Arbeitskreis Hardware

Prof. Dr. Michael Rohs, Dipl.-Inform. Sven Kratz

michael.rohs@ifi.lmu.de

MHCI Lab, LMU München

Goals

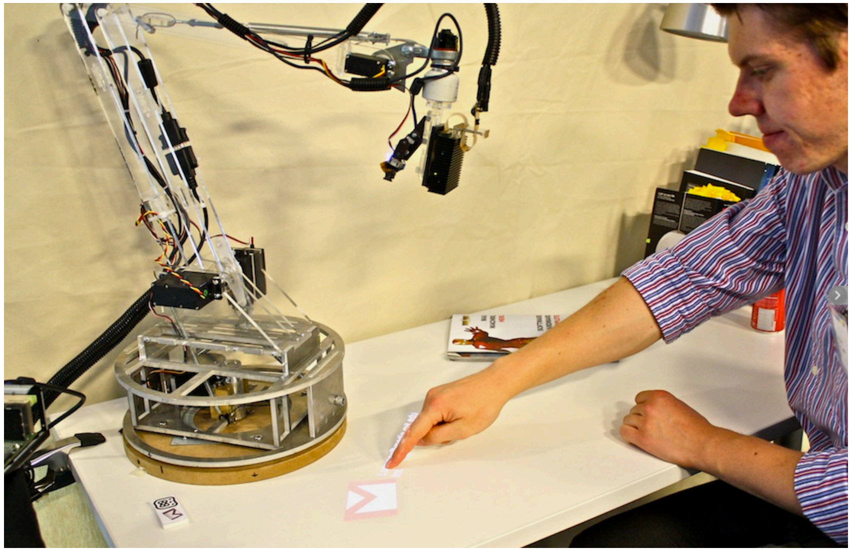
- Learn how to…
- Build embedded interactive systems
- Build interactive devices and objects
- Build actuated installations
  - Example: actuated projector
- Build mobile robots

Robotized Objects

http://www.youtube.com/watch?v=sYutehhGknI

Actuated Pico Projector

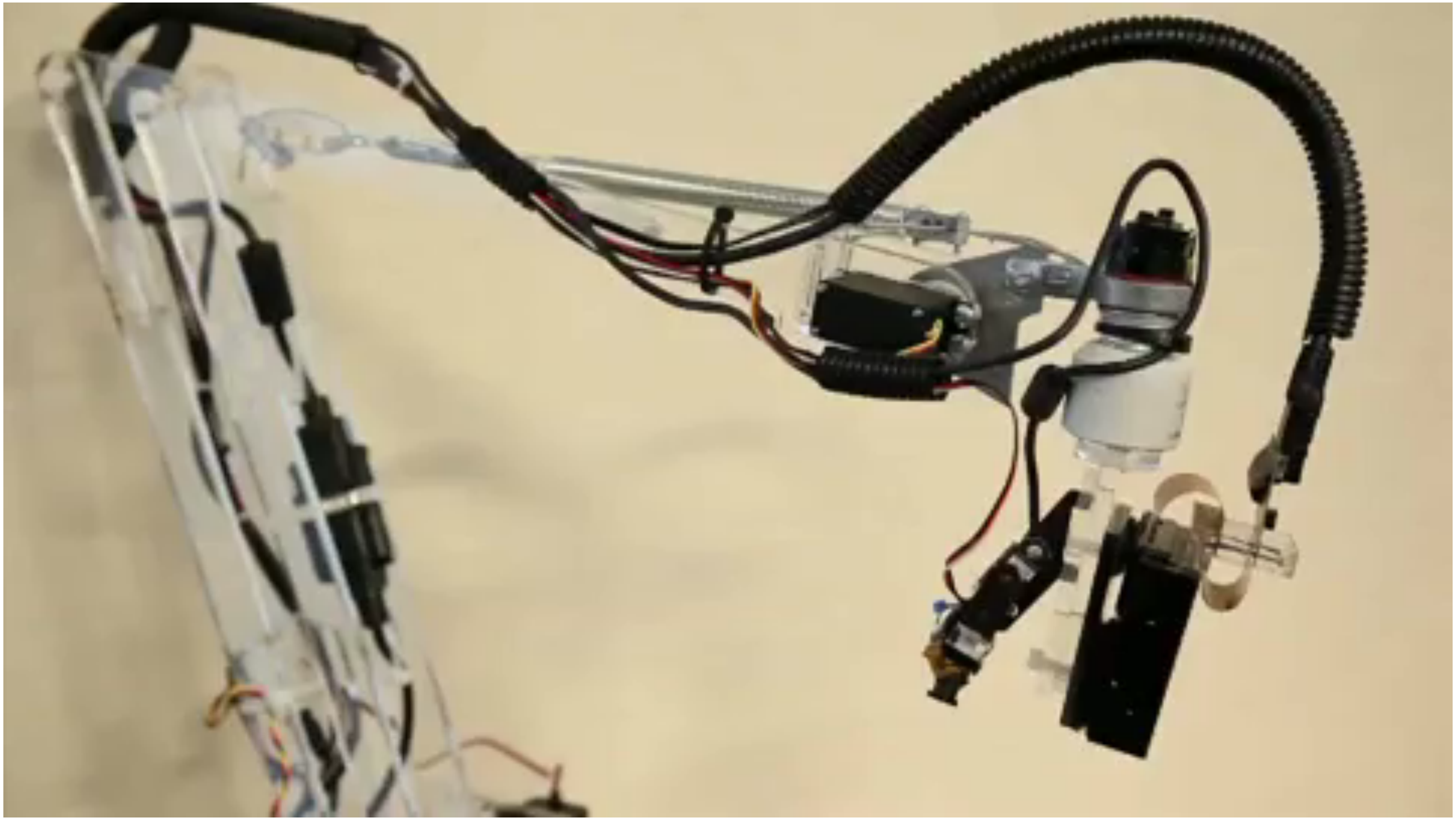
- LuminAR, MIT Media Lab
  - Actuated desk lamp projector
- Pinhanez: large steerable projector (on per room)
  - Large projector → cf. mainframe computer
- Steerable pico projector (one per desk)
  - Personal projector → cf. personal computer



Linder, Maes: LuminAR: Portable robotic augmented reality interface design and prototype. UIST 2010, Demo.

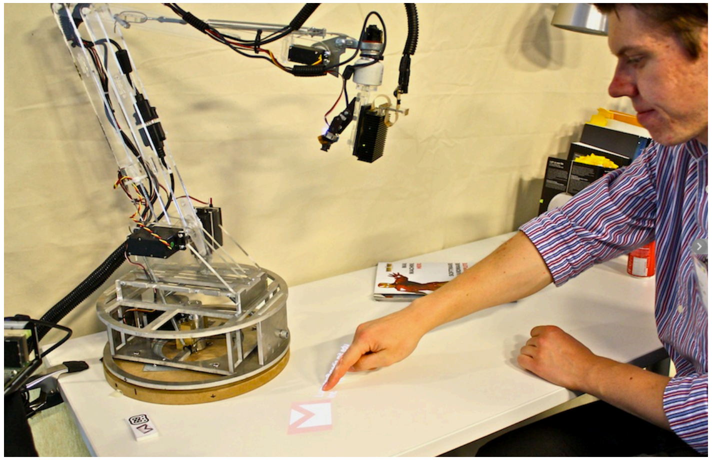
http://direct.media.mit.edu/people/natan/current/luminar.html

Actuated Pico Projector



http://www.youtube.com/watch?v=XV5V-dQW8CI

Actuated Pico Projector



Linder, Maes: LuminAR.

- Illuminate objects of interest
- Guide user's attention
  - on-tabletop
  - walls, shelf, door, etc.
- Tangible interaction with the projector itself
  - Physical input: move lamp
  - Virtual input: interact with projection
  - Physical output: actuation
  - Virtual output: projection
- Issues
  - Safety, mechanical stability, energy

Organization

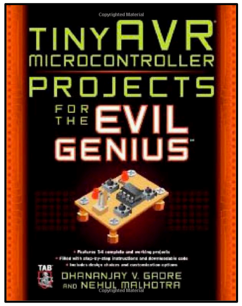
- **Objective:** Learn about embedded interactive systems
  - Just for fun, **no ECTS credits!**
- **Date:** Mondays 18-20
  - presentation and discussion of new topic
  - work on topic / project
- Schedule overview
  - 15 sessions
- Hardware components provided
  - May buy AVR programmer (15 EUR) and power supply (7 EUR)
  - May need to buy materials (e.g. used in laser cutter)

Schedule (preliminary)

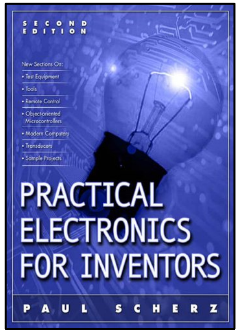
Date	Topic
17.10.	Introduction to embedded interaction, microcontrollers, hardware & software tools
24.10.	Soldering ISP adapter, AVR architecture, electronics basics, USB to serial chips
31.10.	LED displays, LED multiplexing, transistors, electronics basics
7.11.	AVR analog-digital-converter, sensors, op-amps
14.11.	PCB design & fabrication, EAGLE, 3D printing, OpenSCAD
21.11.	Actuation (servo / stepper motors), I2C: interfacing to other chips
28.11.	storage on memory cards, capacitive sensors
5.12.	OpenSCAD tutorial
12.12.	Eagle tutorial
19.12.	Displays (character LCDs, graphics LCDs), audio (speakers, amplification, op-amps)
9.1.	Communication: fixed-frequency RF, ZigBee, Bluetooth
...	
6.2.	Project

Books

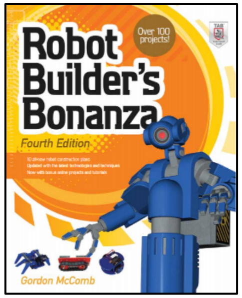
- Dhananjay Gadre, Nehul Malhotra: tinyAVR Microcontroller Projects for the Evil Genius, McGraw-Hill, 2011



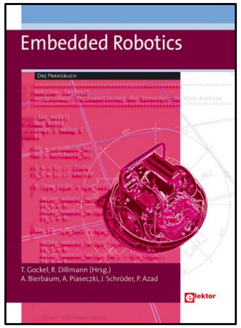
- Paul Scherz: Practical Electronics for Inventors, 2. Auflage, McGraw-Hill, 2006



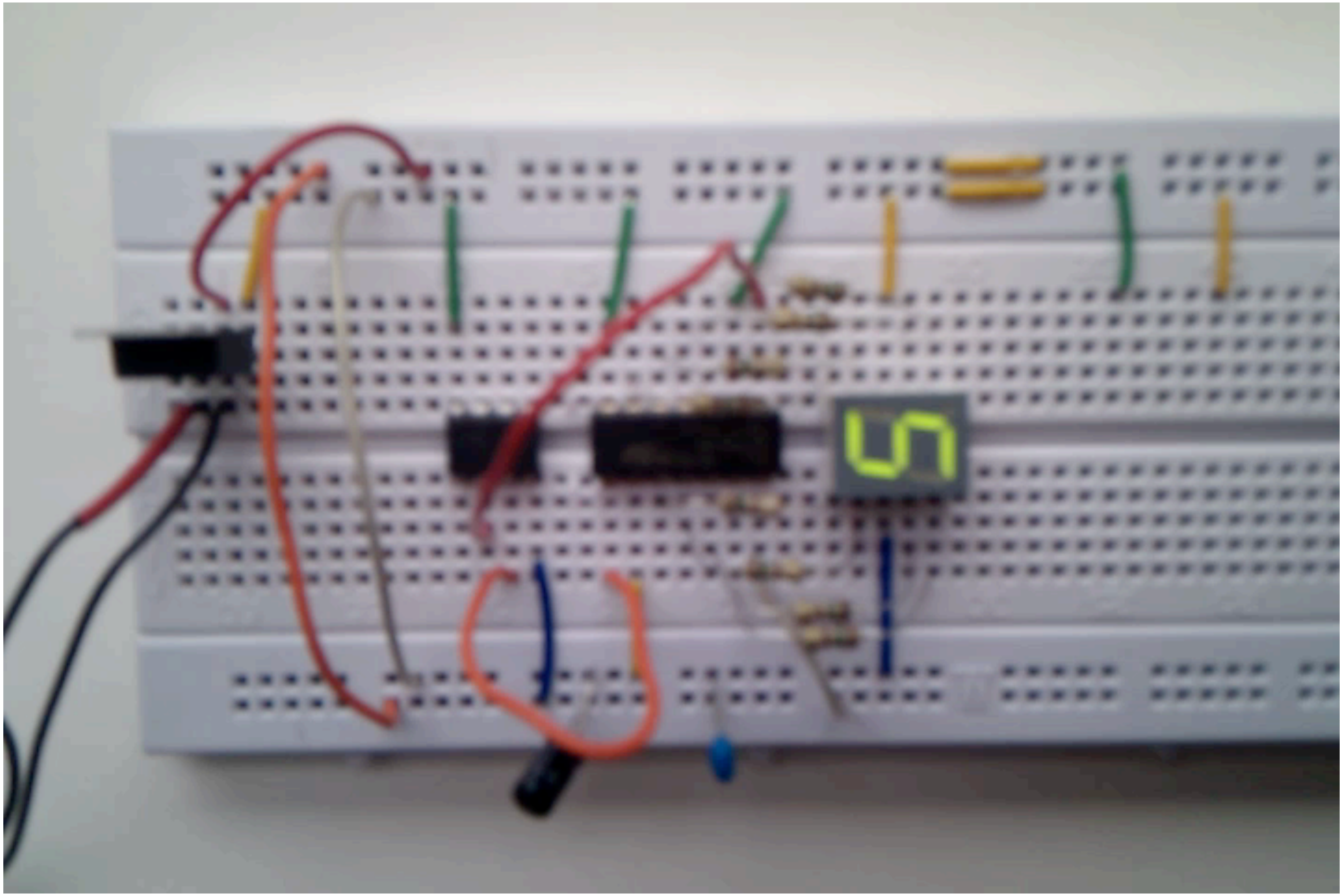
- Gordon McComb: Robot Builder's Bonanza, 4. Auflage, McGraw-Hill, 2011



- Alexander Bierbaum, Alexander Piaseczki, Joachim Schröder, Pedram Azad, Tilo Gockel, Rüdiger Dillmann: Embedded Robotics - Das Praxisbuch, Elektor-Verlag, 2005



