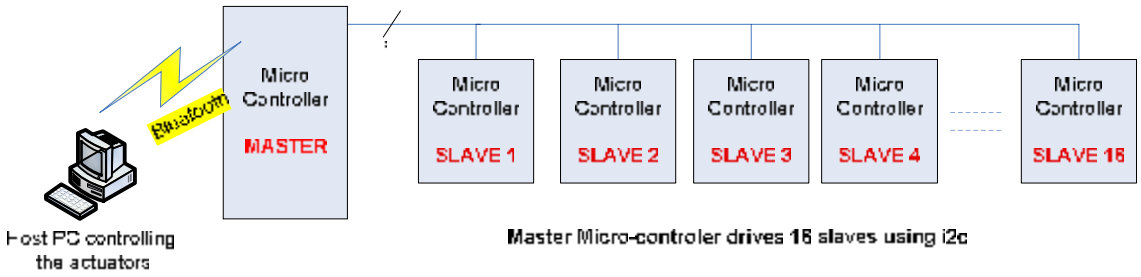
**"Metazoa Ludens",** I'm doing the embedded hardware system for the platform. In the platform there are 16 actuators, each connected by a worm axel & driven by separate stepper motors. In the platform the linear movements should be precise & mapped to the virtual scaling. So the linear movement has a resolution of 100 & to obtain that resolution we use precision stepper motors. But for the top bait system I have suggested to use servo motors since it gives a better torque-speed relation than steppers.

## Selecting a proper communication protocol

The initial idea was to control each piston separately with a dedicated Bluetooth connection with the host PC. So it would require 16 separate controller boards & 16 Bluetooth connections. My first idea is to use FPGA to control the actuators. I gathered some knowledge about VHDL & FPGA programming on my second month of training. Since FPGA supports concurrent processing I only need one FPGA to implement everything. But since this project is selected for ISEA 2006 & needed to be completed before September, I had to use micro controllers because it reduces design time.

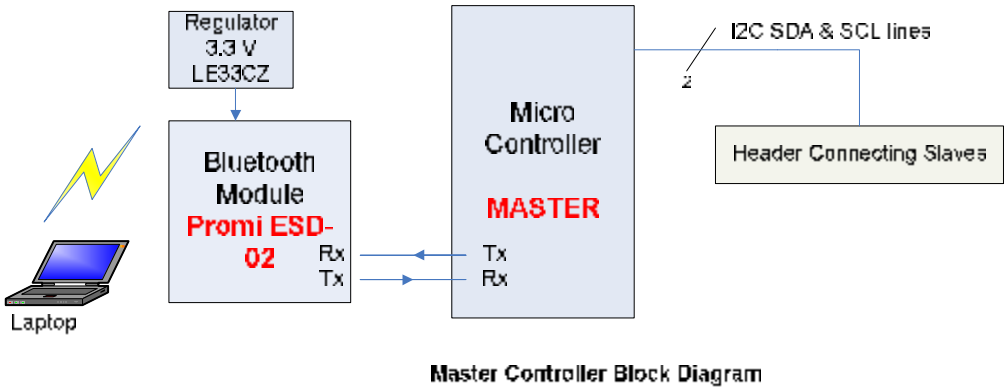
The main barrier was to minimize the hardware when using micro-controllers. First design was to implement a bus line & introduce a new communication protocol to the system. But then there is a speed limitation as we are not precise on protocol timing. After some research work on protocol speeds & comparing with the required speeds I was able to select the **$I^2C$ protocol** to communicate between the master & slave devices, I came up with a design to drive 16 actuators using one master micro-controller & 16 slave micro-controllers. This minimizes to one Bluetooth device.

**$I^2C$ protocol** is a 2 wire bus architecture which is more likely to Ethernet & it was totally new to me. Since no lab members have used $I^2C$ protocol in any other projects I had to do it from the scratch. So I followed the user manuals, specification sheets & documentation given by Philips Semiconductors & within 2 days I was able to catch up everything.




Above is the schematic diagram for communication between micro-controllers. I implemented the new protocol and embedded hardware with 4 slave devices in a proto board. The tests were successful. I created a VB program to test & simulate the pistons & checked with the proto type.

## Designing master controller




Master controller is receiving the data over Bluetooth & filtering data according to the protocol I have introduced. After filtering it will transmit over $I^2C$ SDA & SCL lines. Every data packet transmitted over $I^2C$ has a source & destination address. The destination device which matching has address will accept the transfer. Also destination device will send a reply when it's completed receiving. All these conditions are handled in micro-controller level.

