuzen, zenbakizko balioa testu-kate bihurtzen duena.

`Str(22334)="22334"`	Zenbakia	⇒	Testu katea
`Var("21")=21`	Testu katea	⇒	Zenbakia

SetVar instrukzioaren sintaxia *SetVar (VarNo, Source, Number)* da. Metodo honen lehen argumentua aldatu nahi dugun aldagaiaren izena da (0 eta 31 bitartekoa). Bigarrena, esleituko dugun balioaren jatorria, kasu honetan konstante bat, eta hirugarrena aldagaiari esleituko diogun balioa.

Aldagaiaren balioa eskuratzeko ondoko instrukzioa erabiliko dugu:

```
txtPollBalioa.Text=PBrickCtrl.Poll(VAR,Val(txtPollAld))
```

Instrukzio honen bitartez aldagai baten balioa eskuratu nahi dugula jakinaraziko diogu, eta zein aldagaiarena. Kasu honetan, txtPollAld testu-laukiko aldagaiaren zenbakizko balioa zein den jakin nahi dugu.

- Egikari ezazu berriro programa.
- Piztu ezazu RCX.
- Eslei iezaiozu 50 000 balioa 23 izeneko aldagaiari.

Errore bat gertatuko da, 50 000 zenbakia handiegia baita (> 32767). Saka ezazu *Terminar* botoia *Error* elkarrizketa-koadroa ixteko.

- Eslei iezaiozu 245 balioa 50 izeneko aldagaiari.

Beste errore bat gertatuko da, aldagaien balioa 0 eta 31 bitartekoa izan behar duelako.

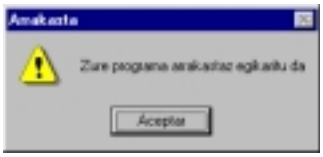
- Irten zaitez programatik.

6.4. Mezu-koadroak

Batzuetan, programak erabiltzaileari informazioren bat eman behar dionean, monitorean mezu bat erakusten du, gehienetan *Aceptar* botoi batekin, erabiltzaileak mezua jaso dezan. Visual Basic-en zeure mezuak egin ditzakezu *MsgBox* instrukzioa erabiliz. Adibidez, *cmdMezua* izeneko komando bat baduzu, mezu-koadro batekin lotu dezakezu jarraian adierazten den moduan:

```
Private Sub cmdMezua_Click()
      MsgBox “Zure programa arrakastaz egikaritu da”, _
      vbExclamation, ”Arrakasta”
End Sub
```

Kode honek, cmdMezua botoia sakatzean, 6.2 irudian ikus dezakezun mezu-koadroa sortuko du.



6.2 irudia
Hau da egin duzun mezu-koadroa

Aldagai baten balioa eskuratzen saiatu aurretik, beharrezkoa da egiaztatzea aldagai horren izena 0 eta 31 bitarteko zenbaki bat den. Horretarako Visual basic-eko If ... Then ... Else kontrol egitura erabiliko dugu.

- Egin itzazu ondoko aldaketak *cmdPoll_Click* prozeduraren kodean. Hauta ezazu nahi duzun errore-mezua.

```
Private Sub cmdPoll_Click()
      If Val(txtPollAld)<0 Or Val(txtPollAld)>31 Then
            ‘Idatz ezazu hemen nahi duzun mezua MsgBox instrukzioa_
      erabiliz
      Else
            TxtPollBalioa.Text=PBrickCtrl.Poll(Ald,Val(txtPollAld))
      End If
End Sub
```

- Gorde ezazu programa
- Piz ezazu RCX.
- Egikari ezazu programa.
- Irakur ezazu 41 izeneko aldagaia.

Errore-mezu bat agertuko da gertatutakoa jakinaraziz.

6.3 irudia
Errore-mezua.

6.4.1. Kodearen deskribapena

*CmdPoll*_Click prozeduraren lehenengo lerroan If ... Then ... Else kontrol egitura dago:

```
If Val(txtPollVar)<0 Or Val(txtPollVar)>31 Then
```

Hemen bi baldintza egiaztatzen dira:

1. Ea *txtPollAld*-en zenbakizko balioa 0 baino txikiagoa den eta
2. Ea *txtPollAld*-en zenbakizko balioa 31 baino handiagoa den.

Bietako bat egiazkoa bada, balioa neurriz kanpokoa da. Bietako bat betetzean emaitza egiazkoa izateko OR eragile logikoa erabili behar da.

Baina hau ez da instrukzio hau idazteko modu bakarra. Bigarren moduan AND eragile logikoa erabiliko dugu.

```
If Val(txtPollAld)>=0 And Val(txtPollAld)<=31 Then
```

Instrukzio honek egiaztatzen du ea balioa 0 edo handiagoa den, eta (*AND*) batera, 31 edo txikiagoa den. Bi baldintzak batera bete behar direnez, *AND* eragile logikoa erabiliko dugu

Lehenengo metodoa honela labur dezakegu:

```
If Baldintza Then
Errore bat gertatu da
Else Dena ongi
```

Eta bigarren metodoa:

```
If Baldintza Then
Dena ongi
Else Errore bat gertatu da
```

6.4.2. Ariketa

Egin itzazu programan behar diren aldaketak aldagaien izena egiaztatzeaz gain aldagaiei esleituko zaizkien balioak –32768 eta 32767 bitartean egon daitezen. Erabil itzazu mezu-koadroak erabiltzaileari gertatutako errorearen berri emateko.

6.5. Aldagaien balioen irakurketa sistematikoa

6.5.1. Kontrol-egitura iteratiboak

Askotan, programa bat egiten denean badago behin eta berriro errepikatu beharreko zerbait. Hau gertatzen denean, begizta iteratibo bat erabili beharko dugu (iteratiboak behin eta berriro errepikatzen dela esan nahi du).

Begizta iteratibo baten adibidea While ... Wend da:

```
Dim i As Integer
i= 0
While i < 10
      Text1 = Str(i)+””
```

```
      i = i+1
Wend
```

Adibide honen hasieran **i** aldagaiari (zenbaki osoa) 0 balioa esleitu zaio. Programa *While* instrukzioa aurkitzen duenean baldintza (**i**<10) egiazkoa ala faltsua den egiaztatzen du. Fase honetan egiazkoa da, eta **i** aldagaiari unitate bat gehituko dio. *Wend* instrukzioa errepikatu behar den kodea bertan bukatzen dela adierazten du. Puntu honetan, programak atzera egiten du *While* instrukzioraino, eta berriro egiaztatzen du baldintza. Egiazkoa bada (bigarren aldi honetan bada: **i**=1), unitate bat gehituko dio **i**-ri. Prozesu hau **i**-ren balioa 10 izan arte errepikatuko da, une horretan baldintza beteko ez delako. Horren ondorioz, begizta ez da berriz egikarituko, eta programak *Wend* ondorengo instrukzioari ekingo dio (begizta 10 aldiz egikaritu ondoren).

Begizta bat sortzeko beste era bat For .... Next kontrol-egitura erabilitzea da. Beharbada, egitura hau erabiliz argiago gelditukо da begizta zenbat alditan errepikatuko den. Ikus dezagun adibide baten bitartez:

```
Dim i As Integer
For i = 0 to 10
      Text1 = Str(i)+””
Next i
```

i aldagaia zenbaki oso moduan hasieratu ondoren, programak *For ... Next* begizta bati hasiera ematen dio. Hasieran 0 balioa esleitzen zaio **i** aldagaiari, eta begizta 0-tik 10-erako balioekin egikarituko da. Koskatutako lerroak **i**-ren balioa idazten du eta *Next i* unitate bat gehitzen dio i-ri 10era iritsi arte. Honela begizta osatuko da, eta **i**-ren balioa 10 izango da.

Visual Basic-en beste bi begizta mota daude: Do ... While ... Loop eta Do ... Loop ... Until. Azter ditzagun adibideen bitartez.

```
Dim i As Integer
i= 0
While i < 10
      Text1 = Str(i)+””
      i = i+1
Wend
```

```
Dim i As Integer
i= 0
Do
      Text1 = Str(i)+””
      i = i+1
Loop While i < 10
```

Ezkerreko begiztak 9 zenbakia idatziko du bukaeran eta eskuinekoak ere, baina badago ezberdintasun bat bien artean. Bien arteko ezberdintasuna baldintza egiaztatzen den unean datza. Ezkerrekoan begiztaren hasieran egiaztatzen da, eskuinekoan bukaeran egiten den artean. Honen ondorioak beste adibide baten bitartez ikusiko ditugu. Kode zati berri honetan, **i**-ri 20 balioa esleituko diogu (aurrekoetan 0 zen).

```
Dim i As Integer
i= 20
While i < 10
      Text1 = Str(i)+””
      i = i+1
Wend
```

```
Dim i As Integer
i= 20
Do
      Text1 = Str(i)+””
      i = i+1
Loop While i < 10
```

Ezkerreko begiztan egiaztapena hasieran egiten denez, testu-laukian ez da inolako emaitzarik erakutsiko. Eskuinekoan berriz, egiaztapena begizta behin egikaritu ondoren egiten denez, 20 balioa erakutsiko da, eta **i**-ren amaierako balioa 21 izango da. Ezkerrekoan begizta egikaritu ez denez **i**-ren balioak ez du aldaketarik izango.

Orain RCXko 32 aldagaien balioak irakurriko dituen programa bat egingo dugu. Banan-banan egitea neketsua eta aspergarria litzateke, baina While ... Wend kontrol-egitura erabiliz kode labur eta argia idatzi ahal izango dugu.

- Gehi iezazkiozu ondoko kontrolak formularioari:

Kontrol mota	Propietatea	Balioa
CommandButton	Nombre Caption	cmdPollGuztiak &Guztiak irakurri
TextBox	Nombre Font Multiline ScrollBars Text	TxtAldGuztiak (aukera ezazu nahi duzun letra-tipoa) True[7] 2 - Vertical (Hutsik utzi)

6.2 taula

6.4 irudia:
formularioa osagai berriekin.

- Idatz ezazu ondoko kodea:

```
Private Sub cmdPollGuztiak_Click()
      Dim iKontadorea As Integer
      Dim strAldGuztiak As String
      Dim strOraingoLerroa As String
      Dim strLFCR As String
      StrLFCR = Chr(13)+Chr(10)
      iKontadorea = 0
      While iKontadorea <= 31
            strOraingoLerroa = Str(iKontadorea) + ":" + _
      Str(PBrickCtrl.Poll(ALD, iKontadorea))
            strAldGuztiak = strAldGuztiak + strLFCR + strOraingoLerroa
            iKontadorea = iKontadorea + 1
      Wend
      TxtAldGuztiak.Text = strAldGuztiak
End Sub
```

6.5.2. Kodearen deskribapena

Hasieran Visual Basic-eko aldagai batzuk hasieratzen dira:

- iKontadorea: While ... End begiztako errepikapenak zenbatzeko erabiltzen da.
- StrAldGuztiak: irakurritako aldagaien balioak gordeko ditu (une bakoitza arte irakurrita daudenak).
- StrOraingoLerroa: irakurri behar den aldagaiaren izena gordetzen du.

[7] Multiline propietatearen balioa True (egiazkoa) denez, testu-laukiaren portaera ComboBox-en List propietateak duenarenren antzekoa izango da.

- StrLFCR: lerro aldaketa sartzen du balioak zerrenda moduan aurkezteko.

Ikusiko duzun bezala, *txtAldGuztiak* lerro batzuk beteko dituen testu-kate luze bat erakusten du. Testu-katea lerroz lerro banatzeko **strLFCR** testu-katea sartu dugu aldagaien artean. **strLFCR** katea hurrengo aldagaia lerro berri batean jartzeko erabiltzen da.

```
StrLFCR = Chr(13)+Chr(10)
```

Chr(13) lerro aldaketa karakterea da, eta *Chr(10)* line feed karakterea.

While ... Wend begiztak 0 balioarekin hasi behar du, eta 31rekin bukatu. Hau, begizta hasi aurretik **iKontadorea** aldagaiari 0 balioa esleituz lortzen da, eta baldintza egiazkoa izango da kontadorearen balioa 31 edo txikiagoa den artean.

Azter dezagun ondoko instrukzioa:

```
strLineaActual = Str(iKontadorea) + ":" + Str(PBrickCtrl.Poll(ALD,_
iKontadorea))
```

Kode lerro honen bitartez, **strOraingoLerroa** aldagaiari irakurtzen ari garen aldagaiaren izena eta balioa esleituko dizkiogu. Testu-kateen batuketa honen batugaiak aldagaiaren izena, bi puntu eta aldagaiaren balioa dira.

Ikus dezagun orain hurrengo lerroa:

```
strAldGuztiak = strAldGuztiak + strLFCR + strOraingoLerroa
```

strAldGuztiak testu-kateari beste bi testu kate gehitu dizkiogu: **strLFCR** katea hurrengo aldagaia lerro berri batean erakusteko, eta **strOraingoLerroa**. Begiztaren bukaeran aldagai honek RCXko aldagai guztien balioak pilatuta izango ditu testu-kate batean.

Prozeduraren bukaeran **strAldGuztiak** aldagaiaren balioa **txtAldGuztiak** testu-laukiari esleituko zaio.

6.5.3. Aldagai baten aldaketen detekzioa

Bosgarren kapituluan sentsoreen balioak irakurtzeko Timer kontrola nola erabili ikusi genuen. Active-X kontrolak irakurketa hau era automatikoan egin dezake (RCXko aldagaien aldaketak kontrolatzeko).

- Jar ezazu komando-botoi berri bat formularioan, eta eman iezaiozu **cmdAutoIrakur** izena. *Caption* propietatearen balioa **Auto&Irakurketa** izango da.
- Idatz ezazu ondorengo kodea:

```
Private Sub cmdAutoIrakur_Click()
      'Irakurketa automatikoaren konfigurazioa
      PBrickCtrl.SetEvent ALD,6,MS_200
End Sub
```

Kode honek 6 izeneko aldagaiaren irakurketa automatikoari hasiera ematen dio, 200 milisegundoko denbora-tarteka egingo dena. Baina, honez gain, aldaketaren berri emango digun beste zerbait behar dugu, hain zuzen ere, mezu-koadro bat.

Código bistan zaudela, eman itzazu ondoko urratsak:

- Hauta ezazu PbrickCtrl *Código* leihoko goiko ezkerraldean dagoen *Objeto* zerrendan (honek ezabatu behar den eredu bat sortuko du).
- Hauta ezazu *VariableChange* leiho bereko goiko eskuin aldean dagoen *Procedimiento* zerrendan.

- Idatz ezazu ondoko kodea sortu berria den ereduaren barruan.

```
Private Sub PBrickCtrl_VariableChange(ByVal Number As Integer, ByVal_
Value As Integer)
      'irakurritako datua mezu-koadro batean erakutsiko du
      MsgBox Str(Value), vbInformation, str(Number) + " aldagaia aldatu_
      da"
End Sub
```

6 izeneko aldagaian aldaketaren bat gertatzen bada, *PbrickCtrl_VariableChange* gertaera aplikaziora bidaltzen da. Hortik aurrera bakoitzak erabaki behar du zer egin nahi duen. Gure adibidean, mezu bat bidaliko dugu monitorera erabiltzaileak aldagai hori aldatua izan dela jakin dezan.

- Egikari ezazu programa.
- Piztu ezazu RCX.
- Saka ezazu *AutoIrakurketa*[8] botoia.
- Eman iezaiozu 333 balioa 6 izeneko aldagaiari.

Mezu-koadro bat agertuko da aldaketaren berri emanez.

6.6. Ariketa

Jar ezazu botoi berri bat formularioan aldagai guztiei 0 balioa emateko.

[8] Aldagaiaren balioa (kasu honetan 6 izenekoarena) 0 ez bada, mezu-koadroa *AutoIrakurketa* botoia sakatu bezain laster agertuko da. Hala gertatu ezkero, Saka ezazu *Aceptar* botoia eta segi aurrera.

7 Robot autonomoak

Orain arte egin ditugun programak zuzenean ordenagailuan egikaritu dira denbora errealean (adibidez, motoreen kontrola formulario baten bitartez). Metodo honi *berehalako kontrola* deritzo. Kapitulu honetan beste metodo bat ikusiko dugu, hain zuzen ere, programak RCXra transferitzeko aukera emango diguna. Honela, RCXk programak egikaritu ahal izango ditu nahiz eta IR dorretik urruti egon. Ordenagailutik transferitutako atazak bestelako instrukziorik jaso gabe egikaritzen direnean, robotak era autonomoan funtzionatzen duela esaten da.

Kapitulu honetan eta ondorengoetan A eranskinean deskribatzen den robot oinarria erabiliko dugu. Bete behar duen funtzioren arabera sentsore bat edo beste erantsiko diogu. Aurrera jarraitu aurretik munta ezazu robot oinarria.

7.1. Kapitulu honetako edukiak

1. Robot autonomoen programazioa: programen egitura nagusia.
2. Programen transferentzia RCXra. Transferentzia-erroreen kontrola.
3. Kontrol egiturak: egitura iteratiboak eta baldintzazkoak.
4. Roboten zentzumen-aparatua eta motore-aparatua lotu: zentzumen-aparatuaren irakurketen arabera erabaki bat edo beste hartu.

7.2. Programa baten egitura

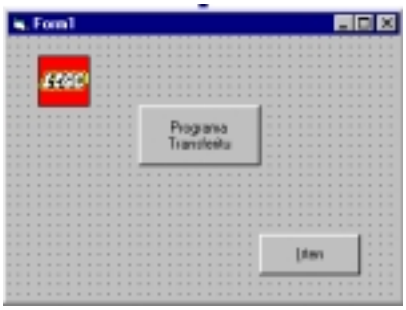
RCXk bost programa slot dauzka. Slot bakoitzean 8 subrutina eta 10 ataza gorde ditzake. Atazak batera egikari daitezkeen kode zatiak dira. Kapitulu honetan alde batetik bestera oztopoak ekiditen mugituko den robot bat egingo dugu.

7.2.1. Proiektu proposamena

Lehen proiektu honen helburua oso sinplea da. Aurrera hiru segundoz mugitu ondoren beste hiru segundoz atzera egingo duen robot bat muntatu eta programatuko dugu, bukaeran hasierako toki berean geldi dadin.

7.2.2. Programaren edizioa

- Abiaraz ezazu Visual Basic.
- Hauta ezazu Lego ikonoa *Nuevo Proyecto* leihoan, eta saka ezazu *Aceptar* botoia.
- Aurreko kapituluetan egindako era berean gorde itzazu fitxategiak **Jaitsi** izenarekin. Horretarako sor ezazu C:\VBLego\Kap07 direktorioa zure fitxategiak bertan gordetzeko.
- Egin ezazu formularioa 7.1 taulako datuetan oinarrituz.



7.1 irudia:
hau izango da gure proiektuaren formularioa

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	frmTransferitu Transferitu programa
CommandButton	Nombre Caption	cmdProgrJaitsi &Transferitu Programa
CommandButton	Nombre Caption	cmdIrten &Irten

7.1 taula

Kodea hauxe izango da::

```
Private Sub cmdProgrJaitsi_Click()

    With PBrickCtrl

        .SelectPrgm 2 'edo SelectPrgm PRGM_3 konstantea

        .BeginOfTask 0

            .On A_MOTOREA+ B_MOTOREA
            .SetFwd A_MOTOREA+ B_MOTOREA
            .Wait 2, SEG_3
            .SetRwd A_MOTOREA+ B_MOTOREA
            .Wait 2, SEG_3
            .Off A_MOTOREA+ B_MOTOREA

        .EndOfTask

    End With

End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm
End Sub

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub
```

7.2.3. Proiektuaren egikaritzea

- Gorde ezazu proiektua.

- Piztu ezazu RCX.
- Egikari ezazu programa.
- Saka ezazu *Transferitu Programa* botoia (edo <Ctrl+T>).
- RCXren LCDan 3 zenbakia agertuko da, aukeratutako programa 3 dela adieraziz.
- Saka ezazu RCXko Run botoia.

Robota hiru segundoz mugituko da aurrera, beste hiru segundoz atzera eta hasierako tokian gelditukо da.

7.2.4. Kodearen deskribapena

Azter dezagun programa hau urratsez urrats. PbrickCtrl erreferentzia behin eta berriro erabili behar ez izateko hasieran *With ... End With* instrukzioa erabiliko dugu.

```
With PBrickCtrl
```

Hortik aurrera ez dugu berriro PBrickCtrl idatzi beharko.

