Software-Hardware mapping in a Robot Design

Pavol Jusko, David Obdrzalek, Tomas Petrusek
Charles University, Prague

Introduction

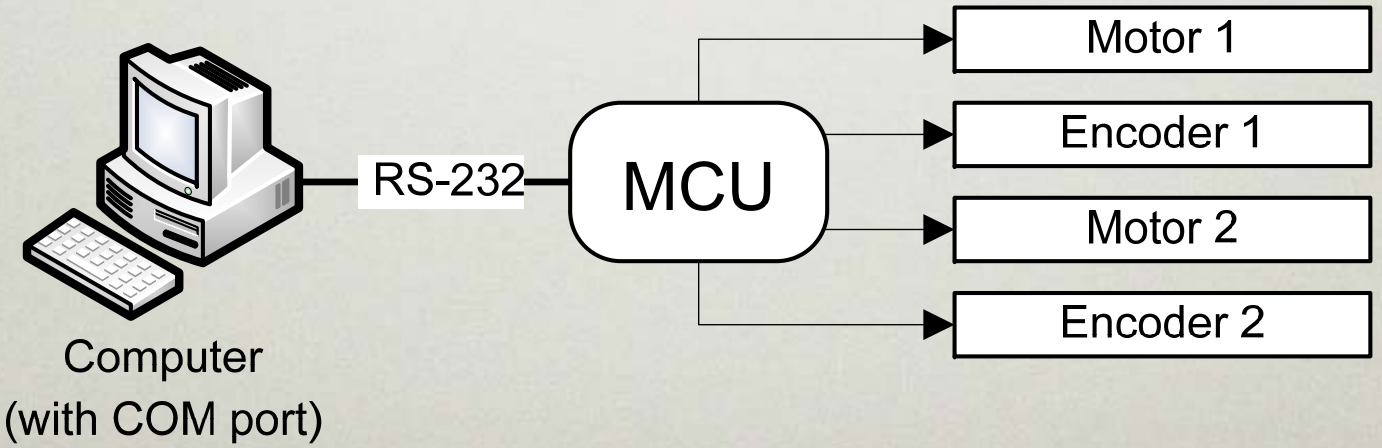
- MART Robotic Team
- How we improved the design and the implementation of our robot
  - Built from scratch, then improved
  - PC based
- Software - Hardware mapping

Agenda

- Original design
  - Hardware
  - Software
  - Pros and Cons
- New design
  - Hardware changes
  - Hardware modules
  - Software changes
- Conclusion

