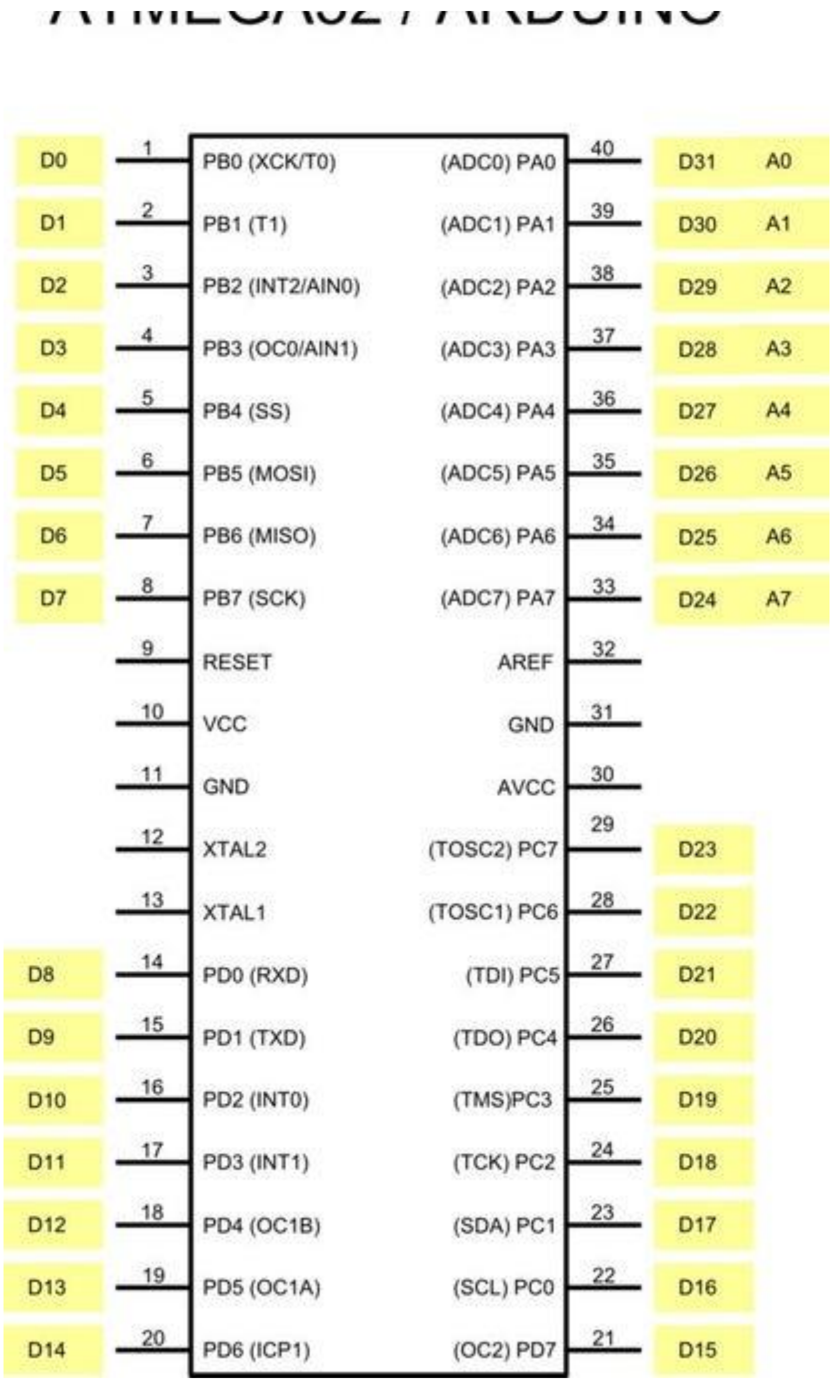
Using Atmega32 With Arduino IDE


ATMEGA32 / ARDUINO
D0 1 PB0 (XCK/T0) (ADC0) PA0 40 D31 A0
D1 2 PB1 (T1) (ADC1) PA1 39 D30 A1
D2 3 PB2 (INT2/AIN0) (ADC2) PA2 38 D29 A2
D3 4 PB3 (OC0/AIN1) (ADC3) PA3 37 D28 A3
D4 5 PB4 (SS) (ADC4) PA4 36 D27 A4
D5 6 PB5 (MOSI) (ADC5) PA5 35 D26 A5
D6 7 PB6 (MISO) (ADC6) PA6 34 D25 A6
D7 8 PB7 (SCK) (ADC7) PA7 33 D24 A7
9 RESET AREF 32
10 VCC GND 31
11 GND AVCC 30
12 XTAL2 (TOSC2) PC7 29 D23
13 XTAL1 (TOSC1) PC6 28 D22
D8 14 PD0 (RXD) (TDI) PC5 27 D21
D9 15 PD1 (TXD) (TDO) PC4 26 D20
D10 16 PD2 (INT0) (TMS)PC3 25 D19
D11 17 PD3 (INT1) (TCK) PC2 24 D18
D12 18 PD4 (OC1B) (SDA) PC1 23 D17
D13 19 PD5 (OC1A) (SCL) PC0 22 D16
D14 20 PD6 (ICP1) (OC2) PD7 21 D15