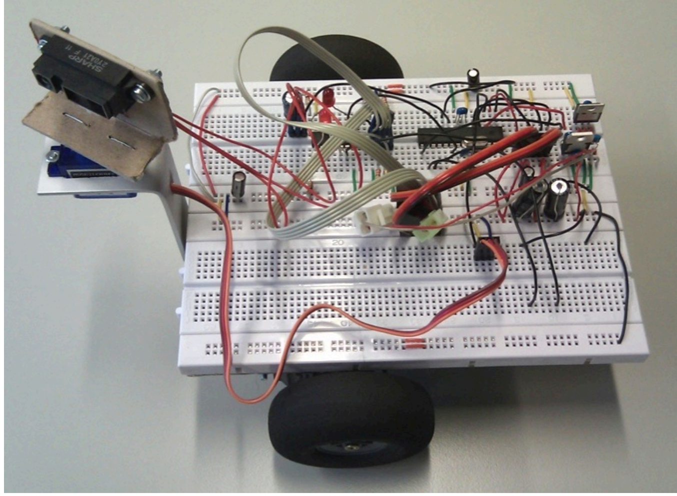
Simple Hardware…

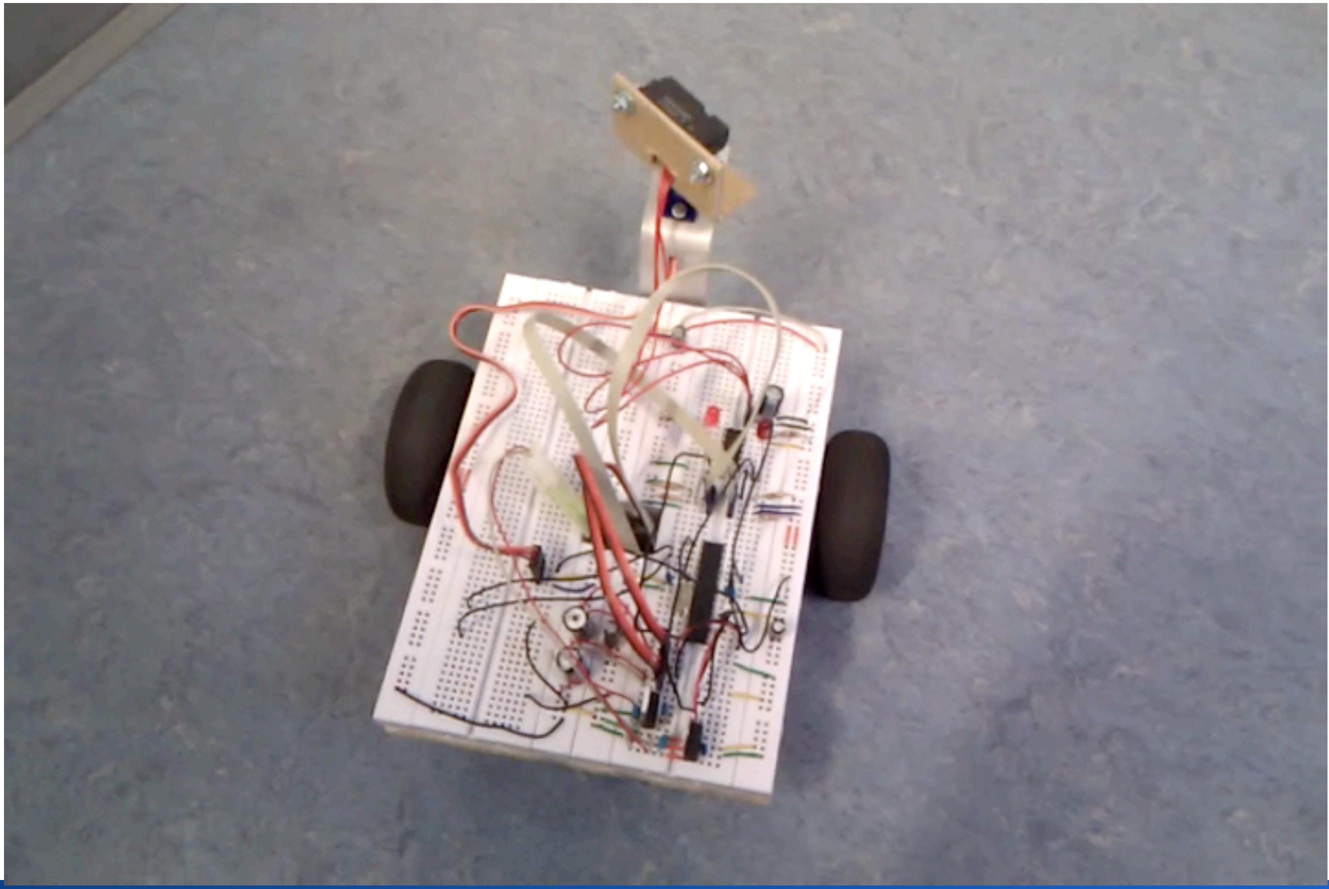


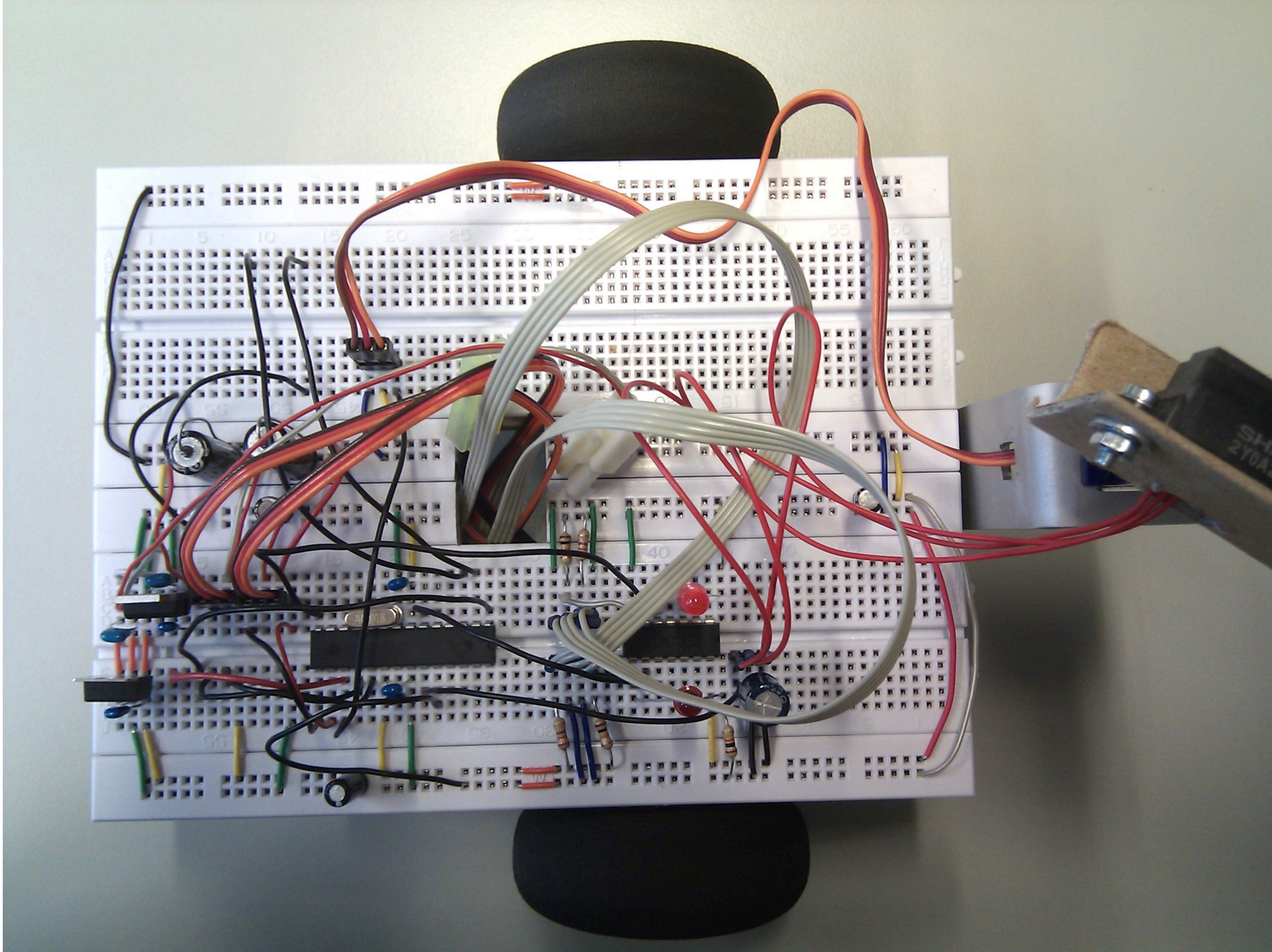
Actuation…

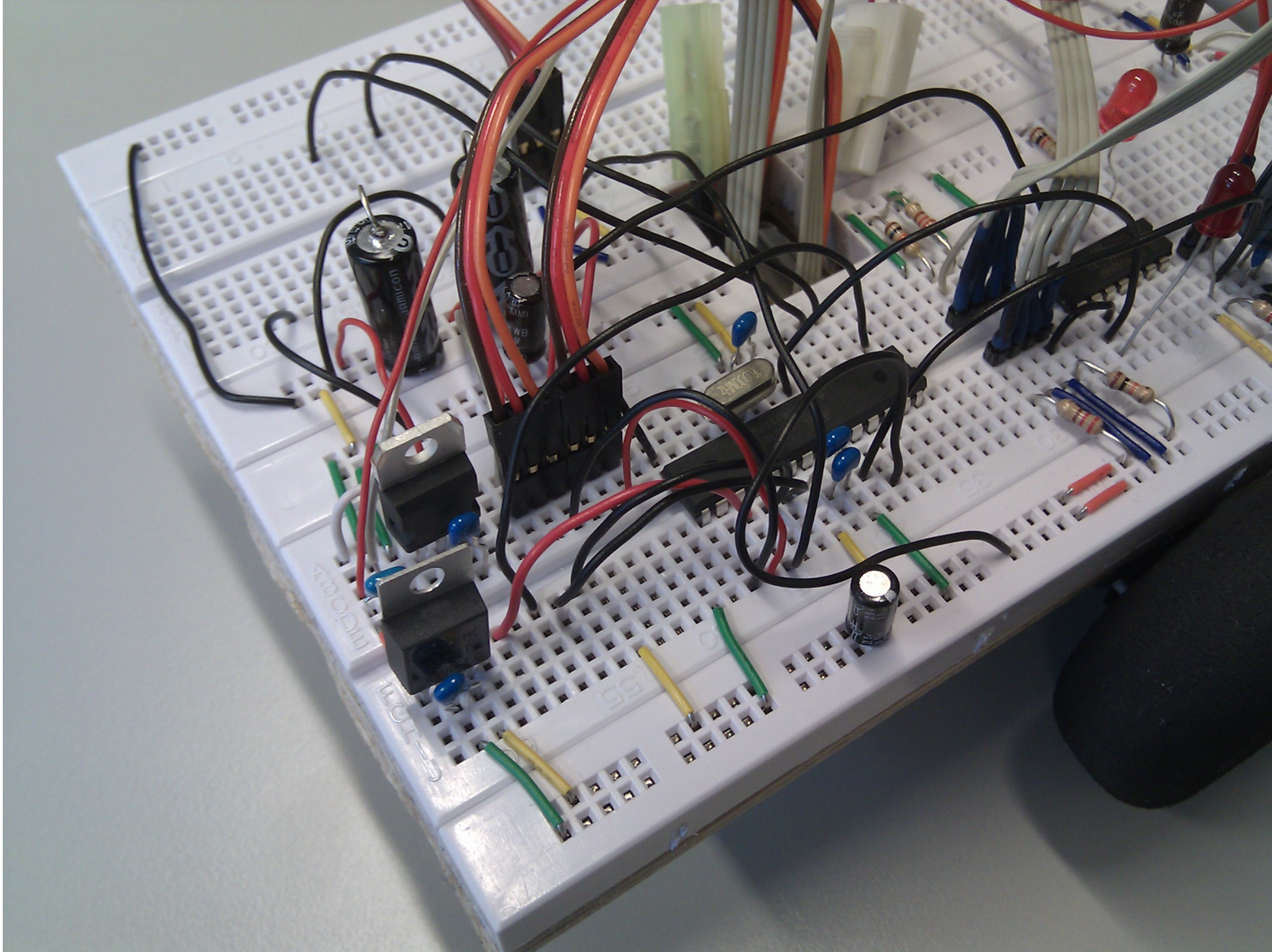
Mobile Robots

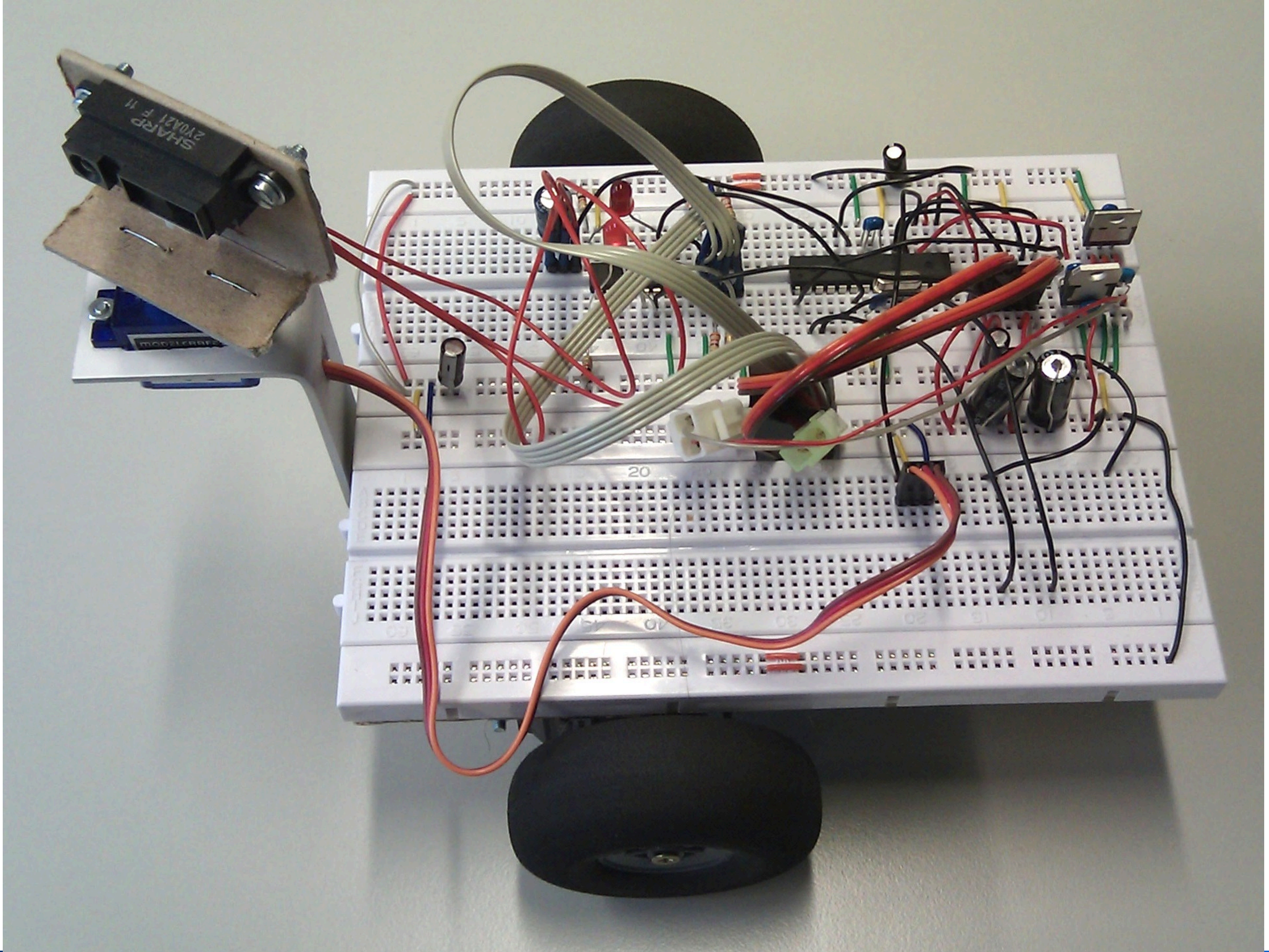
- Human-robot interaction hot topic
- Robot tasks
  - manual tasks
  - cleaning
  - communicate
  - observe

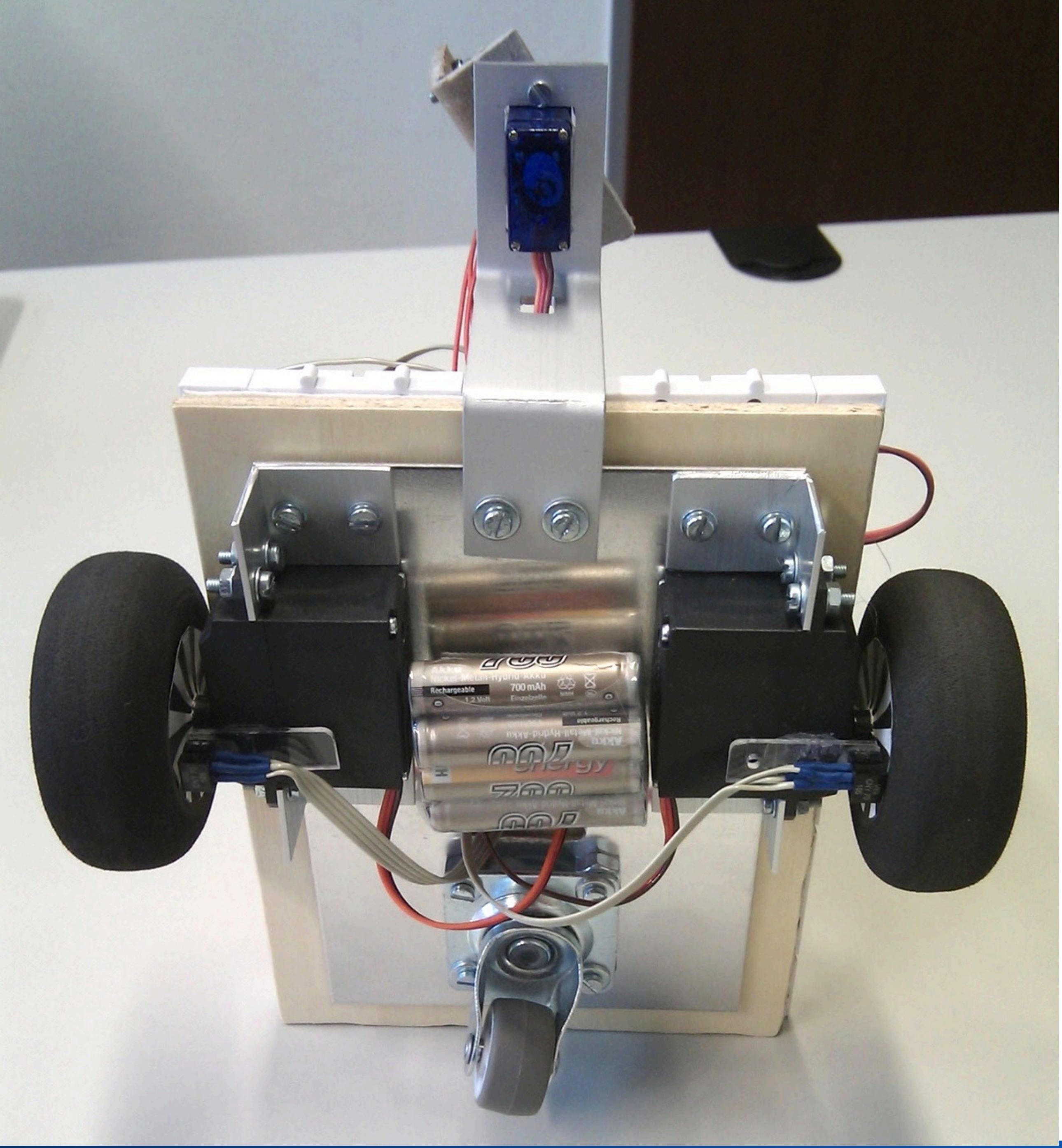


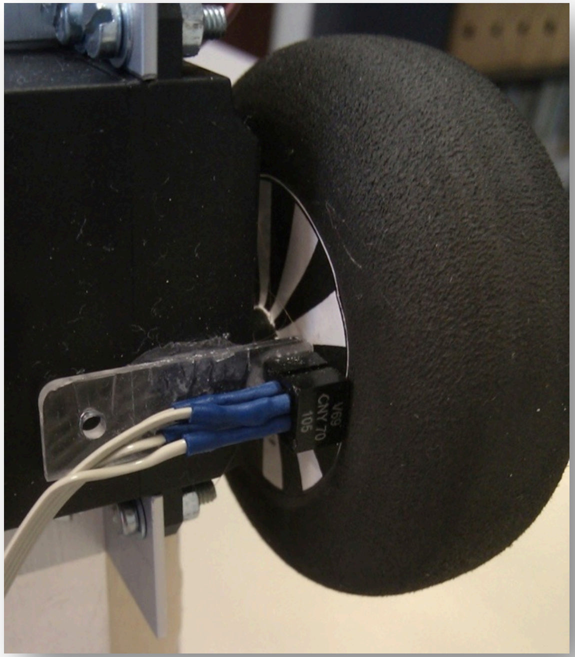


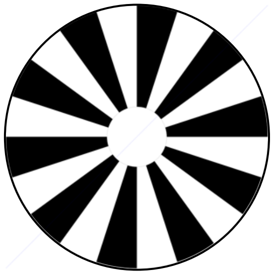





SHARP
2Y0A21 F 11
20
40

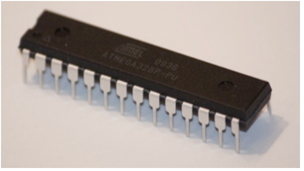


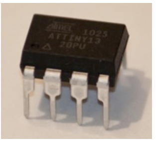




Rotary Encoder

Technologies and Tools

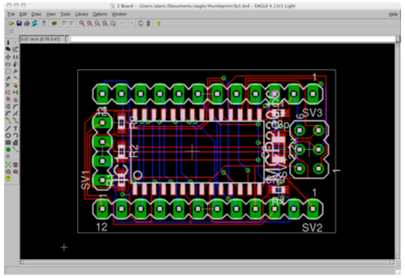




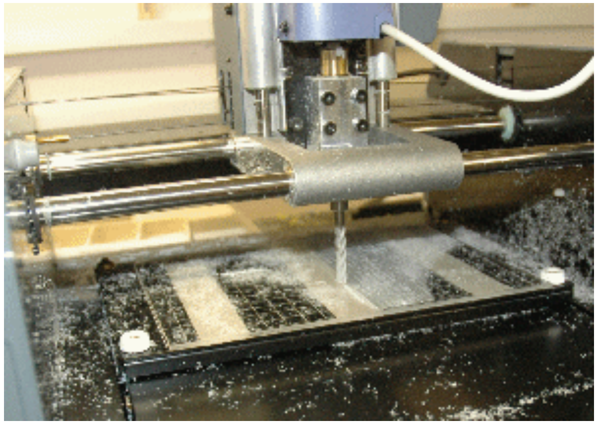
ATtiny, Atmega microcontroller

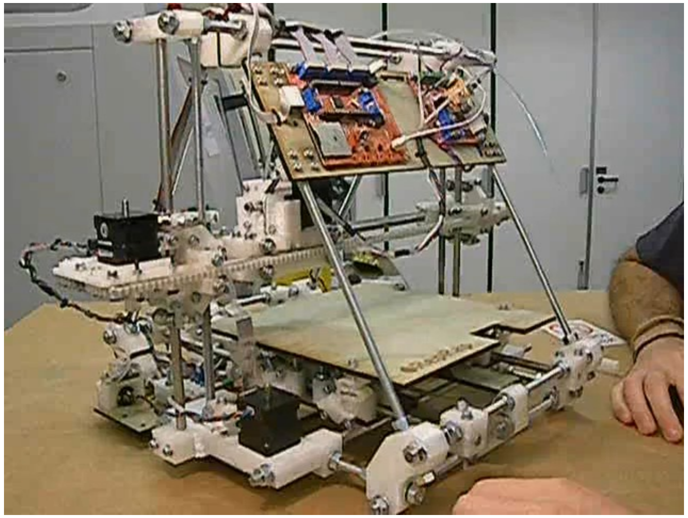
Milling, drilling, cutting PCB: Roland Modela

PCB Design: EAGLE



Printing casings: RepRap 3D printer



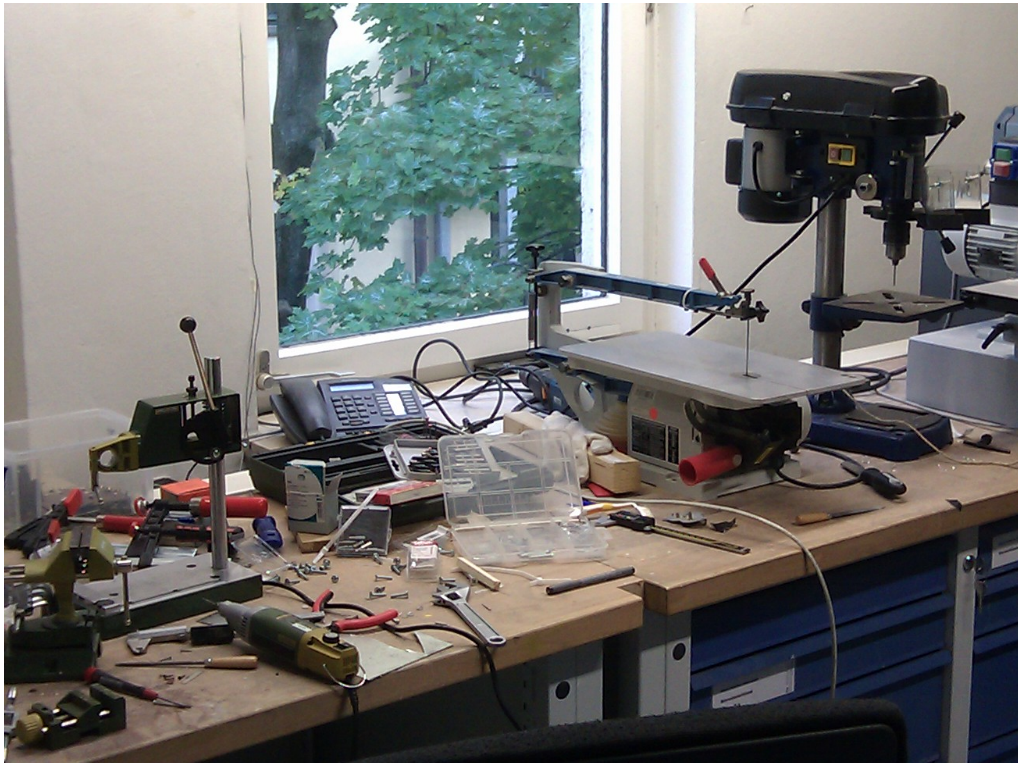


www.rolanddg.com/product/3d/3d/mdx-20_15/mdx-20_15.html

en.wikipedia.org/wiki/RepRap
www.reprap.org/wiki/Mendel

Tools

Workshop: Drilling, cutting, etc.



Laser cutter: Cutting, engraving

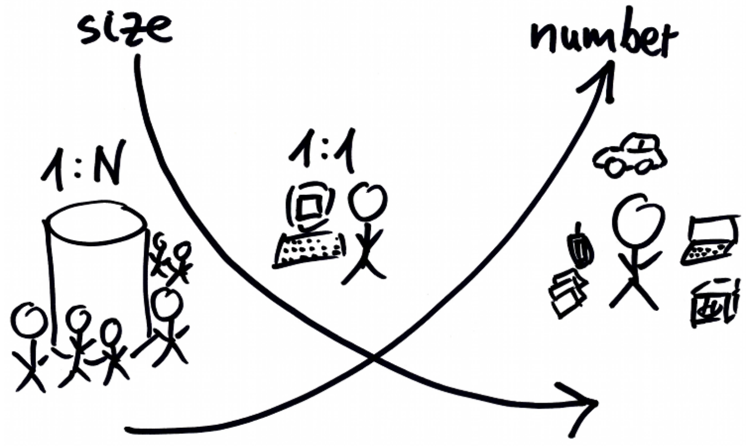
Embedded Systems

- Computer systems with dedicated functionality
  - Cf. general-purpose computer (PC)
  - Microcontrollers, digital signal processors, sensors, actuators
- Often not perceived as a “computer”
  - Users may not know that a computer system is inside
- Examples
  - Wrist watches, mp3 players, digital cameras, GPS receivers, bike computers, heart rate monitors, cars (motor, ABS, ESP), traffic lights, microwave ovens, dishwashers, washing machines, door openers, weather stations, TV sets, remote controls, DVD players, factory automation systems, telephone switches, networked thermostats, implantable medical devices, toys

Technological Enablers

- Processing & storage
  - Cheap, fast, reliable, small, large capacity, energy efficient
  - Moore's Law
- Networking
  - Cheap, fast, reliable, global, local, wireless, ad-hoc, low power
- Displays
  - Cheap, small, high quality, energy efficient, integrated
- Sensors & actuators
  - Cheap, small, accurate, invisible, many types

Ubiquitous Computing



"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

Mark Weiser

- Computers embedded in everyday things
- Technology moves into the background
- Computers in the world, instead of world in the computer

- Mobile devices as always available mediators
- Entry point into the digital world

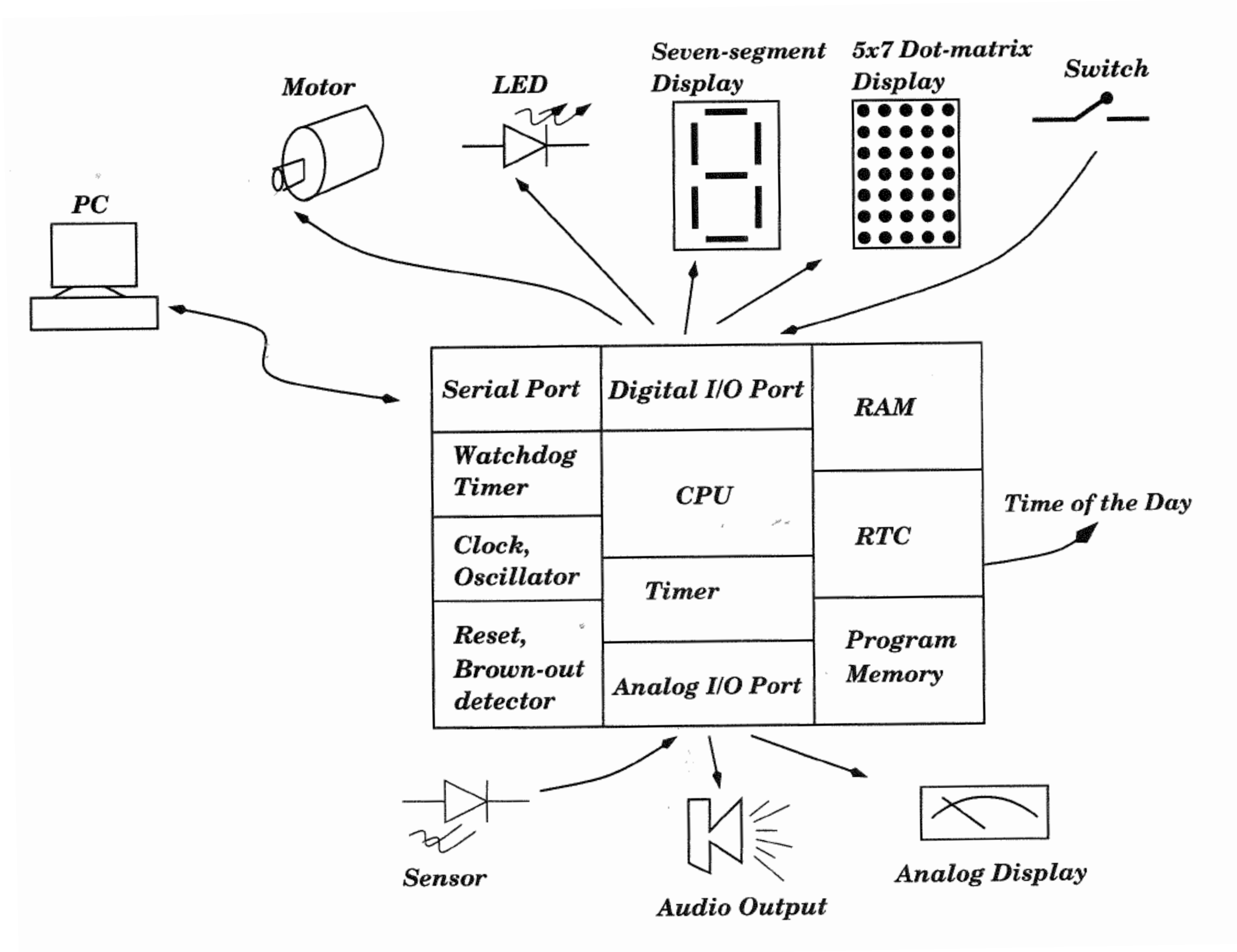
Embedded & Tangible Interaction

- Challenges for human-computer interaction
  - How to interact with so many systems?
  - How to keep users from constant interruptions and distractions?
  - Device interaction happens in an everyday situation. How to take that into account?
  - What are novel forms of interaction?
  - Design opportunities?
- Interaction themes
  - Natural interfaces
  - Context-aware applications
  - Automatic capture and access
  - Continuous interaction

