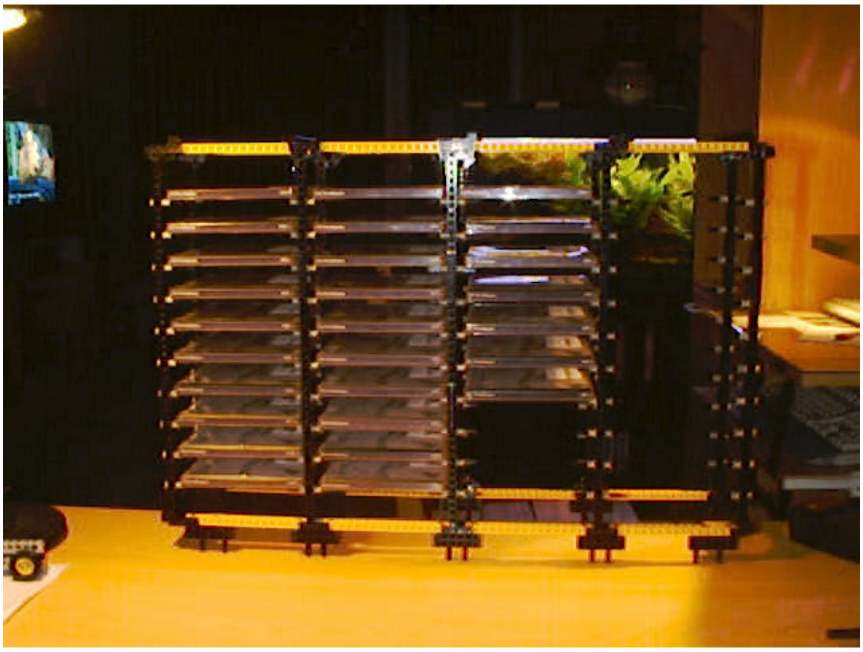
ENHANCED CD-WALL



Introduction:

CD-WALL is my first real MindStorms creation.
In my childyears I played a lot with Lego (from duplo to basic, and also the trains).
When I was 10 years old, I received my first technic sets, But when I was 13 years old, my full attention went to the computer. So that's how I became a software-developer.
During my lifetime I didn't lost my interest in technics and electronics, and that's why I developped My own home control system. I designed electronic circuitboards and attached every door, window, light and electronic device that you can imaging to my PC.
And then, suddenly, when I was 24 years old, I saw a Lego CyberMaster kit in the store.
I was directly in love with it. I went home and I spended some time on the internet to find out everthing about the CyberMaster. A few days later I went back to the store and then I bought my CyberMaster.
A week after that I bought a few other technic sets, and I dismanteld my Cybermaster. I have taken out the internal motors and modified it so you could attach 4 external motors.
I came already early in my mind that CyberMaster wasn't the ideal solution for me, My big wish was to buy a MindStorms set, but they were not available at that time in Belgium (Europe).
And then suddenly, half november 1999, MindStorms finally got to Belgium. (Jieepieee)
I bought my first set, and in my opinion I was the first here in Belgium that actually owned a MindStorms set.
A few weeks later I bought another MindStorms set, and also the Droid Development Kit.
Actually, the Droid Development Kit was for my wife, because she loves Star Wars.
From the beginning it was clear to me that MindStorms offers a lot of possibilities, and my mind was full of interesting ideas. Because I'm a softwaredevelopper I had a hard time at my work because the MilleniumBug, and I had to work a lot of extra hours. Last weekend I finally started building, and the CD-WALL is actually my first good worked-out MindStorms project.

This document will give you a lot of detailled pictures, accompanied with comment from me, so please feel free to distribute this document or take parts of it as a reference.

If you have some hot ideas that you can't figure out how to build, then please feel free to contact me, so I can help you. Also if you have made something and can't figure out how to program it, just ask me and I will help you out.

