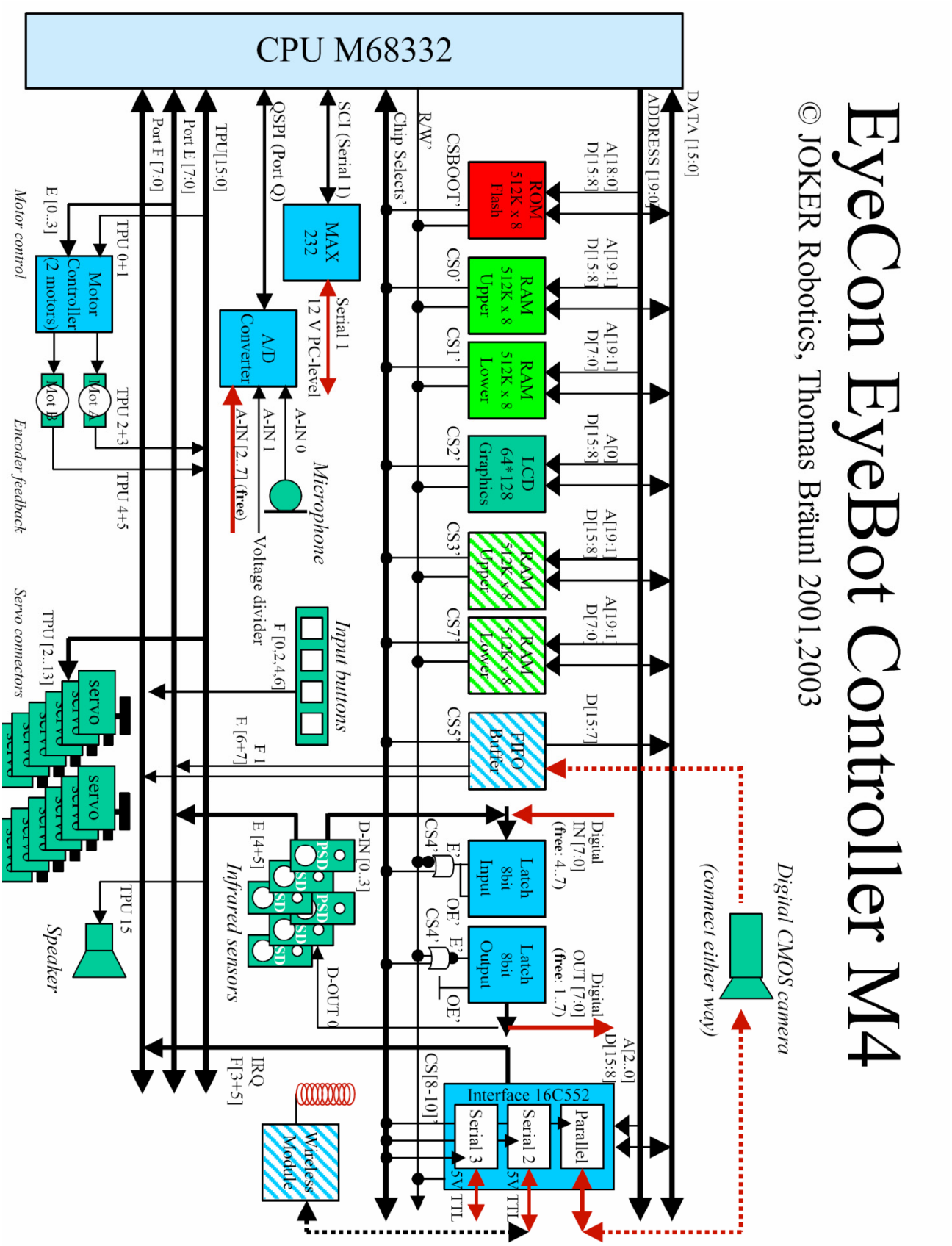
DD2426 Robotics and Autonomous Systems

Project notes A
April 1 2008

- Outline
  - The Eyebot controller
    - Operating system
    - Compiler
  - Eyebot connections
    - Camera
    - motors
    - servos
    - bumper switches, etc
  - Building the robot
    - Tools and materials
  - Soccer rules in brief
  - The lab and workshop
  - Lab0