**Protocol structure is**

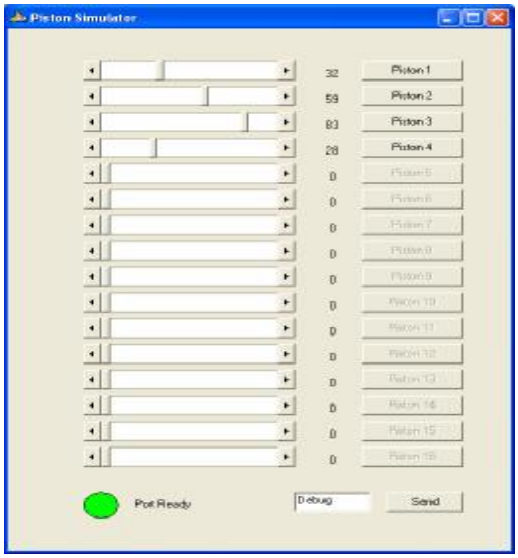
| Start Mark (H01) | Piston ID | Displacement Index |
|---|---|---|

**Protocol Format**

Piston ID is supported only ***even numbers*** & displacement index is ***(0 - 100).*** As the start mark you have to use ***01***. All the protocol data should be sent as hex characters. Each followed by the previous. So all together you have to send 3 bytes of data to control one actuator

**Application software**

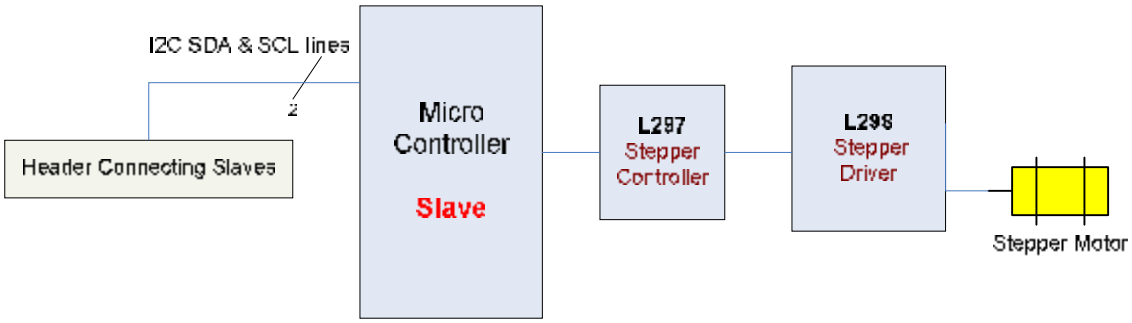
First I did the communication part using **$I^2C$** & integrated with Bluetooth communication with the PC. Then introduce the new protocol & tested with it. I created a small visual basic program to interface the actuators




According to the program you have to give adjust the horizontal slider to a desired height & press the piston button. Actuator will move accordingly to the height & stop there.

## Designing slave driver

I haven't done any stepper driving using a PIC earlier & it was totally new experience for me. I studied couple of codes given by a student & it was quite straight forward. So I gathered necessary details using few online media & data sheets provided by the specific vendors & started work related to stepper driving. First I tried to use a more compact chip that can control & drive the stepper through a single IC. There are some IC's available in the marker but compared to normal usage of 2 IC's its quite expensive. *(Examples: A3980, TMC428-I)*



