ETH Course 402-0248-00L: Electronics for Physicists II (Digital)

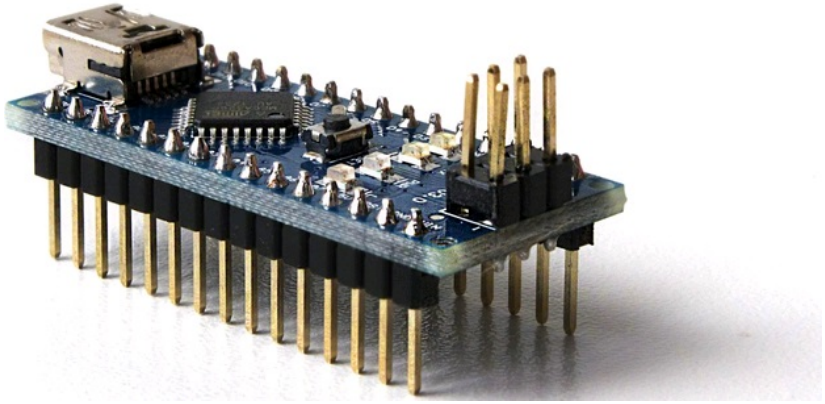
- **Taught by:** Tobi Delbruck, Alejandro Linares
- **Department:** ETH Zurich Department of Physics (D-PHYS)
- **Day/Time:** Weekly on Fridays from 1315 - 1700 in Picaardsaal, ETH Hoenggerberg Campus, HPT Room C103
- **Breaks:** No class in some weeks, check schedule on wiki.
- **Language:** English.
- **Credits:** 4 credit points.
- **Exam**: There is no exam but students must successfully complete the class exercises. Attendance sheet will be used.
- **Class wiki: google** "dig delbruck"

www.ini.uzh.ch/~tobi/wiki/doku.php?id=dig:start

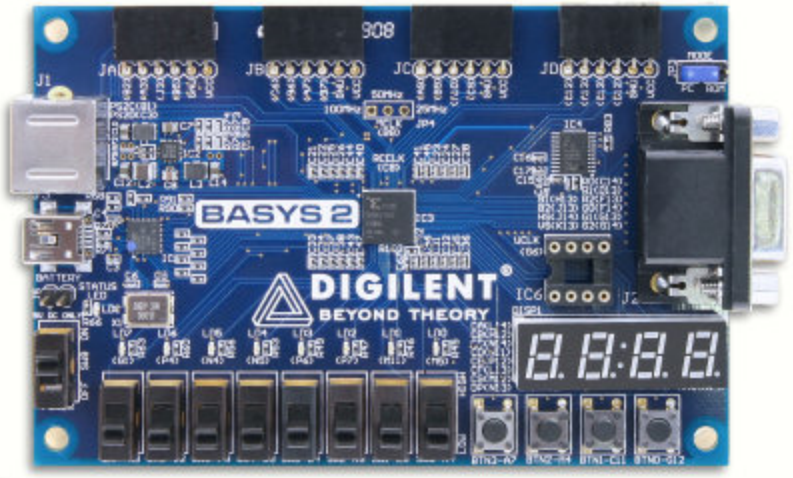
Prerequisites

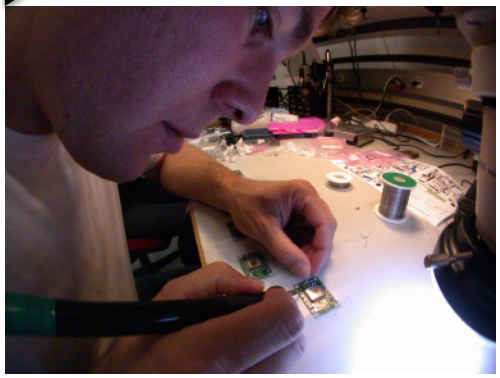
- **Companion course:**
  Electronics for Physicists I (Analog), Fall semester, taught by Roland Horisberger.
- The digital course complements the analog course by teaching how to build systems that convert and process analog information.
- You should have had some programming experience, preferably with C. Students (or at least each group of 2-3 students) need a laptop computer, Windows or Linux (but only Windows supported by Tobi). Mac OS can use VM.