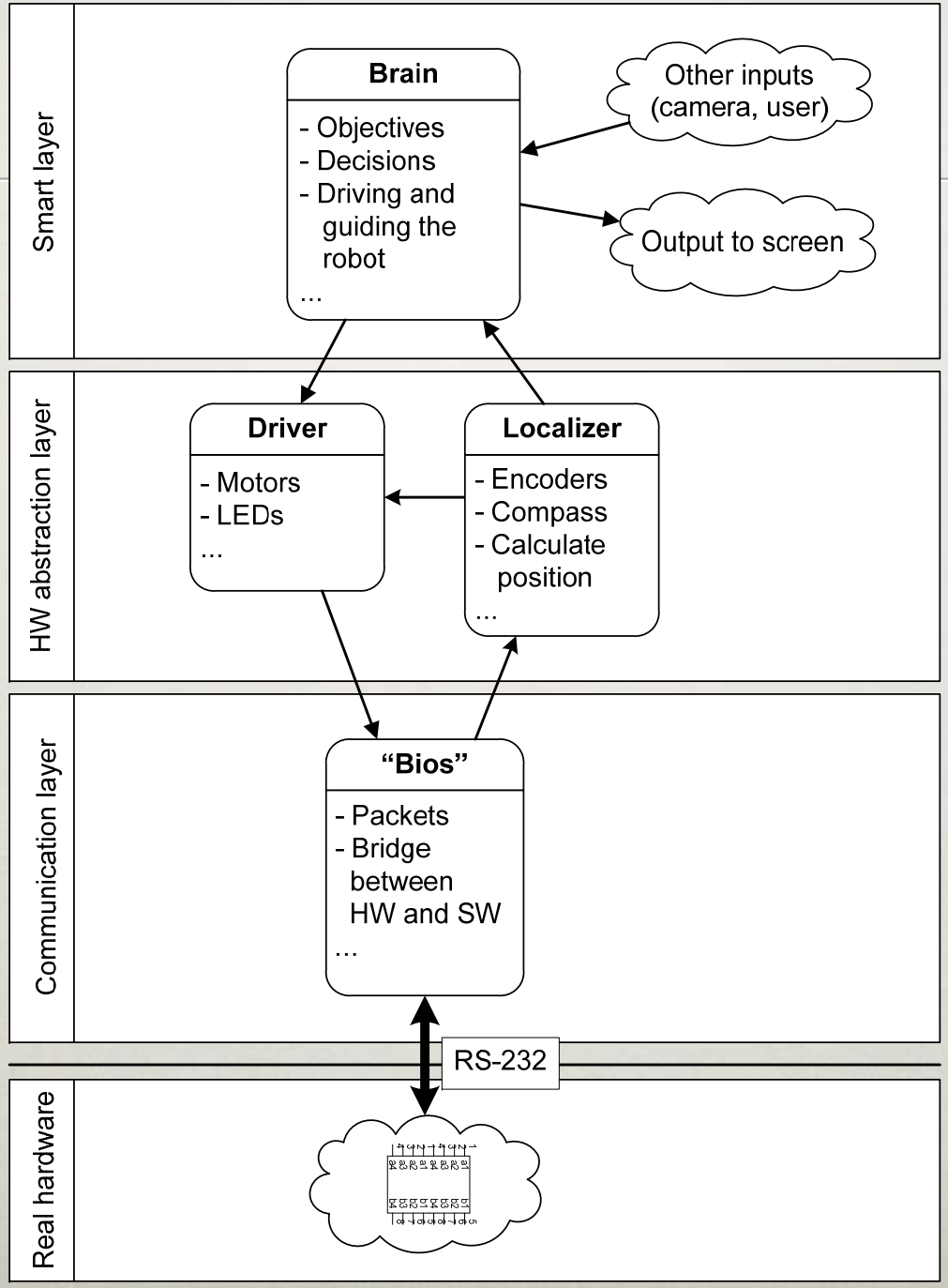
First idea

- One MCU controlling all peripherals
- RS-232 – Serial Port
- Packet oriented protocol



Software design

- What we wanted
  - universal software
  - for universal robot
  - with any kind of hardware
- Layered design
  - Communication layer
  - Hardware abstraction layer
  - Smart layer

