ISRN UTH-INGUTB-EX-E-2010/02-SE

Examensarbete 15 hp
September 2011

The Memec Hexapod Robot

a demonstration platform

Fredrik Persson
Mattias Lindström

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

The Memec Hexapod

Fredrik Persson & Mattias Lindström

This thesis shows how to replace a microcontroller unit (MCU) on a six legged robot, involving adapting hardware and developing software. The robot is based on the mechanics from Lynxmotion's Phoenix Hexapod Robot, which is a mechanical vehicle with 6 legs and 18 servos. The robot is controlled by two controller boards, which with available software can run pre-made movement sequences or be controlled by a Playstation2 hand controller. One of the controller boards needs to be replaced by another more powerful MCU from another manufacturer. The new MCU comes from Avnet-Memec, which is a major emiconductor components distributor. They want to use the Hexapod Robot as a platform for demonstrating the capabilities of the franchises on their line card to attract potential customer's interest and awareness. The original Basic Atom Pro 28 MCU needs for that purpose to be replaced with hardware from an Avnet-Memec franchise. Avnet-Memec has decided to use the circuit board MBS270 from Mobisense Systems based on the ARM-processor Marvel PXA270 running Embedded Linux operating system. The project was successful and resulted in a Phoenix Hexapod Robot with a new more powerful and versatile MCU from an Avnet-Memec franchise. The robot can run pre-made movement sequences as with the old MCU, but now with a much more powerful processor with support for features like wifi, video acquisition and much more. The robot was shown by Avnet-Memec at the Scandinavian Electronics Event fair 2010 in Älvsjö, Stockholm, Sweden, and was appreciated among visitors and other exhibitors.

Handledare: Johan Enoksson
Ämnesgranskare: Lars Ericsson
Examinator: Nora Masszi
ISRN UTH-INGUTB-EX-E-2010/02-SE

Sammanfattning

Avnet-Memec, en av världens största distributörer av halvledar-komponenter, har ett behov av att demonstrera kapaciteten hos sina leverantörers produkter. För att göra detta vill de låta styra en robot med produkter från Avnet-Memecs leverantörer. Roboten skall baseras på mekanik från den sexbenta The Phoenix Hexapod Robot från Lynxmotion, som rör sig framåt med 18 servon styrda av två mikrodatorer. En av mikrodatorerna skall ersättas med en mikrodator från Avnet-Memecs leverantör Marvel. I robotens originalutförande kan den med befintlig programvara röra sig enligt förprogrammerade rörelsemönster eller styras med en Playstation2 handkontroll. Målet med examensarbetet är att roboten skall fungera på samma sätt med den nya mikrodatorn från Marvel. Utöver detta vill Avnet-Memec också låta programmera en egen rörelsesekvens. Projektet delas upp i tre steg; Ersättning av hårdvara, programmering av rörelsesekvenser, samt addering av diverse funktioner som t.ex. styrning med Playstation2 handkontroll. De två första delarna utfördes, medan den sista inte hann slutföras helt på grund av tidsbrist. Roboten kunde med den nya mikrodatorn efter projektets slut röra enligt den generella standarden för rörelsesekvenser hos Hexapoder och visades upp av Avnet-Memec på Scandinavian Electronics Event 2010 i Älvsjö.

Table of contents

Figures

1 Introduction

Avnet-Memec is a major semiconductor components distributor with a specialized and highly technical line card. They have a need to demonstrate the capabilities of the franchises on the line card to attract potential customer's interest and awareness. A demonstration platform will be built based on the mechanics from the Phoenix Hexapod Robot from Lynxmotion, see figure 1 for illustration. It uses 18 servos, 3 for each leg, and is equipped with a standard servo controller board and a host controller board. It is programmed through a serial bus or USB.

The first step in this thesis project is to replace the standard host controller board with a board with a microcontroller from a Memec franchise (Silabs, Zilog, Marvel). The software for controlling and programming the Hexapod will need to be developed.

The second step will be to program the Hexapod with certain moving sequences that will be used for demonstration and promotion.

The third step is to develop features to the Hexapod, such as connecting a playstation2 hand control, camera to send and receive video over WLAN, Ultrasound anti-collision, robot sound generation through loudspeaker. This third step may include development and assembly of required hardware in case standard hardware is not available. The priority is to make the hand control to work.

Disposable time for this project is 10 weeks for two people including writing this report.

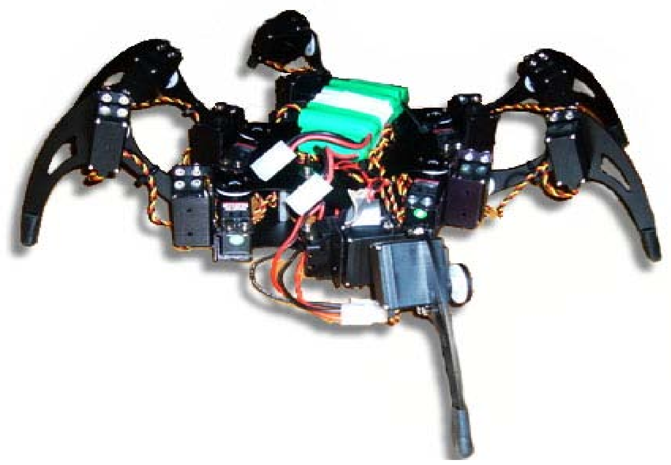


Fig. 1 The Phoenix Hexapod Robot

2 Aim of the project

The company Avnet-Memec wanted to demonstrate their components on a science fair with the help of the six legged Phoenix Hexapod Robot. The microcontroller that was attached to the robot in the original setup was meant to be replaced with a small Linux based ARM computer from Mobisense Systems, a computer which Avnet-Memec's distributes. The result that was desired by our employer was also to make the robot to be controlled by a Playstation hand controller and to make an automatic movement sequence that could attract customer to Avnet-Memec's section at the fair.