Microcontrollers

- Integrates processor, memory, I/O peripherals, and sensors on a single chip
  - Replaces many traditional hardware components in a single chip
  - Lower cost, fewer additional components, smaller circuit board
  - Very memory efficient (sleep modes)
  - Software flexibility through software
- Memory types
  - Flash: program
  - RAM: working memory (stack, heap)
  - EEPROM: non-volatile memory
- Interrupt-driven I/O
  - Sources: signal changes, timer overflow, ADC conversion done
  - Interrupts can wake microcontroller from low-power sleep state

Microcontrollers



Source: Gadre, Malhotra: tinyAVR projects

Microcontrollers

- I/O Pins
  - Used as input or output (controlled by software)
  - Serial communications (UART, I^2C, SPI)
  - Signal generation (PWM, timers)
  - Analog input (ADC conversion)
- Development
  - In-circuit programming and debugging, field update of firmware
  - Programming in assembly language or C
- Selectable clock frequencies
  - Lower clock rate → less energy
- No floating point unit (typically)

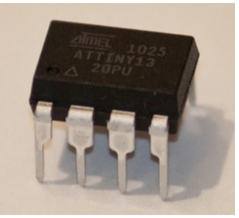
Atmel AVR: ATtiny, ATmega

- 8-bit RISC chip, Harvard architecture
- ATtiny

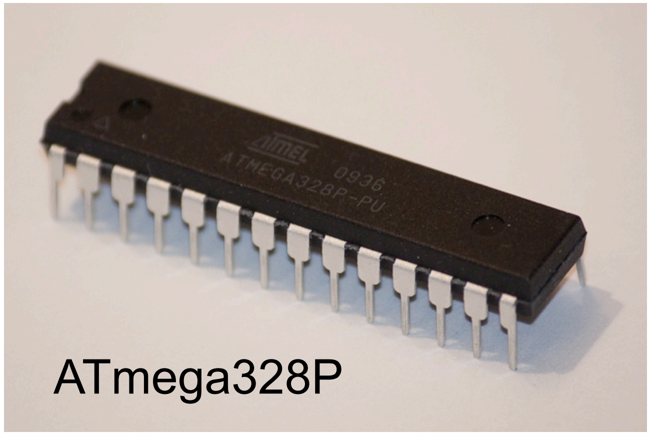
  1–8 kB program memory

  6–32-pin package

  ATtiny13

  www.atmel.com/dyn/products/param_table.asp?category_id=163&family_id=607&subfamily_id=791
- ATmega

  ATmega328P

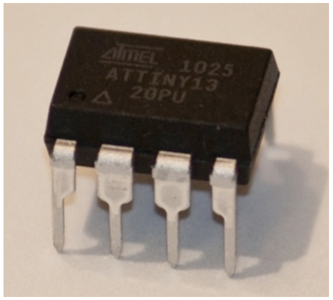
  4–256 kB program memory

  28–100-pin package

  Extended instruction set
    - Multiply instructions
    - Handling larger program memories

  www.atmel.com/dyn/products/param_table.asp?category_id=163&family_id=607&subfamily_id=760
- Large family of devices, specific features

Many types of AVRs: Choose depending on required features



ATtiny13

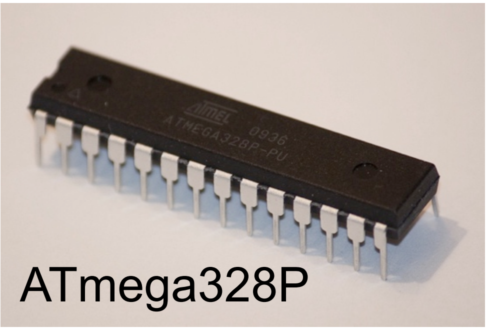
ATtiny13

- 6 I/O pins, 1.8-5.5V operation
- 20 MPIS @ 20 MHz (clock rate selectable), internal oscillator
- 64B RAM, 64B EEPROM, 1kB Flash program memory
- 8-bit timer, 2 PWM channels, 10-bit ADC, analog comparator
- Price: €1.15

ATtiny45

- 6 I/O pins, 1.8-5.5V operation
- 20 MPIS @ 20 MHz (clock rate selectable), internal oscillator
- 256B RAM, 256B EEPROM, 4kB Flash program memory
- 2 8-bit timers, 4 PWM channels, 10-bit ADC, analog comparator, SPI, TWI, temperature sensor
- Price: €2.05

Many types of AVRs: Choose depending on required features



ATmega8

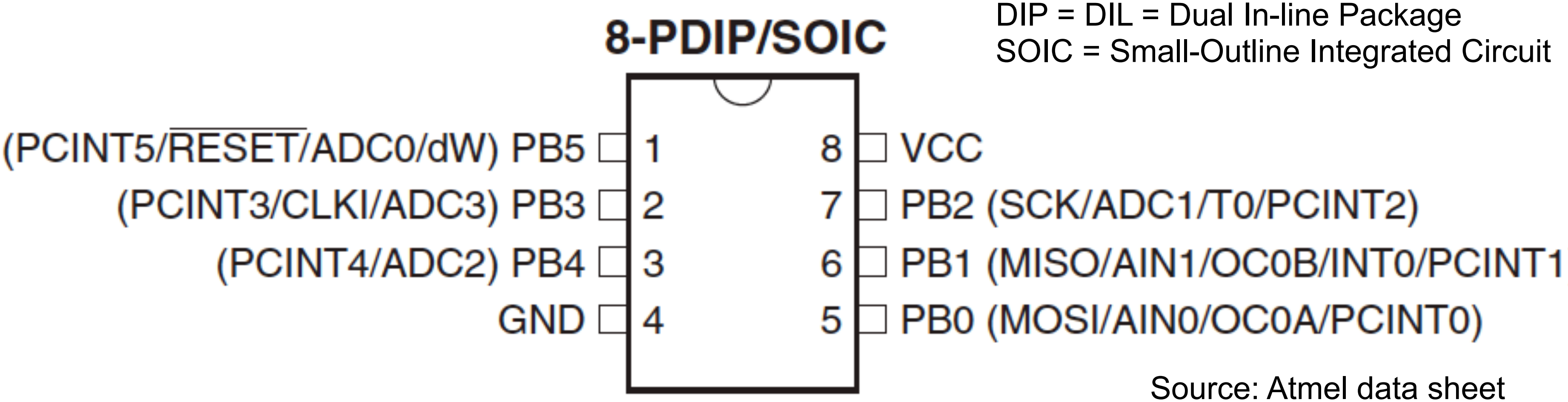
- 23 I/O pins, 2.7-5.5V operation
- 16 MPIS @ 16 MHz (clock rate selectable), internal oscillator
- 1kB RAM, 512B EEPROM, 8kB Flash program memory
- 2 8-bit timers, 1 16-bit timer, 3 PWM channels, 10-bit ADC, analog cmp., SPI, TWI, USART
- Price: €2.60

ATmega328P

- 23 I/O pins, 1.8-5.5V operation
- 20 MPIS @ 20 MHz (clock rate selectable), internal oscillator
- 2kB RAM, 1kB EEPROM, 4kB Flash program memory
- 2 8-bit timers, 1 16-bit timer, 6 PWM channels, 10-bit ADC, analog cmp., SPI, TWI, USART, temperature sensor
- Price: €3.30

Pinout ATtiny13

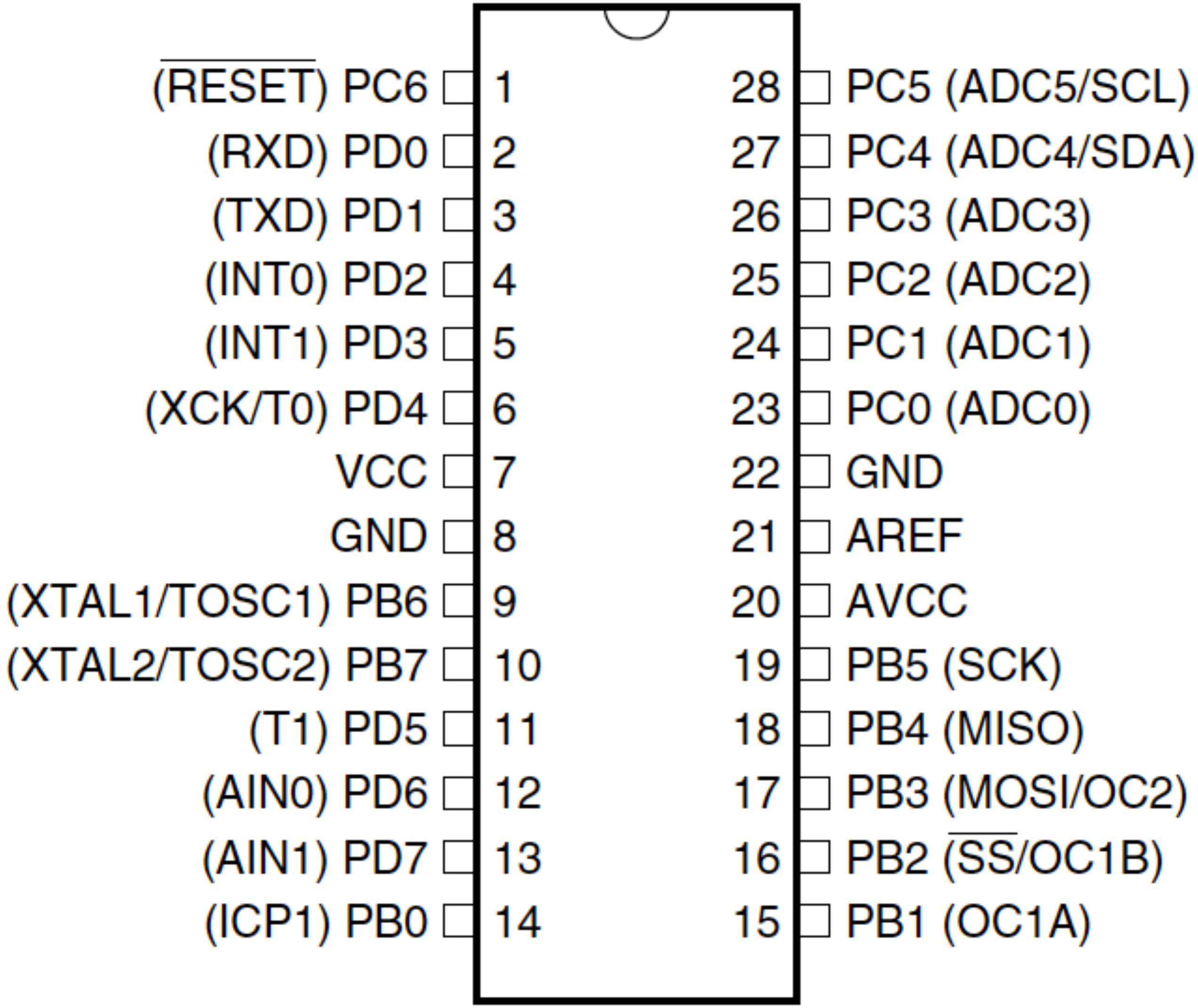
DIP = DIL = Dual In-line Package
SOIC = Small-Outline Integrated Circuit



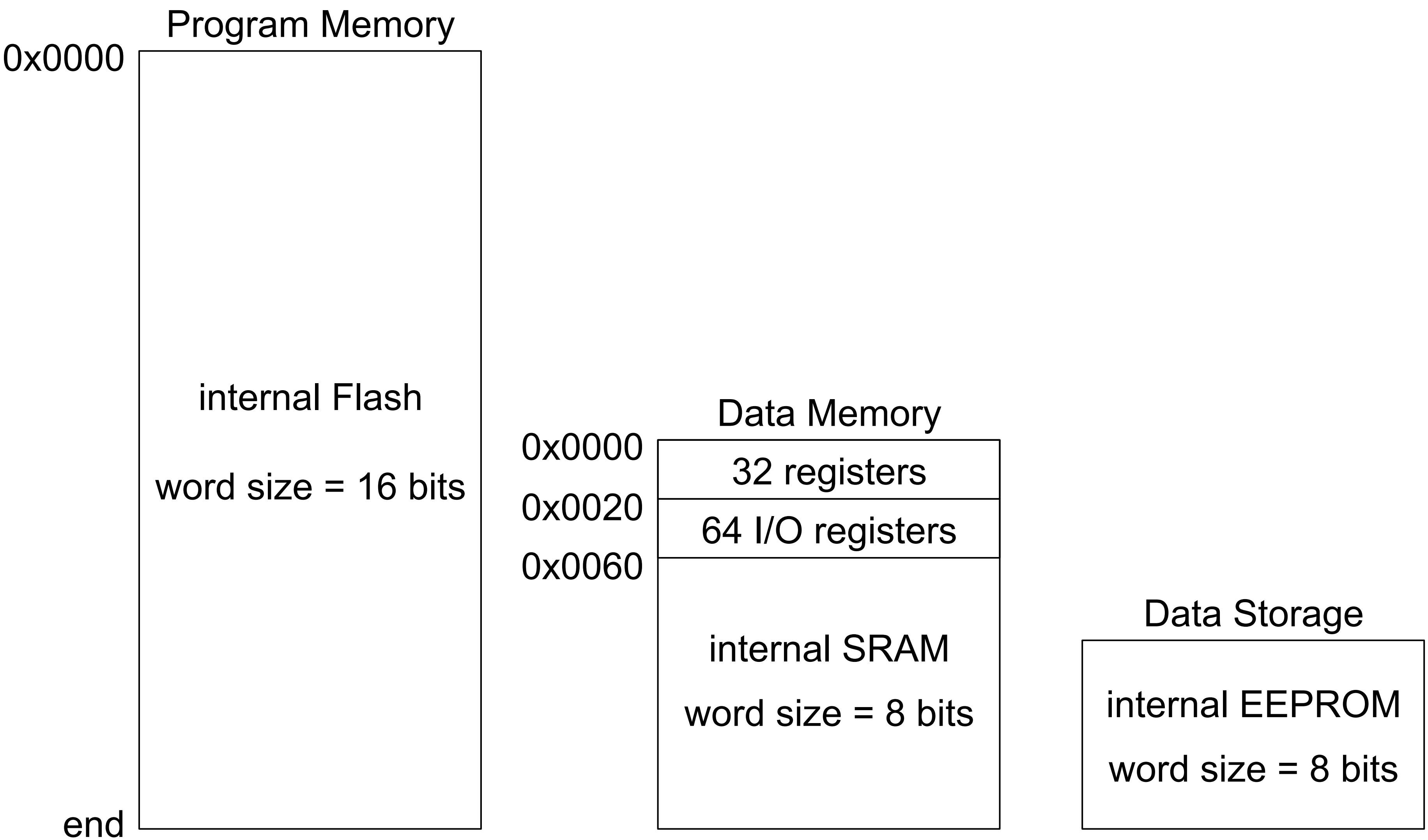
Source: Atmel data sheet

- Multiplexed pin functions, software configurable
  - Example: Flash/EEPROM programming via SPI:
    MOSI = master out, slave in (from programmer to ATtiny)
    MISO = master in, slave out (from ATtiny to programmer)
    SCK = serial clock
  - Example: ADC1 = ADC input channel 1
  - Example: PCINT3 = pin change interrupt 3

Pinout ATmega8

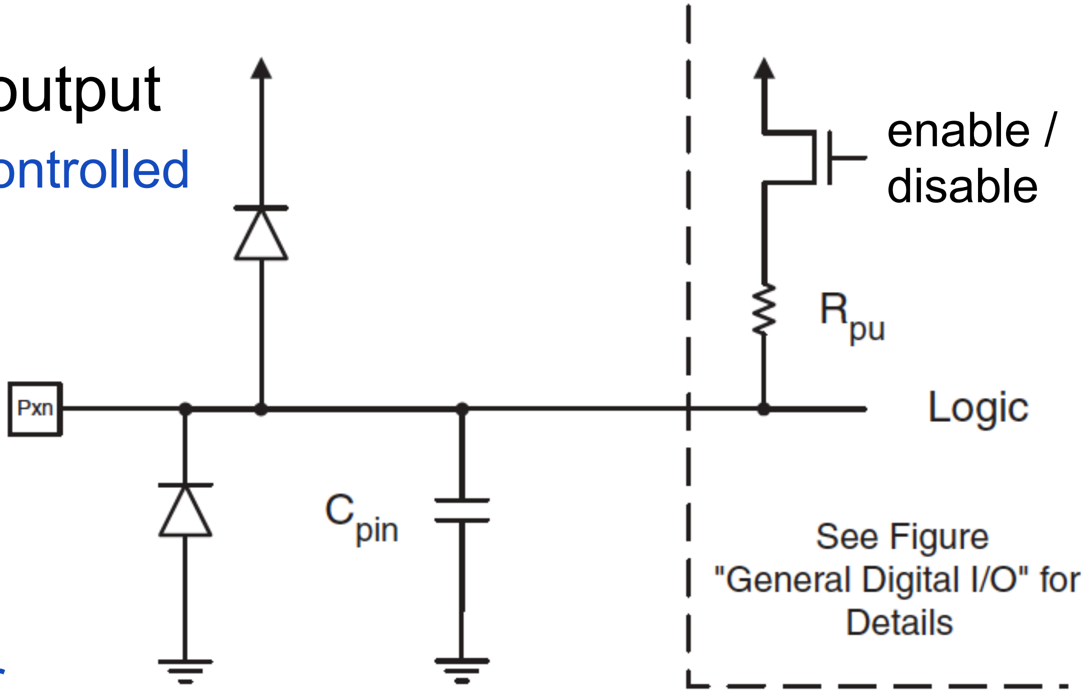


AVR Memory Layout



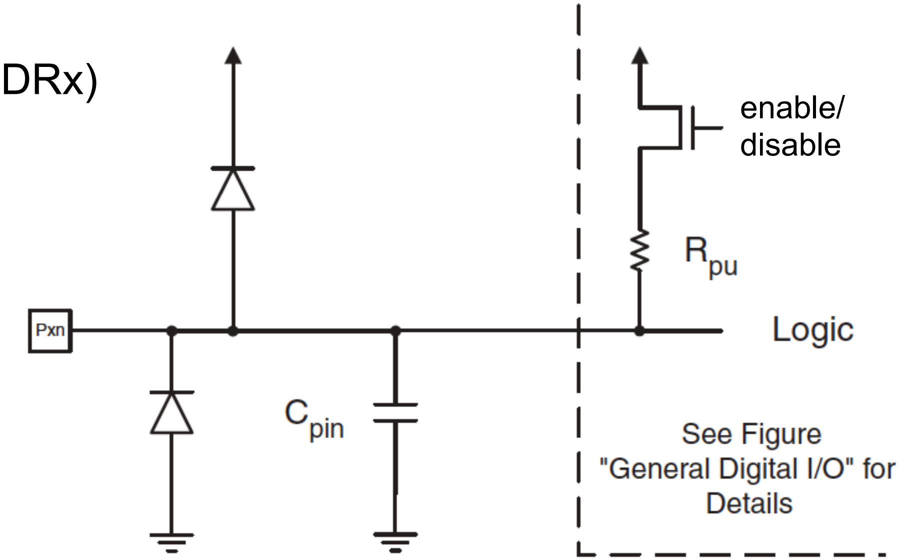
AVR I/O Ports

- I/O pin either input or output
  - Individually software-controlled
- Pin as output
  - States: low, high
  - Can drive 40mA (→ LED)
- Pin as input
  - Internal pull-up resistor (enabled/disabled in software)
  - high resistance state (high-Z) if pull-up disabled



Accessing the I/O Ports

- Three memory addresses for each I/O port
  - Data Direction Register: DDRx
    - 1 = output
    - 0 = input
  - Data Register: PORTx
    - if input: 1 = pull-up enabled, 0 = pull-up disabled
    - if output: 1 = PIN driven high, 0 = PIN driven low
  - Port Input Pins: PINx
    - read: PIN state (independent of DDRx)
    - write 1: toggles PORTx



AVR I/O Ports: Pin Control Example

PIN	0	1	2	3	4	5	6	7
in/out	out	out	out	out	in	in	in	in
value	1	1	0	0	pullup	hi-z	hi-z	hi-z

Assembly

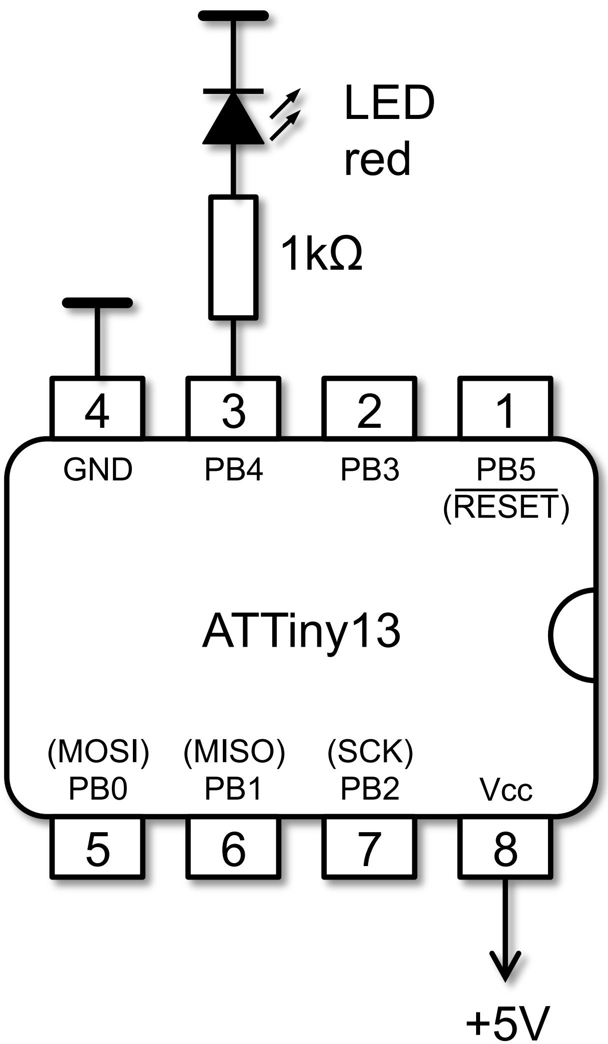
```
ldi r16, (1<<PB4) | (1<<PB1) | (1<<PB0)
ldi r17, (1<<DDB3) | (1<<DDB2) |
         (1<<DDB1) | (1<<DDB0)
out PORTB,r16
out DDRB,r17
nop      // synchronization
in r16,PINB
```

C

```
unsigned char i;
PORTB = (1<<PB4) | (1<<PB1) | (1<<PB0);
DDRB = (1<<DDB3) | (1<<DDB2) |
       (1<<DDB1) | (1<<DDB0);

__no_operation(); // synchronization
i = PINB;
```

