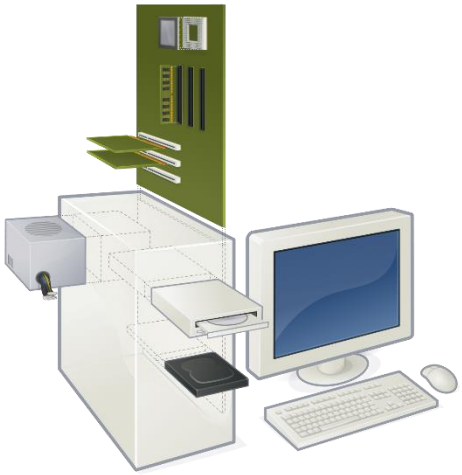
به نام خدا

برنامه‌ریزی تراشه AVR

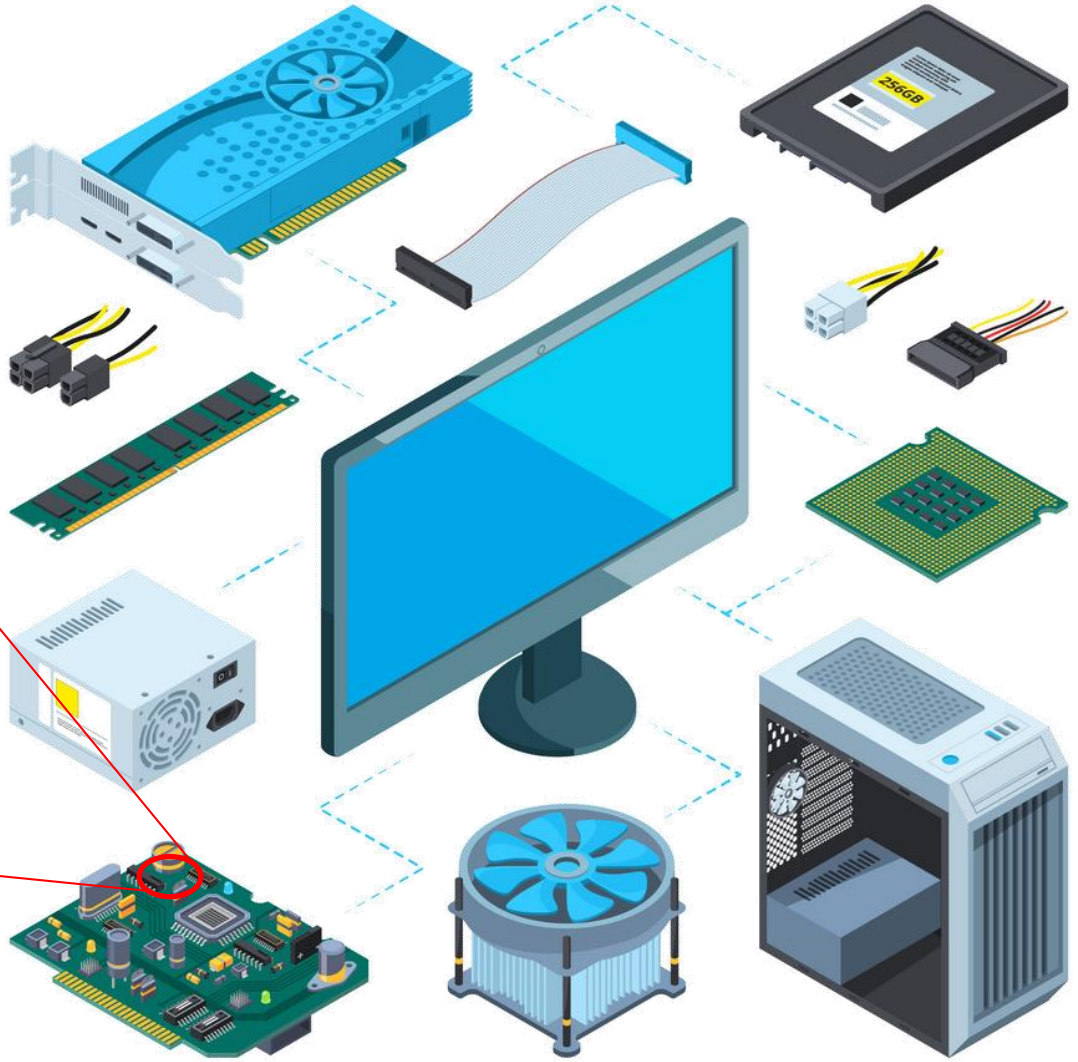
معرفی ابزارهای کاربردی کار با تراشه‌های AVR (Toolchain)

Dr. Aref Karimiafshar
A.karimiafshar@ec.iut.ac.ir

کامپیوترهای آماده به کار



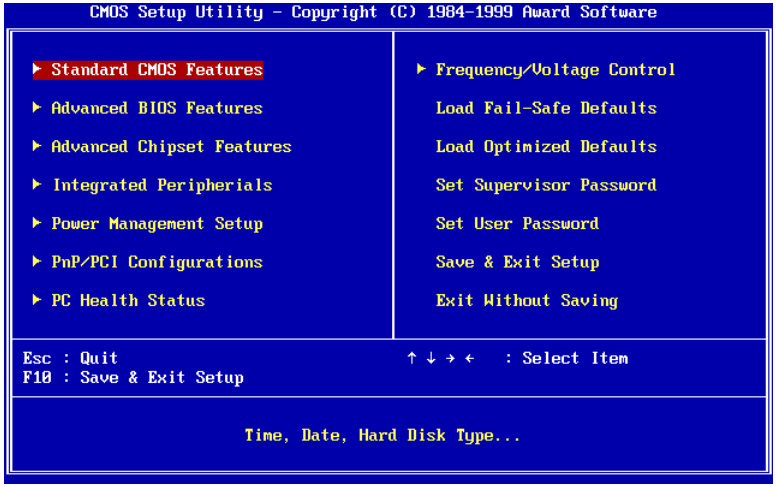
سرهم کردن قطعات



--

راه انداز سیتسم

- BIOS (basic input/output)
  - Motherboard firmware
    - Software which runs at a lower level than operating system
  - Tells the computer
    - What drive to boot from
    - How much RAM you have
    - Controls other key details like CPU frequency

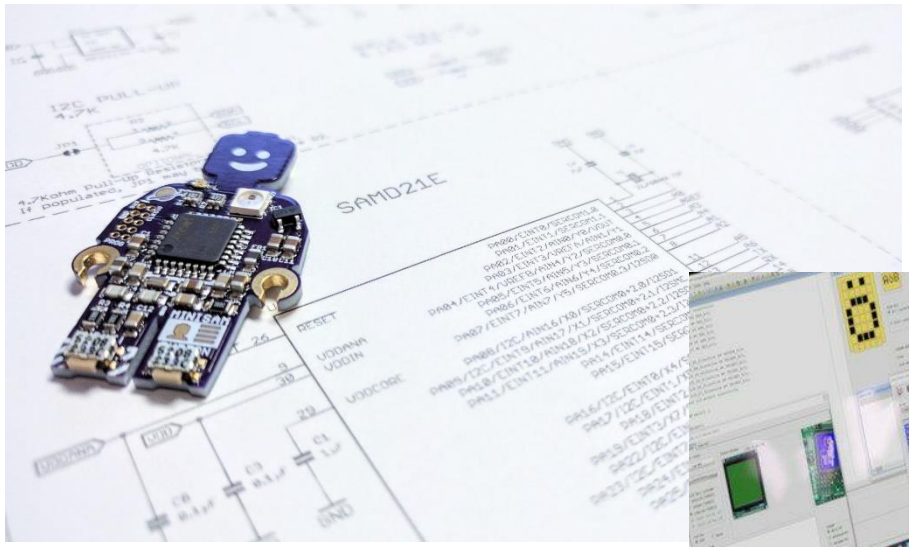


-
بایاس تقریبا راه اندازی سیستم است ینی یک نرم افزاری داریم که یک مقدار سطح پایین تر از سیستم عامل عمل میکنه
به این برنامه سطح پایین firmware می گن که توی یک رام قرار داره که تحت عنوان بایاس می شناسیمش

میکروکنترلر

توی میکروکنترلر با کامپیوتری در ارتباط هستیم که این خام خام است ینی یک کامپیوتر داریم که برنامه بایاس هم ندارد

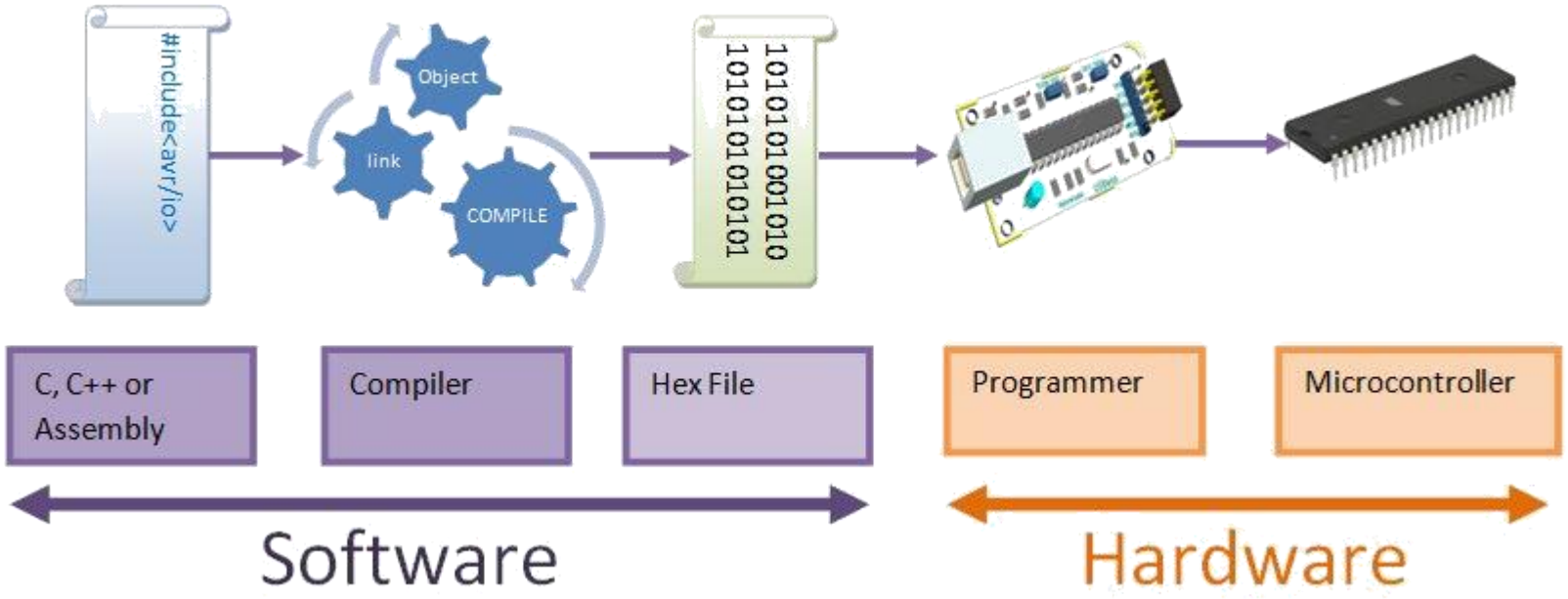
- میکروکنترلر
  - خالی از هر گونه برنامه
  - نیازمند برنامه‌ریزی