3 The Phoenix Hexapod Robot

The Phoenix Hexapod Robot is a mechanical vehicle that walks with six legs. It has an aluminum body and is equipped with 18 servos, three on each leg, all controlled and synchronized by two controller boards. Each leg has three degrees of freedom which gives the robot a great deal of flexibility in how it can move. The robot is modular, which allow you to choose among different servos and controller boards. Our robot was initially equipped with:

- Basic Atom Pro28-M microcontroller
- Bot Board II carrier board for the Basic Atom
- SSC-32 Servo Controller
- HS-645MG Servos

Neither one of the microcontrollers is delivered with software, but Lynxmotion refers to Kåre Halvorsen and Jeroen Janssen. They are two robot enthusiasts and creators of software for controlling the hexapod robots with a Nintento Wii remote or a Playstation2 controller. Halvorsen also offer some pre-made movement sequences for download[10].

4 Movement sequences

A more or less official standard exists when it comes to storing movement sequences for hexapod robots. Csv-files with semicolon-separated values are used to store angles for each and every servo in every step. There are two major Windows program for handling these files;

- Lynxmotion Visual Sequencer is a Windows program for controlling up to 32 servos using the SSC-32 servo controller board. The program, which can be bought from Lynxmotion's website, lets you make your own movement sequences or import others. Everything is stored in csv-files. Finally the Visual Sequencer generates Basic Atom code and programs the microcontrollers.

- Phoenix Excel Program (PEP) is a Microsoft Excel spreadsheet made by the robot enthusiast Kåre Halvorsen. It can be downloaded for free from Lynxmotion’s website and is a big help for generating complex moving and walking sequences for hexapod robots. PEP lets you position the robot in terms of x, y and z-coordinates and without having to position each servo one by one. The spreadsheet also contains some pre-made sequences for such as raise/lower body, rotations and various walking gaits. After you are done making your sequences you still need the Visual Sequencer for generating the BASIC code and programming the microcontrollers.

5 The first step – replacing controller board

Avnet-Memec wants to have the BasicAtom Pro28 and its carrier board replaced with products from their own franchises. They have chosen the board MBS270 from Mobisense Systems which is based on a Marvel PXA270 V2 processor. The PXA270 offers a lot of development possibilities with its 520 Mhz ARM-processor, 64 MB SDRAM and 32 MB Flash. It runs with Linux kernel 2.6.26, bringing high performance and much functionality to embedded applications. It can communicate in many ways such as UART, USB, Bluetooth and Ethernet and it has a preinstalled video server to mention some of the features.

5.1 Adapting electronics

The Hexapod's servo controller board and host controller board communicates using serial communication. To configure this we need to install a couple of jumpers on the SSC-32, see SSC-32 manual [7] for further details. We want to enable DB9 serial port and configure baud rate. We choose 38.4k, the same baud rate as the original Basic Atom uses. The PXA270 requires 3.5 – 4.2 volts with its original power module and the battery voltage is 7.2 volt. Therefore we needed a different power module that had an input ranging from 6 to 30 volt. This made it possible to use the onboard battery pack. We temporarily used an external adjustable power supply while waiting for the real power module to arrive after ordering it. With a steady power source we were not depending on the capacity of the batteries.

5.2 Software deployment

5.2.1 Development toolchain

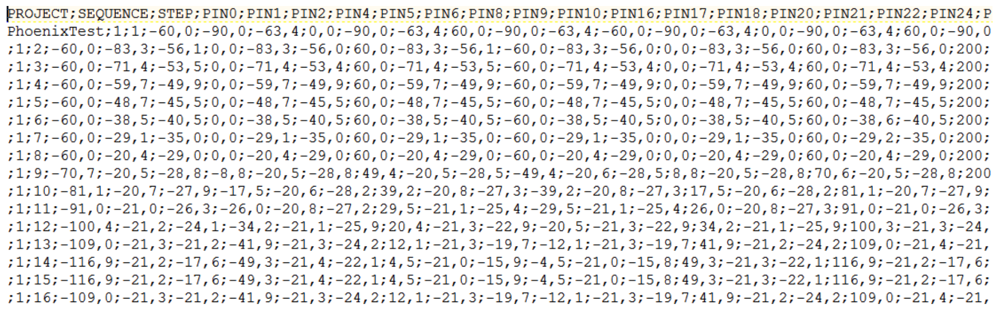
When developing software for a different platform than the PC, a set of computer programs is needed. It is called a development toolchain which consists of a compiler, a linker, a set of libraries, a text editor for writing source code and a debugger. The toolchain recommended for the PXA270, and which we used in this project, is "arm-linux-gnu-eabi". This is for compiling code for the ARM-architecture on embedded computers. The compiler is GCC, which stands for the GNU Compiler Collection, and can be used for various languages such as Ada, C, C++, Fortran, Java and Objective-C. It also includes system libraries needed for ARM-architecture. A linker is for the final step of the compilation and links the object files, the system libraries and the start up files to create executable binaries. See section 10.1 for detailed toolchain installation instructions.

5.2.2 Transferring files using SSH and SCP

SSH stands for secure shell and is a network protocol used for exchanging information between two devices over a network. It uses public-key cryptography and is mainly used in UNIX systems for executing commands from a remote machine in a shell on the target. Secure Copy, or SCP, is based on the SSH protocol, and is a way to securely transferring computer files between a local and a remote host. We used SCP because this is an easy way to transfer files. See section 10.2 for detailed instructions.