.SelectPrgm 2 : Programa gordeko den slot-a aukeratzeko erabiltzen da. Instrukzioaren argumentua 0 eta 4 bitarteko zenbaki bat da (0 lehen programa slot-i dagokio eta 4 bosgarrenari) Zenbakien ordez RCXdatuak.bas moduluan definitutako konstanteak erabil daitezke.

.BeginOfTask 0 : instrukzio honek ataza nagusiari hasiera ematen dio. RCXn programak atazez osaturik daude, eta guztiek ataza nagusi bat behar dute, 0 izenez ezagutzen dena (RCXdatuak.bas modulua erabiltzen baduzu NAGUSIA konstantea idatz dezakezu 0-ren ordez). Hau izango da RCXko **Run** botoia sakatzean egikarituko dena. Atazen egitura hau izango da:

```
.BeginOfTask 0
.1 instrukzioa
.2 instrukzioa
...
.n instrukzioa
.EndOfTask
```

.BeginOfTask eta .EndOfTask artean dagoen kodeak **Run** botoia sakatzean zer gertatuko den deskribatzen du. Bi instrukzio hauen artean dagoen kodea RCXra transferituko da, eta ez besterik.

1 eta 3 motoreak martxan jarri ondoren, programaren egikaritzea hiru segundoz gelditukо da (Wait). Motoreen biraketa noranzkoa aldatu eta programaren egikaritzea beste hiru segundoz gelditukо da. Bukatzeko motoreak gelditukо ditu eta programa amaituko da. Hau guztia ahal duten potentzia handienaz egingo dute motoreek.

7.3. Erroreen antzematea

Gure programari gerta daitezkeen erroreak antzemateko sistema gehituko diogu. Orain arte, egia esan baikortasunez, RCXra transferitutako aginduak behar bezala jasoak izan direla suposatu dugu. Ikus dezagun zer gertatzen den hala ez bada.

RCXra transferentzia bukatu bezain laster, ActiveX kontrolak DownloadDone gertaera bidaltzen du. Hau ere gertatuko da transferentzia bukatu aurretik erroreren bat gertatzen bada. Gertaera formulariotik kanpo egongo da.

```
PBrickCtrl_DownloadDone(ByVal ErrorCode As Integer, ByVal DownloadNo As_
Integer)
```

ErrorCode-ren balioa 0 bada, transferentzia era arrakastatsuan egin dela esan nahi du. Baina balioa 1 bada, transferentzian erroreren bat gertatu dela esan nahi du, eta *DownloadNo* aldagaiak errorea zein ataza edo subrutinari dagokion adieraziko digu.

7.3.1. Kodearen edizioa

- Hauta ezazu PbrickCtrl *Código* leihoko goiko ezkerraldean dagoen *Objeto* zerrendan (honek ezabatu behar den eredu bat sortuko du).
- Hauta ezazu *DownloadDone* leiho berean goiko eskuin aldean dagoen *Procedimiento* zerrendan.
- Idatz ezazu ondoko kodea sortu berria den ereduaren barruan:

```
Private Sub PBrickCtrl_DownloadDone(ByVal ErrorCode As Integer, ByVal _
DownloadNo As Integer)
      If ErrorCode=0 Then           'Transferentzia arrakastatsua
            PBrickCtrl.PlaySystemSound SWEEP_DOWN_SOINUA
            MsgBox "Transferentzia
arrakastatsua",vbInformation,"Status"
      Else                          'Erroreak transferentzian
            MsgBox "Erroreak transferentzian",vbCritical,"Status"
      End If
End Sub
```

7.3.2. Programaren egikaritzea

- Gorde eta egikari ezazu proiektua.
- Itzal ezazu RCX.
- Saka ezazu *Programa Transferitu* botoia.

Segundo batzuen ondoren, transferentzia behar bezala ez dela egin jakinarazten duen errore--mezua agertuko da.

- Egin ezazu klik *Aceptar* botoian mezu-koadroa ixteko.
- Orain, piztu ezazu RCX.
- Saka ezazu berriro *Transferitu Programa* botoia.

Transferentzia arrakastatsua bada, RCXk SWEEP_DOWN_SOINUA joko du, eta mezu-koadroa agertuko da transferentzia behar bezala egin dela jakin dezagun.

7.4. Kontrol-egiturak

RCXn kontrol-egiturak Visual Basic-en bezala erabiltzen dira. Oinarrizkoak hiru dira:

.Loop

.While

.If ... Else

7.4.1. Loop

Loop kontrol egitura Loop eta EndLoop dauden instrukzioak hainbat aldietan errepikatuko ditu.

```
PBrickCtrl.Loop KONST, 4
      PBrickCtrl.PlaySystemSound BEEP_SOINUA
PBrickCtrl.EndLoop
```

Kontrol egitura honen lehenengo zatiak (Loop) egitura zenbat alditan errepikatu behar den adierazten du. Kasu honetan konstante batek definituko du zenbat alditan errepikatu behar den, hain zuzen, lau alditan. Egitura honek era esplizituan adierazten du zenbat errepikapen gertatu behar diren, beraz, argiagoa da beste batzuk baino. Dena dela, baditu bere alde txarrak ere, beste egitura batzuetan errepikapenak kontrolatzen dituen aldagaia begiztaren barruan erabil daitekeelako.

EndLoop metodoak unitate batean gutxiagotzen kontadorearen balioa (kasu honetako hasieran lau dena), eta egiaztatzen du ea emaitza zero den. Hala bada, begizta bukatuko da eta hurrengo instrukzio lerroa egikarituko da, bestela, begiztako instrukzioak berriro egikarituko dira.

Adibideko kodeak BEEP_SOINUA lau aldiz joko du.

Egitura honen kasu berezia Loop KONST,BETI (Loop 2,0) da. Begiztaren hasiera honelakoa denean, mugagabeko begizta izango da. Hau datu iturria konstante bat denean gertatzen da, baina aldagai bada (adibidez Loop ALD,5) eta aldagaiaren balioa 0 bada, begizta ez da egikarituko, eta programak begiztaren ondoko instrukzioarekin jarraituko du.

7.4.2. While

While ... EndWhile kontrol egitura Visual Basic-eko Do While ... Loop kontrol egituraren antzekoa da. Bere sintaxia ondokoa da:

While (Source1, Number1, RelOp, Source2, Number2)

Lehenengo bi argumentuak konparaketaren lehen zatiari dagozkio, eta azkeneko biak bigarrenari. *RelOp*-ek balioak nola konparatu behar diren deskribatzen du. Konparaketak lau era ezberdinetan egin daitezke:

Zenbakia	Konstantea	Deskribapena
0	HANDI_BAINO	Handiagoa baino
1	TXIKI_BAINO	Txikiagoa baino
2	BERDIN	Berdin
3	EZBERD	Ezberdin

7.2 taula

```
With PBrickCtrl
      .SetVar 6,KONST,1
      .While SENBALIO,SENTSOREA_1,BERDIN,ALD,6
            .PlaySystemSound BEEP_SOINUA
            .Wait KONST,MS_500
      .EndWhile
End With
```

Aurreko kodeko bigarren lerroan RCXko 6 izeneko aldagaiari 1 balioa esleitu zaio. While kontrol egiturako konparaketako lehen balioa 1 sentsorearen irakurketa da, eta bigarrena, 6 aldagaian gordetako balioa (kasu honetan 1). Beraz sentsorearen irakurketa 1 den bitartean, RCXk BEEP_SOINUA joko du segundo erdiro.

7.4.3. If ... Else

If ... [Else] ... EndIf kontrol egiturak bi balio konparatzen ditu, While egiturak egiten duen bezala.

If (Source1, Number1, RelOp, Source2, Number2)

Baldintza egiazkoa bada, *If* instrukzioaren ondoren dauden aginduak egikarituko dira, eta betetzen ez bada, *Else*-ren ondorengoak. Hala ere, *Else* (bestela) instrukzioa aukerakoa da, eta begizta aukera hori eman gabe buka daiteke.

```
With PBrickCtrl
      .SetVar 6, KONST, 800
      .If SENBALIO, SENTSOREA_3, TXIKI_BAINO, ALD, 6
            .On A_MOTOREA
      .Else
            .On B_MOTOREA
      .EndIf
End With
```

7.5. Oztopoak ekiditen dituen robota

7.5.1. Proiektu proposamena

Egin dezagun orain oztopoak ekiditeko gai den robot bat. Kapitulu honetako hasieran muntatutako robot oinarriari ukitze sentsorea erantsiko diogu (ikus instrukzioak A eranskinean).

Programa hau era ezberdinetan egin daiteke. Batzuk besteak baino hobeak izango dira, eta hemen horietako bi aztertuko ditugu. Lehenengoa idazteko ondoko algoritmoa erabiliko dugu:

```
Robotak zerbait ukitzen badu hau egingo du:
      - atzera egin segundo batez
      - alde batera biratu
Ukitzen ez duenean aurrera egingo du
```

7.5.2. Programaren edizioa

- Jar ezazu komando-botoi berri bat formularioan, eta eman iezaiozu **cmdUkitu** izena.
- Eman iezaiozu **Ukitu &Programa** balioa *Caption* propietateari.
- Idatz ezazu ondoko kodea:

```
Private Sub cmdUkitu_Click()
      With PBrickCtrl
            .SelectPrgm PRGM_4 ‘ 4. programa-slot-ean gordeko da.
            .BeginOfTask NAGUSIA
                  .SetSensorType SENTSOREA_1, UKITZE_SENTS
                  .SetPower A_MOTOREA + C_MOTOREA, KONST, 3
                  .Loop KONST, BETI
                        .If SENBALIO, SENTSOREA_1, BERDIN, KONST, 1
                        ' Sentsorea sakatua?
                              .SetRwd A_MOTOREA + C_MOTOREA
                              .Wait KONST, SEG_1       ' atzera
                              .Off C_MOTOREA
                              .Wait KONST, SEG_1       ' biraketa
                              .Off A_MOTOREA
```

```
                        .Else
                              .SetFwd A_MOTOREA + C_MOTOREA
                              .On A_MOTOREA + C_MOTOREA      ' aurrera
                        .EndIf
                  .EndLoop
            .EndOfTask
      End With
End Sub
```

- Gorde ezazu proiektua.
- Piztu ezazu RCX.
- Egikari ezazu proiektua.
- Transferi ezazu programa RCXra **Ukitu Programa** botoia sakatuz.
- Jar ezazu robota lurrean eta saka ezazu RCXko **Run** botoia.

Ikusiko duzunez, robotak talka egiten duenean atzera egiten du eta biratzen du oztopoa ekiditeko.

7.5.3. Kodearen deskribapena

Hasteko, RCXko 4. programa-slot-a aukeratzen du programa gordetzeko.

Ataza nagusiko lehenengo bi lerroetan (BeginOfTask NAGUSIA lerroaren ondoko biak) ukitze sentsorea konfiguratu eta motoreek eman behar duten potentzia finkatzen da.

Ondoren amaierarik gabeko begizta bati hasiera ematen zaio:

```
Loop KONST, BETI
```

Begizta honek errepikatzen duena hauxe da:

```
Ukitze-sentsorea presionatua badago
      Desplazamenduaren noranzkoa segundu batez aldatuko du eta beste
      segundu batez biratu
Bestela
      Aurrera.
```

Lur motak eragina du robotak mugitzeko duen eran, hain zuzen, baldosen gainean edo alfonbra baten gainean mugitzea berdina ez baita. Horregatik, beharbada programaren egikaritzea gelditzen den denbora (Wait) egokitu beharko da.

7.6. Ariketak

1. Alda ezazu aurreko programan sentsorea Raw modura, eta egin itzazu *If* egituraren baldintzan behar diren aldaketak. Esperimentazioaren bitartez ikus dezakezu zein irakurketa ematen duen ukitze-sentsoreak presionatua dagoenean eta ez dagoenean (edo erdisakatua dagoenean).
2. Aurreko programa aztertu ezkero, antzeman daiteke, sentsorea presionatua ez dagoenean, programak motoreei eskatzen diela behin eta berriro martxan jartzeko. Egin itzazu behar diren zuzenketak motorea atazaren hasieran martxan jartzeko, eta bakarrik eska dezan aurrera egiteko oztopoa ekidin ondoren.
3. Aurreko aldean sentsore bakar baten ordez, bi jarriko dituzu talkak ezker aldean edo eskuin aldean gertatzen diren jakiteko. Ezkerrekoaz talka egiten badu, robota eskuin aldera biratuko da, eta eskuinekoaz egiten badu ezker aldera.

8 Marra jarraitzailea

Kapitulu honetan egingo ditugun roboten zentzumen-aparatuaren oinarria argi-sentsoreak izango dira. Robot hauekin egiten den esperimentazioan inguruko argia eragina izan dezake, beraz programaren atalaren bat egokitu beharko da.

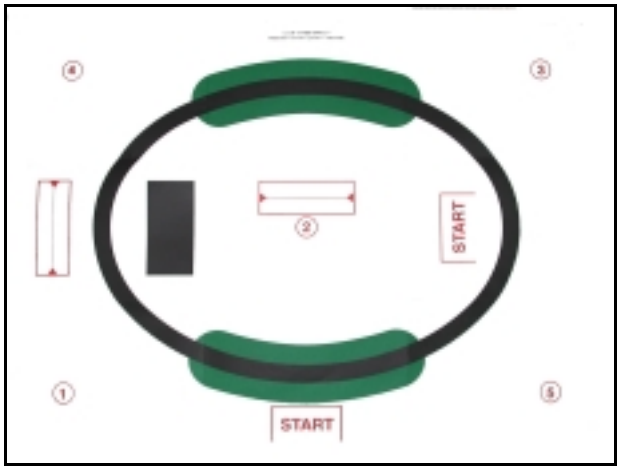
8.1. Kapitulu honetako edukiak

1. RCXren programazioan sakontzea.
2. Argi-sentsorea hurbiltasun detektagailu gisa erabiltzea: argi infragorriaren igorpena eta detekzioa.

8.2. Marra jarraitzailea

8.2.1. Proiektu proposamena

Proiektu klasiko bat garatzen hasiko gara: lurrean marraztutako marra beltz bat jarraitzeko gai den robot bat egingo dugu. Honelako robotekin lan egiteko, atzealde zuri baten gainean dauden marra beltzak dituzten zirkuituekin lan egitea gomendatzen da (beltz asea behar da, eta ez koloretako arkatzez lor daitekeena). MindStorms kit-ak dakarren posterrean obalo baten itxurako marra beltz bat dago (ikus 8.1 irudia), eta oso egokia da lanean hasteko.



8.1 irudia
LEGO MindStorms “Test Pad”

Proiektu hau egiteko argi-sentsore bat edo bi erabil daitezke. Edonork pentsatuko du errazagoa dela bi sentsore erabiliz, horrela marratik zein aldetik atera den segituan jakingo dugulako. Hala ere, programa egoki batekin sentsore bakar bat nahikoa izan daiteke, eta estrategia ezberdinak garatzeko oso proposamen interesgarria da.

Erabiliko dugun robotak aurreko kapituluan erabilitako oinarri bera izango du, baina argi-sentsoreari dagokion modulua erabiliko dugu (ikus A eranskina). Argi sentsorea 3 sarreran konektatu behar da, bestela programak ez du behar bezala funtzionatuko.

Programa diseinatzen hasteko robota obaloaren inguruan erlojuaren orratzen noranzkoa mugituko dela erabakiko dugu. Honela, marratik beti ezker aldetik aterako da, eta honen ondorioz,

bidera itzultzeko beti eskuin aldera biratu beharko du. Behin erabaki hau hartuta, programa egiteak ez du zailtasun handirik:

> Robotak aurrera egingo du sentsoreak marra beltzaren gainean dagoela esaten digun artean.
>
> Marratik ateratzean eskuineko motorea gelditukо du eta denbora tarte batez (zenbatekoa izan behar duen esperimentazioak esango digu) ezkerreko motorea izango da martxan egongo den bakarra (hau da, eskuin aldera biratuko du).

8.2.2. Kodearen edizioa

Robota muntatu ondoren, aurreko algoritmoa kode bihurtuko dugu:

- Ireki ezazu aurreko kapituluan egindako proiektua.
- Jar ezazu botoi berri bat formularioan eta eman iezaiozu **cmdMarra** izena.
- Eman iezaiozu **&Marra Programa** balioa *Caption* propietateari.
- Idatz ezazu ondoko kodea:

```
Private Sub cmdMarra_Click()
      With PBrickCtrl
            .SetSensorType SENTSOREA_3, ARGI_SENTS
      End With
End Sub
```

Kode honen bitartez marraren gainean eta marratik kanpo argi-sentsoreak egiten dituen irakurketak ezagutu ahal izango ditugu. 3 sarreran konektatutako sentsorea portzentajezko moduan konfiguratutako argi-sentsorea izango da.

- Gorde eta egikari ezazu programa.
- Piztu ezazu RCX.
- Saka ezazu *Marra Programa* botoia RCXko 3 irteerako sentsorea konfiguratzeko
- RCXko View botoiaren bitartez hauta ezazu 3 sentsorea bere irakurketa ikusteko (LCDko gezia 3 sentsoreari begira egon behar du 8.2 irudian ikusten den bezala).

8.2 irudia
RCXko LCDan gezia 3 sentsoreari begira

- Mugi ezazu argi-sentsorea posterraren alde zuri eta beltzen gainean zuriari eta beltzari dagozkion irakurketak ezagutzeko. Mugi ezazu sentsorea robotean muntatuta dagoenean mugituko den modu berean.
- Idatz ezazu ondoko kodea:

```
Private Sub cmdMarra_Click()

      Const ARKU_DENBORA = 5        ' 5 aldagaiaren izena
      Const ARGI_MUGA = 6           ' 6 aldagaiaren izena
```

```
        With PBrickCtrl
              .SelectPrgm PRGM_4
              .BeginOfTask NAGUSIA
                    .SetVar ARKU_DENBORA, KONST, MS_50
                    .SetVar ARGI_MUGA, KONST, XX 'Sartu hemen zure balioa
                    .SetSensorType SENTSOREA_3, ARGI_SENTS
                    .SetPower A_MOTOREA + C_MOTOREA, KONST, 6
                    .On A_MOTOREA + C_MOTOREA
                    .Loop KONST, BETI
                          .While SENBALIO, SENTSOREA_3, HANDI_BAINO, ALD,_
                                ARGI_MUGA
                                .Off C_MOTOREA
                                .Wait ALD, ARKU_DENBORA
                          .EndWhile
                          .On C_MOTOREA
                    .EndLoop
              .EndOfTask
        End With
End Sub
```

- Gorde eta egikari ezazu zure proiektua.
- Transferi ezazu programa RCXra
- Jar ezazu robota posterraren gainean. Robota sentsorea marra beltzaren gainean eta erlojuaren orratzen noranzkoan mugitzeko moduan jarri behar da.
- Saka ezazu RCXko **Run** botoia.

Robota posterreko marra beltzari jarraituz mugituko da.

8.2.3. Kodearen deskribapena

Programaren hasieran bi aldagaiei izen propioa emango diegu programa irakurgarriagoa izan dadin. 5 aldagaiari ARKU_DENBORA izena emango diogu, eta 6 aldagaiari ARGI_MUGA izena. ARKU_DENBORA aldagaiak biratzeko garaian C motorea zenbat denboraz gelditukо den definitzen du (denbora honetan, marra beltzera itzultzeko arku baten moduko ibilbidea deskribatuko du). ARGI_MUGA aldagaiak marra beltzak eta atzealde zuriak islatzen duten argi-intentsitateen arteko muga definitzen du.

RCXra transferituko den zatiaren hasieran aurreko bi aldagaiei balio egokia esleitzen zaie. Aldagaiak konstanteen ordez erabiltzeko arrazoia lana erraztea da, era honetan, ARGI_MUGA edo ARKU_DENBORA-n aldaketaren bat egin behar badugu, lerro bakar batean egin beharko dugu, eta ez, balio hori erabiltzen den toki guztietan.

Sentsorea eta motoreen potentzia konfiguratu ondoren bi motoreak martxan jarri eta amaigabeko begizta batean (*Loop*) sartzen da programa. Argi-sentsoreak robota marratik kanpo dagoela detektatzen badu, programak C motorea gelditukо du, eta denbora-tarte bat (ARKU_DENBORA) itxaron ondoren, berriro egiaztatuko du ea marratik kanpo dagoen. Hau robota berriro marraren gainean egon arte egingo du (*While* begizta), eta une horretan, C motorea martxan jarriko du.

