Arduino

A free development system based on Atmel AVR 8 bit microcontrollers.

http://s.id/sisdig2017

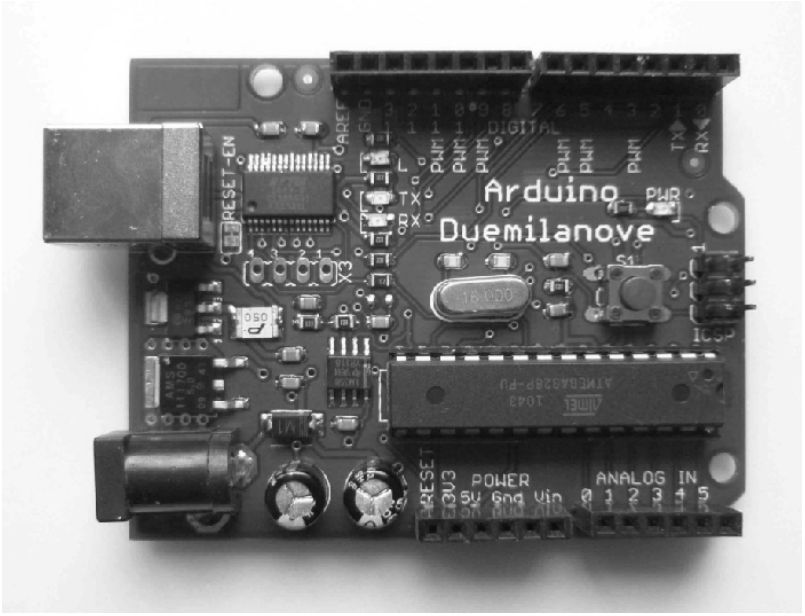
What is AVR

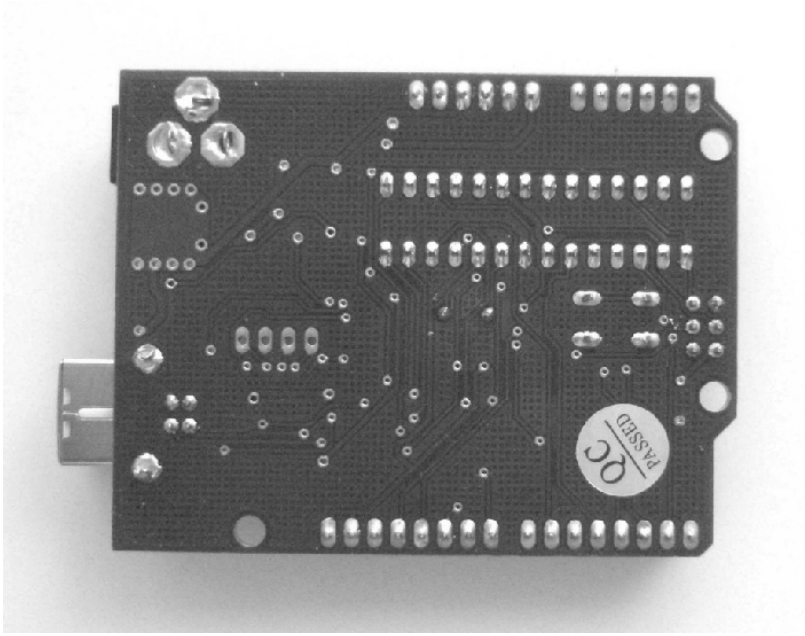
- RISC architecture microcontroller
- Designed for high level languages, developed in Trondheim, Norway in 1996
- Classic: AT90S1200, AT90S2343, AT90S2313, AT90S4433, AT90S8515, AT90S8535
- ATtiny22, ATtiny25-85, ATtiny2313 ...
- ATmega8, ATmega16, ATmega 48-328 ...
- Flash programmable memory
- ATmega self programming
- RAM, EEPROM and peripherals

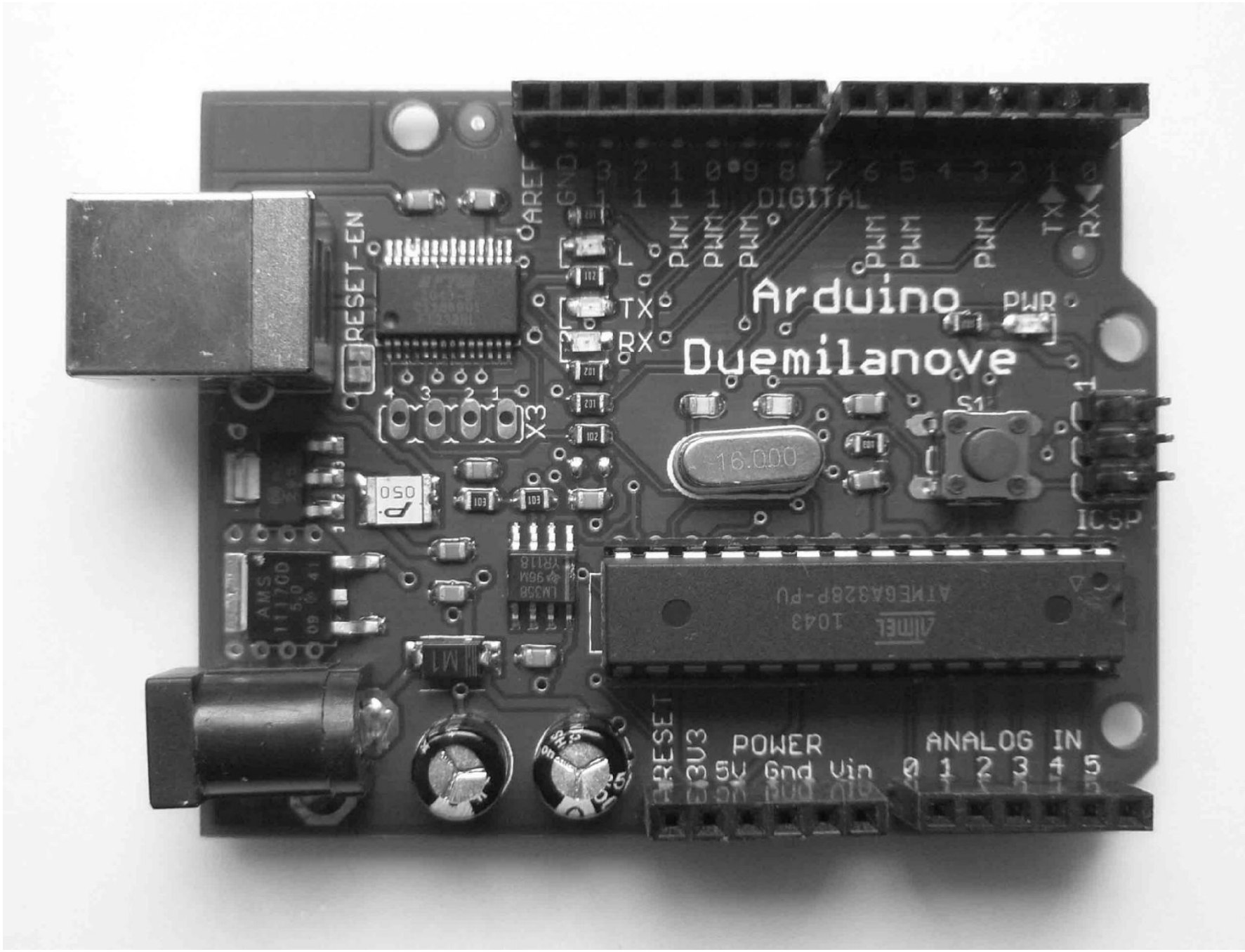
What is Arduino

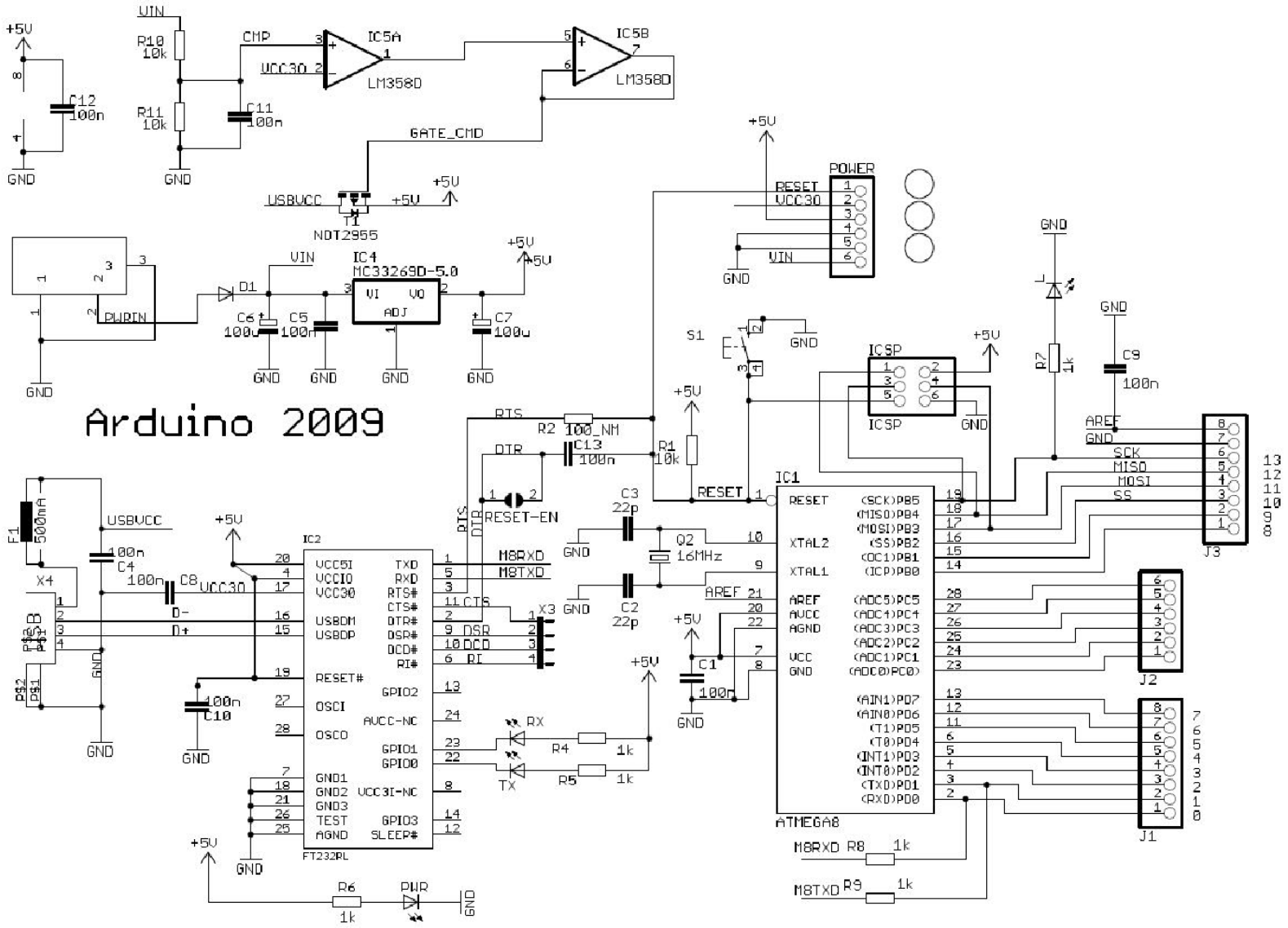
- Open Source Hardware, you can make your own board, or buy one.
- Cheap, easily available.
- Open Source Software.
- Very widespread, many projects openly available.
- Extra HW (shields) available.