Enjoy and a happy building with MindStorms to all of you ;-)
Greetz,
Snoozy

http://**www.snoozysplace.be.tf**
mailto: **snoozysplace@be.tf**

Index:

General Overview:

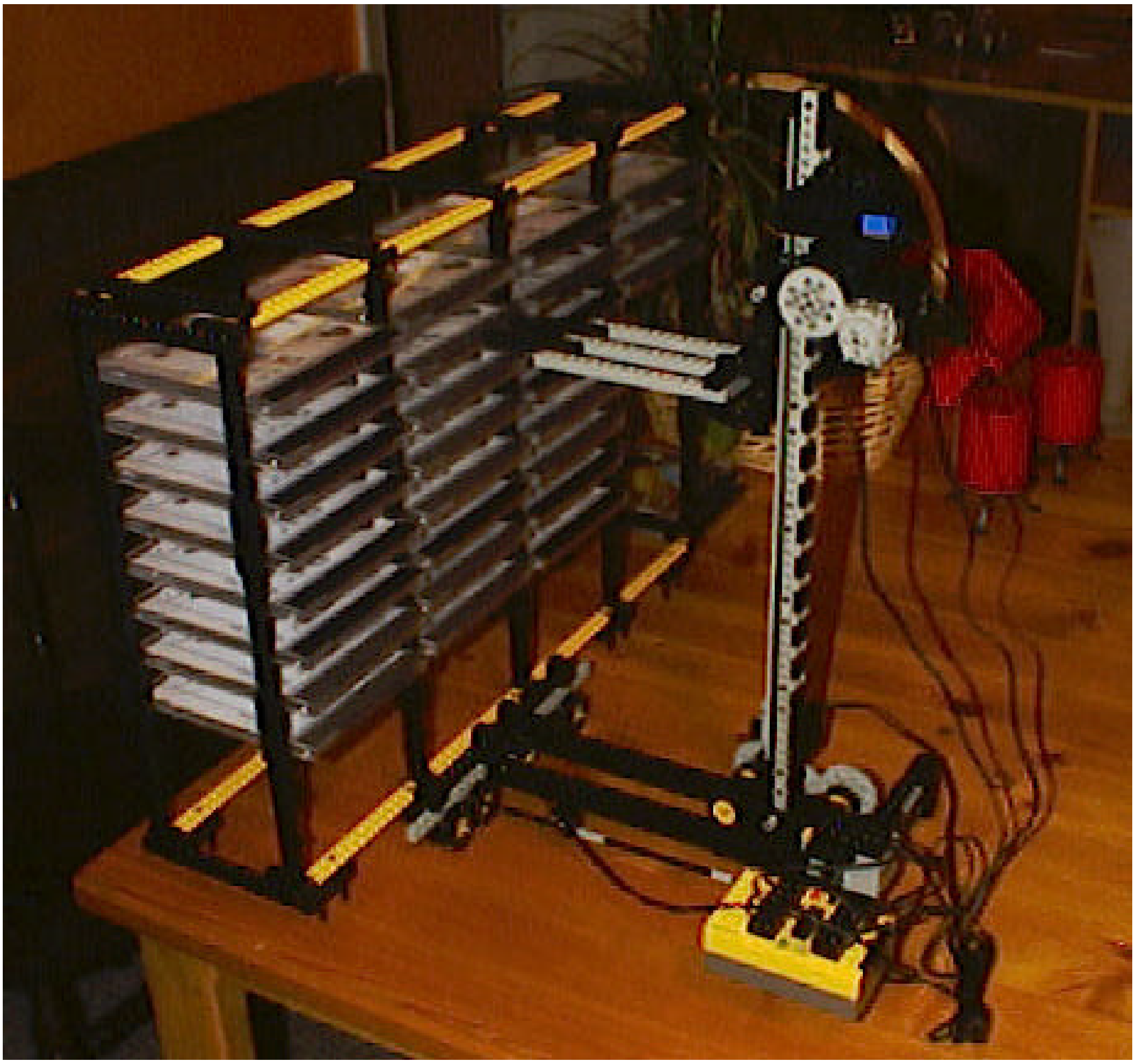


Fig. 1

(Fig. 1) This is a picture of the first version of the CD-WALL. It is actually also the only picture that I own of this first version, because just before I wanted to start taking pictures, it smashed the ground.