1st half: Embedded systems with microcontrollers

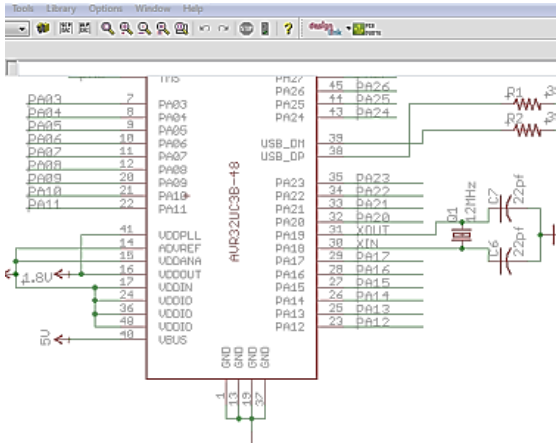


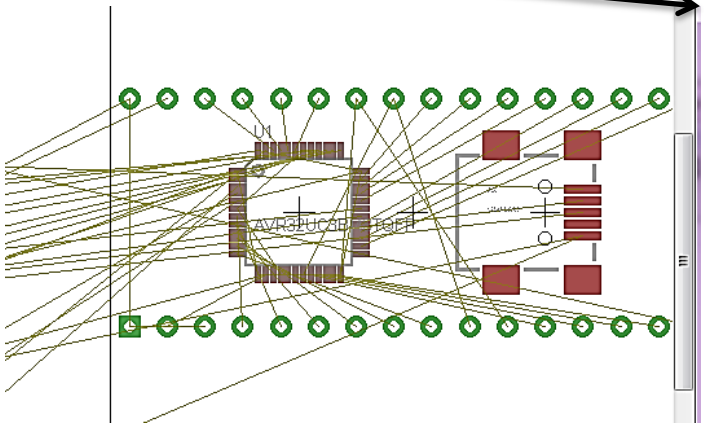
2nd half: Logic design with FPGA





PCB SMD assembly



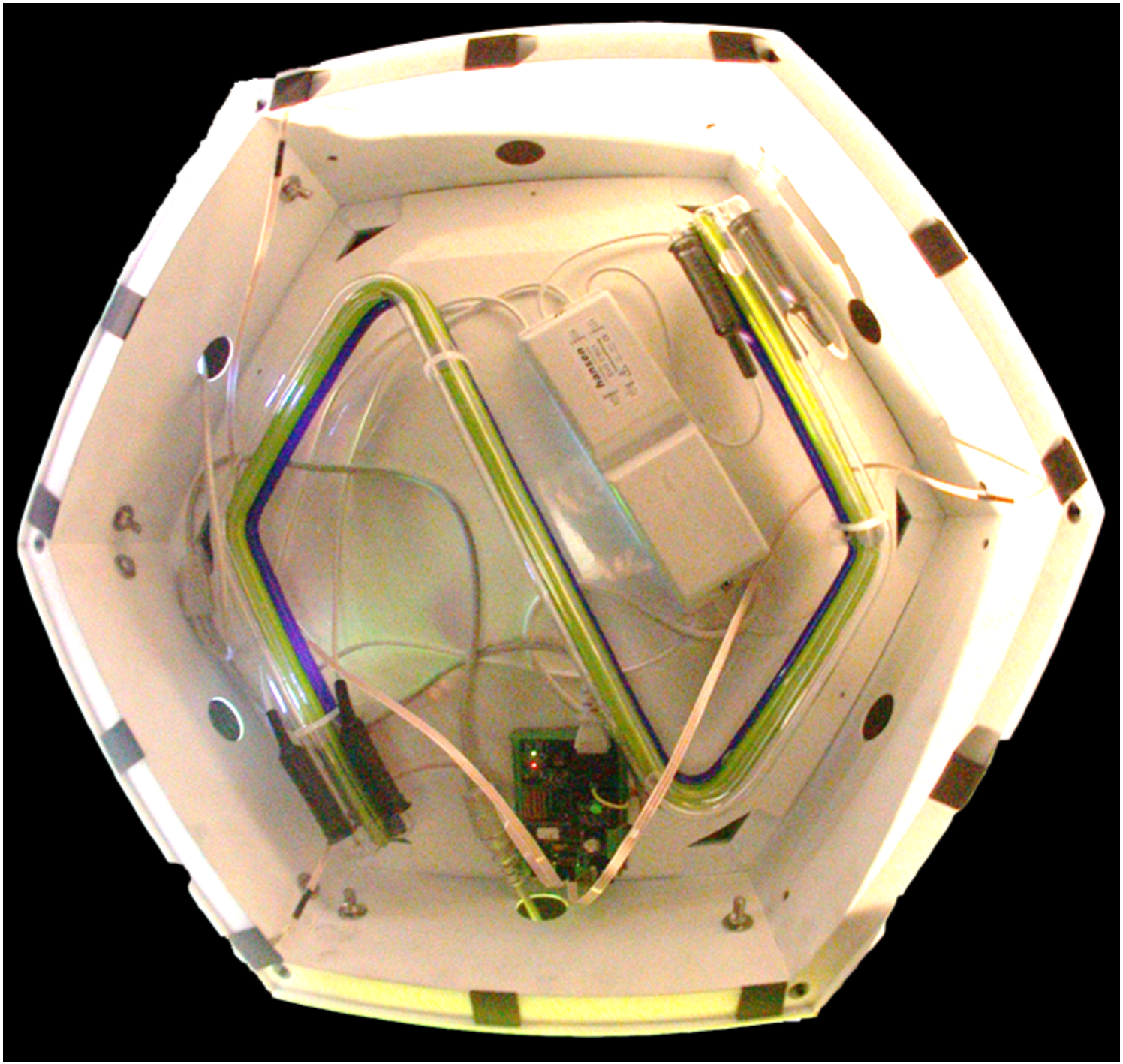


PCB design and layout

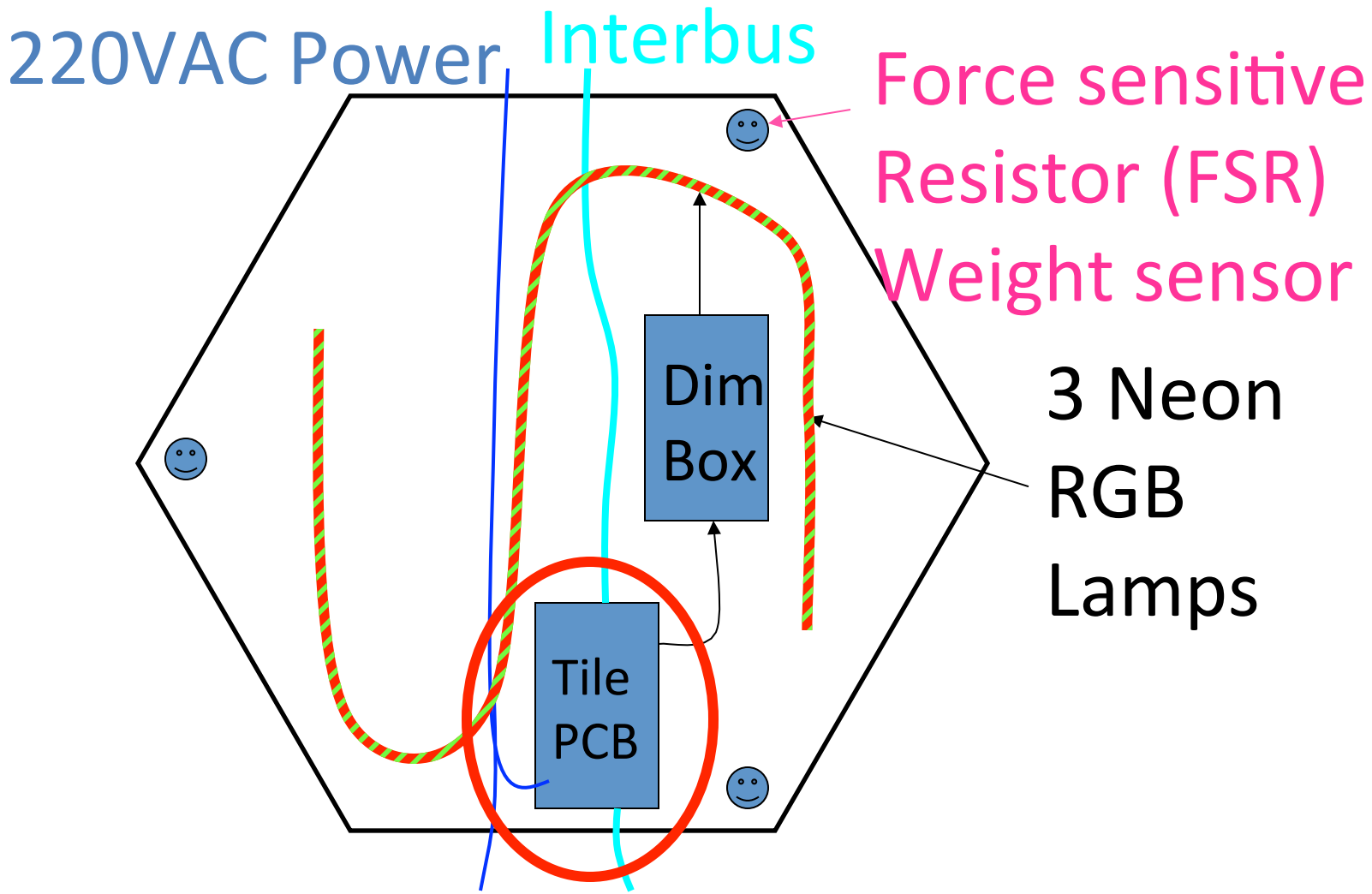
Embedded system design example

Ada's luminous tactile floor

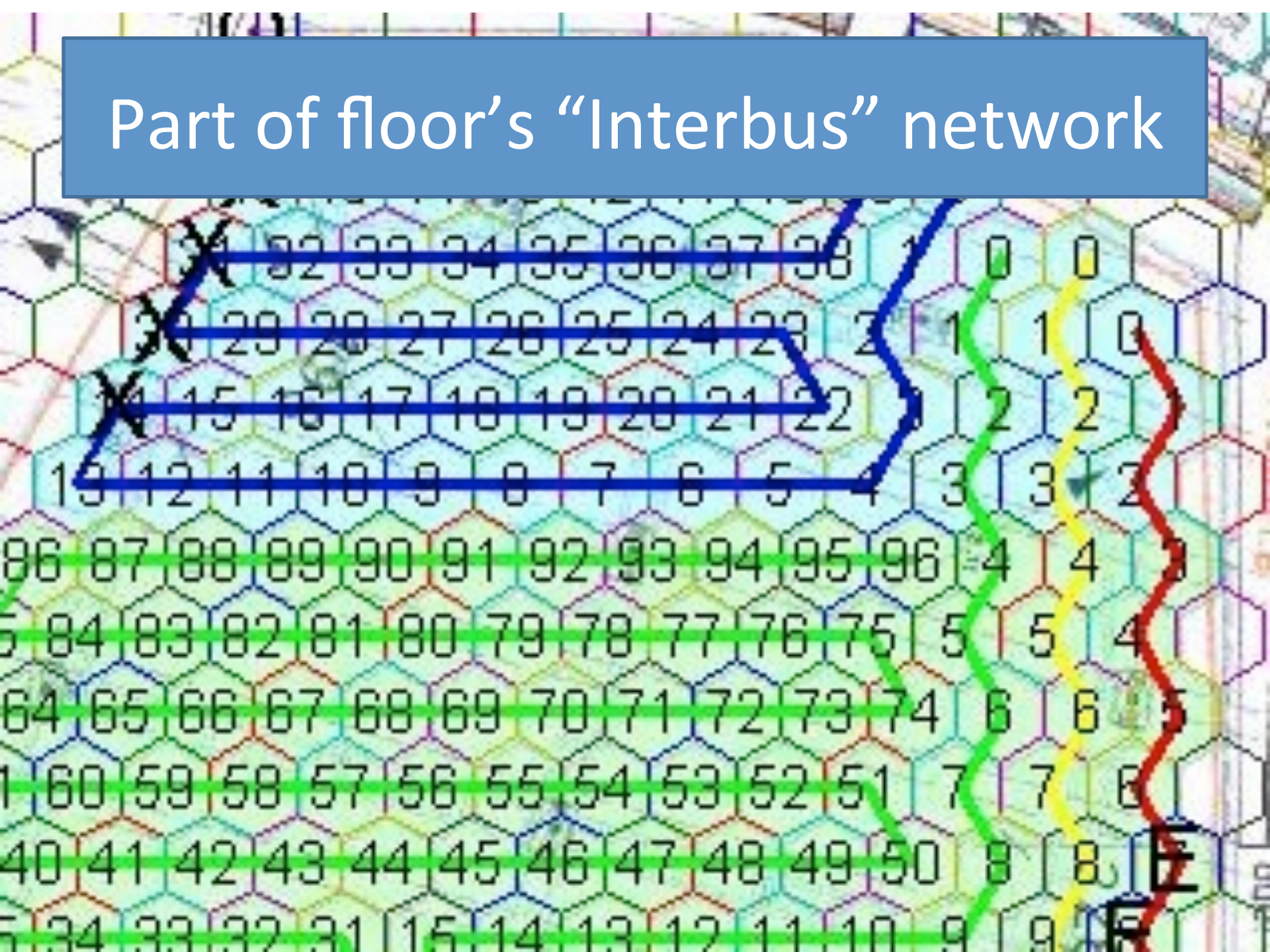
Ada’s luminous tactile floor



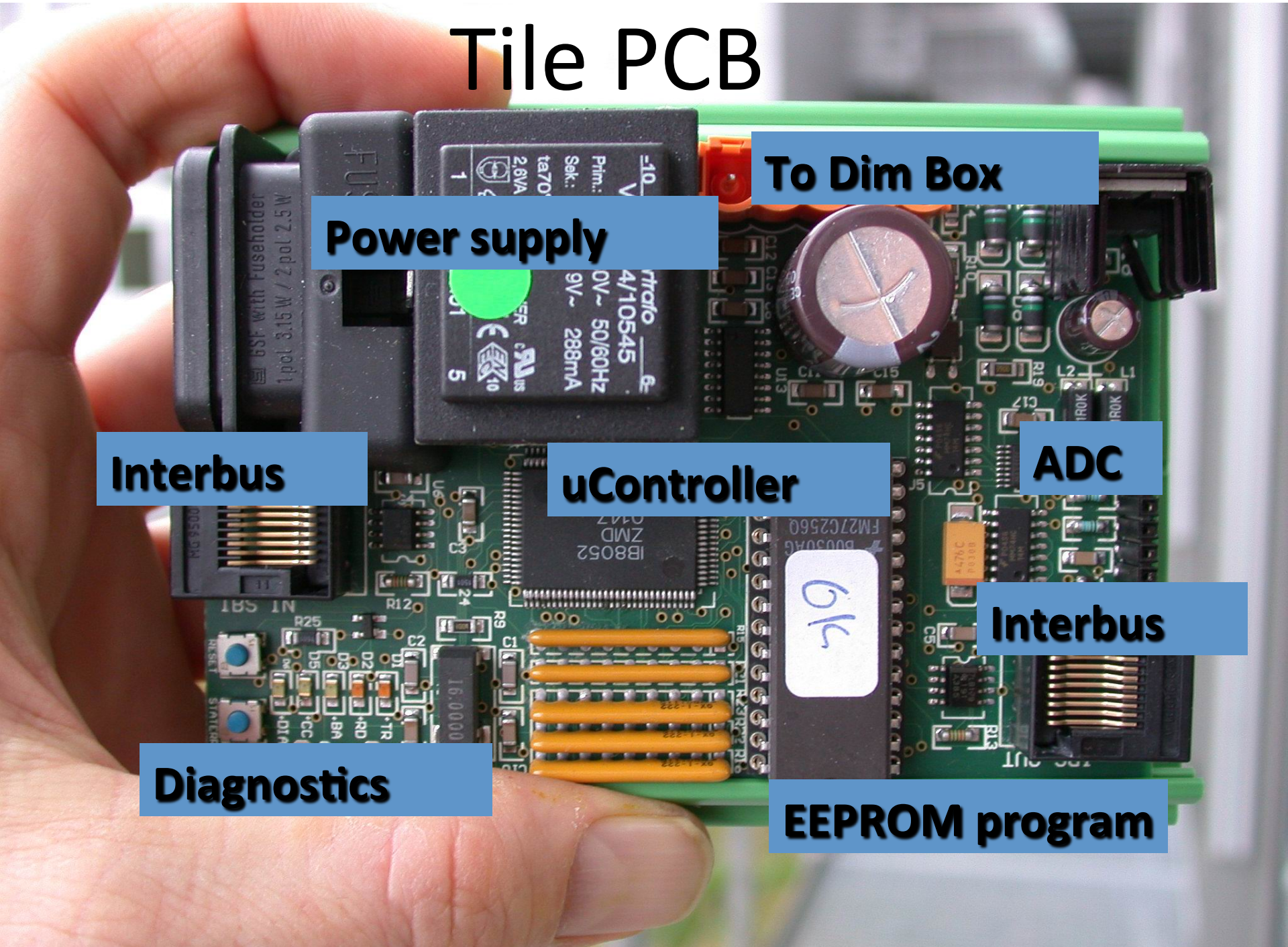
Neon Tile


220VAC Power
Interbus
Force sensitive Resistor (FSR) Weight sensor
Dim Box
3 Neon RGB Lamps
Tile PCB

Part of floor's "Interbus" network



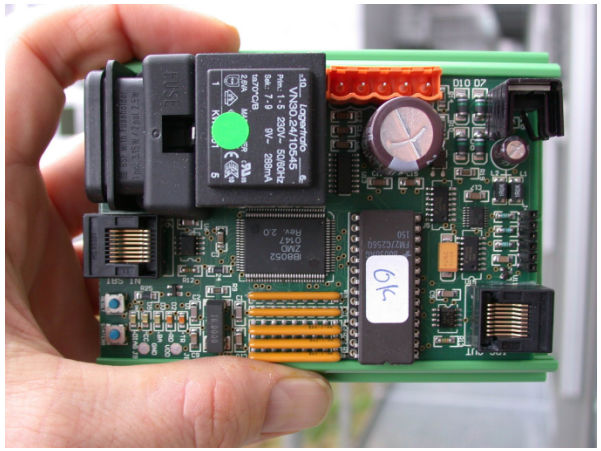
Tile PCB


To Dim Box
Power supply
Interbus
uController
ADC
Interbus
Diagnostics
EEPROM program

PCB assembly

+

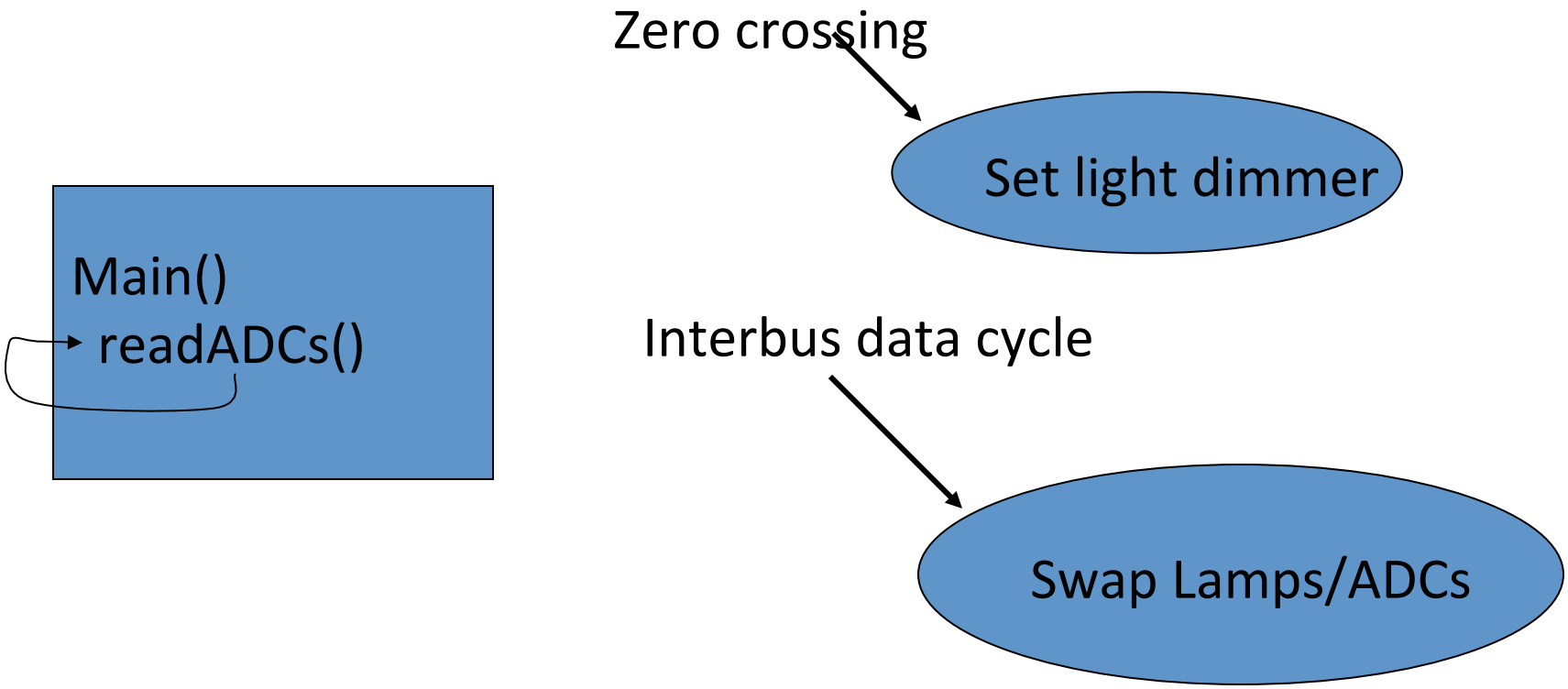
=



X 500 @25 CHF/board

Tile Microcontroller Program

Interrupt service routines (ISRs)



1000 lines C/assembler code running on 8052 derivative
Two demo tiles have been running continuously for >5 years!

The ATmega168P / 328P

- AT = Atmel: Big microcontroller company
- mega: microcontroller family
- 16: 16KB Flash memory / 32: 32KB Flash
- 8: 8-bit architecture
- P: PicoPower Technology. Optional. For low power battery-based applications.