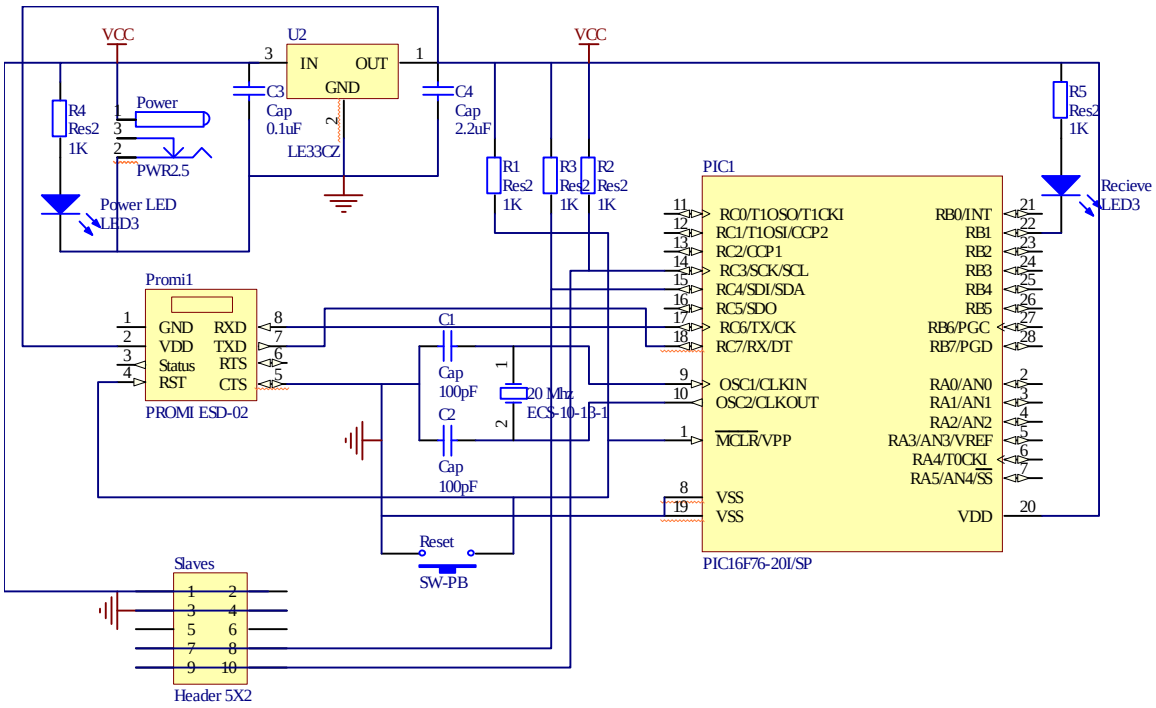

**Slave Controller Block Diagram**

So I continued the normal way of controlling the stepper using a micro-controller attached. I used L297 stepper controller IC & L298 Driver IC to design the circuit. It's quite easy because the typical circuit was given in the data sheet as well. So I had to just integrate the L297 with the micro-controller so that I can control the stepper through the signals given over $I^2C$. After reading the L297 data sheet I was able to gather how to send control signals in order to drive the stepper.

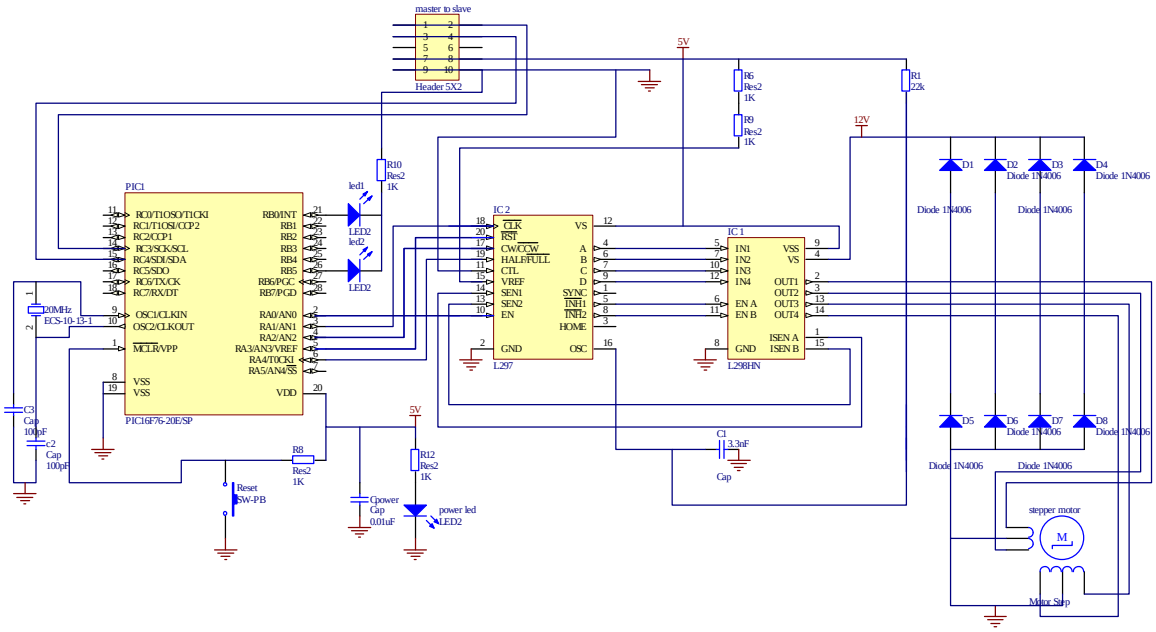
**Few new features that I have introduced in the stepper driver:**

- Actuator is able to self reset on power-on regardless of the current position.
- After stopping on a previously given height, the second command can be executed from that place onwards, don't have to reset it to 0.
- During the actuator moving you can send a second command, so it will intelligently calculate the new path differences & act immediately.