The general idea of this project is that if you have a lot of CDs, it always takes time to locate something. Even if your CDs resides in a DataBase, you still have to search for them in your rows and rows of CDs. This machine will take care of that for you. It all starts when you enter a new CD. You put it on the fork, and you fill in all his data in the DataBase-application, the PC knows where you have free spots in your rack, and will command the RCX to place the CD in an open slot. From this moment on, the PC and RCX knows where a particular CD is. You need to do this for all your CDs.
Afterwards you can easily look up a specific song and/or artist in the DataBase. If you find what you need in the DataBase, you can simply command the RCX via the software to “deliver” you this CD.
Because the Row and Column where the CD resides is know by the PC, it is perfectly possible to give you the needed CD without the need of searching thru you collection of CDs.

So we basically need 2 things: a place to store the CDs, this will be our CD-Rack, and a mechanism to put and retrieve the CDs from the Rack automatically.

CD-Rack Overview:

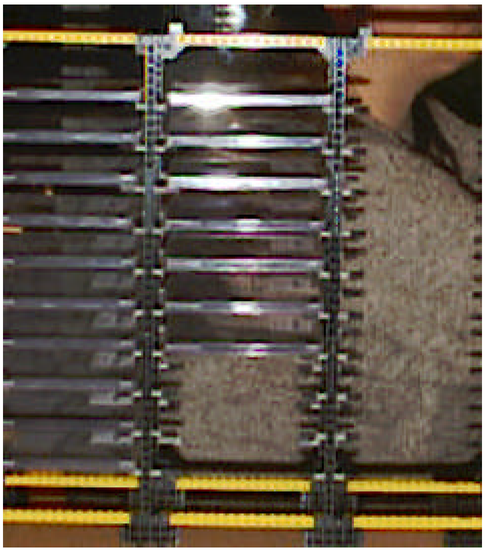


Fig. 2

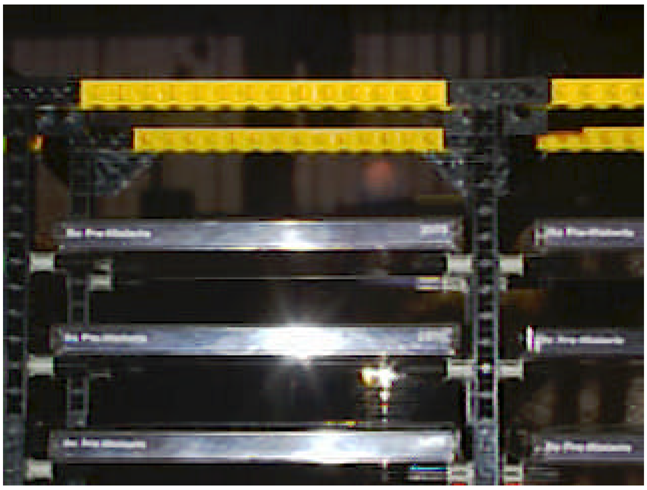


Fig. 3

In Fig. 2 you can see a general overview of a complete rack. (I have 4 racks mounted together)
In Fig. 3 you can see that on the left and right of the CD, there is still some space left, this is to prevent that a CD is getting stuck when it is inserted in the slot via the forklift mechanism.
With a big construction like this, you can't actually work accurate enough, so this is my solution for this problem. You will also see that the forklift is a few studs smaller then the pinions where the CD rests on, this is also done to prevent CDs getting stuck.
In Fig. 4 you have a topview of a CD-Rack, you can see that the CD is a bit out of the rack on the front and on the back, this is also to have a small marge so the CDs won't drop op the pinions.