The Eyebot controller board

- 35 MHz 32-bit Motorola 68332 microcontroller
- 2 MB RAM and 512 KB flash ROM
- Digital camera interface
- Large LCD with input buttons
- Digital and analog I/O pins
- Speaker and microphone


EyeCon EyeBot Controller M4
© JOKER Robotics, Thomas Bräunl 2001,2003
CPU M68332
DATA [15:0]
ADDRESS [19:0]
ROM 512K x 8 Flash
RAM 512K x 8 Upper
RAM 512K x 8 Lower
LCD 64*128 Graphics
FIFO Buffer
Latch 8bit Input
Latch 8bit Output
Interface 16C552
Serial 3
Serial 2
Parallel
Wireless Module
Digital CMOS camera
(connect either way)
SCI (Serial 1)
MAX 232
Serial 1
12 V PC-level
QSPI (Port Q)
A/D Converter
A-IN 0
A-IN 1
A-IN [2..7] (free)
Microphone
Voltage divider
Input buttons
Chip Selects'
R/W'
CSBOOT'
CS0'
CS1'
CS2'
CS3'
CS7'
CS5'
CS4'
CS[8-10]'
Port F [7:0]
Port E [7:0]
TPU[15:0]
Motor control
E [0..3]
Motor Controller (2 motors)
TPU 0+1
TPU 2+3
TPU 4+5
Encoder feedback
Servo connectors
TPU [2..13]
servo
TPU 15
Speaker
Digital IN [7:0] (free: 4..7)
Digital OUT [7:0] (free: 1..7)
D-IN [0..3]
D-OUT 0
Infrared sensors
PSD
SD
IRQ
F[3+5]
5V TTL

The C programming language

Similar to java (or rather java is similar to C) in syntax, but:

- Is not object oriented
- Has no runtime mechanisms like e.g. garbage collection or exception handling
- Does not protect memory
  - Pointers directly to memory
  - Possible to read/write outside bounds of a vector
  - Possible to do any type cast

Concisely covered in for example the book by Kernighan and Ritchie

Study example code and ask if in doubt.

It is also possible to use C++ for programming the Eyebot.

Operating system: “RoBIOS”

- Input/output system calls for keys and LCD - Standard Unix (‘`stdio.h`’) like e.g. `printf()` **BUT**: simplified Eyebot versions are usually preferred

- Camera interface and image processing calls

- Timer, serial port and audio I/O calls

- Cooperative or preemptive multithreading support with semaphore synchronisation.

- Abstraction of connected hardware like motors, encoders, servos, range sensors
  - Uses a ‘hardware description file’, HDT

- Further abstraction (‘VW interface’) of differential drive locomotion accepts speed and turning rate commands. Can also go to a specified position and follow lines and arcs.

- Low level control of I/O ports

Robios system calls

For a list of all the calls built into Robios, go to
`http://robotics.ee.uwa.edu.au/eyebot/`
→ Controller → Libraries

Image Processing
Key Input
LCD Output
Camera
System Functions
Multitasking
Semaphores
Timer
Download and RS-232
Audio
PSD Sensors
Servos and Motors
V-Omega Driving Interface
Bumper / Infrered Sensors
Latches (digital I/O)
Parallel Port
A/D Converter
Radio Communication
Compass
TV Remote Control

(The mystical 'semantics' types are defined in
`/usr/local/eyebot/RoBIOS/hdt_sem.h`)

Compiler: gcc

- The standard gnu C/C++ compiler for Motorola 68K processors. Also handles assembler code.

- Invoke with the gcc68 script as in (one line)

```
gcc68 –o robot.hex robot.c -I
/usr/local/eyebot/RoBIOS/inlucde
```

- Produces a `.hex` file that can be downloaded to the Eyebot (use the COM port) by doing

```
dl robot.hex
```

- Takes a myriad of switches, e.g.
  - `-o` specifies the output file name
  - `-O2` do optimisation
  - `-O3` even more optimisation
  - `-c` do not link (just make object files)

- `make` is a convenient tool for invoking the compiler and avoiding to recompile unchanged source files.
  - Requires a custom `Makefile`
  - See example supplied in *eyebot/eyedemo*
  - just type `make` to compile and link

Versioning system: CVS

CVS lets you ‘commit’ every change you make to your source into its ‘repository’.