"µC Hello World": Blinking an LED

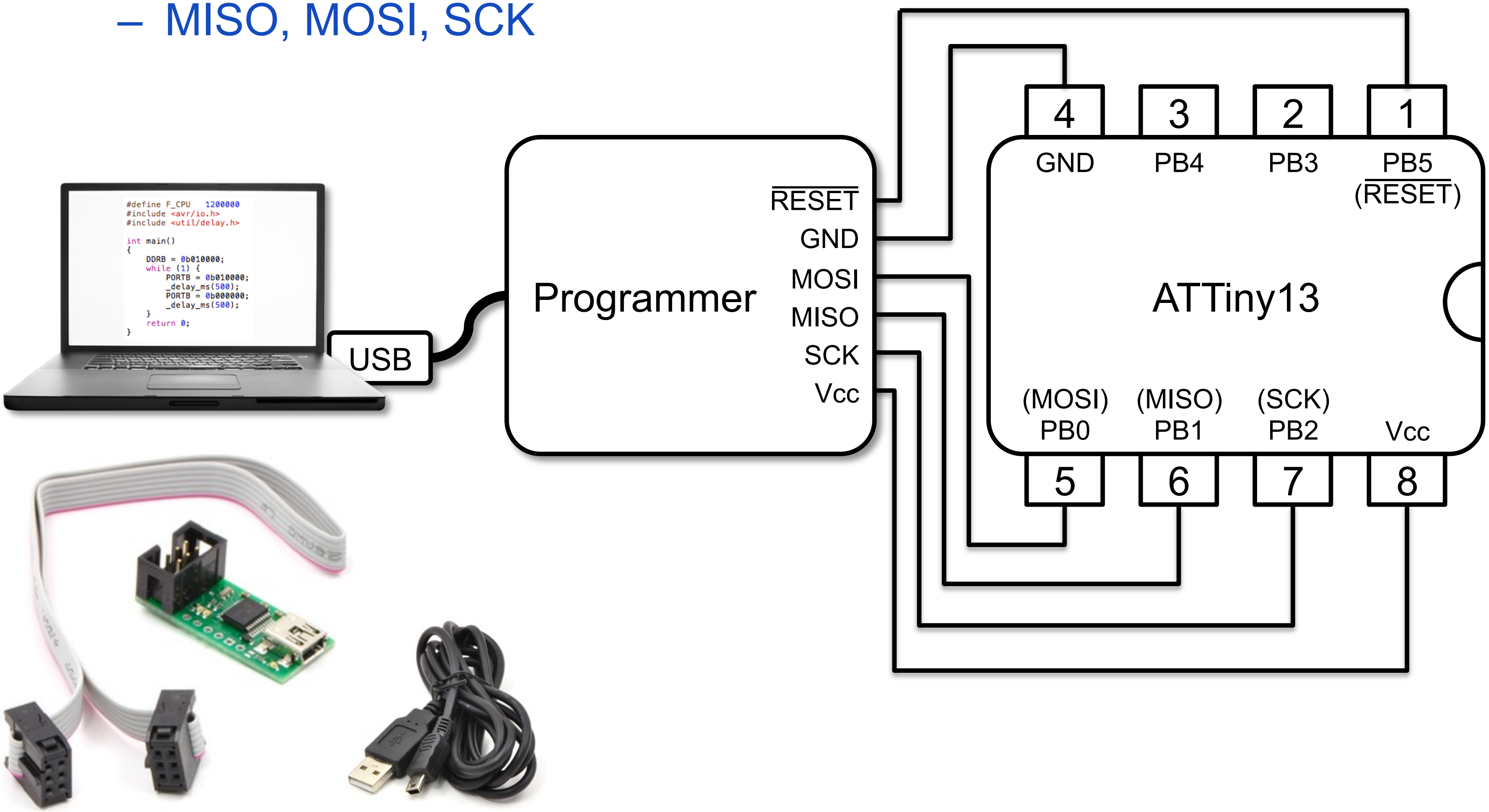



```
#define F_CPU 1200000
#include <avr/io.h>
#include <util/delay.h>

int main()
{
    DDRB = 0b010000;
    while (1) {
        PORTB = 0b010000;
        _delay_ms(500);
        PORTB = 0b000000;
        _delay_ms(500);
    }
    return 0;
}
```

Downloading the Program to the µC

- Serial programming via Serial Peripheral Interface (SPI)
  - MISO, MOSI, SCK



Memory Programming

- Tasks
  - Download/upload program code to/from Flash memory
  - Download/upload data to/from internal EEPROM
  - Configuring the microcontroller (“fuse bits”)
- Programming options
  - Serial programming
    - In-system programming (ISP)
    - High-voltage serial programming (HVSP, only 8-pin controllers)
  - High-voltage parallel programming
    - If RESET pin used as I/O pin: high-voltage programming
  - debugWire on-chip debug system
    - Uses RESET pin for debugging and Flash/EEPROM programming

AVR Configuration via "Fuse Bits"

- AVRFuses tool
  - http://www.vonnieda.org/software/avrfuses
- Online fuse calculator
  - http://www.engbedded.com/fusecalc/
- ATtiny13 datasheet, 17.2 Fuse Bytes
  - ATtiny13 has two fuse bytes
  - Default: high byte = 0b11111111, low byte = 0b01101010

Table 17-3. Fuse High Byte

Fuse Bit	Bit No	Description	Default Value
–	7	–	1 (unprogrammed)
–	6	–	1 (unprogrammed)
–	5	–	1 (unprogrammed)
SELFPRGEN(1)	4	Self Programming Enable	1 (unprogrammed)
DWEN(2)	3	debugWire Enable	1 (unprogrammed)
BODLEVEL1(3)	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0(3)	1	Brown-out Detector trigger level	1 (unprogrammed)
RSTDISBL(4)	0	External Reset disable	1 (unprogrammed)

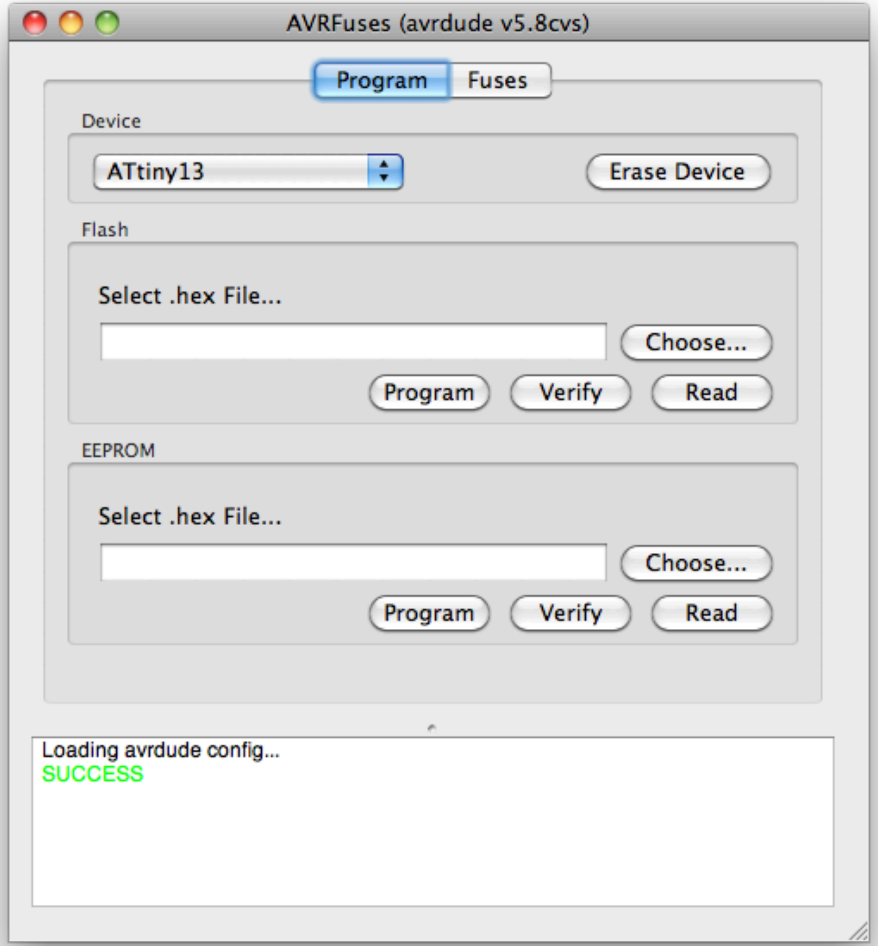
Table 17-4. Fuse Low Byte

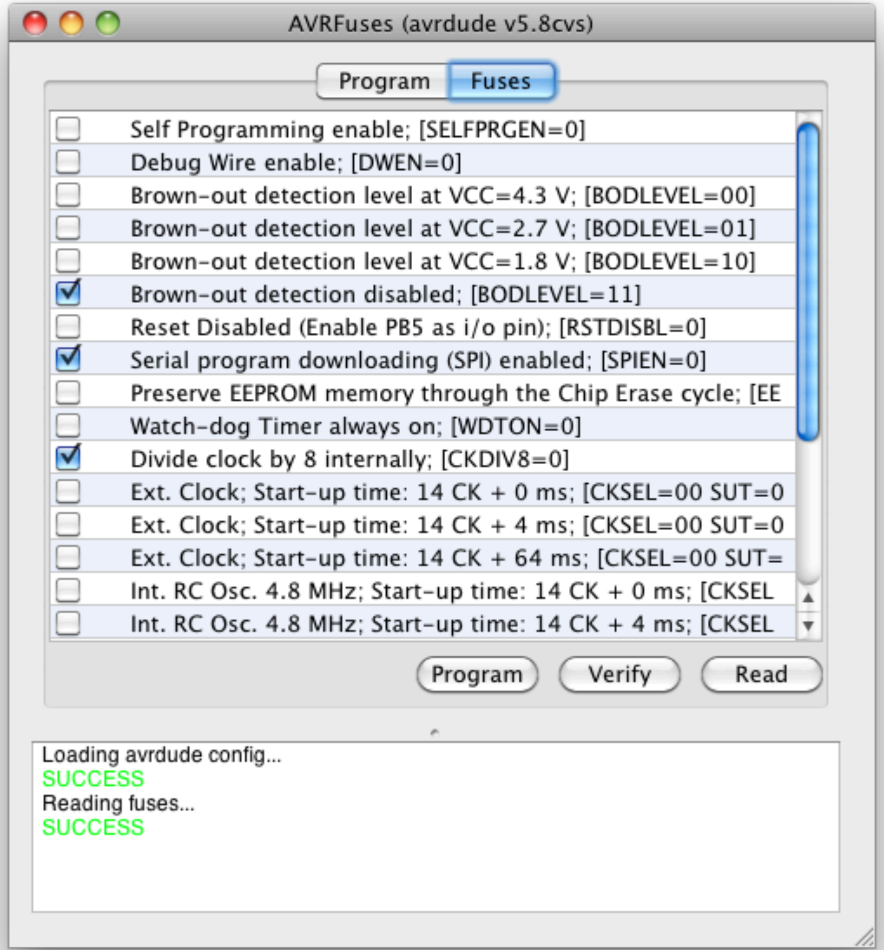
Fuse Bit	Bit No	Description	Default Value
SPIEN(1)	7	Enable Serial Programming and Data Downloading	0 (programmed) (SPI prog. enabled)
EESAVE	6	Preserve EEPROM memory through Chip Erase	1 (unprogrammed) (memory not preserved)
WDTON(2)	5	Watchdog Timer always on	1 (unprogrammed)
CKDIV8(3)	4	Divide clock by 8	0 (programmed)
SUT1(4)	3	Select start-up time	1 (unprogrammed)
SUT0(4)	2	Select start-up time	0 (programmed)
CKSEL1(5)	1	Select Clock source	1 (unprogrammed)
CKSEL0(5)	0	Select Clock source	0 (programmed)

AVR Configuration via “Fuse Bits”

Caution: Wrong fuse bit settings may render chip unusable!

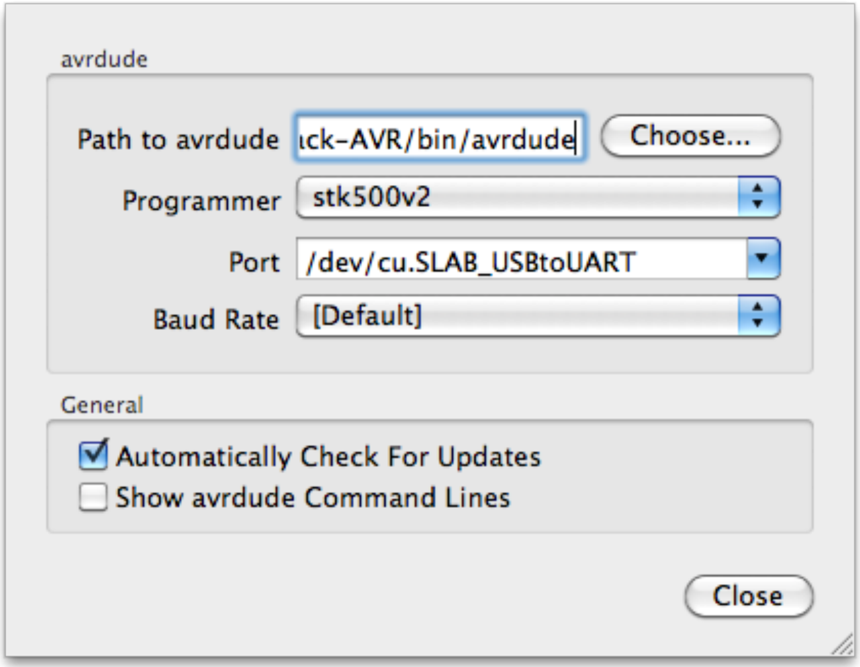
Tool: AVRFuses (www.vonnieda.org/AVRFuses/)





Configuring AVRFuses for the Programmer and USB Port

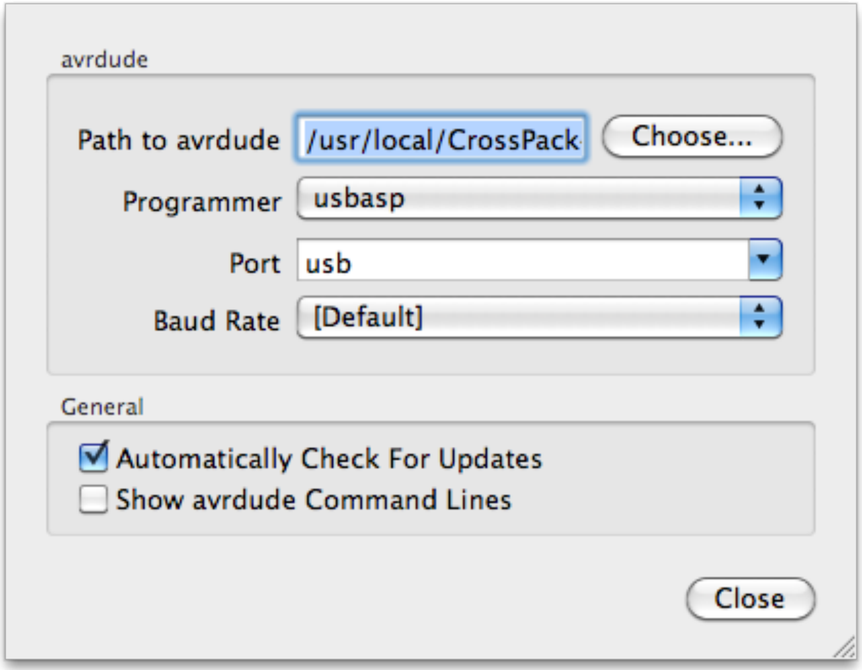
mySmartUSB light:



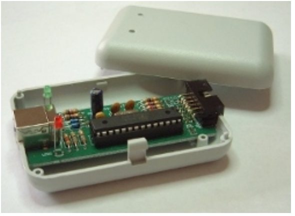
/dev/cu.SLAB_USBtoUART

http://shop.myavr.ch/index.php?sp=article.sp.php&artID=200006

USBasp:



http://www.fischl.de/usbasp/



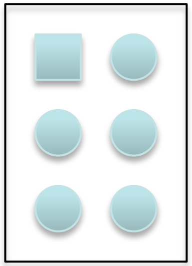
USB Drivers for “mySmartUSB light”

- USB chip CP2102 from Silicon Laboratories
- Windows

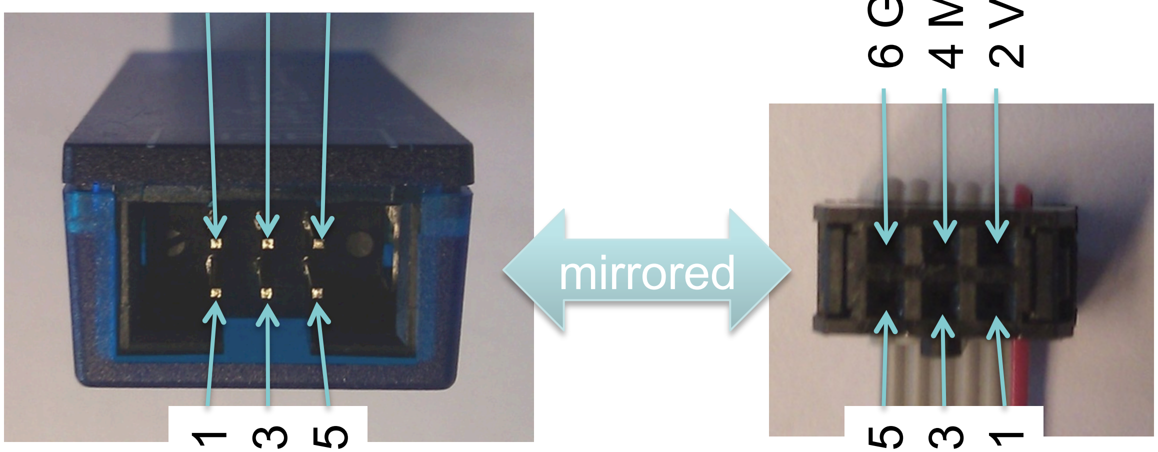
  http://shop.myavr.ch/index.php?sp=article.sp.php&artID=200006
- Mac OS X, Linux

  http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx

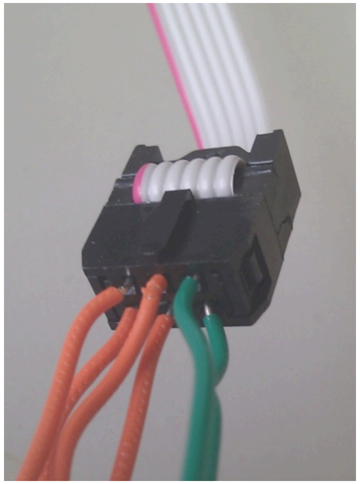
AVR ISP Connector



- Image of small PCB with one row of connecotrs
- http://itp.nyu.edu/physcomp/Tutorials/AVRCProgramming-Programmer

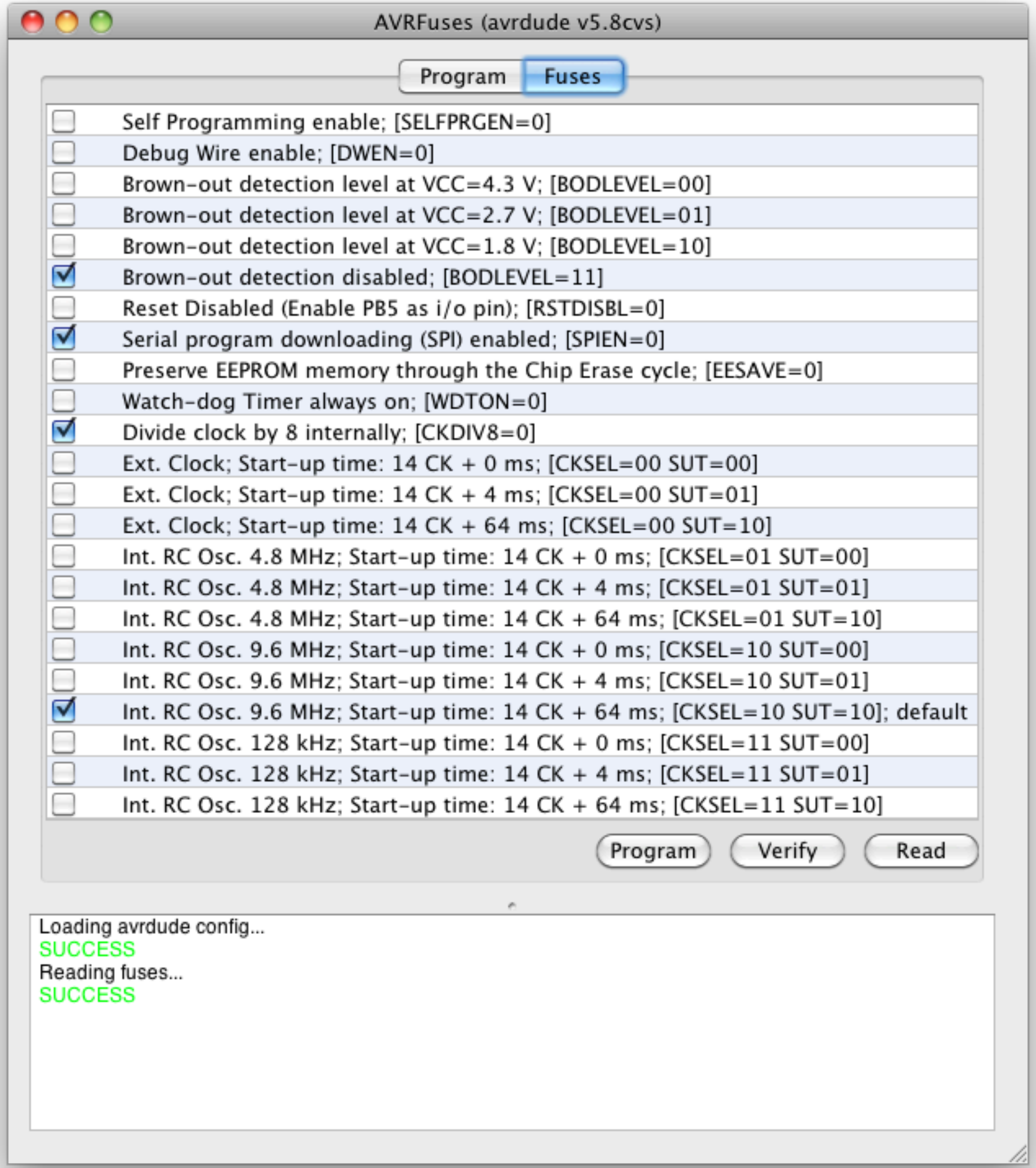


not recommended:



better solution:
solder small PCB
with 6x1 pins

- Fuses show factory configuration of ATtiny13
- Brown-out detection
  - reset when Vcc below level
- Reset disabled
  - use reset pin as I/O pin: dangerous!
- Start-up time
  - delay until conditions are stable



AVR Clock Options

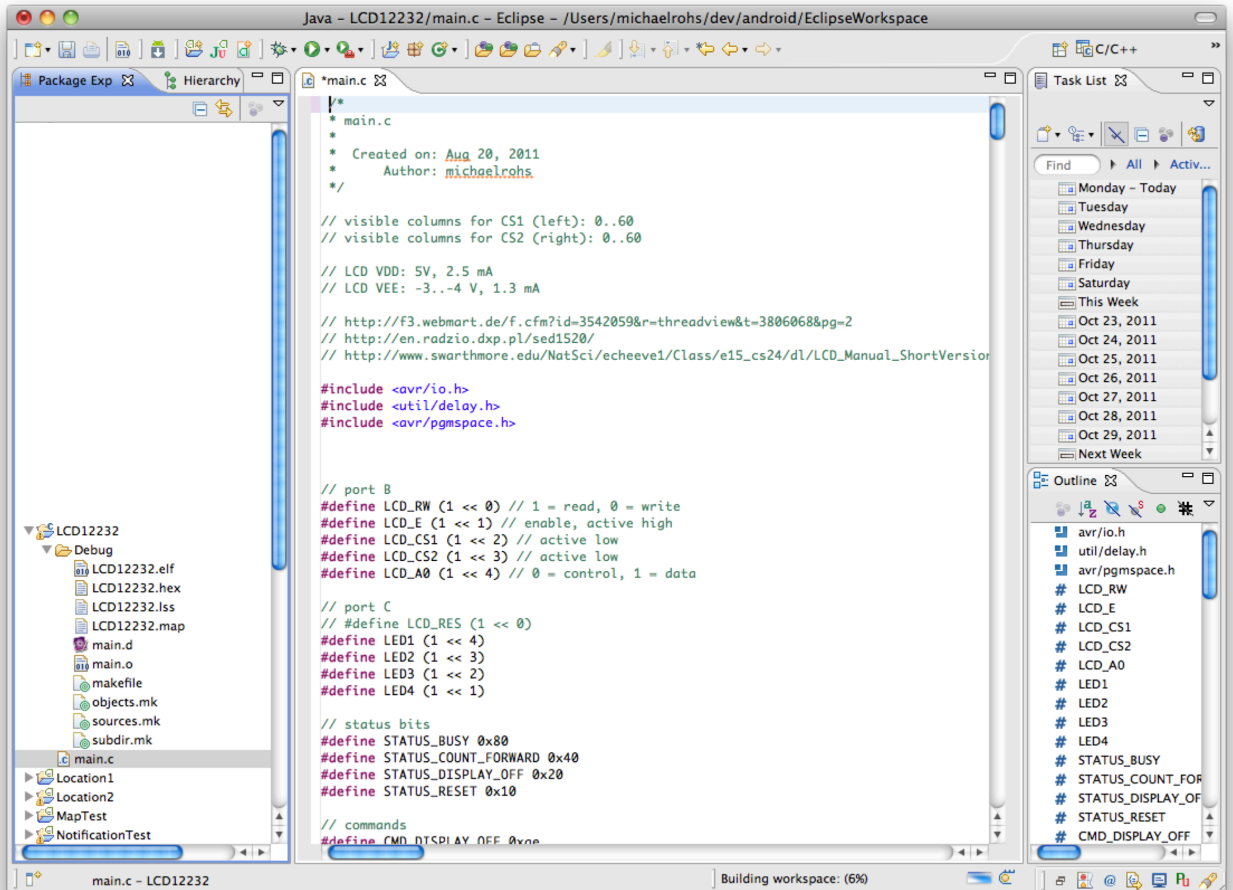
- Clock frequency can be chosen
  - Application requirements, power consumption
  - Clock prescaler register (divide clock by factor)
  - Component clocks can be disabled to reduce power consumption
- Clock source can be chosen
  - Internal resistor capacitor (RC) oscillator
    - Convenient, but not precise (temperature, operating voltage)
    - ATtiny13: 4.8MHz, 9.6MHz (at 3V and 25°C), 128kHz (low power)
  - External crystal oscillator
    - Highly precise, requires external quartz
- Clock source distributed to modules
  - CLK_{CPU}, $CLK_{I/O}$, CLK_{flash}, CLK_{ADC}
  - CLK_{ADC} allows switching off other clocks during ADC conversion