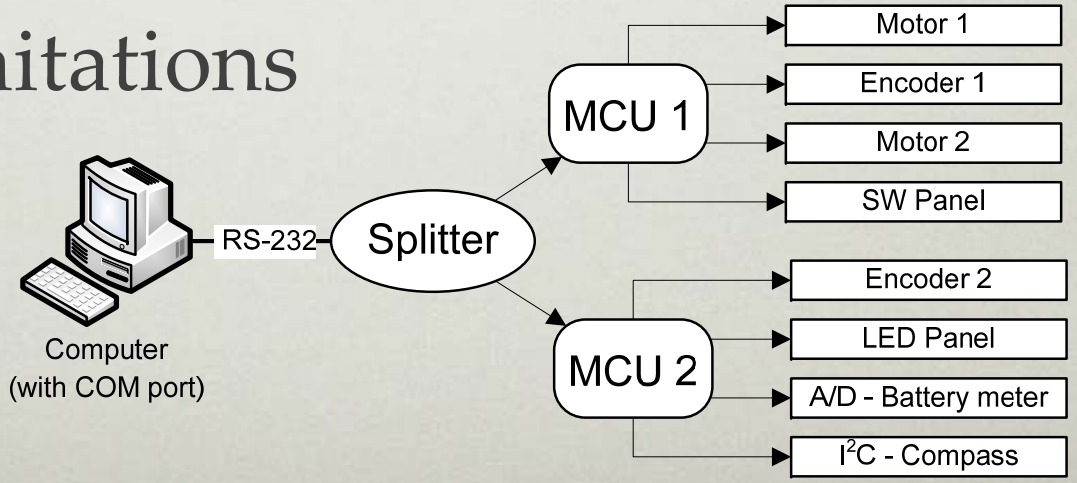

Smart layer
Brain
- Objectives
- Decisions
- Driving and guiding the robot
...
Other inputs (camera, user)
Output to screen
HW abstraction layer
Driver
- Motors
- LEDs
...
Localizer
- Encoders
- Compass
- Calculate position
...
Communication layer
"Bios"
- Packets
- Bridge between HW and SW
...
RS-232
Real hardware

Pros and Cons

- Pros
  - Easy to create
  - Worked well on two contests
    - Eurobot 2007
    - Robotour 2007

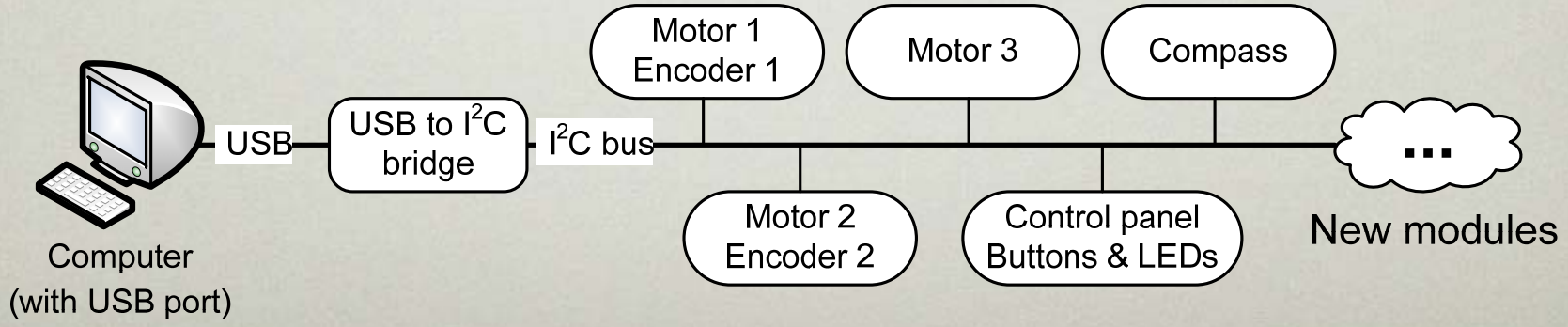
Pros and Cons

- Cons
  - "Hardcoded" solution
  - Packet handling is not maintainable
  - Hard to extend
  - MCU limitations