Later you can ‘check out’ the source, in its current state or in the state of any previous date.

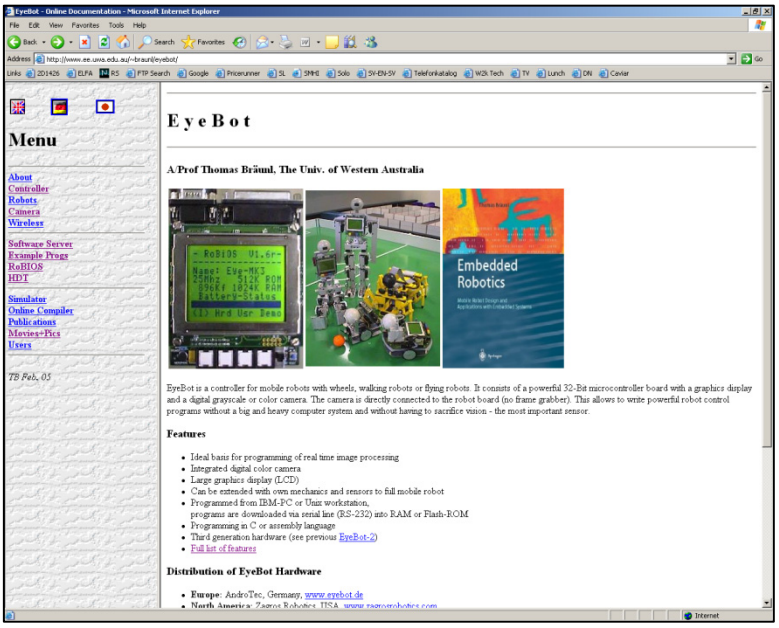
The CVS manual is on the web (`http://ximbiot.com/cvs/manual`).

To use CVS begin by importing your sources as described at the beginning of section 3.

Other parts of the manual useful for getting started:

- 1.3 Sample session with `check out`, `commit`, `release` and `diff`
- 7 Adding and removing source files
- 4 (most importantly 4.4) Tagging source files e.g. for finding all files belonging to a specific version (’release’) of the whole project.
- **Subversion** is an alternative versioning system.