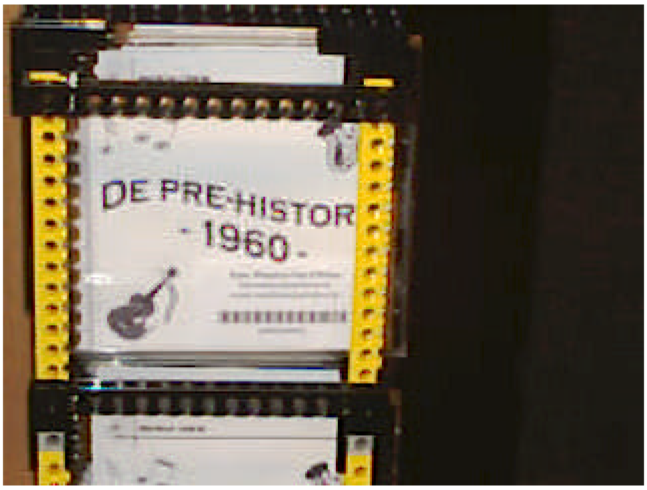


Fig. 4

Forklift Mechanism Overview:

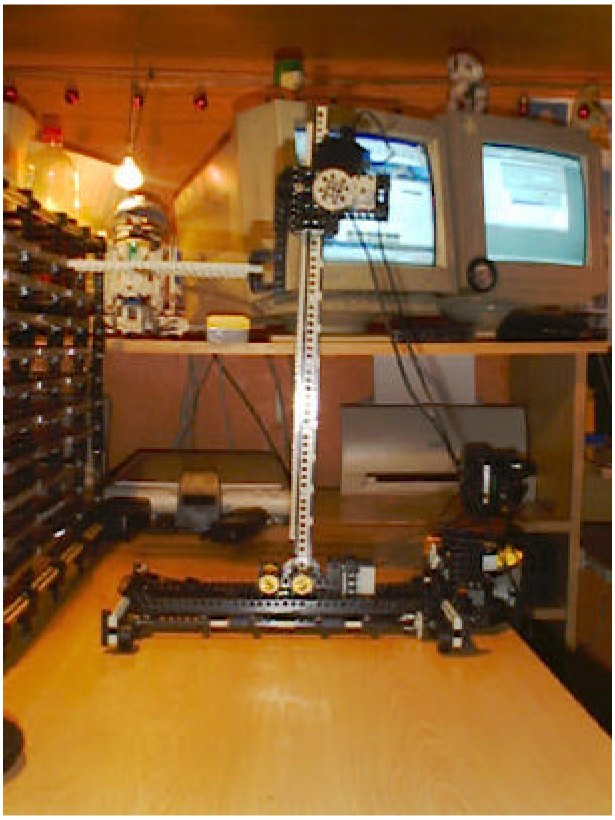


Fig. 5

Here you see a complete picture of the Forklift mechanism.
It has 4 main functions:

1. Move left / right
2. Move forward / backward
3. Move up / down
4. Tilting CD / Dropping CD

For every function is a motor needed, but you must notice that function 3 & 4 use the same motor.
The RCX must also be able to control every function accuratly, this is ofcourse done by sensors.
In fact, one RCX is enough to control the whole CD-WALL.
On the next pages you will find a detailled description of every function and the accompanied Lego structure needed.

Function 1 : Left / Right Movement:
(This function is used to select a CD-Rack - column)

For the Left / Right Movement, there are techically spoken 2 problems to solve:

1. Going ‘straight’ left / right
2. Accuratly positioning before every CD-Rack

The First problem is solved by also ‘powering’ the wheels on the other side of the forkliftmechanism. You can’t use another motor for this, because there is a tolerance on every motor, so they actually never have precisely the same speed. You actually need to transfer the speed and torque from one side to another, using one single motor. In Fig. 6 you can see the motor on top, that divides his speed and torque to both sides of the sleigh that moves forward and backward, you also see long axles connected to it. In Fig. 7 you see the wheels connected to the long axles, on the other side of the construction.