Arduino Duemilanove (2009)



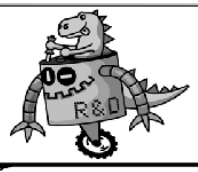



Arduino
Duemilanove
DIGITAL
PWR
ICSP
POWER
ANALOG IN
RESET-EN
16.000


Arduino 2009
+5V
VIN
R10 10k
R11 10k
CMP
IC5A
LM358D
IC5B
LM358D
C12 100n
C11 100n
GND
GATE_CMD
USBVCC
T1
NDT2955
IC4
MC33269D-5.0
VI
VO
ADJ
D1
C6 100u
C5 100n
C7 100u
PWRIN
POWER
RESET
VCC30
VIN
S1
ICSP
L
R7 1k
C9 100n
AREF
SCK
MISO
MOSI
SS
J3
J2
J1
RTS
DTR
R2 100_NM
C13 100n
R1 10k
RESET-EN
C3 22p
C2 22p
Q2 16MHz
IC1
ATMEGA8
RESET
XTAL2
XTAL1
AVCC
AGND
VCC
(SCK)PB5
(MISO)PB4
(MOSI)PB3
(SS)PB2
(OC1)PB1
(ICP)PB0
(ADC5)PC5
(ADC4)PC4
(ADC3)PC3
(ADC2)PC2
(ADC1)PC1
(ADC0)PC0
(AIN1)PD7
(AIN0)PD6
(T1)PD5
(T0)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
F1 500mA
USBVCC
C4 100n
C8 100n
VCC30
D-
D+
X4
X3
IC2
FT232RL
VCC5I
VCCIO
VCC30
USBDM
USBDP
RESET#
OSCI
OSCO
GND1
GND2
GND3
TEST
AGND
TXD
RXD
RTS#
CTS#
DTR#
DSR#
DCD#
RI#
GPIO2
AVCC-NC
GPIO1
GPIO0
VCC3I-NC
GPIO3
SLEEP#
M8RXD
M8TXD
CTS
DSR
DCD
RI
C10 100n
RX
TX
R4 1k
R5 1k
R6 1k
PWR
M8RXD R8 1k
M8TXD R9 1k

Arduino Cheat Sheet V.02c

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace



Structure
void setup() **void loop()**

Control Structures
```
if (x<5){ } else { }
switch (myvar) {
  case 1:
    break;
  case 2:
    break;
  default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } while (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto          // considered harmful :-)
```

Further Syntax
// (single line comment)
/* (multi-line comment) */
#define DOZEN 12 //Not baker's!
#include <avr/pgmspace.h>

General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) | (or) ! (not)

Pointer Access
& reference operator
* dereference operator

Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
0b11011111 //**Binary**
0x7B // **Hex** number
7**U** // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4e5 // 240000

Data Types
void
boolean (0, 1, false, true)
char (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte (0 to 255)
int (-32,768 to 32,767)
unsigned int (0 to 65535)
word (0 to 65535)
long (-2,147,483,648 to 2,147,483,647)
unsigned long (0 to 4,294,967,295)
float (-3.4028235E−38 to 3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes

Strings
```
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
  //Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";
```

Arrays
```
int myInts[6];
int myPins[ ] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
```

Conversion
char() **byte**()
int() **word**()
long() **float**()

Qualifiers
static // persists between calls
volatile // use RAM (nice for ISR)
const // make read-only
PROGMEM // use flash

Digital I/O
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

Analog I/O
analogReference([DEFAULT,INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source.
analogWrite(pin, value) // PWM

Advanced I/O
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin, [HIGH,LOW])

Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
delay(ms)
delayMicroseconds(us)

Math
min(x, y) **max**(x, y) **abs**(x)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
pow(base, exponent) **sqrt**(x)
sin(rad) **cos**(rad) **tan**(rad)

Random Numbers
randomSeed(seed) // Long or int
long **random**(max)
long **random**(min, max)

Bits and Bytes
lowByte() **highByte**()
bitRead(x,bitn) **bitWrite**(x,bitn,bit)
bitSet(x,bitn) **bitClear**(x,bitn)
bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries:

Serial.
begin([300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 28400, 57600, 115200])
end()
int **available**()
int **read**()
flush()
print()
println()
write()

EEPROM (#include <EEPROM.h>)
byte **read**(intAddr)
write(intAddr,myByte)

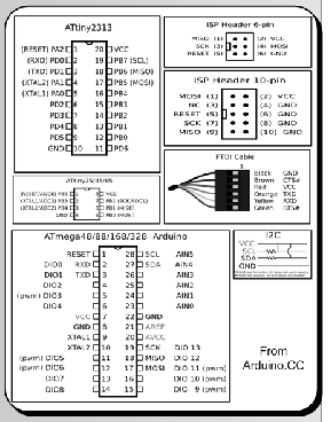
Servo (#include <Servo.h>)
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000, 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()

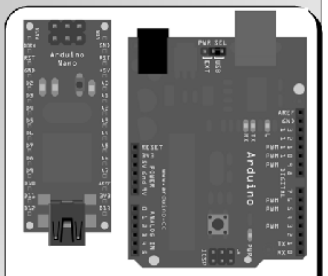
SoftwareSerial(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
begin(longSpeed) // up to 9600
char **read**() // blocks till data
print(myData) or **println**(myData)

Wire (#include <Wire.h>) // For **I2C**
begin() // Join as master
begin(addr) // Join as slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(mybyte) // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission() // Step 3
byte **available**() // Num of bytes
byte **receive**() //Return next byte
onReceive(handler)
onRequest(handler)

	ATMega168	ATMega328	ATmega1280
Flash (0k for bootloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

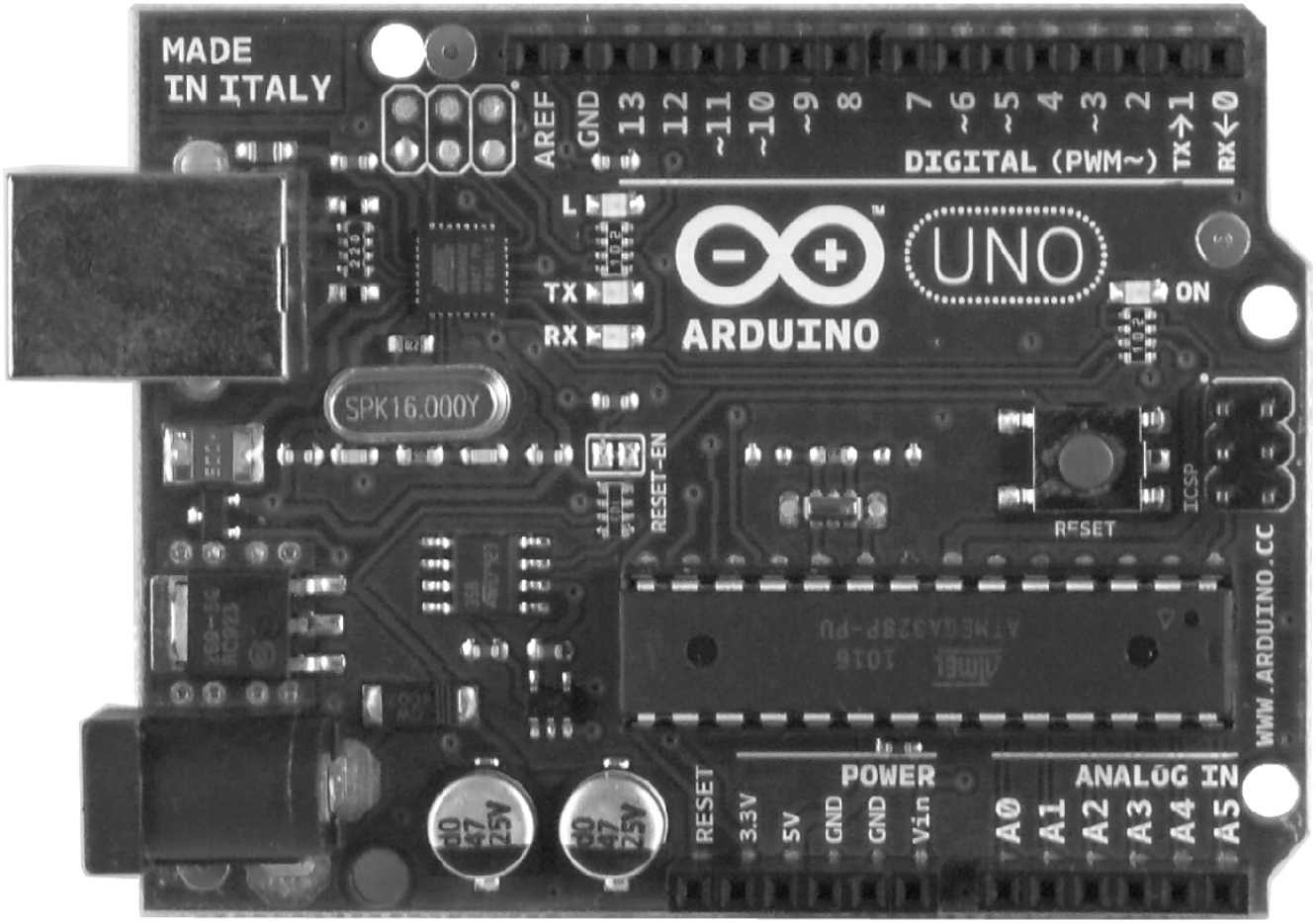
	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of I/O	14 + 6 analog (can be 14 + 8)	54 + 16 analog 0 - RX0 1 - TX0 14 - RX3 15 - TX3 16 - RX2 17 - TX2 19 - RX1 18 - TX1
Serial Pins	0 - RX 1 - TX	0 (RX0) 1 (TX0) 19 (RX1) 18 (TX1) 17 (RX2) 16 (TX2) 15 (RX3) 14 (TX3)
External Interrupts	2 (int 0) 3 (int 1)	2 (int 0) 3 (int 1) 21 (int 2) 20 (int 3) 19 (int 4) 18 (int 5)
PWM pins	3, 5, 6, 9, 10, 11 Timer 0 Timer 1 Timer 2	2 - 13 44 - 46
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog 4 - SDA Analog 5 - SCL	20 - SDA 21 - SCL