میکروکنترلر

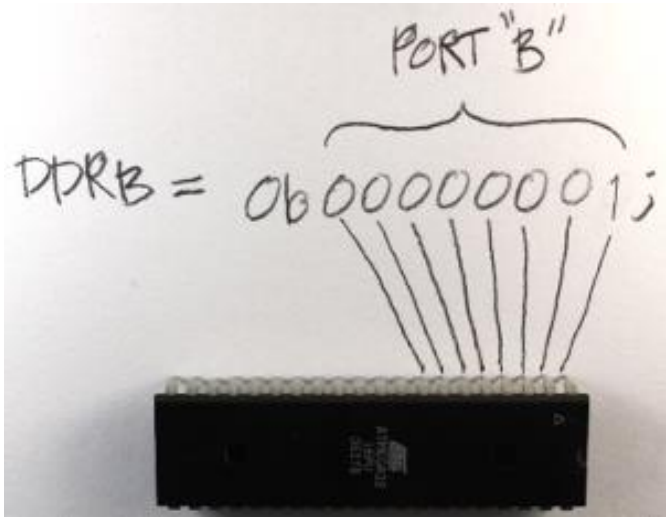
برای اینکه بتونیم یک میکروکنترلر را برنامه ریزی و قابل استفاده بکنیم
نیازمند به دو دسته ابزار داریم
ابزارهای نرم افزاری مثل تکست ادیتور یا ...
ابزارهای فیزیکی توی این ابزار نیاز به یک پروگرمر داریم

- برنامه‌ریزی
  - ابزارهای فیزیکی
  - نرم افزارهای مدیریت و برنامه‌ریزی



Program & Data

- Program
  - a sequence of instructions specifying how the data is to be processed.
- Data
  - input to the program, either supplied during runtime or pre-stored in the computer




```
DATA_BUFFER          EQU          0x20
DATA_BUFFER2         EQU          0x21
CMD_BUFFER           EQU          0x22
REF_BUFFER           EQU          0x24

ADDR_INDEX           EQU          0x25                ;STARTING ADDRESS IN EEPROM
ADDR_L               EQU          0x26                ;STARTING ADDRESS L
ADDR_H               EQU          0x27                ;STARTING ADDRESS H
COUNTER_INDEX        EQU          0x29                ;COUNTER
BIT_INDEX            EQU          0x2A                ;BIT INDEX

CMD_SET_CURSOR EQU   21H          ;SET CURSOR
CMD_TXHOME           EQU    40H    ;SET TXT HM ADD
CMD_TXAREA           EQU    41H    ;SET TXT AREA
CMD_GRHOME           EQU    42H    ;SET GR HM ADD
CMD_GRAREA           EQU    43H    ;SET GR AREA
CMD_OFFSET           EQU    22H    ;SET OFFSET ADD
CMD_ADPSET           EQU    24H    ;SET ADD PTR
CMD_SETDATA_INC      EQU    0C0H   ;WRITE DATA AND INCREASE ADP
CMD_AWRON            EQU    0B0H   ;SET AUTO WRITE MODE
CMD_AWROFF           EQU    0B2H   ;RESET AUTO WRITE MODE
```

-

program

مجموعه دستورات را میگیم برنامه ینی یک مجموعه از دستورات مشخص می کنند داده هایی که قراره پردازش بشن یا ... چه کنترلی داشته باشند

data

در کنار برنامه یکسری داده هایی رو هم داریم که این ها همزمان با اون برنامه اصلی ما باید برن و توی حافظه اصلی قرار بگیرند و این ها معمولا داده هایی هستند که به عنوان ورودی های اولیه توی برنامه ما مورد استفاده قرار میگیرند

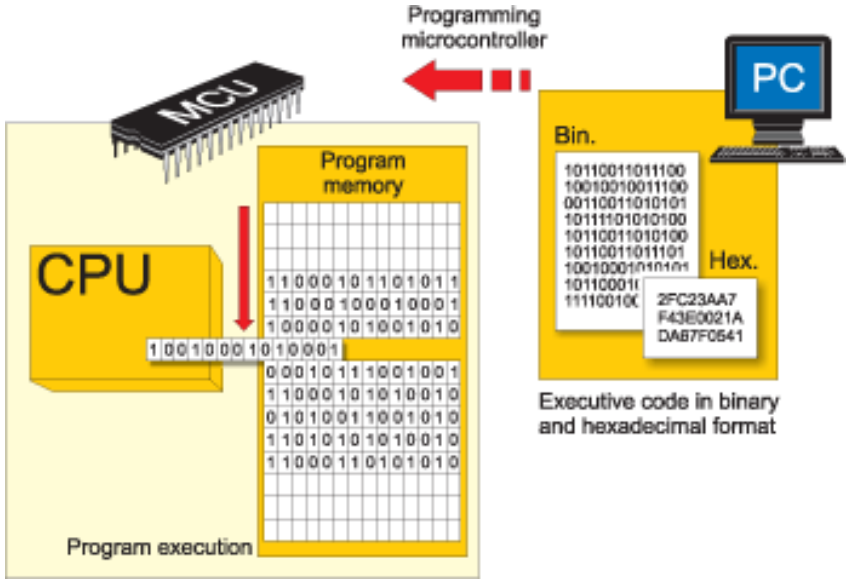
Program

- The microcontroller executes user program loaded in its memory.
- In the case of Harvard architecture
  - program memory is usually of ROM type (PROM, OTP ROM, EPROM, Flash…)
- Program in ROM memory is called executable code, or firmware
- Executable code is sequence of zeros and ones – binary code
- It is organized in 8, 12, 14, 16, 24, 32-bit wide words
  - depending on the microcontroller's architecture.
- Every word is considered by the CPU as a command being executed during the operation of the microcontroller.
- Set of commands understood by CPU is called machine language
- Machine language is CPUs native language

-
پس ما قراره یک برنامه یا Program را در میکروکنترلر بریزیم و این برنامه مشخص میکنه که ما می خوایم این میکروکنترلر چه کاری برای ما انجام بده و برنامه می ره و توی حافظه میکروکنترلر قرار می گیره
این برنامه می ره توی رام قرار می گیره که ما اصطلاحا به این میگیم Executable code ینی اون کدی که اجرا میشه و قابل اجرا است و اون عملیات مارو محقق میکنه و این Executable code ما از یک مجموعه صفر و یک ها تشکیل شده که بهش binary code هم میگن و این binary code ما در قالب یکسری کلماتی که اندازش 8 یا 12 یا 14 یا 16 یا 24 یا 32 تنظیم بشه و هر کدوم از این اندازه کلمات وابسته به اون معماری که ما واسه اون میکروکنترلر داریم این کلمات یکسری دستوراتی رو توی دلش خودشون دارند
این پردازشگر این میکروکنترلر میاد این کلمات رو اجرا میکنه که به این دستوراتی که توی این کلمات قرار میگیره machine language این زبان ماشین وابسته باست به cpu ینی از یک cpu به یک cpu دیگر کاملا متفاوته

Program

- For practical reasons, as it is much easier for us to deal with hexadecimal number system, the executable code is often represented as a sequence of hexadecimal numbers called a Hex code.
- Executable code can be written directly using text editor, and stored into the Flash memory by appropriate software tool – flash programmer



-
هر چند که ما به این میگیم binary code ولی بازم بخاطر جنبه های عملیاتی این هارو به عنوان هگز کد هم می شناسن
ما می تونیم این هگز کد یا باینری کد رو مستقیما توی تکست ادیتور بنویسیم

Programming Languages (PL)

```
27bdffd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

Can You Understand This?

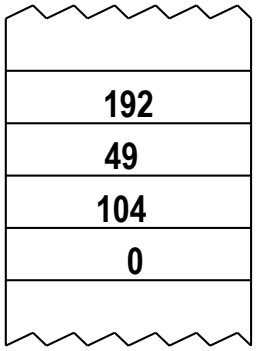
```
    addiu   sp,sp,-32
    sw      ra,20(sp)
    jal     getint
    nop
    jal     getint
    sw      v0,28(sp)
    lw      a0,28(sp)
    move    v1,v0
    beq     a0,v0,D
    slt     at,v1,a0
A:  beq     at,zero,B
    nop

