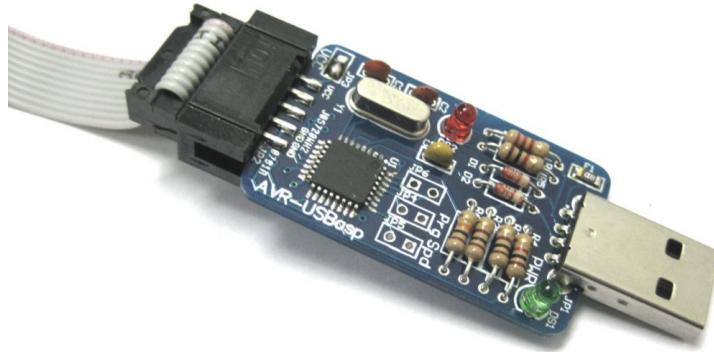




Cytron USBASP Programmer

AVR USBASP



User's Manual

V1.0

October 2011

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologies's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Index

1. Introduction	1
2. System Overview	3
3. Packing List	4
4. Board Layout and specification	5
5. Installation	8
6. Example Schematic	12
7. Software Using	13
7.1 AVRdude	13
7.1.1 Programming for LED Blinking	13
7.1.2 Programming for Bootloader	16
7.2 Arduino Software	19
8. Warranty	26

1. Introduction

AVR USBasp is a USB in-circuit programmer and it can use to program most of the Atmel AVR controllers. It simply consists of an ATmega8 and a couple of passive components such as resistors, capacitors, LEDs and ect. The programmer uses a firmware-only USB driver and there is no special USB controller is needed. By using AVR USBasp, it is easier and simpler, it just needs one step to finish the process which is to connect the AVR USBasp with computer and with microcontroller, then program it. AVR USBasp has been designed with capabilities and features of:

- USBasp works under multiple platforms. Linux, Mac OS X and Windows are tested.
- Its speed for the programming is up to 5kBytes/sec.
- Its SCK option is supported to the targets with low clock speed (<1.5Mhz).

USBasp Method

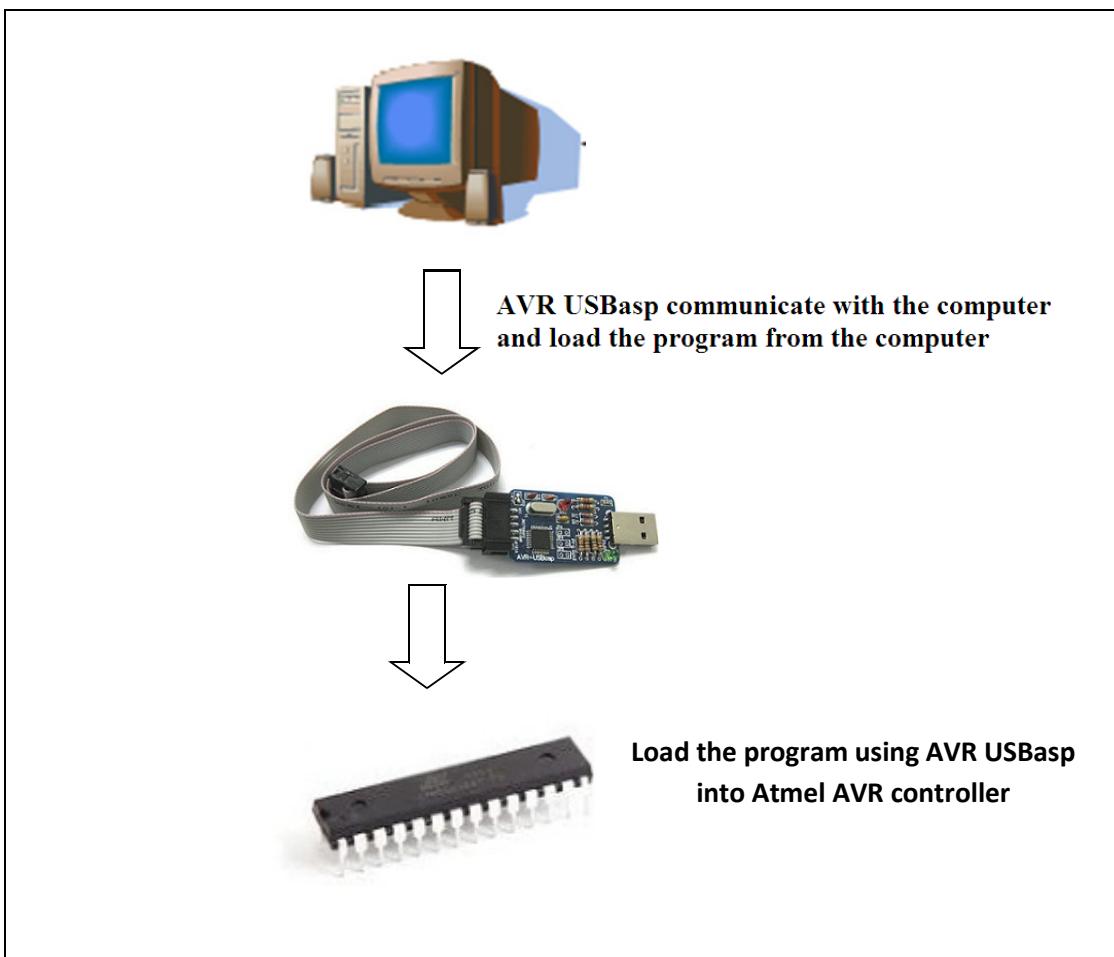


Table below shows some of the microcontrollers that are supported by the AVR USBasp Programmer, other than the list provided below, there is also having others series of microcontrollers which are also supported by AVR USBasp Programmer.