5.2.3 Read the servo angles represented in a csv-file

We are going for a modular solution which means the csv-files and the reading software are separated. This lets you change csv-files without recompiling your program. Our "csv-handler" software needs to manage three main things. The first thing is to read the servo angles represented in a csv-file. The csv-files generated from the Visual Sequencer and the Phoenix Excel Program both share the same structure. They contain servo angles, pin number (servo number), movement time, project name, sequence names, step numbers and comments, as shown in figure 2. Our software needs to focus on what is relevant for the SSC-32 commands, in this case servo angle (in degrees) and movement time (in ms) to corresponding pin number. Fig. 2 shows an example of a csv-file. In Appendix 1 you can see a more complete and detailed one.



```
PROJECT;SEQUENCE;STEP;PIN0;PIN1;PIN2;PIN4;PIN5;PIN6;PIN8;PIN9;PIN10;PIN16;PIN17;PIN18;PIN20;PIN21;PIN22;PIN24;P
PhoenixTest;1;1;-60,0;-90,0;-63,4;0,0;-90,0;-63,4;60,0;-90,0;-63,4;-60,0;-90,0;-63,4;0,0;-90,0;-63,4;60,0;-90,0
;1;2;-60,0;-83,3;-56,1;0,0;-83,3;-56,0;60,0;-83,3;-56,1;-60,0;-83,3;-56,0;0,0;-83,3;-56,0;60,0;-83,3;-56,0;200;
;1;3;-60,0;-71,4;-53,5;0,0;-71,4;-53,4;60,0;-71,4;-53,5;-60,0;-71,4;-53,4;0,0;-71,4;-53,4;60,0;-71,4;-53,4;200;
;1;4;-60,0;-59,7;-49,9;0,0;-59,7;-49,9;60,0;-59,7;-49,9;-60,0;-59,7;-49,9;0,0;-59,7;-49,9;60,0;-59,7;-49,9;200;
;1;5;-60,0;-48,7;-45,5;0,0;-48,7;-45,5;60,0;-48,7;-45,5;-60,0;-48,7;-45,5;0,0;-48,7;-45,5;60,0;-48,7;-45,5;200;
;1;6;-60,0;-38,5;-40,5;0,0;-38,5;-40,5;60,0;-38,5;-40,5;-60,0;-38,5;-40,5;0,0;-38,5;-40,5;60,0;-38,6;-40,5;200;
;1;7;-60,0;-29,1;-35,0;0,0;-29,1;-35,0;60,0;-29,1;-35,0;-60,0;-29,1;-35,0;0,0;-29,1;-35,0;60,0;-29,2;-35,0;200;
;1;8;-60,0;-20,4;-29,0;0,0;-20,4;-29,0;60,0;-20,4;-29,0;-60,0;-20,4;-29,0;0,0;-20,4;-29,0;60,0;-20,4;-29,0;200;
;1;9;-70,7;-20,5;-28,8;-8,8;-20,5;-28,8;49,4;-20,5;-28,5;-49,4;-20,6;-28,5;8,8;-20,5;-28,8;70,6;-20,5;-28,8;200
;1;10;-81,1;-20,7;-27,9;-17,5;-20,6;-28,2;39,2;-20,8;-27,3;-39,2;-20,8;-27,3;17,5;-20,6;-28,2;81,1;-20,7;-27,9;
;1;11;-91,0;-21,0;-26,3;-26,0;-20,8;-27,2;29,5;-21,1;-25,4;-29,5;-21,1;-25,4;26,0;-20,8;-27,3;91,0;-21,0;-26,3;
;1;12;-100,4;-21,2;-24,1;-34,2;-21,1;-25,9;20,4;-21,3;-22,9;-20,5;-21,3;-22,9;34,2;-21,1;-25,9;100,3;-21,3;-24,
;1;13;-109,0;-21,3;-21,2;-41,9;-21,3;-24,2;12,1;-21,3;-19,7;-12,1;-21,3;-19,7;41,9;-21,2;-24,2;109,0;-21,4;-21,
;1;14;-116,9;-21,2;-17,6;-49,3;-21,4;-22,1;4,5;-21,0;-15,9;-4,5;-21,0;-15,8;49,3;-21,3;-22,1;116,9;-21,2;-17,6;
;1;15;-116,9;-21,2;-17,6;-49,3;-21,4;-22,1;4,5;-21,0;-15,9;-4,5;-21,0;-15,8;49,3;-21,3;-22,1;116,9;-21,2;-17,6;
;1;16;-109,0;-21,3;-21,2;-41,9;-21,3;-24,2;12,1;-21,3;-19,7;-12,1;-21,3;-19,7;41,9;-21,2;-24,2;109,0;-21,4;-21,
```

Fig. 2 Example of a csv-file displayed in plain text

5.2.4 Convert data to SSC-32 format

The second thing our csv-reading software needs to do is to convert the values in the csv-files into commands for the SSC-32 servo controller board. Pulse-proportional servos, as used in this project, uses signals that consist of positive going pulses ranging from 0.5 to 2.5 ms long, repeated 50 times a second. The servos position their output shafts in proportion to the width of the pulse and the SSC-32 together with our servos (HS-645MG) have a range of up to 180°. The minimum position value 500 corresponds to 0.50ms pulse (-90°), and the maximum position value 2500 corresponds to a 2.50ms pulse (+90°). The servos are centered at a position value of 1500. A one unit change in position value produces a 1us (microsecond) change in pulse-width, which represents 0.09°. Hence the positioning resolution is 0.09° (180°/2000). See fig. 3.

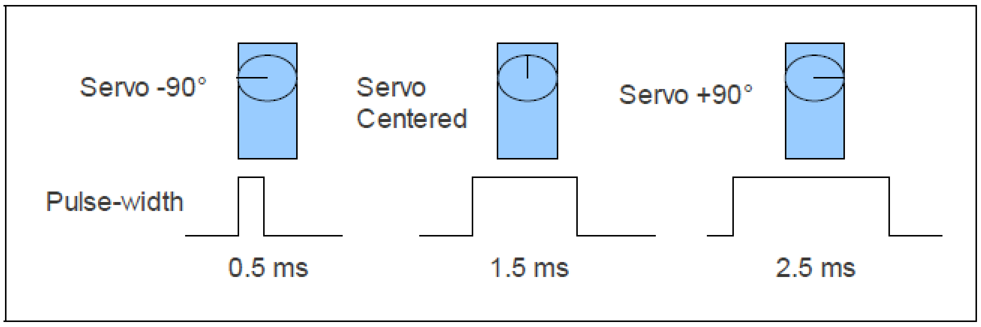


Fig. 3 Pulse-width and corresponding servo angles.