Further Eyebot information

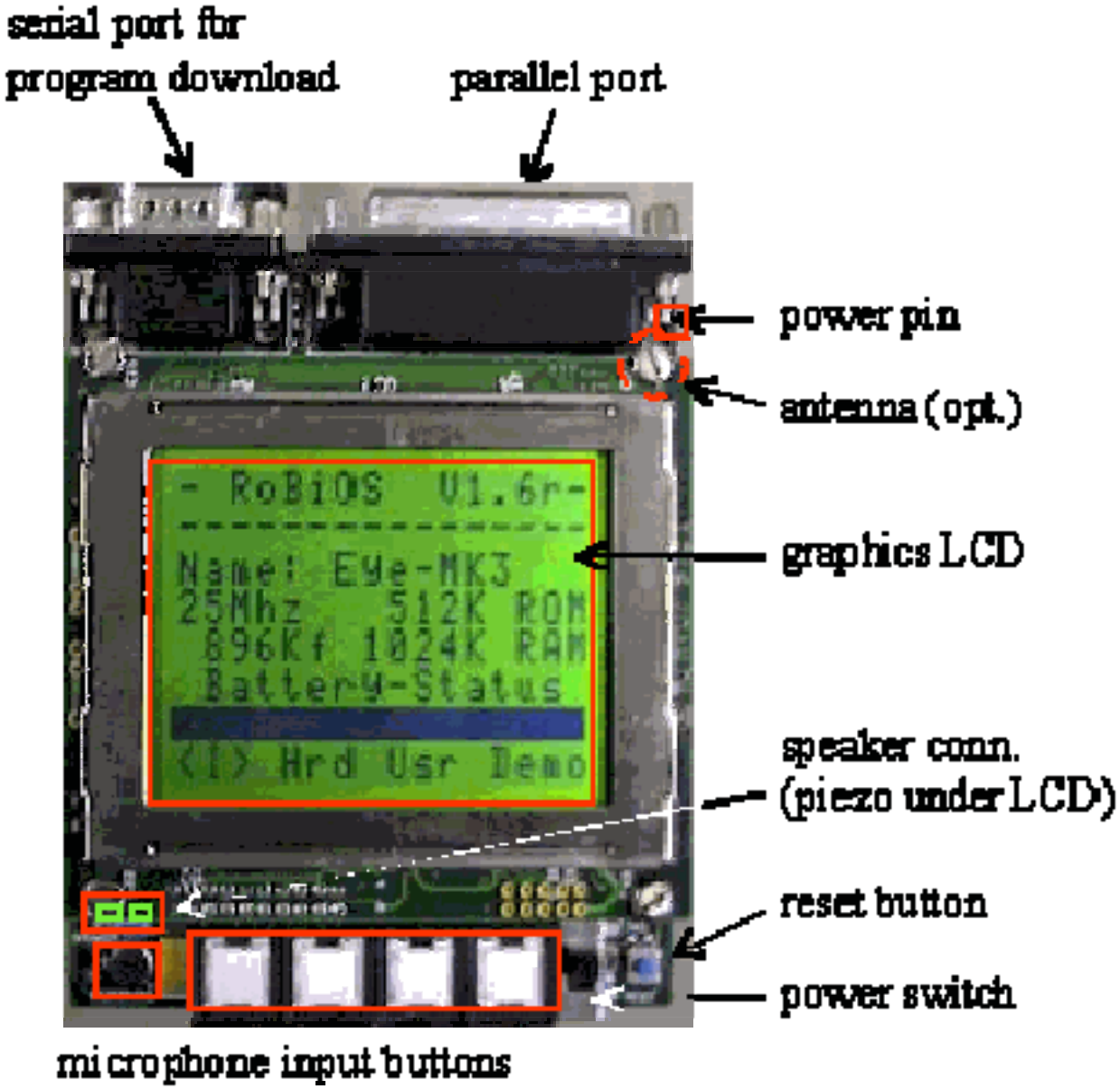


- Go to `robotics.ee.uwa.edu.au/eyebot/` for online documentation.

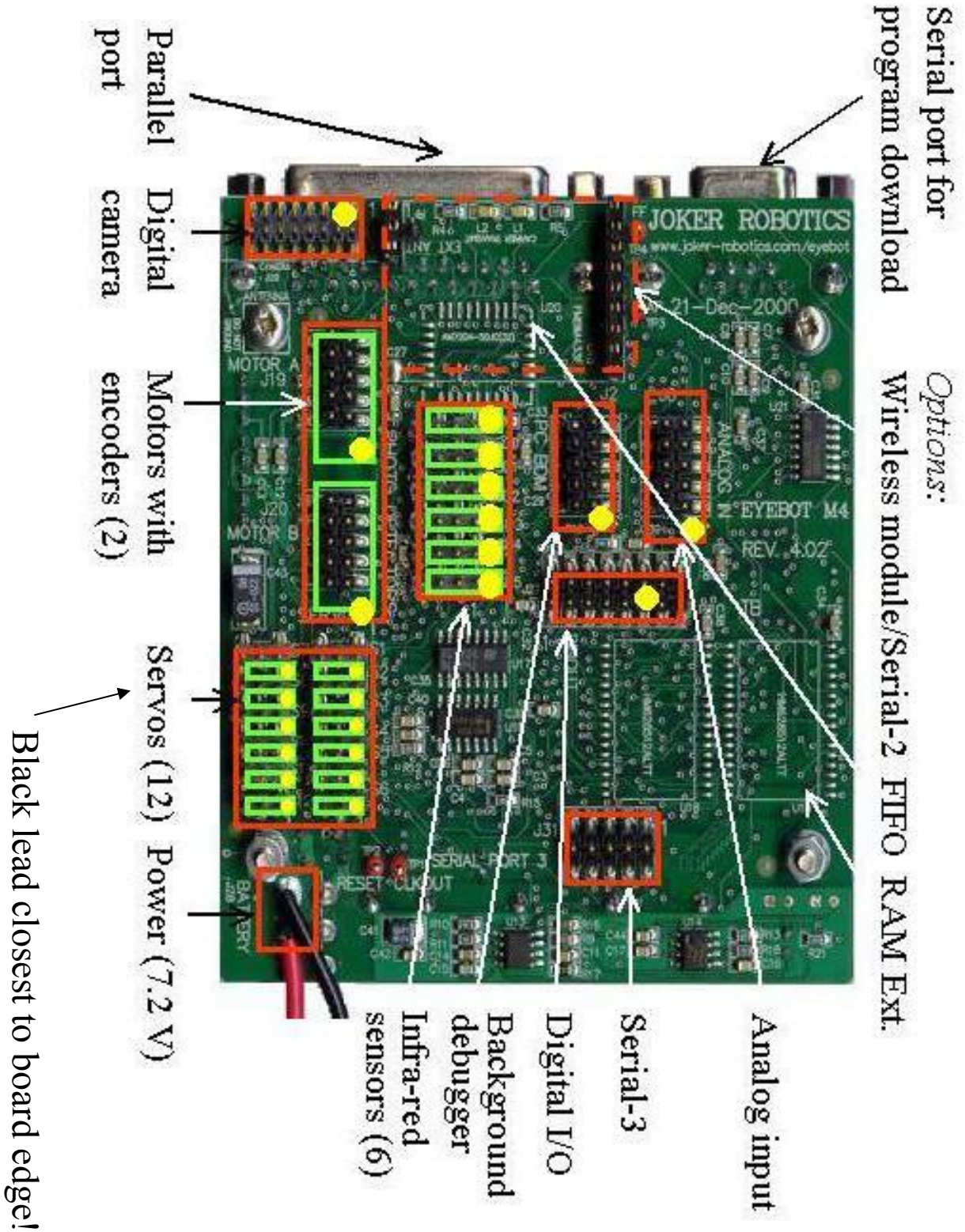
- Software installed on the lab computers can be downloaded for Linux/Windows from the eyebot web page under ‘Software Server'