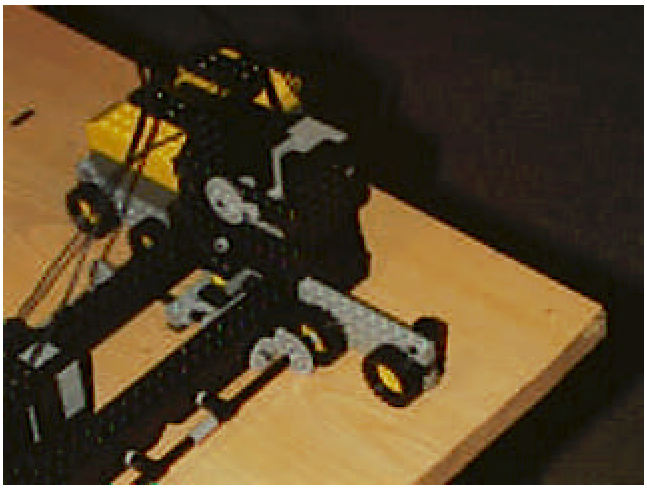


Fig. 6

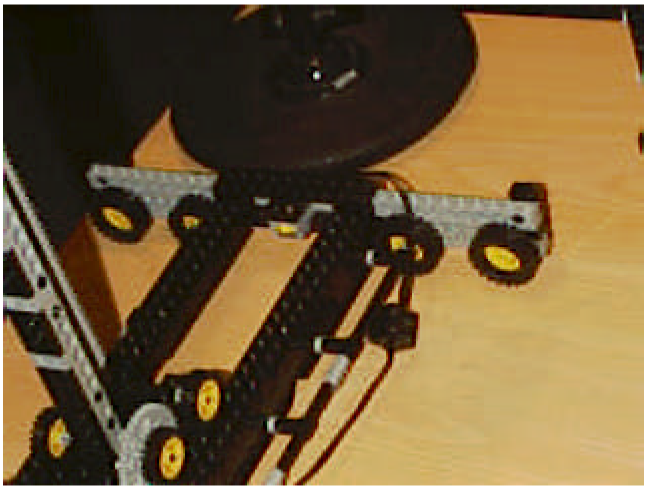


Fig. 7

For the second problem, I mounted a lightsensor just behind the motor for the left/right movement, witch I use in combination with black thin markers on the floor on witch the forkliftmechanism moves. In Fig.8 you see the markers. Fig. 9 is a bottomview of the lightsensor, you may notice that it is completely surrounded by black bricks, to block off external lightsources that could cause false readings.

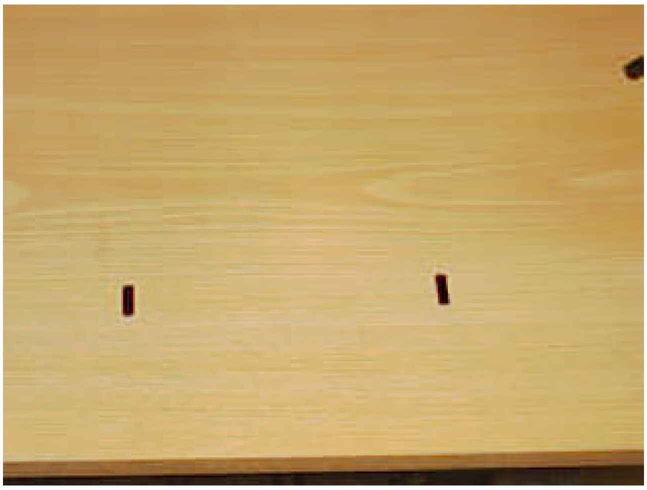


Fig. 8

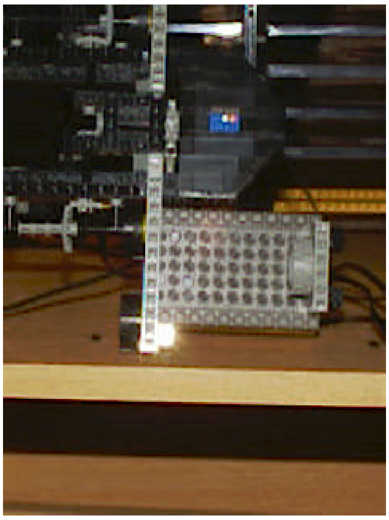


Fig. 9

Function 2 : Forward / Backward Movement:
(This Function is used to transport the CD in and out the CD-Rack)

Technically there are not much problems to solve this, but you need to take care of 2 aspects:

1. Find a way to let the RCX knows when it reached the end or beginning of the sleigh.
2. The distance that the forklift can move forward/backward needs to be larger then the length of a CD.