    b       C
    subu    a0,a0,v1
B:  subu    v1,v1,a0
C:  bne     a0,v1,A
    slt     at,v1,a0
D:  jal     putint
    nop
    lw      ra,20(sp)
    addiu   sp,sp,32
    jr      ra
    move    v0,zero
```

Is This Better?

ولی اینطوری اگر بخوایم بنویسیم خیلی قابل فهمیدن نیست

PL: Machine Languages

- A CPU only understands its own machine language => *portability* problem
  - In Motorola 68000, the following 4 bytes is an instruction to move the value in register D3 to memory address 104.



  - In Intel 80486, the same sequence of bytes represents different instructions
- Coding is tedious and error prone

-

وقتی از زبان ماشین استفاده میکنیم اون قابلیت جابه جا پذیری رو نداره ینی وقتی بخوایم از یک پردازنده به یک پردازنده دیگه بریم این برای ما خیلی مشکل ساز است

PL: Assembly Languages

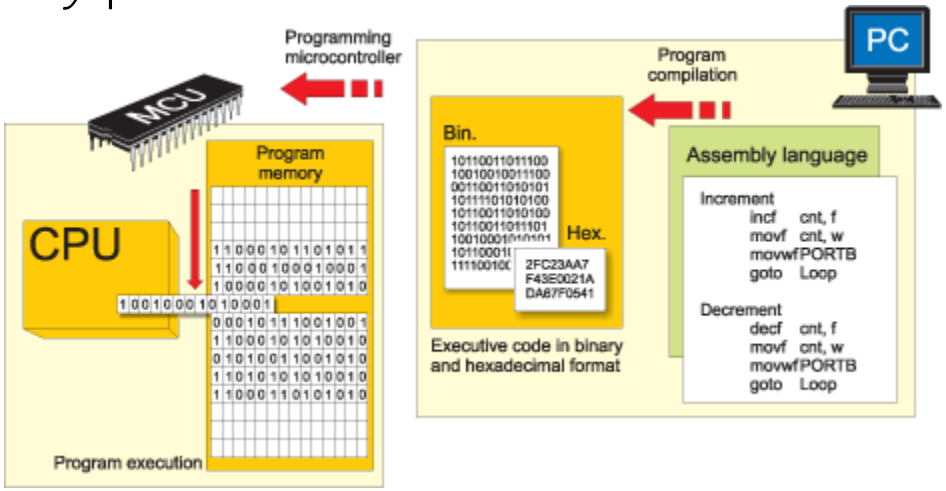
- As we know process of writing executable code using machine language is very difficult
- Binary/Hex words from machine language vocabulary can be represented in the form of meaningful abbreviations
- These abbreviations are called assembly instructions
- Set of assembly instructions is called assembly language.
- Programs written using assembly instructions, in order to be executed by microcontroller, should be translated (compiled) into executable code
- Compiling can be performed using special PC tool called Assembler

PL: Assembly Languages

- Use English-like abbreviations
  - e.g. the MC68000 machine instruction is written as:

    MOVE D3, 104
  - Slightly easier for human to understand
  - Different CPUs have different instruction sets
    - e.g. MOVE D3, 104 is not a valid instruction in Intel 80486 because it doesn't even have a register called D3

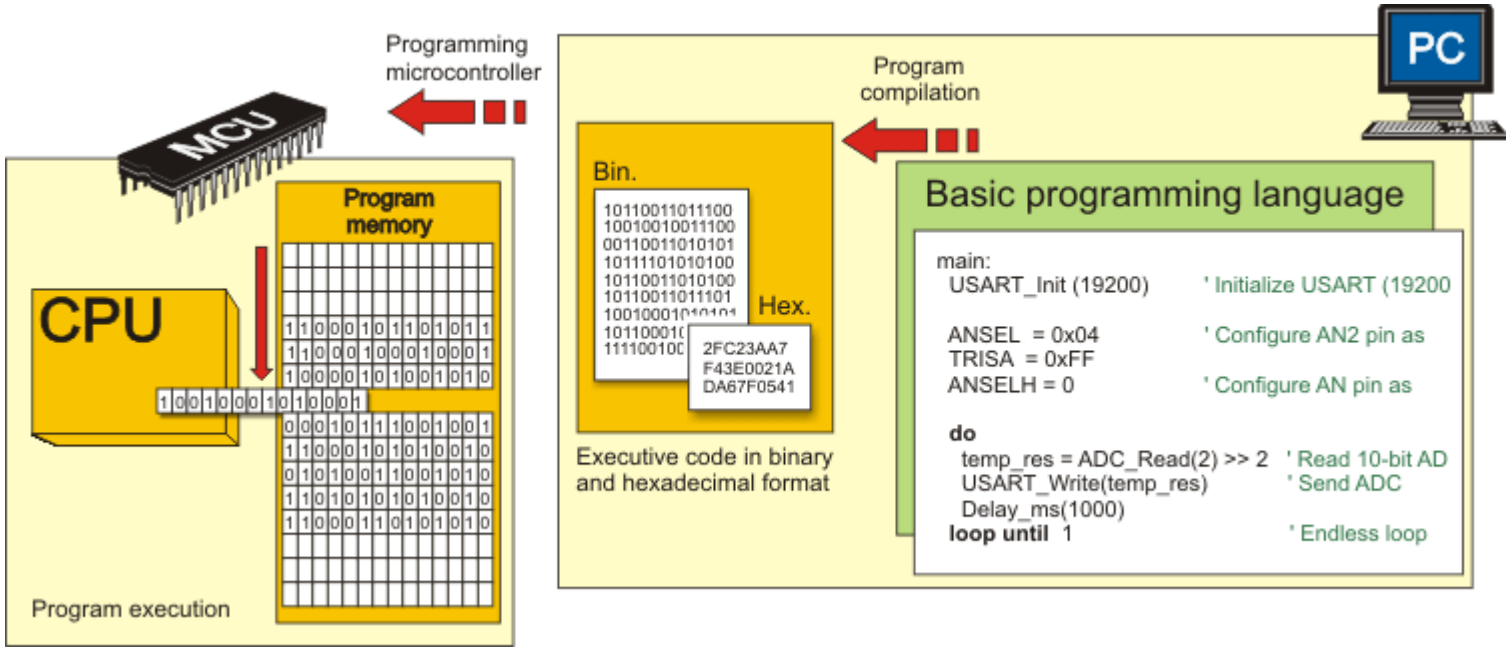
  => *portability* problem



PL: High-level Languages

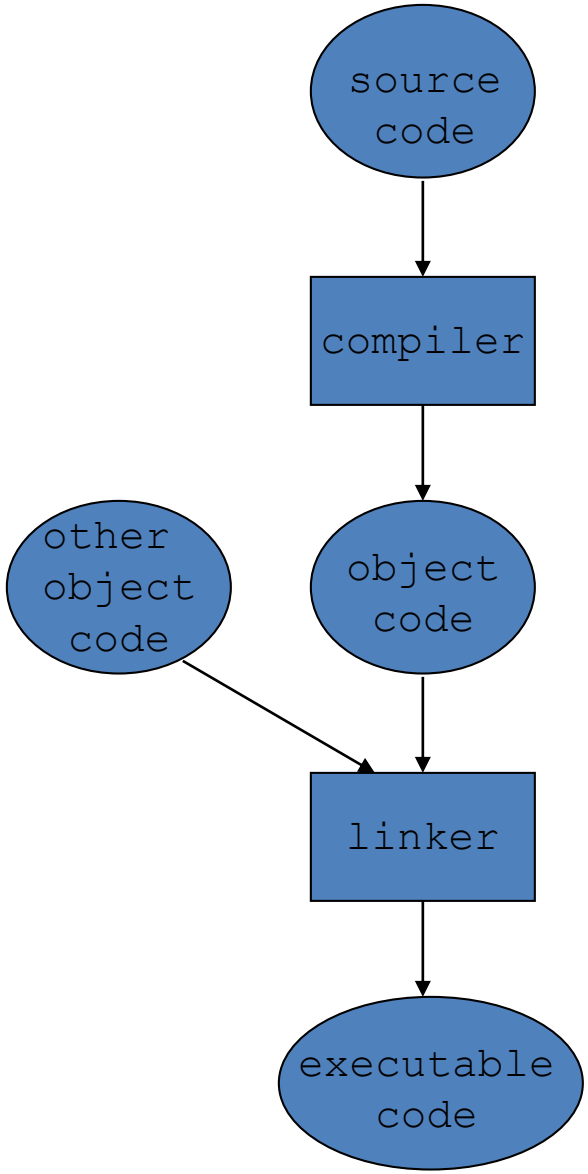
- Close(r) to human language
- One single statement accomplishes substantial tasks
- Need a compiler/linker
  - to translate into machine language
- More portable
  - the same program (more or less) works for different machines
- e.g.
  - Fortran, COBOL, Basic, Pascal, Ada, Modula, C, C++, Lisp, Prolog, Java, Perl...

PL: High-level Languages

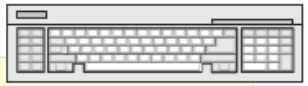


PL: Compiler & Linker

- Compiler
  - Translates a program in high-level language into an *object program* (or *object code*). The original program is called the *source program*/*code*.
- Linker
  - Combines the object code of a program with other pre-compiled object codes to generate an *executable code* which is in machine language



From source code to Hex file and programming



```
program test
main:
   TRISA = 0x00          ' Configure PORTA pins as outputs
     PORTA = 0xFF        ' Turn PORTA LEDs ON
end.
```

Program written in Basic

Compiling program into assembly code

```
' ADDRESS     OPCODE        ASM
' ------------------------------------------
                           GOTO 3
0x0000        0x2803
main:
TRISA  = 0x00     ' Configure PORTA pins as outputs
0x0003        0x1683        BSF         STATUS, 5
0x0004        0x1303        BCF         STATUS, 6
0x0005        0x0185        CLRF        TRISA
PORTA = 0xFF      ' Turn PORTA LEDs ON
0x0006        0x30FF        MOVLW       255
0x0007        0x1283        BCF         STATUS, 5
0x0008        0x0085        MOVWF       PORTA
0x0009        0x2809        GOTO        $+0
```

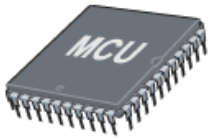
Compiled program

Compiling program into a hex code