Over time I have used all kinds of Atmel microcontrollers in various projects.

One of the most suitable was ATmega32. I have a small collection of development boards for Atmega32/16, some bought as-is, some made on stripboard.

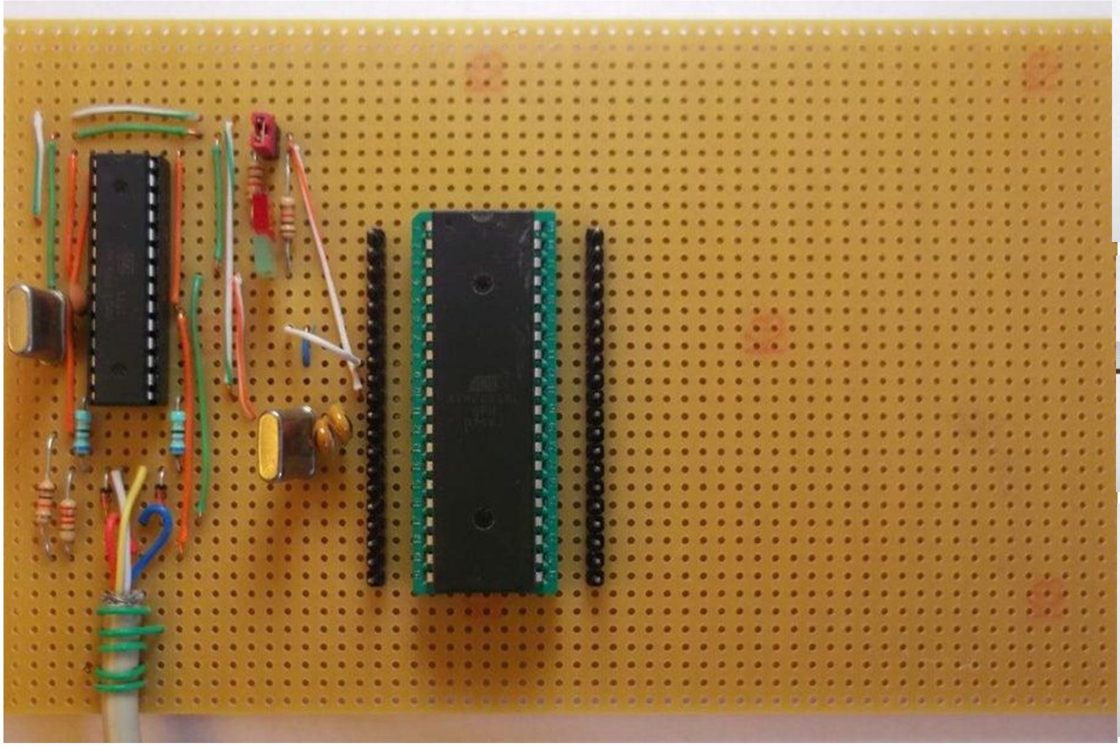
Although the original Arduino boards offers a pleasant experience and a rapid development of the projects, when it was necessary to "extract" these projects to give them final shape I hit the well-known problem: arduino board must be "locked" in the project.

It is not only the price. Although a microcontroller in thru-hole package is significantly bigger than SMD version, it is still much smaller than an Arduino board.

And still I want to use the Arduino IDE to benefit from the collection of (very well made) (ready-to-use) libraries.