Mega Series				
ATMEGA48	ATMEGA8	ATMEGA88	ATMEGA8515	ATMEGA8535
ATMEGA16	ATMEGA162	ATMEGA163	ATMEGA164	ATMEGA165
ATMEGA168	ATMEGA169	ATMEGA169P	ATMEGA32	ATMEGA324
ATMEGA325	ATMEGA3250	ATMEGA328P	ATMEGA329	ATMEGA3290
ATMEGA64	ATMEGA640	ATMEGA644	ATMEGA645	ATMEGA6450
ATMEGA649	ATMEGA6490	ATMEGA128	ATMEGA1280	ATMEGA1281
ATMEGA2560	ATMEGA2561			

Tiny Series				
ATTiny12	ATTiny13	ATTiny15	ATTiny24	ATTiny25
ATTiny26	ATTiny2313	ATTiny44	ATTiny45	ATTiny84
ATTiny85				
Classic Series				
AT90S2313	AT90S2323	AT90S2343	AT90S1200	AT90S8515
Can Series				
AT90CAN32	AT90CAN64	AT90CAN128		
PWM Series				
AT90PWM2	AT90PWM3			

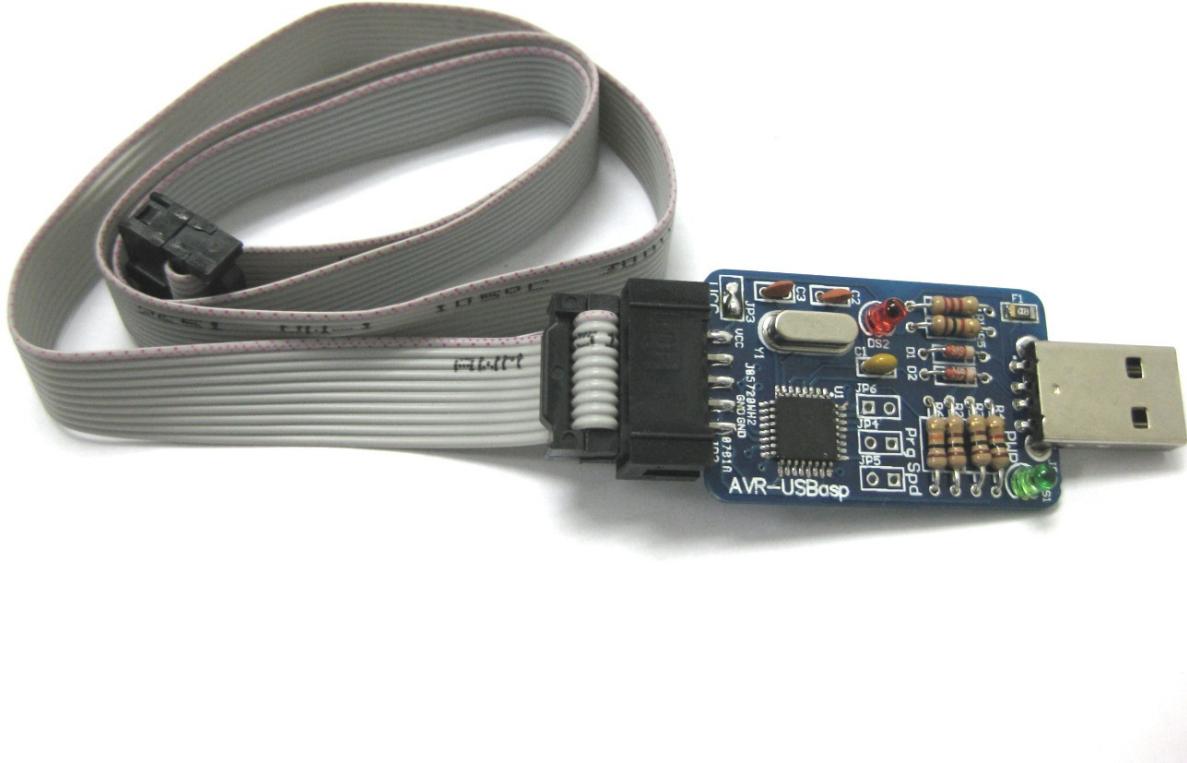
2. System Overview



Cautions: There is a 5V which supply directly from USB port of computer to AVR USBasp; it is advised not to use this power source to power application circuit or device. Wrong connection such as wrong polarity, wrong voltage, shorted might permanently damage computer.

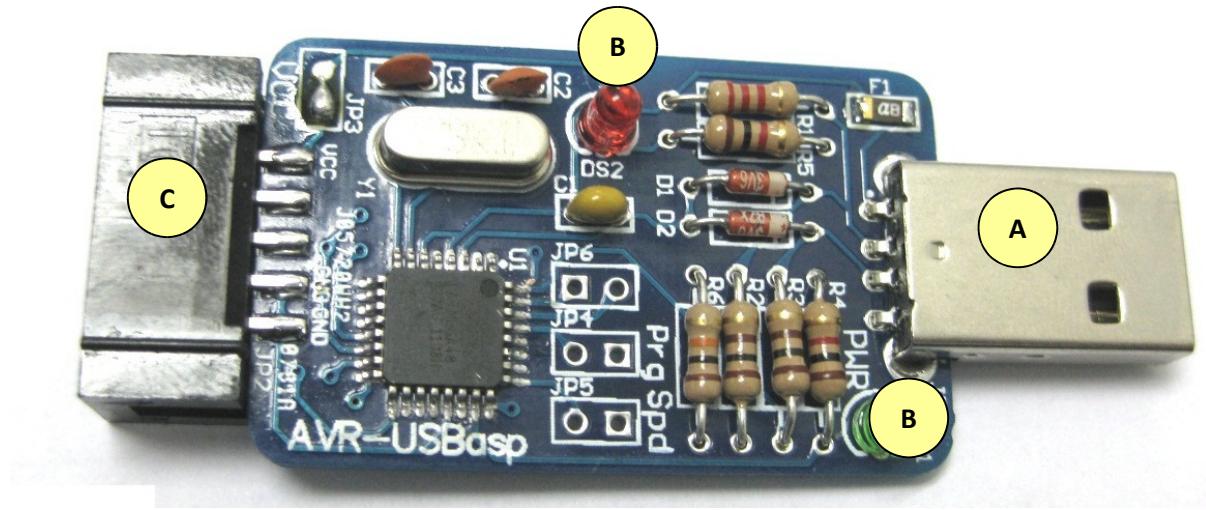
3. Packing List

Please check the parts and components according to the packing list. If there are any parts missing, please contact us at sales@cytron.com.my immediately.