```
:020000000328D3
:0E000600831603138501FF308312850009283D
:04400E00F22F000786
:00000001FF
```

Executable code of the program (hex code)

Loading program into the microcontroller



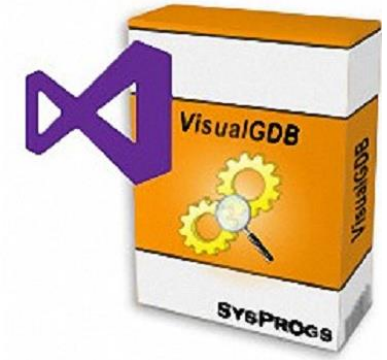
Toolchain

- A collection of tools/libraries
  - To create applications for AVR microcontrollers
    - Compiler
    - Assembler
    - Linker
    - Standard C library (AVR-libc)

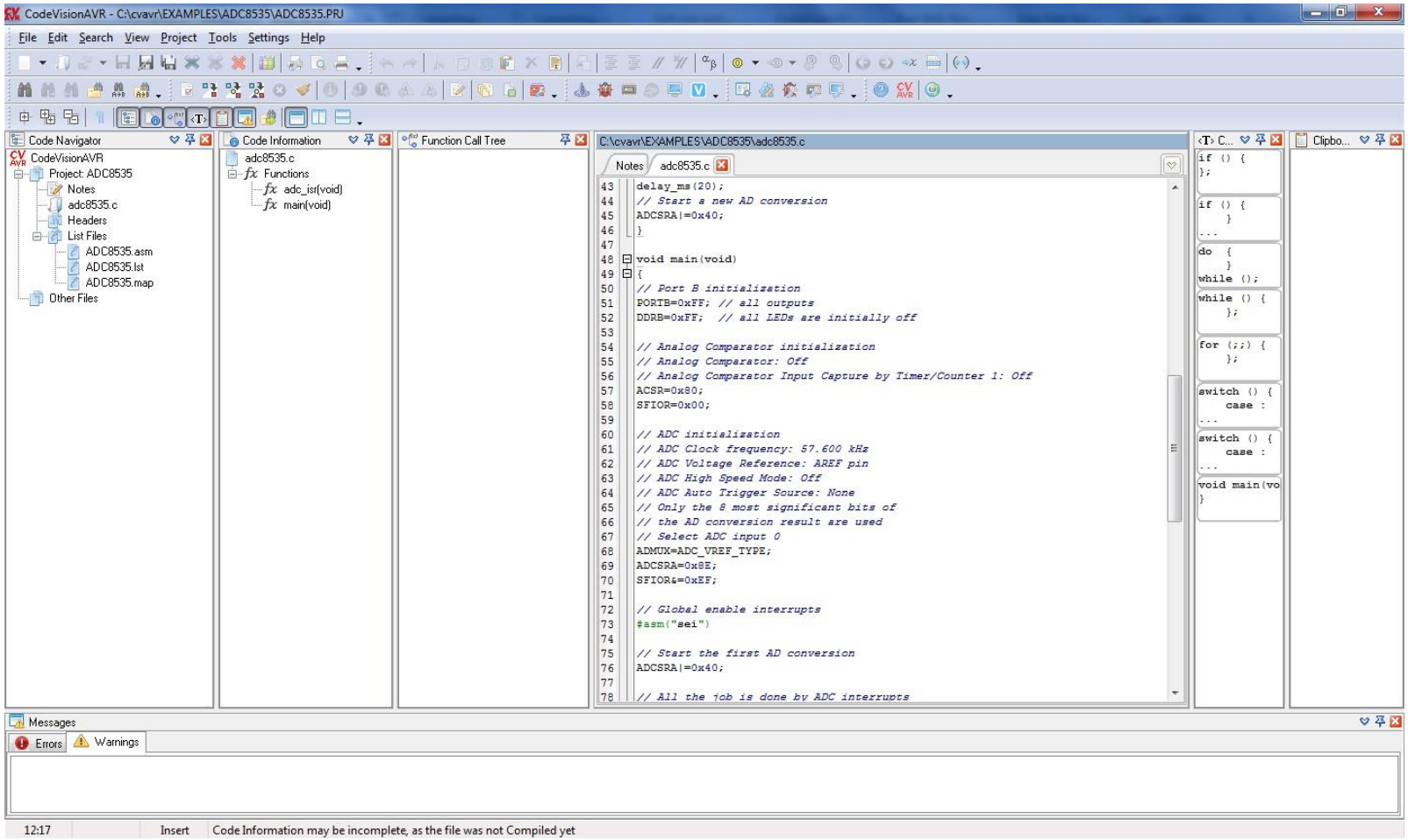
- For AVR
  - There are free software tools available
  - Even some of it is open-source!

IDE

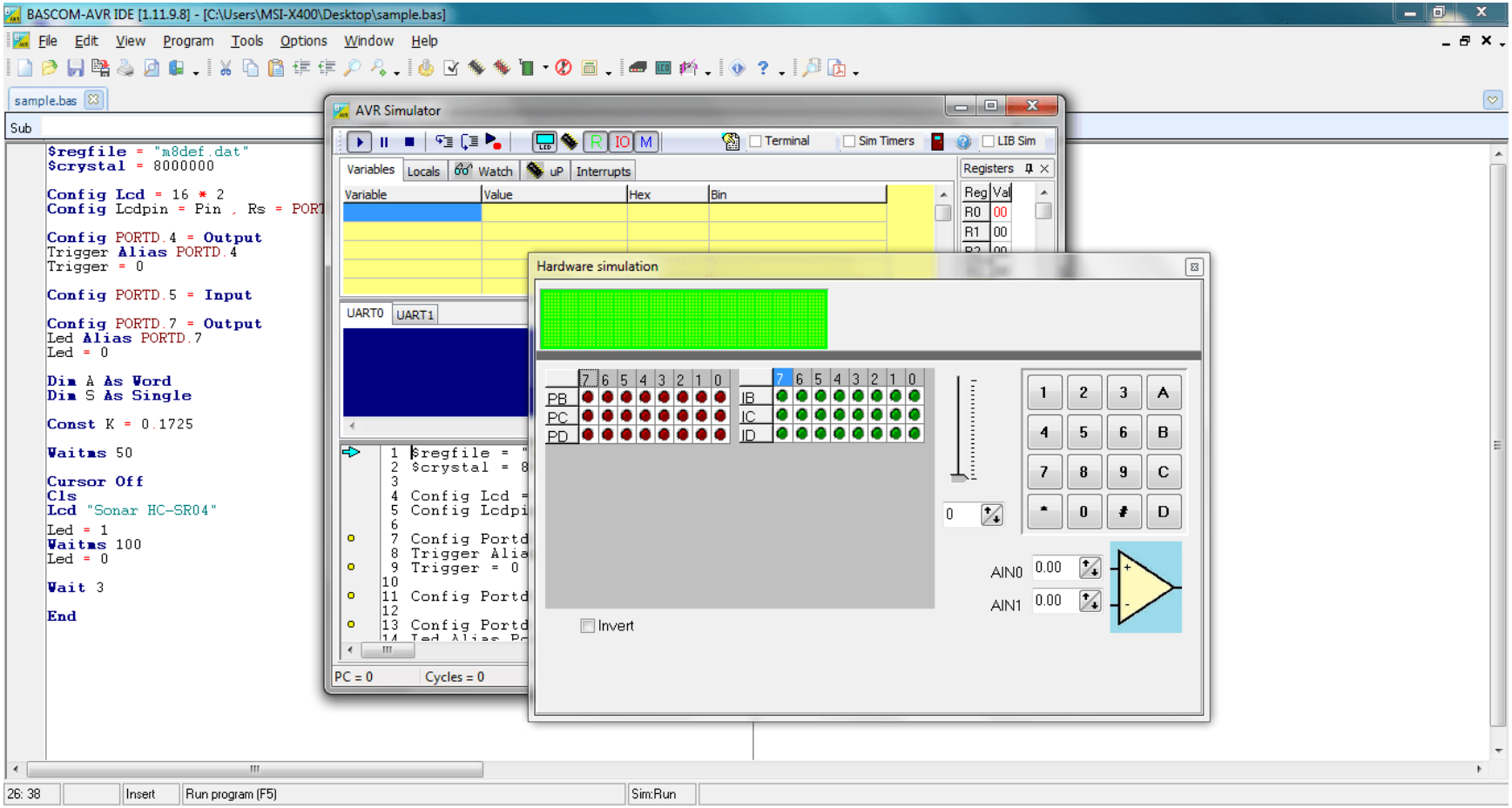
- Integrated development environment (IDE)
  - CodeVisionAVR
  - BASCOM-AVR
  - WinAVR
  - Visual Studio
  - Eclipse (AVR Eclipse Plugin)
  - Atmel Studio (Microchip Studio)



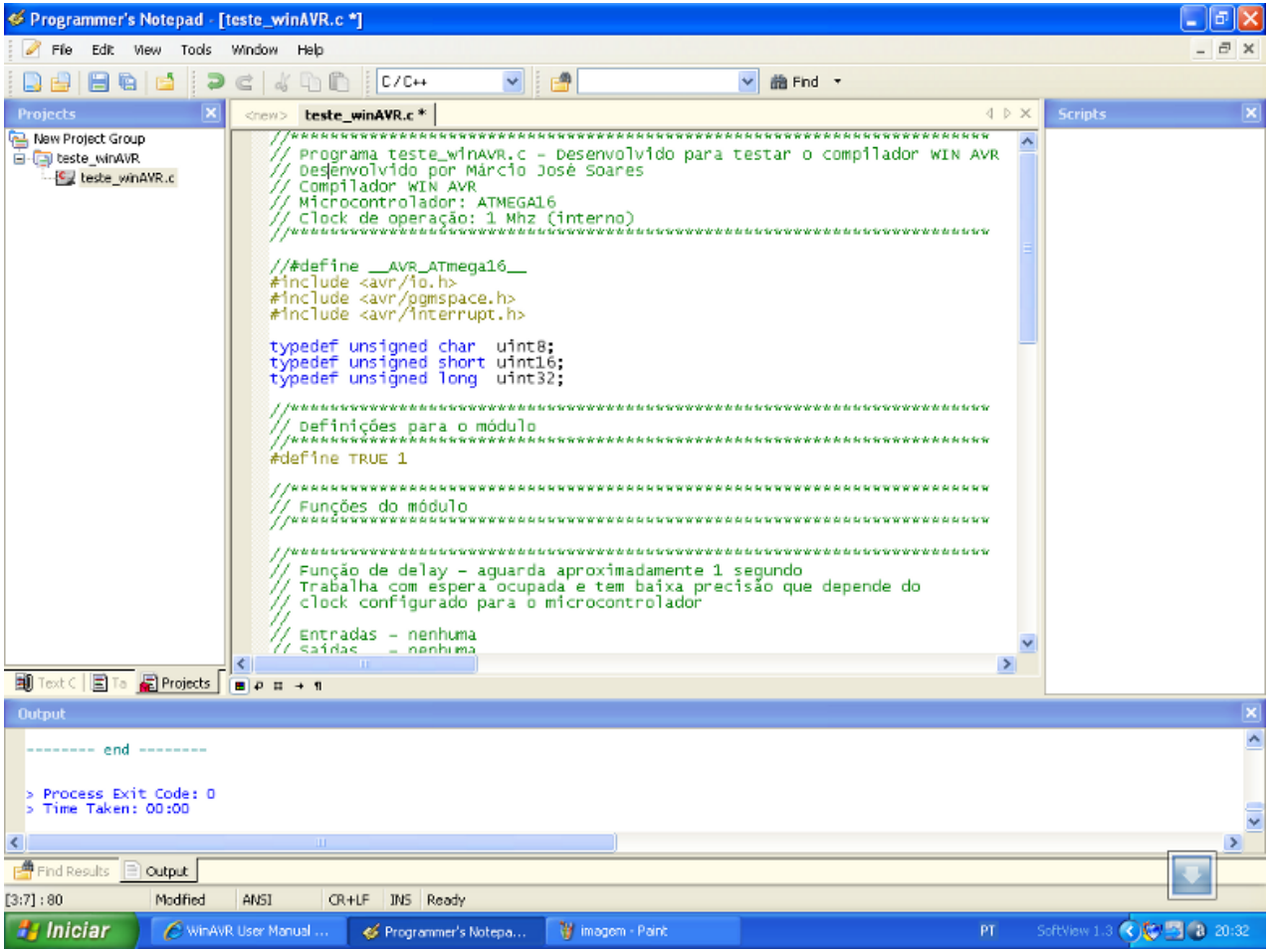
CodeVisionAVR



BASCOM-AVR


BASCOM-AVR IDE [1.11.9.8] - [C:\Users\MSI-X400\Desktop\sample.bas]
File Edit View Program Tools Options Window Help
sample.bas
Sub
$regfile = "m8def.dat"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Rs = PORT