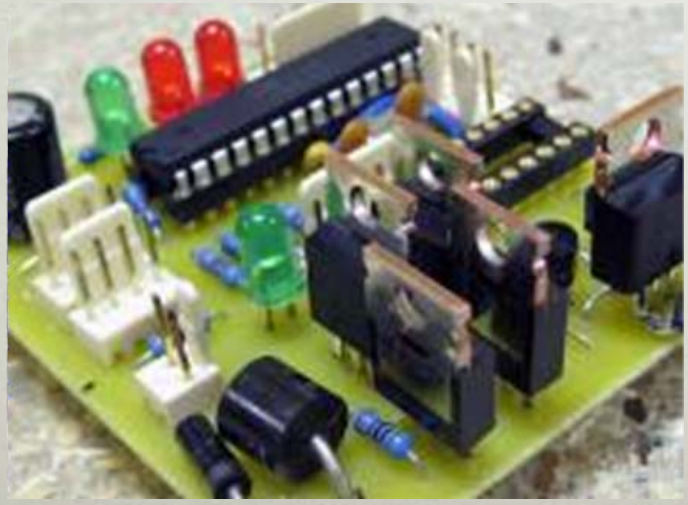
Hardware changes

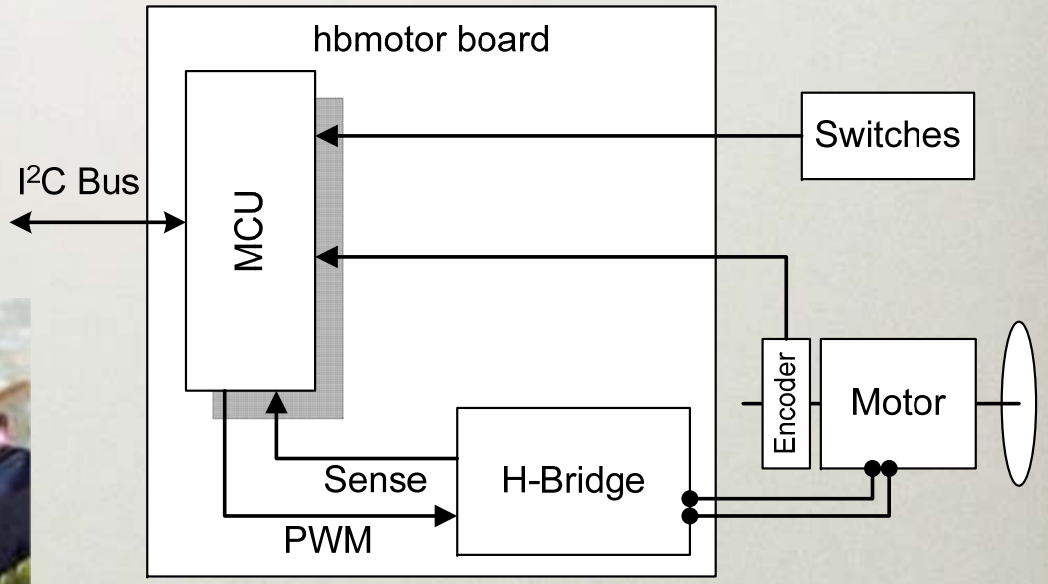
- Removed one-to-one concept
- Bus topology
  - I²C high speed bus with SMBus
- USB to I2C bridge
- Independent modules on the bus