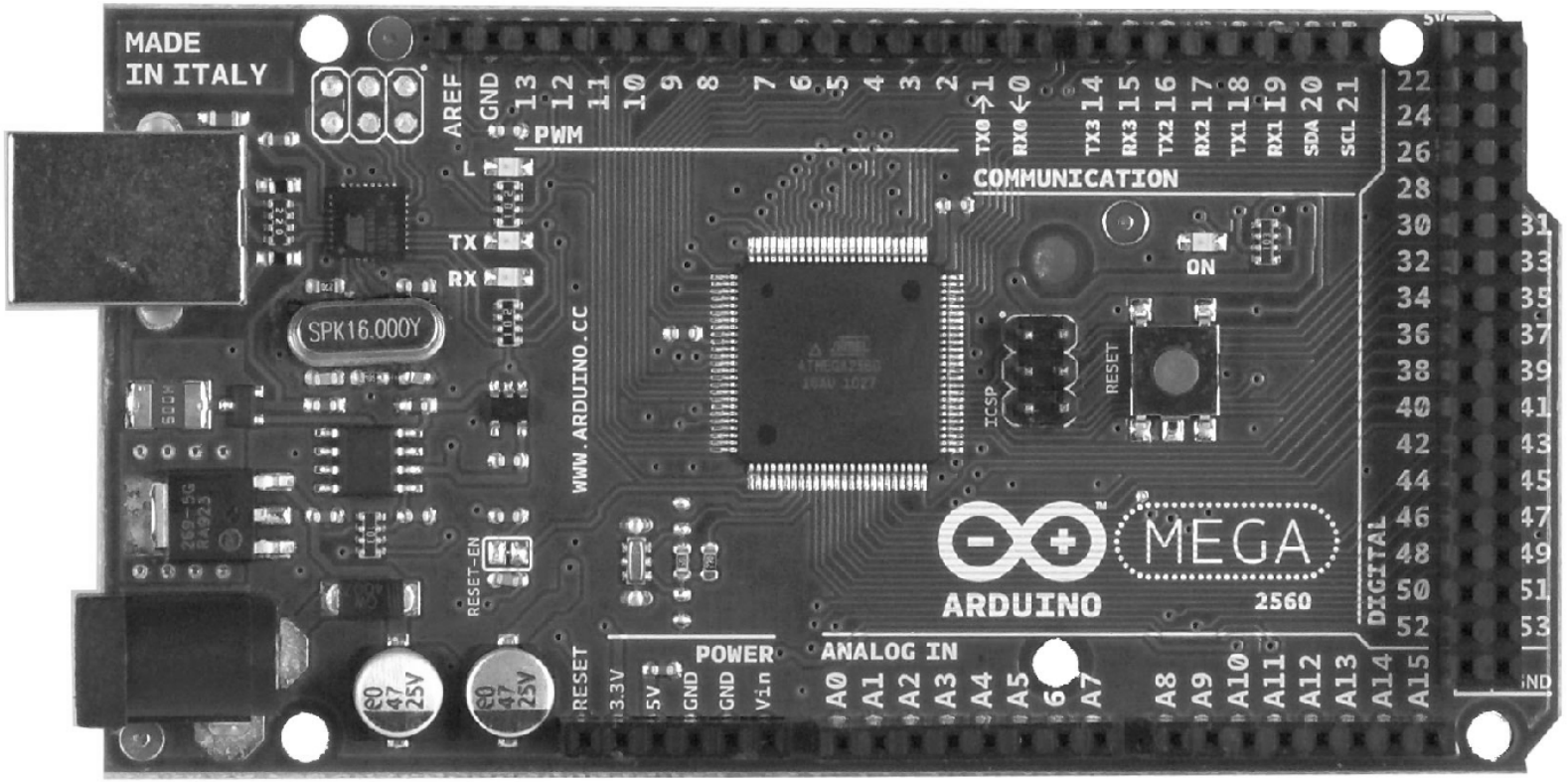


Pics from Fritzing.Org under C.C. license

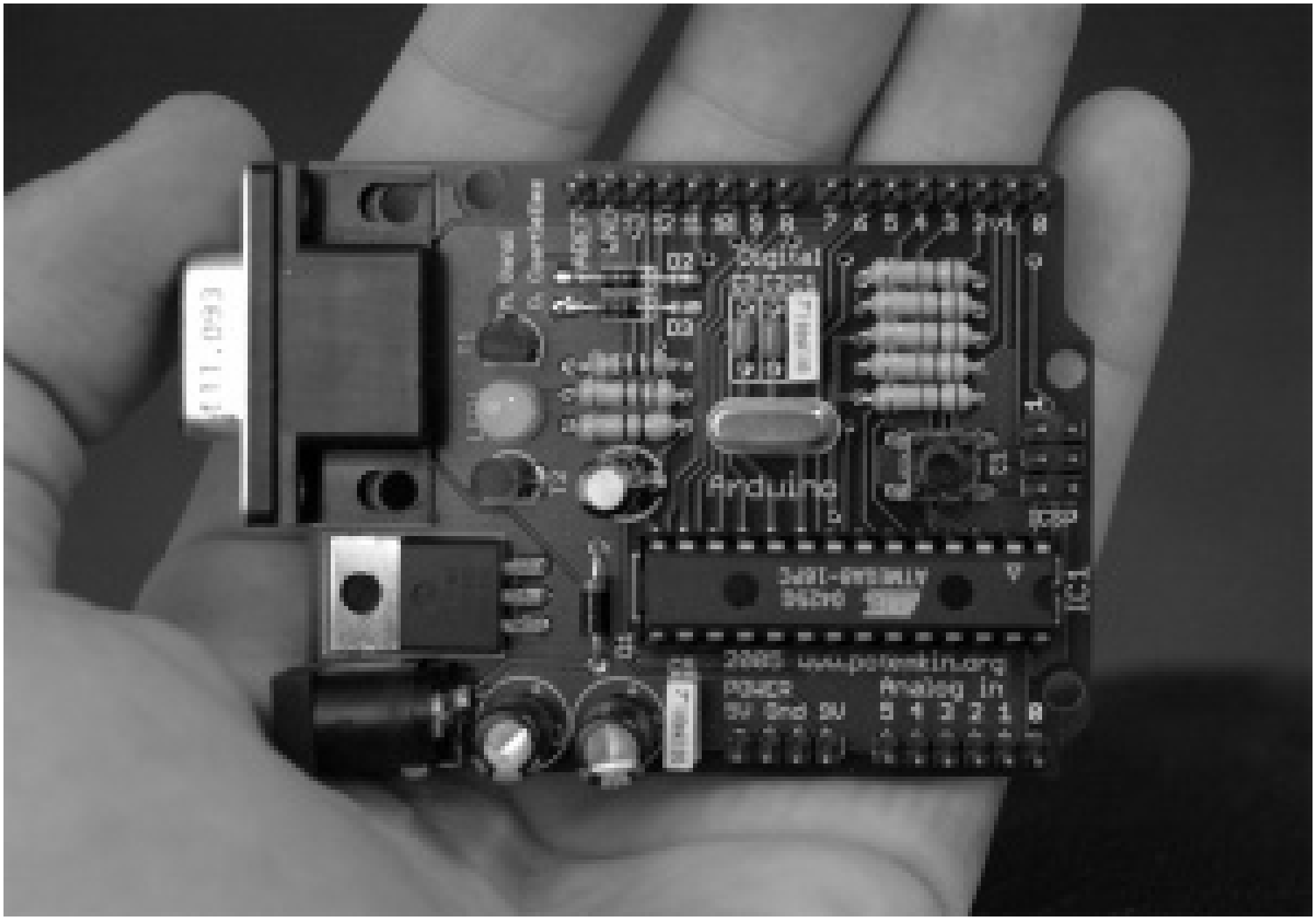
Arduino Uno


MADE
IN ITALY
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
SPK16.000Y
RESET-EN
RESET
ICSP
WWW.ARDUINO.CC
POWER
ANALOG IN
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