The SSC-32 commands needs to be formatted as follows;

Servo Move or Group Move:

# <ch> P <pw> S <spd> … # <ch> P <pw> S <spd> T <time> <cr>

<ch>	=	Channel number in decimal, 0 – 31.
<pw>	=	Pulse width in microseconds, 500 – 2500.
<spd>	=	Movement speed in µs per second for one channel. (Optional)
<time>	=	Time in ms for the entire move, affects all channels, 65535 max. (Optional)
<cr>	=	Carriage return character, ASCII 13 (Required to initiate action)
<esc>	=	Cancel the current command, ASCII 27.

Servo Move Example: "#5 P1600 T1000 <cr>"

The example above will move servo number 5 to position 1600. It will take 1 second to complete the move regardless of how far the servo has to travel to reach the destination. All commands are executed with a carriage return character (ASCII 13).

When calculating the correct pulse-width we use the following equation;

$$\boxed{\textit{PWM Value} = \textit{0 degree PWM value} + \textit{degree value} / \textit{pos. resolution}}$$

where PWM value = calculated pulse-width in milliseconds, 0 degree PWM value = the pulse-width when servos are centered (1500 ms), degree value = destination, and positioning resolution = 0.09°.

All angles in the csv-files are in reference to the center of the servos (P1500), i.e the values in the csv-files are the "degree value" in the equation above.

For example: A position of 45° clockwise from center = 1500 + 45 / 0.09 = 2000 ms = P2000

The other thing we need to retain from the csv-file is the movement time, the "T" value. It is written in ms, so this doesn't need recalculation.

5.2.5 Send commands to the SSC-32

The last task for our csv-reading software is to send all the commands above to the SSC-32. There is open source code available for RS232 (serial port) usage in Linux [8] licensed under GPL version 2, which basically means that you are free to change and share it [5]. We use these included functions for sending and reading bytes, but to set up the serial communication on the PXA270 some configuration needs to be done. The baud rate needs to be set to match the SSC-32 as described in section 5.1, 38.4k. In addition to that we need to configure it to communicate with 8 bits, no parity, 1 stop bit and No flow control which are all hard coded settings in the SSC-32.

6 The second step – programming moving sequences

After finishing "The first step" in this thesis project, the PXA270 can now handle csv-files. This means that we can use the Visual Sequencer or the Phoenix Excel Program in exactly the same way as we would do with the original Basic Atom to generate desired movement sequences. We choose to use Phoenix Excel Program (PEP) when creating a demo run sequence because it is free and fairly straightforward when it comes to creating complex movement sequences (see figure 4). Our own developed sequence consists of a walk, rotation, lay down, raise up and some pointing with the front legs. This sequence together with a by us modified version of Halvorsen's Phoenix3 sequence [10] were developed. In PEP you position each leg in terms of x, y z-coordinates and the program automatically calculates each servo csv-value out of that and puts it in a csv-file. For more specific instructions on how to make movement sequences, see the PEP user manual [3].

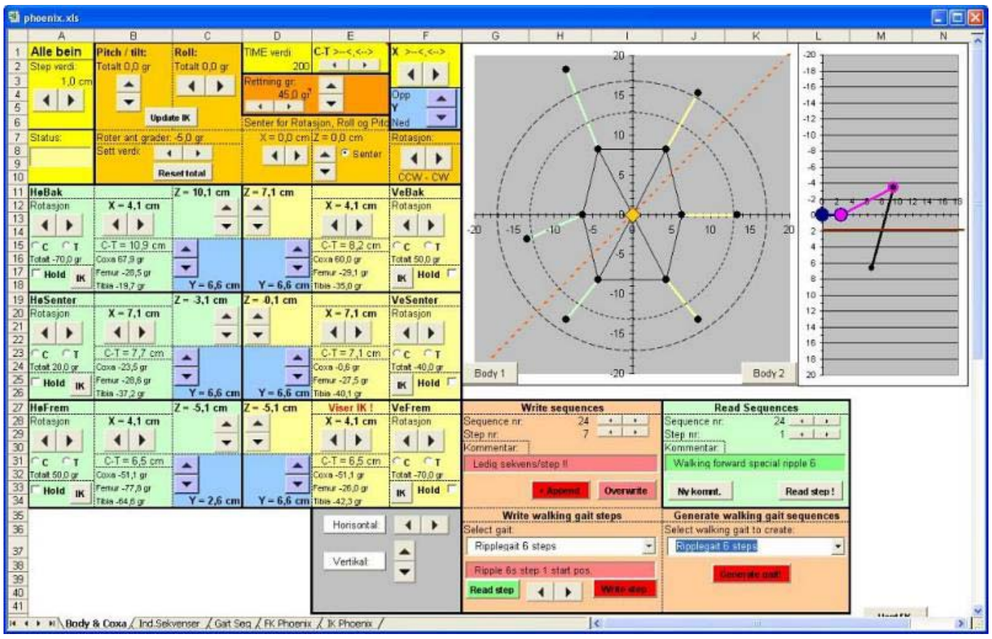


Fig. 4 The Phoenix Excel Program (PEP) created by Kåre Halvorsen.