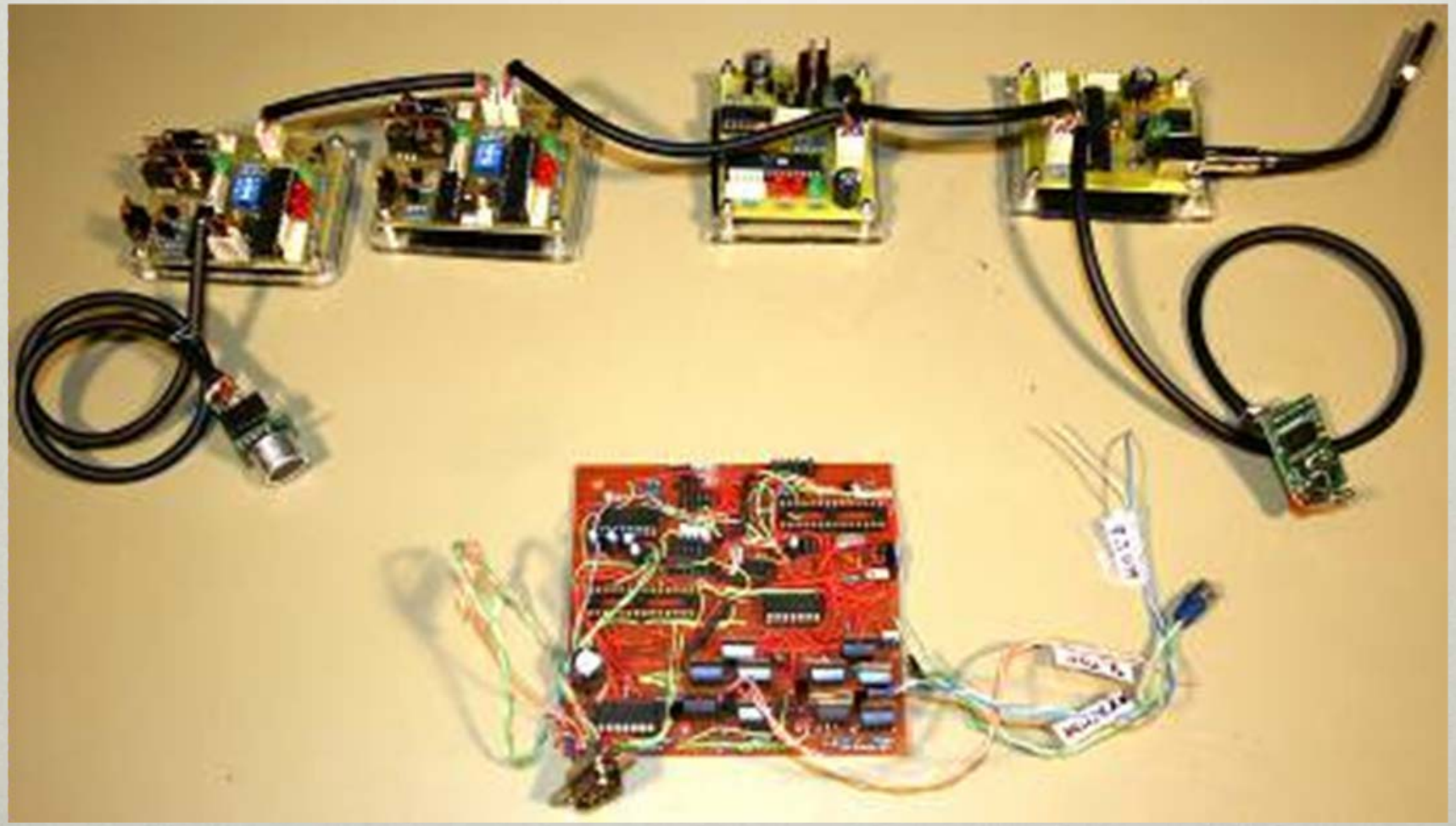
HBmotor board

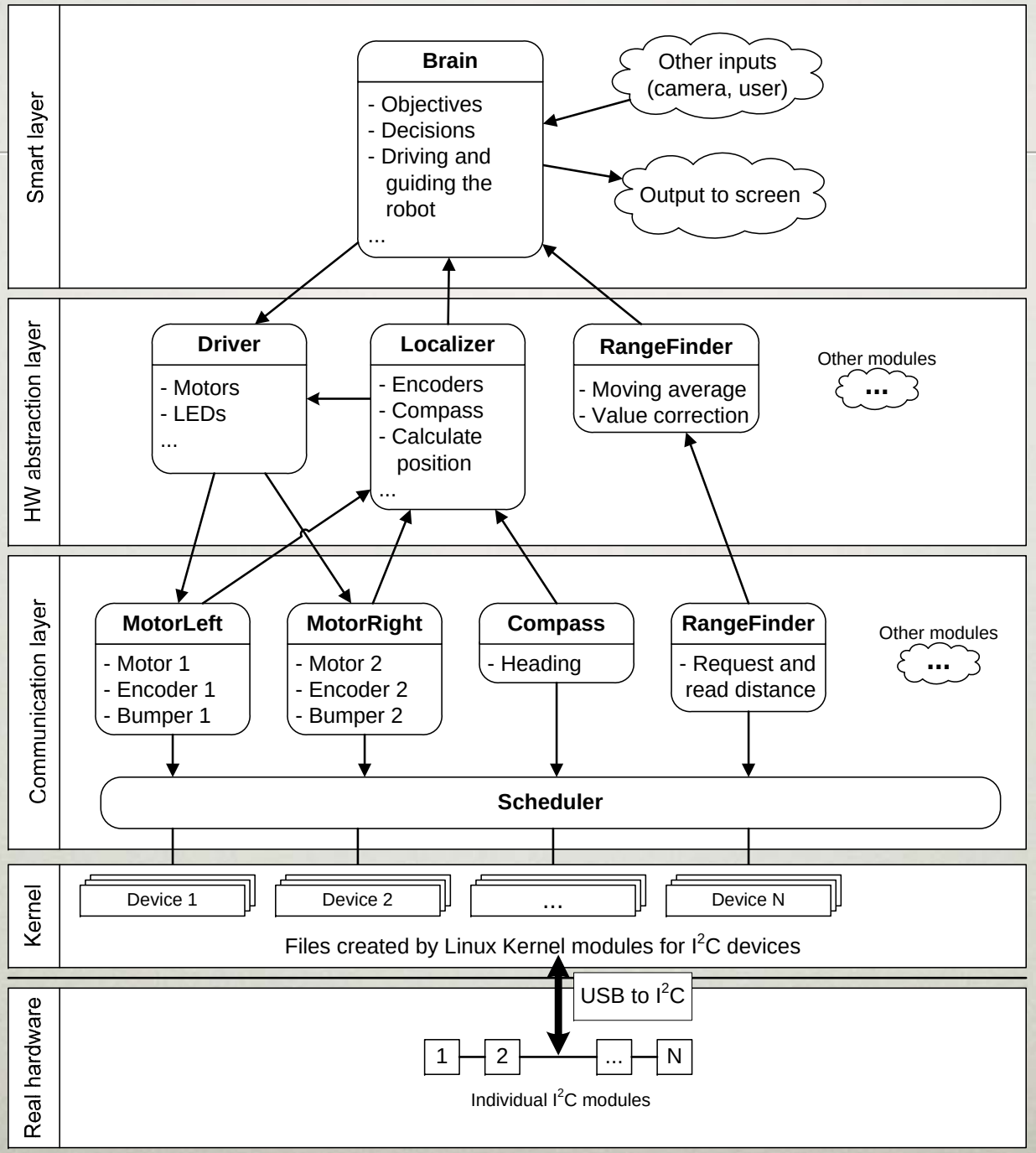
- MCU (Atmel AVR)
- H-Bridge
- Encoders
- Switches





Hardware comparison




Smart layer
Brain
- Objectives
- Decisions
- Driving and guiding the robot
...
Other inputs (camera, user)
Output to screen
HW abstraction layer
Driver
- Motors
- LEDs
...
Localizer
- Encoders
- Compass
- Calculate position
...
RangeFinder
- Moving average
- Value correction
Other modules
...
Communication layer
MotorLeft
- Motor 1
- Encoder 1
- Bumper 1
MotorRight
- Motor 2
- Encoder 2
- Bumper 2
Compass
- Heading
RangeFinder
- Request and read distance
Other modules
...
Scheduler
Kernel
Device 1
Device 2
...
Device N
Files created by Linux Kernel modules for I2C devices
Real hardware
USB to I2C
1
2
...
N
Individual I2C modules

Conclusion

- More universal design
- Better maintainability
- Better extendibility
- Easier changes
- More possibilities for the future

Thank you for your time
Questions?

Pavol Jusko, David Obdrzalek, Tomas Petrusek
Charles University, Prague

http://mart.matfyz.cz, petrusek@gmail.com