## Schematic design & prototyping




Above is the master controller board schematic that I have designed. I introduced 2 indicators. I used surface mounted LED's in order to reduce the power consumption. One indicator is for power & the other one is for Bluetooth data receiving indicator. When the master controller connected to a PC LED will be solid & when sending a command it will blink.



Above is the slave driver schematic. This controlled by the $I^2C$ bus line. I have introduced few indicators. One is for the power, one for the actuator activity. So when the actuator is moving it will blink. When the actuator is parked at the reset position it will be indicated separately by another LED.

For prototyping, I used one master with four slaves, but LED's instead of steppers. I checked most of the codes with $I^2C$ & the Bluetooth. Well it worked fine & that gave me the courage to continue the work for next stage.

## PCB Designs

Since I have completed the coding for the slave & master parts of the system I started designing the PCB. In university I have used **OrCAD 10.0** several times & it was quite easy for me. So I started the design using OrCAD. I started from the master board, but I couldn't find any related footprints or library components in OrCAD especially for the un-common components such as L297, L298 & Promi ESD-02. There was one option in OrCAD which allows you to search online in the cadence libraries & active parts database *(http://www.activeparts.com/index.asp)* I tried to download the library parts from the active parts database. It shows the results when I search for a component, but wont allow me to download. Later when I search in the help forums, it has stated that only the latest version of Capture CIS (v10.5) is allowed to download.

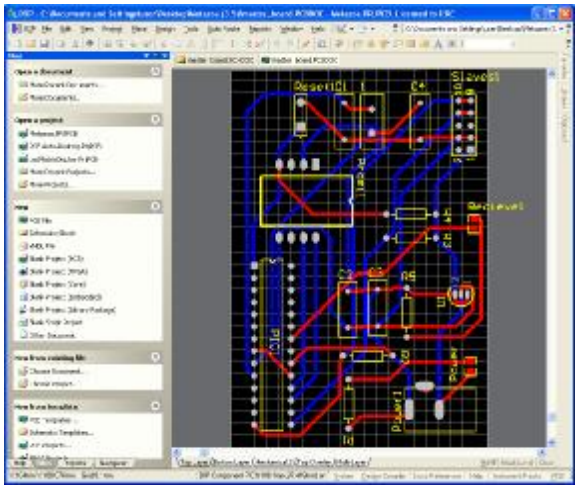
So I asked from lab members about how they normally design PCB's. They have been using Protel 99 but I found that it's not sophisticated to support auto routing features & such advanced library management. So I searched back for a better option & found a software called **Altium Designer 6.0** *(http://www.altium.com)*

**Altium Designer** is the world's first and only unified electronic product development system that allows engineers to take a design from concept to completion within a single application. Altium Designer brings together hardware, software and programmable hardware development within a unified environment that allows all aspects of an electronic product to be designed and managed within a single system. When I tried to

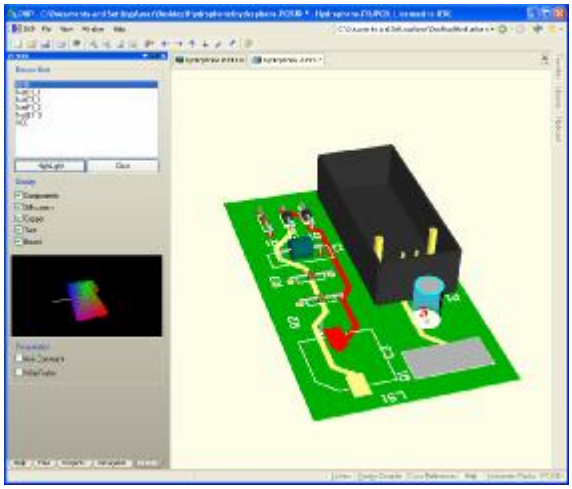
gather more information found that it's an integration of **Circuit Studio, Nvisage, Nexar, Protel, CAMtastic & OrCAD**.

I nearly spent about one week to get the full software features to be familiarize. It's fully supported by multilayer *(16 layers)* auto routing. It's actually meant for board level designers like high grade PCB manufacturers for motherboard; add on card developers to make their designs more efficient. It also supports simulation processes like in OrCAD, FPGA/CPLD device programming & debugging like in Nexar.

So I designed the PCB design for the master controller & later finished the slave boards as well. A very good feature on Altium Designer is, it can finally experience the 3D view of the board. It even gives the 3D view of the components as well. Especially this is very useful when trouble shooting the boards after manufacturing in case of a fault. It can specifically highlight the given NET & we can clearly see it even from both sides.