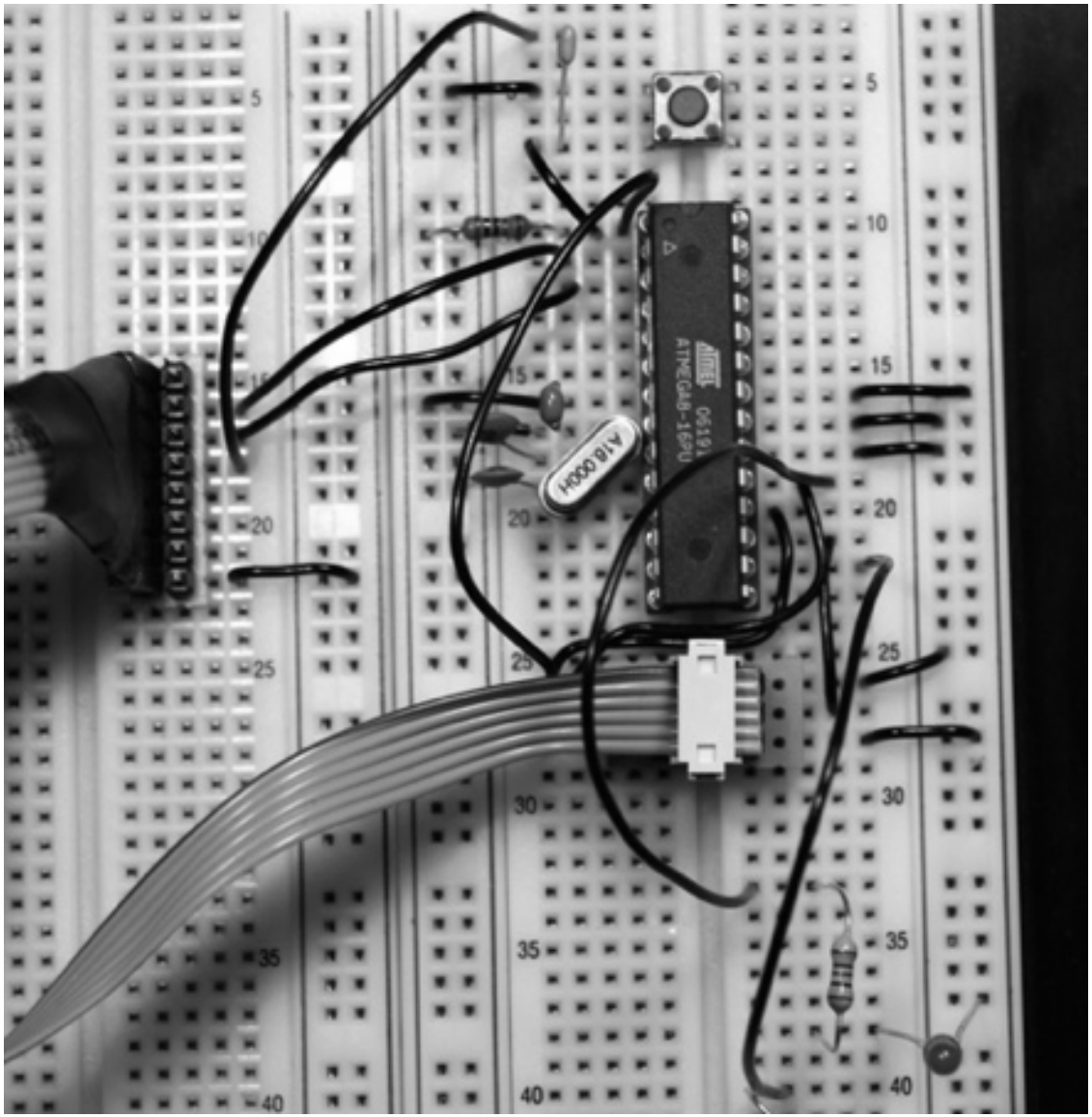
Arduino Mega 2560



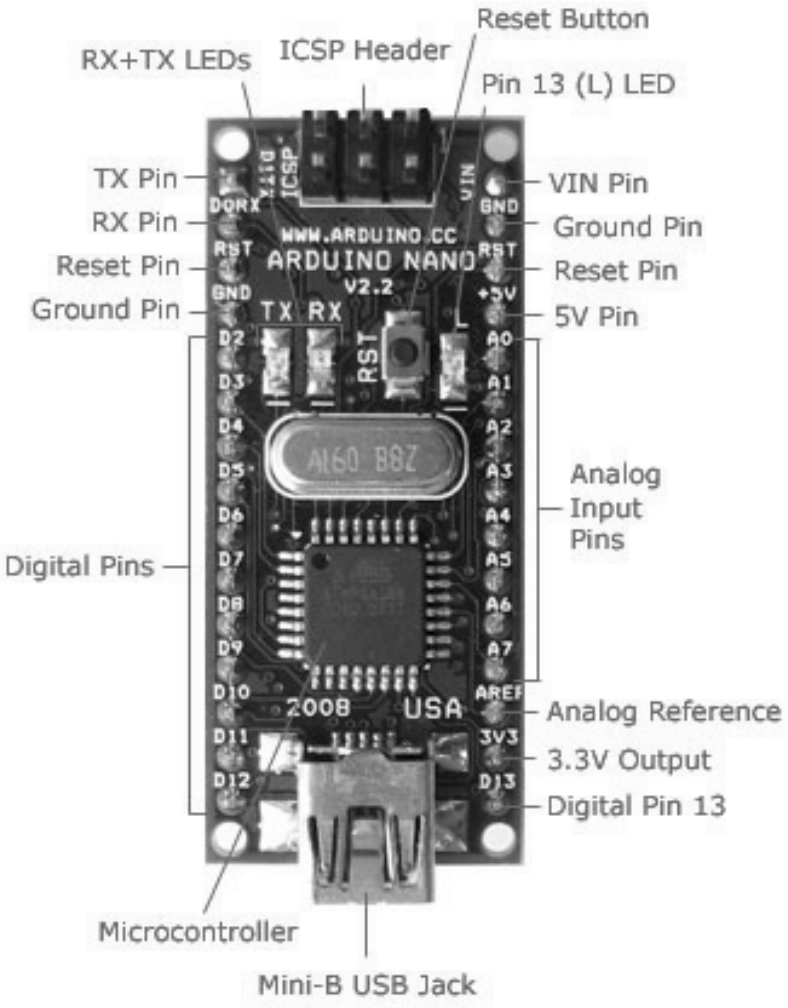
Original Arduino with RS-232

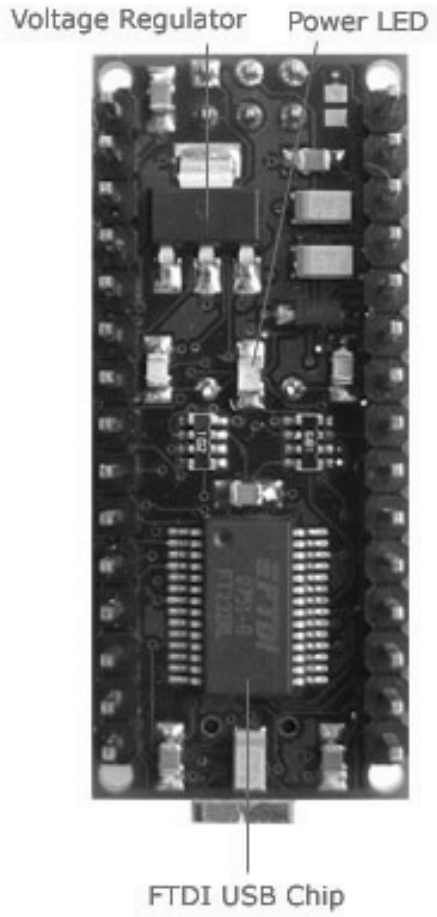


Arduino on breadboard

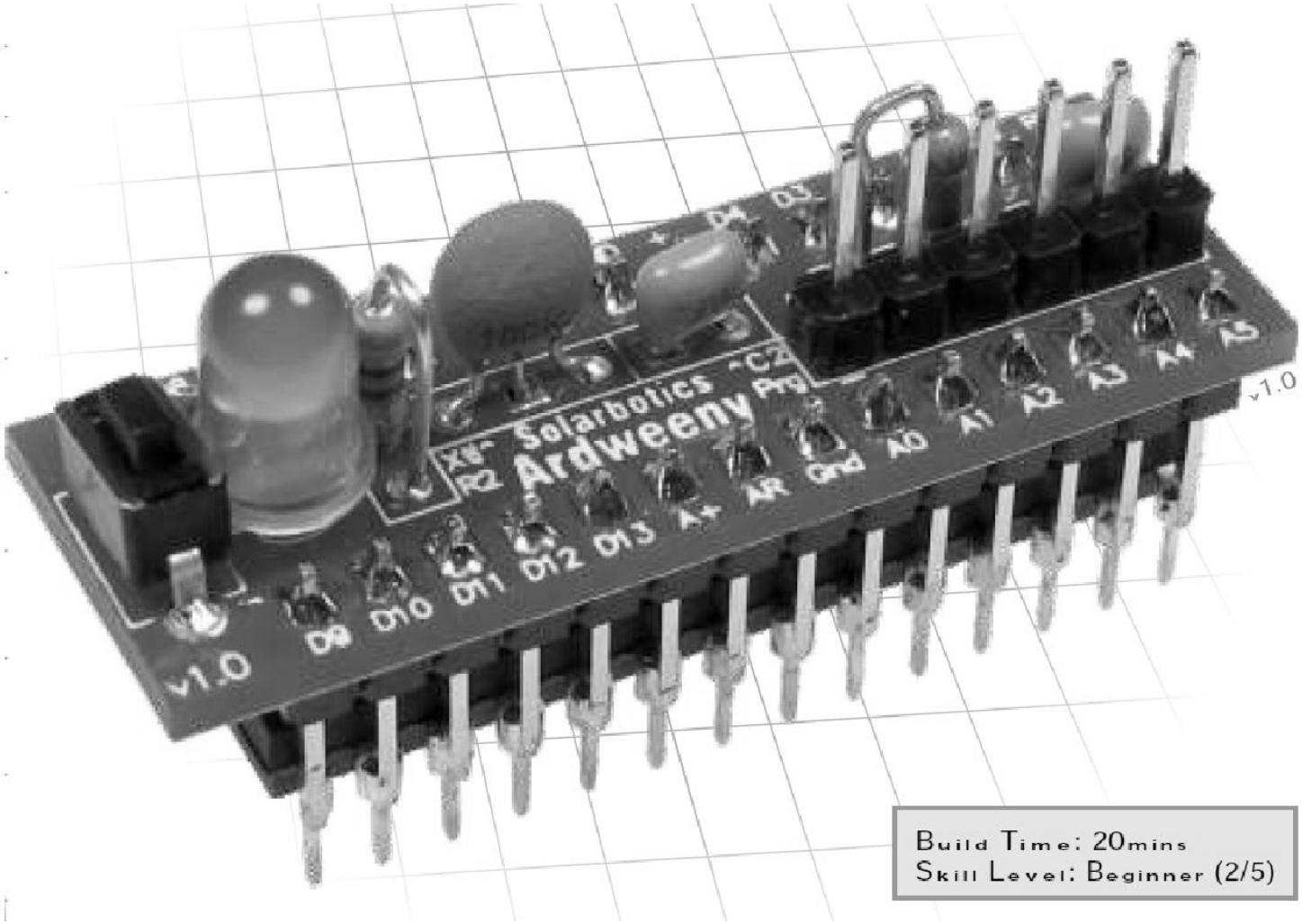