8.2.4. Ariketak

1. Egin berria dugun proiektuko robotak bakarrik jarrai dezake marra erlojuaren orratzen noranzkoan. Saia zaitez behar diren aldaketak egiten robota marra edozein noranzkoan jarraitzeko gai izan dadin.

Iradokizuna: bila ezazu marra alde batean, eta ez baduzu aurkitzen beste aldean dagoela esan nahiko du. Beste estrategia bat marraren mugaren gainetik joatea da (irakurketa tarteko balio bat izango da).

2. Aurrekoan egin dituzun aldaketak softwarean izan dira. Oraingo honetan softwarea eta hardwarea aldatu behar dituzu. Egin ezazu marra jarraituko duen robot berri bat, baina bi argi-sentsorekin.

3. Egin ezazu obalotik atera gabe alde batetik bestera mugituko den robot bat.

8.3. Hurbiltasun robota

Oztopoak ekiditeko ukitze-sentsore bat erabiltzea nahiko sistema primitiboa da. Hobe litzateke robotak oztopoak detektatzea talka egin aurretik.

Hau egiteko argi-sentsorea eta RCXren ezaugarriak ondo ezagutzea beharrezkoa da. Batetik argi-sentsoreak argi infragorriarekiko nolabaiteko sentikortasuna du, eta bestetik, RCXk infragorriren igorgailua da (dakizunez, ordenagailua eta RCXren arteko komunikazioak argi infragorriren bitartez egiten da). Beraz, RCXk argi infragorria igor dezake denbora-tarte berdinetan *SendPBMessage* erabiliz, eta argi-sentsoreak irakurketetan gertatzen diren gorabeherak erabil ditzake oztopo batetik gertu dagoen jakiteko.

- Egin itzazu robotean A eranskinean jasotzen diren aldaketak.
- Jar ezazu aurreko formularioan komando-botoi berri bat eta eman iezaiozu **cmdHurbil** izena. *Caption* propietatearen balioa **&Hurbiltasun Programa** izango da.
- Idatz ezazu ondoko kodea:

```
Private Sub cmdHurbil_Click()

      Const AZKEN_IRAK = 10
      Const GORABEHERA = 11

      With PBrickCtrl
            .SelectPrgm PRGM_5

            .BeginOfTask NAGUSIA
                  .SetVar GORABEHERA, KONST, 100
                  .StartTask ATAZA_1
                  .StartTask ATAZA_2
            .EndOfTask

            .BeginOfTask ATAZA_1
                  .Loop KONST, BETI
                        .SendPBMessage KONST, 0
                        .Wait KONST, MS_10
                  .EndLoop
            .EndOfTask

            .BeginOfTask ATAZA_2
                  .SetSensorType 2, ARGI_SENTS
                  .SetSensorMode 2, MODU_RAW, 0
                  .SetFwd A_MOTOREA + C_MOTOREA
                  .On A_MOTOREA + C_MOTOREA
                  .Loop KONST, BETI
                        .SetVar AZKEN_IRAK, SENBALIO, SENTSOREA_3
                        .SumVar AZKEN_IRAK, ALD, GORABEHERA
```

```
                        .If SENBALIO, SENTSOREA_3, HANDI_BAINO, ALD,_
                        AZKEN_IRAK
                              ' Oztopo bat antzeman du
                              ' Robotak oztopoa ekidin behar du
                              ' eta berriz aurrera egin
                        .EndIf
                  .EndLoop
            .EndOfTask

      End With
End Sub
```

- Gorde eta egikari ezazu proiektua.
- Transferi ezazu **Hurbiltasun programa** robotera.
- Egikari ezazu programa.

Robota oztopo batera hurbiltzen denean, atzera egingo du, eta ekiditen saiatuko da.

8.3.1. Kodearen deskribapena

Prozeduraren hasieran bi aldagaiei izen propioa emango diegu, programa irakurgarriagoa izan dadin.

```
Const AZKEN_IRAK = 10

Const GORABEHERA = 11
```

Ataza nagusian balio zehatz bat emango diogu GORABEHERA aldagaiari. Zenbat eta txikiagoa izan balioa hau, hainbat eta sentikortasun handiagoa izango du robotak.

Programa honetan, ataza bat baino gehiago erabiliko dugu. RCXko **Run** botoiak ataza nagusiari ematen dio hasiera bakarrik, beraz, ataza nagusian gainerakoei hasiera eman beharko diegu. Bi ataza batera egikarituko dira: lehenengoak (ATAZA_1) aldian-aldian seinale infragorriak igorriko ditu, eta bigarrenak oztopoak ekiditeko erabakiak hartuko ditu.

```
.BeginOfTask ATAZA_1
      .Loop KONST, BETI
            .SendPBMessage KONST, 0
            .Wait KONST, MS_10
      .EndLoop
.EndOfTask
```

Ataza honen funtzioa aldian-aldian seinale infragorria igortzea da, 10 ms-ko denbora-tarteak utzita. Horretarako *SendPBMessage* metodoa erabili dugu.

Bigarren atazaren hasieran (ATAZA_2) sentsore mota eta modua (Raw[9] modua) finkatzen dira. Bi motoreak martxan jarri ondoren amaigabeko begizta bati hasiera ematen dio:

```
            .Loop KONST, BETI
                  .SetVar AZKEN_IRAK, SENBALIO, SENTSOREA_3
                  .SumVar AZKEN_IRAK, ALD, GORABEHERA
                  .If SENBALIO, SENTSOREA_3, HANDI_BAINO, ALD,
                  AZKEN_IRAK
                        ' Oztopo bat antzeman du
                        ' Robotak oztopoa ekidin behar du
                        ' eta berriz aurrera egin
                  .EndIf
```

[9] Raw moduak (0 … 1023) portzentajezko moduak (0 ... 100) baino sentikortasun handiagoa du, hau da zehaztasun handiagoa.

```
.EndLoop
```

Sentsorearen oraingo irakurketa AZKEN_IRAK aldagaiari esleituko zaio. Ondoren, GORABEHERA aldagaiaren balioa (100 adibide honetan) gehituko diogu. Uneren batean sentsorearen irakurketak AZKEN_IRAK aldagaiaren balioa gainditzen badu, argi-sentsoretik gertu oztoporen bat dagoela esan nahi du (argi infragorria islatzen duen zerbait dago).

8.3.2. Ariketa

Idatz itzazu **Hurbiltasun** programari falta zaizkion lerroak robotak oztopoak ekidin ditzan.

9 Datuen erregistroa

9.1. Kapitulu honetako edukiak

Seigarren kapituluan RCXk 32 aldagai erabil ditzakeela ikasi dugu. Beharbada, egiten ditugun proiektu gehienetan ez dugu besterik beharko, baina batzuetan, datu kopuru handiagoa gorde nahi izango dugu, adibidez, robot mugikor bat mugitzen denean gertatzen diren argi-intentsitatearen aldaketak erregistratu nahi baditugu, beste zerbait beharko dugu. Horretarako, RCXk "datalog" eskaintzen digu, hau da, datuen erregistroa.

Erregistratutako datuak ezin ditugu beste aldagaien moduan erabili, eta ordenagailura transferituko ditugu, ondoren, hala nahi badugu, grafiko bateen bitartez erakusteko. Datu horiek ordenagailuan gordetzeko aldagaiak beharko ditugu, baina asko izango direnez, beste aldagai mota bat erabiliko dugu, matrizeak hain zuzen.

1. Visual Basic-eko aldagaiak: matrizeak.
2. Datuen erregistroa RCXn.
3. Menuak formularioetan.
4. Tresnen koadroa: kontrol grafikoa RCXk erregistratutako datuak formularioan era grafikoan aurkezteko.

9.2. Matrizeak (array)

Matrize bat aldagai zerrenda bat baino ez da. Matrizeetan antzekoak diren datu kopuru handiak gorde daitezke, adibidez, une ezberdinetan irakurritako argi-intentsitateak. Honelako datuak gordetzeko aldagai arruntak (matrizeak ez direnak) erabiliko bagenitu, datu bakoitzari izen ezberdina eman beharko genioke, eta zail samarra litzateke halako datu multzoak kontrolatzea.

Matrize bat izen bereko aldagai zerrenda bat da. Ikus dezagun adibide baten bitartez: jakintzagai ezberdinetan ikasle talde batek izandako emaitzak gordzea izango da helburua.

```
Dim Emaitza As Integer        'ikaslearen nota
```

Instrukzio honek *Emaitza* aldagaia zenbaki oso moduan hasieratzen du. Gure adibidean ikasle baten azterketako emaitza gordetzeko balioko luke. Gelan ikasle bat baino gehiago balego, ikasle bakoitzeko aldagai bat hasieratu beharko litzateke. Luzea eta neketsua litzatekeen lan hau errazteko matrize motako aldagaiak oso erabilgarriak dira.

Matrizearen elementuak azpindizeen bitartez bereizten dira. Ezberdintasuna izenean egon beharrean (Emaitza01, Emaitza02, Emaitza03...) azpindizeetan egongo dira:

Emaitza01	Emaitza(1)
Emaitza02	Emaitza(2)
…	…

Parentesi arteko zenbakiak matrizearen azpindizeak dira. Azpindizeak ez dira matrize izenaren zati, baizik eta matrizearen elementu ezberdinak bereizteko erabiltzen diren argumentuak. Aldagaiak erabiliz azterketa bateko emaitzen batezbestekoa kalkulatzeko aldagai guztien izenak

idatzi beharko genituzke, matrizeekin Loop ... Next begizta bat erabil dezakegun bitartean. Ikus dezagun 40 ikasle dituen talde bati dagokion kodea:

Aldagaiekin:

```
iGuztira = Emaitza1 + Emaitza 2 + Emaitza 3 + … + Emaitza 40
iBatazBeste = iGuztira / 40
```

Matrizeekin:

```
For iKontadorea = 1 To 40
      iGuztira = iGuztira + Emaitza(iKontadorea)
Next iKontadorea
iBatazbeste = iGuztira / 40
```

Kasu honetan, nahiz eta 40 ikasle bakarrik izan, kodea laburragoa da matrizeak erabiltzen baditugu.

9.2.1. Matrizeen hasieratzea

```
Dim NireMatriza(10) As Integer
```

Matrize honek 11 elementu ditu [NireMatriza(0)tik NireMatriza(10)-ra]. 0 *beheko muga* eta 10 *goiko muga* dira.

Beharrezkoa bada, beheko muga matrizea hasieratzeko orduan zehatz daiteke.

```
Dim NireMatriza(10 To 20) As Integer
```

Hau 11 elementu dituen matrizea hasieratzeko beste modu bat da, beheko muga 10 eta goiko muga 20 dituena.

9.2.2. Matrize dimentsioanitza

Matrize dimentsioanitza azpindize bat baino gehiago duen matrizea da. Dimentsio bakarreko matrizea balio zerrenda bat den artean, matrize dimentsioanitzak balio taula baten antza du. Taulen artean erabiliena bi dimentsioduna da (bi azpindize dituen matrizea). Ikasleen adibidearekin jarraituz, demagun ikasle bakoitzak sei azterketa egin dituela. Datu horiek guztiak gordetzeko ondoko matrizea erabiliko dugu:

```
Dim MatrizeAnitza(1 To 40, 1 To 6) As Integer
```

Hau 40 lerro eta 6 zutabe dituen taula bat hasieratzea bezalakoa da, non lerro bakoitza ikasle bat den eta zutabe bakoitza azterketa bat.

9.3. Datalog

Datalog RCXren barruan dagoen inguru bat da. Datalog-en ondoko elementuetatik datozen irakurketak gorde daitezke:

- Tenporizadoreak.
- Aldagaiak.
- Sentsoreak
- Erlojua (denbora).

Datuak erregistratzen hasi aurretik, datalog-en tamaina definitu behar da. Hau SetDatalog(Tamaina) metodoaren bitartez egiten da. Tamainak zenbat datu gorde nahi ditugun adierazten du (datu bakoitzak 3 byte betetzen ditu).

Datalog-en datu bat gordetzeko, DatalogNext(Source,Number) metodoa erabiliko dugu.

Datu-iturria (Source)		Zenbakia (Number)	
Konstantea		Konstantea	
0	ALD	0 – 31	
1	TENPOR	0 – 3	TENP_1, TENP_2, TENP_3, TENP_4,
9	SENBALIO	0 – 2	SENTSOREA_1, SENTSOREA_2, SENTSOREA_3
14	ERLOJU	0	

9.1 taula

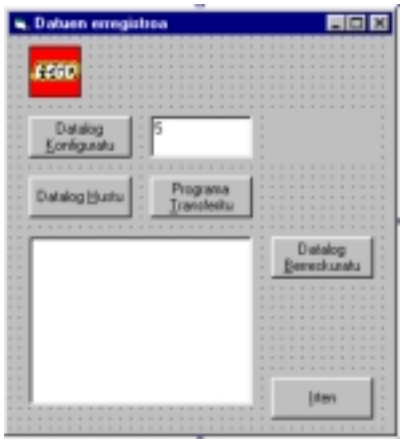
Programaren egikaritzea bukatu ondoren, informazioa RCXtik ordenagailura transferi daiteke UploadDatalog(From,Size) metodoaz.

9.4. Proiektua

- Eman iezaiozu hasiera Lego proiektu berri bati.
- Gorde ezazu **Datuen erregistroa** izena emanez.
- Egin ezazu formularioa 9.2 taulako datuetan oinarrituz.

Kontrol Mota	Propietatea	Balioa
Form	Nombre Caption	frm_Datalog Datuen erregistroa
CommandButton	Nombre Caption	cmdSetDLTamaina Datalog &Konfiguratu
CommandButton	Nombre Caption	cmdGarbituDL Datalog &Hustu
CommandButton	Nombre Caption	cmdKargaDL Datalog &Berreskuratu
CommandButton	Nombre Caption	cmdJaitsi &Programa transferitu
CommandButton	Nombre Caption	cmdIrten &Irten
TextBox	Nombre Text	TxtDLtamaina 5
Label	Nombre Caption	LblDatalog (Hutsik utzi)
List Box	Nombre	LstDatalog

9.2 taula



9.1 irudia:
datuen erregistrorako formularioa

9.4.1. Kodea

Idatz ezazu ondoko kodea:

```
Private Sub cmdKargaDL_Click()
    Dim Mat As Variant
    Dim iKontadorea As Integer
    ' Datalog Mat matrizera tranferitu
    Mat = PBrickCtrl.UploadDatalog(0, Val(TxtDLtamaina.Text) + 1)
    If IsArray(Mat) Then
        For iKontadorea = LBound(Mat, 2) To UBound(Mat, 2)
            LstDatalog.AddItem "Mota: " + Str(Mat(0, iKontadorea)) +_
             " Zbk. :" + Str(Mat(1, iKontadorea)) + "Valor :" +_
             Str(Mat(2, iKontadorea))
        Next iKontadorea
    Else
            MsgBox "Ez da baliozko matrizea"
    End If
End Sub

Private Sub cmdIrten_Click()
    PBrickCtrl.CloseComm
    End
End Sub

Private Sub cmdJaitsi_Click()
    With PBrickCtrl
        .SelectPrgm PRGM_4
        .BeginOfTask NAGUSIA
            .SetSensorType SENTSOREA_2, ARGI_SENTS
            .SetVar 10, KONST, 1234
            .Loop KONSTANTEA, 3
                .DatalogNext TENPOR, TENP_4 ‘ tenporizadorearen balioa
                .Wait KONST, SEG_1
            .EndLoop
            .DatalogNext SENBALIO, SENTSOREA_2 ‘ sentsorearen irakurketa
            .DatalogNext ALD, 10 ‘ 10 aldagaiaren balioa
            .DatalogNext TENPOR, TENP_4 ‘ tenporizadorearen balioa
        .EndOfTask
    End With
End Sub

Private Sub cmdGarbituDL_Click()
    PBrickCtrl.SetDatalog 0 'Datalog husten du
```

```
End Sub

Private Sub cmdSetDLTamaina_Click()
    If PBrickCtrl.SetDatalog(Val(TxtDLtamaina.Text)) Then
        DatalogLbl.Caption = "Datalog-en tamaina " + TxtDLtamaina + " da"
    Else
        DatalogLbl.Caption = "Ez dago behar besteko memoriarik"
    End If
End Sub

Private Sub Form_Load()
    PBrickCtrl.InitComm
End Sub
```

9.4.2. Programaren egikaritzea

- Gorde ezazu programa.
- Egikari ezazu programa.
- Konekta ezazu argi-sentsorea 2 sarreran.
- Piztu ezazu RCX.
- Idatz ezazu 7 zenbakia testu-laukian eta saka ezazu *Datalog Konfiguratu* botoia (ez ezazu idatz 50 baina handiagoa den zenbakirik testu-laukian)
- Saka ezazu *Programa transferitu* botoia
- Saka ezazu RCXko **Run** botoia.
- Programaren egikaritzea bukatzen denean, saka ezazu *Datalog Berreskuratu* botoia.

Zazpi sarrera agertuko dira ListBox-ean, 0ri dagokion datalog-eko sarrerak datalog-en tamaina gordetzen du. Gainerako sarrerak *DatalogNext* metodoaz datalog-en gordetako balioak dira. 1, 2 eta 3 zenbakidun datuak erregistratu diren tenporizadorearen balioak dira, hurrengoa sentsorearen irakurketa, eta bukatzeko 10 izeneko aldagaiaren balioa eta berriro tenporizadorearen balioa.

Ohartuko zarenez, *Datalog Konfiguratu* botoia sakatzean, koadrante bat agertzen da RCXko LCD pantailan. **Run** botoia sakatzean, zirkulua betetzen hasten da (hau da, koadrante gehiago agertzen dira). Datalog-en edukia ezabatzeko, egin ezazu klik *Datalog Hustu* botoian.

9.4.3. Kodearen deskribapena

cdmSetDLTamaina: prozedura honek *txtDLTamaina* testu-laukiko balioan oinarrituz datalog-en tamaina finkatzen du. Gehienezko tamaina 2000 ingurukoa da, baina alda daiteke. Behar besteko memoriarik ez badago, *SetDatalog*(Val(txtDLTamaina.Text)) metodoak huts egiten du, eta errore-mezu bat lblDatalog label-ean agertuko da.

cmdJaitsi: RCXra transferitutako programak TENP_4 tenporizadorearen balioa segundo bateko tartea utziaz hiru aldiz gordeko du datalog-en. Ondoren, sentsorearen irakurketa datalog-en gordeko da, eta bukatzeko 10 izeneko aldagaiaren balioa eta berriro tenporizadorearena.

cmdKargaDL: prozedura honek datalog RCXtik **Mat** izeneko matrizera pasatzen du.

```
Mat = PBrickCtrl.UploadDatalog(0, Val(TxtDLTamaina.Text) + 1)
```

Datalog-en lehen elementutik (0) hasiko da, eta bukaerara iritsi arte jarraituko du. 1 gehitzeko arrazoia datalog-en lehen elementuak bere tamaina gordetzen duela da, adibidez, sei datu gorde badira, zazpi elementu egongo dira (datuak gehi tamaina).

Itzulitako matrizea bi dimentsioduna da: hiru lerro eta txtDLTamaina + 1 zutabe.

Matrizea zuzena bada:

```
For iKontadorea = LBound(Mat, 2) To UBound(Mat, 2)
            LstDatalog.AddItem "Mota: " + Str(Mat(0, iKontadorea))_
             + " Zbk. :" + Str(Mat(1, iKontadorea)) + "Balioa :" +_
             Str(Mat(2, iKontadorea))
Next iKontadorea
```

Matrizearen beheko muga definituta dago (hau da, lehen elementuaren posizioa) eta goiko muga ere (hau da, azken elementuaren posizioa). Eta bien arteko elementu bakoitzari sarrera bat dagokio.

Mota		Zenbakia	Irakurketa
0	ALD	0 – 31	Itzulitako
1	TENPOR	0 – 3	irakurketak
9	SENBALIO	0 – 2	
14	ERLOJU	0	

9.3 taula

Datalog-en edukia ezabatzeko bere tamaina zerora gutxitzen da. Hau egitean, koadrantea RCXko LCD pantailatik desagertuko da.

```
Private Sub cmdHustuDL_Click()
    PBrickCtrl.SetDatalog 0 ‘ hustu datalog
End Sub
```

9.5. Programa grafikoa

Aurreko programan, datalog-etik eskuratutako datuak zerrenda baten moduan aurkeztu dira. Hala ere, gehienetan, datuak era grafikoan irudikatzea askoz adierazgarriagoa izaten da. Kapitulu honetako bigarren proposamen honen helburua aurreko kapituluan egindako hurbiltasun programan hainbat aldaketa egitea da: argi sentsorearen irakurketak erregistratuko dira, eta ondoren, era grafikoan irudikatuko dira.