AVR Development Toolchain & IDEs

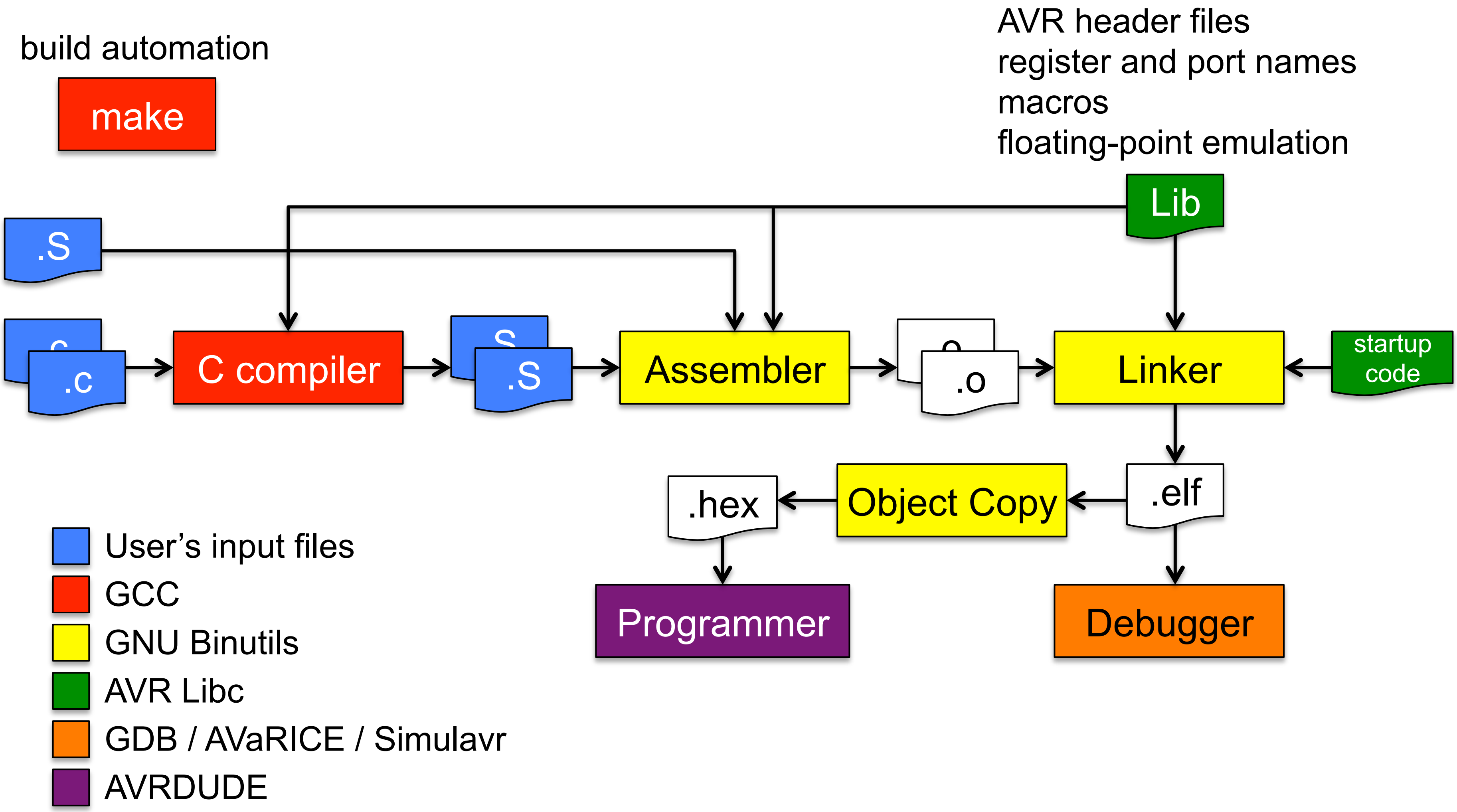
- Free AVR toolchain
  - GNU C compiler: avr-gcc (gcc.gnu.org)
  - C library: avr-libc
  - Down-/Uploader: avrdude (www.nongnu.org/avr-libc/)
- CrossPack for Mac OS X
  - avr-gcc on Mac OS X, Xcode can be used (but not required)
  - http://www.obdev.at/products/crosspack/index.html
  - oder: “sudo port install avr-gcc” (mit MacPorts)
- WinAVR for Windows
  - IDE for avr-gcc on Windows
  - http://winavr.sourceforge.net
- Atmel AVR Studio
  - http://www.atmel.com

AVR Development Toolchain & IDEs

- Eclipse and avr-eclipse
  - http://avr-eclipse.sourceforge.net



AVR-GCC Toolchain Overview



Source: http://www.avrfreaks.net/wiki/index.php/Documentation:AVR_GCC/AVR_GCC_Tool_Collection

AVR Libc

- Free Software toolchain for Atmel AVR microcontrollers
  - avr-binutils
  - avr-gcc
  - avr-libc

- AVR Libc
  - C library for use with GCC on Atmel AVR microcontrollers

- AVR Libc Home Page
  - http://www.nongnu.org/avr-libc/

CrossPack: Creating a Project (Mac OS X)

bash$ avr-gcc-select 3

Current default compiler: gcc 3

bash$ avr-project BlinkLED

Using template: /usr/local/CrossPack-AVR-20100115/etc/templates/TemplateProject

bash$ cd BlinkLED/

bash$ ls -l

total 0

drwxr-xr-x 4 michaelrohs staff 136 Apr 2 22:44 BlinkLED.xcodeproj

drwxr-xr-x 4 michaelrohs staff 136 Apr 2 22:44 firmware

bash$ cd firmware/

bash$ ls -l

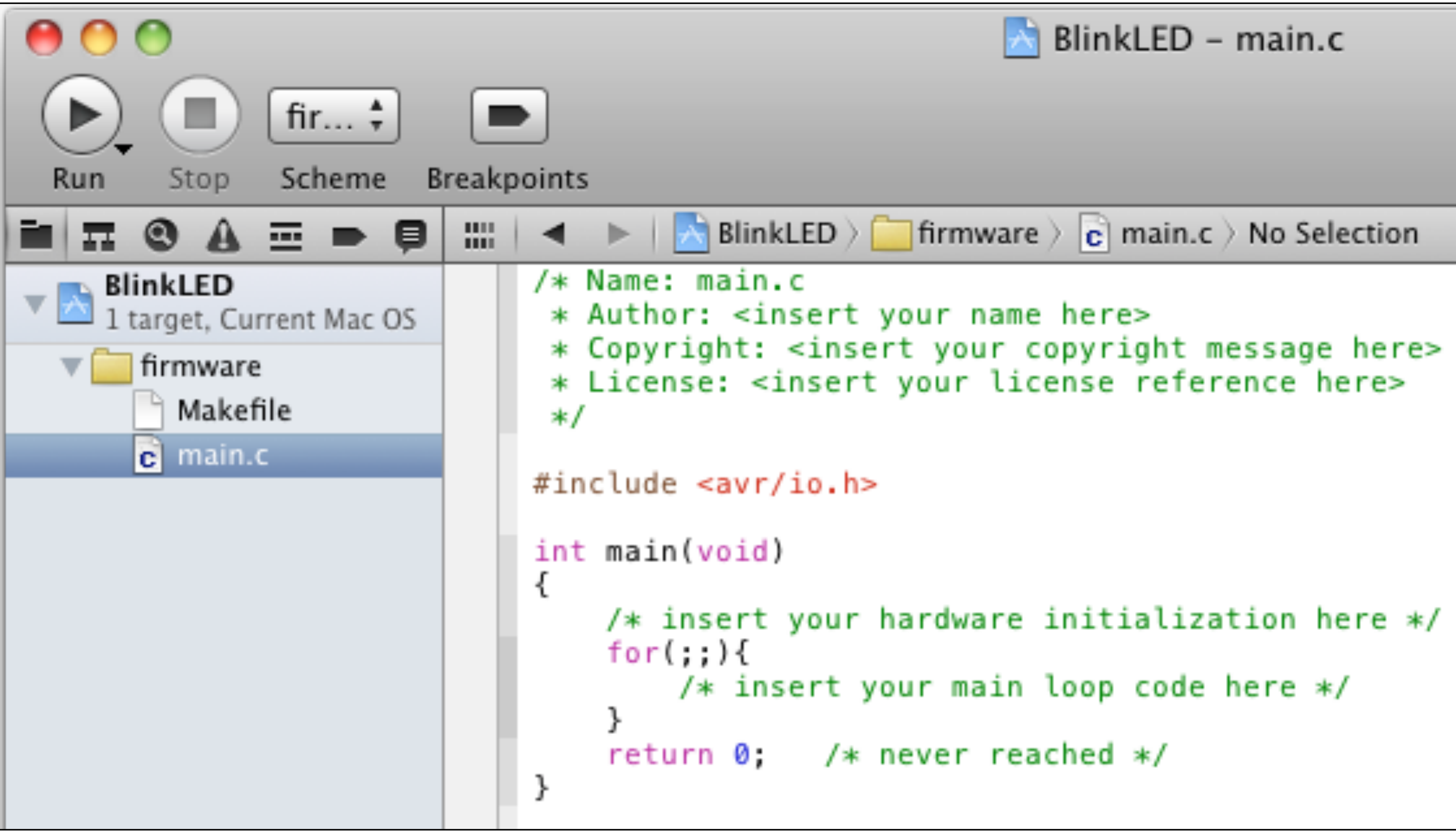
total 24

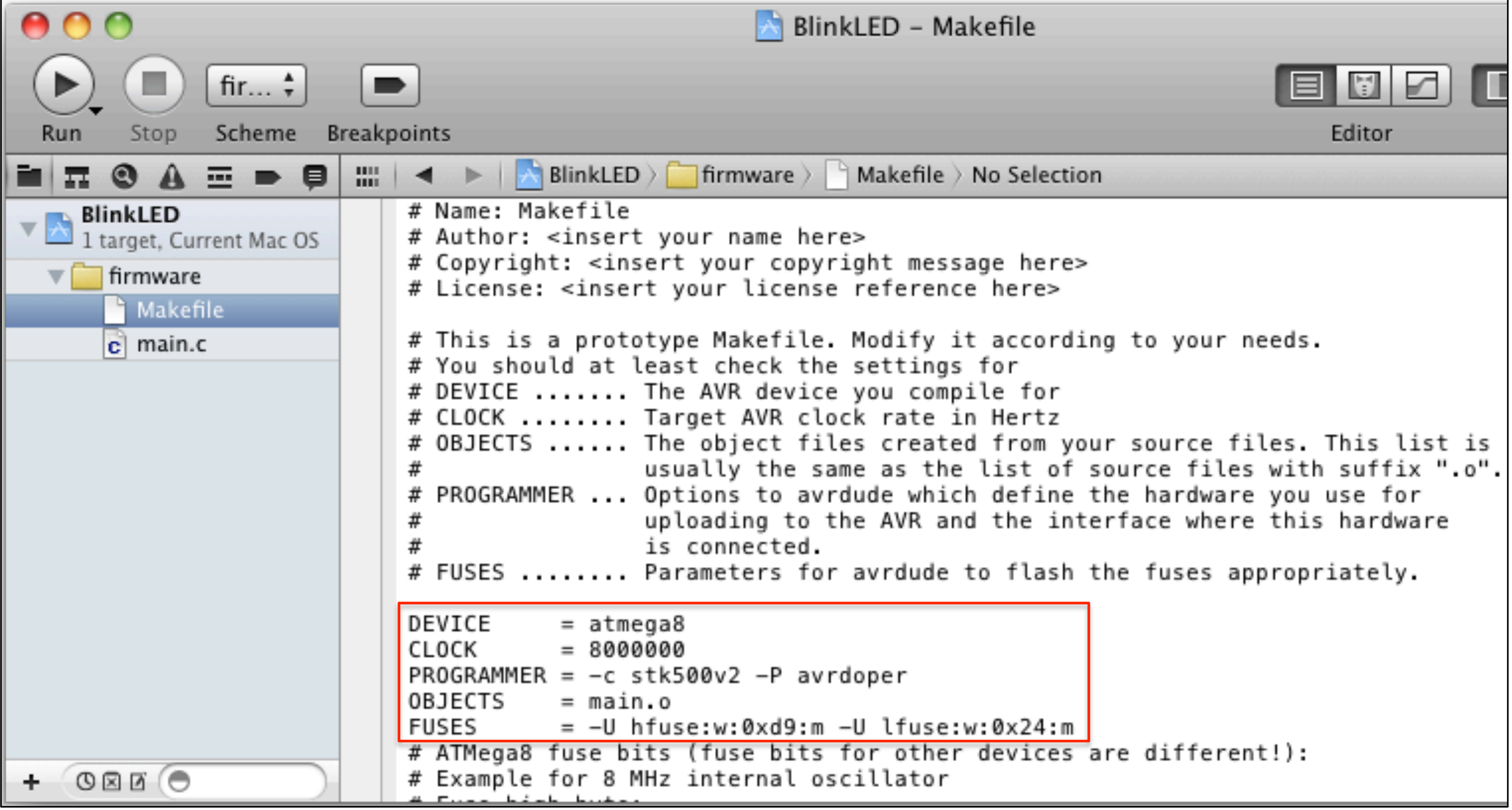
-rw-r--r-- 1 michaelrohs staff 4139 Apr 2 22:44 Makefile

-rw-r--r-- 1 michaelrohs staff   348 Apr 2 22:44 main.c

double-click to open Xcode project

Generated Project in XCode





- Adapt Makefile as required
  - DEVICE, CLOCK, FUSES
  - PROGRAMMER
  - OBJECTS

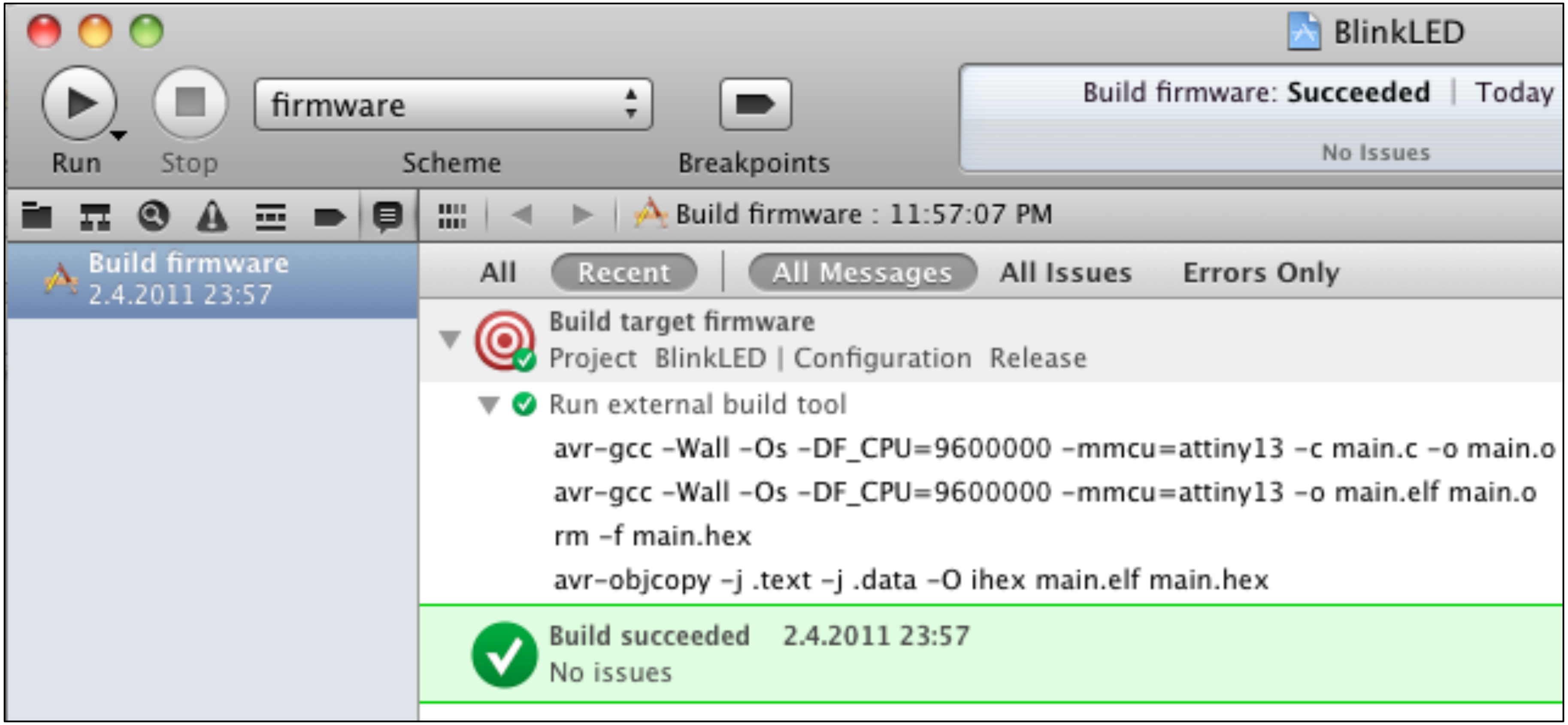
```
DEVICE     = attiny13
CLOCK      = 9600000
PROGRAMMER = -c USBasp
OBJECTS    = main.o
FUSES      = -U lfuse:w:0x7a:m -U hfuse:w:0xff:m
```

"Fuses Calculator"

- http://www.engbedded.com/fusecalc/
  - configure fuses
  - click "Apply feature settings"
  - use "AVRDUDE arguments" (bottom of page)
- For example (for AVRtiny45):

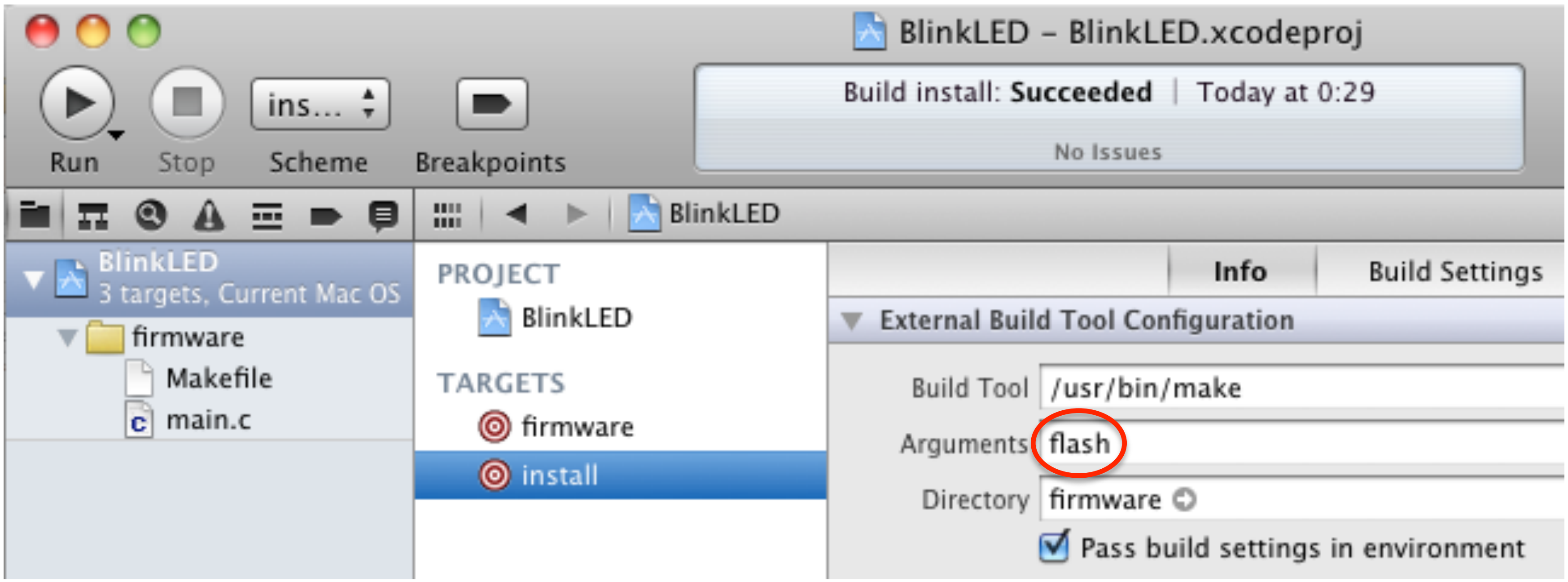
  -U lfuse:w:0xe2:m -U hfuse:w:0xdf:m -U efuse:w:0xff:m
- May want to use datasheet to verify settings