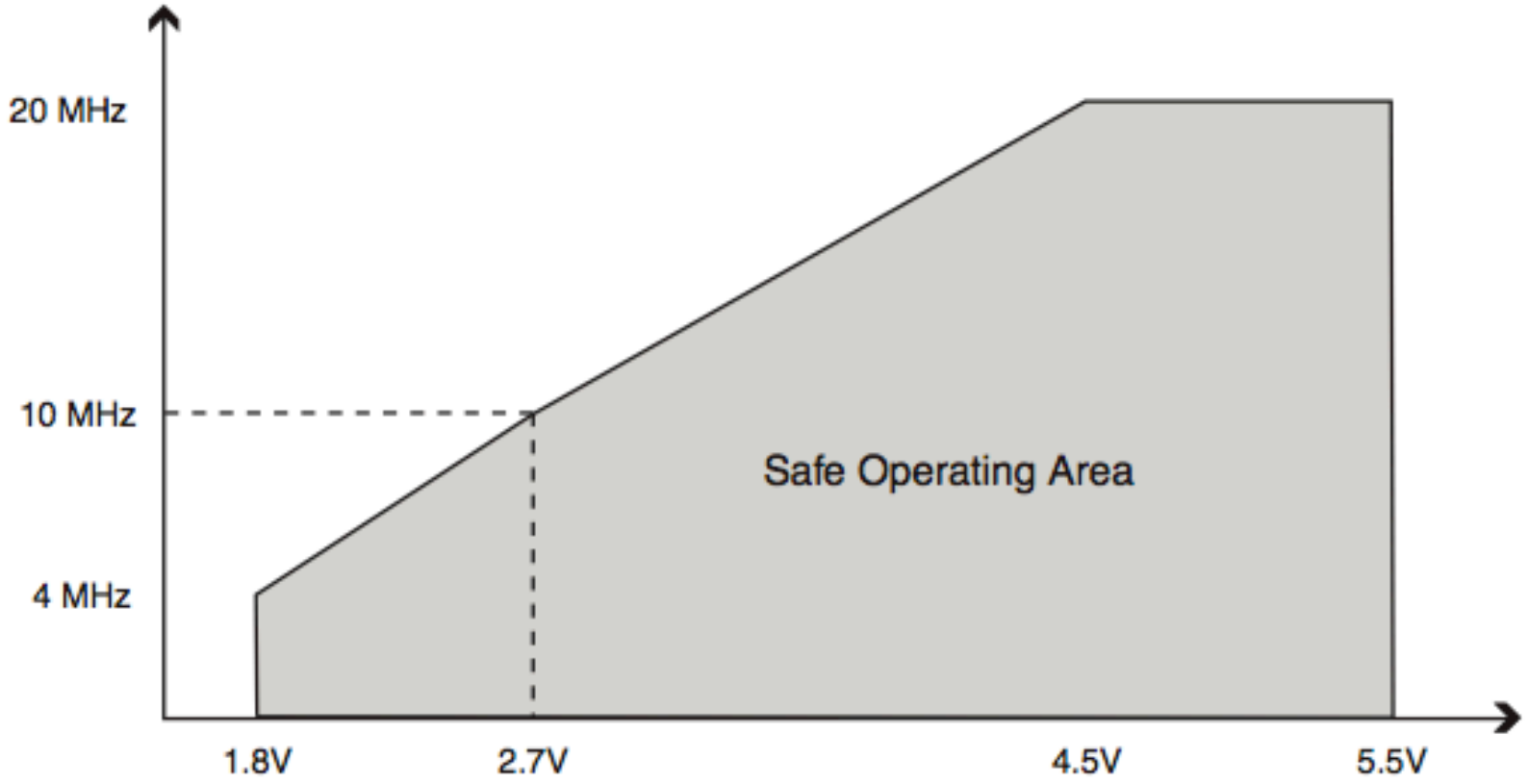
ATmega16/328P capabilities

- **System Functions**
  - Power and Clock Manager
  - Low Freq Internal Oscillator
  - Watchdog Timer
  - Real-Time Clock Timer
- **Interrupt Controller**
  - Fixed priority. One level of interruption. Interruptions with flag (can remember) or without. Global Interrupt Enable (I-bit) is disabled during an interrupt service.
- **NO Universal Serial Bus (USB)**
  - This micro hasn't USB. The nano board provide an USB-USART interface from FTDI company.

- One 16-bit **Timer/ Counter (TC)** with Auto-Reload and PWM
- Two 8-bit **Timer / Counter (TC)** with AR and PWM
- One 8-channel 10-bit **Analog-To-Digital Converter (ADC)**, 76.9ks/s
- SPI, USART, I2C

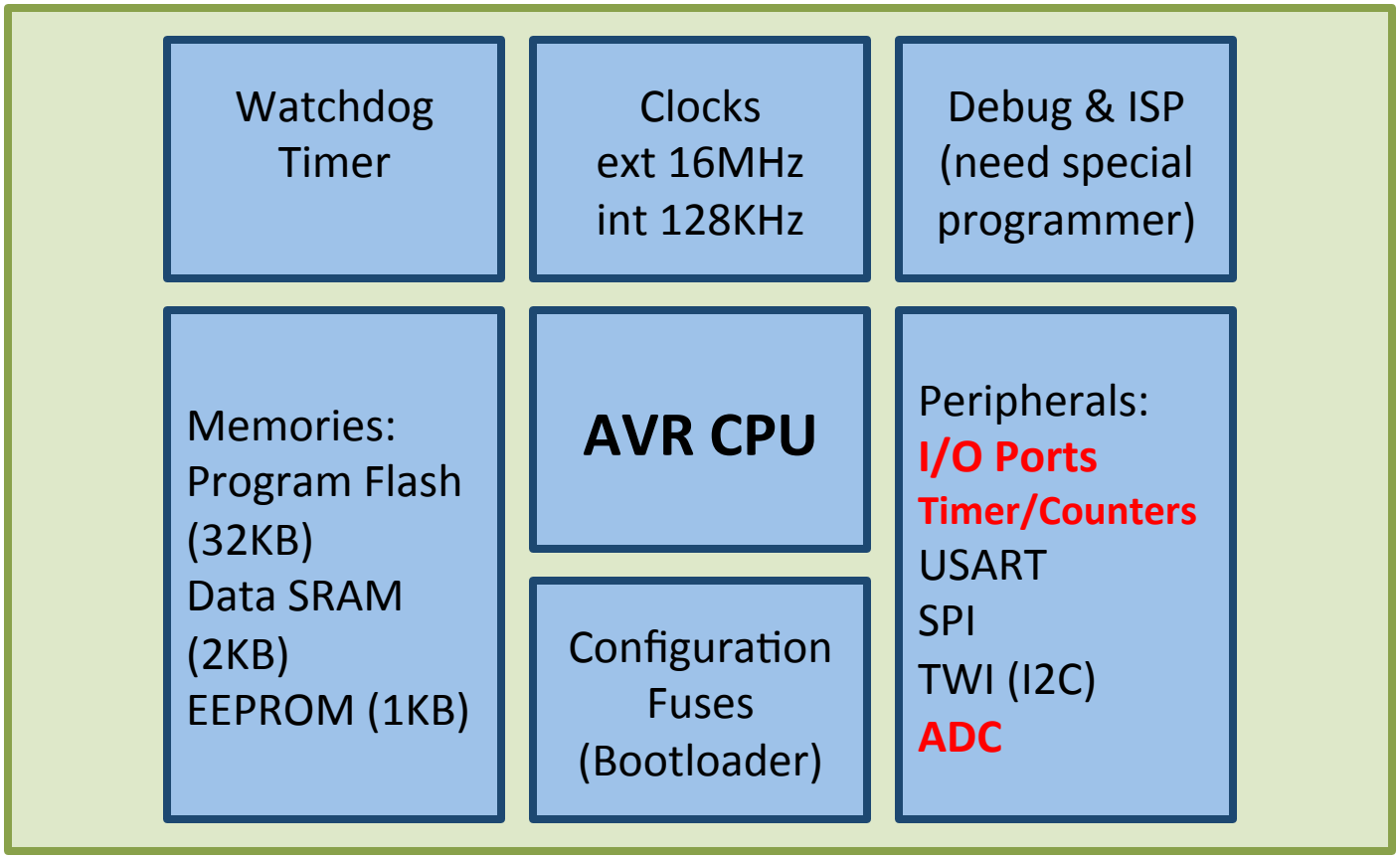
Clocks

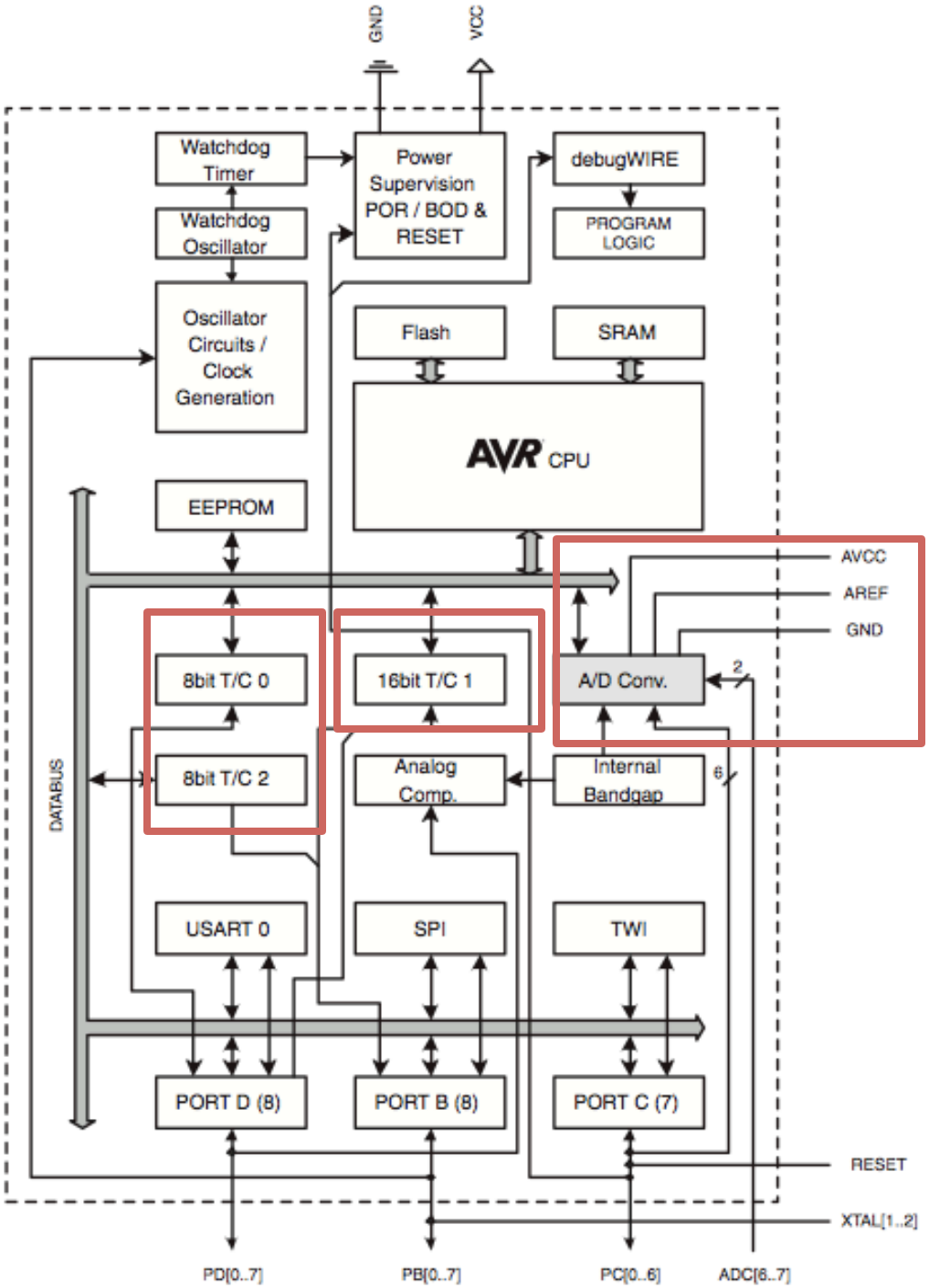
Figure 28-1. Maximum Frequency vs. V_{CC}



Arduino ATmega328P uses a 16MHz Resonator with 5v Vcc, to keep compatibility to ATmega8