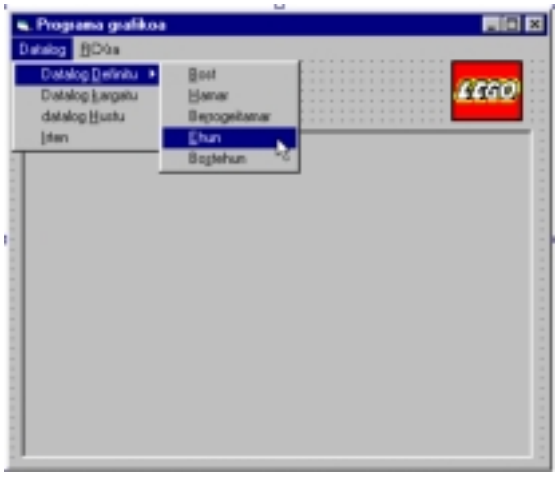
Programa honetan, menua, prozedurak eta irudi-koadroak (PictureBox) erabiliko ditugu.

- Eman iezaiozu hasiera Lego proiektu berri bati.
- Gorde ezazu fitxategi guztiei **Grafikoa** izena emanez.

Proiektu honetako formularioan grafiko bat jarri behar dugunez, toki gutxi izango dugu kontrolak bertan jartzeko. Horregatik, menuak erabiliko ditugu.

9.5.1. Menuak

Aurreko programetan komando-botoiak erabili ditugu, eta oraingo honetan horien ordez menuak jarriko ditugu. 9.2 irudian lortu nahi dugun emaitza ikus daiteke.



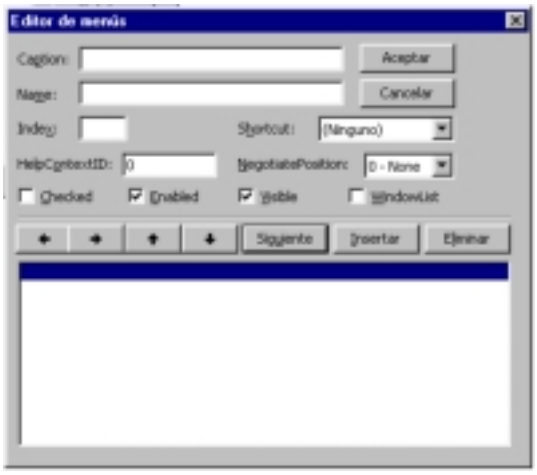
9.2 irudia:
menuak formularioan

- Egin ezazu formularioa 9.4 taulako datuetan oinarrituz.

Kontrol Mota	Propietatea	Balioa
Form	Nombre Caption	frm_Grafikoa Programa Grafikoa

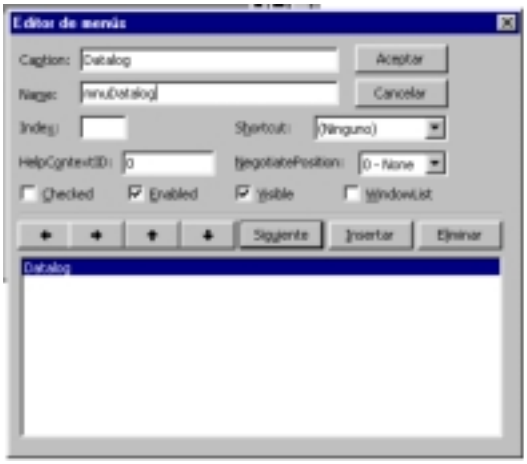
9.4 taula

- Hauta ezazu formularioa.
- Hauta ezazu *Herramientas* menuko *Editor de menús* aukera.



9.3 irudia:
menuen editorea

- Idatz ezazu **&Datalog** *Caption* testu-laukian.
- Idatz ezazu **mnuDatalog** *Name* testu-laukian.

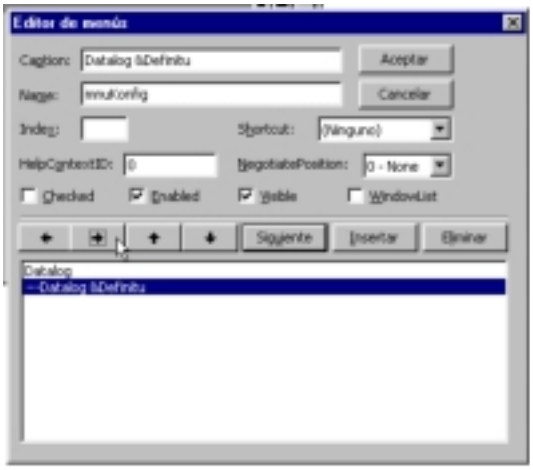


9.4 irudia:
datalog sarrera menuen editorean

- Saka ezazu *Editor de menús*-eko *Siguiente* botoia, eta hurrengo lerroa urdinez nabarmendua ageriko da.
- Idatz ezazu **Datalog &Definitu** *Caption* testu-laukian.
- Idatz ezazu **mnuKonfig** *Name* testu-laukian.

Datalog Definitu elementua koskatu behar da, *Datalog* menukoa delako.

- Saka ezazu *Editor de menús*-eko eskuin alderako gezia.



9.5 irudia
Datalog Definitu elementua koskatua

- Saka ezazu *Siguiente* botoia.
- Idatz ezazu **Datalog &Kargatu** *Caption* testu-laukian.
- Idatz ezazu **mnuKargatu** *Name* testu-laukian.
- Saka ezazu *Siguiente* botoia.
- Idatz ezazu **Datalog &Hustu** *Caption* testu-laukian.
- Idatz ezazu **mnuHustu** *Name* testu-laukian.
- Saka ezazu *Siguiente* botoia.
- Idatz ezazu **&Irten** *Caption* testu-laukian.

- Idatz ezazu **mnuIrten** *Name* testu-laukian.

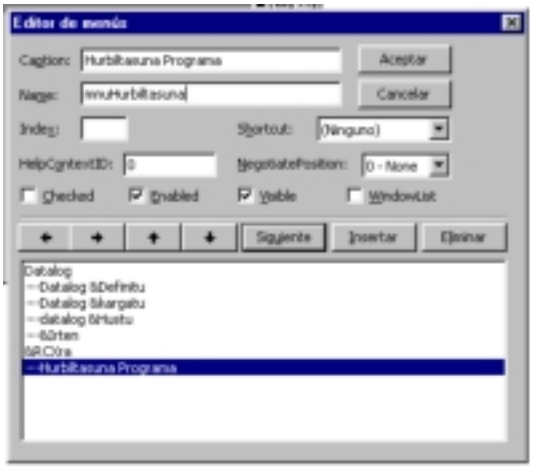
Datalog menua osorik dago. Orain, RCXra programa transferitzeko erabiliko dugun menua egingo dugu.

- Saka ezazu *Siguiente* botoia.
- Idatz ezazu **&RCXra** *Caption* testu-laukian.
- Idatz ezazu **mnuTransferitu** *Name* testu-laukian.

Menu berri baten izena denez, eta ez menu baten menpe dagoen elementu bat, ez du koskatua egon behar.

- Saka ezazu *Editor de menús*-eko ezker alderako gezia koska kentzeko.
- Saka ezazu *Siguiente* botoia.
- Idatz ezazu **&Hurbiltasuna programa** *Caption* testu-laukian.
- Idatz ezazu **mnuHurbil** *Name* testu-laukian.
- Saka ezazu *Editor de menús*-eko eskuin alderako gezia elementu hau koskatzeko.

Formularioa osorik dago, eta ondo egin baduzu menuen editoreak 9.6 irudiko itxura izango du.



9.6 irudia:
menuen editorea edizioaren amaieran

- Saka ezazu *Editor de menús*-eko *Aceptar* botoia.
- Gorde ezazu proiektua.

frmGrafikoa formularioak 9.7 irudiko itxura izango du. *Datalog* edo *RCXra* menuetan klik egiten baduzu, eskaitzen dituzten aukerak ikusi ahal izango dituzu.

9.7 irudia
Formularioa bukatua

- Egikari ezazu programa.

Menuek eskaintzen dituzten aukerak hauta daitezke, baina jakina denez, ez da ezer gertatuko, ez diegulako koderik lotu.

- Saka ezazu *Programa Grafikoa*-ko goiko eskuineko X ikonoa programatik ateratzeko.

9.5.2. Submenuak

Datalog menuko lehen aukera *Datalog Definitu* da. Arestian ikusi dugu *SetDatalog* metodoak datalog-en tamaina aldatuko duen parametro bat behar duela. Horretarako aukera ezberdinak eskainiko dizkigun submenu bat sortuko dugu.

- Hauta ezazu *Herramientas* menuko *Editor de menús* aukera.
- Hauta ezazu *Datalog Kargatu* elementua eta saka ezazu *Insertar* botoia.
- Idatz ezazu **&Bost** *Caption* testu-laukian.
- Idatz ezazu **mnuBost** *Name* testu-laukian
- Saka ezazu eskuin alderako gezia gehiago koskatzeko.
- Hauta ezazu *Datalog Kargatu* elementua eta saka ezazu *Insertar* botoia.
- Idatz ezazu **&Hamar** *Caption* testu-laukian.
- Idatz ezazu **mnuHamar** *Name* testu-laukian
- Saka ezazu eskuin alderako gezia gehiago koskatzeko.
- Edita itzazu modu berean ondoko elementuak:

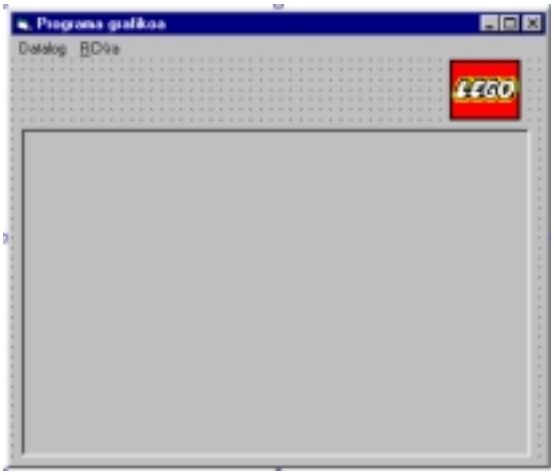
Caption	Name
Be&rrogeitamar	mnuBerrogeitamar
&Ehun	mnuEhun
Bo&stehun	mnuBostehun

9.5 taula

- Gorde ezazu proiektua.

9.5.3. Kontrol grafikoa

- Hauta ezazu tresnen koadroko PictureBox kontrola, eta jar ezazu formularioan.
- Eman iezaiozu **picGrafikoa** balioa *Nombre* propietateari. Orain **frmGrafikoa** formularioak 9.7 irudiko itxura izango du.



9.8 irudia
Formularioa bukatua

9.5.4. Programa grafikoaren kodea

- Idatz ezazu ondoko kodea:

```
' Aldagai guztiak deklaratu behar dira
Option Explicit

Private Sub Form_Load()
    PBrickCtrl.InitComm
End Sub
```

Orain *Datalog* menuko *Irten* elementuaren kodea idatziko duzu.

- *Objeto* bistan zaudela, egin ezazu klik *Datalog* menuan eta hauta ezazu *Irten* aukera.

Honek komando-botoi baten gainean egiten den klik-bikoitzaren antzeko ondorioa du. *Código* leihoan *mnuIrten_Click* prozeduraren eredua agertuko da.

- Idatz ezazu ondoko kodea:

```
Private Sub mnuIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub
```

- Gorde eta egikari ezazu programa.
- Hauta ezazu *Datalog* menuko *Irten* aukera.

Ondorioz, programaren egikaritzea bukatuko da.

9.5.5. Prozedurak

Datalog Definitu submenuan sortu behar den datalog-en tamaina definitzeko aukera ezberdinak eskaintzen dira. Datalog konfiguratu ondoren, behar bezala sortu den egiaztatu behar da

(hau da, ea behar besteko tokirik zegoen). Aukera bakoitzean egiaztapen hau egingo duen kodea idatzi beharrean, hau automatikoki egingo duen prozedura bat idatz dezakegu.

- *Código* leihoa irekita dagoela, hauta ezazu *Herramientas* menuko *Agregar Procedimiento* aukera.
- Idatz ezazu **KonfDatalog** *Name* testu-laukian.

9.9 irudia
Agregar Procedimiento elkarrizketa-leihoa

- Saka ezazu *Aceptar* botoia.

Prozedura berrirako eredua agertuko da:

```
Public Sub KonfDatalog()

End Sub
```

- Orain Datalog prozedurako lehen lerroa aldatu behar da:

```
Public Sub KonfDatalog(Tamaina As Integer)

End Sub
```

- Idatz ezazu ondoko kodea:

```
Public Sub KonfDatalog(Tamaina As Integer)
    If PBrickCtrl.SetDatalog(Tamaina) Then
        MsgBox "Datalog-en tamaina: " + Str(Tamaño), vbInformation
    Else
        MsgBox "Ez dago behar besteko memoriarik", vbCritical
    End If
End Sub
```

- Hauta ezazu *Objeto* bistan *Datalog* menuko *Datalog Definitu* ⇒ *Bost* aukera.

Ondoko gertaera-prozedura eredua agertuko da:

```
Private Sub mnuBost_Click()

End Sub
```

- Idatz ezazu ondoko kodea:

```
Private Sub mnuBost_Click()
    KonfDatalog 5
End Sub
```

Instrukzio honek *KonfDatalog* prozedura egikarituko du. 5 zenbakia pasako dio prozedurari, eta, *Tamaina* aldagaiak 5 balioa hartuko du.

- Errepika itzazu aurreko urratsak *Datalog Definitu* submenuko gainerako elementuekin.

Orain Hurbiltasuna Programa aukerari kodea lotuko diogu.

- Hauta ezazu *Objeto* bistan *RCXra* menuko *Hurbiltasuna Programa* aukera.
- Idatz ezazu ondoko kodea (kodea aurreko kapitulukoa bezalakoa da, baina ataza bat gehiago du).

```
Private Sub mnuHurbil_Click()
    Const AZKEN_IRAKUR = 10
    Const GORABEHERA = 11

    With PBrickCtrl
        .SelectPrgm PRGM_5

        .BeginOfTask NAGUSIA
            .SetVar GORABEHERA, KONST, 100
            .StartTask ATAZA_1
            .StartTask ATAZA_2
        .EndOfTask

        ' atazak mezuak bidaltzen ditu etengabe, 2 segundo bakoitzeko

        .BeginOfTask ATAZA_1
            .Loop KONST, BETI
                .SendPBMessage KONST, 0
                .Wait KONST, MS_50
        .EndOfTask

        'ibilgailuaren kontrola duen ataza

        .BeginOfTask ATAZA_2
            .SetSensorType SENTSOREA_2, ARGI_SENTS
            .SetSensorMode SENTSOREA_2, MODU_RAW, 0
            .SetFwd A_MOTOREA + C_MOTOREA
            .On A_MOTOREA + C_MOTOREA

            .StartTask ATAZA_3
            .Loop KONST, BETI
                .SetVar AZKEN_IRAKUR, SENBALIO, SENTSOREA_2
                .SumVar AZKEN_IRAKUR, ALD, GORABEHERA
                .If SENBALIO, SENTSOREA_2, HANDI_BAINO, ALD, AZKEN_IRAKUR
                    .SetRwd A_MOTOREA + C_MOTOREA
                    .Wait KONST, SEG_1
                    .Off C_MOTOREA
                    .Wait KONST, SEG_1
                    .SetFwd A_MOTOREA + C_MOTOREA
                    .On C_MOTOREA
                .EndIf
            .EndLoop
        .EndOfTask

        ' Irakurketak datalog-en gordetzen ditu hamar aldiz segundoko

        .BeginOfTask ATAZA_3
            .Loop KONST, 100
                .DatalogNext SENBALIO, SENTSOREA_2
                .Wait KONST, MS_100
            .EndLoop
            .Off A_MOTOREA + C_MOTOREA
            .StopAllTasks
        .EndOfTask
```

```
    End With
End Sub
```

Lotu iezaiozu kodea *Datalog* menuko *Datalog Kargatu* aukerari.

- Hauta ezazu *Objeto* bistan *Datalog* menuko *Datalog Kargatu* aukera.
- Idatz ezazu ondoko kodea:

```
Private Sub mnuKargatu_Click()
    Dim iLoteak, i, iKonta As Integer
    Dim Mat As Variant
    Dim iX, iGoi, iBehe As Integer
    Dim iGutX, iGehX, iGutY, iGehY As Integer

    Mat = PBrickCtrl.UploadDatalog(0, 1)
    iGoi = Mat(2, 0)

    'grafikoaren mugak definitu
    iGutX = 0: iGehX = iGoi ' x 0 eta elementu kopuru balioaren artekoa
    iGutY = 500: iGehY = 850 'y 500 eta 850 balioen artekoa
    iX = 0 'x 0 koordenatuan hasiko da

    picGrafikoa.Cls
    picGrafikoa.Scale (iGutX, iGehY)-(iGehX, iGutY)
    picGrafikoa.ForeColor = QBColor(4)

    iLoteak = Int(iGoi / 50) 'zatiduraren alde osoa

    For iKonta = 0 To iLoteak
        iBehe = iKonta * 50
        If iGoi <= 50 Then
            Mat = PBrickCtrl.UploadDatalog(iBehe, iGoi)
        Else
            Mat = PBrickCtrl.UploadDatalog(iBehe, 50)
        End If
        iGoi = iGoi - 50
        If IsArray(Mat) Then
            For i = LBound(Mat, 2) To UBound(Mat, 2)
                iX = iX + 1
                picGrafikoa.Line -(iX, Mat(2, i))
            Next i
        Else
            MsgBox "Ez da baliozko matrizea"
        End If
    Next iKonta

End Sub
```

- Idatz ezazu ondoko kodea *Datalog Hustu* aukerarako:

```
Private Sub mnuHustu_Click()
    KonfDatalog 0 'Datalog ezabatu
End Sub
```

Programaren edizioa bukatuta dago, beraz egikaritzeko ordua iritsi da. Aurreko kapituluko *Hurbiltasun* robot bera erabiliko dugu.

- Gorde ezazu proiektua.
- Egikari ezazu proiektua.
- Hauta ezazu *Datalog* menuko *Datalog Definitu* ⇒ *Ehun* aukera.

- Hauta ezazu *RCXra* menuko *Hurbiltasuna Programa*. Honela, programa RCXra transferituko dugu.
- Saka ezazu RCXko **Run** botoia.
- Programaren egikaritzea bukatzen denean, hauta ezazu *Datalog* menuko *Datalog Kargatu* aukera (hau egiteko, robota IR dorretik gertu jarri behar da, komunikazioetan arazorik egon ez dadin).

Formularioko PictureBox kontrolean grafiko bat agertuko da, hain zuzen, 9.10 irudian agertzen den grafikoaren antzekoa. Adibide honetan robotak bi oztopo aurkitu ditu. Robota oztoporantz hurbiltzen den abiadurak grafikoko minimoen zabalera baldintzatzen du.

9.10 irudia
Argi-sentsoreak egindako irakurketak era grafikoan irudikatuta.

- Irten zaitez programatik. Datalog ezabatu nahi baduzu, hauta ezazu *Datalog* menuko *Datalog Hustu* aukera programatik atera aurretik.

9.5.6. Kodearen deskribapena

Argi-sentsoreak argiaren intentsitatea neurtzen hasten denean, ATAZA_3 hasieratua izango da.

```
.BeginOfTask 3
      .Loop KONST, 100
          .DatalogNext SENBALIO, SENTSOREA_2
          .Wait KONST, MS_100
      .EndLoop
      .Off A_MOTOREA + C_MOTOREA
      .StopAllTasks
.EndOfTask
```

Ataza honek Loop begizta ehun aldiz errepikatuko du. Begiztaren hasieran argi--sentsorearen irakurketa datalog-en erregistratuko du. 100 errepikapenak egin ondoren, ataza guztiak geldituko ditu (hau da, programa geldituko da).

mnuKargatu_Click: prozedura honek grafikoa PictureBox-en jarriko du. Horretarako lehenengo urratsa datalog-en edukia ordenagailura transferitzea izango da.