7 The third step – attaching a Playstation 2 controller

The robot enthusiast Jeroen Janssen has written BASIC Atom Pro code [12] for controlling the Phoenix Hexapod with a Playstation2 controller. Avnet-Memec wants his solution to work with the PXA270 and to accomplish this we need to port Janssen's code. His code is written in mbasic, a programming language particular for the Basic Atom processor family. We need to analyze and understand the mbasic code so we can program the same functionality for our processor. We choose C as programming language in this step as well, and besides the use of standard library functions for math and string operations we also need to communicate with the hardware in terms of…

…using timers to calculate gait speed

Janssen is counting tics per ms in his code when calculating movement speed. 16-bit timers, as found on the MBASIC Pro, is not sufficient for counting this, so Janssen's solution is to count the number of overflows using interrupts. Since the PXA270 has 32-bit timers, this will not be an issue. For instance we can use the Stop Watch operation on the PXA270 (see developer's manual [9] for details)

…using the SPI bus for interfacing with the Playstation 2 controller

Serial Peripheral Interface Bus or SPI is a synchronous serial data link standard that operates in full duplex mode. The Playstation2 interfaces to its controllers using a protocol that basically matches SPI. There is a clock (SCLK) to synchronize bits of data across two channels: Master Out, Slave In (MOSI) and Master In, Slave Out (MISO). MOSI bits are data moving from the master device to a slave device and MISO bits are data moving from the slave out to the master. Additionally there is a slave select (SS) channel — one per slave device — that tells the slave whether or not it is active and should listen to data bits coming across the MOSI channel, or send data bits across the MISO channel.

In addition to the SPI protocol, the Playstation2 also uses an Acknowledge (ACK) line. This is used to acknowledge to the Playstation2 console each "frame" of data. Further details of this protocol can be found at [4].

The programming language mbasic (for the BasicATOM processor) includes simple syntaxes for shifting data bits in and out and setting pins logically high and low. It gets more complex when configuring this for the new processor with Embedded Linux, where the registers have to be set by bitmasks. Polarity and speed of the clock is also things you manually need to configure. We tried to configure the SPI according to the PXA27x Developer's Manual [9] and modified a code written in C for the PXA-270 [11] published in GPL.

After connecting the wires of the controller to the MBS270 and configuring the registers, we never got any response from the controller as expected and we had to abandon this step because of lack of time. Another student, Hans Persson, continued this in his thesis project.

8 Results and conclusions

The Hexapod now can read and run movement sequences stored in csv-files. We didn't have the time to make it to run automatically on startup, so we need at this point execute the csv-reader from the host via Ethernet to the target. See section 10.3 for further instructions. It was very difficult to estimate the amount of time needed for each part of the project and we didn't manage to accomplish all three steps completely. Still this thesis became a success and The Memec Hexapod robot was eventually shown by Avnet-Memec at the Scandinavian Electronics Event fair 2010 in Älvsjö, Stockholm, Sweden. There it was alerted by visitors, exhibitors and media (see Appendix 4), which meant that Avnet-Memec's, and hence our goal of the thesis project was fulfilled.

9 Suggestions on further development

- **Make the movement sequence autostart.**
  Automatically run movement sequences on startup.

- **PS2 controller**
  Because the time ran out and we did not have time to complete the "Third Step" and implement the PS2 controller, this is an obvious continuation.

- **Video acquisition**
  The MBS270 comes with a preinstalled video server which makes it quite easy to transfer video. Avnet-Memec wish to send video wireless to a personal computer.

10 HOWTO (on a Linux host)

10.1 Install PXA270 development toolchain

You must be root to install the toolchain if you unpack it in any other location than your home directory:

`$ cd /` (if you want it to work for all users) otherwise you can choose to install it in your home directory.

```
 $ cd (if you want it to work only for you)
 $ mkdir  mbs_toolchain
 $ cd ~/mbs_toolchain
```

Then install the toolchain by extracting all files.

```
 $ tar xvjf /path/to/arm-linux-gnueabi.tar.bz2
```

`-x`	=	extract files from an archive
`-v`	=	verbosely list files processed
`-j`	=	filter the archive through bzip2
`-f`	=	use archive file or device ARCHIVE

This creates a full toolchain tree in /usr/local/arm/oe.
(/home/user/mbs_toolchain/usr/local/arm/oe)

Now, you need to include the /usr/local/arm/oe/bin directory into your PATH:

```
 $ PATH=$PATH:/usr/local/arm/oe/bin
 ($
PATH=$PATH:/home/user/mbs_toolchain/usr/local/arm/o
e )
```

Put a statement like that into your shell startup file, e.g. .bashrc or .profile.

To test your toolchain installation, you can simply compile an empty file:

```
 $ touch xyz.c
 $ arm-linux-gnueabi-gcc --version -c xyz.c
```

This should give you something like:

```
arm-linux-gnueabi-gcc (GCC) 4.1.2
```

Cross compiling using gcc

Now your system is ready to compile code for your PXA270 processor. To do so just type the following to make an executable binary:

```
 arm-linux-gnueabi-gcc –o nameofbinary
file_to_compile.c
```

10.2 Connect host to target (SSH) and transfer files (SCP)

Below follows an instruction how to set up SSH and to use SCP:

Install openssh-server as superuser:

```
 /etc/init.d/ssh start
 iptables -A INPUT -p tcp --dport ssh -j ACCEPT
 $ ifconfig eth0 192.168.1.1 up
 $ ssh root@192.168.1.95
 root@192.168.1.95's password:
 root@mbs270:~$
 scp /your/file root@192.168.1.95 :/where/to/put/it
```

Environment

If you like to have a dedicated directory to work in and have easy access to, you can do the following:

`$ mkdir -p ~/work/mbs270` (the -p is for make both directories)

```
 $ export MBS270=~/work/mbs270
```

This only make a temporary work environment, if you want to make it permanent, put it in your .profile file. Then you only have to type:

`$ cd MBS270` to end up in your work directory.

For editing your code you can use your favorite editing tool. If you don't have one yet you can use Eclipse, Qtcreator, Codelite or just a simple text editor.