1. 1 x AVR-USBASP
2. 10 ways programmer rainbow cable

4. Board Layout and Specification



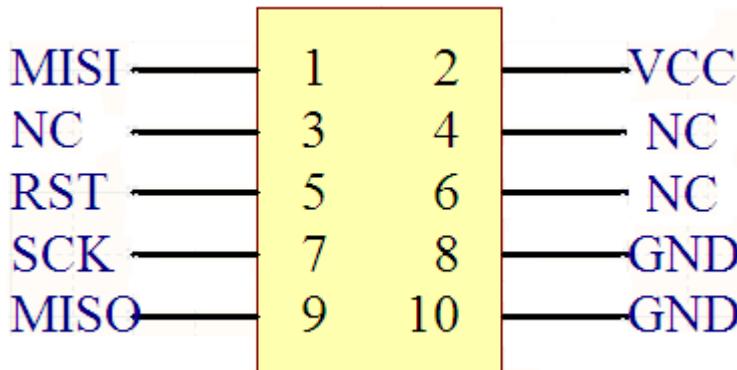
Label	Function
A	USB A type (male)
B	LEDs
C	10 pins IDC connector for interface to microcontroller
D	JP4 and JP5 for programmer and speed
E	Vcc supply

A – USB A type (male). This is for USB connection.

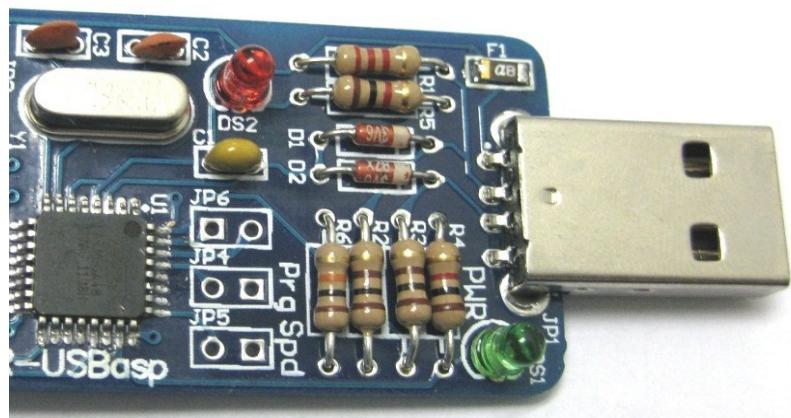
B – The USBASP programmer has 2 LEDs. The function is listed below:

- LED 1 (Green LED) – Power
- LED 2 (Red LED) – Programmer communicating with the target device

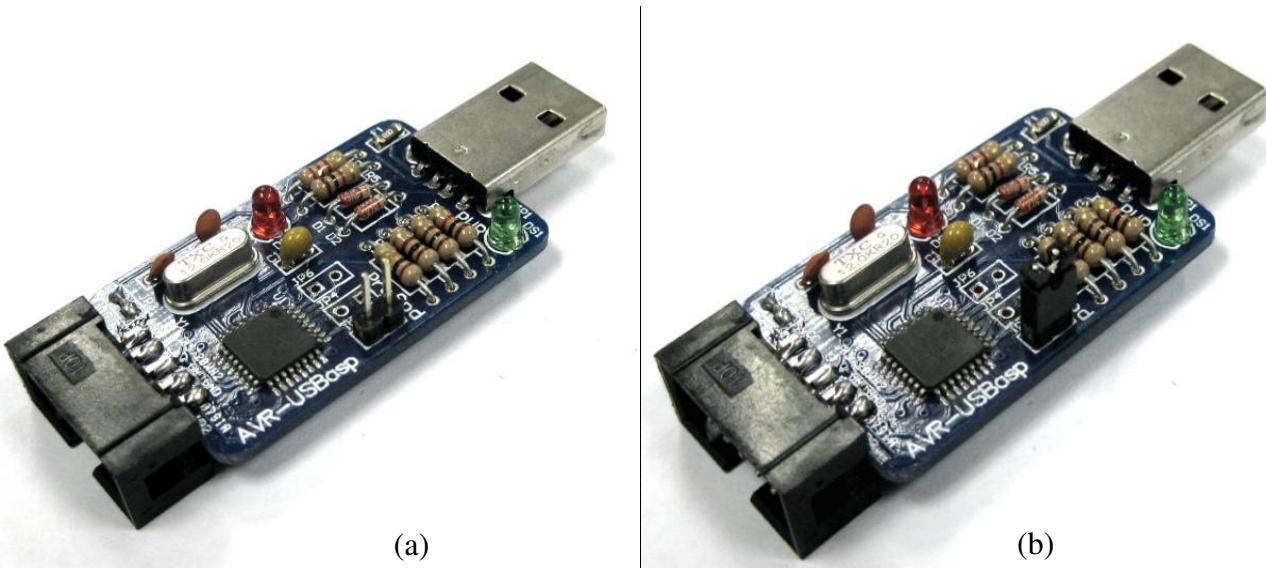
C – The 10 pins ISP connection provides an interface to the microcontroller. This interface uses a 10 pin IDC connector and the pinout is shown in figure below.



D – JP4 is for programmer and JP5 is for speed. Normally we are not using JP4 because JP4 function is to update the firmware on it. JP5 is only used for new microcontroller. It will not be used for others than new microcontroller because we want to increase the speed of the process of programming.



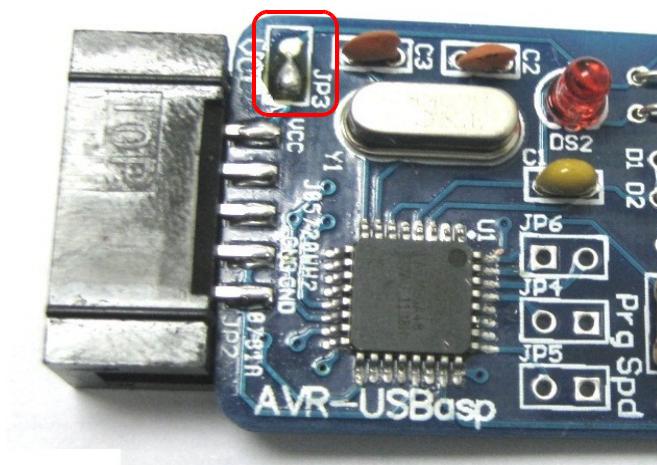
Below is example picture to show JP5 used for new microcontroller. Users may solder header pin on JP5 and use mini jumper to connect it.



If JP5 not used (not new microcontroller), users may remove the mini jumper.



E – Vcc supply is to supply power from USB to Atmega target board. User may desolder JP3 if user didn't want to use it.