Lehenengo lerroak datalog-eko lehen elementua *Mat* matrizera transferitzen du, eta ondoren, datalog-ek duen elementu kopuruari dagokion balioa *iGoi* aldagaiari esleitzen dio.

```
Mat = PBrickCtrl.UploadDatalog(0, 1)
iGoi = Mat(2, 0)
```

Grafikoaren koadroaren koordenada mugak definitu behar dira, datuak behar bezala irudikatzeko.

```
iGutX = 0: iGehX = iGoi       ‘x 0 eta elementu kopuru balioaren artekoa
iGutY = 500: iGehY = 850 ‘y 500 eta 850 balioen artekoa
iX = 0                          'x 0 koordenatuan hasiko da
```

x ardatzean datalog-eko elementuak eta **y** ardatzean elementu bakoitzari dagokion argi-sentsorearen irakurketa irudikatzen dira. Grafikoaren koadroa garbitu ondoren, eskala definituko dugu: lehen koordenatua goiko ezkerreko izkina eta bigarrena beheko eskuineko izkina dira. *Forecolor* propietateak grafikoaren kolorea definitzen du (kasu honetan gorria).

```
picGrafikoa.Cls
picGrafikoa.Scale (iGutX, iGehY)-(iGehX, iGutY)
picGrafikoa.ForeColor = QBColor(4)
```

Datuen transferentzia 50 elementu edo gutxiagoko loteetan egin behar da. Beraz, 50 elementu baino gehiago transferitu behar direnean aparteko zenbat aldi behar diren kalkulatu behar da. Horretarako ondorengo espresioa idatzi dugu:

```
iLoteak = Int(iGoi / 50)      ‘zatiduraren alde osoa
```

Transferitu beharreko datuak 69 badira, datalog bi lotetan transferituko da, lehenengoa 50 elementukoa eta bigarrena 19koa. *iLoteak* aldagaiaren balioa 1 izango litzateke aparteko transferentzia bat beharko litzatekeela adierazteko. Ondoren For ... Next begizta bati hasiera ematen zaio.

```
    For iKonta = 0 To iDenbora
        iBehe = iKonta * 50
        If iGoi <= 50 Then
            Mat = PBrickCtrl.UploadDatalog(iBehe, iGoi)
        Else
            Mat = PBrickCtrl.UploadDatalog(iBehe, 50)
        End If
        iGoi = iGoi – 50

                ’hemengo kodea aurrerago deskribatzen da

    Next iKonta
```

Begiztaren hasieran kontadorearen (*iKonta*) balioa 0 da, eta *iLoteak* baliora iristean gelditukо da. *iBehe*-k transferitu behar den lehen elementuaren posizioa gordetzen du. Transferitu beharreko loteak hiru badira, *iBehe*-ren lehen balioa 0 izango da, ondoren 50, eta bukatzeko 100. If ... Then ... Else egiturak batera zenbat elementu transferituko diren zehazten du: 50 baino gutxiago badira kopuru zehatza transferituko da, baina gehiago badira, 50 elementuko lotea transferituko da eta transferitzeko zenbat datu gehiago gelditzen diren kalkulatuko da (iGoi= iGoi — 50).

Ikus dezagun orain falta den kode zatia:

```
  If IsArray(Mat) Then
      For i = LBound(Mat, 2) To UBound(Mat, 2)
          iX = iX + 1
          picGrafikoa.Line -(iX, Mat(2, i))
      Next i
  Else
```

```
            MsgBox "Ez da baliozko matriza"
        End If
    Next iKonta
```

IsArray() funtzioaren bitartez matrizea baliozkoa den egiaztatuko dugu (hau da, ea arrakastaz transferitu den). Baliozkoa bada matrizearen goiko eta beheko muga definituta egongo dira, eta For ... Next begiztaren bitartez grafikoa marraztuko du, *i* kontadorearen balio bakoitzeko puntu bat. *iX* aldagaiak marraztutako azken puntuaren x koordenatua gordetzen du, eta 1 gehitzen diogu lerroaren hurrengo puntua marrazteko. Hurrengo instrukzioak puntu bakar baten koordenatua erabiltzen du, aurreko puntutik datorren zuzenaren bukaera eta hurrengoaren hasiera izango dena (koordenatua OraingoX, OraingoY).

```
picGrafikoa.Line -(iX, Mat(2, i))
```

9.6. Ariketak

Hartzen den irakurketa kopurua aldatu nahi baduzu, alda ezazu ATAZA _3 atazako aldi kopurua. Irakurketak datalog-en erregistratzen diren maiztasuna ere alda daiteke.

10 Roboten arteko komunikazioak

RCX bat baino gehiago izanez gero, elkarren artean komunikatzeko aukera ematen dieten programak idatz ditzakegu. Horretarako infragorrien ataka erabiliko dugu, eta robotek elkarlanean zerbait egin ahal izango dute (edo elkarren artean borrokatu). Honek eskaintzen digun beste aukera bat robot handiagoak eraikitzea da motore eta sentsore gehiagorekin.

10.1. Kapitulu honetako edukiak

1. Robotek elkarren artean komunikatzeko erabiltzen dituzten metodoak.
2. Robot nagusia eta esklaboak. Komunikazio protokoloen definizioa.

10.2. Lan metodologia

RCXen arteko komunikazioak infragorrien bitartez egiten dira, hain zuzen ere, *SendPBMessage* metodoa erabiliz. Metodo honek 0 eta 255 arteko zenbaki bat transmitituko du, eta igorlea den RCXtik gertu dagoen beste edozein RCXk menua jaso eta gorde ahal izango du. RCXen artean besteen kontrola duen RCXri Nagusia (Master) izena ematen zaio eta gainerakoei Esklabo izena (Eslave). RCXk bere barneko memorian gordeta duen mezua *ClearPBMessage* metodoaz ezaba dezake. Instrukzio honek mezuari “0” barneko balioa esleitzen dio.

10.3. Nagusia eta Esklaboa

Egingo dugun programa sinpleak RCX batek (Nagusia) beste bati (Esklaboa) mezu bat nola bidaltzen dion erakutsiko du.

- Has zaitez ohiko moduan. Formularioko datuak 10.1 taulan dituzu.
- Gorde ezazu zure proiektua fitxategi guztiei **RCXKom** izena emanez.

Kontrol mota	Propietatea	Balioa
Form	Nombre Caption	FrmRCXtik_RCXra RCXen arteko komunikazioak
CommandButton	Nombre Caption	cmdNagusia &Nagusia transferitu
CommandButton	Nombre Caption	cmdEsklaboa &Esklaboa transferitu
CommandButton	Nombre Caption	cmdIrakur &Irakur
CommandButton	Nombre Caption	cmdIrten &Irten
TextBox	Nombre Text	txtIrakur (Hutsik utzi)

10.1 taula

- Idatz ezazu ondoko kodea:

```
Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub

Private Sub cmdIrakur_Click()
      TxtIrakur.Text = Str(PBrickCtrl.Poll(PBMESS, 0))
End Sub

Private Sub cmdNagusia_Click()
      With PBrickCtrl
            .SelectPrgm PRGM_3
            .BeginOfTask NAGUSIA
                  .SendPBMessage KONST, 123
            .EndOfTask
      End With
End Sub

Private Sub cmdEsklabua_Click()
      With PBrickCtrl
            .SelectPrgm PRGM _4
            .BeginOfTask NAGUSIA
                  .ClearPBMessage
                  'Wait for Message
                  .While PBMESS, 0, BERDIN, KONST, 0
                        .Wait KONST, MS_50
                  .EndWhile
                  .PlaySystemSound SWEEP_DOWN_SOINUA
            .EndOfTask
      End With
End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm
End Sub
```

- Gorde ezazu zure programa.
- Egikari ezazu zure programa
- Piztu ezazu RCX bat (lehenengo hau *Nagusia* izango da) eta saka ezazu *Nagusia transferitu* botoia.
- Itzal ezazu *Nagusia* eta piztu ezazu beste RCX (hau *Esklaboa* izango da).
- Saka ezazu *Esklaboa transferitu* botoia.
- Piztu ezazu berriro *Nagusia*.
- Saka ezazu *Esklaboko* **Run** botoia, eta ondoren, *Nagusikoa*.

Dena ondo egin baduzu, Esklaboak SWEEP_DOWN_SOINUA emitituko du.

- Itzal ezazu *Nagusia* eta saka ezazu formularioko *Irakur* botoia. 123 zenbakia testu-laukian agertuko da, eta honela mezua behar bezala igorri dela egiaztatuko da.

10.3.1. Kodearen deskribapena

Nagusia programa egikaritzen denean 123 mezua igortzen du, eta ondoren, gelditu egiten da. Ordurako, *Esklaboa* programa egikaritzen ari da mezu baten zain. Jasotzen duen unean, soinu bat igorriko du eta egikaritzeari amaiera emango dio.

Garrantzi handikoa da komunikazioetarako protokoloak definitzea, hau da, besterik egin aurretik mezu bakoitzaren esanahia zehaztea.

10.3.2. Ariketa

Egiten ditugun proiektuetan arazorik ez izateko, RCX *Nagusiak* mezu bat bidaltzen duenean egokia izaten da mezua jasoa izan dela ziurtatzea. *Esklaboak* mezua jaso duela ziurtatzeko, *Nagusiari* erantzun beharko dio mezua behar bezala jaso duela adierazteko. Egin itzazu kodean behar diren aldaketak hala izan dadin (adibidez, 1 mezuaren bitartez eman dezala erantzuna).

10.4. Urrutiko kontrola

Orain aurreko kapituluetan egin dugun robot mugikorra beste RCX baten bitartez kontrolatuko dugu. *Nagusiak Esklaboaren* jokaera kontrolatuko du. Azken honek hiru agindu bete beharko ditu:

- Aurrera (1 mezua)
- Atzera (2 mezua)
- Gelditu (3 mezua)

- Munta ezazu robot oinarria (ikus A eranskina).
- Egin itzazu ondoko aldaketak kodean:

```
Private Sub cmdNagusia_Click()
      With PBrickCtrl
            .SelectPrgm PRGM _3

            .BeginOfTask NAGUSIA
                  .ClearPBMessage
                  .SendPBMessage KONST, 1 ‘ aurrera
                  .Wait KONST, SEG_3
                  .SendPBMessage KONST, 2 'atzera
                  .Wait KONST, SEG_3
                  .SendPBMessage KONST, 3 ' gelditu
            .EndOfTask

      End With
End Sub

Private Sub cmdEsklabua_Click()
      With PBrickCtrl
            .SelectPrgm PRGM_4
            .BeginOfTask NAGUSIA
                  .ClearPBMessage
                  .Loop KONST, BETI
                        ‘ Mezuen zain
                        .While PBMESS, 0, BERDIN, KONST, 0
                              .Wait KONST, MS_10
                        .EndWhile
                        ' Motoreak martxan jarri: Aurrera
                        .If PBMESS, 0, BERDIN, KONST, 1
                              .SetFwd A_MOTOREA + C_MOTOREA
                              .On A_MOTOREA + C_MOTOREA
                        .EndIf

                        ' Jar ezazu hemen atzera egiteko kodea

                        ' Jar ezazu hemen gelditzeko kodea
```

```
                    .ClearPBMessage
                .EndLoop
            .EndOfTask
    End With
End Sub
```

- Gorde ezazu zure proiektua.
- Egikari ezazu zure proiektua.
- Errepika ezazu aurreko transferentziako prozedura.
- Egikari ezazu *Esklaboa* programa.
- Egikari ezazu *Nagusia* programa.

Esklaboa robota 3 segundoz aurrera egingo du, beste hiru segundoz atzera, eta ondoren, gelditu ko da.

10.5. Ariketak

1. Egin itzazu *Esklaboaren* kodean behar diren aldaketak *Esklaboak* mezuak jaso dituela erantzuteko (mezuak jaso dituela banan-banan baieztatuko du). Baina kasu honetan *Nagusiak* denbora-tarte jakin batean igorpenaren baieztapena jasotzen ez badu, *Nagusia* programak berriz bidaliko dio mezua *Esklaboari*. Protokolo bat erabaki beharko duzu, hau da, zein mezu izango den baieztapena emango duena (0 ... 255).
2. Jar itzazu gela batean robot batzuk era aleatorio batean mugi daitezen. Jar ezazu lurrean objektu bat robotek aurki dezaten (objektuaren ordez atzealde zuri batean dagoen borobil beltz bat izan daiteke). Robot batek aurkitzen duenean gainerakoei jakinaraziko die, eta denak gelditu ko dira.

11 Erreminta osagarriak

Kapitulu honetan aurrekoetan jorratu ez diren hainbat erreminta eta programazio estrategia erabiltzen ikasiko duzu.

11.1. Mutex objektuak

Programa batean ataza batzuk daudenean, batera egikaritzen dira (paraleloan). Hau oso interesgarria da, baina ez da dirudien bezain erraza. Adibidez, gerta daiteke ataza batek motore bat denbora-tarte zehatz baterako aurreraka martxan jartzea, eta motorea martxan dagoenean beste ataza batek kontrako noranzkoan biratzeko agindua ematea. Hainbat kasuetan beharrezkoa izan daiteke, baina beste batzuetan arazoak ekar ditzake. Hau **mutex** bat erabiliz ekidin daiteke.

Mutex objektua bi egoera ezberdinak eduki ditzakeen sinkronizazio objektu bat da: jaberik gabea (0) eta jabe baten menpeko egoeran (1). Une bakoitzean, ataza bakarra izan daiteke mutex--aren jabe. Bere izenaren jatorria konpartitutako baliabideetan (adb. motore bat) duen erabilgarritasunean datza (co-ordinating **mut**ually **ex**clusive access: elkarrentzat baztertzailea den atzipenen koordinazioa).

Adibidez, bi atazek motore bat batera kontrola ez dezaten, ataza bakoitzak mutex-a libre dagoen egiaztatuko du. Hala bada, motorearen kontrola hartuko du, bestela, mutex-a libre gelditu arte zain egongo da.

Mutex-ak aldagaien bitartez inplementatzen dira. mutex-a erabili gabe dagoenean bere balioa 0 izango da. Ataza batek konpartitutako baliabide bat (adb. motore bat) erabili behar duenean aldagaiaren balioa 0 izan arte zain gelditzen da. Atazak mutex-a libre aurkitzen badu, 1 balioa esleitzen dio konpartitutako baliabidea erabiltzeko. Egin beharrekoa bukatzean, berriro esleitzen dio 0 balioa mutex-ari.

```
Private Sub cmdMutexEG_Click()

      Const MUTEX = 6                    ‘ 6 aldagaia mutex-a izango da.

      With PBrickCtrl

            .SelectPrgm PRGM_4
            .BeginOfTask NAGUSIA
                  .SetVar MUTEX, KONST, 0 ' Hasieran erabili gabe
                  .StartTask ATAZA_1
                  .StartTask ATAZA_2
            .EndOfTask

            .BeginOfTask ATAZA_1
                              ‘ Ataza honek motorea erabili nahi badu...
                  .While ALD, MUTEX, BERDIN, KONST, 1
                        .Wait KONST, MS_10
                  .EndWhile
                              ‘ Mutex propiotzat hartzen du
                  .SetVar MUTEX, KONST, 1
                        ‘ Motorearekin behar duena egiten du, eta
                        ‘ ondoren, mutex-a libre utzi du.
                  .SetVar MUTEX, KONST, 0
```

```
            .EndOfTask

            .BeginOfTask ATAZA_2
                        ‘ Ataza honek motorea erabili nahi badu...
                  .While ALD, MUTEX, BERDIN, KONST, 1
                        .Wait KONST, MS_10
                  .EndWhile
                        ‘ Mutex-a propiotzat hartzen du
                  .SetVar MUTEX, KONST, 1
                        ‘ Motorearekin behar duena egiten du, eta
                        ‘ ondoren, mutex-a libre utzi du.
                  .SetVar MUTEX, KONST, 0
            .EndOfTask

      End With
End Sub
```

11.2. Subrutinak

Subrutinak maiz erabiltzen diren kode zatiak gordetzeko erabiltzen dira. Adibidez, motore bat maiz martxan jarri eta gelditu behar baduzu, subrutina bat erabil dezakezu. Honela, motorea martxan jarri eta gelditu behar duzun bakoitzean nahikoa izango da subrutinari dei egitea. Ikus dezagun adibide baten bitartez:

```
Private Sub cmdSubEG_Click()
      Const ONOFF = 3 ‘Subrutinaren izena
      With PBrickCtrl
            .SelectPrgm PRGM_4
            .BeginOfTask NAGUSIA
                        ‘ Hemen kodea
                  .GoSub ONOFF
                        ' hemen kode gehiago
                  .GoSub ONOFF
                        ' hemen kode gehiago
            .EndOfTask

            .BeginOfSub ONOFF
                  .On A_MOTOREA
                  .Wait KONST, SEG_3
                  .Off A_MOTOREA
            .EndOfSub
      End With
End Sub
```

Ez da gomendagarria ataza ezberdinetatik subrutina berari dei egitea, ustekabeko portaerak ekar ditzakeelako.

Subrutinak oso erabilgarriak dira ataza luzeak dituzten programa luzeetan.Programa slot bakoitzean 8 subrutina erabil daitezke, eta beren izenak 0 eta 7 bitarteko zenbakiak izango dira (kasu honetan ere konstanteak erabil daitezke programa irakurterrazago bilakatzeko).

Aurreko kapituluan egin dugun komunikazio programan subrutina bat erabil genezakeen mezuen zain egoteko:

```
Private Sub cmdEsklabua_Click()

      Const MEZUAREN_ZAIN = 6 ' 6 subrutina

      With PBrickCtrl

            .SelectPrgm PRGM_4
```

```
                  ' 10 ms-ko denbora-tartetan mezurik iritsi den PSMESS
                  ' egiaztatzen du
            .BeginOfSub MEZUAREN_ZAIN
                  .While PBMESS, 0, BERDIN, KONST, 0
                        .Wait KONST, MS_10
                  .EndWhile
            .EndOfSub

            .BeginOfTask NAGUSIA
                  .ClearPBMessage
                  .SetFwd A_MOTOREA + C_MOTOREA
                  .Loop KONST, BETI
                              ' Mezuaren zain
                        .GoSub MEZUAREN_ZAIN
                                    ' Hemen kode gehiago
                        .ClearPBMessage
                  .EndLoop
            .EndOfTask

      End With

End Sub
```

11.3. Tenporizadoreak

RCXk 100 ms-ko bereizmena duten lau tenporizadore aske ditu. Tenporizadore hauek independienteak dira, beraz, zeroan banaka jar daitezke. 0an jarri bezain laster berriro abiatzen dira. Tenporizadoreak 0 eta 32 767 balioa eduki dezake, beraz, 3276 segundo zenbatu ditzake (gutxi gorabehera 55 minutu). Lehenengo tenporizadorea zeron jartzeko ondoko instrukzioa erabiliko dugu:

```
PBrickCtrl.ClearTimer TENP_1
```

Instrukzio hau egikaritzean, tenporizadorea berriz 0tik hasiko da, segundo hamarren bakoitzean unitate bat gehituz. Tenporizadorearen balioa eskuratzeko *Poll* metodoa erabiliko dugu:

```
PBrickCtrl.Poll(TENPOR, TENP_1)
```

Tenporizadoreak erabiltzen dituzunean gogoan izan beren bereizmena 100 ms-koa dela eta *Wait* metodoarena 10 ms-koa, hau da, tenporizadoreak 10 tik zenbatzen du segundoko, eta Wait metodoak 100. Ez itzazu RCXdatuak.bas moduluko denbora konstanteak erabil (adb. MS_100) tenporizadorekin egiten diren konparaketetan, konstante horiek 10 ms-ko bereizmenean oinarrituta definitu baitira.

12 Bibliografia

"Lego Mindstorms programming with Visual Basic" David Hanley eta Sean Hearne. http://emhain.wit.ie/~p98ac25/

"Controlling LEGO® Programmable Bricks Technical Reference" LEGO. http://mindstorms.lego.com/

"The Unofficial Guide to LEGO MINDSTORMS Robots". Jonathan B. Knudsen. O'REILLY 1999.