Config PORTD.4 = Output
Trigger Alias PORTD.4
Trigger = 0
Config PORTD.5 = Input
Config PORTD.7 = Output
Led Alias PORTD.7
Led = 0
Dim A As Word
Dim S As Single
Const K = 0.1725
Waitms 50
Cursor Off
Cls
Lcd "Sonar HC-SR04"
Led = 1
Waitms 100
Led = 0
Wait 3
End
AVR Simulator
Terminal
Sim Timers
LIB Sim
Variables
Locals
Watch
uP
Interrupts
Variable
Value
Hex
Bin
Registers
Reg
Val
R0
R1
UART0
UART1
Hardware simulation
PB
PC
PD
IB
IC
ID
AIN0
AIN1
0.00
Invert
PC = 0
Cycles = 0
26: 38
Insert
Run program (F5)
Sim:Run

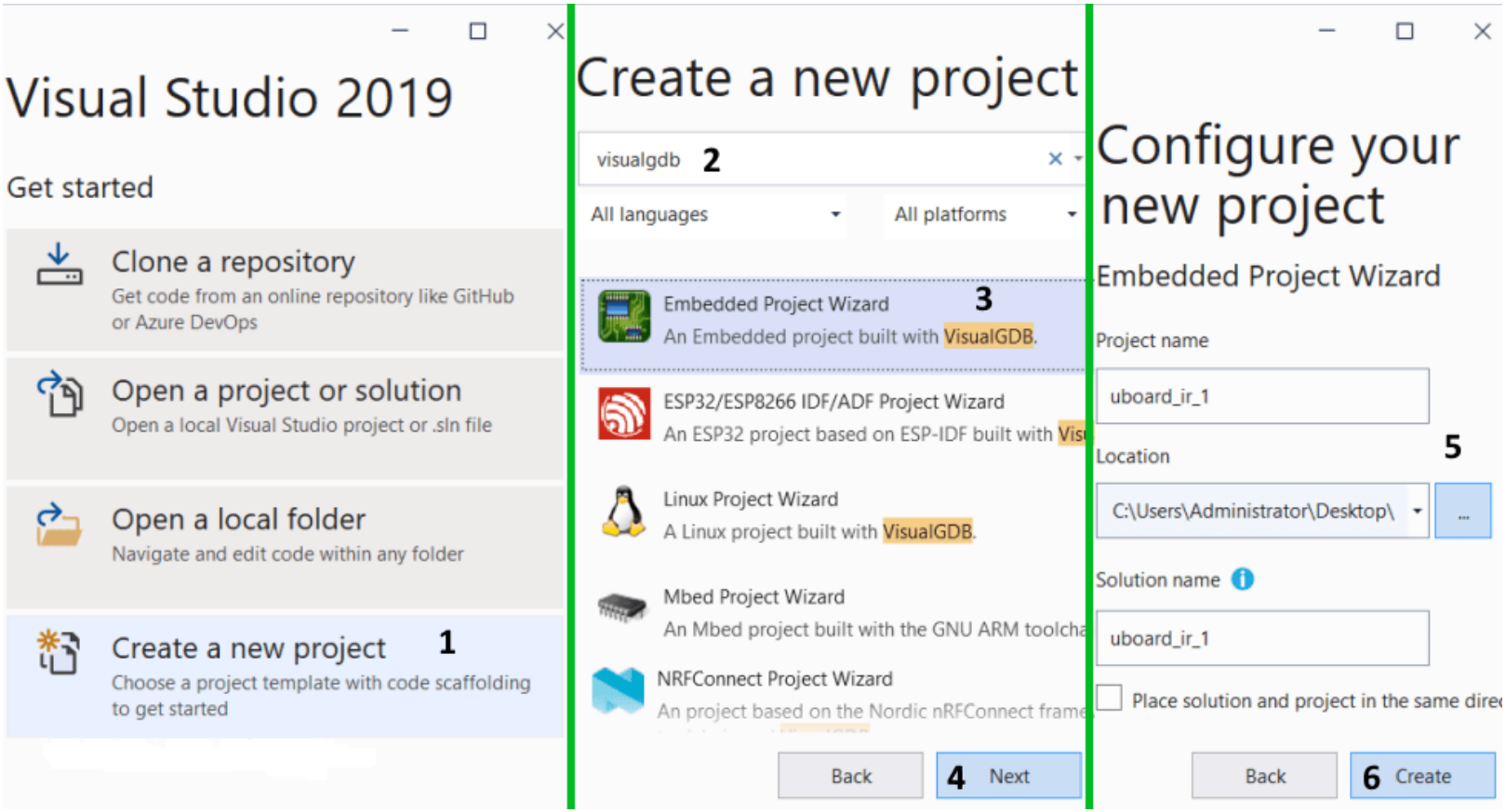
WinAVR


Programmer's Notepad - [teste_winAVR.c *]
File Edit View Tools Window Help
C / C++
Find
Projects
New Project Group
teste_winAVR
teste_winAVR.c
<new> teste_winAVR.c *
Scripts
//Programa teste_winAVR.c - Desenvolvido para testar o compilador WIN AVR
// Desenvolvido por Márcio José Soares
// Compilador WIN AVR
// Microcontrolador: ATMEGA16
// Clock de operação: 1 Mhz (interno)
//#define __AVR_ATmega16__
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
typedef unsigned char uint8;