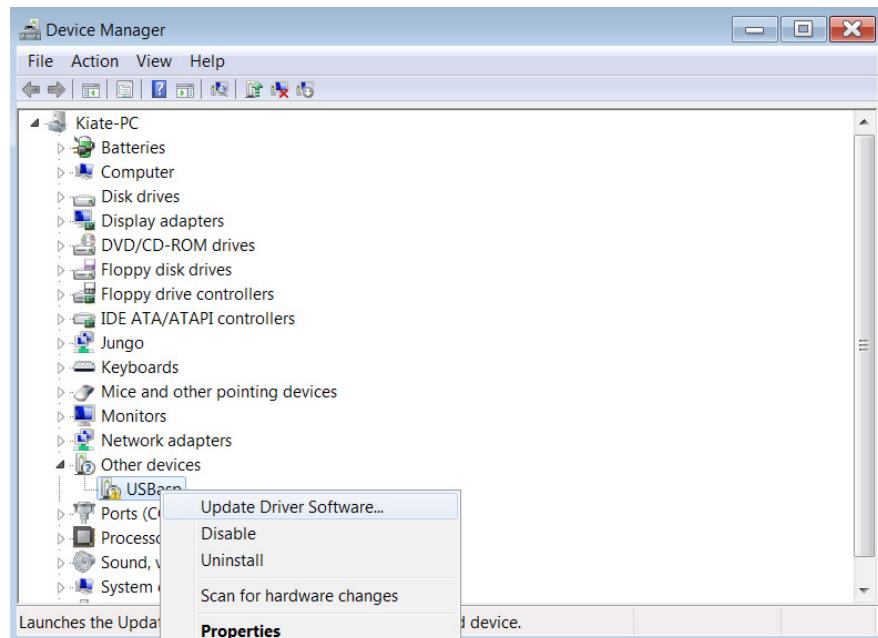


5. Installation

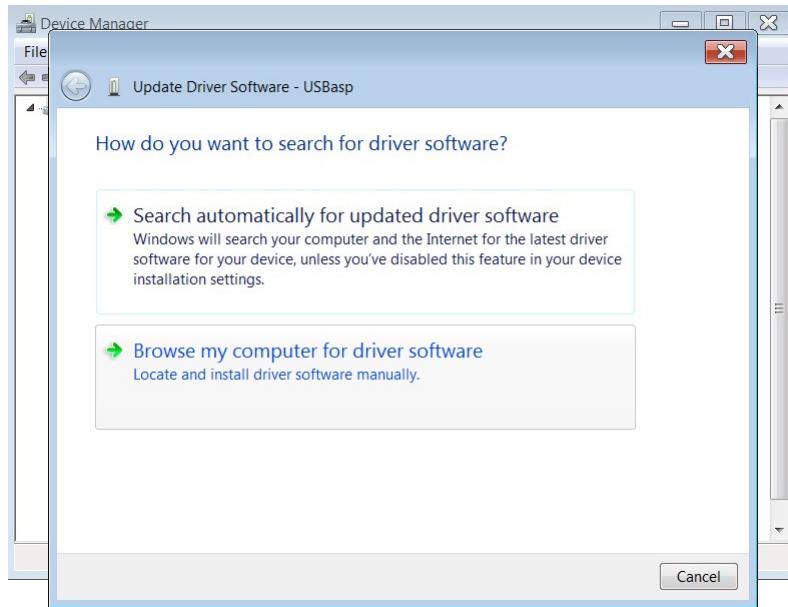
In order to complete the installation, you need to follow several steps:

The first step is to direct connect the AVR USBasp programmer to the USB port of your PC. For the AVR USBasp programmer will work on a wide variety of operating systems, this procedure will only focus on Window Seven.

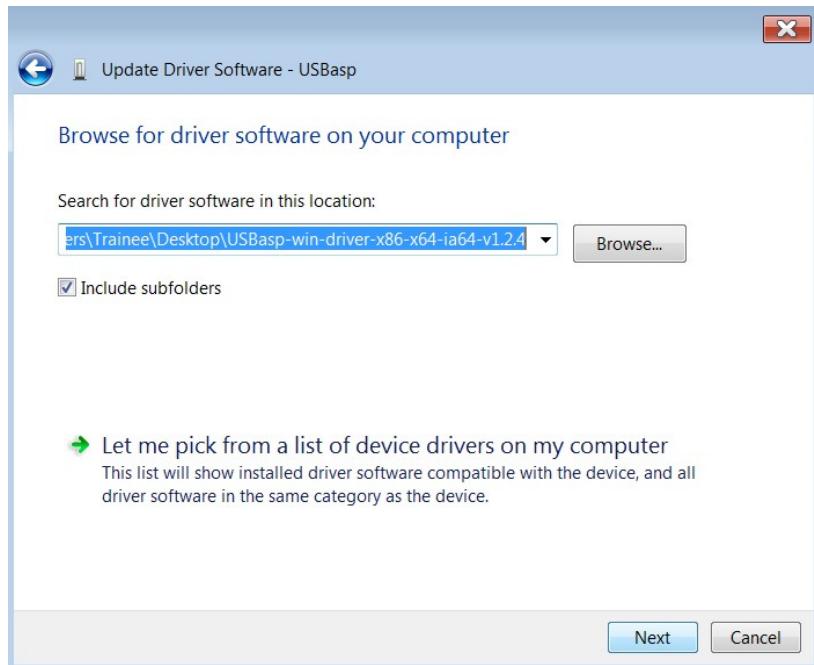
1. Required items
 - a. AVR USBasp programmer
2. AVR USBasp drivers can be downloaded from Cytron Technologies Sdn. Bhd. product website page.
3. Procedure to install the AVR USBasp programmer
 - a. Insert the programmer into and available USB port in your PC.
 - b. Go into the device manager and find the entry for the USBasp and it should be displayed with a yellow alert icon on it. Then right click on the device and select “Update Driver Software”.