To solve the first aspect, you simply put a touchsensor at both ends of the sleigh.
I connected my touchsensors to one inputport. In my program I have built-in a little pause between the moment that the forklift begins to roll and the moment that the RCX starts polling if the forklift already is on the other side of the sleigh. In Fig.6 & Fig. 7 you can see where the touchsensors are located.
In Fig. 10 you can see the actual ridingmechanism of the forklift to move backwards and forwards.
On the front I have mounted 2 extra wheels, so the up/down sleigh will not bend forward when there is a
CD on the forklift.

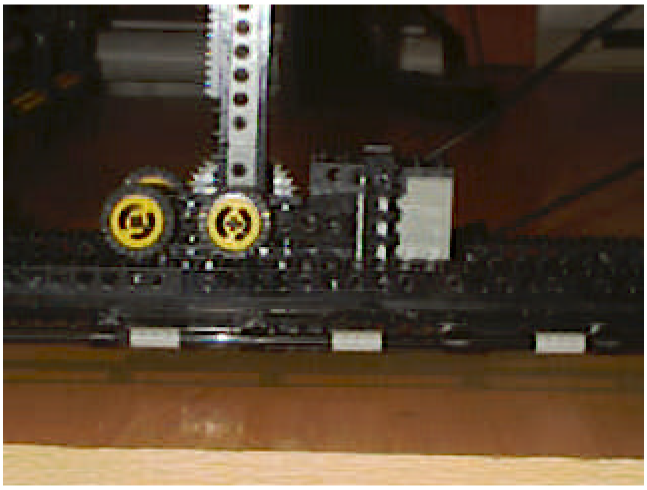


Fig. 10

Function 3 : Up / Down Movement:

When making the Up / Down sleigh, you must be aware of 1 problem:

1. Accuratly positioning.

To solve this problem, The RCX needs to know when the forklift is precisely underneath a CD, so it can move forward and lift the CD, or can move forward and drop a CD.
I have put gray pieces on the back of the up/down sleigh, and with a lightsensor the RCX can notice when the forklift is in position (or counting when he needs to pass several positions), you can see this in Fig. 11.
You must also take care that the motor is geared down enough, to have enough torque, so it can have the same speed when moving up / down with / without a CD (see Fig. 12 & 13).
For good grip and accurat working, I used flat toothplates on the front of the up/down sleigh. (see Fig. 12 & 13)
In Fig. 14 you have a closeup of the fork itself.