typedef unsigned short uint16;
typedef unsigned long uint32;
// Definições para o módulo
#define TRUE 1
// Funções do módulo
// Função de delay - aguarda aproximadamente 1 segundo
// Trabalha com espera ocupada e tem baixa precisão que depende do
// clock configurado para o microcontrolador
// Entradas - nenhuma
Text C
Projects
Output
-------- end --------
> Process Exit Code: 0
> Time Taken: 00:00
Find Results
Output
[3:7] : 80
Modified
ANSI
CR+LF
INS
Ready
Iniciar
WinAVR User Manual ...
Programmer's Notepa...
Imagem - Paint
PT
20:32

BASCOM vs. CodeVision

Visual Studio

- افزونه‌ی VisualGDB



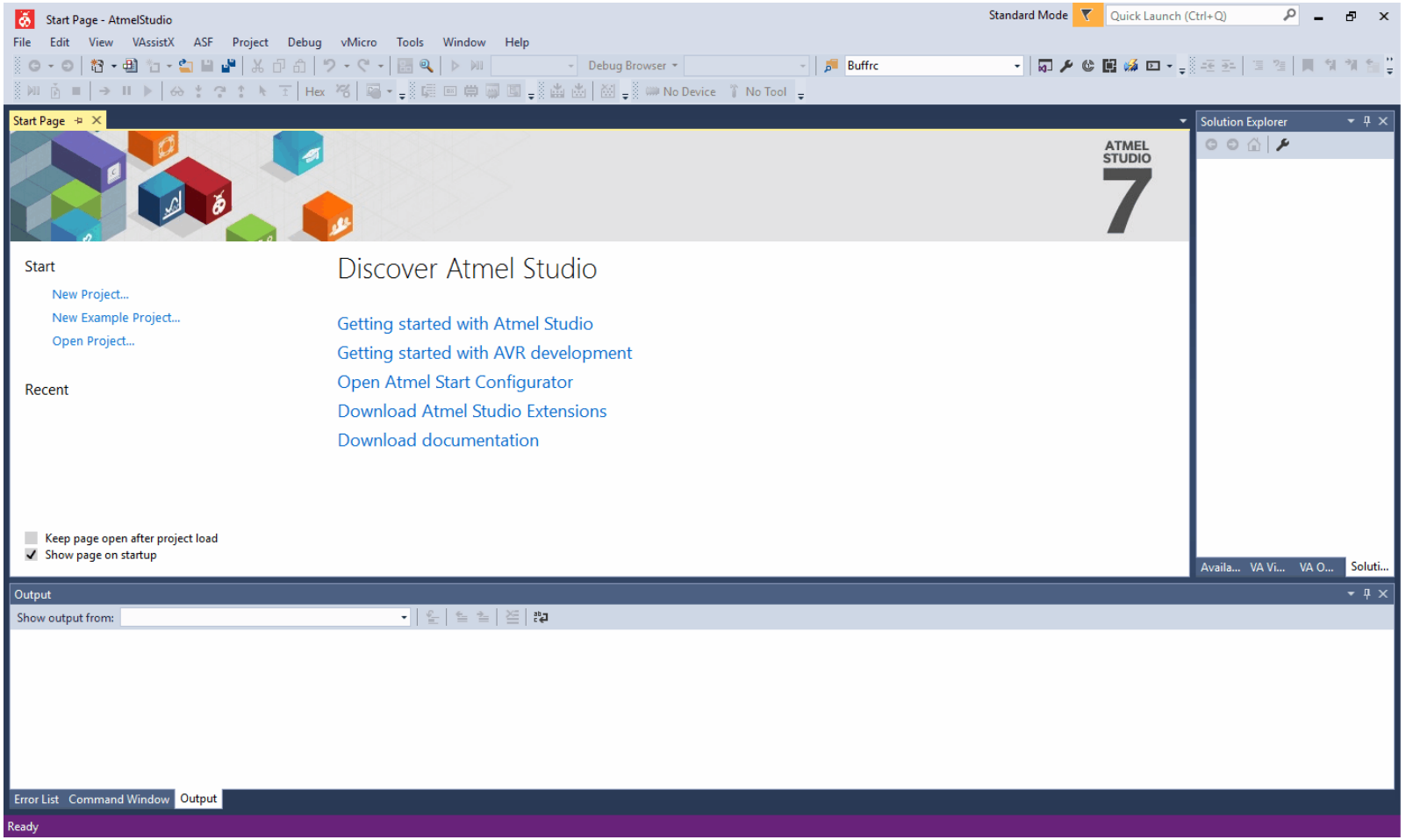
Atmel شرکت

- Atmel (advanced technology for memory and logic)
  - Founded in 1984
  - By George Perlegos

Logo from 1984 to 2012

    - Worked in the memory group of intel in the 1970s
    - Co-founded seeq technology to manufacture EPROM memory
  - Atmel was initially operated as a fabless company
    - 1989, purchase fabrication facility
  - In 2016, acquired by microchip technology