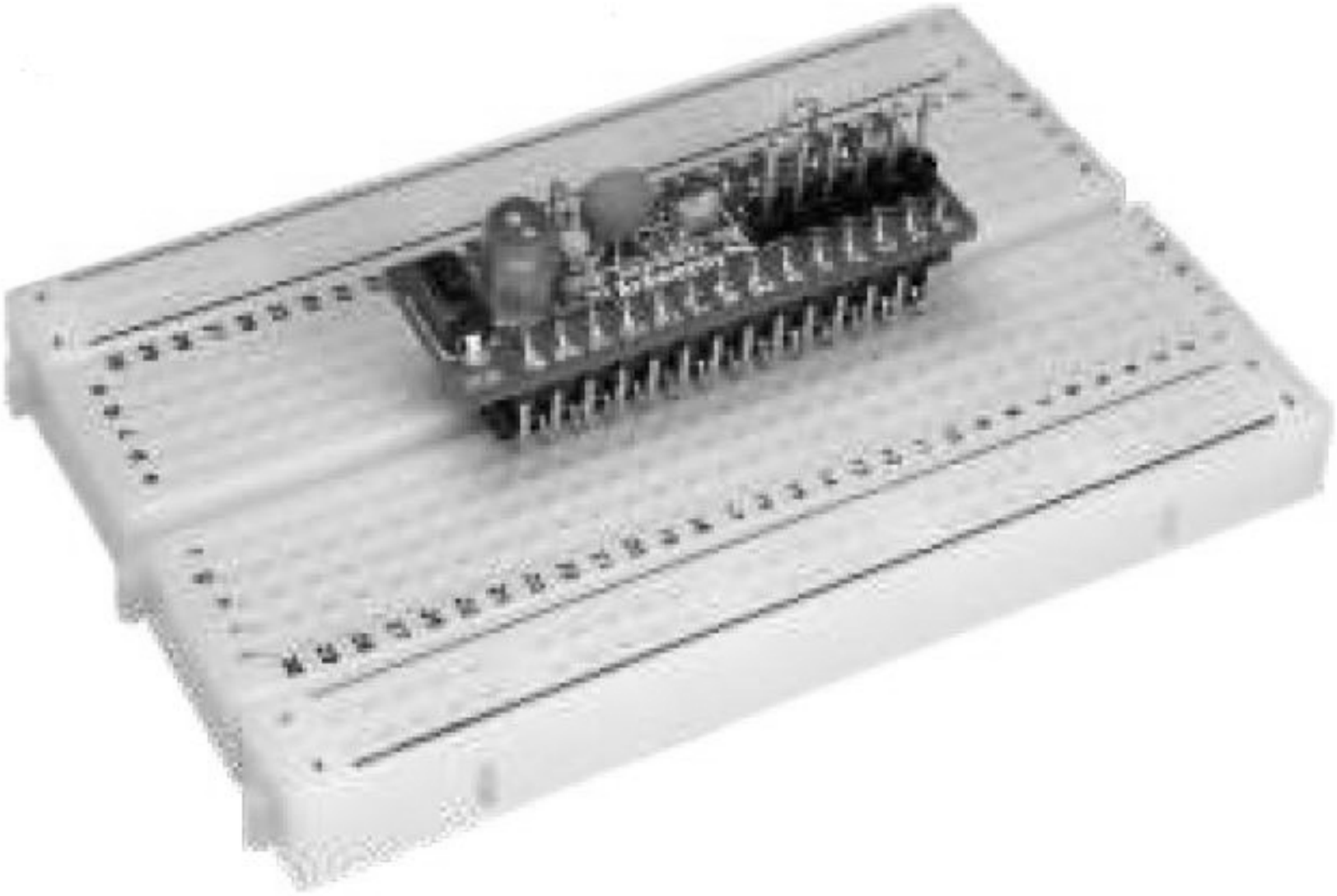
Arduino Nano

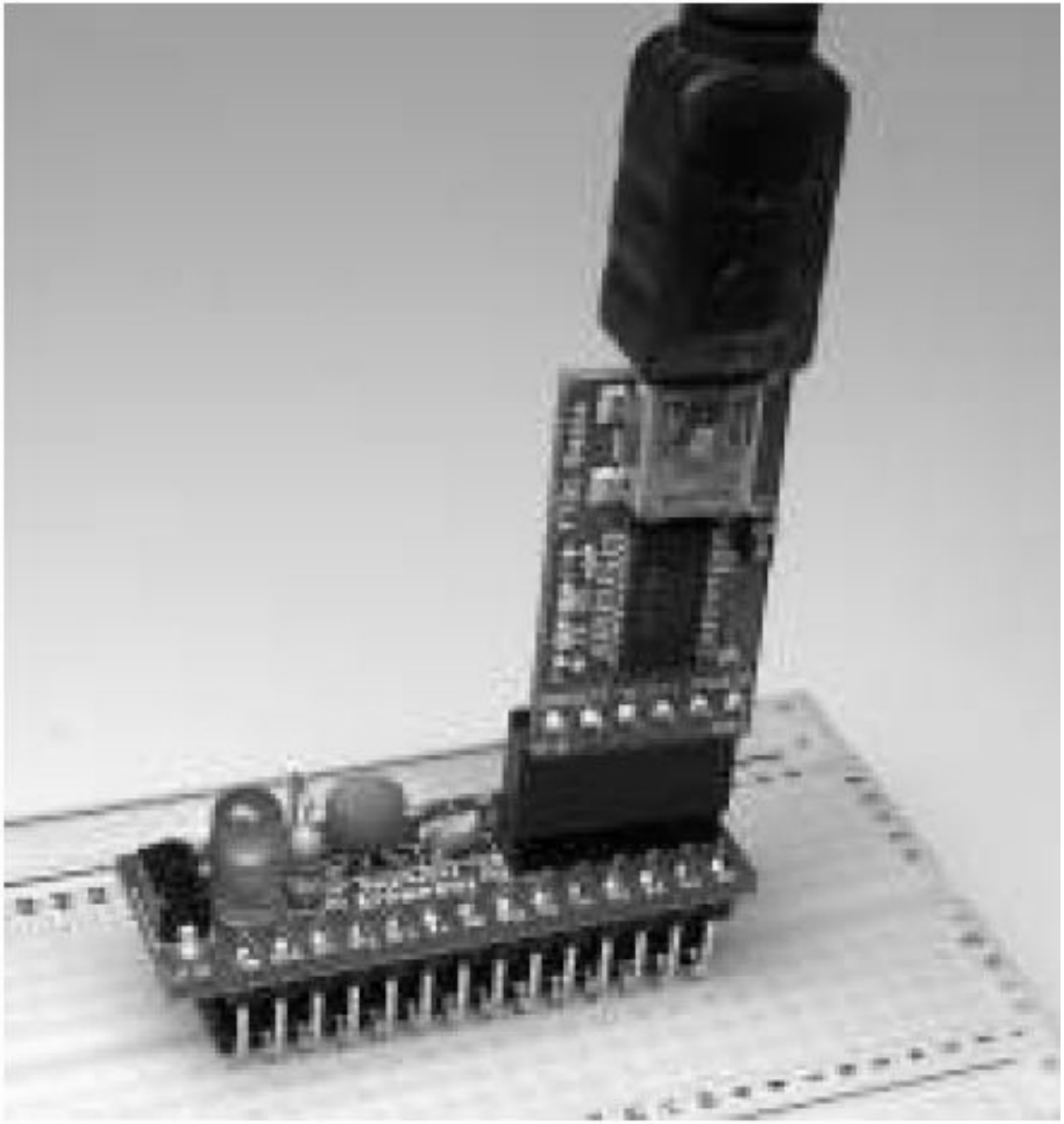




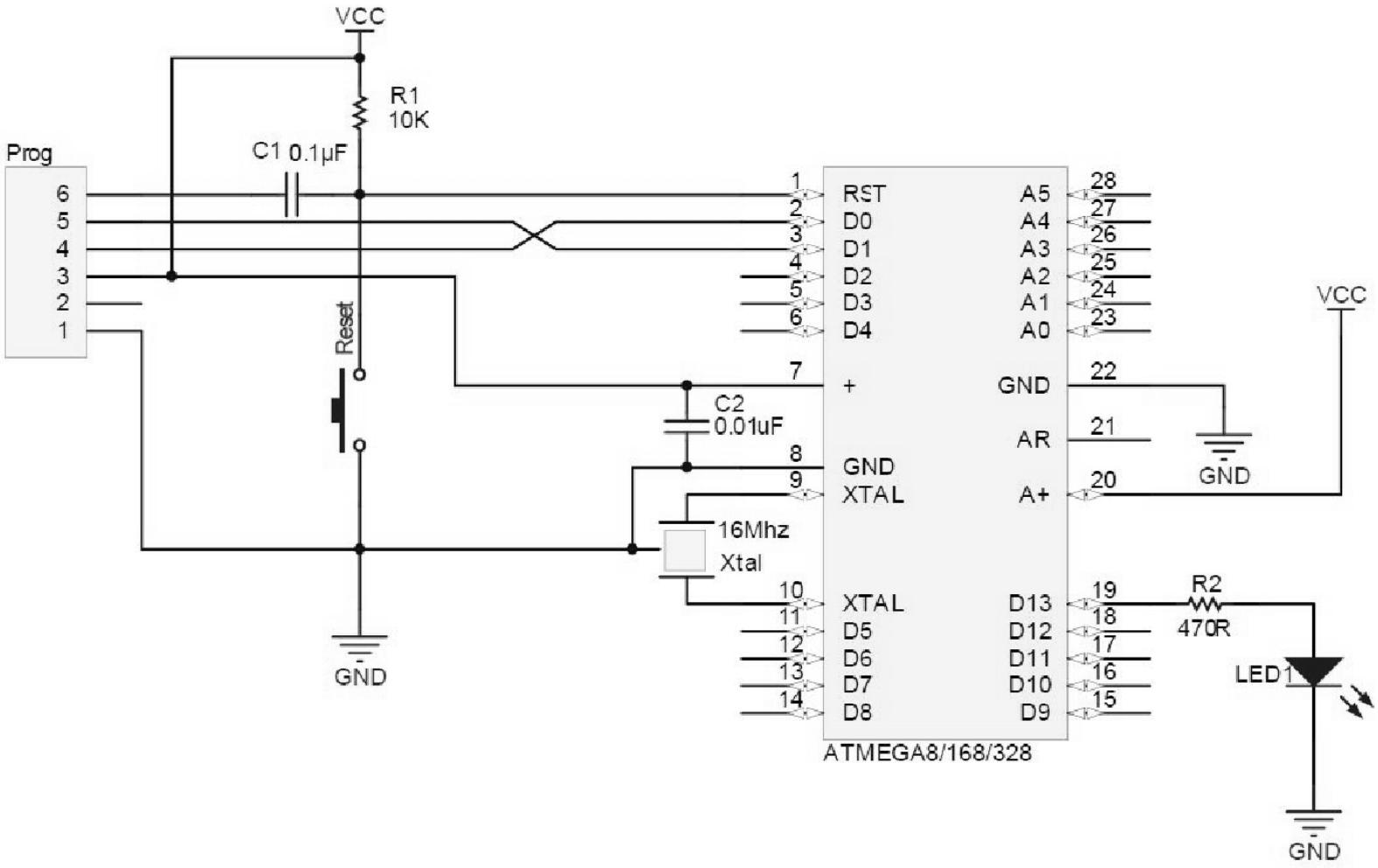
Ardweeny


Solarbotics
Ardweeny
v1.0
Build Time: 20mins
Skill Level: Beginner (2/5)