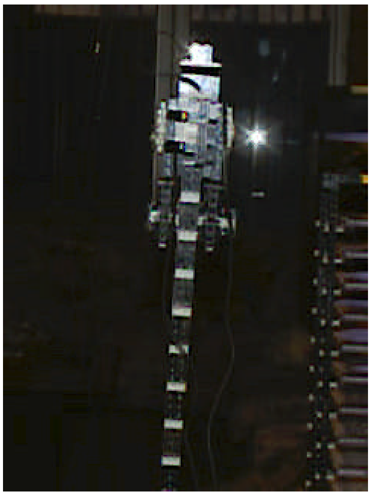


Fig.11

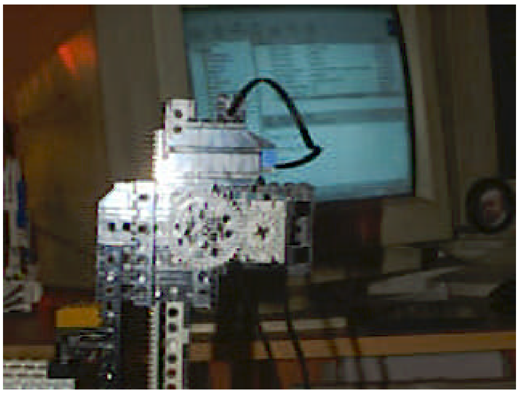


Fig. 12

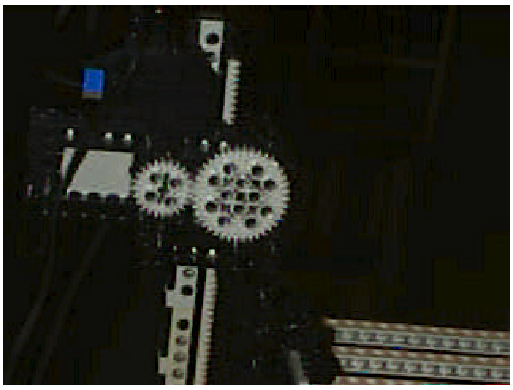


Fig. 13

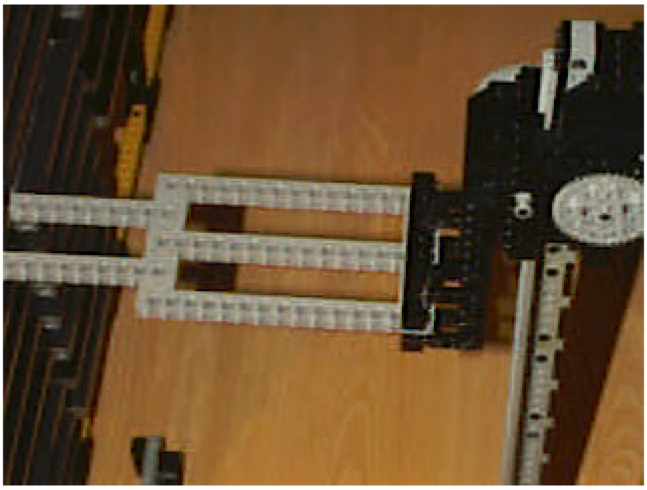


Fig. 14

Working Process: General overview:

What You actually on software-side is:

1. The NQC program for the RCX
2. The Visual Basic application

1. *The NQC program (listing is on page 12 and further)*

This is a simple program that accept SendMessages from the IR-Tower, that where send from the Visual Basic Application.

Here is an overview on what it does when it receives certain messages:

Message Number	Description
1	The RCX will take care that the forklift will move UP exactly one position
2	The RCX will take care that the forklift will move DOWN exactly one position
3	The RCX will take care that the forklift mechanism move LEFT exactly one complete row of CDs
4	The RCX will take care that the forklift mechanism move RIGHT exactly one complete row of CDs
5	The RCX will take care that the forkliftmechanism will move FORWARD, so it will actually stick its fork in the CD-Rack
6	The RCX will take care that the forklift mechanism will move BACKWARD, so it will actually go back outside the CD-Rack, so it can move freely to another position
7	The RCX will lift up the CD that is just above the forklift
8	The RCX will drop the CD in the CD-Rack, on the position where the forklift is in the CD-Rack

On the next page you will find a description on every action that you can do with the CD-Wall.

Working Process: possible actions:

1. *Add a CD*
   You actually start with a complete empty DataBase.
   Every time you add a CD in the software, the Visual Basic Application always knows where the next empty slot is. You add the title of the CD, all songs with there name, artist, songnumber and duration. After the Complete CD is filled in, into the DataBase, the Visual Basic Application will give it also a rownumber and a columnnumber. The Forkliftmechanism will move to the topleft position, and You will be asked to put the CD on the fork. The VisualBasic Application will then start with sending Messages to the RCX until the CD is into an empty slot.

2. *Delete a CD*
   If you want to completely remove a CD from the system, without ever returning it into the CD-WALL, you can mark a CD that it is no longer in use. His data will then be removed from the DataBase, and the slot will be marked as empty and ready for use.

3. *Edit a CD*
   If you find some mistakes into your DataBase, you can simply Edit the information from a selected CD and overwrite the old info in the DataBase.