Building within XCode

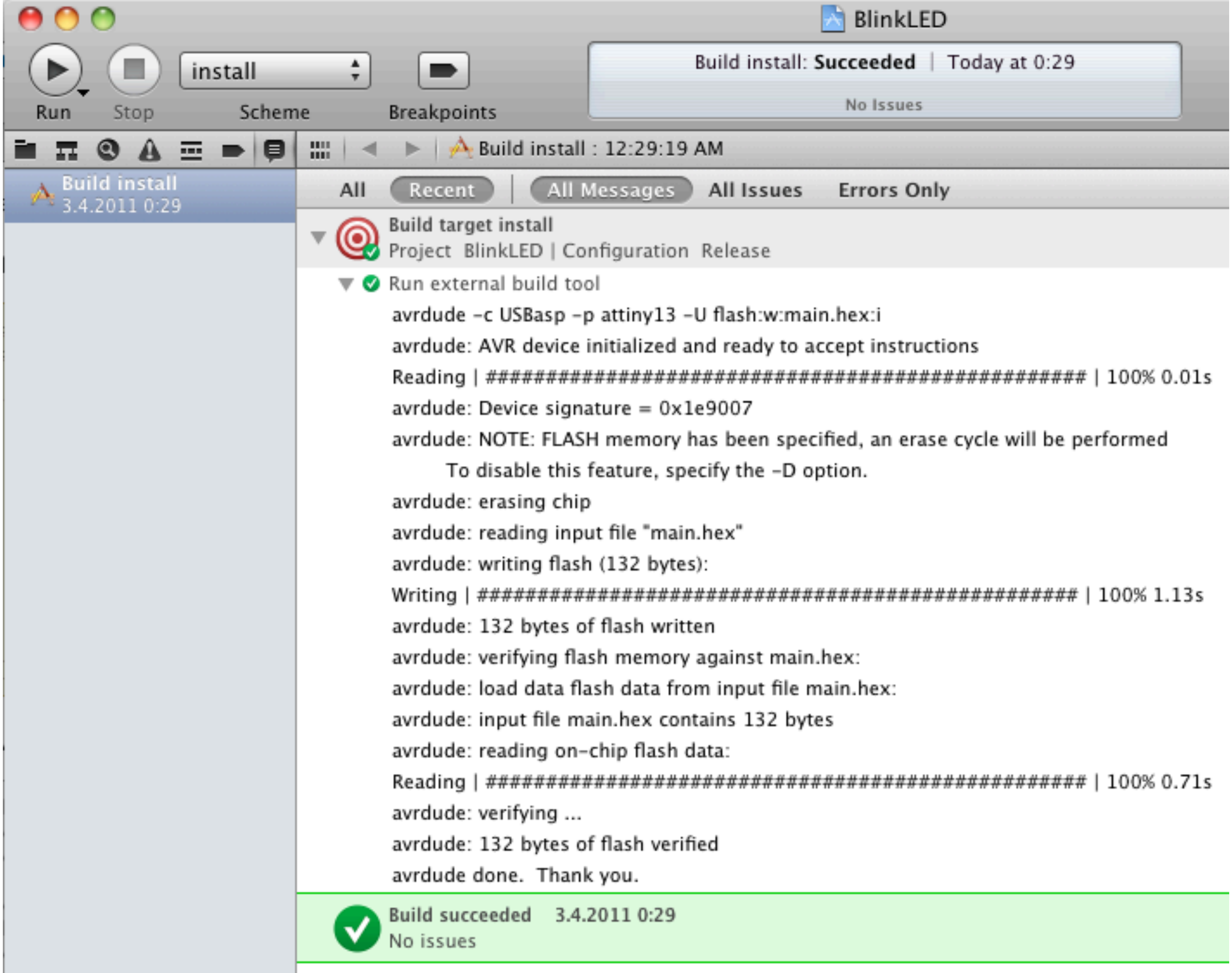


Flashing AVR from within XCode

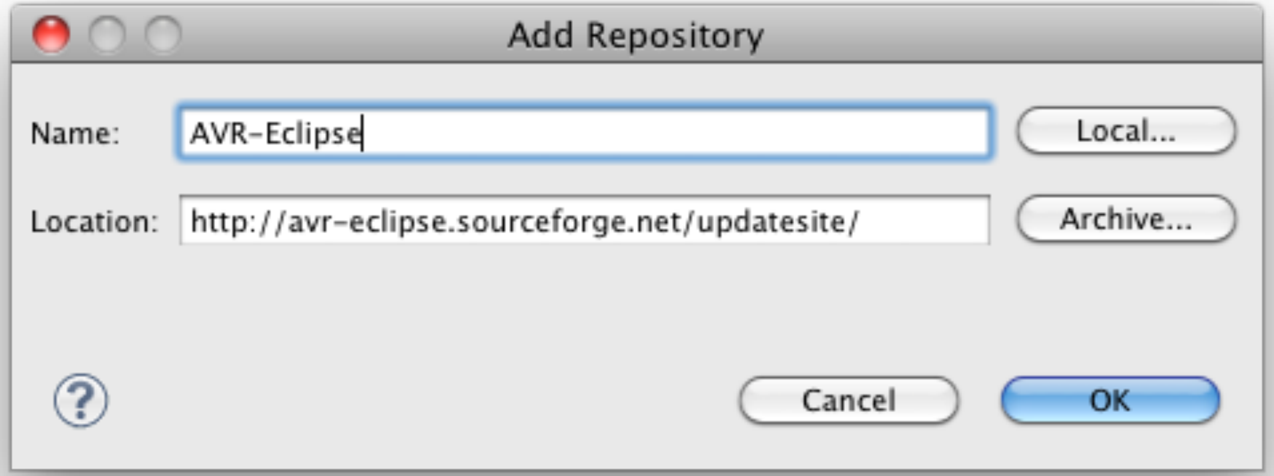
- Duplicate existing “firmware” target
- Rename to “install”
- Change Info | Arguments to “flash”

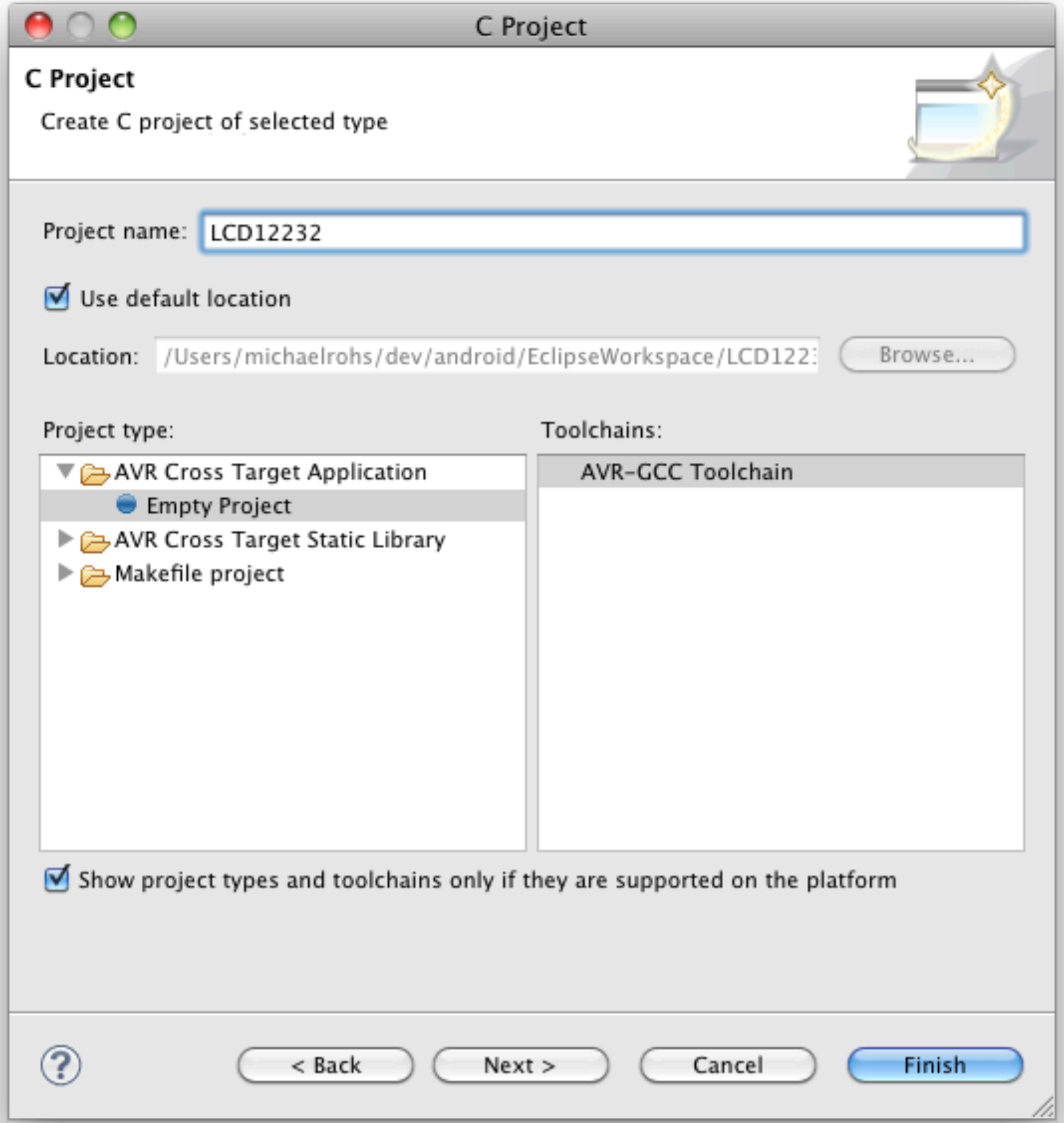


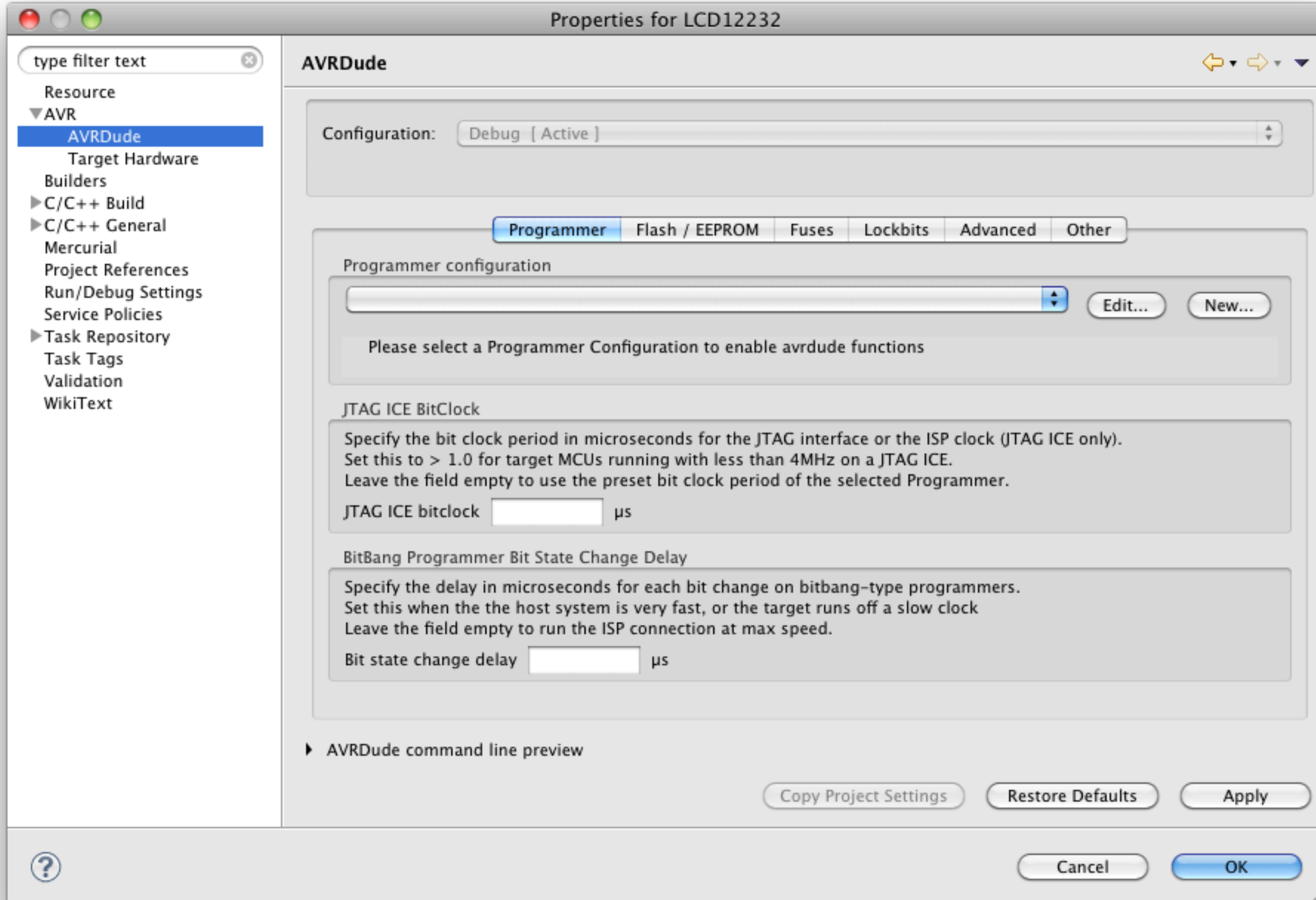
→ store custom template in
~/.CrossPack-AVR/templates/TemplateProject


BlinkLED
Run
Stop
install
Scheme
Breakpoints
Build install: Succeeded | Today at 0:29
No Issues
Build install : 12:29:19 AM
Build install
3.4.2011 0:29
All
Recent
All Messages
All Issues
Errors Only
Build target install
Project BlinkLED | Configuration Release
Run external build tool
avrdude -c USBasp -p attiny13 -U flash:w:main.hex:i
avrdude: AVR device initialized and ready to accept instructions
Reading | ## | 100% 0.01s
avrdude: Device signature = 0x1e9007
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: writing flash (132 bytes):
Writing | ## | 100% 1.13s
avrdude: 132 bytes of flash written
avrdude: verifying flash memory against main.hex:
avrdude: load data flash data from input file main.hex:
avrdude: input file main.hex contains 132 bytes
avrdude: reading on-chip flash data:
Reading | ## | 100% 0.71s
avrdude: verifying ...
avrdude: 132 bytes of flash verified
avrdude done. Thank you.
Build succeeded 3.4.2011 0:29
No issues

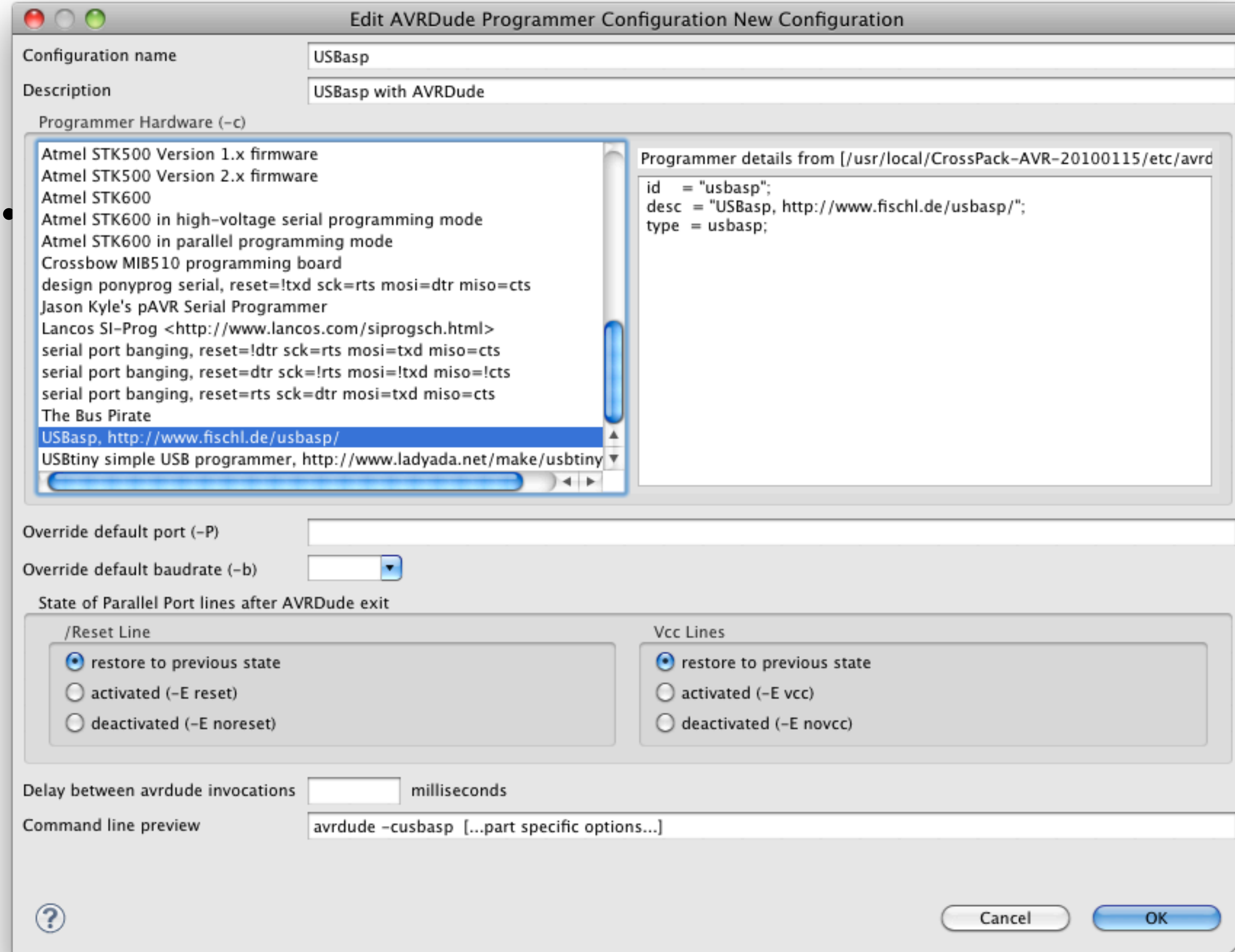
AVR-Eclipse


Add Repository
Name:
AVR-Eclipse
Local...
Location:
http://avr-eclipse.sourceforge.net/updatesite/
Archive...
Cancel
OK

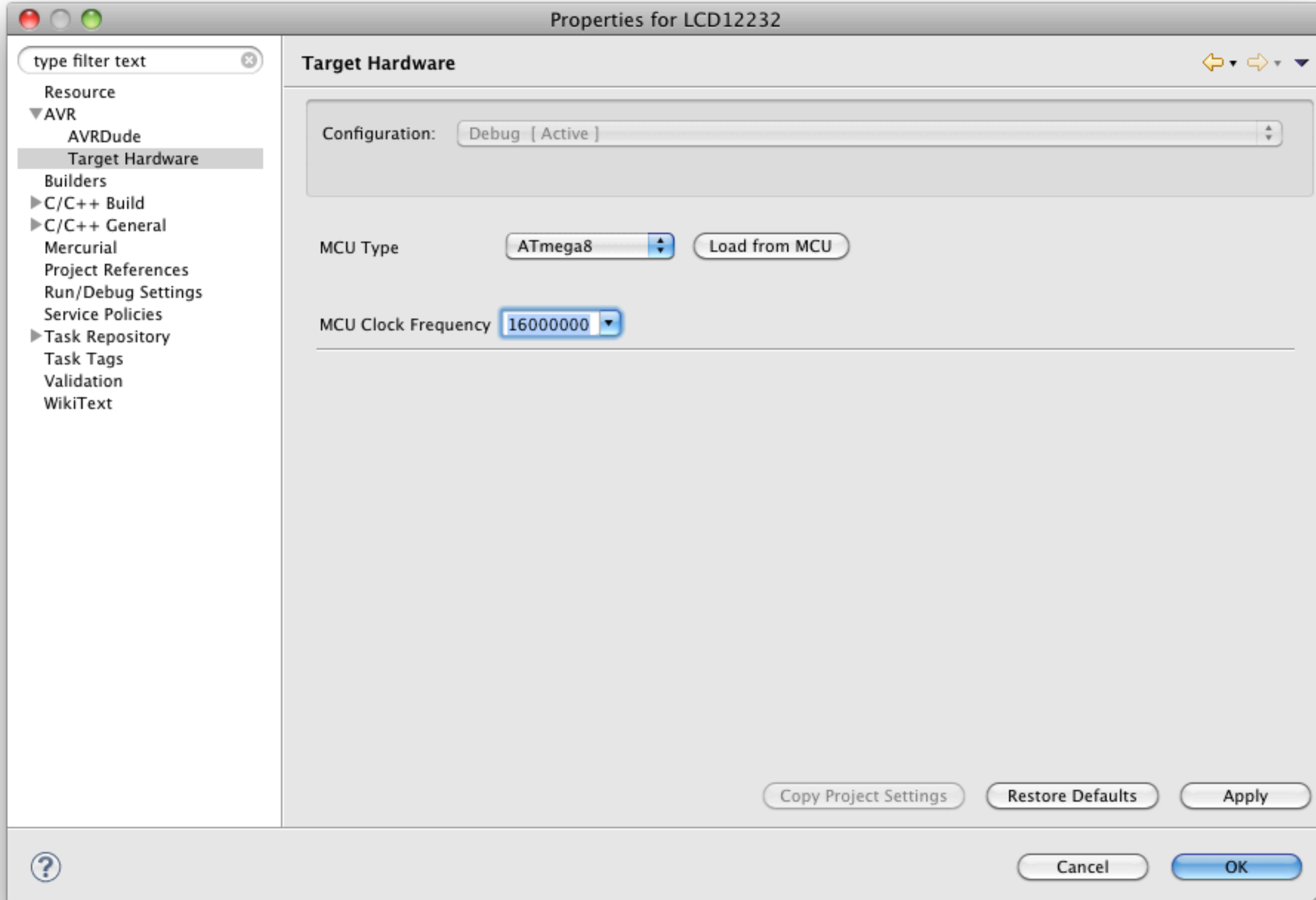

C Project
C Project
Create C project of selected type
Project name:
LCD12232
Use default location
Location:
/Users/michaelrohs/dev/android/EclipseWorkspace/LCD122
Browse...
Project type:
Toolchains:
AVR Cross Target Application
Empty Project
AVR Cross Target Static Library
Makefile project
AVR-GCC Toolchain
Show project types and toolchains only if they are supported on the platform
< Back
Next >
Cancel
Finish