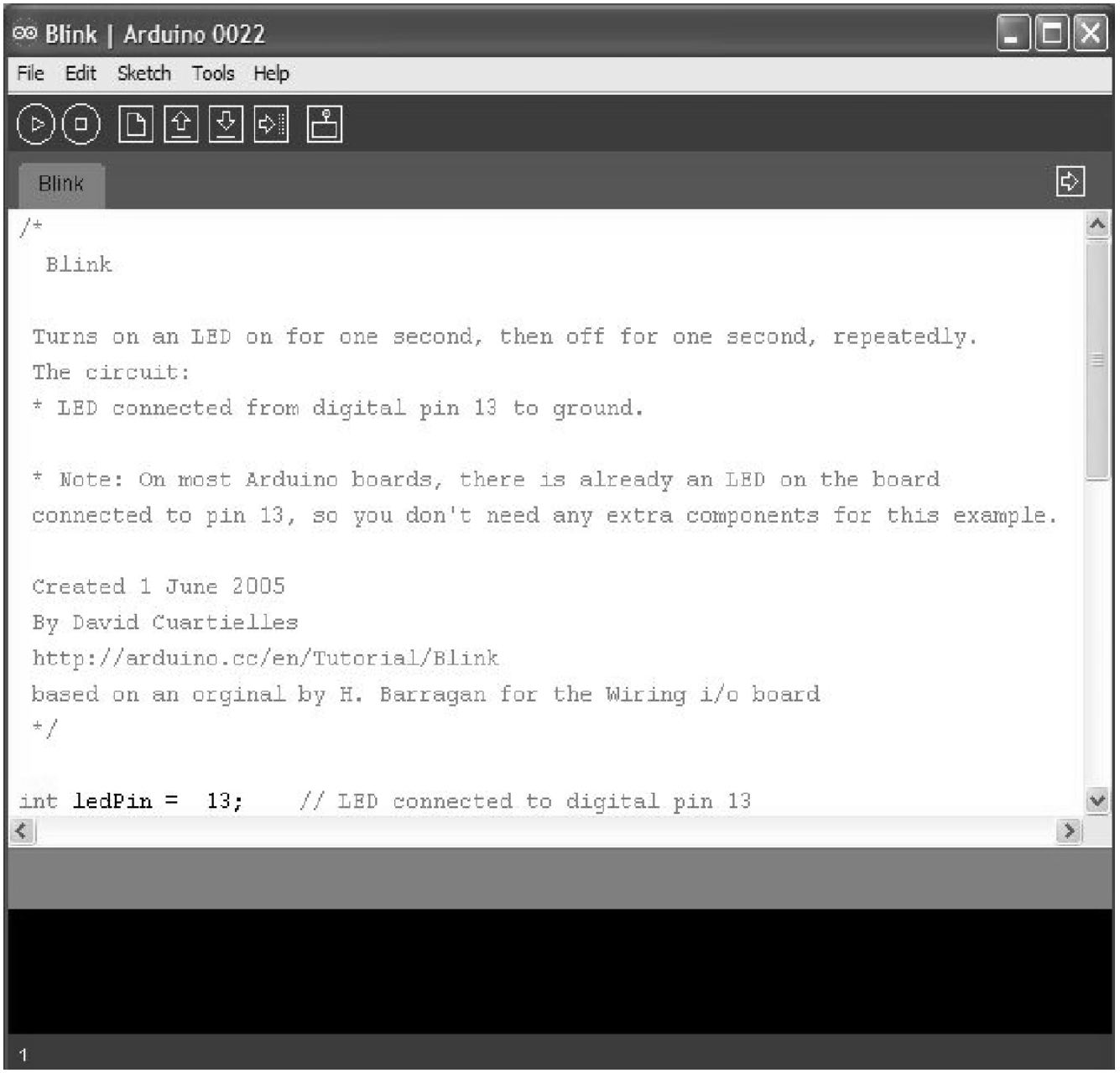




Ardweeny 1.0 Schematic



Arduino IDE

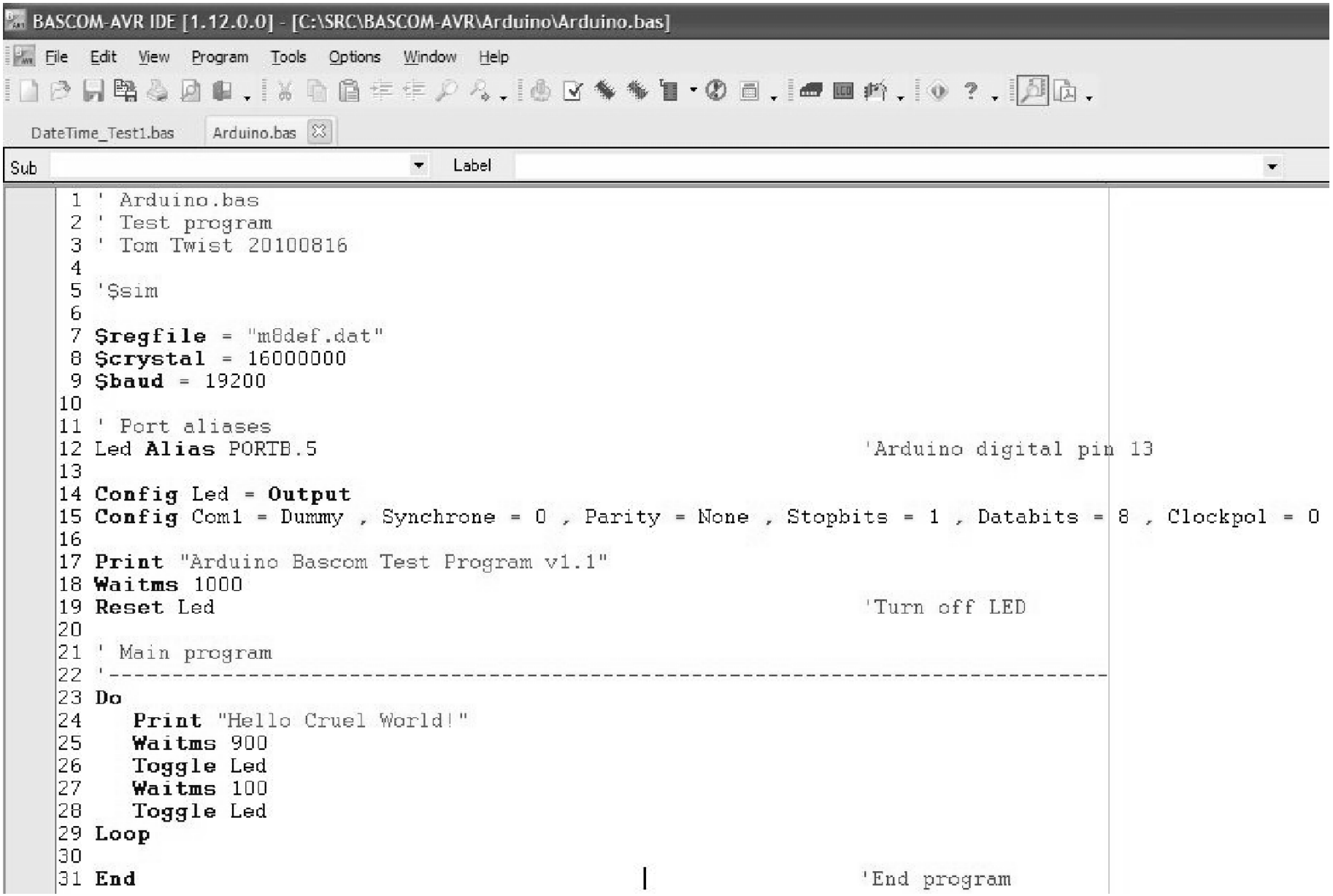

Blink | Arduino 0022
File Edit Sketch Tools Help
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
The circuit:
* LED connected from digital pin 13 to ground.
* Note: On most Arduino boards, there is already an LED on the board
connected to pin 13, so you don't need any extra components for this example.
Created 1 June 2005
By David Cuartielles
http://arduino.cc/en/Tutorial/Blink
based on an orginal by H. Barragan for the Wiring i/o board
*/
int ledPin = 13; // LED connected to digital pin 13
1

Arduino Language

- C like syntax, but simplified
- Abstracts the pin naming to numbers
- Trades efficience for ease of use
- Easy to learn, yet powerful
- Lots of example code
- Easy to reuse C-code from other projects
- Libraries can be written in C++
- Lots of libraries available

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts
void setup()  {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has powervoid
loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(500);  // wait for half a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(500);   // wait for half a second
}
```

BascomAVR IDE



BascomAVR Language

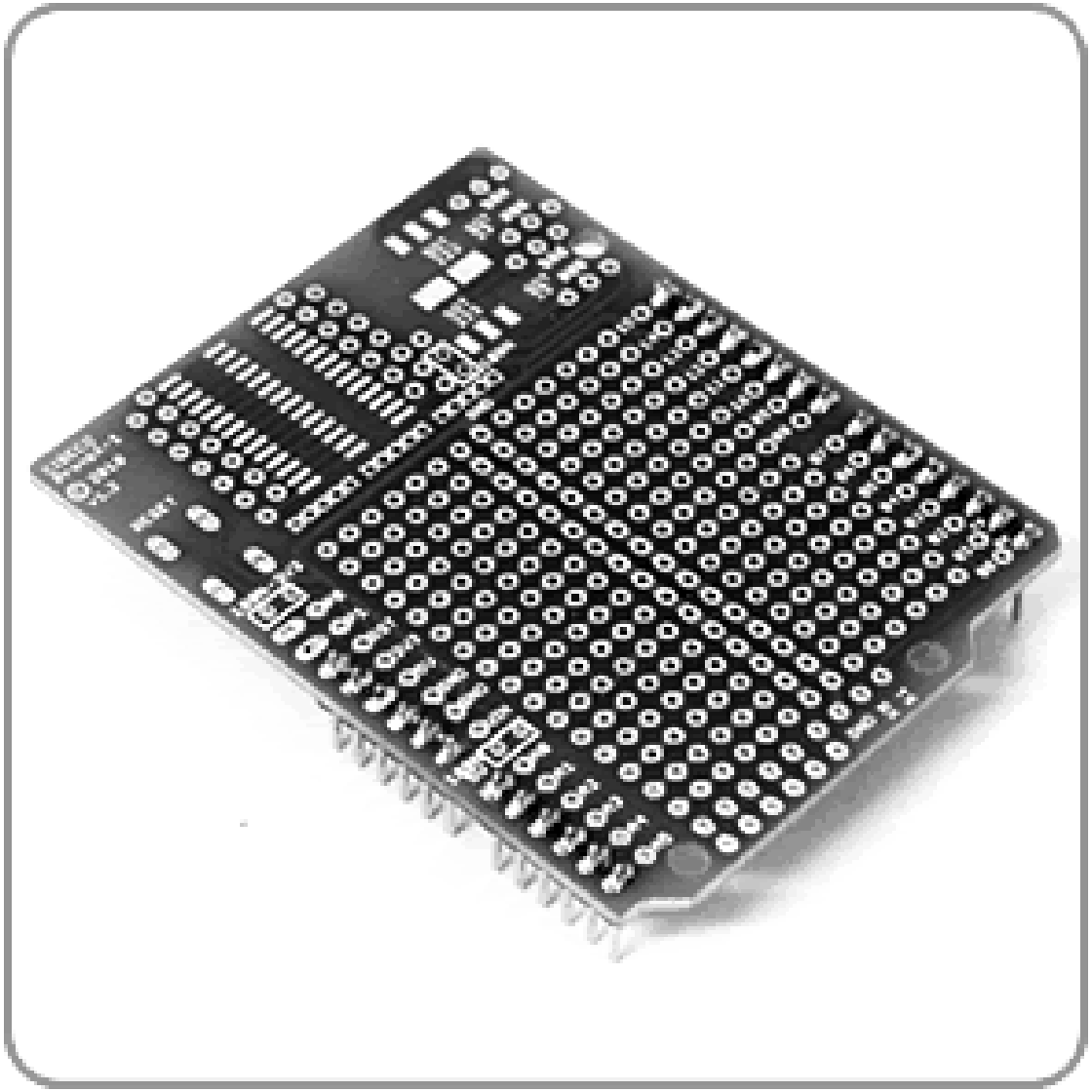
- Structured basic
- Uses AVR pin naming
- Easy to learn, yet powerful
- Very efficience, and compact
- Can use inline assembly code
- Built in simulator
- Lots of example code
- Must have paid version to make libraries
- Lots of libraries available

BascomAVR advantages

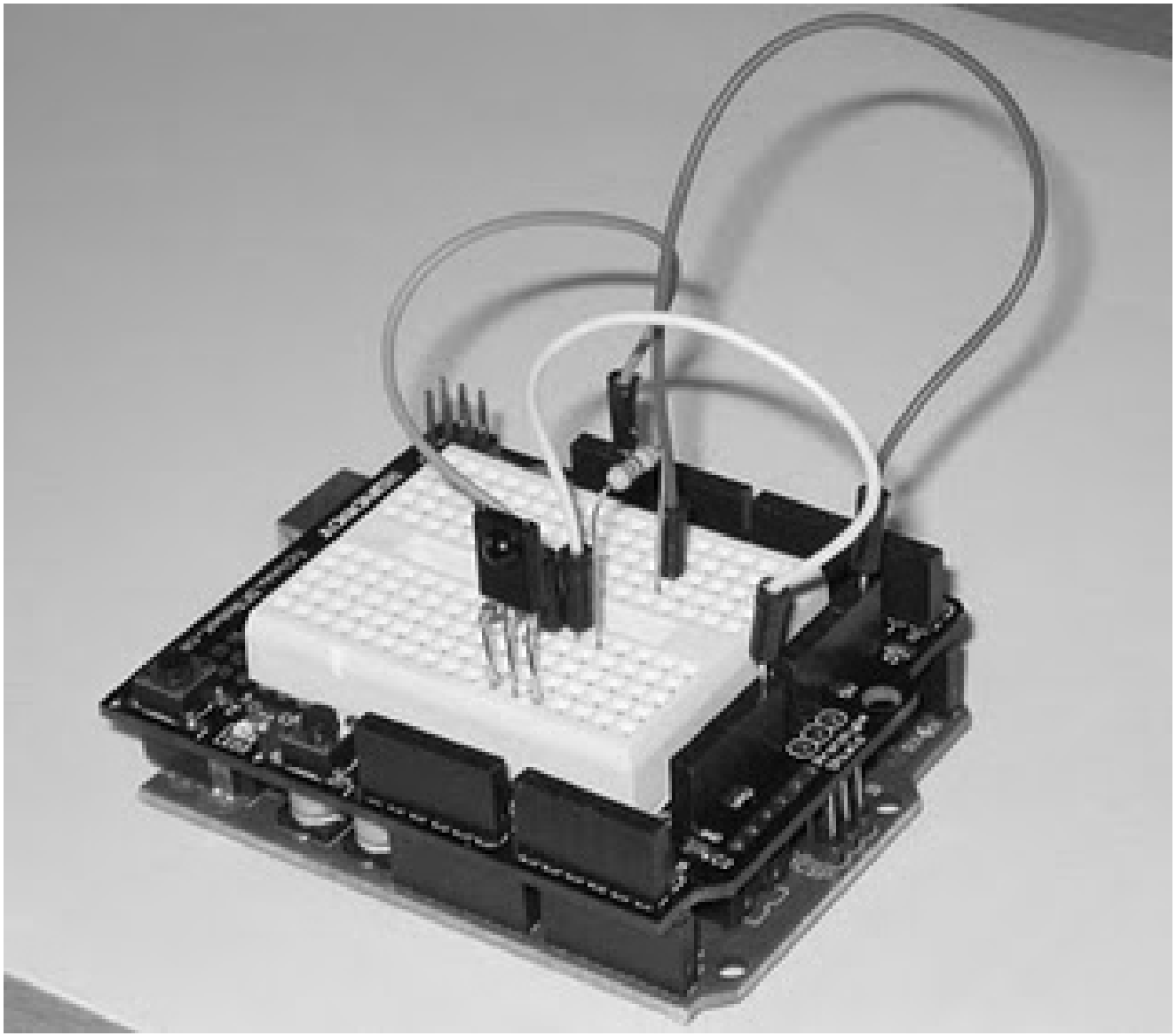
- Free demo version, up to 4kB code
- Supports most AVRs
- Commercial version is inexpensive
- Excellent support
- Active community
- Supports bootloaders
- Supports many programmers
- Can work with AVR Studio