10.3 Run a movement sequence on the "Memec" Hexapod

These are the steps needed for making the robot run a sequence: (Before Hans Persson in his thesis project made it autostart)

1. Connect the D-sub connector marked S0 on the MBS270 to the SSC-32.

2. Connect the MBS270 to your computer using RJ45 cable network.

3. Power up the MBS270 by connecting it to 3.5 – 4.2 Volt DC.

4. Open a terminal on your computer and move to the directory where you have your binary and csv-file, containing the movement sequence.

5. Assuming you have prepared your computer with SSH-server mentioned in section 9.2 and set your ip to 192.168.1.1 type:
   ```
    $ scp demo_run sequence.csv
   root@192.168.1.95:/var/tmp
   ```
   When prompted for a password just hit enter. This will copy the files demo_run and sequence.csv into the `/var/tmp` directory on the MBS270, which is the main memory and will be gone next time you start the card.

6. Open a new terminal and type
   `$ ssh root@192.168.1.95` When prompted for a password, just hit enter.
   `$ cd /var/tmp` Choose directory
   `$ ls` List files in directory to see that they are there.

7. Run the program by typing `$ ./demo_run` You will now be prompted to choose which file you want to run. Type sequence.csv, hit enter and watch your robot take a dance!

11 Reference list

[1]. PXA270 Software manual, at URL
http://www.mobisensesystems.com/fics/SM_mbs270_v2.pdf

[2]. PXA270 Hardware manual, at URL
http://www.mobisensesystems.com/fics/HM_mbs270_v2.pdf

[3]. Phoenix Excel Program manual, at URL
http://www.lynxmotion.com/images/files/PEPman.pdf

[4]. Sony playstation controller information, at URL
http://www.gamesx.com/controldata/psxcont/psxcont.htm

[5]. The GNU General Public License (GPL), version 2, at URL
http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[6]. Linux manual pages, at URL
http://linux.die.net/man/

[7]. SSC-32 users guide, at URL
http://www.lynxmotion.com/images/data/ssc-32.pdf

[8]. RS-232 for linux and win32, Teunis van Beelen, at URL
http://www.teuniz.net/RS-232/index.html

[9]. PXA270 Developers manual by Intel at URL
http://www.balloonboard.org/hardware/300/ds/PXA270-dev-manual.pdf

[10]. Kåre Halvorsen's phoenix3 movement sequence at URL
http://www.lynxmotion.com/images/files/phoenix3.zip

[11]. Direct Register Access SPI C Code For The PXA270 at URL
http://docwiki.gumstix.org/Sample_code/C/SPI

[12]. Jeroen Jansssen's code in mbasic for PS2 hand controll at URL
http://www.lynxmotion.com/images/files/phoenix1.3.bas