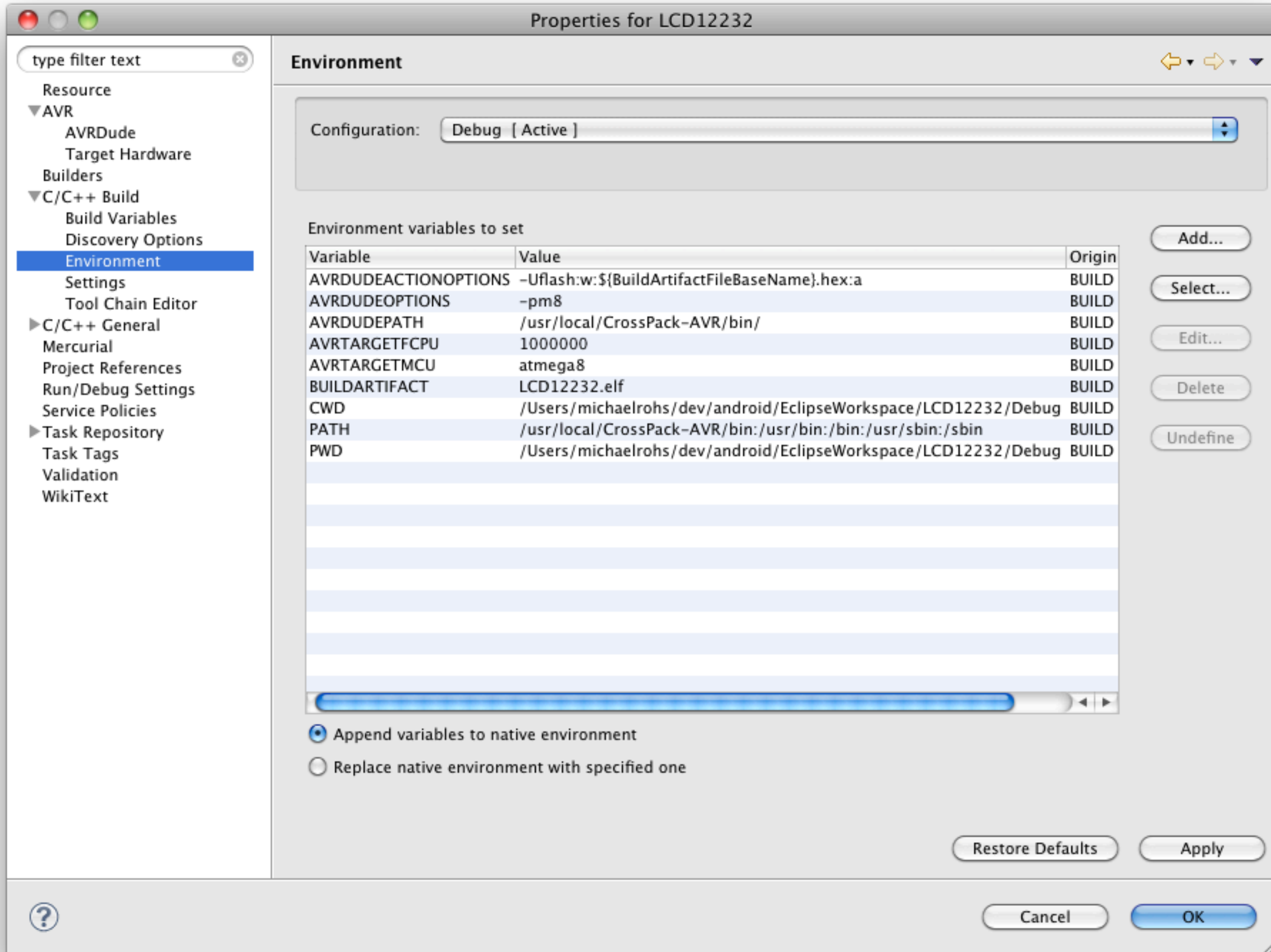


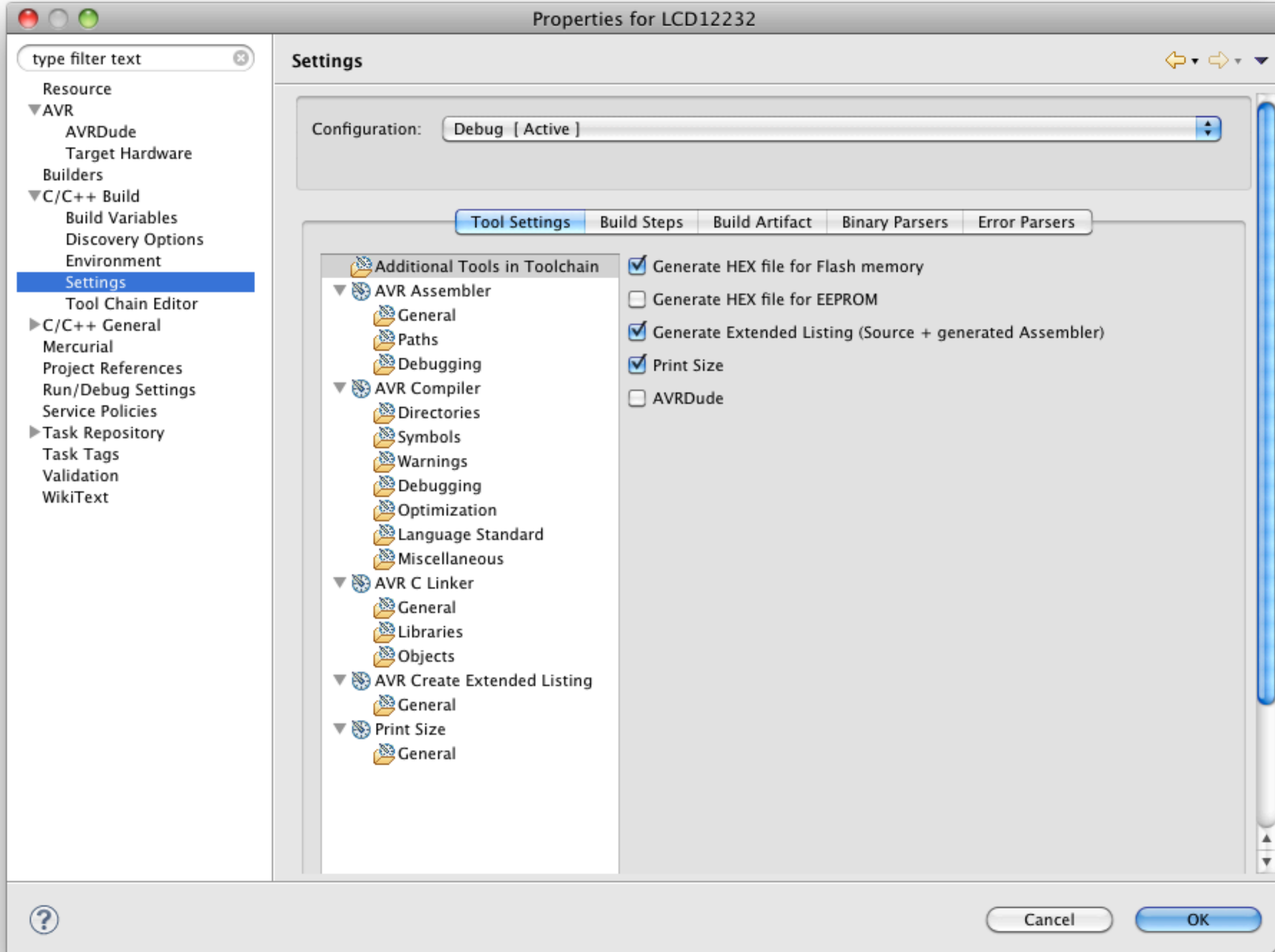
Properties for LCD12232
type filter text
Resource
AVR
AVRDude
Target Hardware
Builders
C/C++ Build
C/C++ General
Mercurial
Project References
Run/Debug Settings
Service Policies
Task Repository
Task Tags
Validation
WikiText
AVRDude
Configuration:
Debug [Active]
Programmer
Flash / EEPROM
Fuses
Lockbits
Advanced
Other
Programmer configuration
Edit...
New...
Please select a Programmer Configuration to enable avrdude functions
JTAG ICE BitClock
Specify the bit clock period in microseconds for the JTAG interface or the ISP clock (JTAG ICE only).
Set this to > 1.0 for target MCUs running with less than 4MHz on a JTAG ICE.
Leave the field empty to use the preset bit clock period of the selected Programmer.
JTAG ICE bitclock
µs
BitBang Programmer Bit State Change Delay
Specify the delay in microseconds for each bit change on bitbang-type programmers.
Set this when the the host system is very fast, or the target runs off a slow clock
Leave the field empty to run the ISP connection at max speed.
Bit state change delay
µs
AVRDude command line preview
Copy Project Settings
Restore Defaults
Apply
Cancel
OK

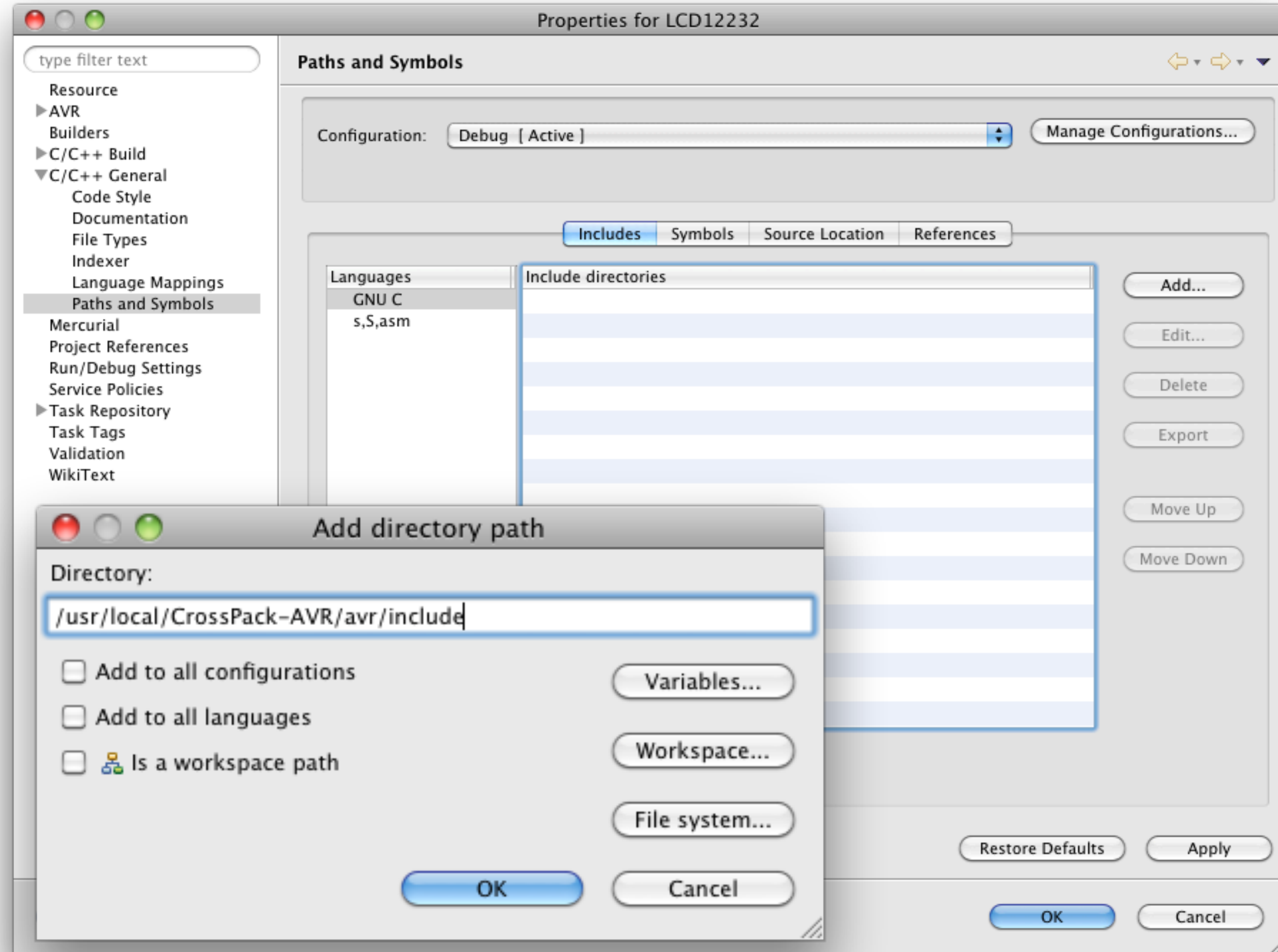


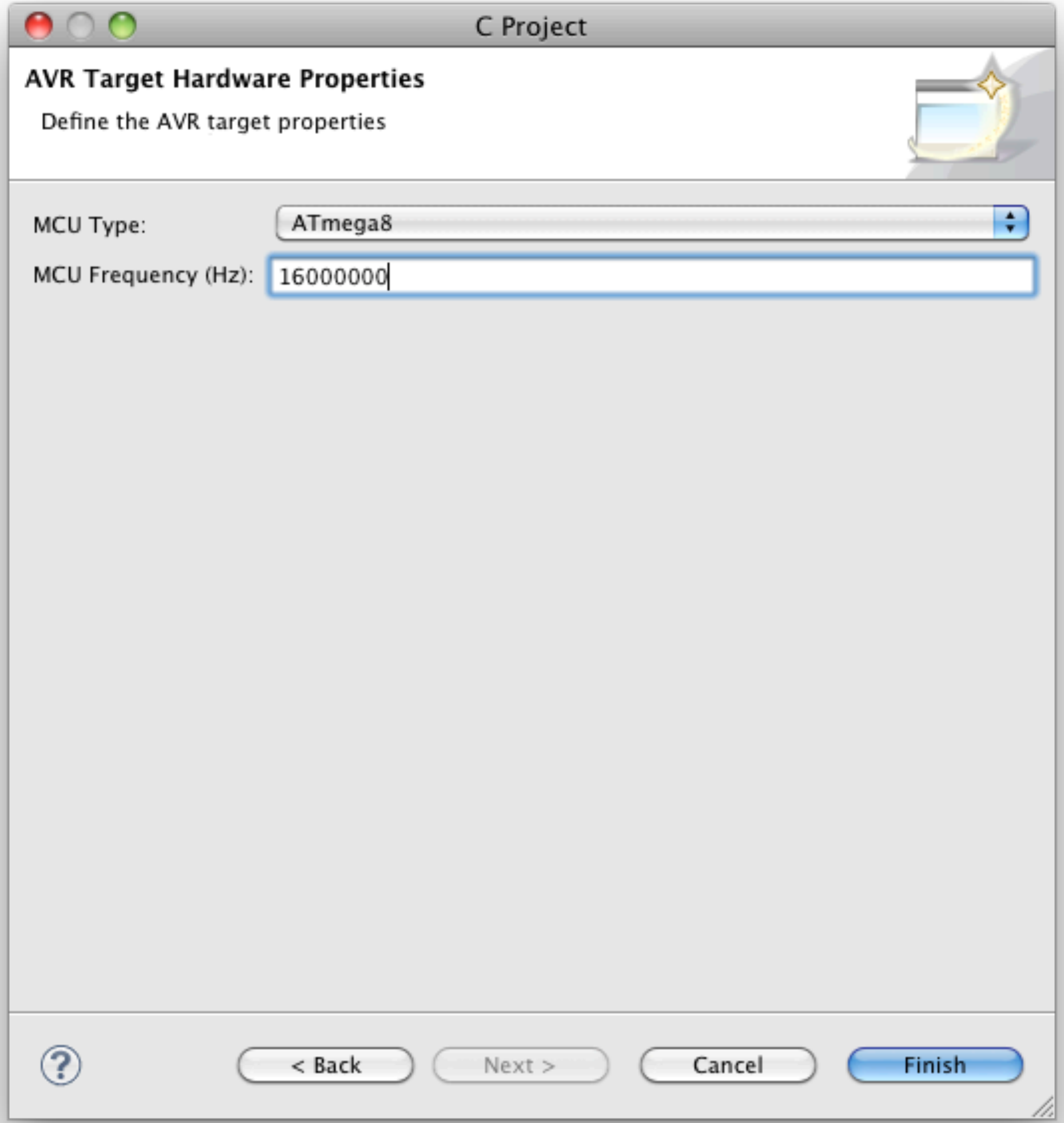
Edit AVRDude Programmer Configuration New Configuration
Configuration name
USBasp
Description
USBasp with AVRDude
Programmer Hardware (-c)
Atmel STK500 Version 1.x firmware
Atmel STK500 Version 2.x firmware
Atmel STK600
Atmel STK600 in high-voltage serial programming mode
Atmel STK600 in parallel programming mode
Crossbow MIB510 programming board
design ponyprog serial, reset=!txd sck=rts mosi=dtr miso=cts
Jason Kyle's pAVR Serial Programmer
Lancos SI-Prog <http://www.lancos.com/siprogsch.html>
serial port banging, reset=!dtr sck=rts mosi=txd miso=cts
serial port banging, reset=dtr sck=!rts mosi=!txd miso=!cts
serial port banging, reset=rts sck=dtr mosi=txd miso=cts
The Bus Pirate
USBasp, http://www.fischl.de/usbasp/
USBtiny simple USB programmer, http://www.ladyada.net/make/usbtiny
Programmer details from [/usr/local/CrossPack-AVR-20100115/etc/avrd
id = "usbasp";
desc = "USBasp, http://www.fischl.de/usbasp/";
type = usbasp;
Override default port (-P)
Override default baudrate (-b)
State of Parallel Port lines after AVRDude exit
/Reset Line
restore to previous state
activated (-E reset)
deactivated (-E noreset)
Vcc Lines
restore to previous state
activated (-E vcc)
deactivated (-E novcc)
Delay between avrdude invocations
milliseconds
Command line preview
avrdude -cusbasp [...part specific options...]
Cancel
OK


Properties for LCD12232
type filter text
Resource
AVR
AVRDude
Target Hardware
Builders
C/C++ Build
C/C++ General
Mercurial
Project References
Run/Debug Settings
Service Policies
Task Repository
Task Tags
Validation
WikiText
Target Hardware
Configuration:
Debug [Active]
MCU Type
ATmega8
Load from MCU
MCU Clock Frequency
16000000
Copy Project Settings
Restore Defaults
Apply
Cancel
OK

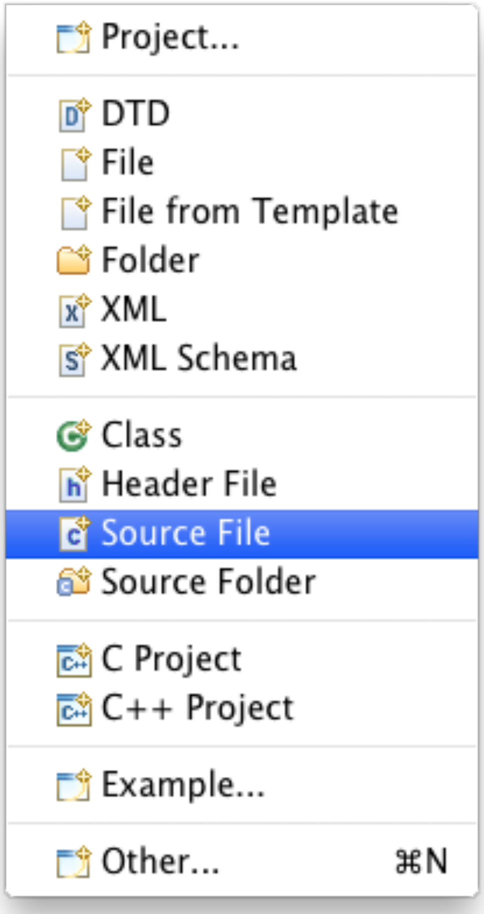

Properties for LCD12232
type filter text
Resource
AVR
AVRDude
Target Hardware
Builders
C/C++ Build
Build Variables
Discovery Options
Environment
Settings
Tool Chain Editor
C/C++ General
Mercurial
Project References
Run/Debug Settings
Service Policies
Task Repository
Task Tags
Validation
WikiText
Environment
Configuration: Debug [Active]
Environment variables to set
Variable Value Origin
AVRDUDEACTIONOPTIONS -Uflash:w:${BuildArtifactFileBaseName}.hex:a BUILD
AVRDUDEOPTIONS -pm8 BUILD
AVRDUDEPATH /usr/local/CrossPack-AVR/bin/ BUILD
AVRTARGETFCPU 1000000 BUILD
AVRTARGETMCU atmega8 BUILD
BUILDARTIFACT LCD12232.elf BUILD
CWD /Users/michaelrohs/dev/android/EclipseWorkspace/LCD12232/Debug BUILD
PATH /usr/local/CrossPack-AVR/bin:/usr/bin:/bin:/usr/sbin:/sbin BUILD
PWD /Users/michaelrohs/dev/android/EclipseWorkspace/LCD12232/Debug BUILD
Add...
Select...
Edit...
Delete
Undefine
Append variables to native environment
Replace native environment with specified one
Restore Defaults
Apply
Cancel
OK

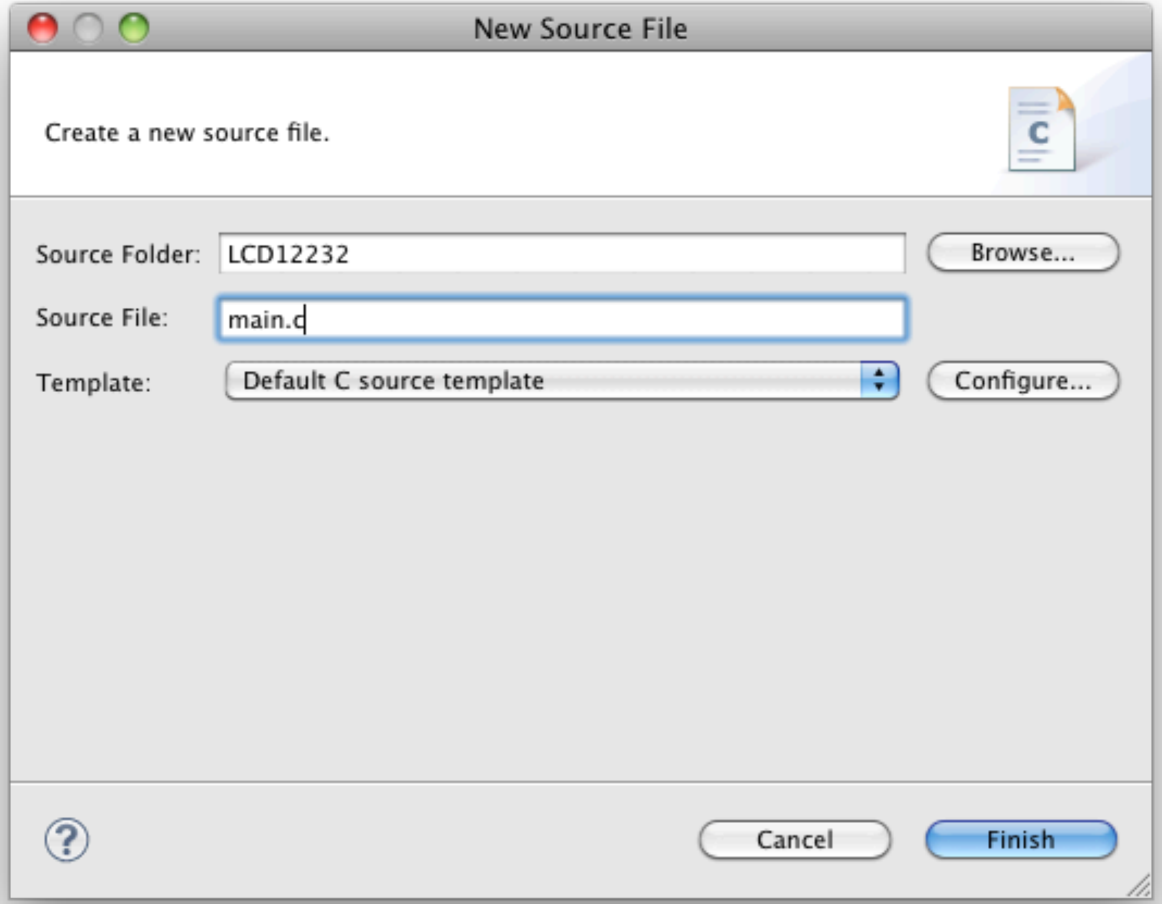

Properties for LCD12232
type filter text
Resource
AVR
AVRDude
Target Hardware
Builders
C/C++ Build
Build Variables
Discovery Options
Environment
Settings
Tool Chain Editor
C/C++ General
Mercurial
Project References
Run/Debug Settings
Service Policies
Task Repository
Task Tags
Validation
WikiText
Settings
Configuration:
Debug [Active]
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
Additional Tools in Toolchain
AVR Assembler
General
Paths
Debugging
AVR Compiler
Directories
Symbols
Warnings
Debugging
Optimization
Language Standard
Miscellaneous
AVR C Linker
General
Libraries
Objects
AVR Create Extended Listing
General
Print Size
General
Generate HEX file for Flash memory
Generate HEX file for EEPROM
Generate Extended Listing (Source + generated Assembler)
Print Size
AVRDude
Cancel
OK


Properties for LCD12232
type filter text
Resource
AVR
Builders
C/C++ Build
C/C++ General
Code Style
Documentation
File Types
Indexer
Language Mappings
Paths and Symbols
Mercurial
Project References
Run/Debug Settings
Service Policies
Task Repository
Task Tags
Validation
WikiText
Paths and Symbols
Configuration:
Debug [Active]
Manage Configurations...
Includes
Symbols
Source Location
References
Languages
GNU C
s,S,asm
Include directories
Add...
Edit...
Delete
Export
Move Up
Move Down
Restore Defaults
Apply
OK
Cancel
Add directory path
Directory:
/usr/local/CrossPack-AVR/avr/include
Add to all configurations
Add to all languages
Is a workspace path
Variables...
Workspace...
File system...
OK
Cancel


C Project
AVR Target Hardware Properties
Define the AVR target properties
MCU Type:
ATmega8
MCU Frequency (Hz):
16000000
< Back
Next >
Cancel
Finish

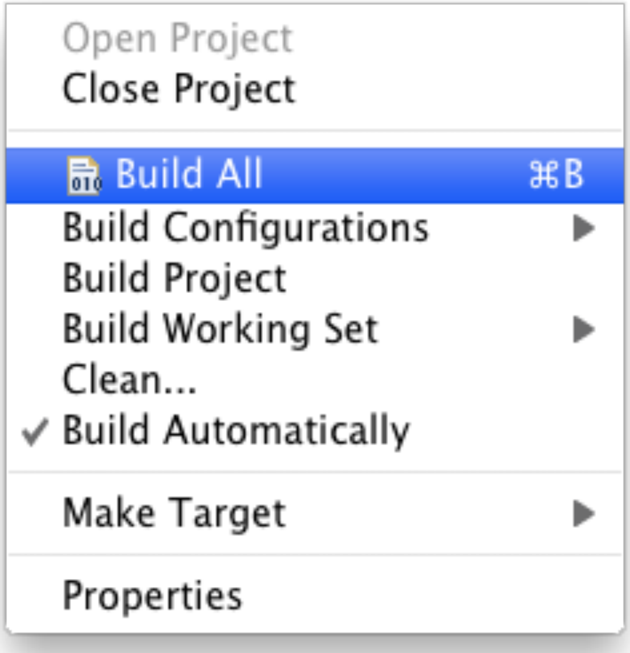
AVR-Eclipse

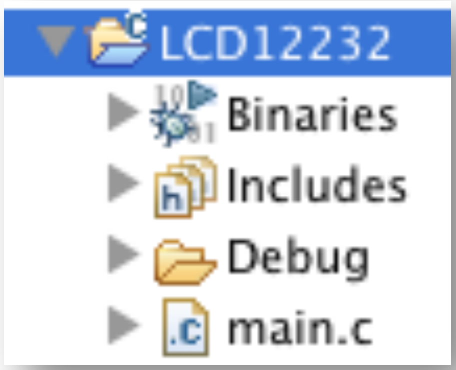

Project...
DTD
File
File from Template
Folder
XML
XML Schema
Class
Header File
Source File
Source Folder
C Project
C++ Project
Example...
Other... ⌘N