ATmega328 simplified block diagram

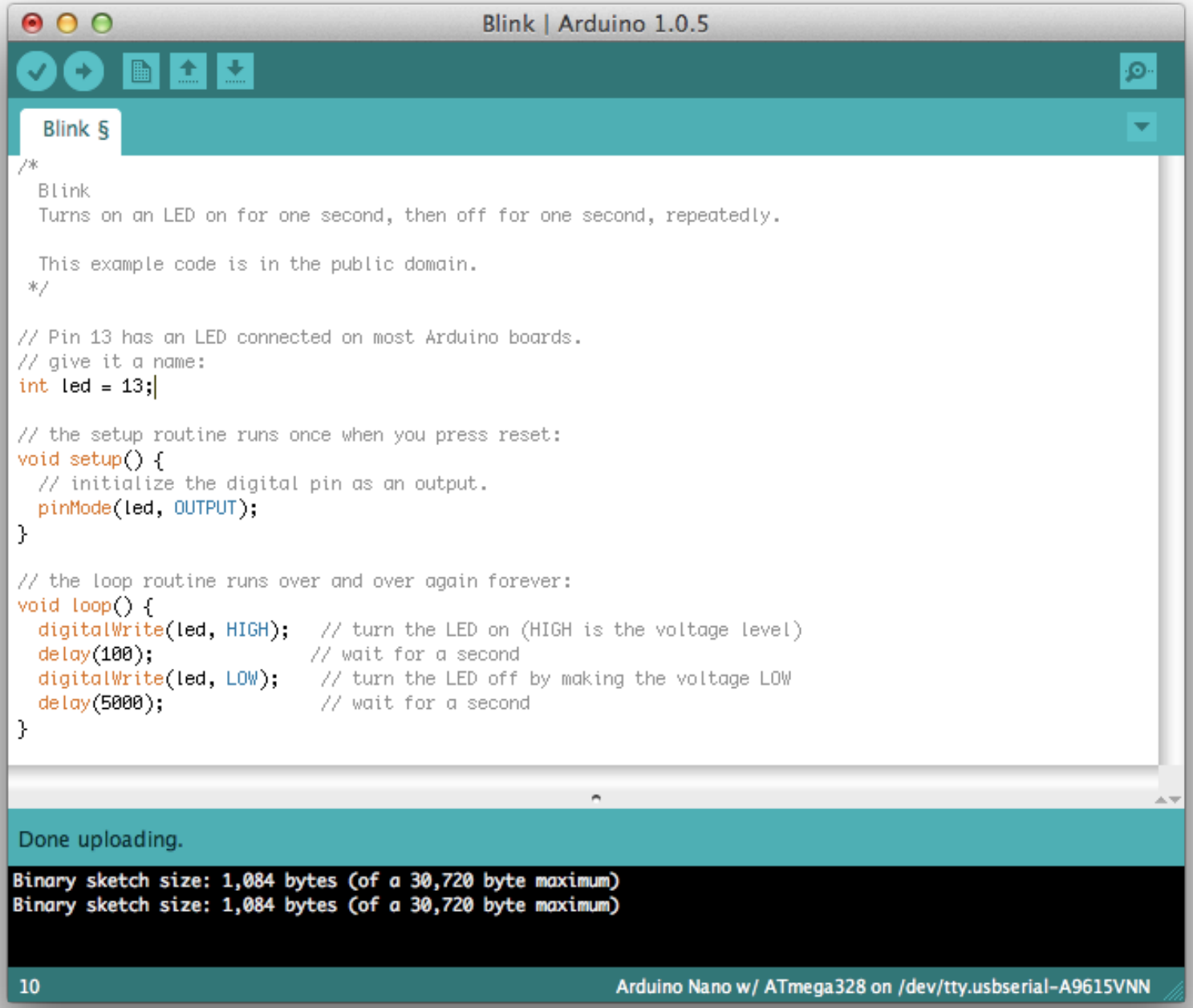



GND
VCC
Watchdog Timer
Watchdog Oscillator
Power Supervision POR / BOD & RESET
debugWIRE
PROGRAM LOGIC
Oscillator Circuits / Clock Generation
Flash
SRAM
AVR CPU
EEPROM
AVCC
AREF
GND
8bit T/C 0
16bit T/C 1
A/D Conv.
2
8bit T/C 2
Analog Comp.
Internal Bandgap
6
DATABUS
USART 0
SPI
TWI
PORT D (8)
PORT B (8)
PORT C (7)
RESET
XTAL[1..2]
PD[0..7]
PB[0..7]
PC[0..6]
ADC[6..7]

Atmega168/328P datasheet

- 440 pages
- Useful for architecture comprehension
- Peripheral descriptions
- Not all details are necessary since Arduino bootloader make things easier!!

Arduino Sketch



C programming for Arduino

ANSI C STYLE

```
void setup(void);
void loop(void);

void main(void) {
  setup(); //perform one-time initializations
  while(1) {
    loop();  //repeat this over and over
  }
}

void setup(void) {
}
void loop(void) {
}
```

ARDUINO CODE STYLE

```
void setup() {
}

void loop() {
}
```

Any other necessary stuff is included by Arduino IDE during compilation time and it Is transparent for the user / programmer (you)

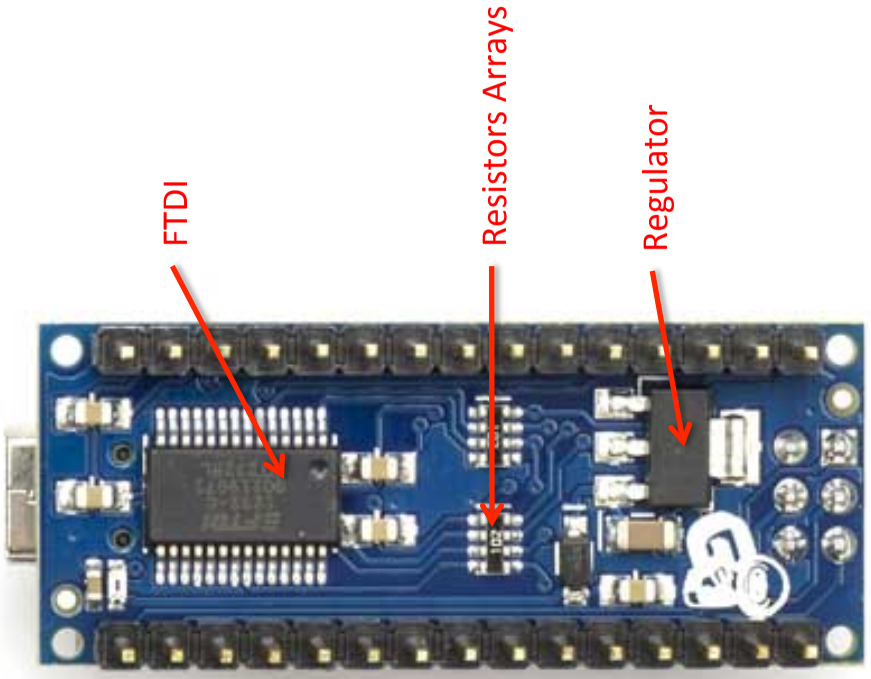
- Managing a LED:
  - Setup: pinMode(13, OUTPUT); // bitSet(DDRB,5);
  - Loop: digitalWrite(13, HIGH); //or LOW
- Managing Serial port:
  - Setup: Serial.begin(rate); //Serial.end();
  - Loop: Serial.println(data);
    Serial.println(data, data_type); //DEC,BIN,HEX,OCT, BYTE
    Serial.print(data, data_type);
    n=Serial.available(); //n=number of bytes (max 128B)
    b=Serial.read(); // Serial.write(b);
    b=Serial.peek();
    Serial.flush()
  - void serialEvent() {}

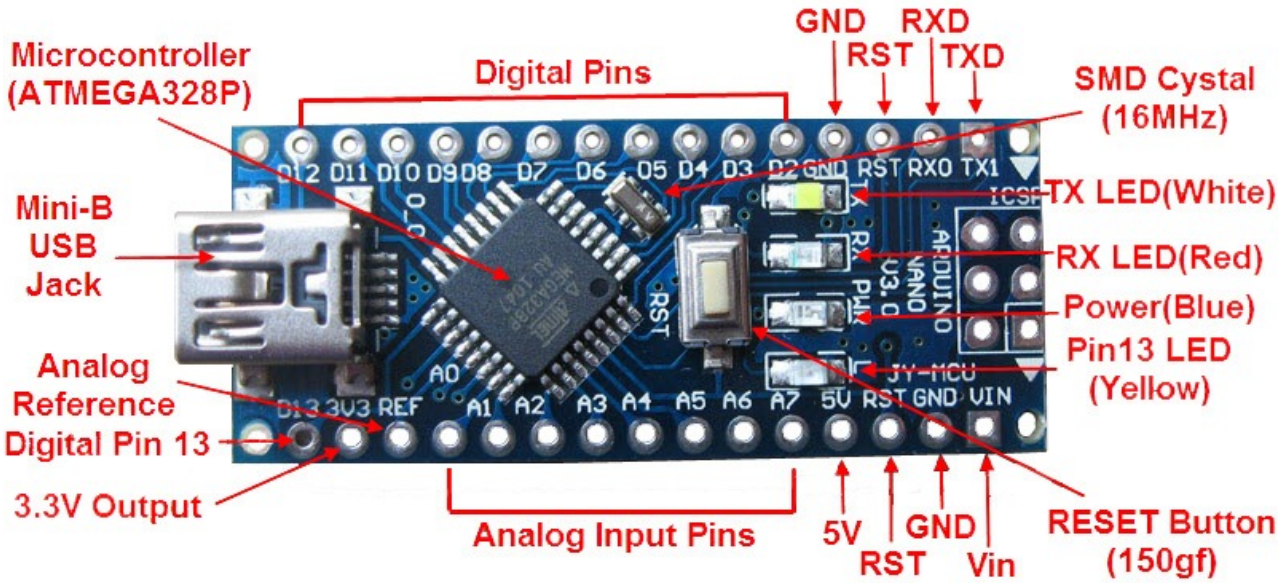
Standard AVR code

```
#include <stdio.h>
#include <avr/io.h>

#ifndef F_CPU
#       error Must define F_CPU or pass it as compiler argument
#endif

FILE uart_out = FDEV_SETUP_STREAM(uart_putChar, NULL, _FDEV_SETUP_WRITE);
void main(void) __attribute__((noreturn));

void uart_init(){
#       define BAUD 9600
#       include <util/setbaud.h>
        UBRR0H = UBRRH_VALUE;
        UBRR0L = UBRRL_VALUE;
#       if USE_2x
                UCSR0A |= _BV(U2X);
#       else
                UCSR0A &= ~_BV(U2X);
#       endif
        UCSR0B |= _BV(TXEN);                    // enable transmit
}

int uart_putChar(char c, FILE *unused){
        if (c == '\n')                          // a line feed character also
                uart_putChar('\r', unused);     // requires a carriage return
        loop_until_bit_is_set(UCSR0A, UDRE0);   // wait for UDR0 to be ready
        UDR0 = c;                               // write character
        return 0;                               // return
}

void main(void){
        uart_init();
        fprintf(&uart_out, "Hello, UART!\n");
        stdout = &uart_out;                     // set up stdout
        printf("UART demo...\n");
        for (uint8_t i = 0; i < 10; i++)
                printf("Number %i ", i);
        while(1);
}
```