    - Previously heard offers, in 2008 (Microchip Technology) and 2015

Atmel Studio

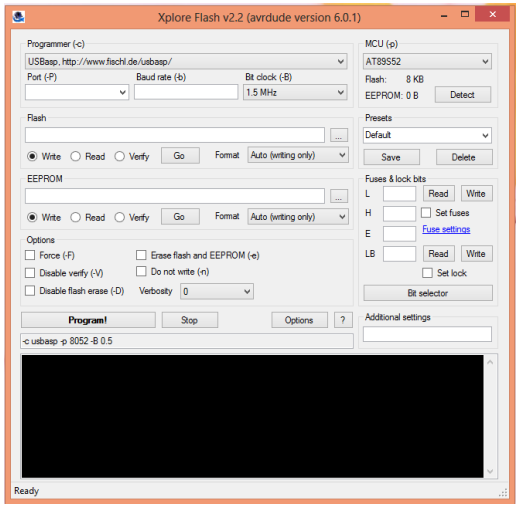


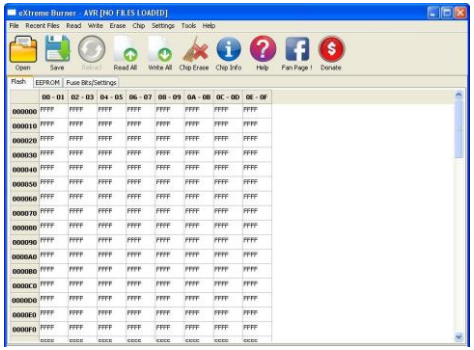
Atmel Studio

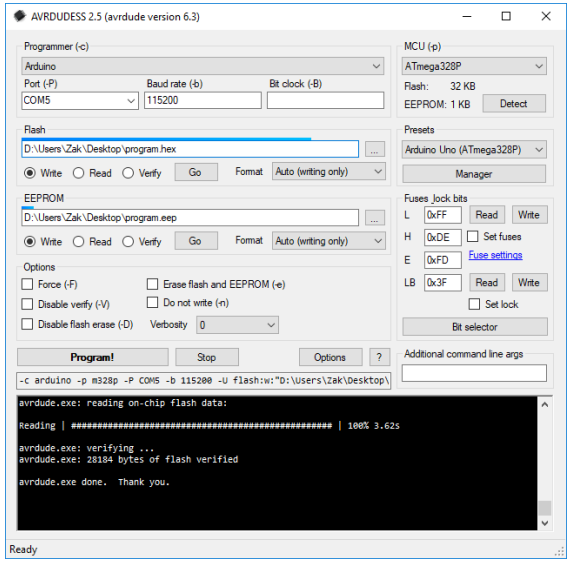
- Atmel Studio ⟶ Microchip Studio
  - After Acquired by Microchip Technology

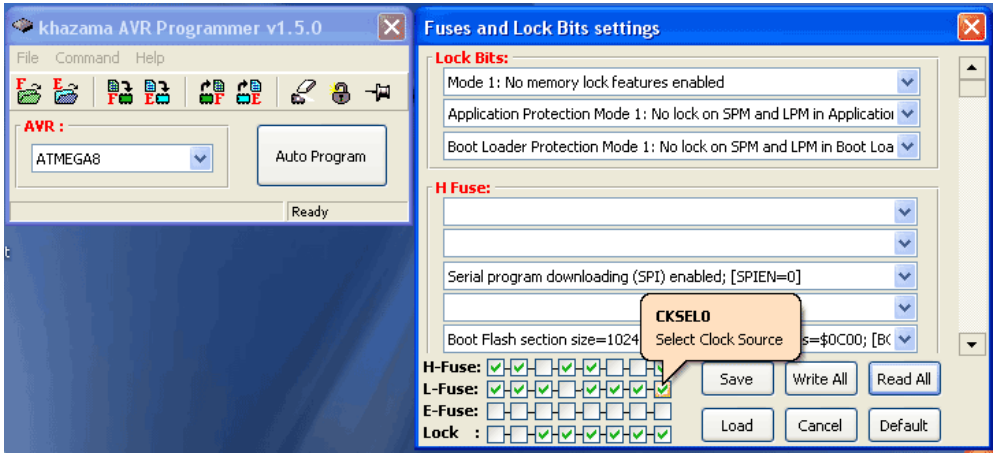
Toolchain

- Programming Software
  - Xplore Flash
  - Extreme Burner
  - Khazama
  - AVRDude (AVRDUDESS – A GUI for AVRDUDE)

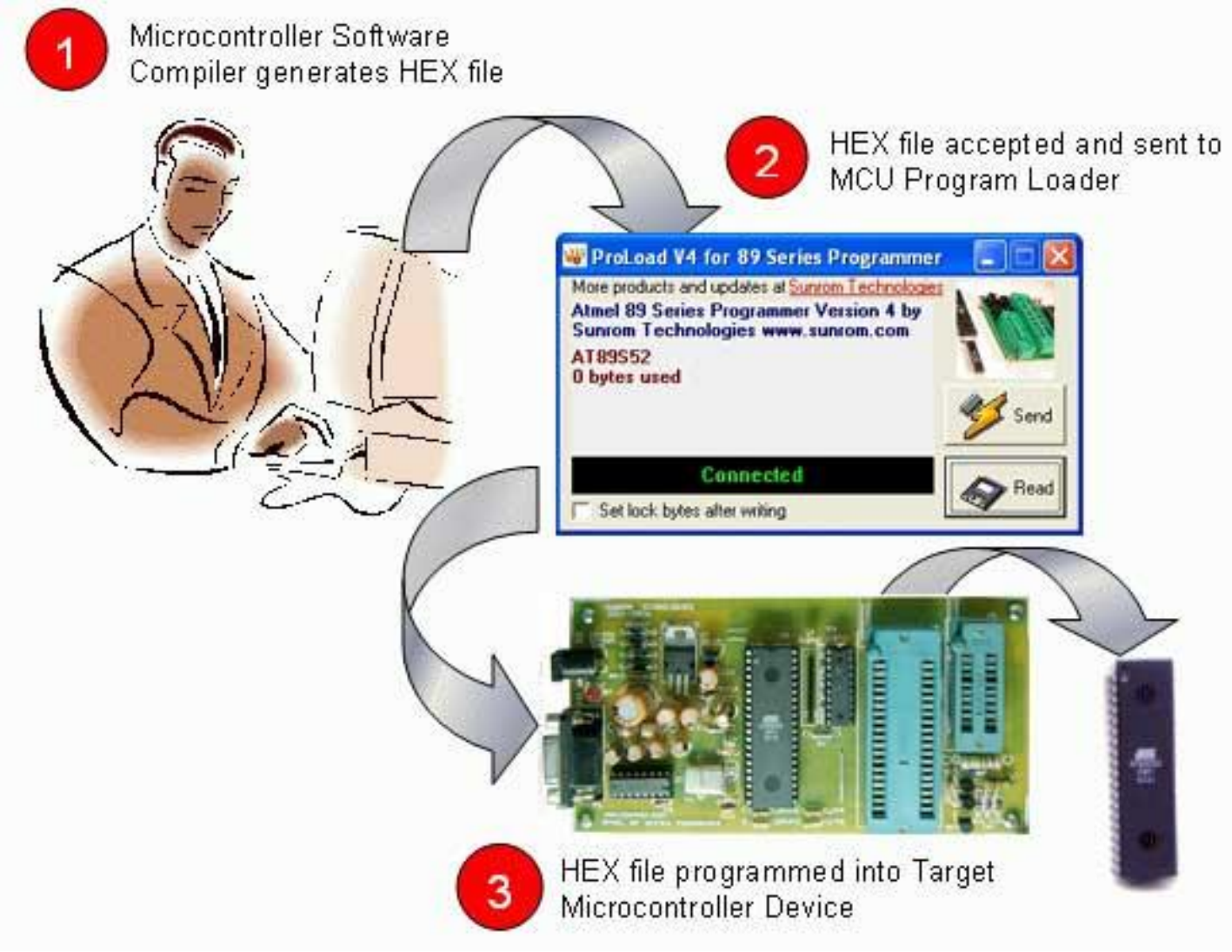




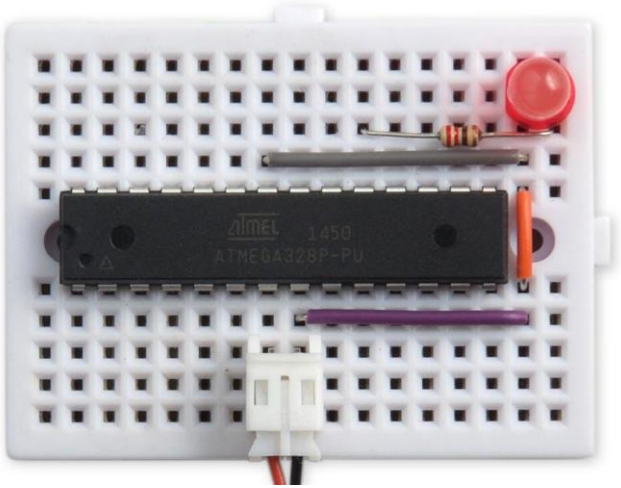


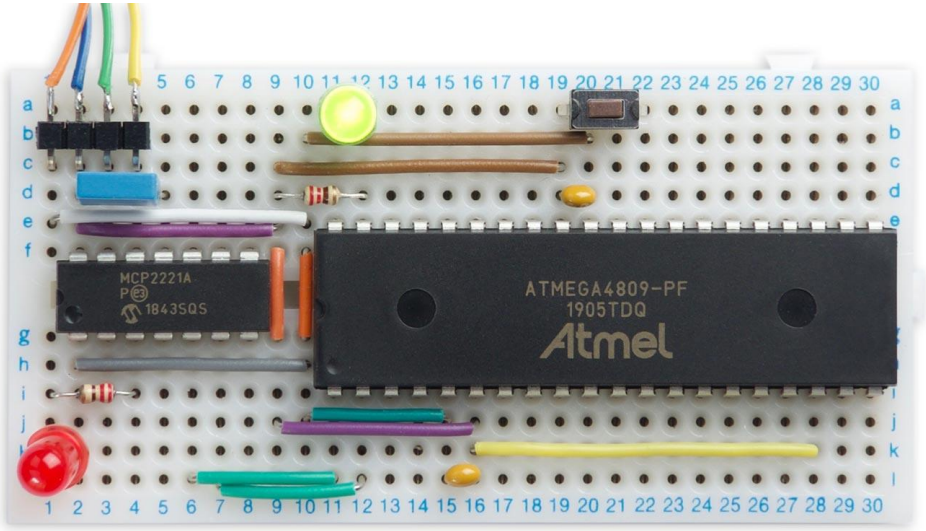


Program AVR µC

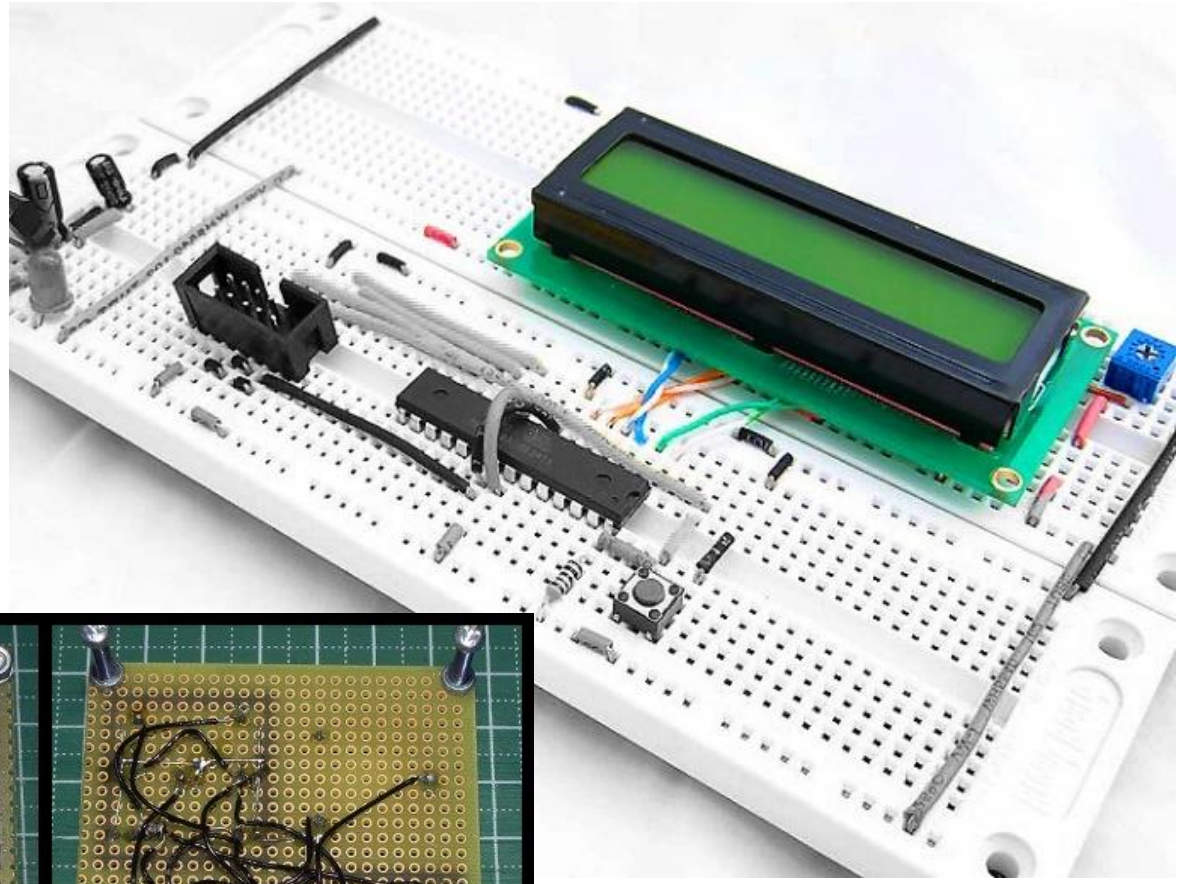


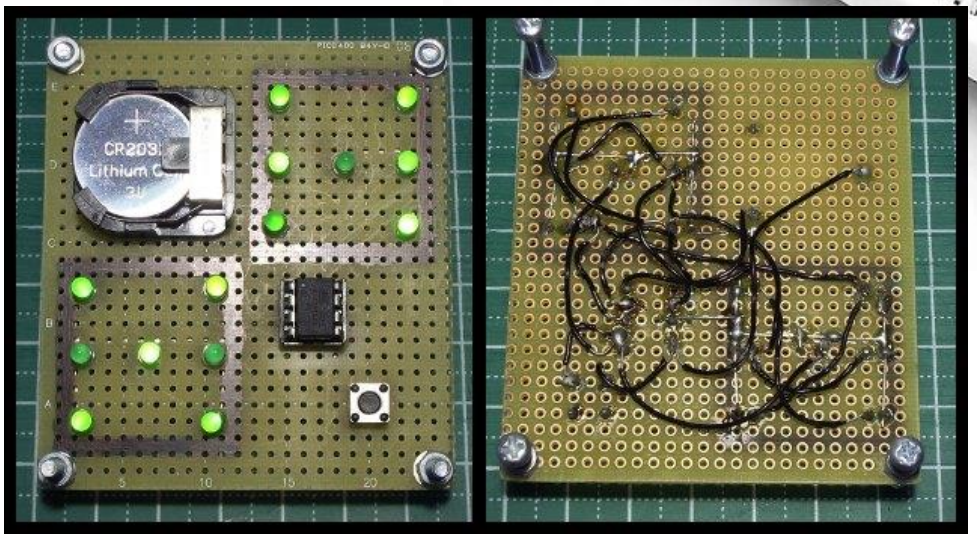
سخت افزار مورد نیاز



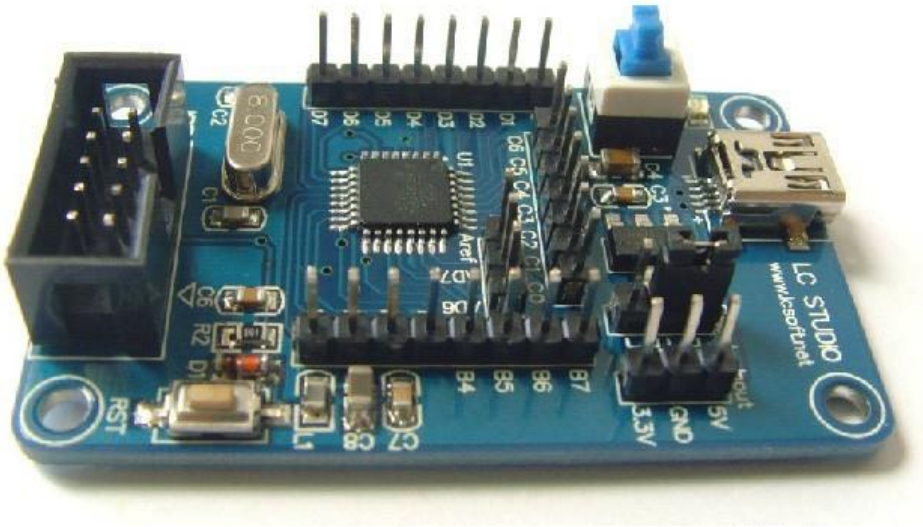


سخت افزار مورد نیاز

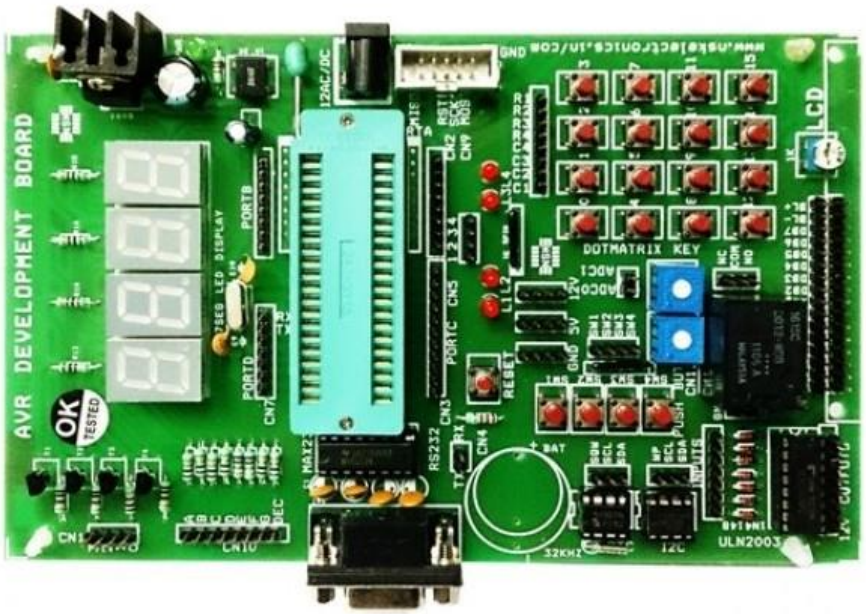


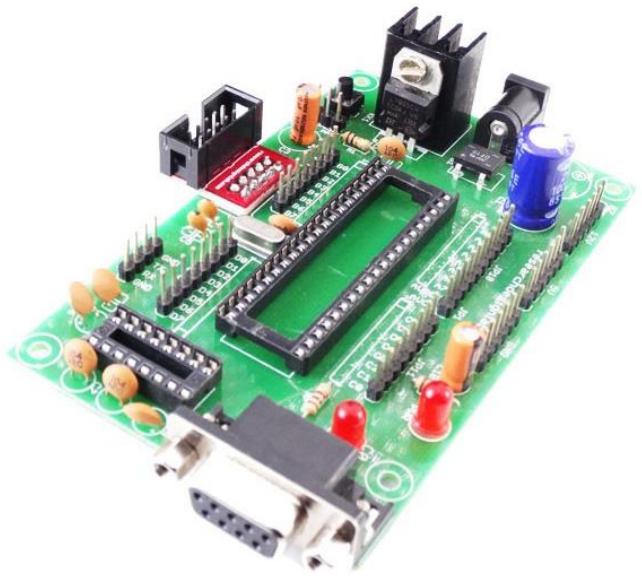


مدار کامل

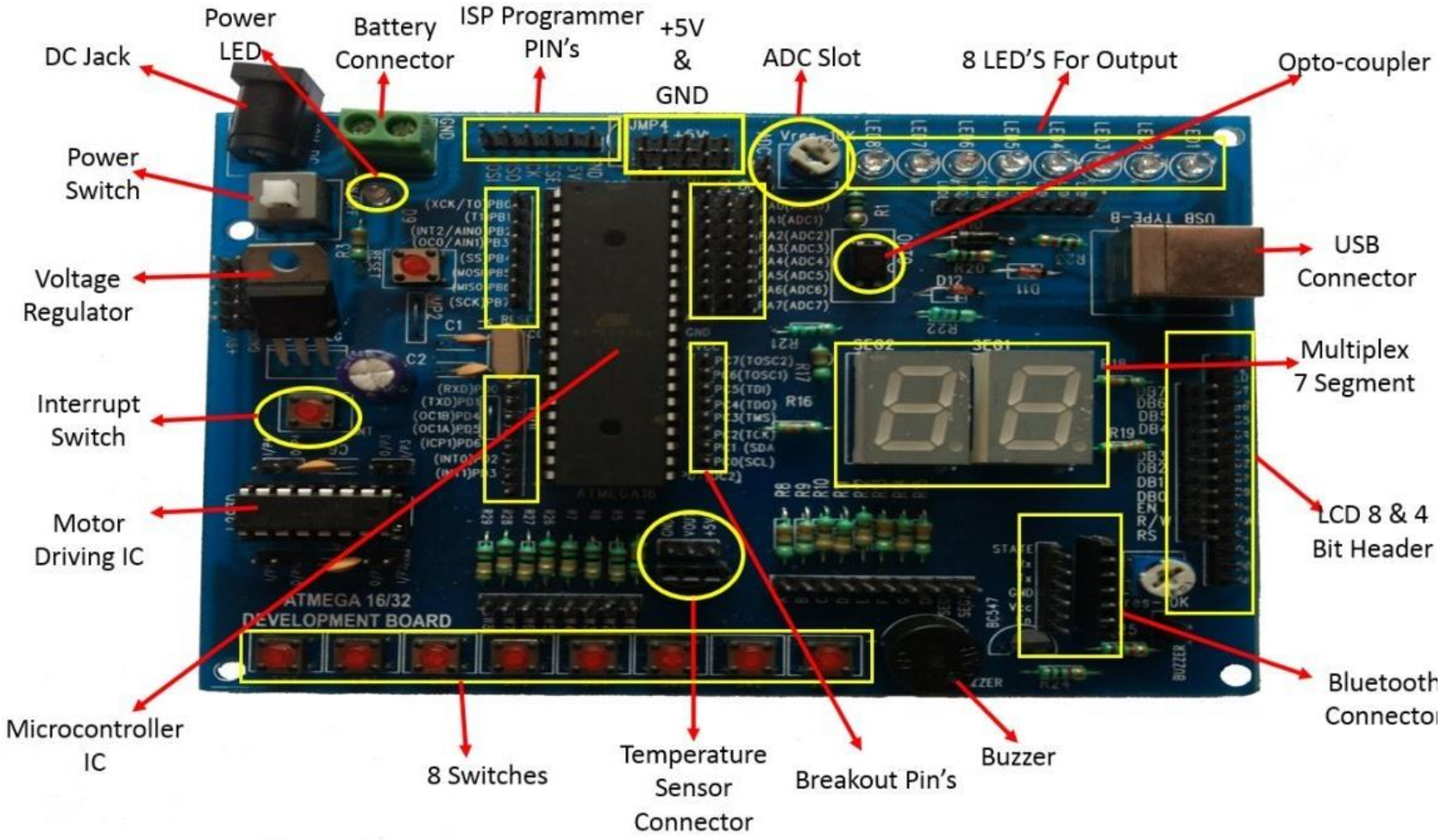


برد توسعه


AVR DEVELOPMENT BOARD
DOTMATRIX KEY
RESET
RS232
ULN2003
LCD



برد توسعه


DC Jack
Power LED
Battery Connector
ISP Programmer PIN's