So I gathered together information useful for this purpose.

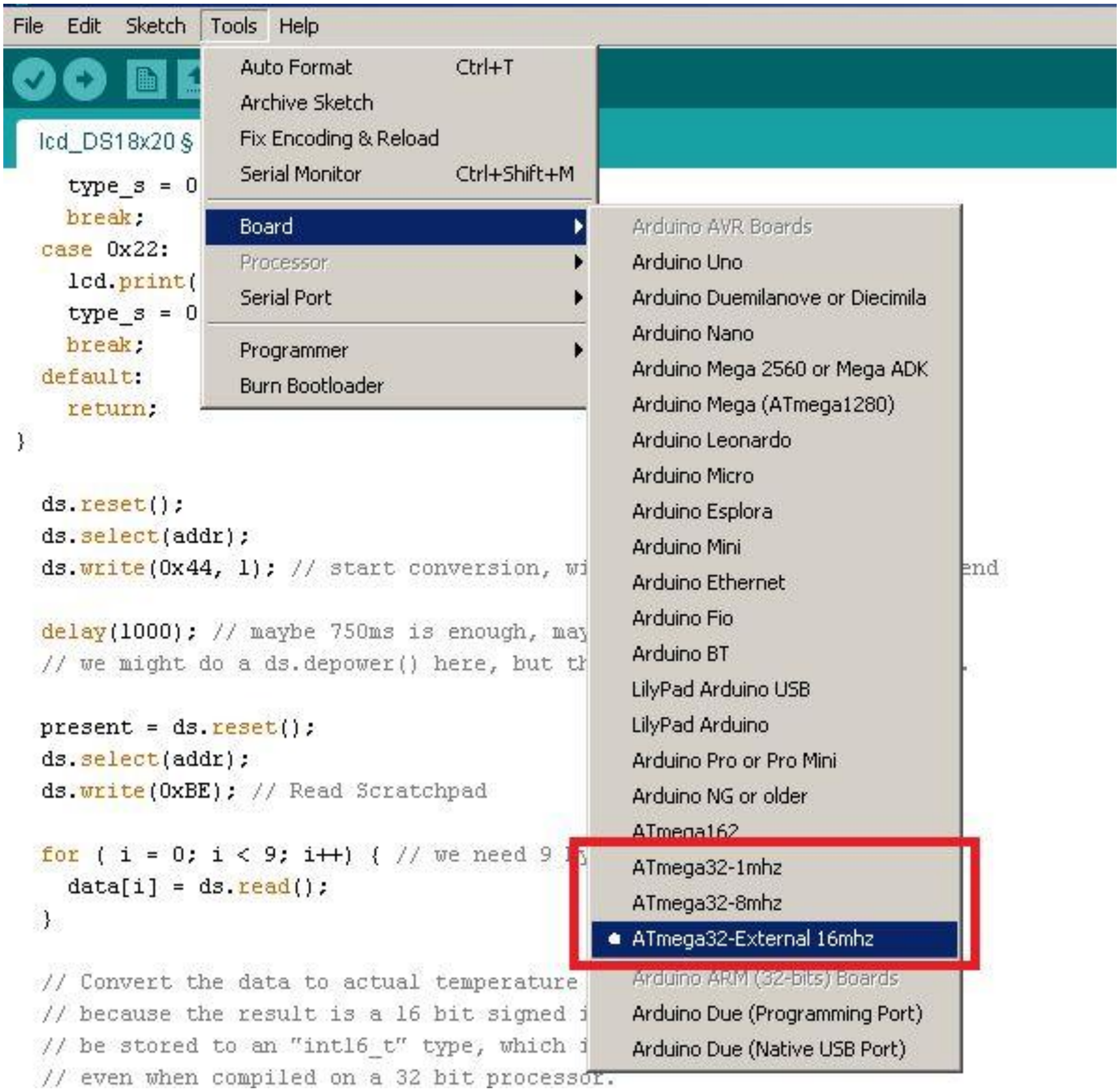
Step 1: Things You Will Need:



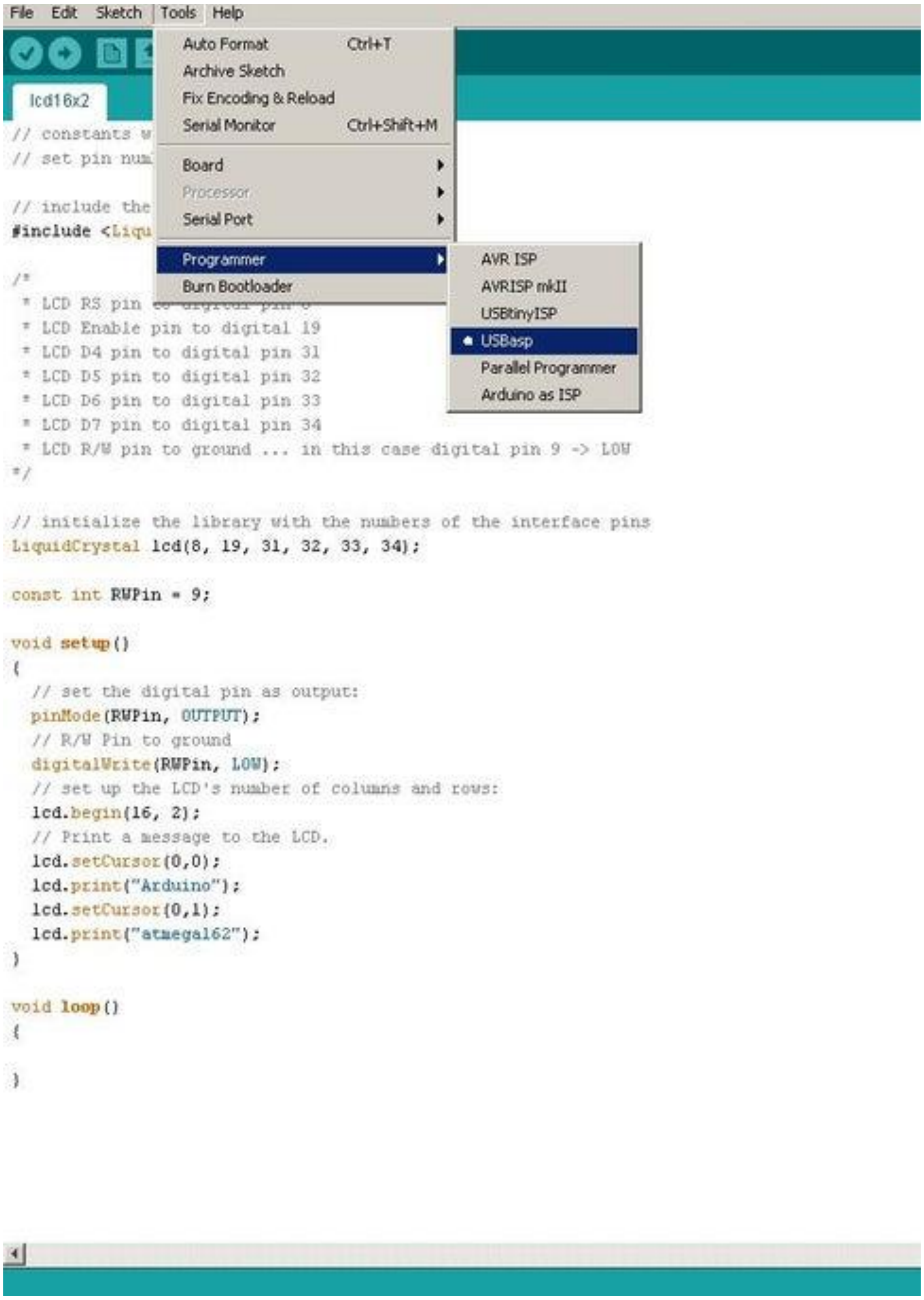
1. Atmega32 microcontroller. :)
2. Programmer (any programmer recognized by Arduino IDE). I use the USBasp programmer.
3. Minimal setup for power-up Atmega32 microcontroller (breadboard, stribpoard or a classical development board).

   We do not need all, photos above are for exemplification only.