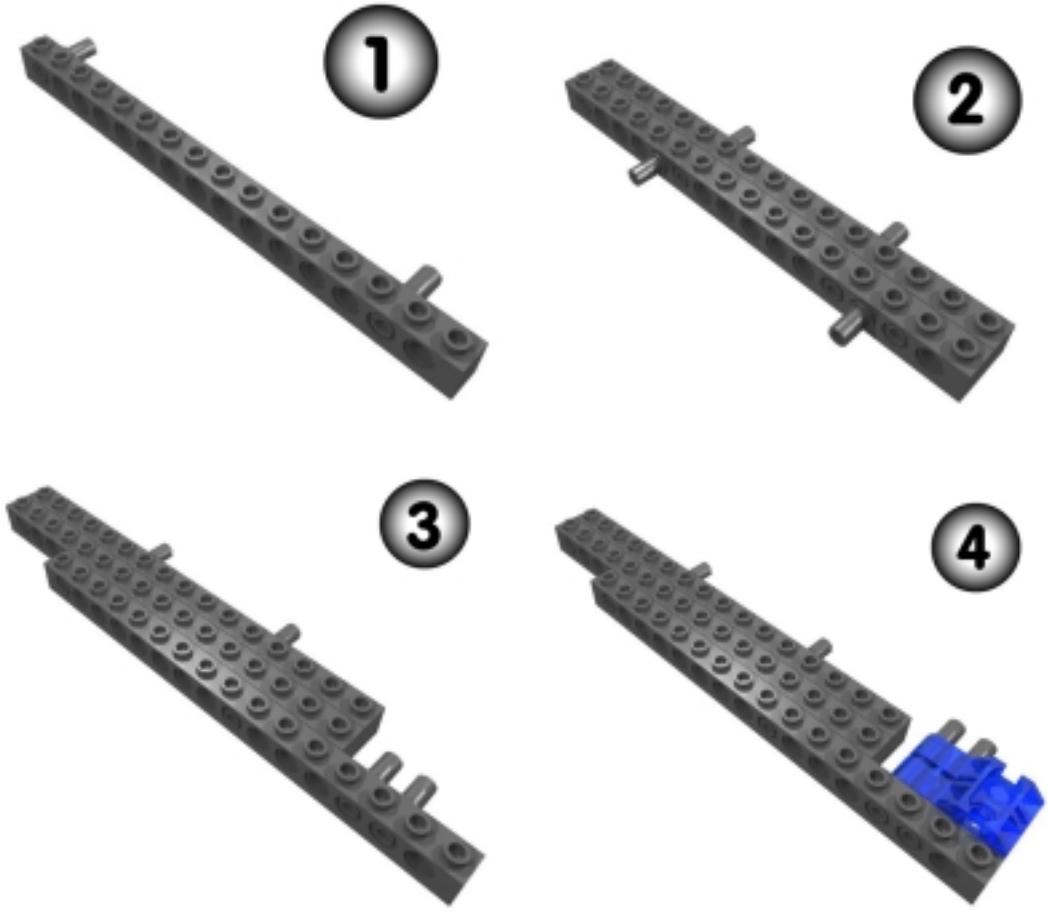
"Introducing Robotics with Lego Mindstorms" Robert Penfold. Bernard Babani (publishing) LTD 2000

"More Advanced Robots with Lego MindStorms" Robert Penfold. Bernard Babani (publishing) LTD 2000

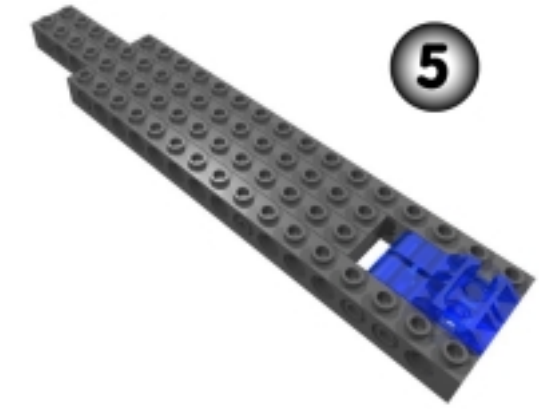
A eranskina: Robotak muntatzeko instrukzioak

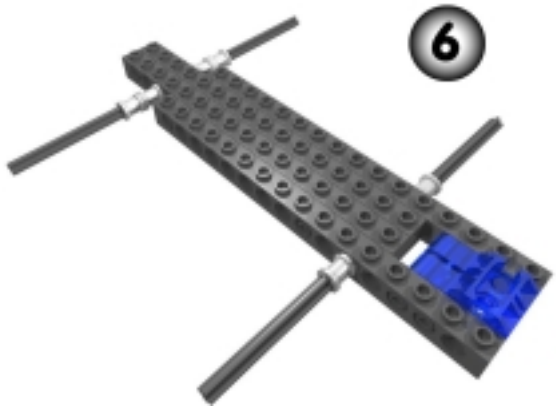
ROBOT OINARRIA[10]

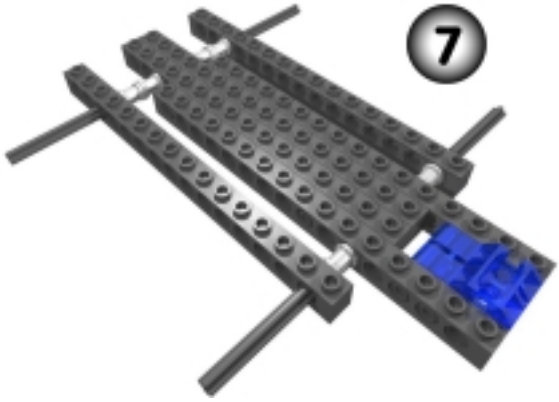
Eskuliburu honetan erabiliko dugun robot oinarria LEGO MindStorms-en konstruktopedia ezberdinetan oinarritutako modelo bat da. Antzekoren bat ere baliagarria izan daiteke, beti ere eskuineko motorea C irteeran eta ezkerrekoa motorea A konektatzen badituzu.

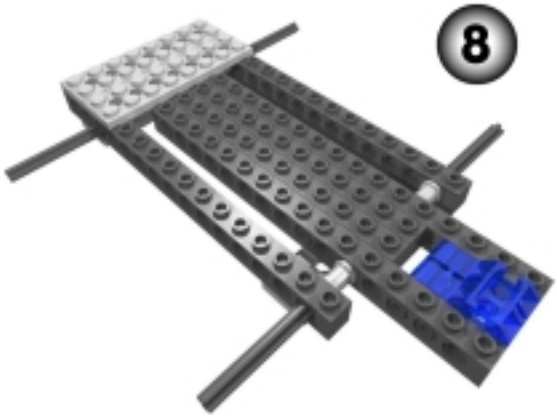


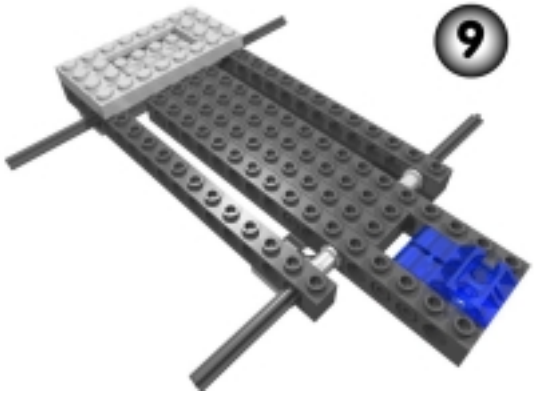
[10] Instrukzio hauek MLCAD erabiliz eginak dira. Eskuliburu honekin batera eskaintzen den instrukzioak.zip artxiboan aurki daitezke.

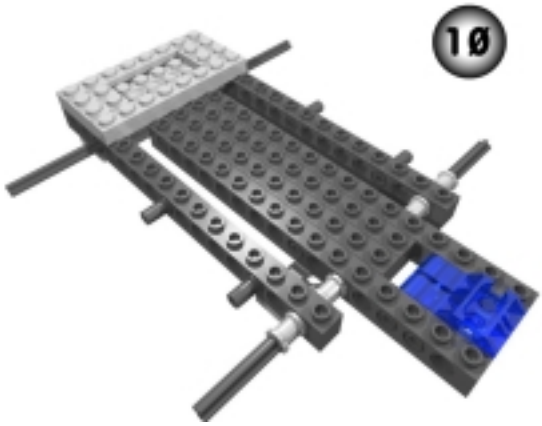

5

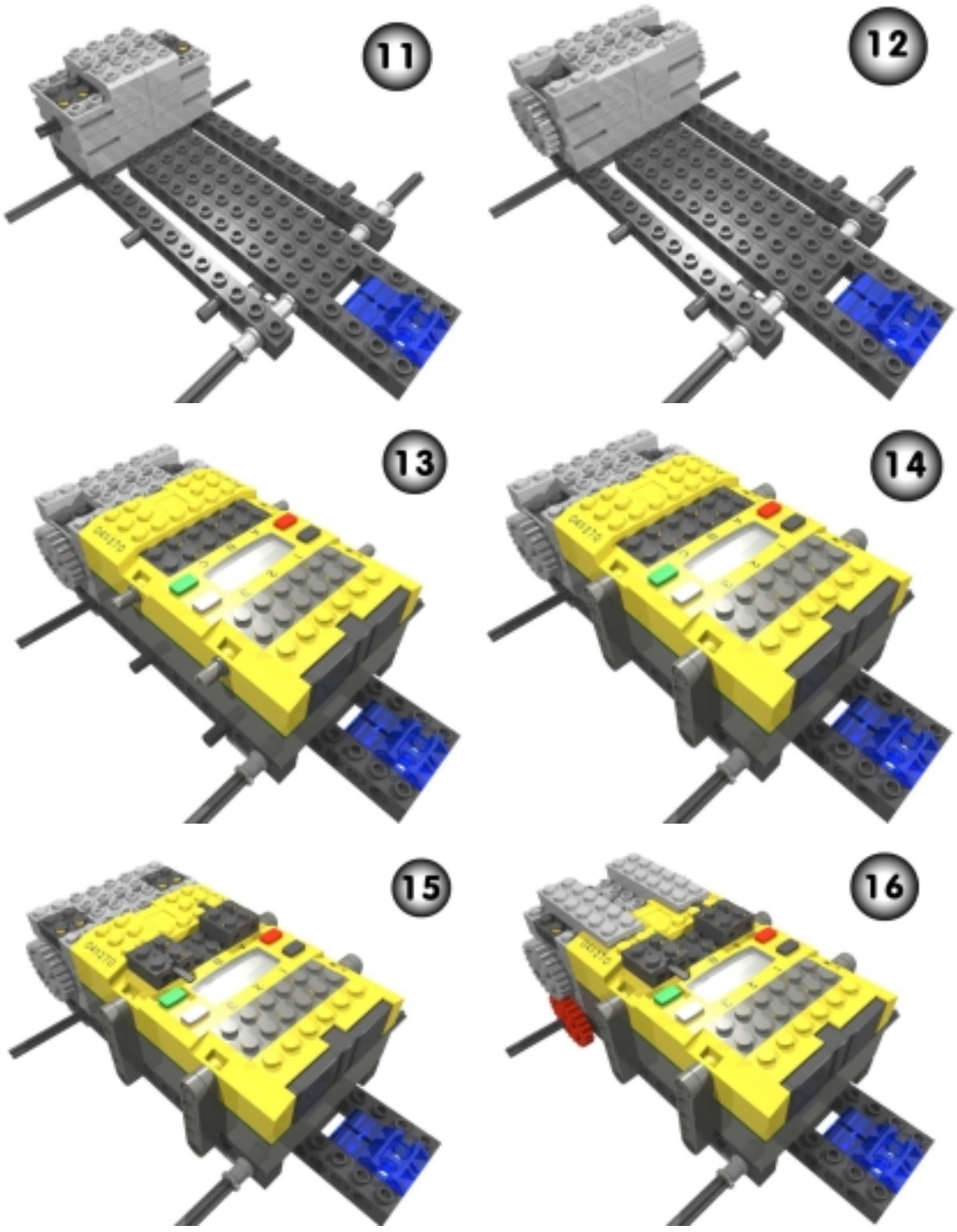

6


7


8

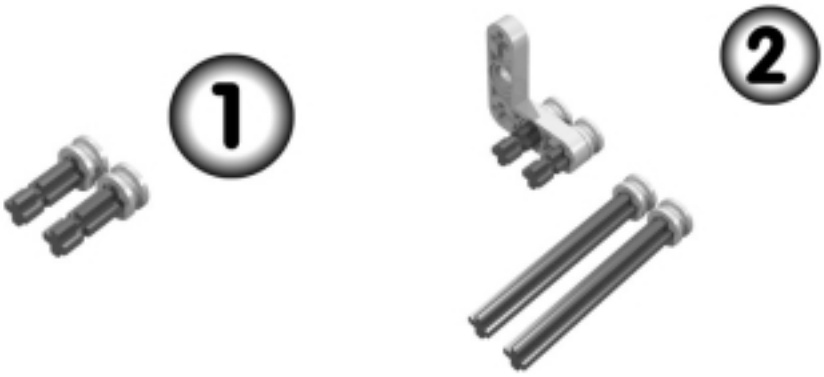

9

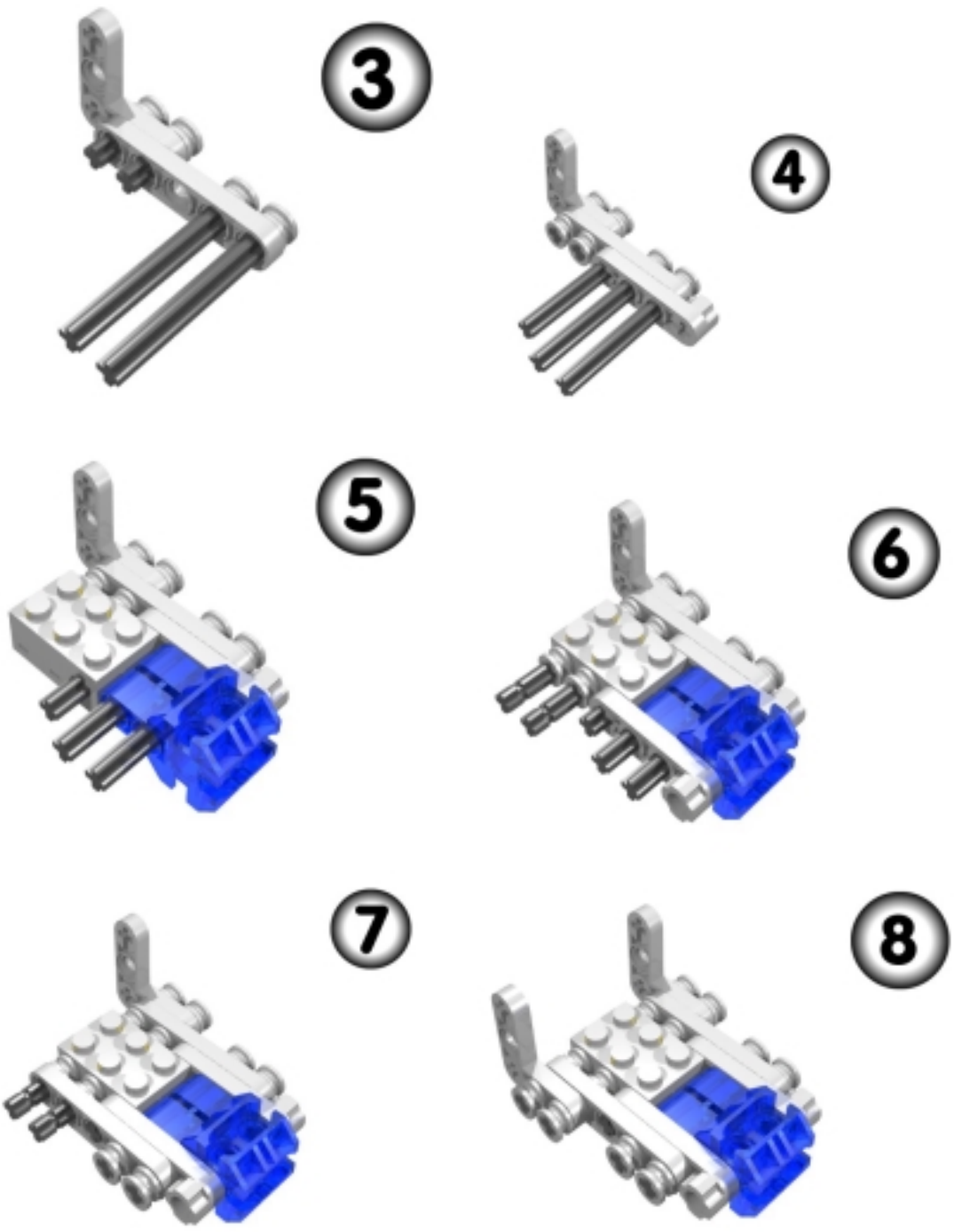

10

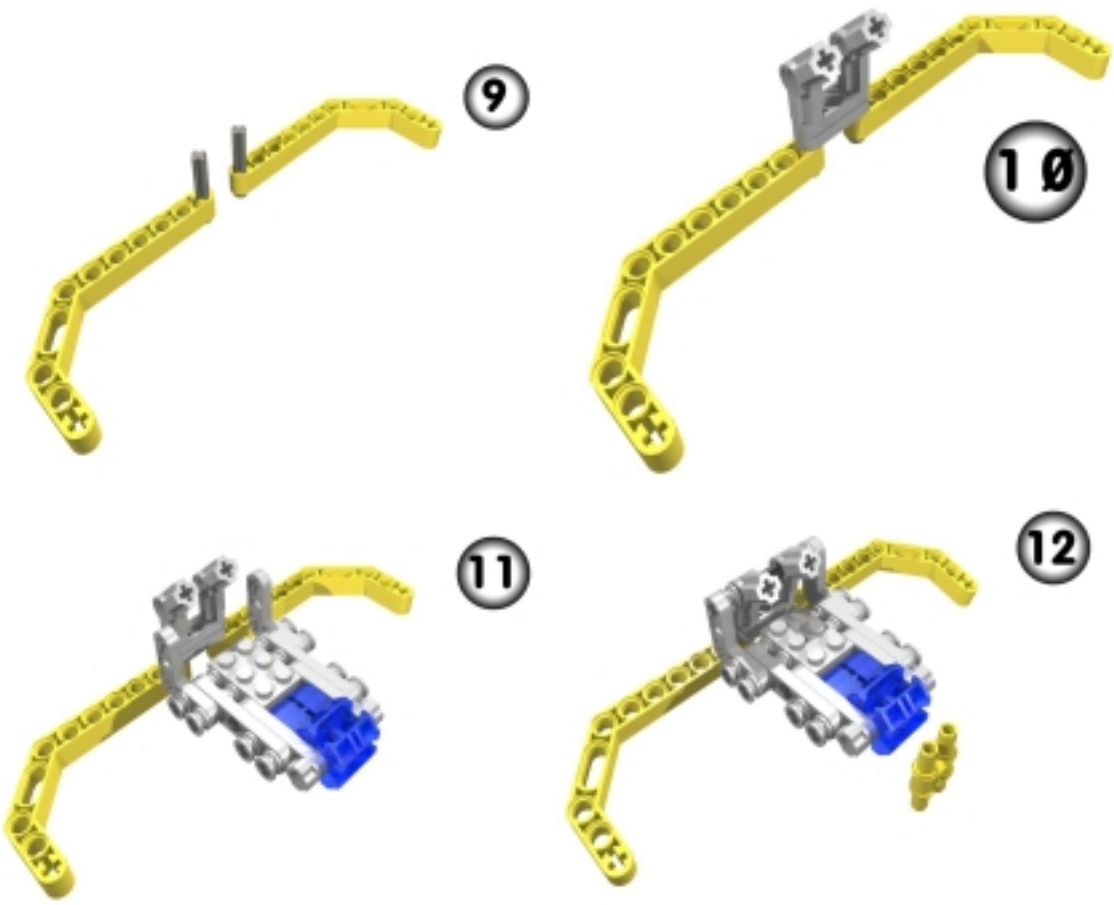

11
12
13
14
15
16

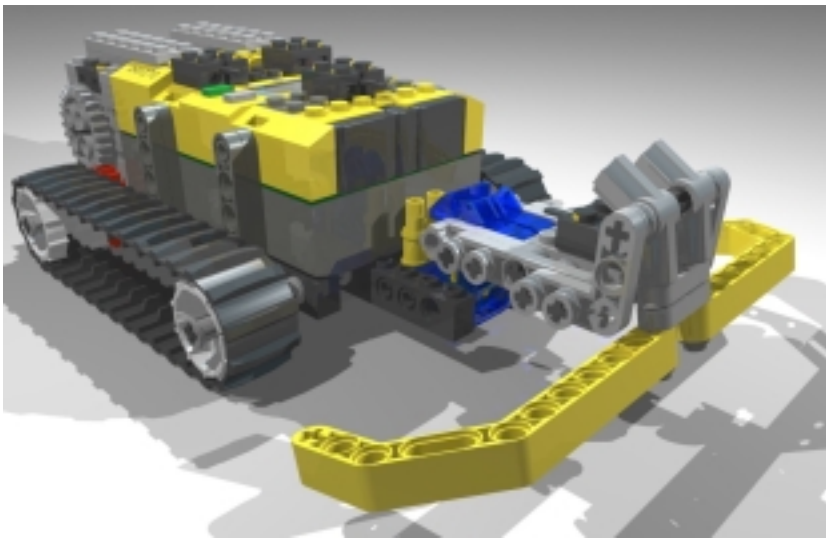
UKITZE-SENTSOREA DUEN MODULUA

Oztopoak detektatu behar dituzun proiektuetan modulo hau erants diezaiokezu aurreko robot oinarriari.