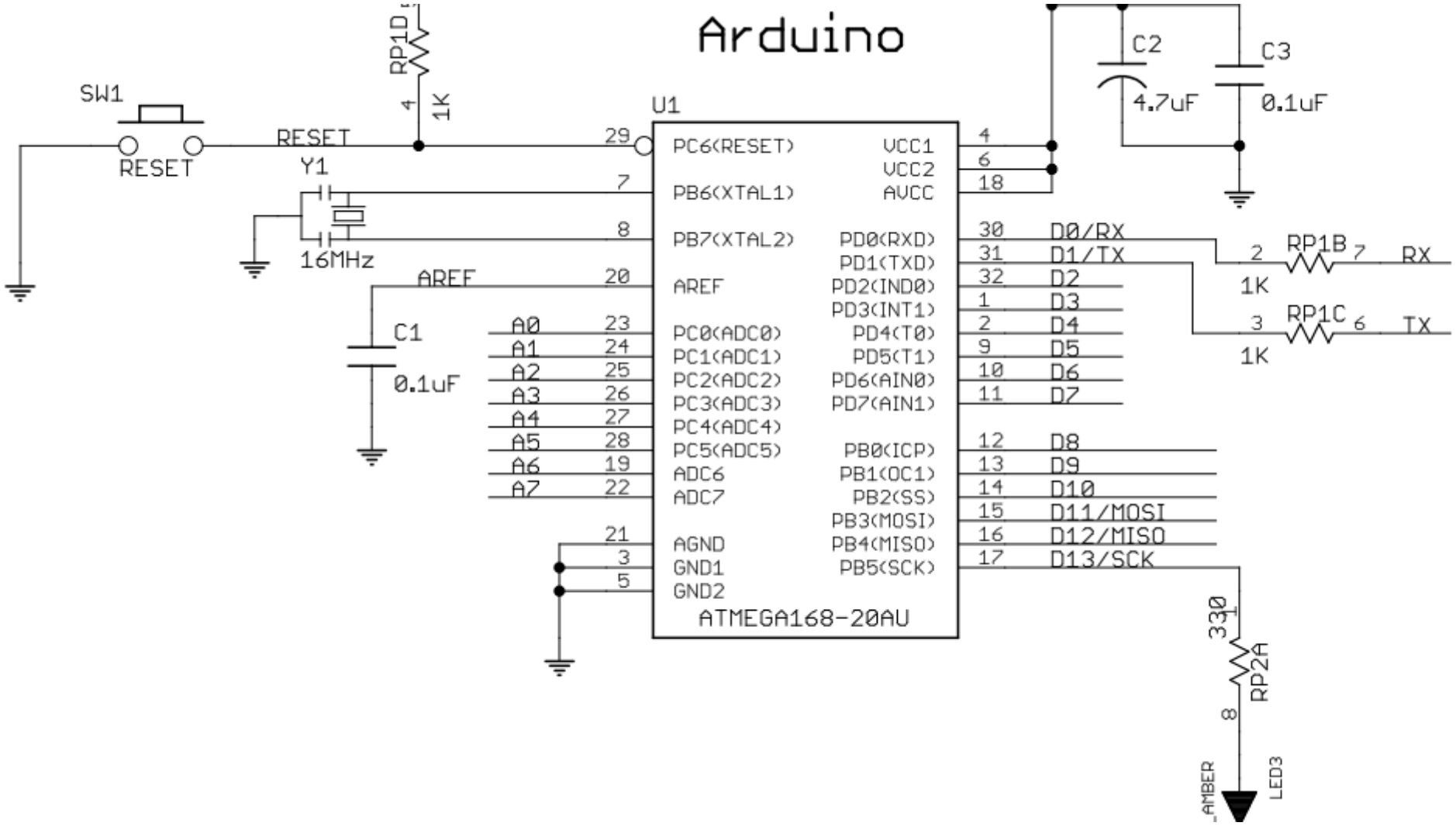
Arduino Serial code

```
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  Serial.println("Hello UART");
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```


FTDI
Resistors Arrays
Regulator


Microcontroller
(ATMEGA328P)
Digital Pins
GND
RXD
RST
TXD
SMD Cystal
(16MHz)
TX LED(White)
RX LED(Red)
Power(Blue)
Pin13 LED
(Yellow)
Mini-B
USB
Jack
Analog
Reference
Digital Pin 13
3.3V Output
Analog Input Pins
5V
GND
RST
Vin
RESET Button
(150gf)

Arduino Nano EFP2 RTC/INI schematics



Exercise 1

- Installing and running Arduino tools.
- Opening and understanding basic blinking Example Sketch and run it.
- Add Serial communication to Serial Monitor for “debugging” your Sketch.
- Arranging time for soldering your own Arduino Nano board and make it compatible with Sketch.
- Exercise 2: probably debugging and repairing your own Arduino Nano board

The AVR32 AT32UC3B1256

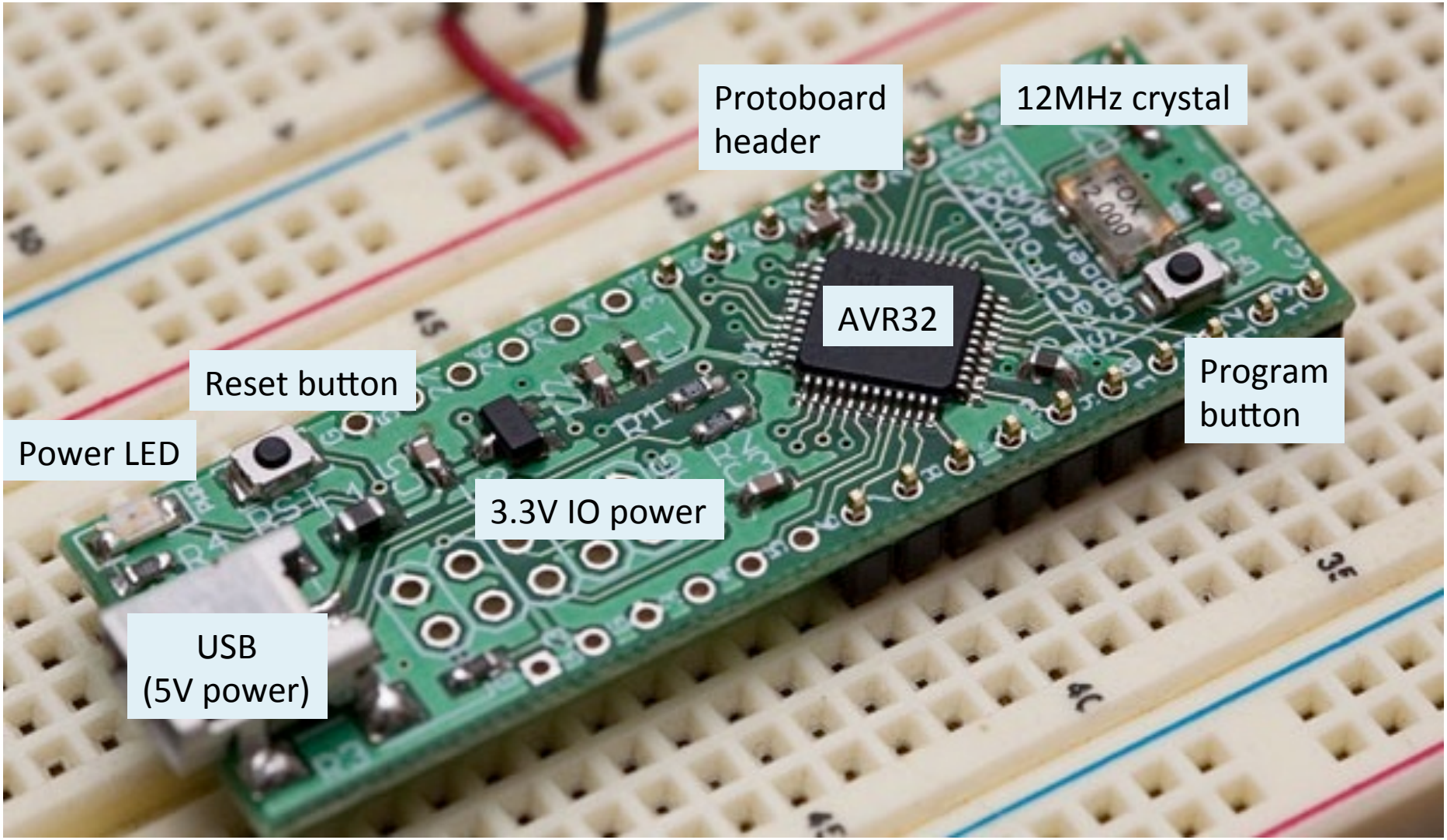
- AT = Atmel: Big microcontroller company
- 32 = 32 bit architecture
- UC3 = Atmel microcontroller family
- B = more powerful and expensive variant ($7 each @25 units)
- 1 = revision
- 256 = 256kB internal high speed flash memory (32kB single cycle SRAM)

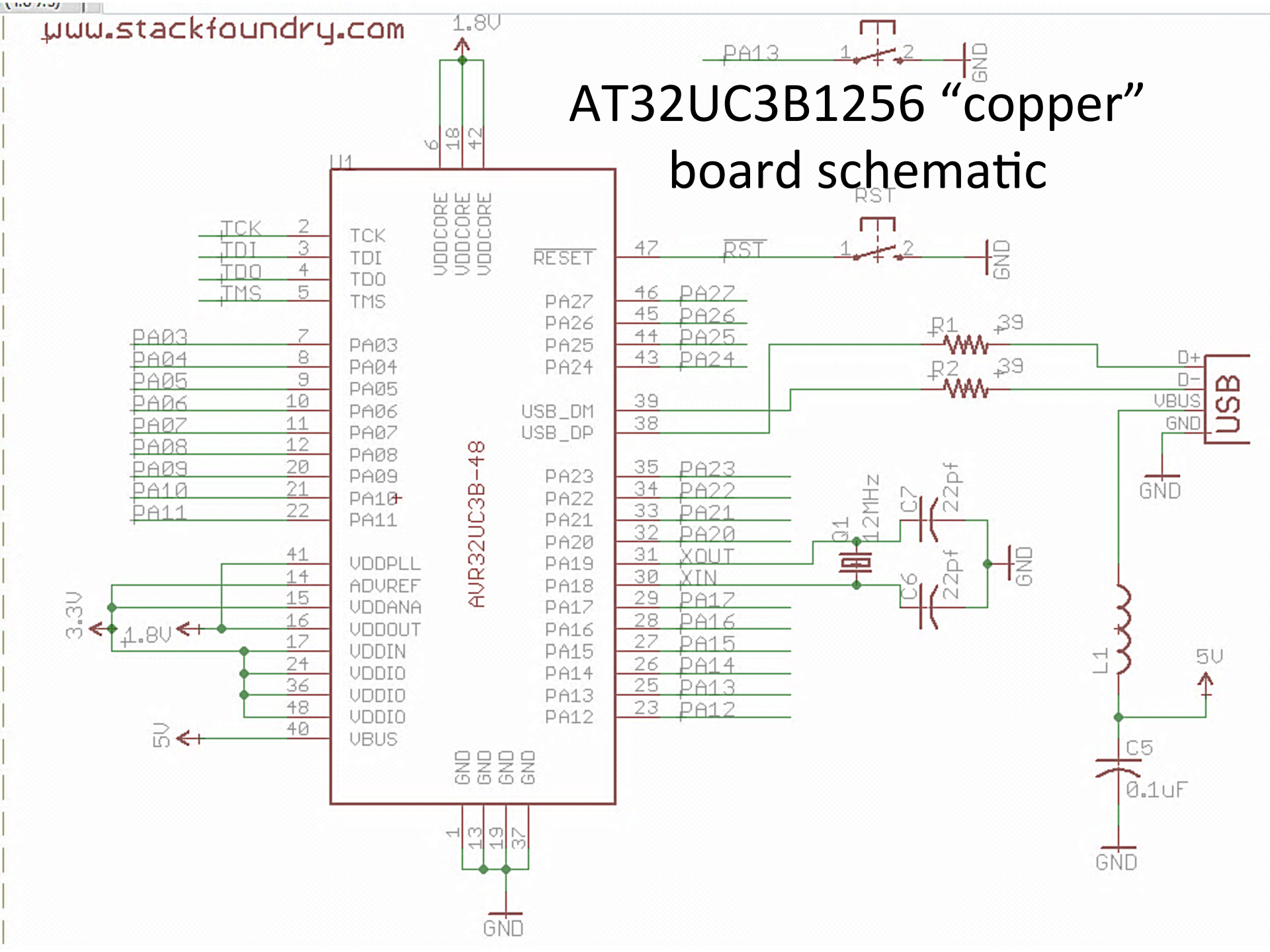
AVR32 capabilities

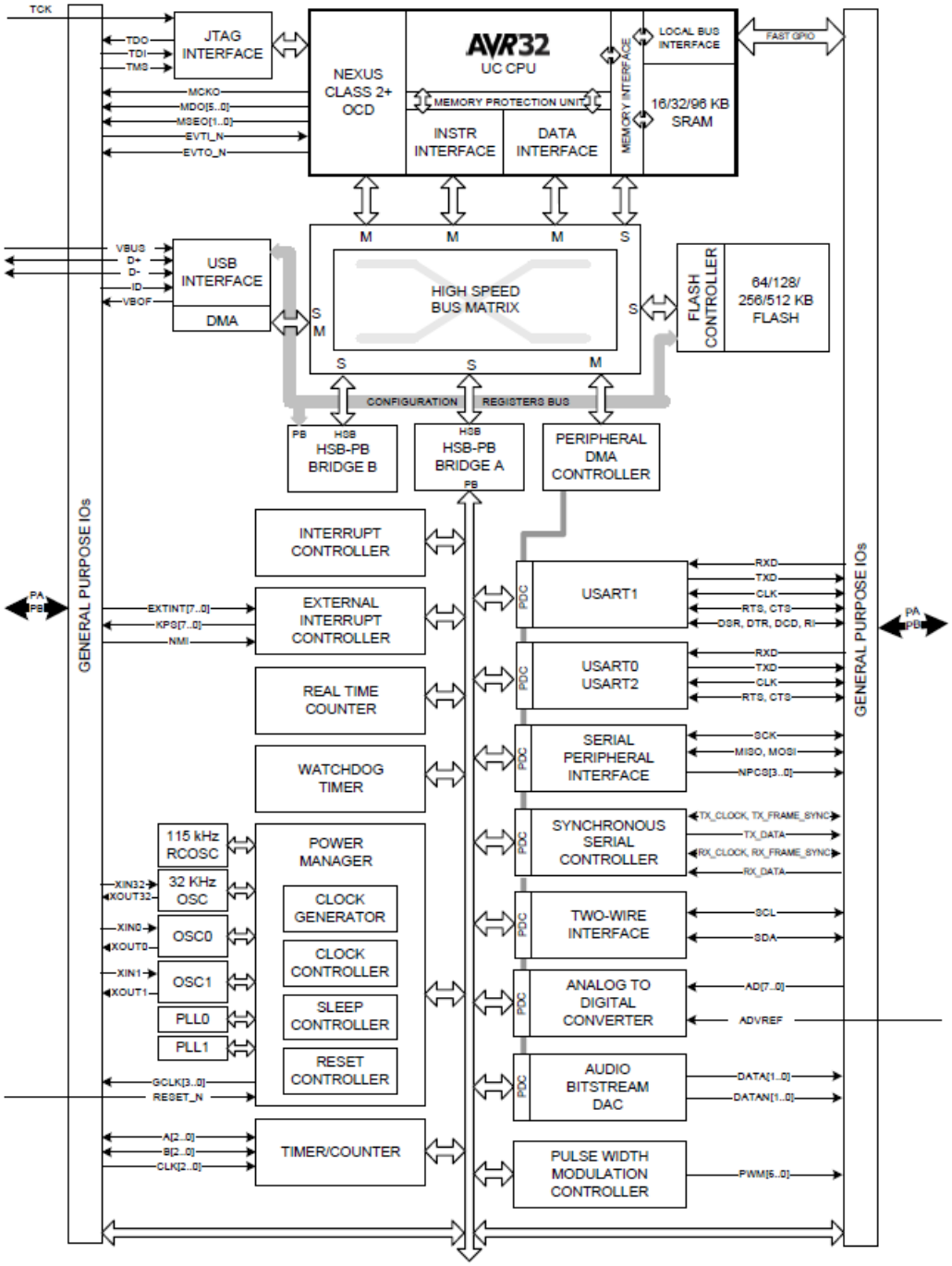
- **System Functions**
  - Power and Clock Manager
  - Two Multipurpose Oscillators
  - Watchdog Timer
  - Real-Time Clock Timer
- **Interrupt Controller**
  - Auto-vectored Low Latency Interrupt Service with Programmable Priority
- **Universal Serial Bus (USB)**
  - Device 2.0 Full Speed (12Mbps~1MBps)

- One Three-Channel 16-bit **Timer/Counter (TC)**
- One 7-Channel 20-bit **Pulse Width Modulation Controller (PWM)**
- One 8-channel 10-bit **Analog-To-Digital Converter (ADC)**, 384ks/s
- plus many more modules

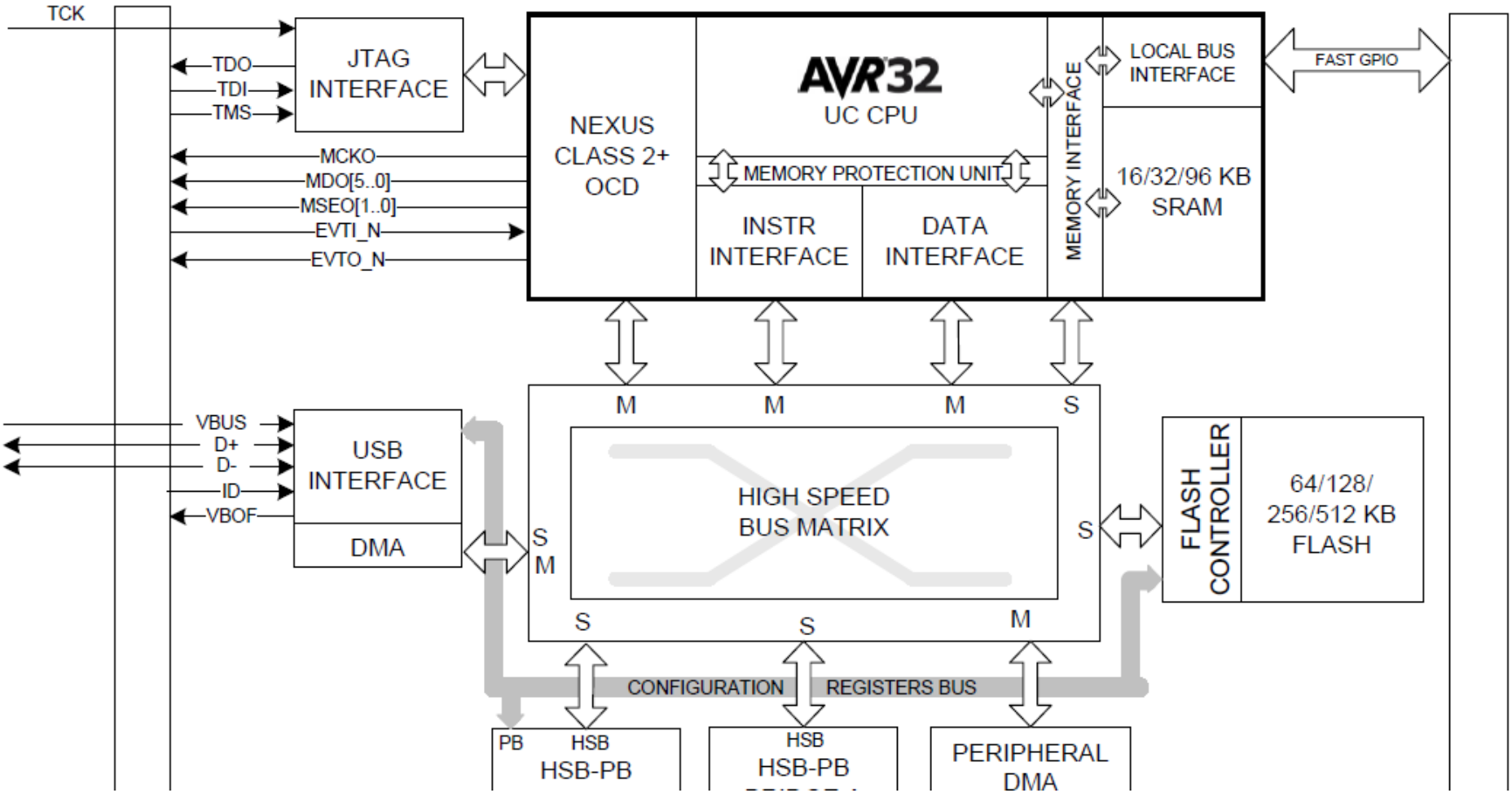
The “bronze” board

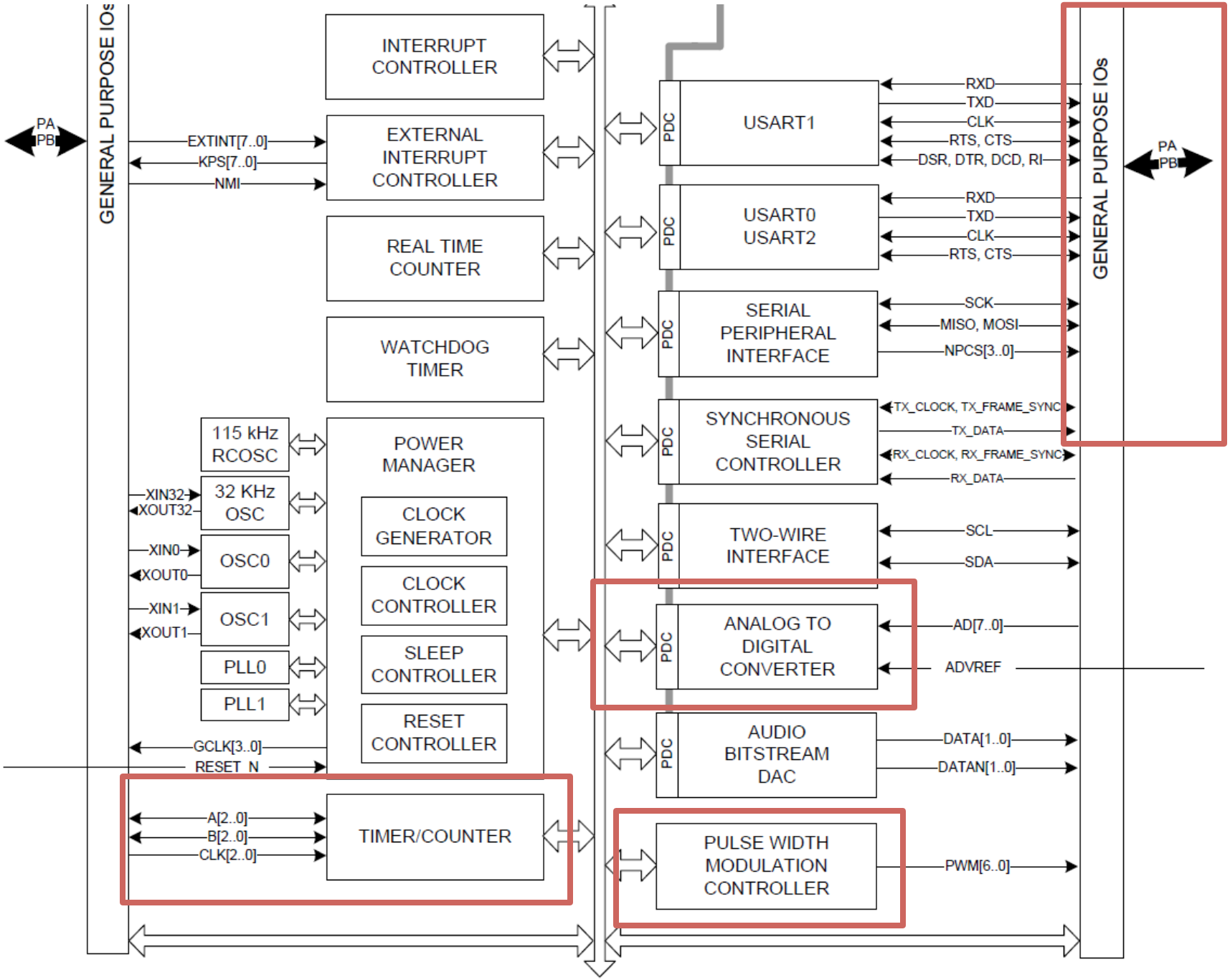



www.stackfoundry.com
AT32UC3B1256 “copper” board schematic
AVR32UC3B-48
U1
1.8V
3.3V
5V
GND
RST
PA13
TCK
TDI
TDO
TMS
VDDCORE
RESET
PA27
PA26
PA25
PA24
USB_DM
USB_DP
PA23
PA22
PA21
PA20
XOUT
XIN
PA19
PA18
PA17
PA16
PA15
PA14
PA12
PA03
PA04
PA05
PA06
PA07
PA08
PA09
PA10
PA11
VDDPLL
ADVREF
VDDANA
VDDOUT
VDDIN
VDDIO
VBUS
R1
R2
39
D+
D-
VBUS
USB
Q1
12MHz
C7
C6
22pf
L1
C5
0.1uF


TCK
TDO
TDI
TMS
JTAG INTERFACE
MCKO
MDO[5..0]
MSEO[1..0]
EVTI_N
EVTO_N
NEXUS CLASS 2+ OCD
AVR32
UC CPU
MEMORY PROTECTION UNIT
INSTR INTERFACE
DATA INTERFACE
MEMORY INTERFACE
LOCAL BUS INTERFACE
FAST GPIO