Step 2: Software Set-up



```
    type_s = 0
    break;
  case 0x22:
    lcd.print(
    type_s = 0
    break;
  default:
    return;
}

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1); // start conversion, wi                              end

  delay(1000); // maybe 750ms is enough, may
  // we might do a ds.depower() here, but th

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE); // Read Scratchpad

  for ( i = 0; i < 9; i++) { // we need 9 by
    data[i] = ds.read();
  }

  // Convert the data to actual temperature
  // because the result is a 16 bit signed i
  // be stored to an "int16_t" type, which i
  // even when compiled on a 32 bit processor.
```



```
// constants w
// set pin num

// include the
#include <Liqu

/*
 * LCD RS pin to digital pin 8
 * LCD Enable pin to digital 19
 * LCD D4 pin to digital pin 31
 * LCD D5 pin to digital pin 32
 * LCD D6 pin to digital pin 33
 * LCD D7 pin to digital pin 34
 * LCD R/W pin to ground ... in this case digital pin 9 -> LOW
*/

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 19, 31, 32, 33, 34);

const int RWPin = 9;

void setup()
{
  // set the digital pin as output:
  pinMode(RWPin, OUTPUT);
  // R/W Pin to ground
  digitalWrite(RWPin, LOW);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.setCursor(0,0);
  lcd.print("Arduino");
  lcd.setCursor(0,1);
  lcd.print("atmega162");
}

void loop()
{

}
```

1. Download zip file attached to this instructables.
2. Locate file boards.txt in arduino folder ...arduino-1.5.2\hardware\arduino\avr\boards.txt
3. Append information from zip archive **boards.txt** to original **boards.tx**t

4. Make a folder named **mega32** in ...arduino-1.5.2\hardware\arduino\avr\variants
5. Copy file **pins_arduino.h** from the zip into folder created at step 4.
6. Start Arduino IDE and select board.
7. Select programmer

Later edit:

I removed link to github projet where I originally found. As long as there source changed and not fit this instructable.

After a message from the **author: Eric Conner,** I put back link to github project where I originally found this library:

https://github.com/eaconner/ATmega32-Arduino

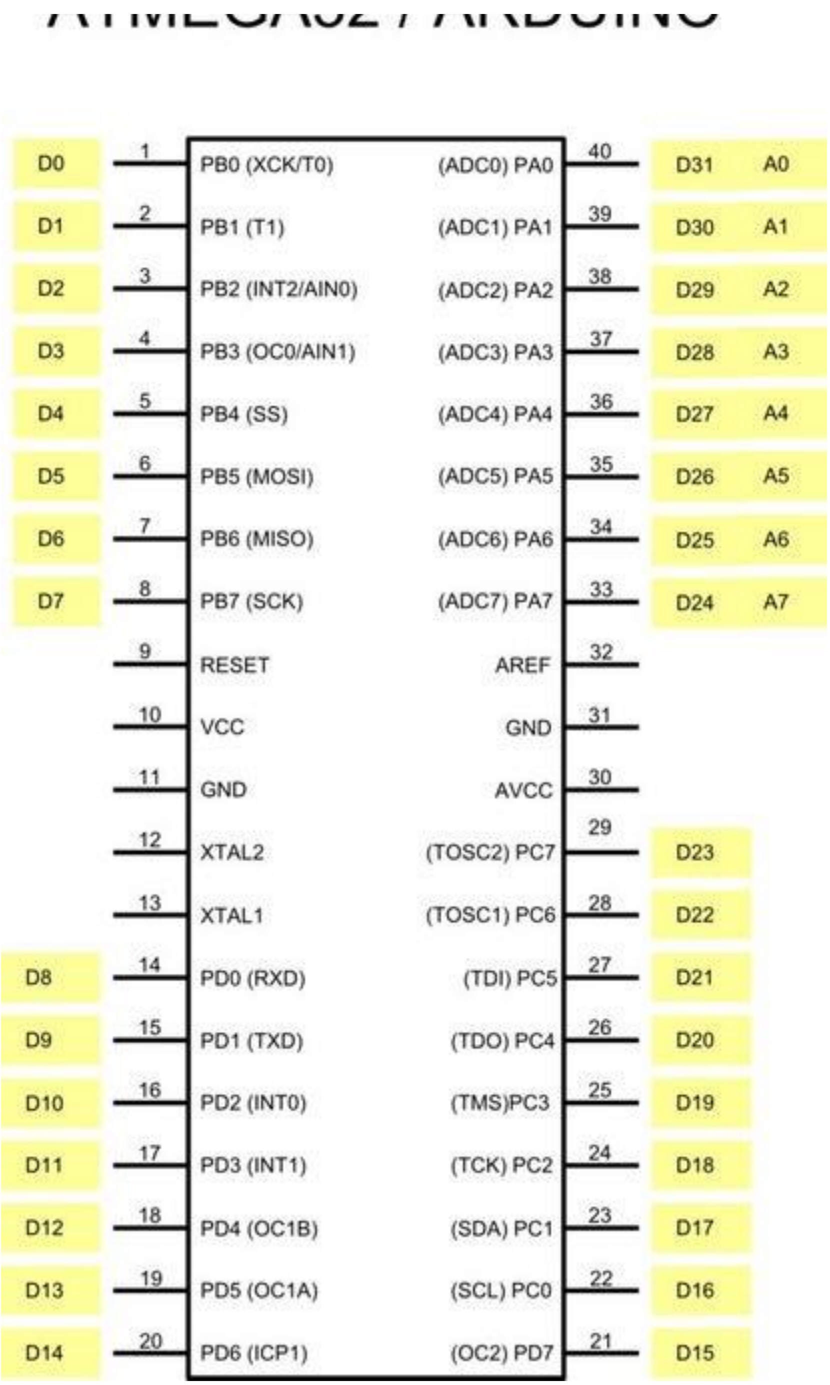
Note: Zip file attached to this article is an older version of above library. This will work as this instructable was written.