12 Appendix

Appendix 1: Extract from a CSV-file

```
PROJECT;SEQUENCE;STEP;PIN0;PIN1;PIN2;PIN4;PIN5;PIN6;PIN8;PIN9;PIN10;PIN16;PIN17;PIN18;PIN20;PIN21;PIN22;PIN24;PIN25;
PIN26;T0;T1;T2;T4;T5;T6;T8;T9;T10;T16;T17;T18;T20;T21;T22;T24;T25;T26;TMAX;TTOTAL;INFO1
Phoenix5;1;1;-59,99970627;-77,35305023;-54,89247894;0;-77,35662842;-54,86826324;60;-77,35134888;-54,89591217;-60;-
77,34999847;-54,88999939;0;-77,35475922;-54,87849426;60,00029373;-77,35748291;-
54,85892487;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;Start all legs up
;1;2;-59,99970627;-65,48595656;-51,7792497;0;-65,49775465;-51,75908948;60;-65,48302987;-51,78197142;-60;-
65,48357677;-51,77679506;0;-65,49239564;-51,76756648;60,00029373;-65,50175092;-
51,75125447;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;1;3;-59,99970627;-54,11821378;-47,79142125;0;-54,13546841;-47,77502001;60;-54,11452192;-47,79349518;-60;-
54,11641713;-47,78902285;0;-54,12780672;-47,78187455;60,00029373;-54,14157224;-
47,76858387;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;1;4;-59,99970627;-43,51533199;-43,10108772;0;-43,53573469;-43,08798845;60;-43,51124023;-43,10259805;-60;-
43,51399002;-43,098754;0;-43,52675658;-43,09341927;60,00029373;-43,54307326;-
43,08278372;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;1;5;-59,99970627;-33,72762602;-37,83457397;0;-33,74955541;-37,82430571;60;-33,72337887;-37,83560236;-60;-
33,72661444;-37,83229915;0;-33,73995017;-37,82851616;60,00029373;-33,75751012;-
37,82015744;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;1;6;-59,99970627;-24,68523237;-32,07252794;0;-24,70766728;-32,06467449;60;-24,68097515;-32,07314355;-60;-
24,68444819;-32,0702984;0;-24,69786654;-32,06784341;60,00029373;-24,71584459;-
32,06142672;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;1;7;-59,99970627;-16,26605241;-25,85616415;0;-16,28840199;-25,85038587;60;-16,26186206;-25,85642095;-60;-
16,2654157;-25,85396176;0;-16,27865339;-25,85265848;60,00029373;-16,29657095;-
25,84791022;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;2;1;-57,74259903;-7,107043587;-21,00016088;3,957409538;-16,47585602;-25,88917209;61,75420755;-25,82738771;-
30,30140993;-57,74296322;-7,106453478;-20,99795747;3,958708112;-16,46616821;-25,89146248;61,75243983;-25,86209149;-
30,29051077;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;X rotate
;2;2;-54,83364277;1,642693142;-15,77788208;7,858467611;-17,03487746;-26,0043062;63,10956958;-35,77535616;-
34,27625303;-54,83410445;1,643277882;-15,77565494;7,861022155;-17,02536944;-26,00664904;63,10619717;-35,80920637;-
34,26249114;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;2;3;-51,06034518;9,984683372;-10,22271045;11,65071392;-17,95571946;-26,19220803;64,13941178;-46,04084944;-
37,70772187;-51,06094331;9,985301078;-10,22043364;11,65444425;-17,94650294;-26,19463549;64,13473854;-46,07272956;-
37,69062375;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
;2;4;-46,12162941;17,90878567;-4,360157241;15,28880056;-19,22299867;-26,44720444;64,89503626;-56,49908497;-
40,51086115;-46,1224198;17,90947501;-4,357800421;15,29359776;-19,21417428;-26,44974487;64,88926074;-56,52762832;-
40,49000025;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;200;
```

```
/***************************************************************************
 * Phoenix Hexapod Robot csv-reader v1.0 (modified for Phoenix3.csv)      *
 * by                                                                      *
 * Fredrik Persson and Mattias Lindstrom for Avnet-Memec (jan 2010)       *

**************************************************************************/

//------------------------------------------------------------------------
// ---- Include Files ---------------------------------------------------
//------------------------------------------------------------------------
#include <stdio.h>  /* required for file operations */
#include <string.h> /* requierd for string operations */
#include "rs232.h"  /* required for communication   */

//------------------------------------------------------------------------
// ----  Definitions ----------------------------------------------------
//------------------------------------------------------------------------
#define PORT 1 //com port
#define PAUS 3000
#define BAUT 38400
#define BUFSIZE 1000
const char * CSV_FILE = "phoenix3_moded.csv";

//------------------------------------------------------------------------
// ---- Global Variables ------------------------------------------------
//------------------------------------------------------------------------
FILE *fr;              /* declare the file pointer */
//char *cellptr;         /* declare cell pointer */
char buffer [15];
char buffer2 [15];
int startValue;
char line[BUFSIZE];
int data[BUFSIZE];
int startCtr;         //start counter, skip first row
int cellCtr;          //cell counter
char *lnPtr;
int *dtPtr;            //data pointer
char negFlag;
int k;                //used in for-loops
int lnIdx;             //line array index
int dtIdx;            //data array index
char flag;
int sNr;              //servo number, used in communication
char powerOfTen;
int rowCtr;

/* mechanical limits */
    int minAngles[18] = {-26, -101, -106, -53, -101, -106,
                         -58, -101, -106, -74, -95, -77,
                         -53, -95, -77, -74, -95, -77};

    int maxAngles[18] = {74, 95, 77, 53, 95, 77,
                         74, 95, 77, 26, 101, 106,
                         53, 101, 106, 58, 101, 106};

//-----------------------------------------------------------------------------
// ----  Function Prototypes ---------------------------------------------------
//-----------------------------------------------------------------------------
void init();            //Initialize variables
void clearArray();      //Clear/reset arrays
int limiter(int pin, int v);          //check for mechanical limi

//-----------------------------------------------------------------------------
// ----  Main Function ---------------------------------------------------------
//-----------------------------------------------------------------------------

main() {
    init();
    clearArray();

    fr = fopen (CSV_FILE, "r");  // open the file for reading

    while (fgets(line,sizeof line,fr) != NULL) { //going through the file

        rowCtr++;

        // reset pointers and index
        lnPtr = line;
        dtPtr = data;
        dtIdx = 0;

        // start going character by character through the line
        while (*lnPtr != '\0') {


            if (*lnPtr == ';' && startCtr<startValue) { // first wait
startValue number of ';'
                startCtr++;
            }

            if (startCtr>=startValue) { // after startValue number of ';'
                startValue=0; // "inactivate" startCtr

                if (*lnPtr == ';') {
                    dtPtr++;
                    dtIdx++;
                    cellCtr++;
                }

                if (cellCtr > 2 && cellCtr < 22) { // read only specific
cells
                    if (*lnPtr == '-') {negFlag=1;}

                    if (*lnPtr >= '0' && *lnPtr <= '9') { // if a digit
                        if (flag == 1) {
                            flag = 0; // check size once per number
                            if (line[lnIdx+1] == ';' || line[lnIdx+1] ==
',') powerOfTen = 'U';
                            else if (line[lnIdx+2] == ';' || line[lnIdx+2]
== ',') powerOfTen = 'T';
                            else if(line[lnIdx+3] == ';' || line[lnIdx+3]
== ',') powerOfTen = 'H';
                        }

                        switch (powerOfTen) {
                            case 'H': //hundreds
                                *dtPtr = ((*lnPtr-48)*100);
                                powerOfTen = 'T';
                                break;

                            case 'T': //tens
                                *dtPtr = *dtPtr+(*lnPtr-48)*10;
                                powerOfTen = 'U';
                                break;

                            case 'U': //units
                                if (*lnPtr == 0) *dtPtr = 0;
                                else *dtPtr = *dtPtr+*lnPtr-48;

                                if (negFlag==1) *dtPtr=*dtPtr*(-1);

                                negFlag=0;
                                flag=1;

                                // if next number is a decimal
                                if (line[lnIdx+1] == ',') {
                                    powerOfTen = 'D';
                                    flag = 0;
                                }
                                break;

                            case 'D': //decimal
                                if (*lnPtr >= '5') { // round to one
decimal
                                    if (*dtPtr>=0) *dtPtr = *dtPtr+1;
                                    else *dtPtr = *dtPtr-1;
                                }

                                while (*lnPtr != ';') {
                                    lnPtr++;
                                    lnIdx++;
                                }

                                lnPtr--;
                                lnIdx--;
                                flag = 1;
                                break;
                            } // end switch
                    } // end of if digit
```

```
                } // end of between cell nbr.
            } // end of startvalue counter
            lnPtr++;
            lnIdx++;
        } // end of row

        // send data to servos
        if (data[3] != '\0') {
            OpenComport(PORT, BAUT);
            sNr=0; // servo/pin number

            for (k=0;k<9;k++) {
                if(sNr == 3 || sNr == 7) sNr++;
                //sprintf(buffer,"#%dP%d",sNr,data[k+3]); // csv-control
                                sprintf(buffer,"#%dP%d",sNr,1500-
(limiter(k,data[k+3])*100)/9); //servos inverted! (scaled by 100/2=50)
                cprintf(PORT, buffer);
                puts(buffer); // test print to screen
                sNr++;
            }

            sNr = 16;
            for(k=16;k<25;k++) {
                if(sNr == 19 || sNr==23) sNr++;
                //sprintf(buffer,"#%dP%d",sNr,data[k-4]); // csv-control

sprintf(buffer,"#%dP%d",sNr,1500+(limiter(k,data[k-4])*100)/9); // (scaled
by 100/2=50)
                cprintf(PORT, buffer);
                //puts(buffer); // test print
                sNr++;
                puts(buffer); //test print
            }

            sprintf(buffer2,"T%d\r",data[21] ); //T=data[21]
            cprintf(PORT, buffer2);
            //puts(buffer2); // test print to screen
            usleep(data[21]*1000 + PAUS);
            printf("End of line, wait for servos to finish...\nrowcounter:
%d\n",rowCtr);
        }
        clearArray();
        lnIdx=0;
        cellCtr=0;

    } // end open file
    CloseComport(PORT); //prova flytta hit
    fclose(fr); // close the file prior to exiting the routine

} // end main


//---------------------------------------------------------------------------
```

```
// ----  Functions -----------------------------------------------------
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
//init - Initialize variables
//-------------------------------------------------------------------------
void init() {
    powerOfTen = 0;     //H=hundred, T=ten, U=unit, D=decimal
    startValue = 42;    //wait startValue number of ';'
    startCtr = 0;        //start counter, skip first row
    cellCtr = 0;         //cell counter
    negFlag = 0;        //if negative number
    k = 0;               //used in for-loops
    lnIdx = 0;           //line array index
    dtIdx = 0;           //data array index
    flag = 1;           //check size of number
    sNr = 0;             //servo number, used in communication
    rowCtr=0;
}

//-------------------------------------------------------------------------
//clearArray - clear/reset arrays
//-------------------------------------------------------------------------
void clearArray() {
    for(k=0;k<BUFSIZE;k++) {
        line[k]='\0';
        }

    for(k=0;k<BUFSIZE;k++) {
        data[k]='\0';
        }
}

//-------------------------------------------------------------------------
//limiter - check for mechanical limits
//-------------------------------------------------------------------------
int limiter(int pin, int v) {
    int tmp;
    if ( minAngles[pin] < v ) tmp = v;
    else tmp = minAngles[pin];

    if ( maxAngles[pin] > v ) tmp = v;
    else tmp = maxAngles[pin];

    return tmp;
}
```