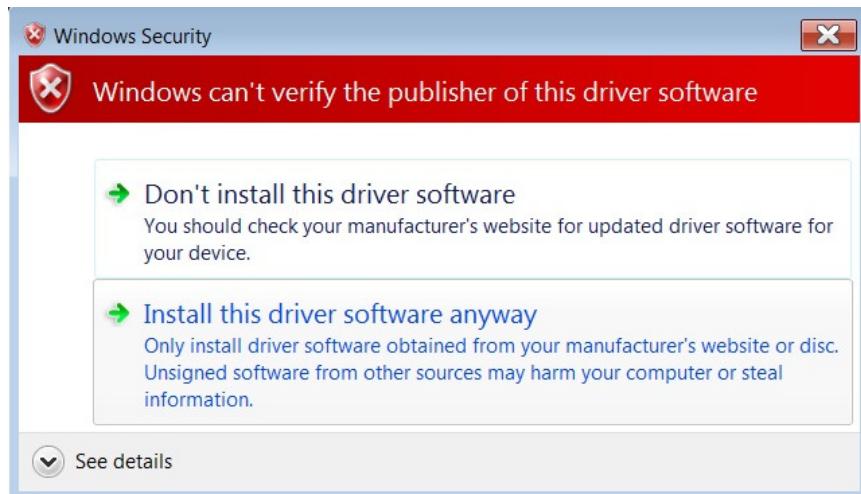
- c. After you left click the “Update Driver Software”, it will come out with “How do you want to search for driver software?” Then choose the second one which is “Browse my computer for driver software” and click into it.



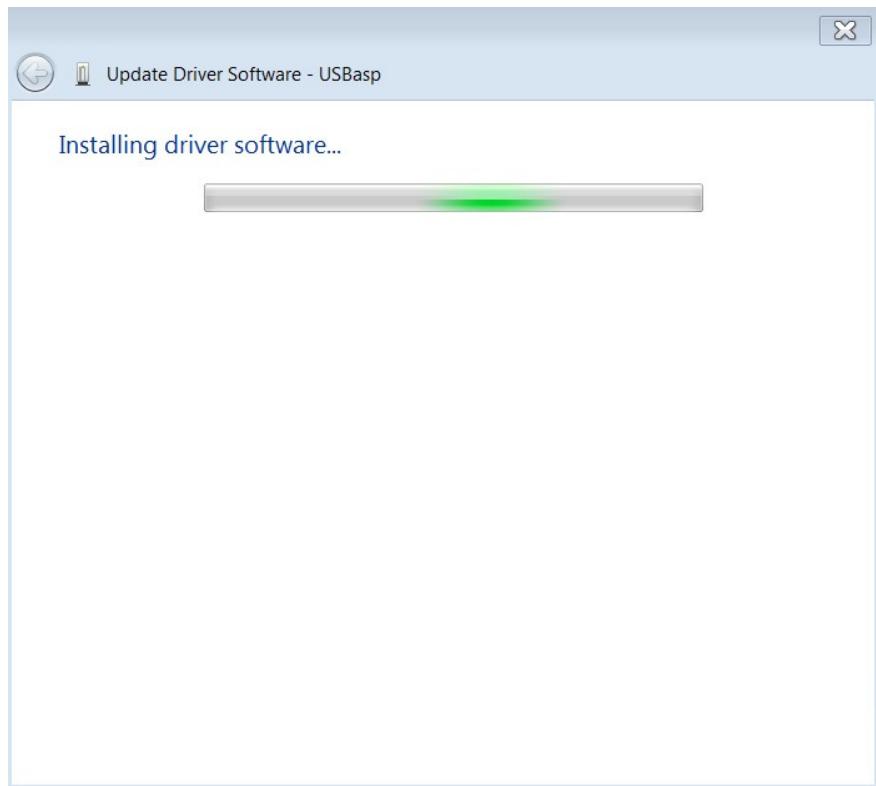
- d. After that, you will see the screen which will prompt out “Browse for driver software on your computer”. In this step, you need to select the folder where you unzipped the driver files then click “Next”.



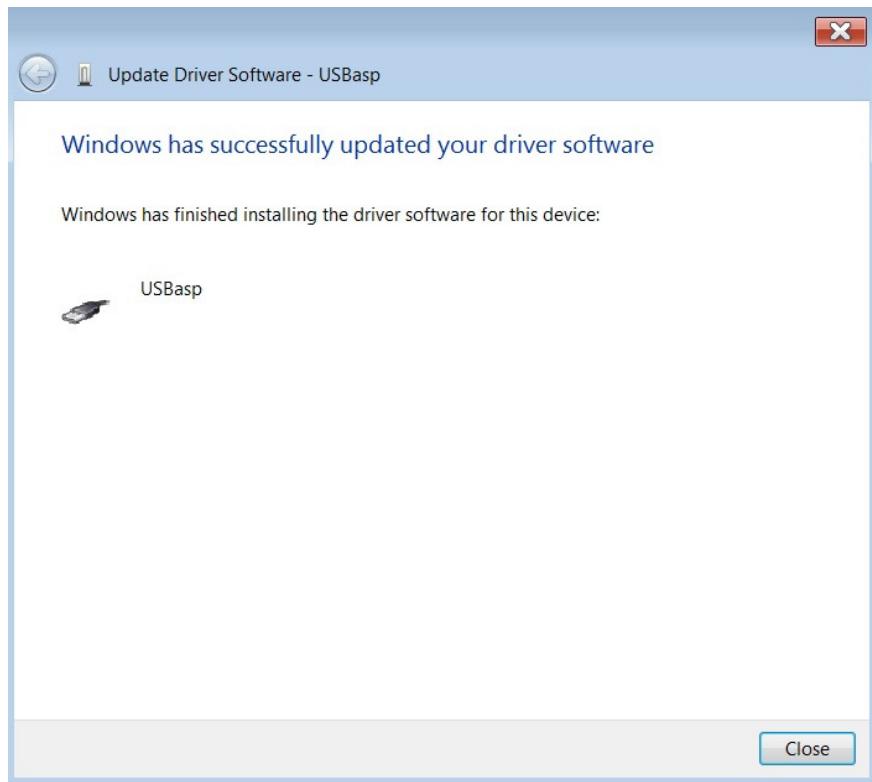
- e. Next, the windows will prompt out a “Windows Security” with a red warning dialog. Do not worry about it, and just click “Install this driver software anyway” and the driver will install.



- f. After click it, the next step is to wait a few seconds to let your computer to process the installation of driver software.

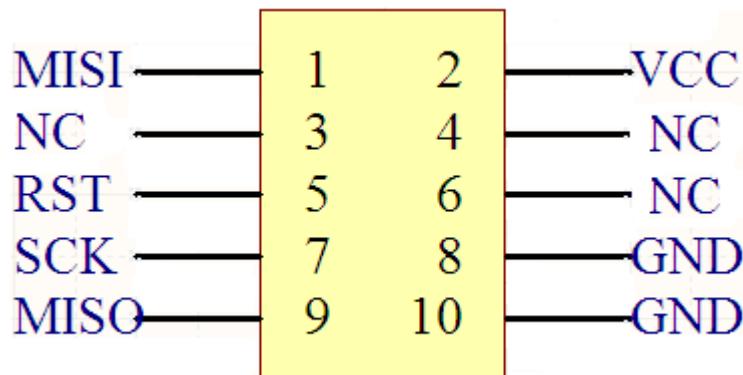


- g. Now, you can use the AVR USBasp to do the programming for the microcontroller.