16/32/96 KB SRAM
M
M
M
S
VBUS
D+
D-
ID
VBOF
USB INTERFACE
DMA
S
M
HIGH SPEED BUS MATRIX
S
FLASH CONTROLLER
64/128/ 256/512 KB FLASH
S
S
M
CONFIGURATION
REGISTERS BUS
PB
HSB
HSB-PB BRIDGE B
HSB
HSB-PB BRIDGE A
PERIPHERAL DMA CONTROLLER
PB
GENERAL PURPOSE IOs
PA
PB
INTERRUPT CONTROLLER
EXTINT[7..0]
KPS[7..0]
NMI
EXTERNAL INTERRUPT CONTROLLER
REAL TIME COUNTER
WATCHDOG TIMER
115 kHz RCOSC
XIN32
XOUT32
32 KHz OSC
XIN0
XOUT0
OSC0
XIN1
XOUT1
OSC1
PLL0
PLL1
POWER MANAGER
CLOCK GENERATOR
CLOCK CONTROLLER
SLEEP CONTROLLER
RESET CONTROLLER
GCLK[3..0]
RESET_N
A[2..0]
B[2..0]
CLK[2..0]
TIMER/COUNTER
PDC
USART1
RXD
TXD
CLK
RTS, CTS
DSR, DTR, DCD, RI
USART0 USART2
SERIAL PERIPHERAL INTERFACE
SCK
MISO, MOSI
NPCS[3..0]
SYNCHRONOUS SERIAL CONTROLLER
TX_CLOCK, TX_FRAME_SYNC
TX_DATA
RX_CLOCK, RX_FRAME_SYNC
RX_DATA
TWO-WIRE INTERFACE
SCL
SDA
ANALOG TO DIGITAL CONVERTER
AD[7..0]
ADVREF
AUDIO BITSTREAM DAC
DATA[1..0]
DATAN[1..0]
PULSE WIDTH MODULATION CONTROLLER
PWM[6..0]
GENERAL PURPOSE IOs
PA
PB

AVR32


TCK
TDO
TDI
TMS
JTAG INTERFACE
MCKO
MDO[5..0]
MSEO[1..0]
EVTI_N
EVTO_N
NEXUS CLASS 2+ OCD
AVR32
UC CPU
MEMORY PROTECTION UNIT
INSTR INTERFACE
DATA INTERFACE
MEMORY INTERFACE
LOCAL BUS INTERFACE
16/32/96 KB SRAM
FAST GPIO
M
M
M
S
VBUS
D+
D-
ID
VBOF
USB INTERFACE
DMA
S
M
HIGH SPEED BUS MATRIX
S
S
S
M
FLASH CONTROLLER
64/128/ 256/512 KB FLASH
CONFIGURATION
REGISTERS BUS
PB
HSB
HSB-PB
HSB
HSB-PB
PERIPHERAL DMA

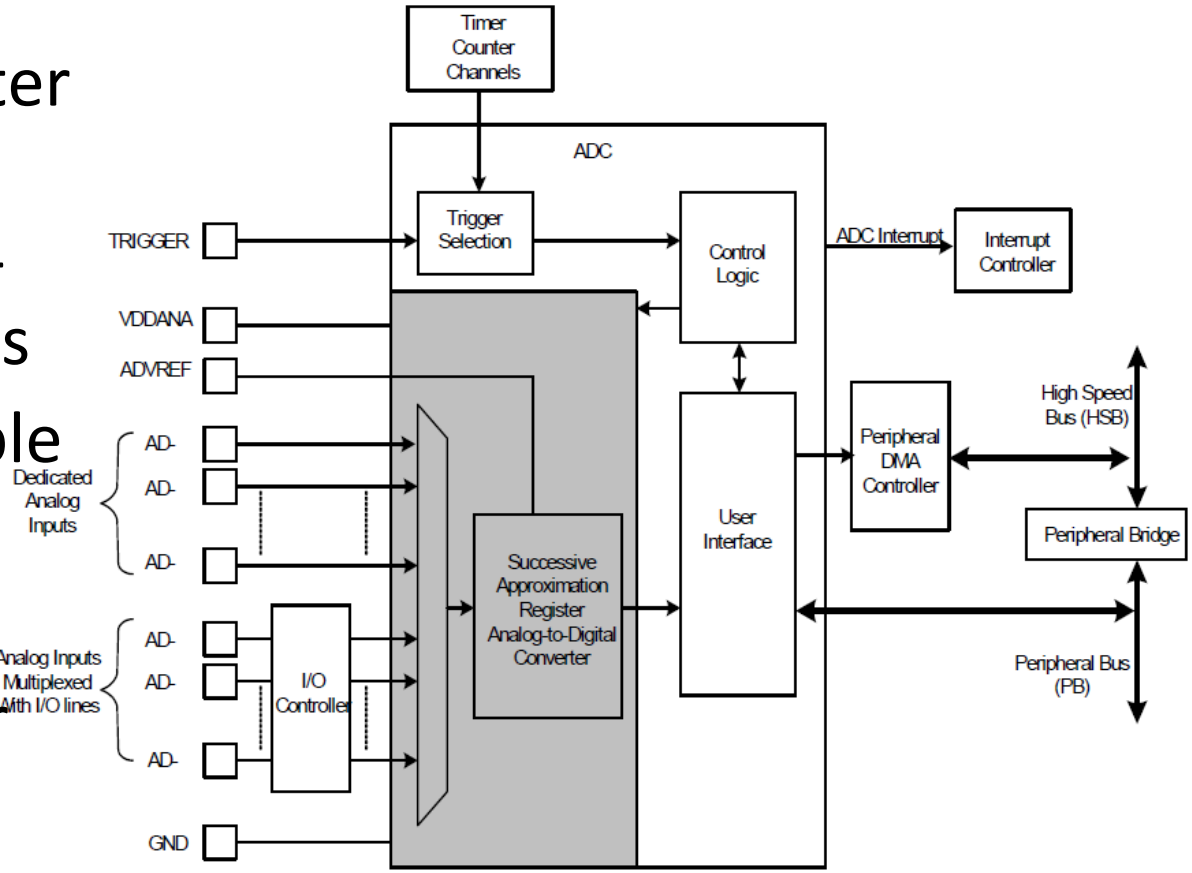

GENERAL PURPOSE IOs
PA
PB
INTERRUPT CONTROLLER
EXTERNAL INTERRUPT CONTROLLER
EXTINT[7..0]
KPS[7..0]
NMI
REAL TIME COUNTER
WATCHDOG TIMER
POWER MANAGER
115 kHz RCOSC
32 KHz OSC
XIN32
XOUT32
OSC0
XIN0
XOUT0
OSC1
XIN1
XOUT1
PLL0
PLL1
CLOCK GENERATOR
CLOCK CONTROLLER
SLEEP CONTROLLER
RESET CONTROLLER
GCLK[3..0]
RESET_N
TIMER/COUNTER
A[2..0]
B[2..0]
CLK[2..0]
PDC
USART1
RXD
TXD
CLK
RTS, CTS
DSR, DTR, DCD, RI
USART0 USART2
SERIAL PERIPHERAL INTERFACE
SCK
MISO, MOSI
NPCS[3..0]
SYNCHRONOUS SERIAL CONTROLLER
TX_CLOCK, TX_FRAME_SYNC
TX_DATA
RX_CLOCK, RX_FRAME_SYNC
RX_DATA
TWO-WIRE INTERFACE
SCL
SDA
ANALOG TO DIGITAL CONVERTER
AD[7..0]
ADVREF
AUDIO BITSTREAM DAC
DATA[1..0]
DATAN[1..0]
PULSE WIDTH MODULATION CONTROLLER
PWM[6..0]

AVR32 datasheet – ~700 pages

- Intro/Core features: 200pp
- Peripherals: 400pp (e.g. USB 180pp, GPIO 19pp)
- Electrical characteristics: 20pp

AVR32 Analog to Digital converter

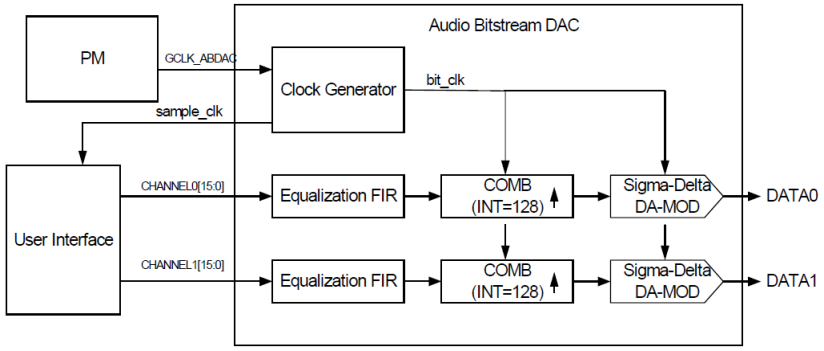
- 10-bit Successive approximation register (SAR) type
- 6 multiplexed single-ended input channels
- Max combined sample rate 384ks/s