Contains documentation of all the OS calls, electrical schematics, example source code, etc.

Eyebot connections (front)



Eyebot connections (back)



Don’t hurt your Eyebot!

The Eyebot board is sensitive to electrical abuse and we have only one extra board.

Short circuits, over voltage, and polarity reversal must be avoided.

- Always turn off the controller before making any connections (even to the battery)!
- Check that you used the proper connectors without reversing or misaligning them before turning power back on.
- Metallic objects must not touch the board except at three of the corner screws (the fourth, labelled ‘antenna’ **must not be grounded**).
- Always ask if you are unsure about a connection you want to make!

Avoid touching the components and headers of the board (handle it by its edges).

Don’t hurt your Eyebot! (part 2)

The digital I/O and analog input headers contain a mix of 5V, GND, input and output pins!

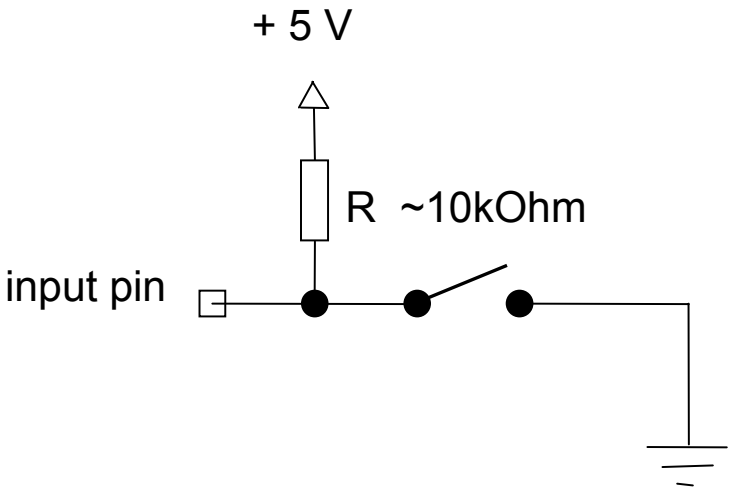
Be sure to check `Controller → Hardware → Connector pinout` in the online documentation, and remember:

- Don’t under any circumstances connect 5V to ground directly or through a small resistance.
- Don’t connect an output pin to 5V or ground.
- Don’t draw too much current from an output pin in any other way.
- Don’t connect a voltage outside the 0..5V range to any pin

The servo connectors normally contain output, GND and **battery voltage** (7.2V) pins. The outputs can be converted to inputs by changing servo entries to bumper entries in the HDT.