```
$regfile = "m8def.dat"
$crystal = 16000000

Led Alias Portb.5                          'Arduino digital pin 13

Config Led = Output
Reset Led                                  'Turn off LED

' Main program

Do
   Waitms 500
   Toggle Led
   Waitms 500
   Toggle Led
Loop

End                                'End program
```

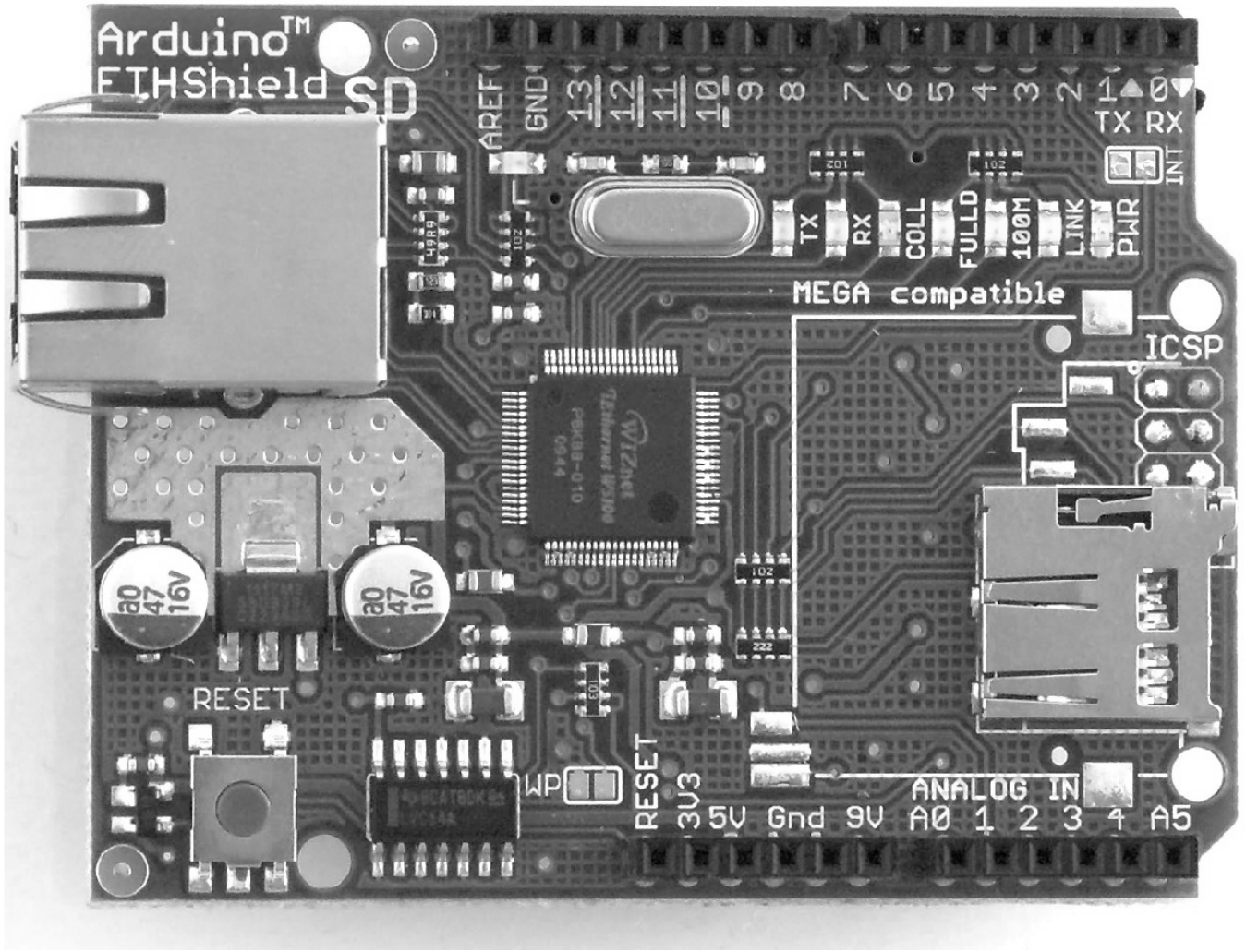
Prototype shield



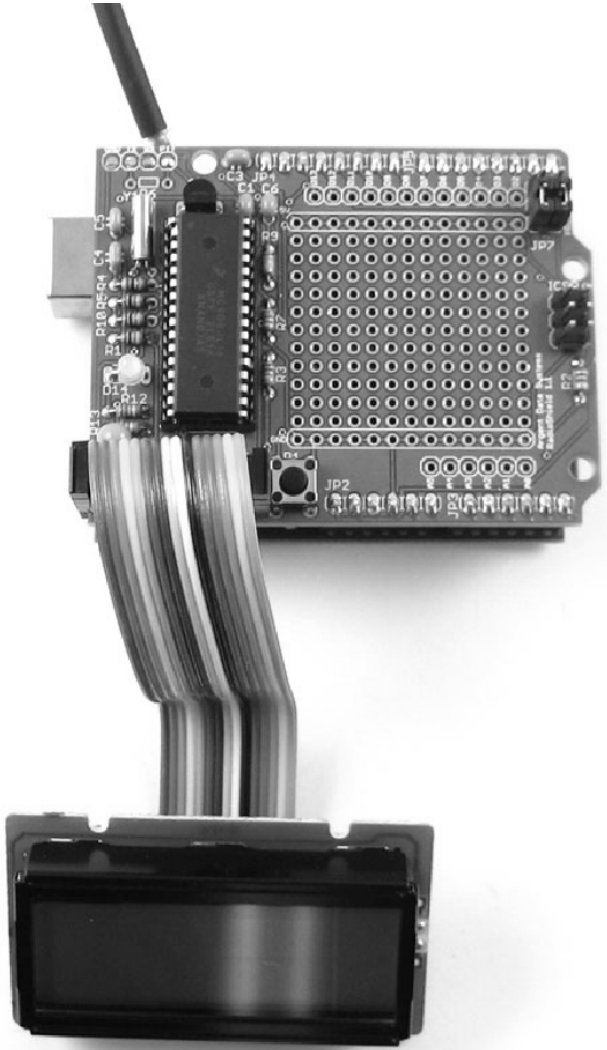
Proto shield w/mini breadboard



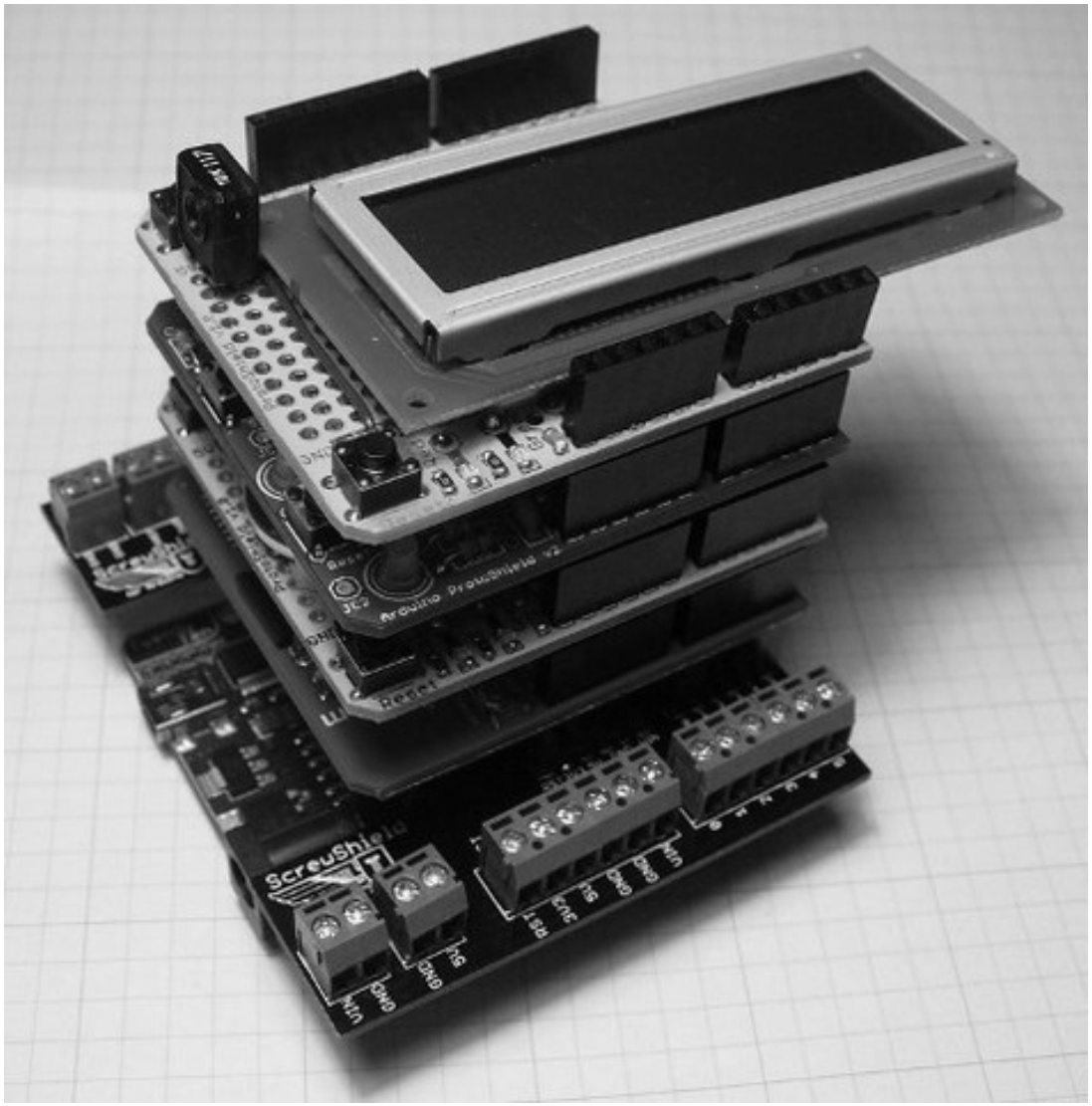
Ethernet shield w/micro-SD reader



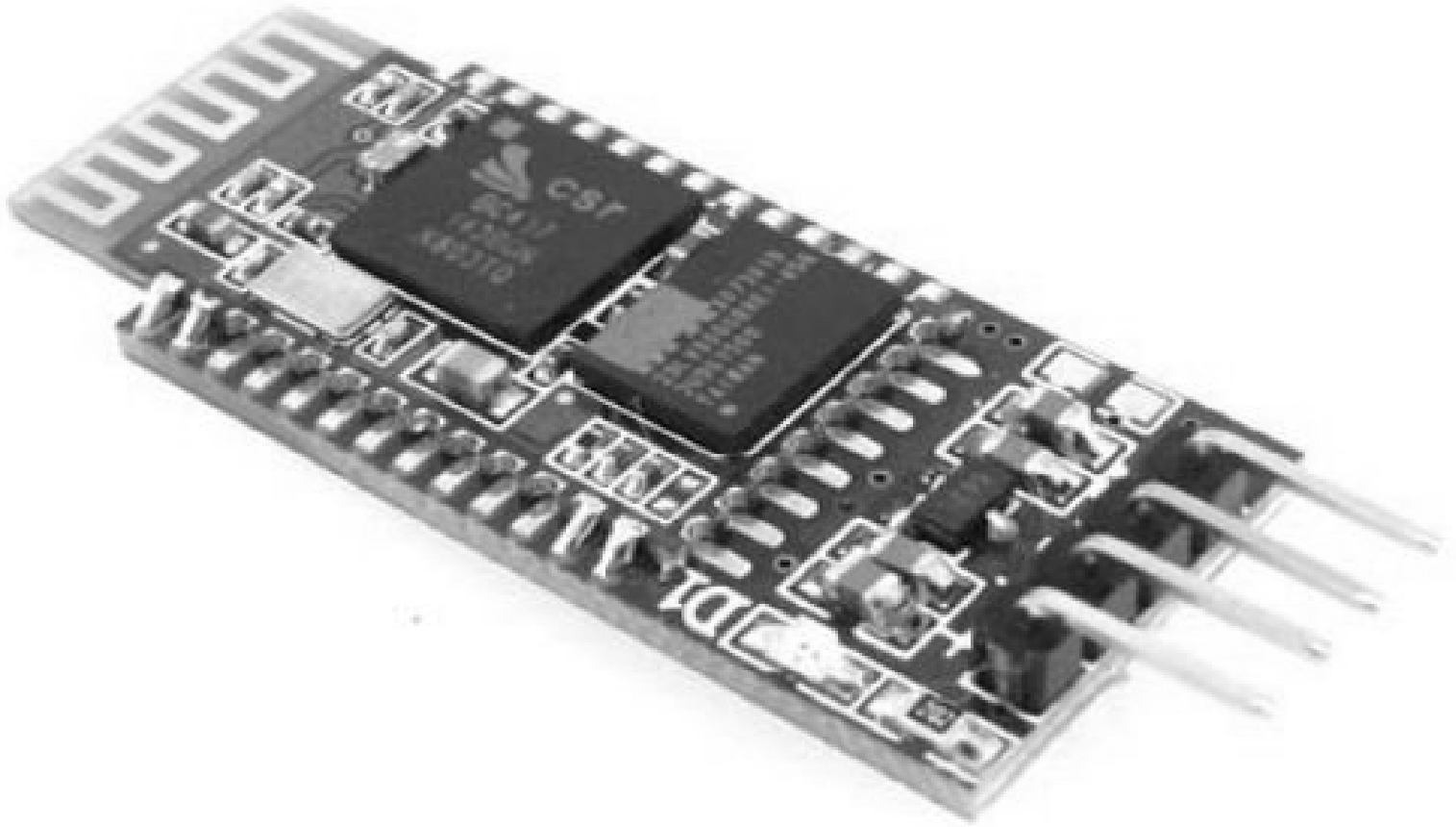
Argentdata radio shield for APRS



Extreme shield stacking

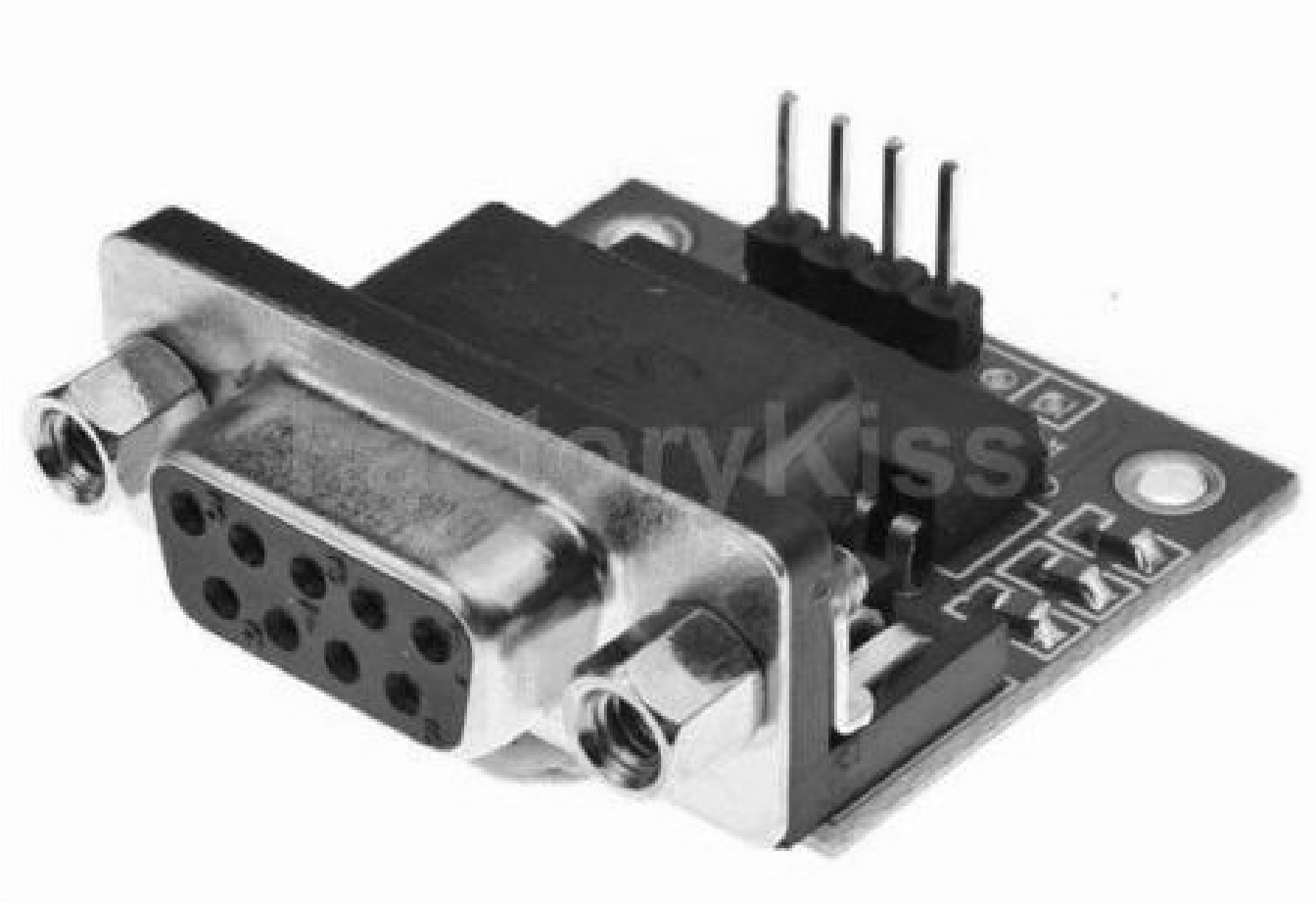


Bluetooth to TTL 5V module

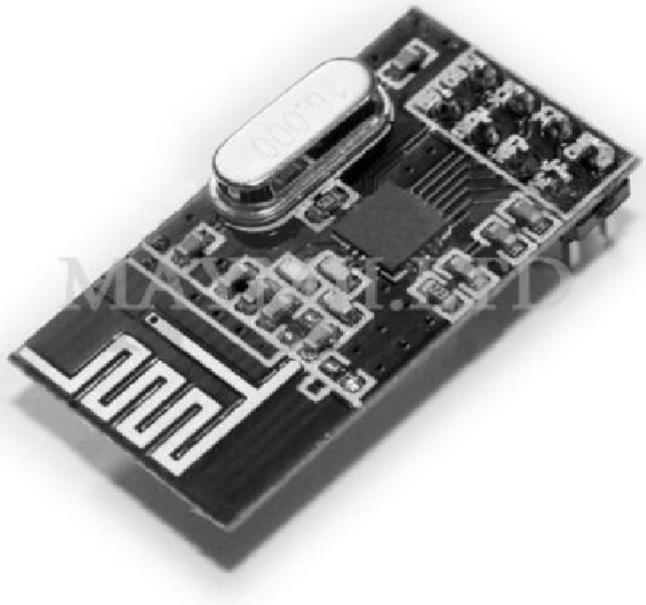


USB to TTL module

RS-232 to TTL module

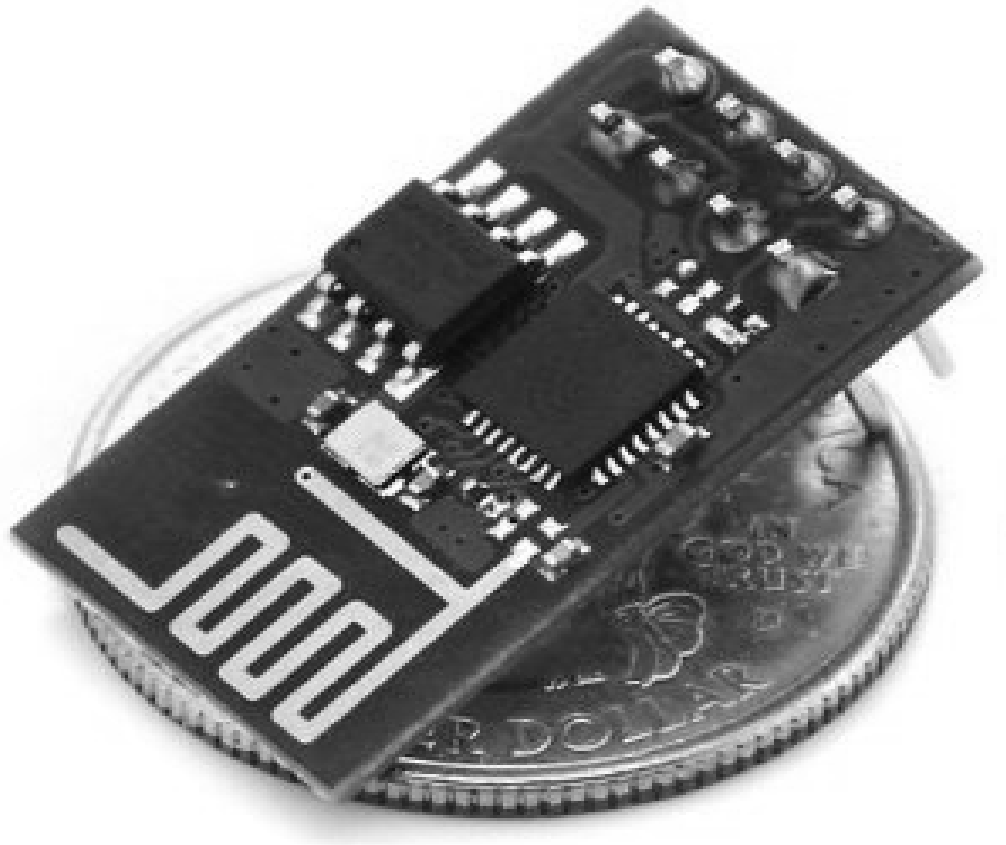


nRF24L01+ RF module



433 MHz ASK modules

ESP-01 ESP8266 WiFi module



Arduino pricing eBay

- Arduino Uno $6.50
- Arduino Mega2560 $11.95
- Arduino Nano $5.95
- Proto shield $3.95
- Ethernet shield w/SD $7.50
- Bluetooth module HC-06 $4.95
- USB to TTL module $0.95
- RS-232 to TTL module $0.99

Pricing RF modules eBay

- nRF24L01+ $0.95
- nRF24L01+ w/PA, ant $4.50
- 433MHz tx/rx modules $0.99
- ESP-01 WiFi module $3.9
- Breadboards:
- Full size $ 2.49
- Half size $ 1.65
- Mini $0.99

Pricing sensor modules eBay

- DHT22 temp & humidity $3.50
- BMP180 barometric $1.55
- DS18S20 temperature $0.99
- HX711 weight IF $0.99
- 1kg loadcell beam $5.49
- MPS20N0040D-D press. $2.49

Resources

- www.atmel.com/avr
- www.avrfreaks.net
- www.arduino.cc
- en.wikipedia.org/wiki/Arduino
- www.mcselec.com (BascomAVR)
- www.argentdata.com
- www.ebay.com
- www.sparkfun.com