- External trigger
- Hardware sequencer
- Peripheral DMA



Audio stereo bitstream DAC

The Audio Bitstream DAC converts a 16-bit sample value to a digital bitstream with an average value proportional to the sample value

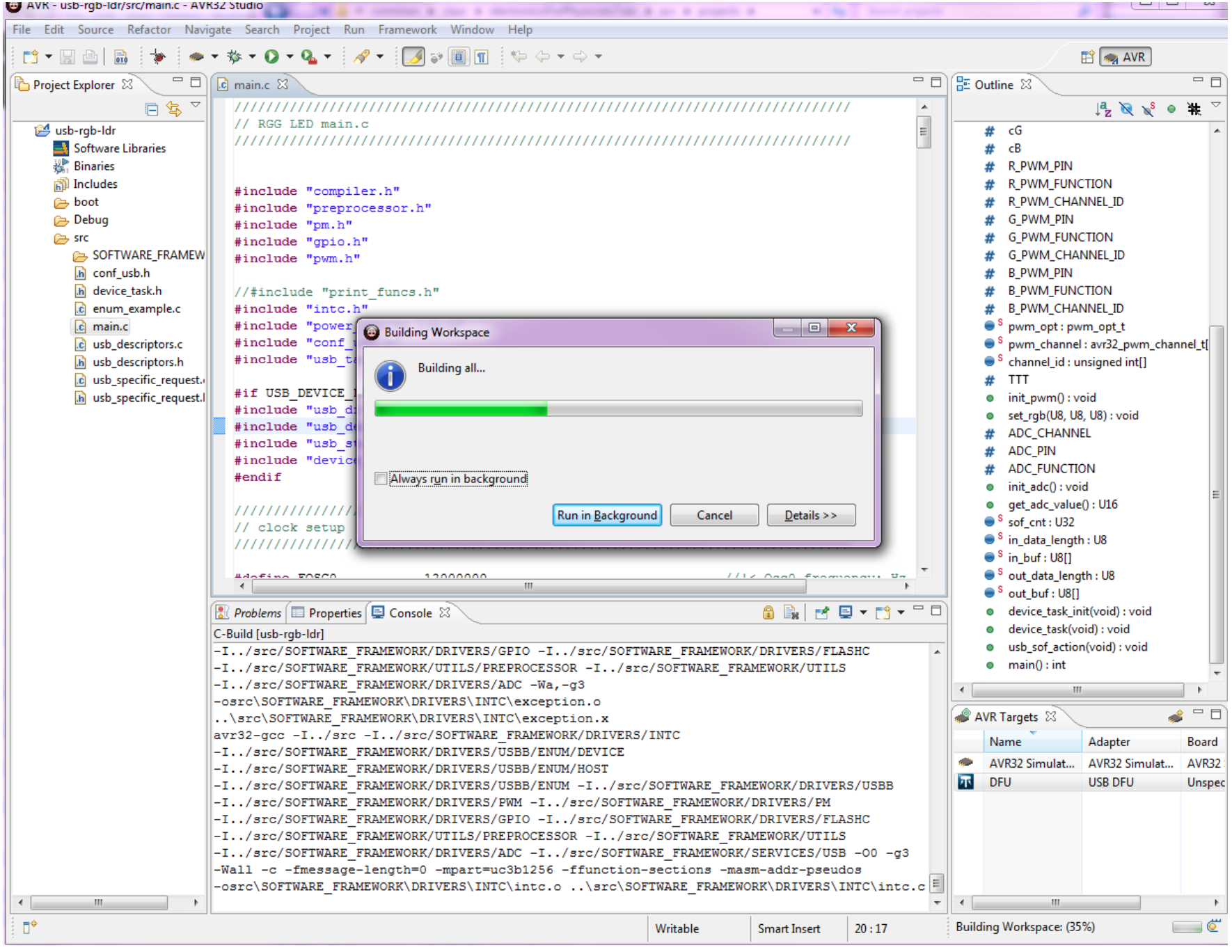
- **Oversampled D/A conversion architecture**
  - **Oversampling ratio fixed 128x**
  - **FIR equalization filter**
  - **Digital interpolation filter: Comb4**
  - **3rd Order Sigma-Delta D/A converters**
- **Digital bitstream outputs**
- **Parallel interface**
- **Connected to DMA Controller for background transfer without CPU intervention**



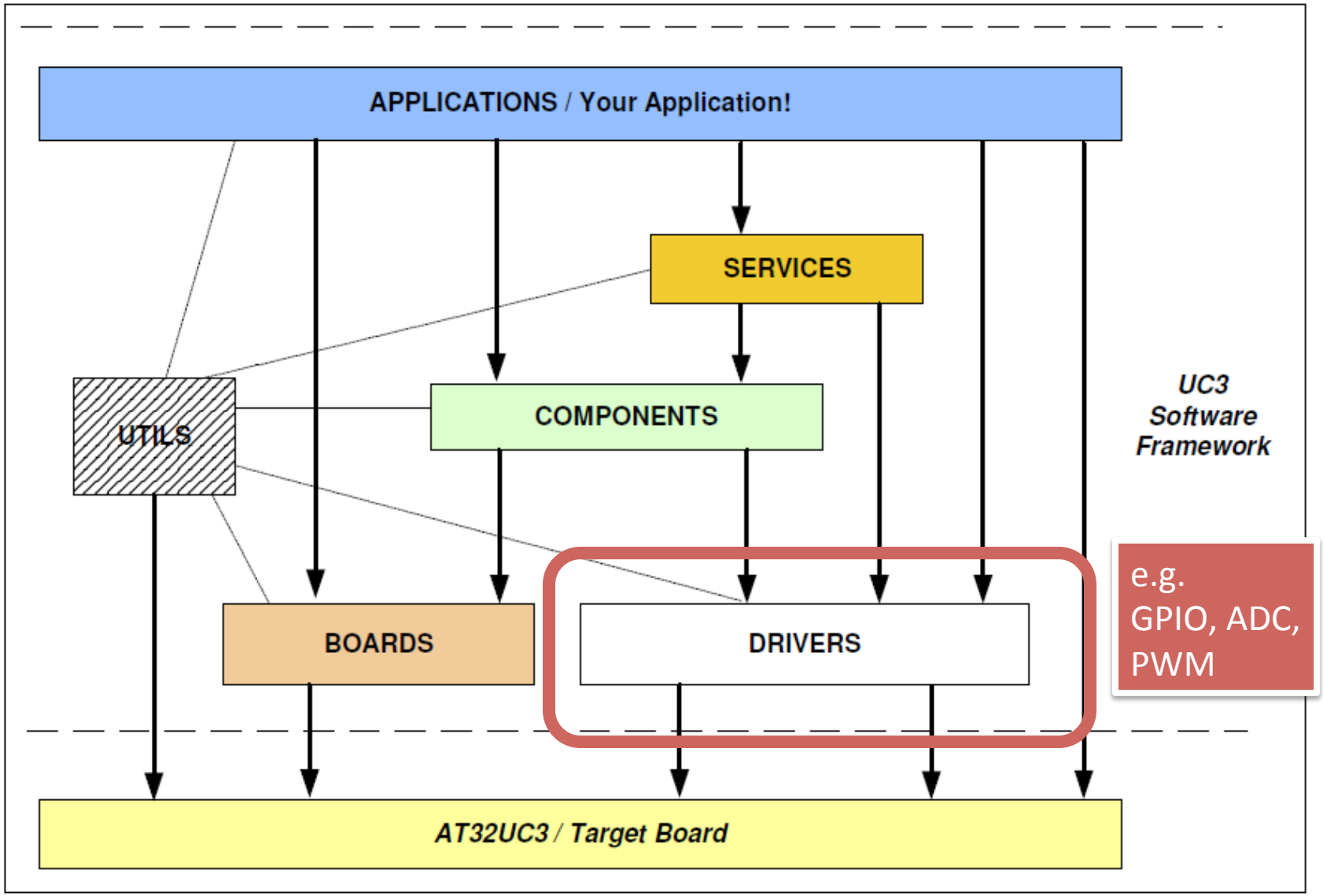
AVR32 Studio

- Eclipse IDE
- gcc ‘tool chain’
- Integrated DFU

IDE=Integrated Development Environment; DFU=Device Firmware Update; gcc=Gnu C-Compiler


AVR - usb-rgb-ldr/src/main.c - AVR32 Studio
File Edit Source Refactor Navigate Search Project Run Framework Window Help
AVR
Project Explorer
usb-rgb-ldr
Software Libraries
Binaries
Includes
boot
Debug
src
SOFTWARE_FRAMEW
conf_usb.h
device_task.h
enum_example.c
main.c
usb_descriptors.c
usb_descriptors.h
usb_specific_request.
usb_specific_request.
main.c
// RGG LED main.c
#include "compiler.h"
#include "preprocessor.h"
#include "pm.h"
#include "gpio.h"
#include "pwm.h"
//#include "print_funcs.h"
#include "intc.h"
#include "power
#include "conf_
#include "usb_t
#if USB_DEVICE_
#include "usb_d
#include "usb_d
#include "usb_s
#include "devic
#endif
// clock setup
Building Workspace
Building all...
Always run in background
Run in Background
Cancel
Details >>
Outline
cG
cB
R_PWM_PIN
R_PWM_FUNCTION
R_PWM_CHANNEL_ID
G_PWM_PIN
G_PWM_FUNCTION
G_PWM_CHANNEL_ID
B_PWM_PIN
B_PWM_FUNCTION
B_PWM_CHANNEL_ID
pwm_opt : pwm_opt_t
pwm_channel : avr32_pwm_channel_t[
channel_id : unsigned int[]
TTT
init_pwm() : void
set_rgb(U8, U8, U8) : void
ADC_CHANNEL
ADC_PIN
ADC_FUNCTION
init_adc() : void
get_adc_value() : U16
sof_cnt : U32
in_data_length : U8
in_buf : U8[]
out_data_length : U8
out_buf : U8[]
device_task_init(void) : void
device_task(void) : void
usb_sof_action(void) : void
main() : int
Problems
Properties
Console
C-Build [usb-rgb-ldr]
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/GPIO -I../src/SOFTWARE_FRAMEWORK/DRIVERS/FLASHC
-I../src/SOFTWARE_FRAMEWORK/UTILS/PREPROCESSOR -I../src/SOFTWARE_FRAMEWORK/UTILS
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/ADC -Wa,-g3
-osrc\SOFTWARE_FRAMEWORK\DRIVERS\INTC\exception.o
..\src\SOFTWARE_FRAMEWORK\DRIVERS\INTC\exception.x
avr32-gcc -I../src -I../src/SOFTWARE_FRAMEWORK/DRIVERS/INTC