Ny mässa ska samla branschen

Elektronikbranschen inviger sin viktigaste mässa – Scandinavian Electronics Event – i Älvsjö

Statssekreterare Jöran Hägglund satte stor tilltro till den svenska elektronikbranschen när han igår invigde Scandinavian Electronics Event i Älvsjö. Regeringens förhoppning är att trenden med utflyttad elektronikproduktion och konstruktion ska vända.

– Visst har det funnits ett stort utflöde och folk som flyttat ut för att göra kortsiktiga vinster, säger Jöran Hägglund.

Han tror att lägre skattetryck i Sverige är en viktig del för att få kvalificerade arbeten att stanna.

– Men minst lika viktigt är att vi intresserar ungdomar att satsa på en teknikutbilding.

Mässan domineras annars av underleverantörer till svensk industri som tar chansen att visa upp sig.

– Mässan består till två femtedelar av komponentföretag, säger Magnus Eriksson som är projektledare för mässan.

Han tror att mässan fortfarande har en funktion att fylla som mötesplats, trots att mer information finns tillgänglig på andra sätt och att få nyheter presenteras på mässan.

– De facto vill folk komma och klämma på saker. Men det kanske inte är lika många på varje företag som behöver gå på en mässa. METRO TEKNIK

Mässan

Lena Norder, vd på branschorganisationen Svensk elektronik.

- Scandinavian Electronics Event ska hållas vartannat år i Stockholm. Årets upplaga är den första.
- Mässan pågår den 13–15 april.
- Årets upplaga har 274 utställare.
- Senaste mässan av samma slag i Stockholm lockade 7 400 personer.
- En majoritet av utställarna är komponentbolag. Men även testföretag, inbyggda system och produktion finns representerade.
- Mässan ordnas av branschorganisationen Svensk Elektronik som har cirka 250 medlemsföretag.

WILL LEACH

▶ Johan Enokssons egenbyggda spindelrobot ska locka besökare till Avnet-Memecs monter på elektronikmässan i Älvsjö.

Kommentar

NERVOSITET I LUFTEN

Det ligger lite nervositet i luften på Stockholmsmässan i Älvsjö när första versionen av SEE drar i gång. Behövs mässor verkligen fortfarande? Och har någon tid att gå efter branschens besparingar? Första dagen är det långt ifrån trångt på mässgolvet trots att mässan är snygg och tillgänglig för att vara en traditionell branschmässa.

JONAS RYBERG
NYHETSCHEF, METRO TEKNIK

Svensk fred säljer.

Debattera försvarsindustrin på teknikdebatt.se

Nu startar debatten på allvar! Gå in på teknikdebatt.se och gör din röst hörd, få svar av makthavarna och följ de avgörande frågorna för din och Sveriges framtid. Teknikdebatt.se – Sveriges nya sajt för bransch- och samhällsdebatt.

Teknikdebatt Sverige AB, teknikdebatt.se, info@teknikdebatt.se, Box 5247, 102 45 Stockholm

!TD TEKNIKDEBATT.SE

Appendix 4: The Phoenix Hexapod Robot