Over time, after some feedback I noticed that source from github has changed.

Also, in comments you will see references to some errors, and how to to correct it.

This article and answers from comments is relate to version attached here, not github (newer) version.

Step 3: Done.



ATMEGA32 / ARDUINO
D0 1 PB0 (XCK/T0) (ADC0) PA0 40 D31 A0
D1 2 PB1 (T1) (ADC1) PA1 39 D30 A1
D2 3 PB2 (INT2/AIN0) (ADC2) PA2 38 D29 A2
D3 4 PB3 (OC0/AIN1) (ADC3) PA3 37 D28 A3
D4 5 PB4 (SS) (ADC4) PA4 36 D27 A4
D5 6 PB5 (MOSI) (ADC5) PA5 35 D26 A5
D6 7 PB6 (MISO) (ADC6) PA6 34 D25 A6
D7 8 PB7 (SCK) (ADC7) PA7 33 D24 A7
9 RESET AREF 32
10 VCC GND 31
11 GND AVCC 30
12 XTAL2 (TOSC2) PC7 29 D23
13 XTAL1 (TOSC1) PC6 28 D22
D8 14 PD0 (RXD) (TDI) PC5 27 D21
D9 15 PD1 (TXD) (TDO) PC4 26 D20
D10 16 PD2 (INT0) (TMS)PC3 25 D19
D11 17 PD3 (INT1) (TCK) PC2 24 D18
D12 18 PD4 (OC1B) (SDA) PC1 23 D17
D13 19 PD5 (OC1A) (SCL) PC0 22 D16
D14 20 PD6 (ICP1) (OC2) PD7 21 D15

It's done.

Now you can try some simple examples that are already in Arduino.

Be very careful about correlation between ARDUINO pin and microcontroller pin.

Here is blink example: Files->Examples->Basics->Blink

Pin13 Arduino == Pin19 (PD5) Atmega32

```
/*Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

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

// 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
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

After some comments I found two errors in the file pins_arduino.h

So I’ll post here the errors and correct values.

erroneous definition of SCL and SDA

```
const static uint8_t SDA = 8; //wrong

const static uint8_t SCL = 9; //wrong
```

must be changed in:

```
const static uint8_t SDA = 17; //correct

const static uint8_t SCL = 16; //correct
```

Since I am not the author of the project on github, it is subject to change beyond my control.

So please use code attached to this instructable and make above modifications.

UPDATE:

For **Serial library to work properly** must be made following changes to the file **HardwareSerial.cpp**
In ...\arduino-1.5.8\hardware\arduino\avr\cores\arduino\HardwareSerial.cpp

will replace:

```
#if defined(__AVR_ATmega8__)

        config |= 0x80; // select UCSRC register (shared with UBRRH)

#endif
```

with:

```
#if defined(__AVR_ATmega8__) || defined(__AVR_ATmega32__) || defined(__AVR_ATmega16__)

        config |= 0x80; // select UCSRC register (shared with UBRRH)

#endif
```