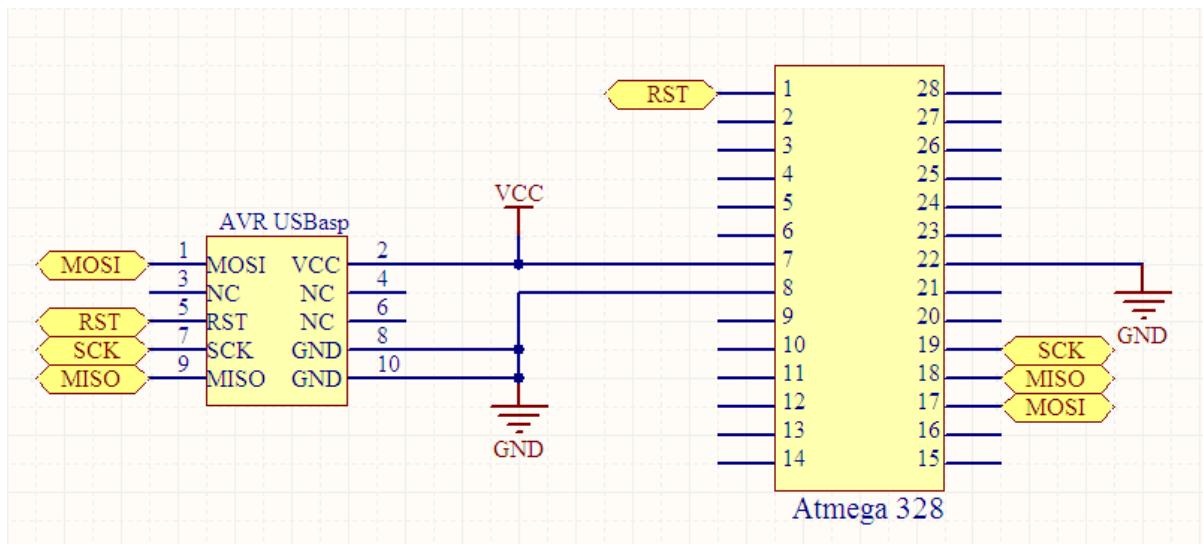


6. Example Schematic

In this example, we will show the port layout for 10 pins ISP connection of AVR USBasp and the connection between AVR USBasp with the microcontroller Atmega328.



10 pins ISP connection of AVR USBasp



Schematic of AVR USBasp with Atmega328

7. Software Using

There are varieties of software which can be work also for the programmer AVR USBasp. These are including:

- AVRdude – Version 5.2 or later. AVRdude is available for many platforms.
- Arduino Software – Normally we are using this software to program most of the Arduino board.
- Khazama AVR Programmer – An AVRdude GUI for MS Windows.
- BASCOM-AVR – Version 1.11.9.6 or later.
- eXtreme Burner – An easy to use GUI application for MS Windows.

For the list of the software above, we have no responsibility to teach users how to use, users must study themselves in order to use it.

7.1 AVRdude

AVRdude is a very popular and common for the command-line program for the programming of AVR chips. Here are some examples that you can try to load the program using AVR USBasp.

7.1.1 Programming for LED Blinking

Command:

```
avrdude -c usbasp -p m328p -e
```

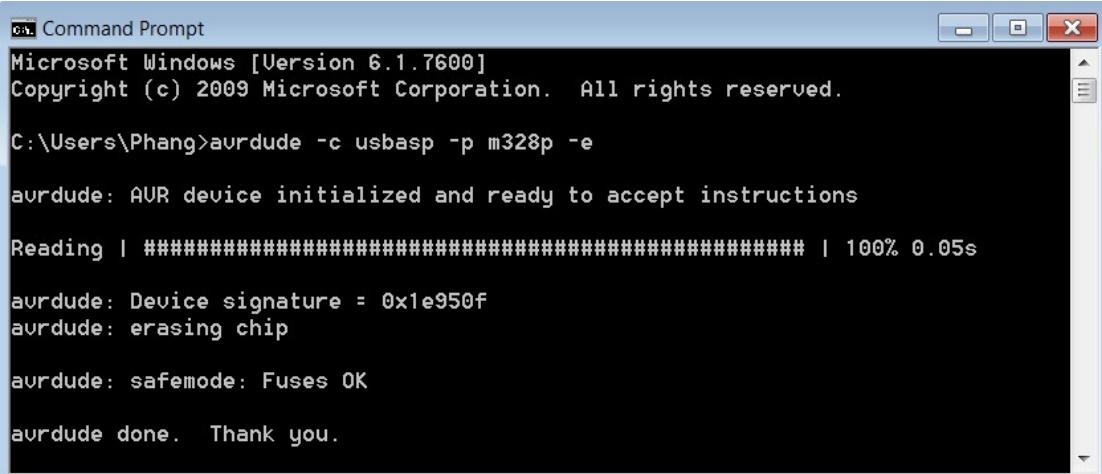
Explanation:

To delete the program of the microcontroller.

-c means which programmer that you are using, in this stage, we are using USBasp.

-p means which microcontroller that you are using, in this stage, we are using 328p.

-e means to delete the program inside the microcontroller.



```
Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Phang>avrdude -c usbasp -p m328p -e
avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.05s

avrdude: Device signature = 0x1e950f
avrdude: erasing chip

avrdude: safemode: Fuses OK

avrdude done. Thank you.
```

Command:

```
avrdude -c usbasp -p m328p -u -U flash:w:Blink.cpp.hex
```

Explanation:

When you write the Blink.cpp.hex for the LED Blinking, you need to make sure that the hex file of LED blinking need to be in the same folder with the drive you has call out just now. For example, the command below shows the drive of C:\Users\Phang, it means that I need to copy the LED Blinking hex file to the folder of C:\Users\Phang.

```
Command Prompt
C:\Users\Phang>avrdude -c usbasp -p m328p -u -U flash:w:Blink.cpp.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.04s

avrdude: Device signature = 0x1e950f
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 "Blink.cpp.hex"
avrdude: input file Blink.cpp.hex auto detected as Intel Hex
avrdude: writing flash (1010 bytes):

Writing | ##### | 100% 0.27s

avrdude: 1010 bytes of flash written
avrdude: verifying flash memory against Blink.cpp.hex:
avrdude: load data flash data from input file Blink.cpp.hex:
avrdude: input file Blink.cpp.hex auto detected as Intel Hex
avrdude: input file Blink.cpp.hex contains 1010 bytes
avrdude: reading on-chip flash data:

Reading | ##### | 100% 0.23s

avrdude: verifying ...
avrdude: 1010 bytes of flash verified

avrdude done.  Thank you.
```

7.1.2 Programming for Bootloader

Command:

```
avrdude -c usbasp -p m328p -e
```

Explanation:

To delete the program of the microcontroller.