Connecting a switch

Connecting extra switches can be useful. For example they can be used as bumpers, indicating contact with obstacles.



Don’t leave the input pin unconnected at any switch position.

Use the proper connectors (‘crimp terminals’) and plastic housings or heat shrink tubing. Ask for a demonstration of the connectors and the special ‘crimp tool’.

Tips for fabricating good bumper switches and ‘whiskers’ follow at the next project lecture.

The lab

The lab is in room 1535 at the fifth floor Osquars backe 2.

There is one Linux computer per group in the lab. They are not automatically backed up! Use ssh/scp to manually back up your work.

The computers are called ras0 – ras8, starting with ras0 closest to the door.

The football field is in the lab as well.

The workshop

A small workshop is located in room 1621. Don’t take the tools (other than perhaps a few screwdrivers and pliers) from the workshop to the lab.

Keep the lab and the workshop tidy! Clean up the benches and the floor before leaving the workshop.

You will have access to the lab and workshop 24 hours a day seven days a week.

Building the robot

You build your robot mainly from aluminium sheets and beams available in the workshop.

Use blind rivets and/or screws to put the construction together.

Before drilling a hole, it is useful to make a small dent at the intended location.

Fix your work piece properly, especially when using power tools!

To mount the motors, you can use a short piece of 1 inch 'L' beam. *Use only the screws supplied with the motors to fix the motors to the beam*. Drill larger holes to make their locations less critical if necessary.

Ask if you need a component that you don't find in the workshop.

BE CARFUL NOT TO HURT YOURSELF OR SOMEBODY ELSE!!!

Handed out components

- An Eyebot controller board and camera.
- Batteries and two chargers.
- Two high quality motors with encoders and gearboxes.
- An RS-232 serial cable.
- The password for one of the lab computers called `ras1-ras8` (the user name is `ras`)
- 3 sheets of aluminiumCC

There are also other components, like e.g. micro switches and R/C servos available in the workshop or from the lab assistant.

All components are on the desks by the computers.

Don't take components from somebody else's desk, not even if replacing them with those from your own

Motors and wheels

The axles of the motors will sustain the weight of the robot, but putting more force than that on them might cause them to bend or become misaligned. *Very few replacements are available.*

Mounting the wheels

Gently push the wheel onto the motor axle.

Then use a hex key (through the small hole under the tyre) to tighten the stop screw. **Don't use more force than necessary!** The correct key has a yellow mark on it.

If this does not reliably fix the wheel to the axle, ask for help.

Some of the motors already have wheels mounted.

The HDT

The HDT file is used to define what hardware is present on your robot.

You have to modify it to tell RoBIOS the distance between the wheels, and the distance traveled per encoder count.

The startup screen and melody can also be changed

Bumpers can be added instead of servos for example.

Have a look in the directories eyebot/hdtorig and eyebot/our_hdt in the home directory of the lab computers.

Demo code

Demo C code adapted for this course and the hardware we use is located in your home directory in *eyebot/eyedemo*

The demo is a single threaded program that does simple colour tracking using the camera attached to a servo. You can

- Choose the target colour
- Do simple white point compensation
- Compute a lookup table to speed up colour computations a lot
- Modify some of the parameters of the image processing
- Send colour images to the PC

This simple colour processing will not be enough for robustly detecting the ball and goals.

Brief soccer rules (more next lecture)

The field is approx. 120 by 240 cm and covered with green felt with white markings.

The ball is an orange-red golf ball.

A white wall surrounds the field except at the goals.

The goals are ~40 cm wide niches in the wall, and coloured yellow and blue (one goal of each colour).

The robots must fit in a 180 mm circle.

The robots cannot grab the ball.

There are no breaks in a game. When a goal is scored the ball is moved to the centre point.

Each robot will carry a big colour marker visible from all directions.

Tackling the opponent is allowed, but dangerous play is not.

Lab0 (getting you started)

Simple lab assignment to get you started.

Present it on

Tuesday April 8, 15-17

or earlier to get a bonus of 2 points added to your exam result.

The task is to build a simple robot that can go approximately straight or turn on the spot depending on some external stimulus of your choice.

You may use the camera, the microphone, a bumper switch, or even the keypad by the LCD to record the trigger stimuli.