4. *Find a song*
   You search for a song via a built-in filter into the Visual Basic Application.
   If you find a song, and you ask to “deliver” the CD, the Visual Basic Application will calculate the position of the CD in function from where the forklift is “parked” now, and will move to that position, take the CD out of the CD-Rack, and move to the topleft position, in the meanwhile the PC will show you the complete info of the CD that you have selected, and also displays the songnumber that you where looking for. It will also give you a unique ID for the CD, you must write this down on a piece of paper, because you will need this number to re-enter the CD into the CD-Rack.

5. *Re-enter a CD back into the CD-Wall*
   You type in the unique ID that the Visual Basic application delivered upon retrieving the CD from the CD-Wall. The forkliftmechanism will then move to the topleft position, you put the CD on the fork, and it will be placed back into the slot that was reserved for this CD.

NQC listing:

I will give you a complete description of the software, and also a listing of the necessary NQC program.

When the Visual Basic application finds the necessary information in his DataBase, and you “ask” to “deliver” that certain CD, the only thing the VisualBasic application then does is steering the mechanism via Sendmessages();
I knows always what the actual position of the mechanism is, and what the needed position is.
It Always calculates the vertical position first, and afterwards the horizontal position.

For example, we are at row 10, column 3, and we need to go to column 2, row 4, the Visual Basic Application will send 6 UP messages and 1 LEFT message.

NQC code:

```
task main()
{
SetSensor(SENSOR_1,SENSOR_LIGHT);
SetSensor(SENSOR_2,SENSOR_LIGHT);
SetSensor(SENSOR_3,SENSOR_TOUCH);
SetSensorMode(SENSOR_1,SENSOR_MODE_PERCENT);
SetSensorMode(SENSOR_2,SENSOR_MODE_PERCENT);
SetSensorMode(SENSOR_3,SENSOR_MODE_BOOL);
SetPower(OUT_A,2);
SetPower(OUT_B,3);
SetPower(OUT_C,2);

// Here starts loop ...
while (true)
{
  // Move one row Up ...
  if (Message()==1)
    {
    ClearMessage();
    OnRev(OUT_B);
    Wait(80);
    while (SENSOR_2 < 32)
    { }
    Wait(50);
    Off(OUT_B);
    }
```

```
// Move one row Down ...
  if (Message()==2)
    {
    ClearMessage();
    OnFwd(OUT_B);
    Wait(80);
    while (SENSOR_2 < 32)
      { }
    Off(OUT_B);
    }
  // Move one column Left ...
  if (Message()==3)
    {
    ClearMessage();
    SetPower(OUT_A,2);
    OnFwd(OUT_A);
    Wait(60);
    SetPower(OUT_A,1);
    while (SENSOR_1 > 48)
      { }
    SetPower(OUT_A,2);
    Off(OUT_A);
    }
  // Move one column Right ...
  if (Message()==4)
    {
    ClearMessage();
    SetPower(OUT_A,2);
    OnRev(OUT_A);
    Wait(60);
    SetPower(OUT_A,1);
    while (SENSOR_1 > 48)
      { }
    SetPower(OUT_A,2);
    Off(OUT_A);
    }
  // Move towards CD-Rack ...
  if (Message()==5)
    {
    ClearMessage();
    OnFwd(OUT_C);
    Wait(10);
    while (SENSOR_3 == 0)
      { }
      Off(OUT_C);
      }
```

```
 // Move away from CD-Rack ...
 if (Message()==6)
  {
  ClearMessage();
  OnRev(OUT_C);
  Wait(10);
  while (SENSOR_3 == 0)
   { }
   Off(OUT_C);
   }
 // Lifting CD ...
 if (Message()==7)
  {
  ClearMessage();
  OnRev(OUT_B);
  Wait(140);
  Off(OUT_B);
  }
 // Dropping CD ...
 if (Message()==8)
  {
  ClearMessage();
  OnFwd(OUT_B);
  Wait(90);
  Off(OUT_B);
  }
 }
}
```