+5V & GND
ADC Slot
8 LED'S For Output
Opto-coupler
Power Switch
Voltage Regulator
Interrupt Switch
Motor Driving IC
Microcontroller IC
8 Switches
Temperature Sensor Connector
Breakout Pin's
Buzzer
USB Connector
Multiplex 7 Segment
LCD 8 & 4 Bit Header
Bluetooth Connector
ATMEGA 16/32
DEVELOPMENT BOARD

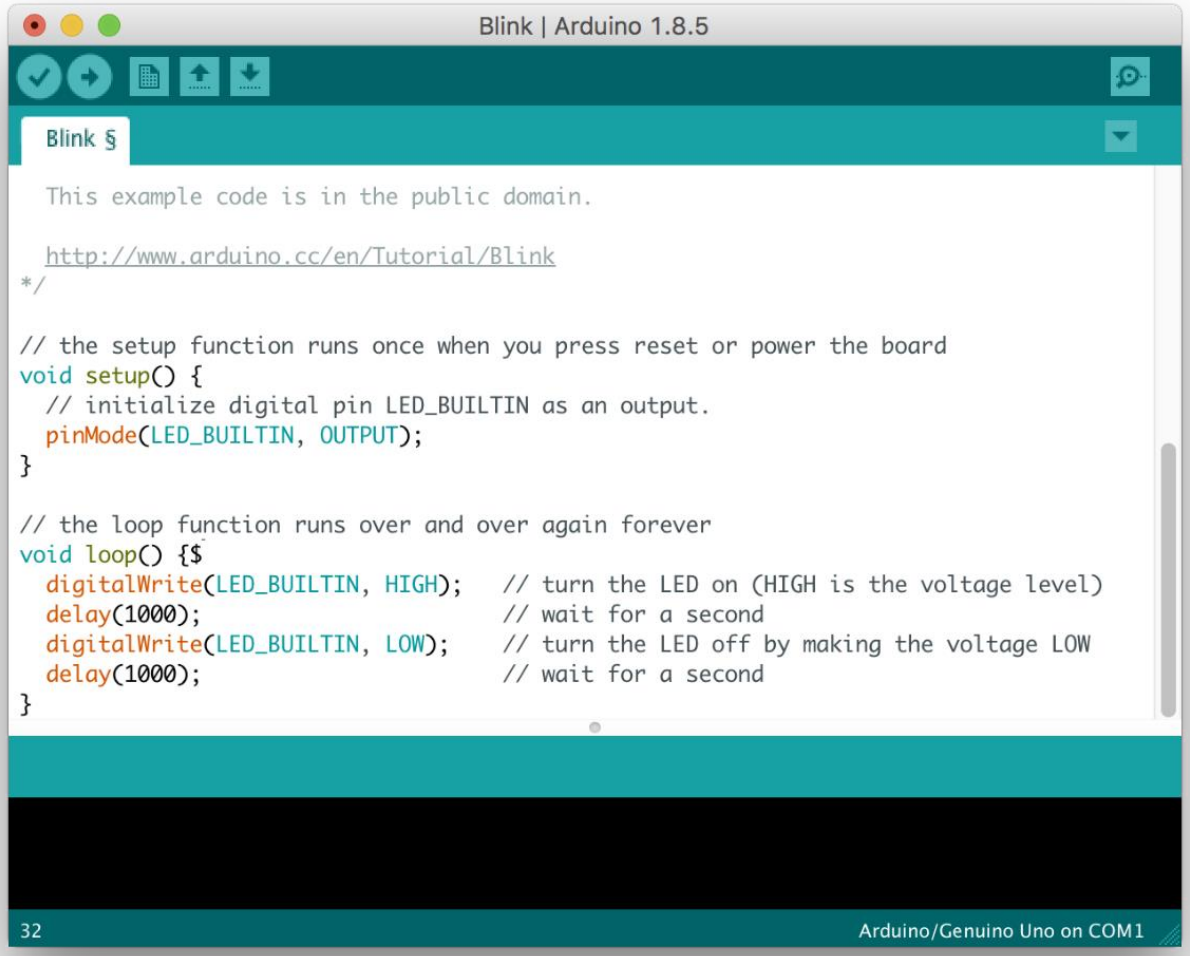
برد آردوینو


Reset Switch
USB Connector
USB Interface Chip
Crystal Oscillator
Voltage Regulator
Power Port
Digital Pins
TX RX LEDs
Microcontroller
Analog Input Pins
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN

برد آردوینو


Battery pack
remote control
2x16 LCD
Joystick cap
Large button
74HC595N
Soil huamidity sensor
Flame
LED-Yellow
RGB
LED-Red
LM35
LED-Blue
IR recalver
Tilt
Buzzer (active)
Buzzer (passive)
7-segLED 4X
7-segLED 1X
Gas Sensor
Pin headers
HC-SR04
ADXL345
830 hole Breadboard
Servo Motor
Stepper driver
PIR Motion
DHT11
RFID
Joystick
2560 R3
LED Matrix
Access key
Access card
DS3231
USB cable
Dupont wire
Jumper Wire

محیط توسعه آردوینو



پایان

موفق و پیروز باشید