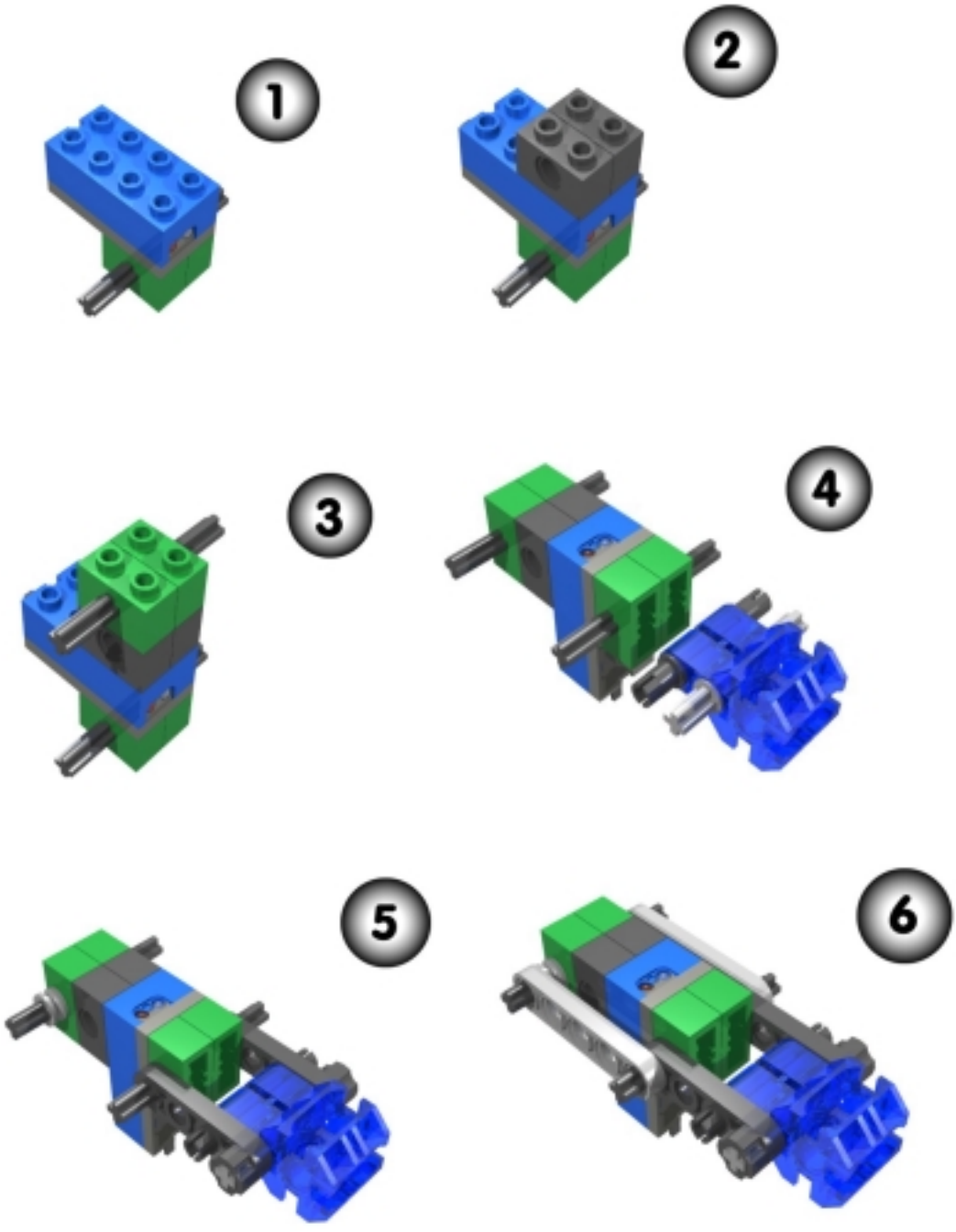

3
4
5
6
7
8

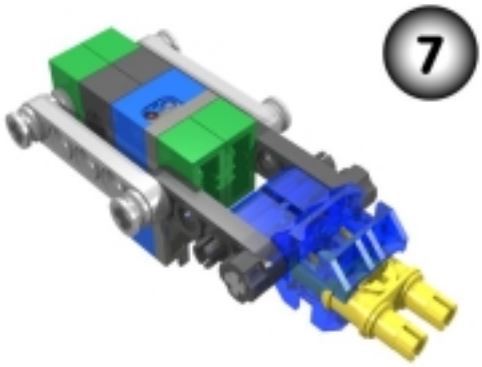

9
10
11
12



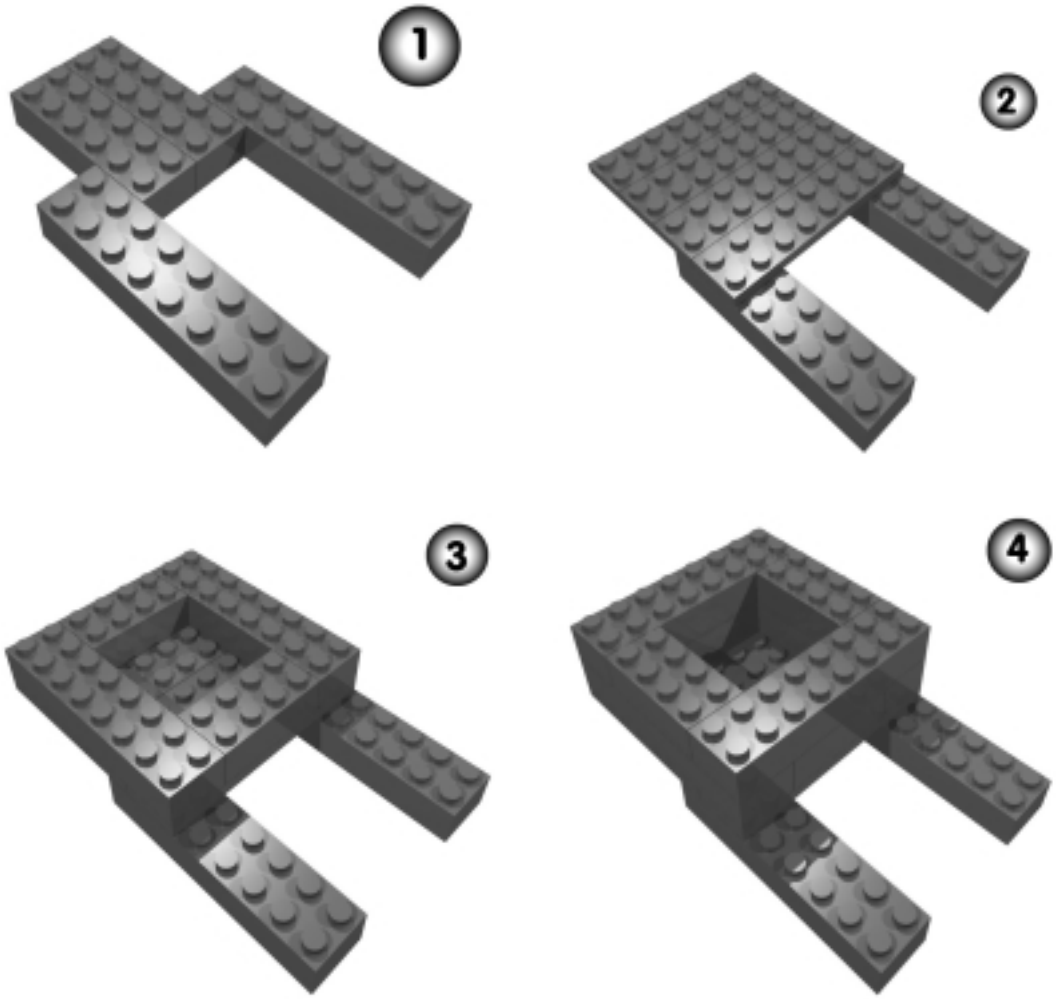
ARGI-SENTSOREA DUEN MODULUA

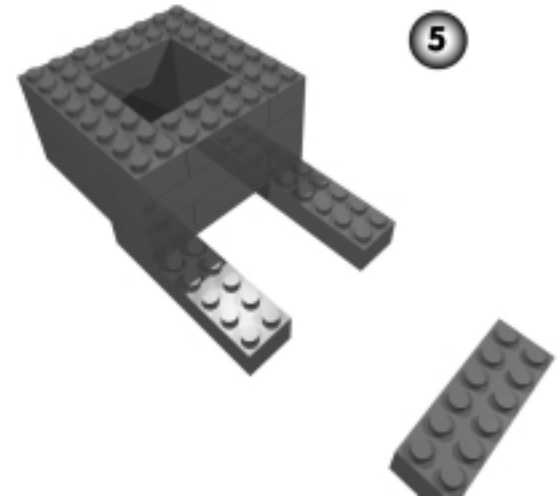
Modulu hau eskuliburu honen autoreak eginikoa da. Mutur luzea izan arren nahiko sendoa da. Zure esku dago hau edo beste bat erabiltzea.





KOLORE DETEKTAGAILUA




5


6

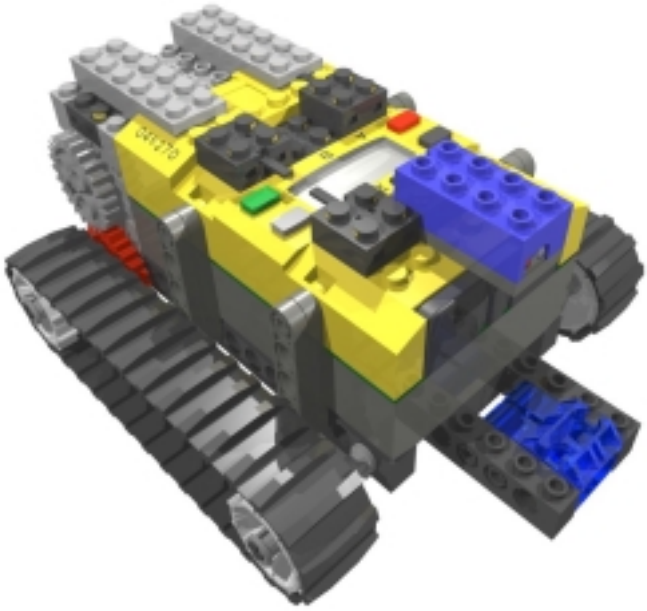

7


8


10

“HURBILTASUNA” ROBOTA

Robot hau egiteko abiapuntua eranskin honetako robot oinarria da, beraz, oinarria egina dagoenean ondoko hau gehitu.



B eranskina: RCXdatuak.bas

```
'=====================================================================
' Lego MindStorms roboten programazioa. Visual Basic.
' Modulu globala
' 1.0 bertsioa
'---------------------------------------------------------------------
' RCX-rako konstanteen izen globalen deklarazioa
'=====================================================================
Option Explicit
'=====================================================================
' Sartu hemen edo programaren hasieran zure konstante propioak
'=====================================================================

'=====================================================================
' Programen posizioak 0 - 4
'=====================================================================
Public Const PRGM_1 = 0
Public Const PRGM_2 = 1
Public Const PRGM_3 = 2
Public Const PRGM_4 = 3
Public Const PRGM_5 = 4
'=====================================================================
' Atazen Izenak - 1 eta 9 arteko atazei izen egokiagoa eman diezaiekezu
' Horretarako ez dituzu hauek derrigorrez ordezkatu edo ezabatu beharrik.
' Konstante propioen atalean egin dezakezu.
'=====================================================================
Public Const NAGUSIA = 0
Public Const ATAZA_1 = 1
Public Const ATAZA_2 = 2
Public Const ATAZA_3 = 3
Public Const ATAZA_4 = 4
Public Const ATAZA_5 = 5
Public Const ATAZA_6 = 6
Public Const ATAZA_7 = 7
Public Const ATAZA_8 = 8
Public Const ATAZA_9 = 9
'=====================================================================
' Sistemako soinuak
'=====================================================================
Public Const KLIK_SOINUA = 0
Public Const BEEP_SOINUA = 1
Public Const SWEEP_DOWN_SOINUA = 2
Public Const SWEEP_UP_SOINUA = 3
Public Const ERRORE_SOINUA = 4
Public Const SWEEP_FAST_SOINUA = 5

'=====================================================================
' Datuen jatorriaren izenak
'=====================================================================
Public Const ALD = 0
Public Const TENPOR = 1
Public Const KONST = 2
Public Const MOTSTA = 3
```

```
Public Const RAN = 4
Public Const TACC = 5
Public Const TACS = 6
Public Const MOTCUR = 7
Public Const KEYS = 8
Public Const SENBALIO = 9
Public Const SENTS_MOTA = 10
Public Const SENTS_MODU = 11
Public Const SENRAW = 12
Public Const BOOL = 13
Public Const ERLOJU = 14
Public Const PBMESS = 15
'=============================================================================
' Sentsoreen izenak
'=============================================================================
Public Const SENTSOREA_1 = 0
Public Const SENTSOREA_2 = 1
Public Const SENTSOREA_3 = 2
'=============================================================================
' Tenporizadoreen izenak
'=============================================================================
Public Const TENP_1 = 0
Public Const TENP_2 = 1
Public Const TENP_3 = 2
Public Const TENP_4 = 3
'=============================================================================
' Takimetro izenak (CyberMaster-arekin bakarrik)
'=============================================================================
Public Const EZKER_TAKIM = 0
Public Const ESKUIN_TAKIM = 1
'=============================================================================
' Irismen izenak
'=============================================================================
Public Const IRISMEN_LABURRA = 0
Public Const IRISMEN_LUZEA = 1
'=============================================================================
' Sentsore motak
'=============================================================================
Public Const EZ_MOTA = 0
Public Const UKITZE_SENTS = 1
Public Const TENPER_SENTS = 2
Public Const ARGI_SENTS = 3
Public Const ANGELU_SENTS = 4



'=============================================================================
' Sentsore moduak
'=============================================================================
Public Const MODU_RAW = 0
Public Const MODU_BOOL = 1
Public Const MODU_KONT_TRANS = 2
Public Const MODU_KONT_PERIO = 3
Public Const MODU_PORTZ = 4
Public Const MODU_CELSIUS = 5
Public Const MODU_FAHRENHEIT = 6
Public Const MODU_ANGELUA = 7
'=============================================================================
' Motore izenak (testu-kateak)
'=============================================================================
```

```
Public Const A_MOTOREA = "0"
Public Const B_MOTOREA = "1"
Public Const C_MOTOREA = "2"
'=============================================================================
' Irteeretako izenak
'=============================================================================
Public Const A_IRTEERA = "0"
Public Const B_IRTEERA = "1"
Public Const C_IRTEERA = "2"
'=============================================================================
' Konparaketa logikoen operadoreak
'=============================================================================
Public Const HANDI_BAINO = 0        'handiago baino
Public Const TXIKI_BAINO = 1        'txikiago baino
Public Const BERDIN = 2             'berdina
Public Const EZBERD = 3             'ezberdina
'=============================================================================
' Nahastea
'=============================================================================
Public Const BETI = 0
'=============================================================================
' Denbora konstanteak (MS = milisegundu)
'=============================================================================
Public Const MS_10 = 1                      ' 10 ms
Public Const MS_20 = (2 * MS_10)            ' 20 ms
Public Const MS_30 = (3 * MS_10)            ' 30 ms
Public Const MS_40 = (4 * MS_10)            ' 40 ms
Public Const MS_50 = (5 * MS_10)            ' 50 ms
Public Const MS_60 = (6 * MS_10)            ' 60 ms
Public Const MS_70 = (7 * MS_10)            ' 70 ms
Public Const MS_80 = (8 * MS_10)            ' 80 ms
Public Const MS_90 = (9 * MS_10)            ' 90 ms
Public Const MS_100 = (10 * MS_10)          '100 ms
Public Const MS_200 = (20 * MS_10)          '200 ms
Public Const MS_300 = (30 * MS_10)          '300 ms
Public Const MS_400 = (40 * MS_10)          '400 ms
Public Const MS_500 = (50 * MS_10)          '500 ms
Public Const MS_700 = (70 * MS_10)          '700 ms
Public Const SEG_1 = (100 * MS_10)          ' 1 s
Public Const SEG_2 = (2 * SEG_1)            ' 2 s
Public Const SEG_3 = (3 * SEG_1)            ' 3 s
Public Const SEG_5 = (5 * SEG_1)            ' 5 s
Public Const SEG_10 = (10 * SEG_1)          '10 s
Public Const SEG_15 = (15 * SEG_1)          '15 s
Public Const SEG_20 = (20 * SEG_1)          '20 s
Public Const SEG_30 = (30 * SEG_1)          '30 s
Public Const MIN_1 = (60 * SEG_1)           ' 1 mn
```

C Eranskina: erreferentzia teknikoa

"Controlling LEGO® Programmable Bricks Technical Reference"-n spirit.ocx-ko metodoei buruzko informazio teknikoa jasotzen da. Bertan, CyberMaster eta RCXrako beharrezkoa den informazio eskura daiteke. Hurrengo lerroetan metodo batzuen azalpena ematen da, baina laburpen bat baino ez da. Beraz, sakondu behar baduzu, badakizu nora jo.

On (MotorList)

Zerrendatzen diren motoreak martxan jartzen ditu.

MotorList: motoreen zerrenda ematen duen testu katea. Bere balioak "0", "1" eta "2" izan daitezke, baina hauek ordezkatzen dituzten konstanteak erabil daitezke: A_MOTOREA...

Erlazionatutako metodoak: Off(MotorList) metodoak motoreak geldituko ditu balazta moduan.

Adibidea: On "02" instrukzioak A eta C motoreak martxan jarriko ditu.

Poll(Source, Number)

Agindu honen bitartez informazioa eskura dezakegu, adibidez, sentsoreen irakurketak, aldagaien edukia, motoreen egoera.

Source eta Number: datu jatorri mota eta zenbakia ematen dute (Ikus 5.3 taula).

Adibidea: PBrickCtrl.Poll 0,7 instrukzioaren bidez, 7 izeneko aldagaiaren edukia jakin ahal izango dugu.

SetFwd (MotorList)

Metodo honek zerrendatzen diren motoreen noranzkoa finkatzen du: aurrerako noranzkoa. Ez du eraginik motoreen bestelako propietateetan.

MotorList: motoreen zerrenda ematen duen testu katea. Bere balioak "0", "1" eta "2" izan daitezke, baina hauek ordezkatzen dituzten konstanteak erabil daitezke: A_MOTOREA...

Erlazionatutako metodoak: SetRew(MotorList) metodoak noranzkoa atzeraka konfiguratzen du eta AlterDir(MotorList) metodoak noranzkoa aldatzen du.

Adibidea: SetFwd "02"

SetPower (MotorList, Source, Number)

Motoreen potentzia finkatzen du.

MotorList: motoreen zerrenda ematen duen testu katea. Bere balioak "0", "1" eta "2" izan daitezke, baina hauek ordezkatzen dituzten konstanteak erabil daitezke: A_MOTOREA...

Source eta Number: datu jatorri mota eta zenbakia ematen dute.

Adibidea: SetPower ("012", 0, 15) hiru motoreek 15 izeneko aldagaian gordeta dagoen zenbakiari dagokion potentzia maila emango dute.

SetSensorMode (Number, Mode)

Sentsorearen irakurketen formatua definitzen du.

Number: sarrera zenbakia

Type: sentsore modua (5.4 taula)

Adibidea: PbrickCtrl..SetSensormode 0, 0: 0 sarreran (RCXko 1 zenbakiduna) konektatutako sentsorea Raw moduan konfiguratuta, beraz, 0 eta 1023 balioa emango du.

SetSensorType (Number, Type)

Sarreran nolako sentsorea dagoen konektaturik definitzeko erabiltzen da.