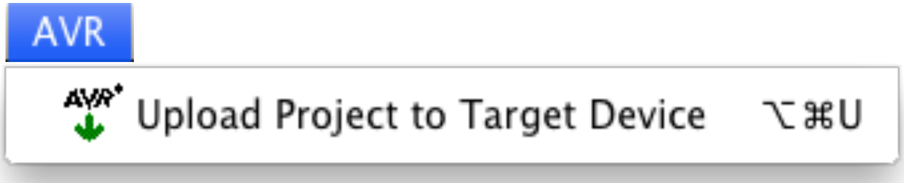

New Source File
Create a new source file.
Source Folder: LCD12232
Browse...
Source File: main.c
Template: Default C source template
Configure...
Cancel
Finish

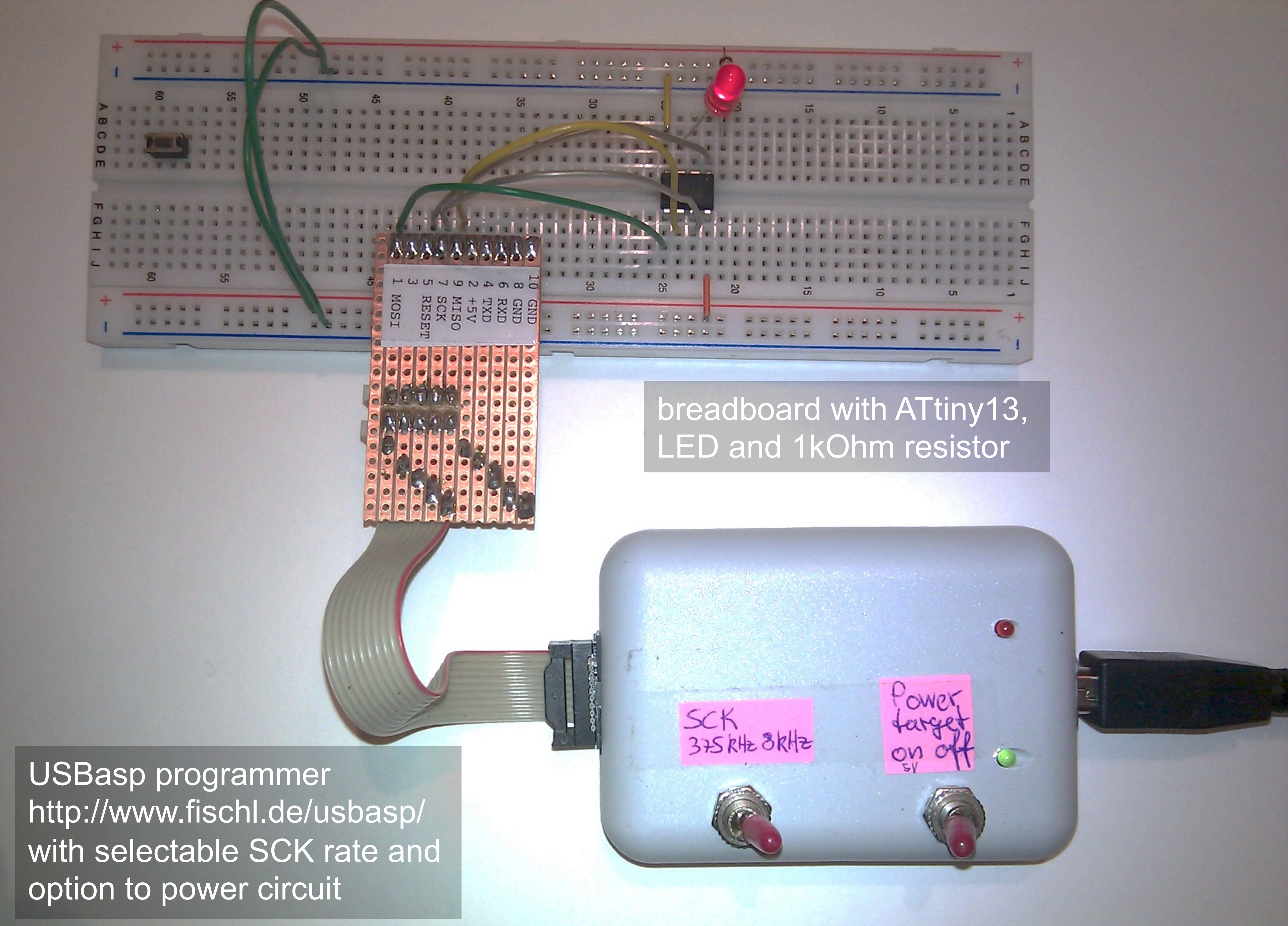
AVR-Eclipse

- Building and uploading the program







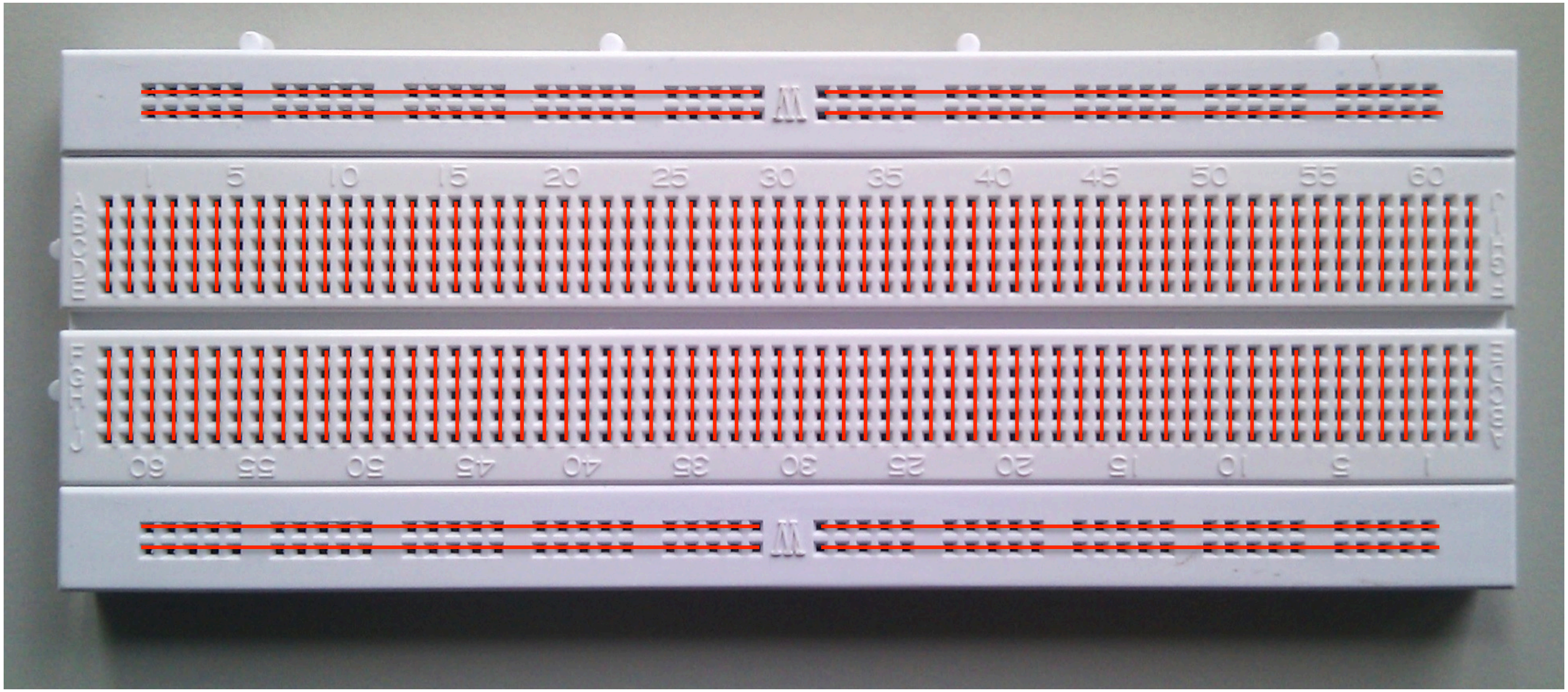


breadboard with ATtiny13,
LED and 1kOhm resistor

USBasp programmer
http://www.fischl.de/usbasp/
with selectable SCK rate and
option to power circuit

Breadboard

- Quick prototyping
  - Changing/adding components is easy
- Can get confusing soon (“spaghetti wires”)



Using the Command Line

bash$ ls

Makefile   main.c

bash$ make

avr-gcc -Wall -Os -DF_CPU=9600000 -mmcu=attiny13 -c main.c -o main.o

avr-gcc -Wall -Os -DF_CPU=9600000 -mmcu=attiny13 -o main.elf main.o

rm -f main.hex

avr-objcopy -j .text -j .data -O ihex main.elf main.hex

bash$ make flash

avrdude -c USBasp -p attiny13 -U flash:w:main.hex:i

with mySmartUSB:
avrdude -p attiny13 -c stk500v2
  -P /dev/cu.SLAB_USBtoUART
  -U flash:w:main.hex:i

avrdude: AVR device initialized and ready to accept instructions

…

avrdude: writing flash (132 bytes):

Writing | ## | 100% 1.13s

…

avrdude: 132 bytes of flash verified

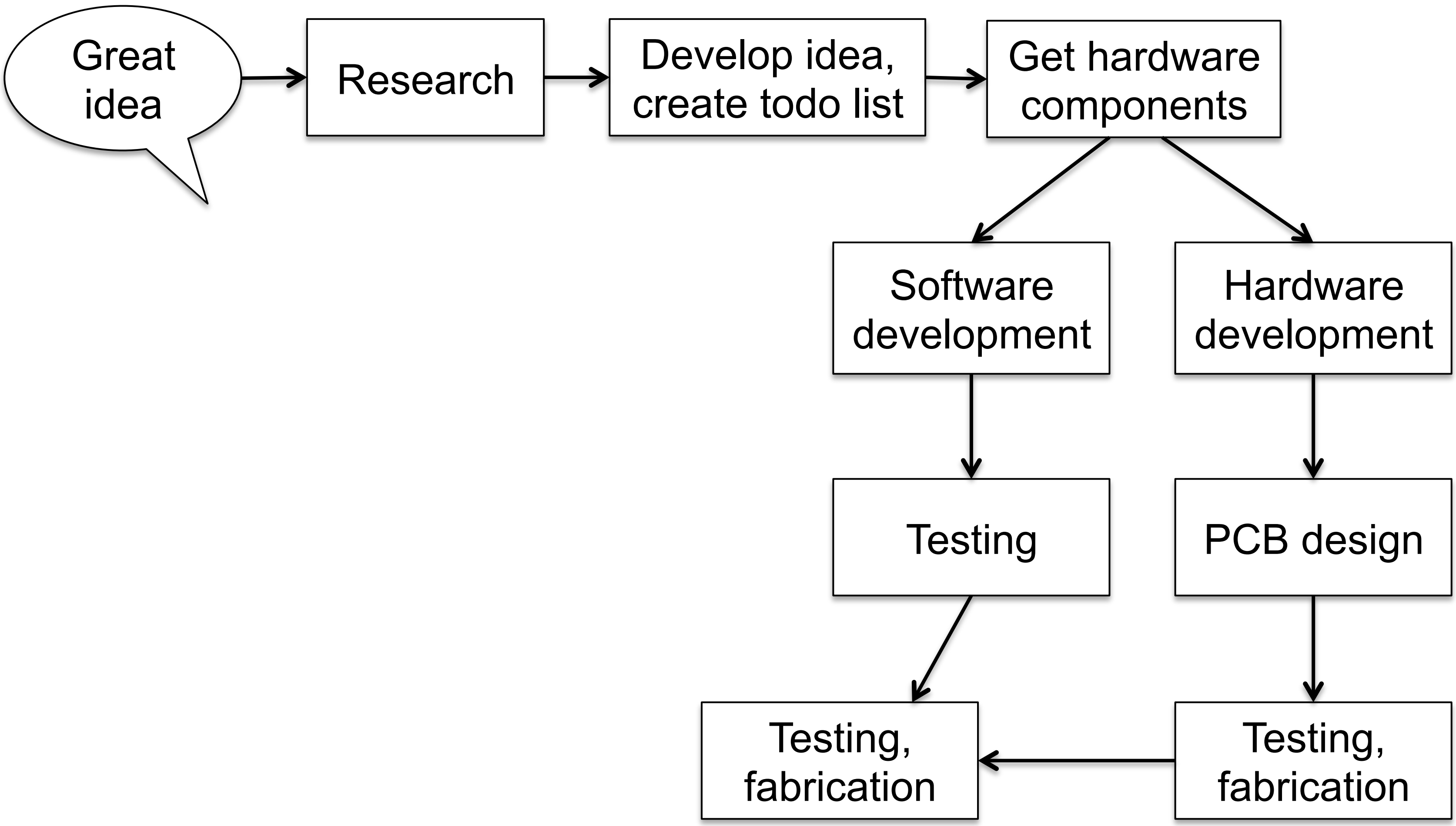
avrdude: safemode: Fuses OK

avrdude done. Thank you.

Assembly Language

- ATtiny have relatively simple instruction sets and are reasonably simple to program
  - ATtiny13: 120 instructions
- http://avra.sourceforge.net/index.html
- make
  - http://www.gnu.org/software/make/manual/make.html
  - http://www.makelinux.net/make3/make3-CHP-2-SECT-4.html
- V-USB
  - http://www.obdev.at/products/vusb/index.html

Development Process



Source: Gadre, Malhotra: tinyAVR projects

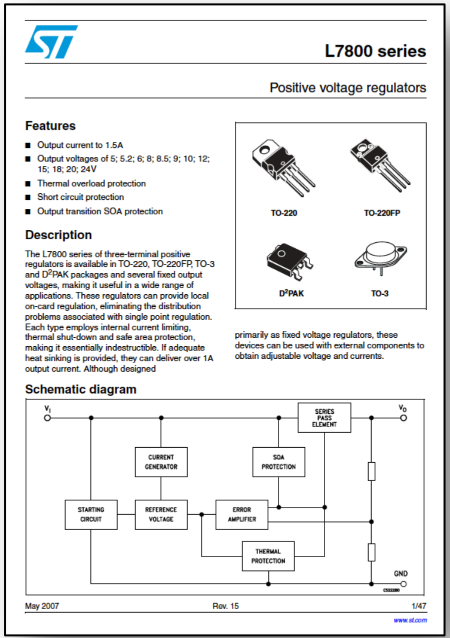
Reading Data Sheets

- Extremely important to read carefully
  - Easy to find online
- Example: 7805 +5V voltage regulator
  - Operate according to “electrical characteristics” ← “max. ratings”

4 Electrical characteristics

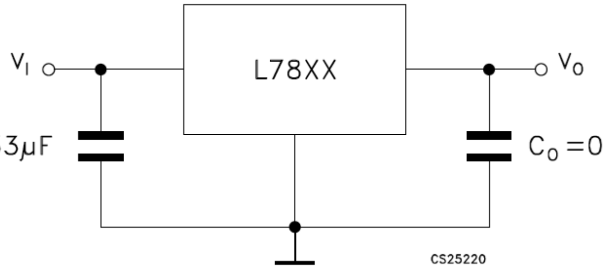
Table 3. Electrical characteristics of L7805 (refer to the test circuits, T_J = -55 to 150°C, V_I = 10V, I_O = 500 mA, C_I = 0.33 µF, C_O = 0.1 µF unless otherwise specified)

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_O	Output voltage	T_J = 25°C	4.8	5	5.2	V
V_O	Output voltage	I_O = 5mA to 1A, P_O ≤15W V_I = 8 to 20V	4.65	5	5.35	V



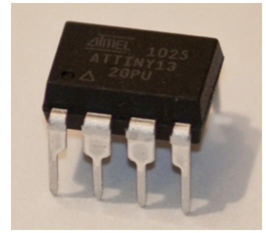
- “Application Circuits” show typical usage

V_I = 10..35V

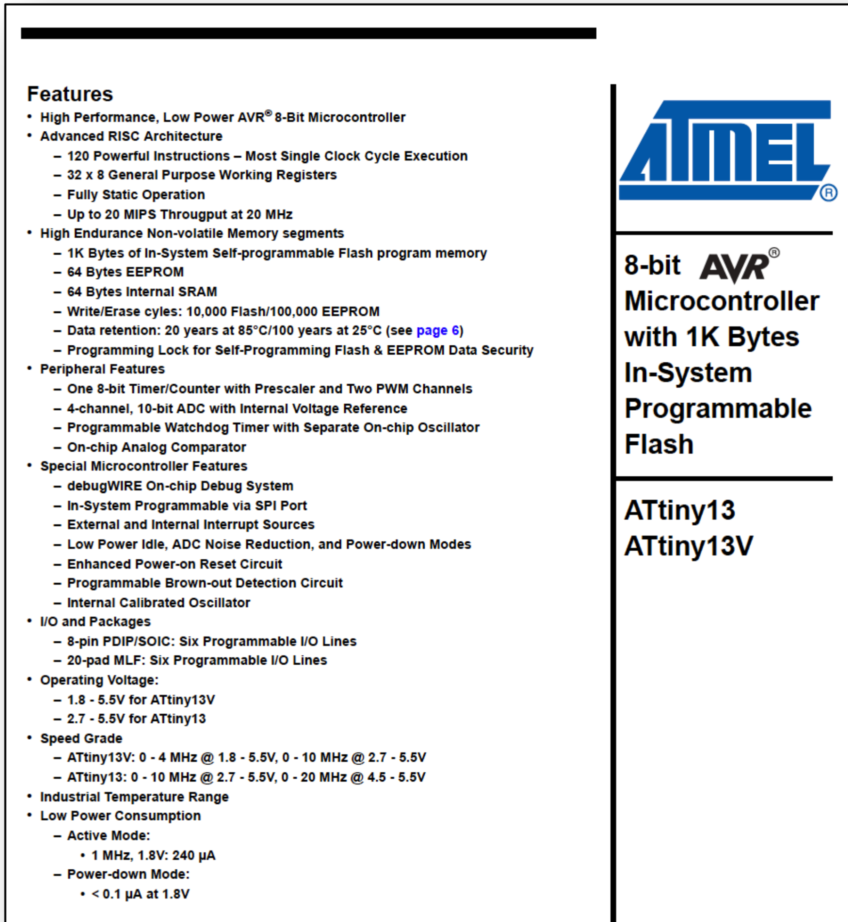


V_O = 5V (for 78**05**)

ATtiny13 Data Sheet



- 8 pins, 176 pages datasheet!



Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Througput at 20 MHz
- High Endurance Non-volatile Memory segments
  - 1K Bytes of In-System Self-programmable Flash program memory
  - 64 Bytes EEPROM
  - 64 Bytes Internal SRAM
  - Write/Erase cyles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C (see page 6)
  - Programming Lock for Self-Programming Flash & EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Prescaler and Two PWM Channels
  - 4-channel, 10-bit ADC with Internal Voltage Reference
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low Power Idle, ADC Noise Reduction, and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit
  - Internal Calibrated Oscillator
- I/O and Packages
  - 8-pin PDIP/SOIC: Six Programmable I/O Lines
  - 20-pad MLF: Six Programmable I/O Lines
- Operating Voltage:
  - 1.8 - 5.5V for ATtiny13V
  - 2.7 - 5.5V for ATtiny13
- Speed Grade
  - ATtiny13V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
  - ATtiny13: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Industrial Temperature Range
- Low Power Consumption
  - Active Mode:
    - 1 MHz, 1.8V: 240 µA
  - Power-down Mode:
    - < 0.1 µA at 1.8V

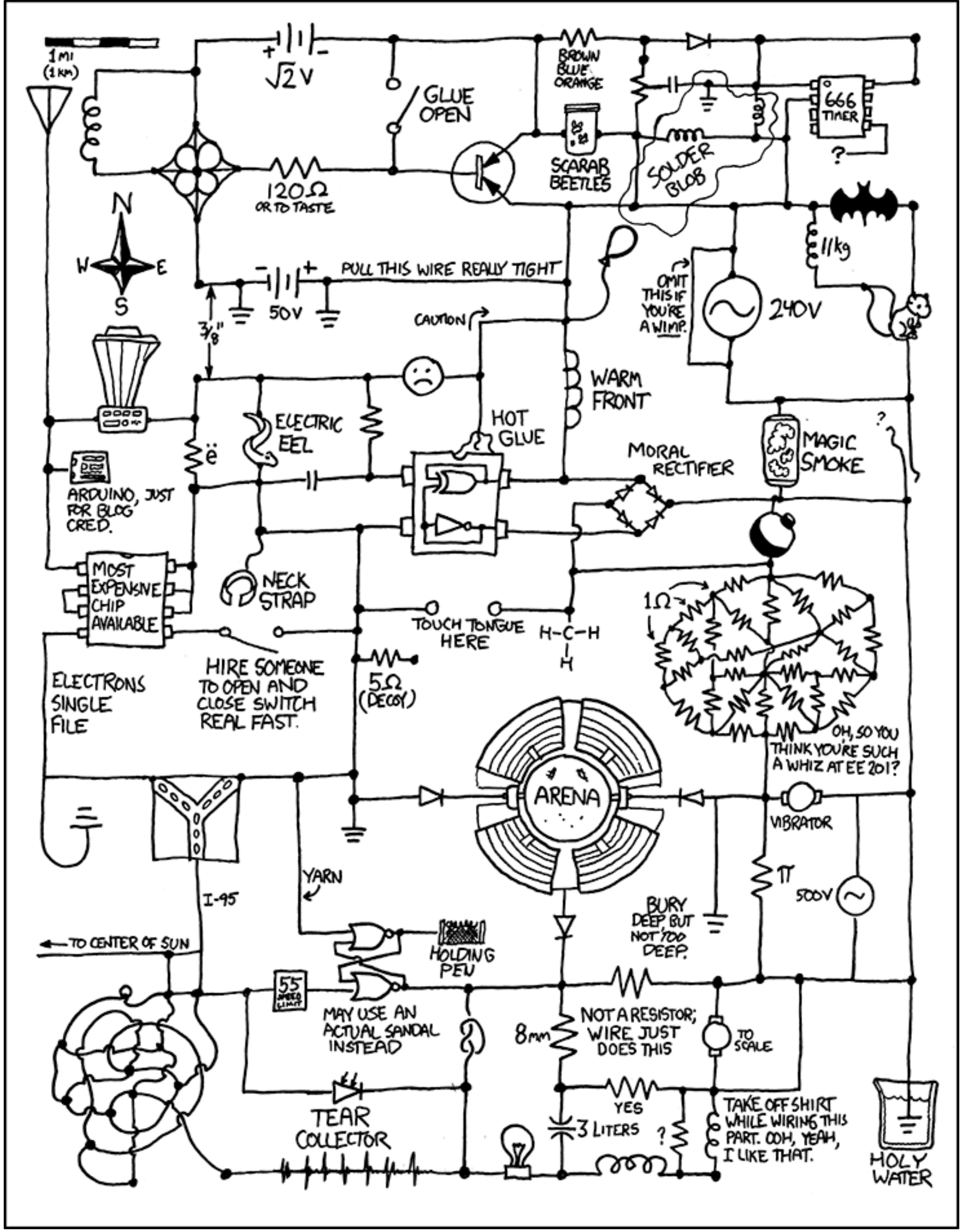
8-bit AVR® Microcontroller with 1K Bytes In-System Programmable Flash

ATtiny13
ATtiny13V

“Homework”

- Install AVR GCC
  - Use Eclipse or Xcode (as described above)

- Have a look into ATtiny13 datasheet
  - Especially first chapters on architecture



http://xkcd.com/730/

Creative Commons Attribution-NonCommercial 2.5 License.