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/USBB/ENUM/DEVICE
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/USBB/ENUM/HOST
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/USBB/ENUM -I../src/SOFTWARE_FRAMEWORK/DRIVERS/USBB
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/PWM -I../src/SOFTWARE_FRAMEWORK/DRIVERS/PM
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/GPIO -I../src/SOFTWARE_FRAMEWORK/DRIVERS/FLASHC
-I../src/SOFTWARE_FRAMEWORK/UTILS/PREPROCESSOR -I../src/SOFTWARE_FRAMEWORK/UTILS
-I../src/SOFTWARE_FRAMEWORK/DRIVERS/ADC -I../src/SOFTWARE_FRAMEWORK/SERVICES/USB -O0 -g3
-Wall -c -fmessage-length=0 -mpart=uc3b1256 -ffunction-sections -masm-addr-pseudos
-osrc\SOFTWARE_FRAMEWORK\DRIVERS\INTC\intc.o ..\src\SOFTWARE_FRAMEWORK\DRIVERS\INTC\intc.c
AVR Targets
Name
Adapter
Board
AVR32 Simulat...
AVR32 Simulat...
AVR32
DFU
USB DFU
Unspec
Writable
Smart Insert
20 : 17
Building Workspace: (35%)

AVR32 Software Framework



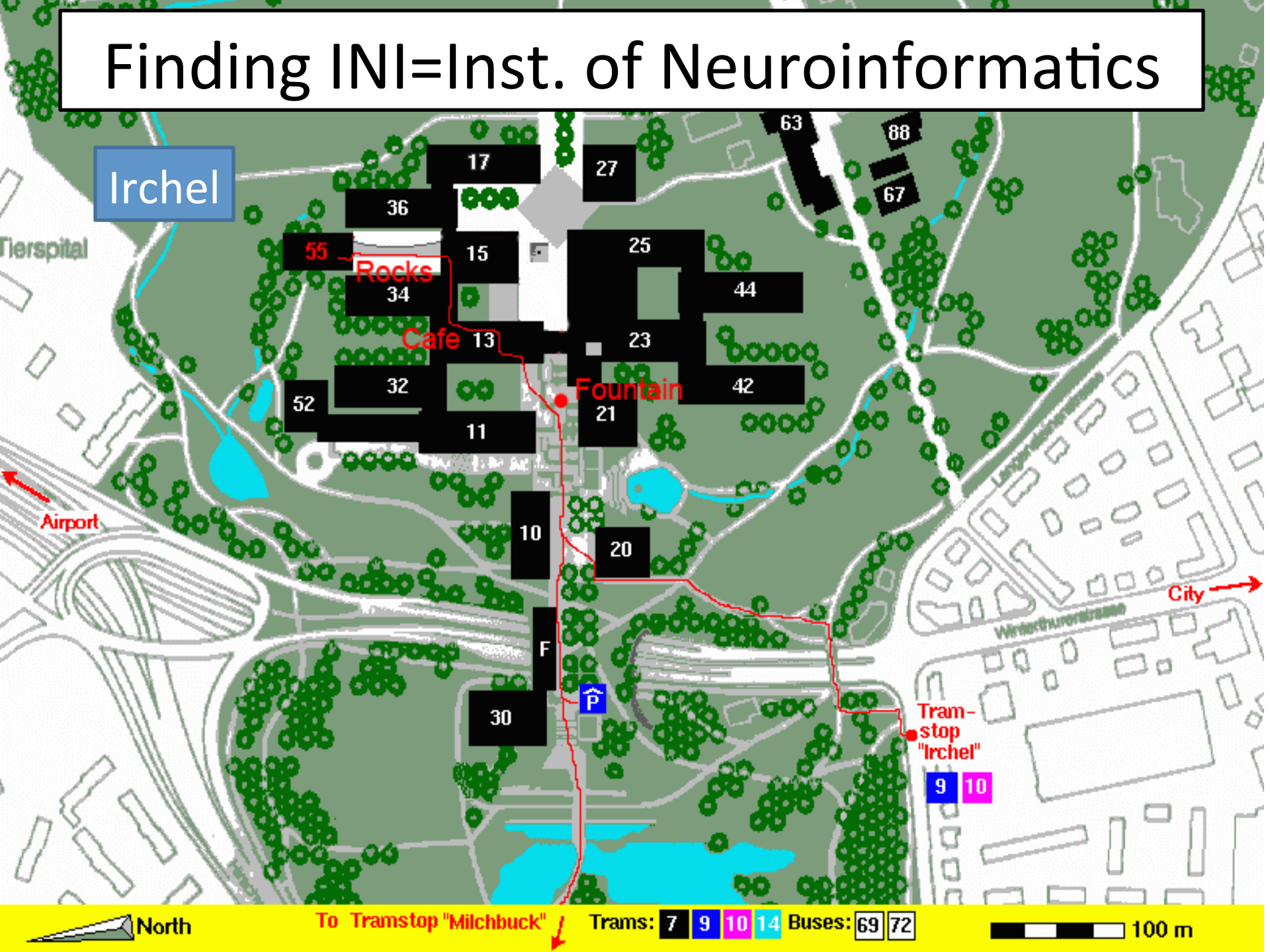
Exercise 1

- Installing and running AVR32 tools.
- Importing Daniel's usb-rgb-ldr* example project.
- Installing libusb USB driver and 'copper' board.
- Programming sample copper board with usb-rgb-ldr.
- Installing python and pyusb.
- Running usb-rgb-ldr on copper board with LDR and RGB LEDs added.
- Arranging time for soldering your own copper or 'bronze' board.
- Exercise 2: probably debugging and repairing your own copper board, adding RGB LED and LDR

* LDR=Light dependent resistor, RGB LED=Red/Green/Blue Light Emitting Diode

Surface mount soldering exercise

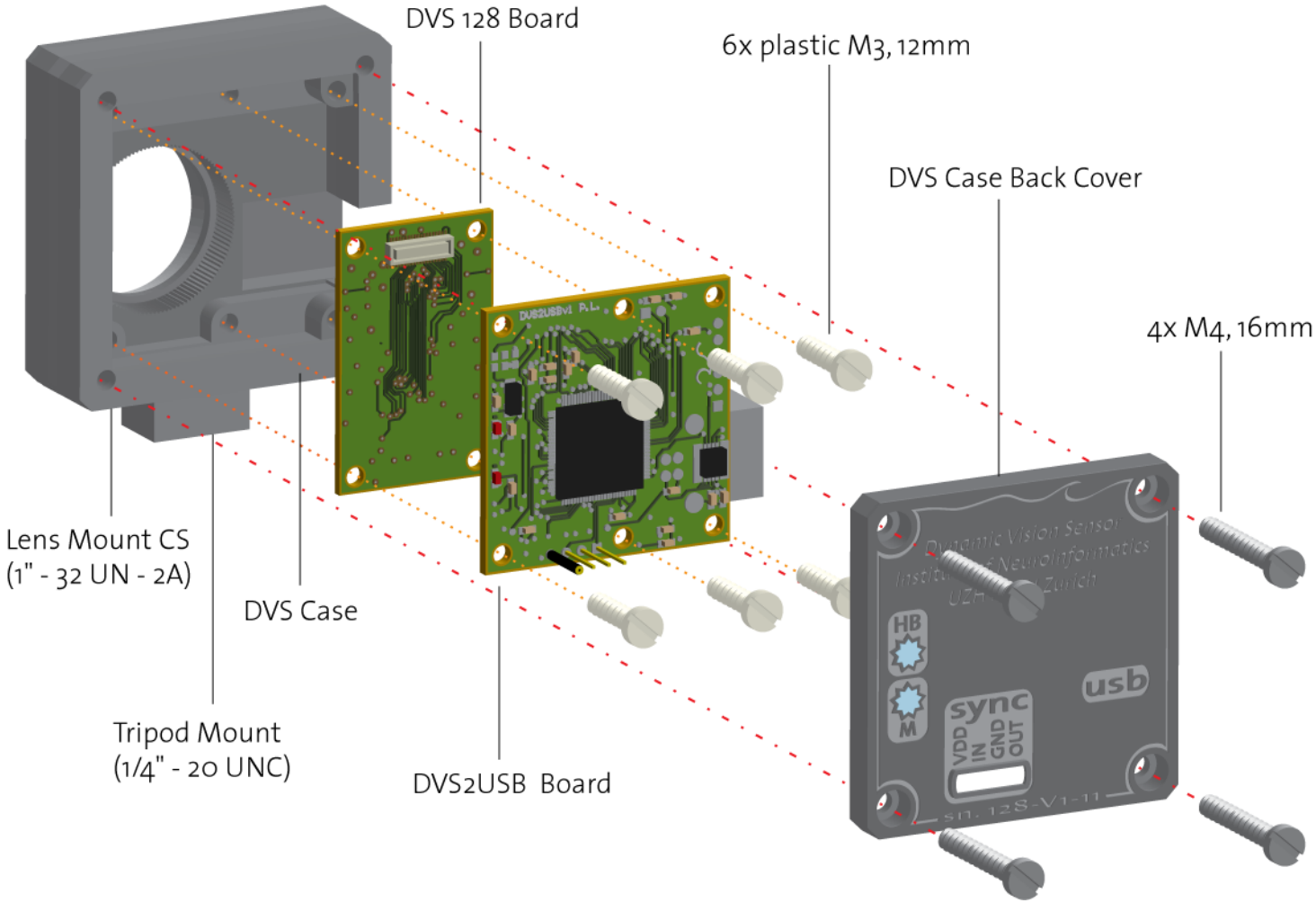
Finding INI=Inst. of Neuroinformatics

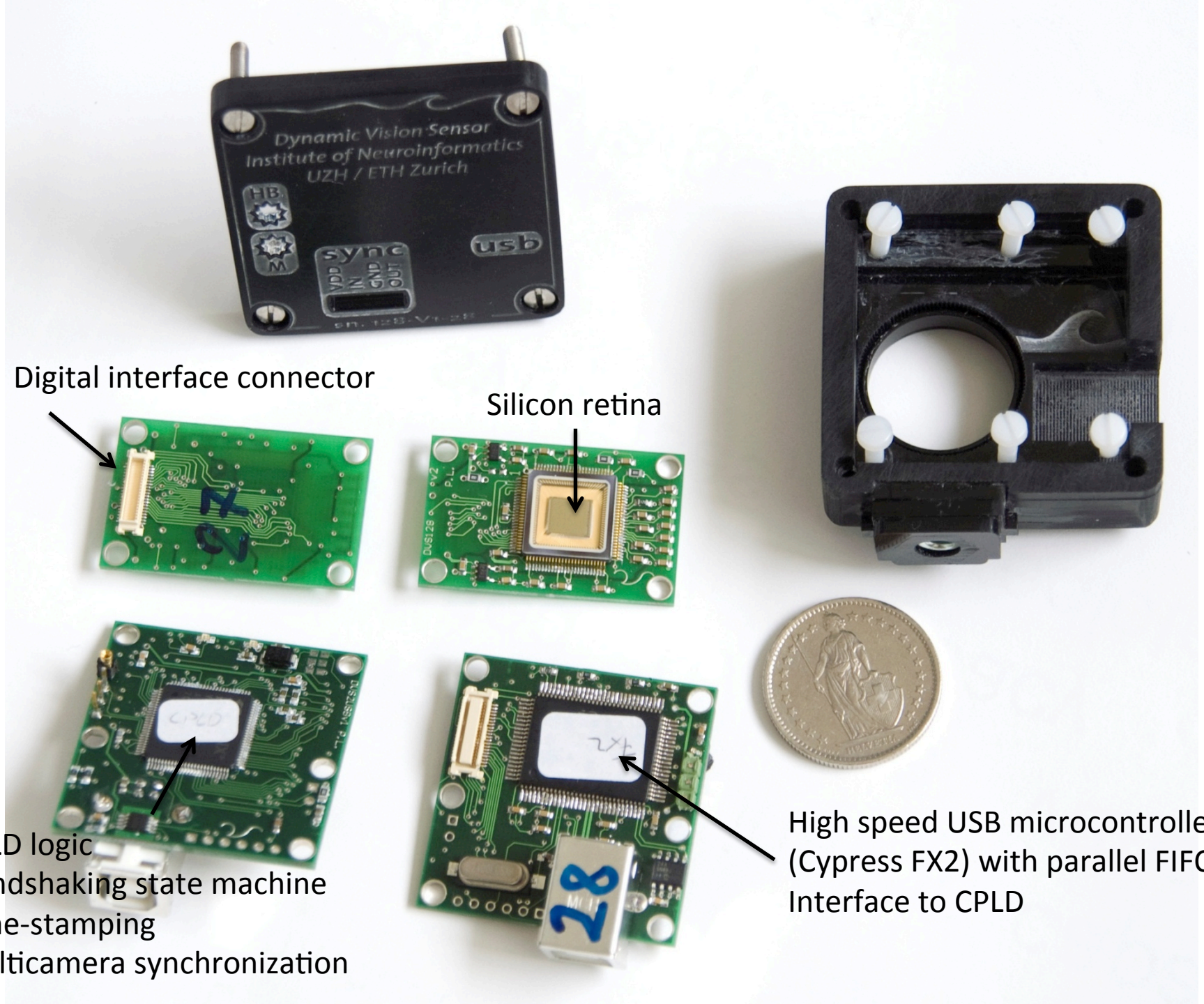



Irchel
17
27
36
55
15
25
Rocks
34
Cafe
13
23
32
Fountain
52
21
11
10
20

USB silicon retina

Camera PCB acquires asynchronous digital data from retina chip, time stamps it, sends data over high speed USB interface to PC. It also receives control commands for biasing, etc. from PC and sends to retina chip serially.


DVS 128 Board
6x plastic M3, 12mm
DVS Case Back Cover
4x M4, 16mm
Lens Mount CS
(1" - 32 UN - 2A)
DVS Case
Tripod Mount
(1/4" - 20 UNC)
DVS2USB Board
HB
M
sync
VDD
IN
GND
OUT
usb


Dynamic Vision Sensor
Institute of Neuroinformatics
UZH / ETH Zurich
HB
M
sync
usb
VDD
IN
GND
OUT
Digital interface connector
Silicon retina
CPLD logic
Handshaking state machine
Time-stamping
Multicamera synchronization
High speed USB microcontroller
(Cypress FX2) with parallel FIFO
Interface to CPLD