Number: sarrera zenbakia

Type: sentsore mota (5.2 taula)

Adibidea: PbrickCtrl..SetSensorType 0, 4 0 sarreran (RCXko 1 zenbakiduna) sentsore angeluarra.

SetVar (VarNo As Integer, Source As Integer, Number As Integer)

Aldagaiei balioa esleitzeko balio du.

VarNo: aldagaia identifikatzen duen zenbakia (0 eta 31 bitartekoa) edo zenbakia ordezkatzen duen konstantea.

Source: Balioaren jatorria: konstantea, sentsore baten irakurketa...

Number: Aldagaiari esleituko zaion balioa.

Adibidea: SetVar 2, 0, 5 instrukzioaz 2 izeneko aldagaiari 5 izeneko aldagaiaren balioa esleitzen zaio.

D eranskina: Erroreen kontrola programen transferentzian

RCXra programak transferitzen direnean, DownLoadDone gertaerak transferentziaren arrakastaz informatzen du.

- Programa RCXra errorerik gabe transferitzen bada, ErrorCode-ren balioa 1 izango da.
- Erroreren bat gertatzen bada, ErrorCoderen balioa 0 izango da.

Jarraian aurkezten den kodea edozein proiektutan erabil daiteke. Proiektu guztietan sartu nahi baduzu, erabiltzen duzun txantiloiaren kodean jaso dezakezu.

Txantiloi horren formularioak D.1 irudiko itxura izango du.

D.1 irudia: formularioa

```
' Aldagai guztiak hasieratu behar dira
Option Explicit
Dim blnZain As Boolean
Dim blnTransferOK As Boolean

Private Sub cmdTransferitu_Click()
      blnZain = True
      ' Idatz ezazu hemen RCXra transferitu nahi duzun kodea

End Sub

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm
      blnZain = False
      blnTransferOK = False
End Sub


Private Sub PBrickCtrl_AsyncronBrickError(ByVal Number As Integer,_
Description As String)
```

```
      If (blnZain) Then
            While (blnZain)
                  DoEvents
            Wend
            MsgBox "Asynchronous Brick Error: " + Str(Number) + " " +_
            Description, vbCritical, "Erroreak transferentzian"
      Else
            MsgBox "Asynchronous Brick Error: " + Str(Number) + " " +_
            Description, vbCritical, " Erroreak transferentzian "
      End If
End Sub

Private Sub PBrickCtrl_DownloadDone(ByVal ErrorCode As Integer, ByVal
DownloadNo As Integer)
      If ErrorCode = 0 Then
            blnTransferOK = True
            'MsgBox " Transferentzia Arrakastatsua "
      Else
            'MsgBox " Erroreak transferentzian "
      End If
      blnZain = False
End Sub

Private Sub PBrickCtrl_DownloadStatus(ByVal timeInMS As Long, ByVal _
sizeInBytes As Long, ByVal taskNo As Integer)
      If (blnZain) Then
            While (blnZain)
                  DoEvents
            Wend
            If (blnTransferOK) Then
                  OutputStats timeInMS, sizeInBytes, taskNo
                  blnTransferOK = False
            End If
      Else
            If (blnTransferOK) Then
                  OutputStats timeInMS, sizeInBytes, taskNo
                  blnTransferOK = False
            End If
      End If
End Sub

' Transferentziari buruzko informazioa mezu-koadro batean
Public Sub OutputStats(Time As String, Size As String, Task As String)
      Dim LFCR As String
      LFCR = Chr(13) + Chr(10)
      MsgBox "Denbora: " + Time + LFCR + "Tamaina: " + Size + LFCR_
      +"Ataza zbk.: " + Task, vbInformation, "Transferentzia_
      arrakastatsua"
End Sub
```

Kodearen deskribapena

Kode honen helburuak hauek dira:

- Programaren transferentzian gerta daitezkeen erroreak detektatzea.
- Programa behar bezala transferitu bada, estatistikak erakustea.
- DownloadDone egikaritzen ari den bitartean besterik ez egitea.

ActiveX kontrolak gertaera bat bidaltzen badu, eta horren ondorioz edozein elkarrizketa--leihoa ireki behar bada, ActiveX kontroletik Visual Basic-era bidalitako gainerako gertaerak desagertuko dira.

Demagun *DownloadDone* gertaera-prozeduran mezu-koadro baten instrukzioa agertzen dela, eta batera *AsyncronBrickError* gertaera-prozeduran beste bat. Programaren transferentzian errore bat gertatuko balitz *DownloadDone* prozedurak monitorean jarritako mezu-koadroa desagertuko litzateke, eta *AsyncronBrickError*-ena irekiko litzateke.

Aurreko kodeak ez die *AsyncronBrickError* eta *DownloadStatus* prozedurei ezer egiten uzten *DownloadDone* prozedura egikaritzen den bitartean (blnZain = True denean), eta *DownnloadStatus* prozedurak bakarrik eskainiko ditu estatistikak transferentzia arrakastaz bukatu bada (blnTransferOK = True).

E eranskina: Motoreen egoeraren azterketa

Motorearen noranzkoa, potentzia... jakin nahi badugu *Poll* metodoa erabiliko dugu. Baina *Poll* metodoa motoreekin erabiltzea ez da gainerako elementuekin egiten dugunean bezain zuzena. Hau informazioa paketatua dagoelako gertatzen da. *Poll* metodoak itzultzen digun informazioa zenbaki oso bat izango da, eta zenbaki hori adierazten digunaren berri izateko zenbaki bitar bihurtu beharko dugu (8 bit kasu honetan).

Bosgarren kapituluko 5.3 taulan bit bakoitzaren esanahia aurkezten da.

Kontrol mota	**Propietate**	**Balio**
Form	Nombre Caption	frmMotorPoll Motoreen informazioa
CommandButton	Nombre Caption	cmdEskura &Informazioa eskuratu
CommandButton	Nombre Caption	cmdIrten I&rten
Text Box	Nombre Text	txtHamar (Hutsik utzi)
Text Box	Nombre Text	txtBitar (Hutsik utzi)
Text Box	Nombre Text	txtOnOff (Hutsik utzi)
Text Box	Nombre Text	Balazta (Hutsik utzi)
Text Box	Nombre Text	txtIrteera (Hutsik utzi)
Text Box	Nombre Text	txtNoranzkoa (Hutsik utzi)
Text Box	Nombre Text	TxtPotentzia (Hutsik utzi)

E.1 taula

E.1 taulako datuekin formulario bat egin ondoren, idatz ezazu jarraian ematen den kodea. Programa honek *Poll* metodoak itzultzen duen zenbaki osoa nola erabili erakusten du.

```
' Aldagai guztiak hasieratu behar dira
Option Explicit

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub


Private Sub cmdEskura_Click()
```

```
      Dim strStatus As String
      Dim iMotorea As Integer             ‘ zenbaki osoa gordetzen du
      Dim bMotorea As String              ‘ balio bitarra gordetzen du

      iMotorea = PBrickCtrl.Poll(MOTSTA, 0) `informazioa eskuratzen du
      txtHamar = Str(iMotorea) ‘ zenbakia testu kate bihurtzen du

      bMotorea = Bin(iMotorea) ‘ zenbaki hamartarra bitar bihurtzen du
      txtBin = bMotorea

‘Beheko maila bilatzen du
      strStatus = Mid(bMotorea, 6, 3)          ‘ 0-2 bitak hartzen ditu
      txtPower = Str(BintoDec(strStatus))      ‘ Balio hamartarra

' Zein noranzkoan?
      If Val(Mid(bMotorea, 5, 1)) = 1 Then           '3. bita hartzen du
            txtNoranzkoa = "Aurrera"           'if = 1 => Fwd(Aurrera)
      Else
            txtNoranzkoa = "Atzera"            'if = 0 => Rev(Atzera)
      End If

' Zein irteratan
      strStatus = Mid(bMotorea, 3, 2)     ' 4 eta 5 bitak hartzen ditu
      txtOutput = Str(BintoDec(strStatus))      ' Balio hamartarra

' Balazta ala Float[11] moduan?
      If Val(Mid(bMotorea, 2, 1)) = 1 Then           '6. bita hartzen du
            txtBalazta = "Balazta"             'if = 1 => Balazta
      Else
            txtBalazta = "Float modua"                'if = 0 => Float
                                                moduan
      End If

'ON / OFF?
      If Val(Mid(bMotorea, 1, 1)) = 1 Then           '7 bita hartzen du
            txtOnOff = "ON"                    'if = 1 => On
      Else
            txtOnOff = "OFF"                   'if = 0 => Off
      End If
End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm
End Sub

Public Function Bin(Number As Integer) As String
      Dim strBit As String
      Dim iPos As Integer
      Dim iZenbakia As Integer
      IZenbakia = Number
      For iPos = 7 To 0 Step -1
            If >= (2 ^ iPos) Then
                  strBit = strBit + "1"
                  iZenbakia = iZenbakia - (2 ^ iPos)
            Else
                  strBit = strBit + "0"
            End If
      Next
```

[11] Float modua: motorea balazta erabili gabe gelditzen da.

```
      Bin = strBit                                  'emaitza itzultzen du
End Function

Public Function BintoDec(Number As String)
      Dim iLuzera As Integer
      Dim bZenbakia As String
      Dim iHam As Integer
      Dim iPos As Integer

      iHam = 0

      bZenbakia = Number
      iLuzera = Len(bZenbakia)

      For iPos = iLuzera To 1 Step -1
            If Mid(bZenbakia, 1, 1) = "1" Then
                  iHam = iHam + (2 ^ (iLuzera - 1))
            End If
            bZenbakia = Mid(bZenbakia, 2, iLuzera)
            iLuzera = iLuzera - 1
      Next

      BintoDec = iHam

End Function
```

Kodearen deskribapena

Informazioa eskuratu sakatzen den bakoitzean, motoreei buruzko informazioa daraman zenbaki oso bat jasotzen da. Baina informazio hori paketatua dago, eta zenbaki osoa testu-kate bitar bihurtu behar da.

```
bMotorea = Bin(iMotorea)        'Balio bitarra
```

Adibidez, zenbaki osoa 79 bada, Bin funtzioak "01001111" testu-katea emango digu, hori baita 79 zenbaki hamartarraren adierazpide bitarra.

Ikus dezagun nola interpretatu informazio hau. E.2 taulan bit bakoitzaren esanahia erakusten da.

7	6	5 4	3	2 1 0
On /Off	Balazta / Float	Irteera zenbakia	Noranzkoa CW/CCW	Potentzia maila

E.2 taula

Gure kasuan interpretazioa hau izango da:

0	1	00	1	111
Geldirik (OFF)	Balazta	0 irteera	Erlojuaren orratzen noranzkoan (CW)	Potentzia=7

E.3 taula

Ikus dezagun banan-banan nola interpretatu testu-kate bitarra.

Motorearen potentzia:

```
strStatus = Mid(bMotorea, 6, 3)          ' 0 1 2 biten balioa hartzen du
txtPotentzia = Str(BintoDec(strStatus))  ' balioa hamartarra
```

Mid() funtzioak bigarren argumentuak definitzen duen posiziotik hasita hirugarren argumentuak eskatzen duen adina karaktere itzultzen du. Adibidez, Mid("Lego Mindstorms", 6, 4) instrukzioa egikaritzearen emaitza “Mind” katea litzateke.

strStatus = Mid(bMotorea, 6, 3) espresioak testu-kate bitarreko seigarren posiziotik aurrera dauden hiru karaktereak strStatus aldagaiari esleitzen dizkio. Zenbaki bitar baten kasuan hauexek izango dira 0, 1 eta 2 bitak. Balio honek aukeratutako motorearen potentziaren balioa adierazten du. Balio hamartarra ezagutzeko *BinToDec* funtzioa erabiliko dugu.

```
txtPotentzia = Str(BintoDec(strStatus))  ' balio hamartarra
```

Funtzio honek testu-kate bitar bat zenbaki hamartar bihurtzen du. txtPotentzia testu--laukian erakutsiko da emaitza.

Ikus dezagun adibide baten bitartez. Poll metodoak itzultzen duen balioa 79 bada, urrats hauek beteko dira:

79	→	“01001111”	→	“111”	→	7
Osoa	→	Bitarra	→	Eskatutako bitak	→	Osoa

Motorearen noranzkoa:

Motorearen noranzkoa hirugarren bitak ematen du, hau da, testu-kateko bosgarren elementua.

```
If Val(Mid(bMotorea, 5, 1)) = 1 Then ‘ hirugarren bita hartu
      txtNoranzkoa = "Aurrera" ‘erlojuaren orratzen noranzkoa
Else
      txtNoranzkoa = "Atzera" ‘erlojuaren orratzen noranzkoaren kontrakoa
End If
```

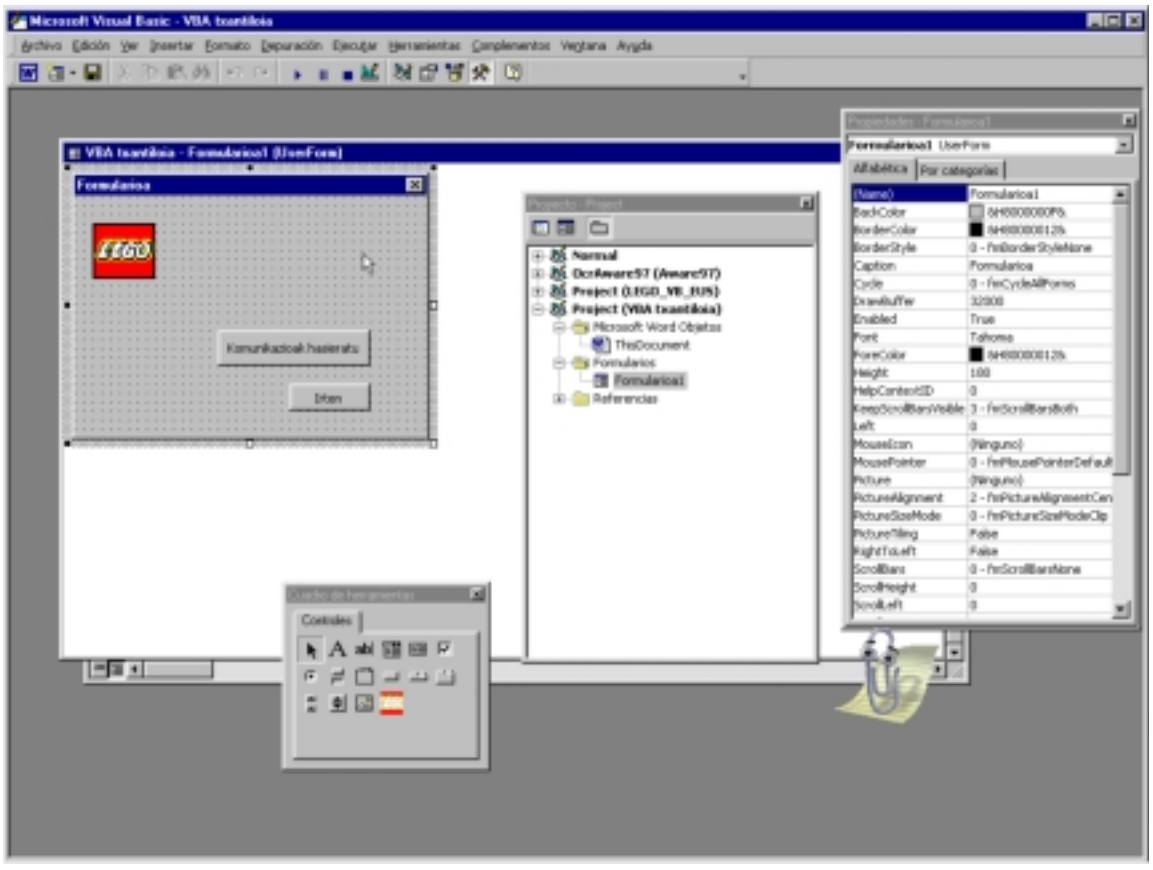
F eranskina: VB erabiltzeko beste aukerak

Visual Basic erabiltzeko beste aukera batzuk daude, lehena, Visual Basic for Aplications erabiltzea, eta bigarrena, BrickCommand.

Visual Basic for Aplications (VBA)

VBA Microsoft Office-ko aplikazioetan aurki daiteke, baita AutoCAD 2000 eta Corel Draw-en azken bertsioetan ere.

Esku liburu honekin batera “VBA txantiloia”, Word-eko dokumentua, eskaintzen da. aplikazio honen bitartez lanean zuzenean hasi ahal izateko. Bertan bestelako instrukzio batzuk ematen dira.



F.1 irudia VBA

Visual Basic baino mugatua da, baina oso erabilgarria da. Gainera, arestian aipatutako horietako aplikazio bat izan ezkero, ez dakar beste kosturik.

BrickCommand

Aplikazioa hau Interneten aurki daitekeen beste aplikazio bat da. Honi esker programak Visual Basic-en editatu eta ondoren, RCXra transferitu ditzakegu.

Azken bertsioa 2.0 da eta http://www.geocities.com/Area51/Nebula/8488/ helbidean aurki daiteke. Doakoa da, baina bere bitartez editatu daitezkeen programek RCXra transferitzeko modukoak izan behar dute, berehalako kontrola onartzen ez duelako (bakarrik programa berak eskaintzen duen panelaz egin daiteke horrelako kontrola). RCXra programak transferitzeko diseinatuta dago, horregatik, programa bat editatzen dugunean InitComm eta CloseComm metodoak ahaz ditzakegu, hortaz BickCommand arduratzen baita.

Edizioan gauza pare bat hartu behar dira kontuan:

- metodoen argumentuak parentesi artean idatzi behar dira tarterik utzi gabe.
- Argumenturik ez duten metodoei ere parentesiak erantsi behar zaizkie.

F.2 irudia: BrickCommand

H eranskina: Firmwarearen transferentzia RCXra

PCtik RCXrekin komunikatu ahal izateko, derrigorrezkoa da aurrez firmwarea transferitzea. RCX piztean erlojua ikusten ez bada (hasieran 00.00 balioa erakusten du) eta View botoia ezin bada erabili firmwarea transferitu beharko dugu. Transferentzia LEGO Dacta edo LEGO MindStorms-ekin datorren softwarea erabiliz egin daiteke, edo jarraian eskaintzen diren Visual Basic-eko prozedurez.

H.1 irudia formularioa

Ondoko prozedura egikaritu ondoren eskuratutako testu-katearen azken bost karaktereak 00.00 badira, RCXk ez du firmware-rik.

```
‘ ROM bertsioa eskuratu
Private Sub cmdRomBertsioa_Click()
      lblRom.Caption = PBrickCtrl.UnlockPBrick
End Sub
```

Firmwarea RCXn instalatzeko lehen urratsa transferitzea da:

```
‘ Firmwarea transferitu
Private Sub cmdDownloadFirmware_Click()
      PBrickCtrl.DownloadFirmware "C:\Archivos de Programa\LEGO_
      MINDSTORMS\Firm\firm0309.lgo"[12]
End Sub

‘ Transferentziaren arrakasta
Private Sub PBrickCtrl_DownloadDone(ByVal ErrorCode As Integer, ByVal_
DownloadNo As Integer)
      If ErrorCode = 0 Then
            MsgBox "Firmware behar bezala transferitua", vbInformation
      Else
            MsgBox "Firmware gaizki transferitua", vbCritical
```

[12] Hau, RCX 1.5-i dagokion firmware bertsioa da. RCX 2.0ri dagokion firmwarea firm0328.lgo da, baina arazoak ematen ditu metodo honen bitartez transferitzeko.

```
      End If
End Sub

Private Sub Form_Load()
      PBrickCtrl.InitComm
End Sub

Private Sub cmdIrten_Click()
      PBrickCtrl.CloseComm
      End
End Sub
```

Transferentziak minutu batzuk iraunduko du, eta bukatzen denean mezu bat agertuko da monitorean. Orain firmwarea ireki behar da. Horretarako erabil ezazu ondoko prozedura:

```
‘Firmwarea ireki
Private Sub cmdIrekiFirmware_Click()
      lblFirmware.Caption = PBrickCtrl.UnlockFirmware("Do you byte, when_
 I knock?") ‘ Oharra: testu hau ezin da aldatu
End Sub
```

LblFirmware etiketak ondoko testua erakutsi behar du:

“This is a LEGO Control OCX communicating with a LEGO PBrick!”

Kale egiten badu, testua hau izango da:

“Unlock failed”

Orain RCX prest egongo da bertan programak gordetzeko.

Aurkibidea