***(PCB Design by auto routing)***

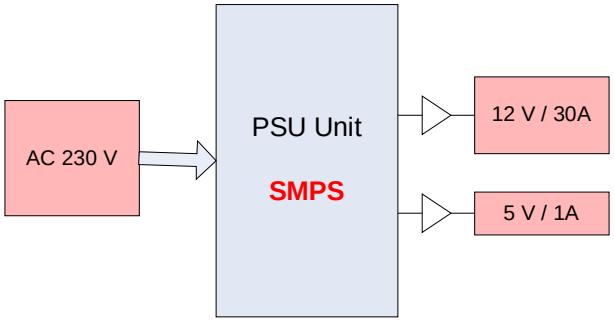


***(PCB 3-D view)***

If you want to add a custom device or component that is not available in the website, you can always design yourself the schematic library & the PCB footprints.

## Power supply design

Each stepper is consuming about 1.5 A in its operating mode. Total of 16 steppers will require about 24 A. Also for other functions it will require a little more power. Designing the power supply in a linear way is not cost effective. (i.e. using Transformers) So I was planning to use a switching mode power supply which can deliver 30 A. Actually SMPS designing is very professional task. Even the PCB have to be designed such away that it minimizes the stray effects. SMPS operation is high frequency & we have to consider the frequency when designing the PCB's. So the best way is we can purchase a switching mode power supply that suits our work. Below is a block diagram for the power supply.

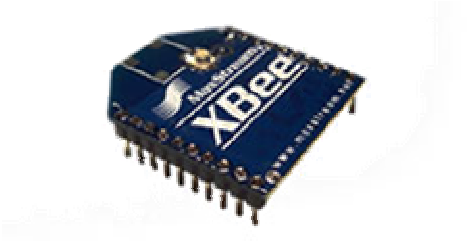



**Switching Mode Power Supply Unit**

China is producing cheap SMPS which suits our work & it's really in good quality.

## Future work & improvements

Since these days wireless communication is getting popular on **ZigBee**, I thought of introducing ZigBee to my project in a later stage. I gathered almost everything related to use ZigBee in a embedded platform & how **AT commands** can be used to communicate with the module. **MaxStream** Company is providing *(http://www.maxstream.net)* the cheapest embedded modules in the brand name of ZBee.




***(XBee™ ZigBee OEM RF Module)***

***(XTend™ OEM RF Module)***

The advantage over traditional Bluetooth is ZigBee consumes low power to transmit even higher ranges. It has been developed up to transmit over a distance of maximum 40 KM. *(XTend™ OEM RF Module)*

Servo motor driving is the next stage of the research work. Since stepper motors are not suitable for high speed applications, we need to use servo motors for the top bait system. So I gathered most of the information related to servo motor driving as well. Servo driving is more likely to drive a brush type DC motor. But it has a feedback signal so that we can control the rotation.

Currently, bottom actuators have to be setup before the game starts. That means during the game play we can't change the actuator positions. When the full system is in running stage we can introduce a new feature such that the actuators are able to move during game play. For this each actuator should output its current position continuously. For my research work on $I^2C$ it is able to give that required speed of executing the command so rapidly. However we have to test it on the system.

**References:**


[1] THE I2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000
www.semiconductors.**philips**.com/acrobat/literature/9398/39340011.pdf

[2] **I2C** manual (part)
www.semiconductors.**philips**.com/acrobat/applicationnotes/AN10216_1.pdf

[3] I²C web reference - www.**i2c**.**philips**.com/

[4] Bluetooth **Promi-ESD-02**
www.initium.co.kr/bizdata/Promi-ESD-**02**_manual1.1_Class%2**02**_.pdf

[5] XBee™ ZigBee OEM RF Module
http://www.maxstream.net/products/xbee/xbee-oem-rf-module-zigbee.php

[6] Switching-Mode Power Supply Design
http://www.smpstech.com/
http://www.smps.us/

[7] PCB Design using Altium Designer 6.0 (Help guides) - http://www.altium.com

[8] Self study about virtual serial port access in visual Basic 6.0

[9] Referred datasheets

- L297, L298
- LE33CZ
- Promi ESD 02, XBee OEM RF Module
- Microchip PIC 16F76
- Spartan 3 family datasheet

[10] VHDL learning
Digital System Design Using VHDL - Charles H. Roth, Jr
EE4218 Embedded Hardware System Design (NUS Lec Notes)

[11] FPGA learning
Spartan 3 family FPGA by Xilinx
Memec 3SLC Users Guide
www.mr.inf.tu-dresden.de/SpartanMC/Memec/Memec_3SLC_UsersGuide_v2_0.pdf