-c means which programmer that you are using, in this stage, we are using USBasp.

-p means which microcontroller that you are using, in this stage, we are using 328p.

-e means to delete the program inside the microcontroller.

```
Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Phang>avrdude -c usbasp -p m328p -e

avrduude: AUR device initialized and ready to accept instructions

Reading | ##### | 100% 0.05s

avrduude: Device signature = 0x1e950f
avrduude: erasing chip

avrduude: safemode: Fuses OK

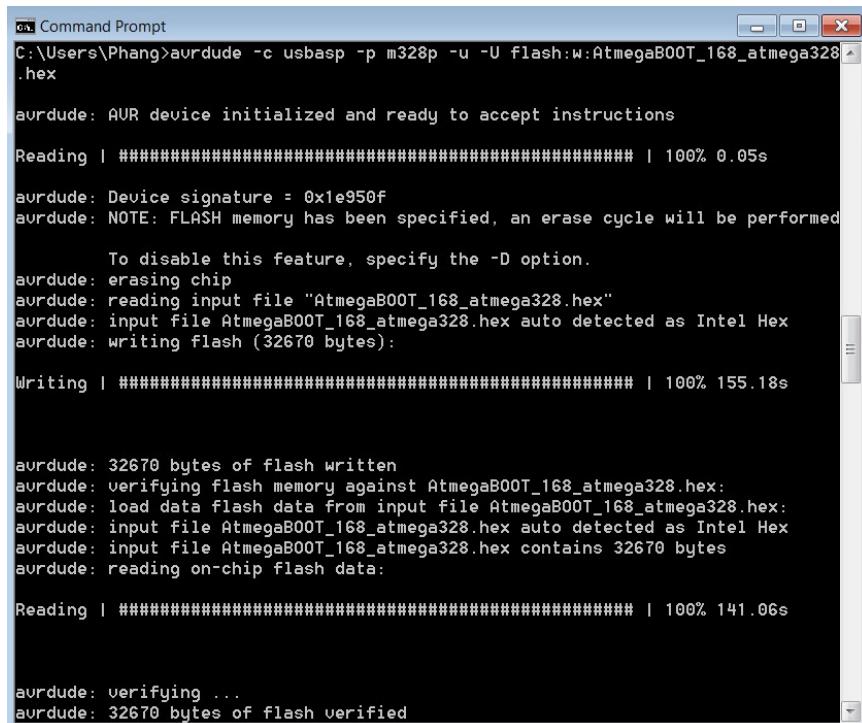
avrduude done. Thank you.
```

Command:

```
avrdude -c usbasp -p m328p -u -U flash:w:AtmegaBOOT_168_atmega328.hex
```

Explanation:

To write the bootloader program into the microcontroller.



```
0x Command Prompt
C:\Users\Phang>avrdude -c usbasp -p m328p -u -U flash:w:AtmegaBOOT_168_atmega328
.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.05s

avrdude: Device signature = 0x1e950f
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 "AtmegaBOOT_168_atmega328.hex"
avrdude: input file AtmegaBOOT_168_atmega328.hex auto detected as Intel Hex
avrdude: writing flash (32670 bytes):

Writing | ##### | 100% 155.18s

avrdude: 32670 bytes of flash written
avrdude: verifying flash memory against AtmegaBOOT_168_atmega328.hex:
avrdude: load data flash data from input file AtmegaBOOT_168_atmega328.hex:
avrdude: input file AtmegaBOOT_168_atmega328.hex auto detected as Intel Hex
avrdude: input file AtmegaBOOT_168_atmega328.hex contains 32670 bytes
avrdude: reading on-chip flash data:

Reading | ##### | 100% 141.06s

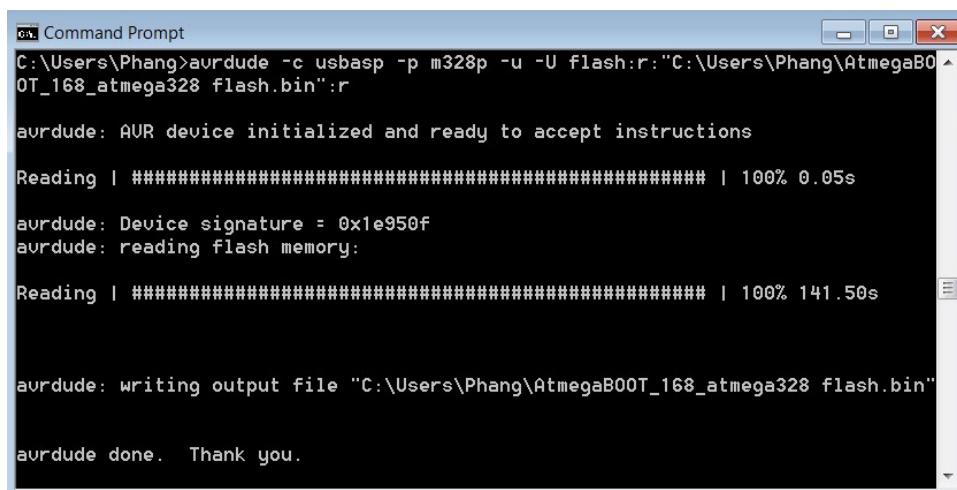
avrdude: verifying ...
avrdude: 32670 bytes of flash verified
```

Command:

avrdude -c usbasp -p m328p -u -U flash:r: "C:\Users\Phang\AtmegaBOOT_168_atmega328 flash.bin":r

Explanation:

Read the flash memory from the microcontroller connected to USBasp programmer and save it in raw binary format in the file name C:\Users\Phang\AtmegaBOOT_168_atmega328 flash.bin.



```
0x Command Prompt
C:\Users\Phang>avrdude -c usbasp -p m328p -u -U flash:r:"C:\Users\Phang\AtmegaBO
OT_168_atmega328 flash.bin":r

avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.05s

avrdude: Device signature = 0x1e950f
avrdude: reading flash memory:

Reading | ##### | 100% 141.50s

avrdude: writing output file "C:\Users\Phang\AtmegaBOOT_168_atmega328 flash.bin"

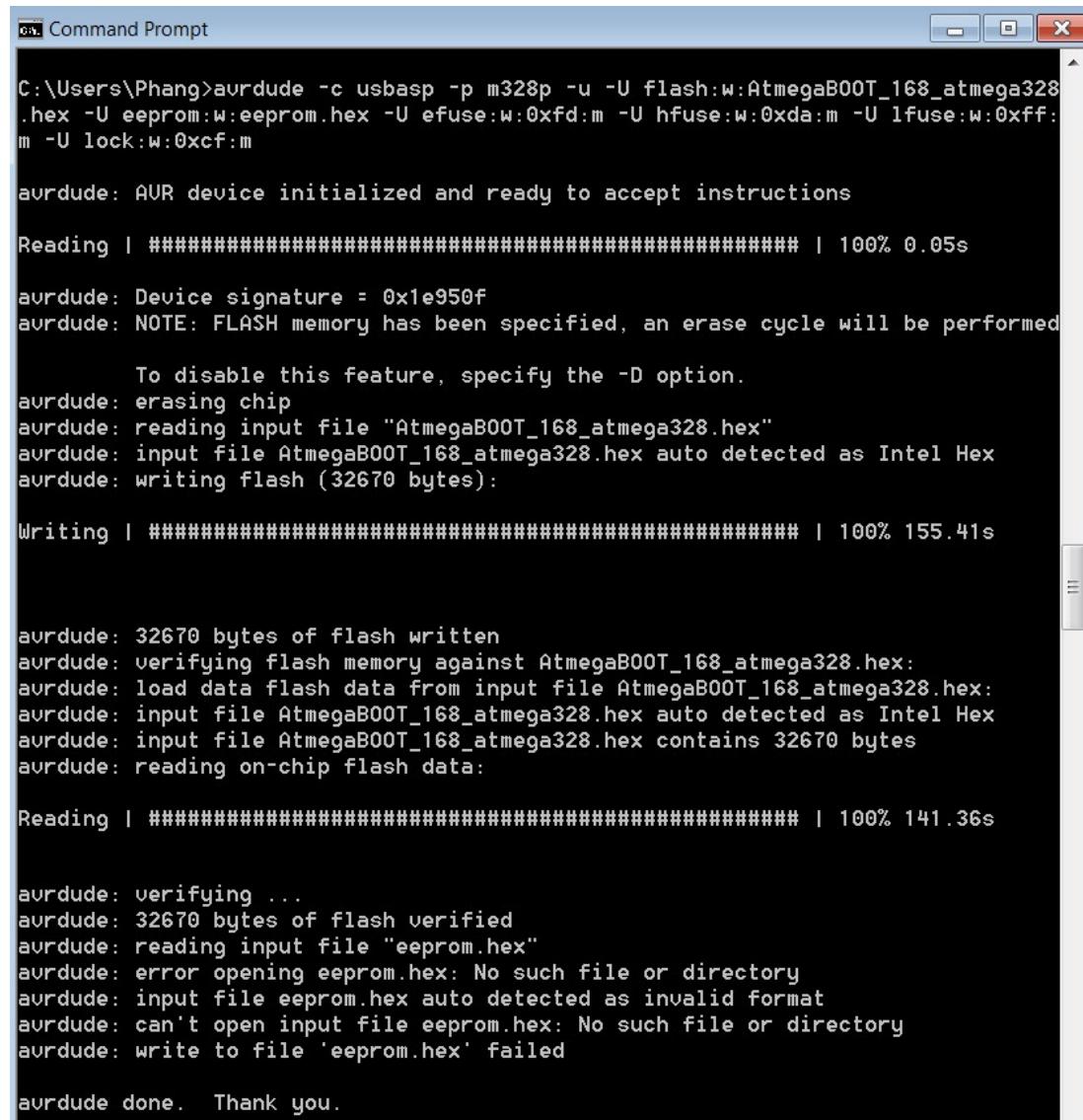
avrdude done. Thank you.
```

Command:

```
avrdude -c usbasp -p m328p -u -U flash:w:AtmegaBOOT_168_atmega328.hex -U
eeprom:w:eeprom.hex -U efuse:w:0xfd:m -U hfuse:w:0xda:m -U lfuse:w:0xff:m -U
lock:w:0xcf:m
```

Explanation:

Using the default programmer, write the file AtmegaBOOT_168_atmega328.hex to flash, .eeprom.hex to EEPROM, and set the Extended, High, Low and lock fuse bytes to 0xfd, 0xda, 0xff and 0xcf respectively.



```
C:\Users\Phang>avrdude -c usbasp -p m328p -u -U flash:w:AtmegaBOOT_168_atmega328
.hex -U eeprom:w:eeprom.hex -U efuse:w:0xfd:m -U hfuse:w:0xda:m -U lfuse:w:0xff:m
-U lock:w:0xcf:m

avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.05s

avrdude: Device signature = 0x1e950f
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 "AtmegaBOOT_168_atmega328.hex"
avrdude: input file AtmegaBOOT_168_atmega328.hex auto detected as Intel Hex
avrdude: writing flash (32670 bytes):

Writing | ##### | 100% 155.41s

avrdude: 32670 bytes of flash written
avrdude: verifying flash memory against AtmegaBOOT_168_atmega328.hex:
avrdude: load data flash data from input file AtmegaBOOT_168_atmega328.hex:
avrdude: input file AtmegaBOOT_168_atmega328.hex auto detected as Intel Hex
avrdude: input file AtmegaBOOT_168_atmega328.hex contains 32670 bytes
avrdude: reading on-chip flash data:

Reading | ##### | 100% 141.36s

avrdude: verifying ...
avrdude: 32670 bytes of flash verified
avrdude: reading input file "eeprom.hex"
avrdude: error opening eeprom.hex: No such file or directory
avrdude: input file eeprom.hex auto detected as invalid format
avrdude: can't open input file eeprom.hex: No such file or directory
avrdude: write to file 'eeprom.hex' failed

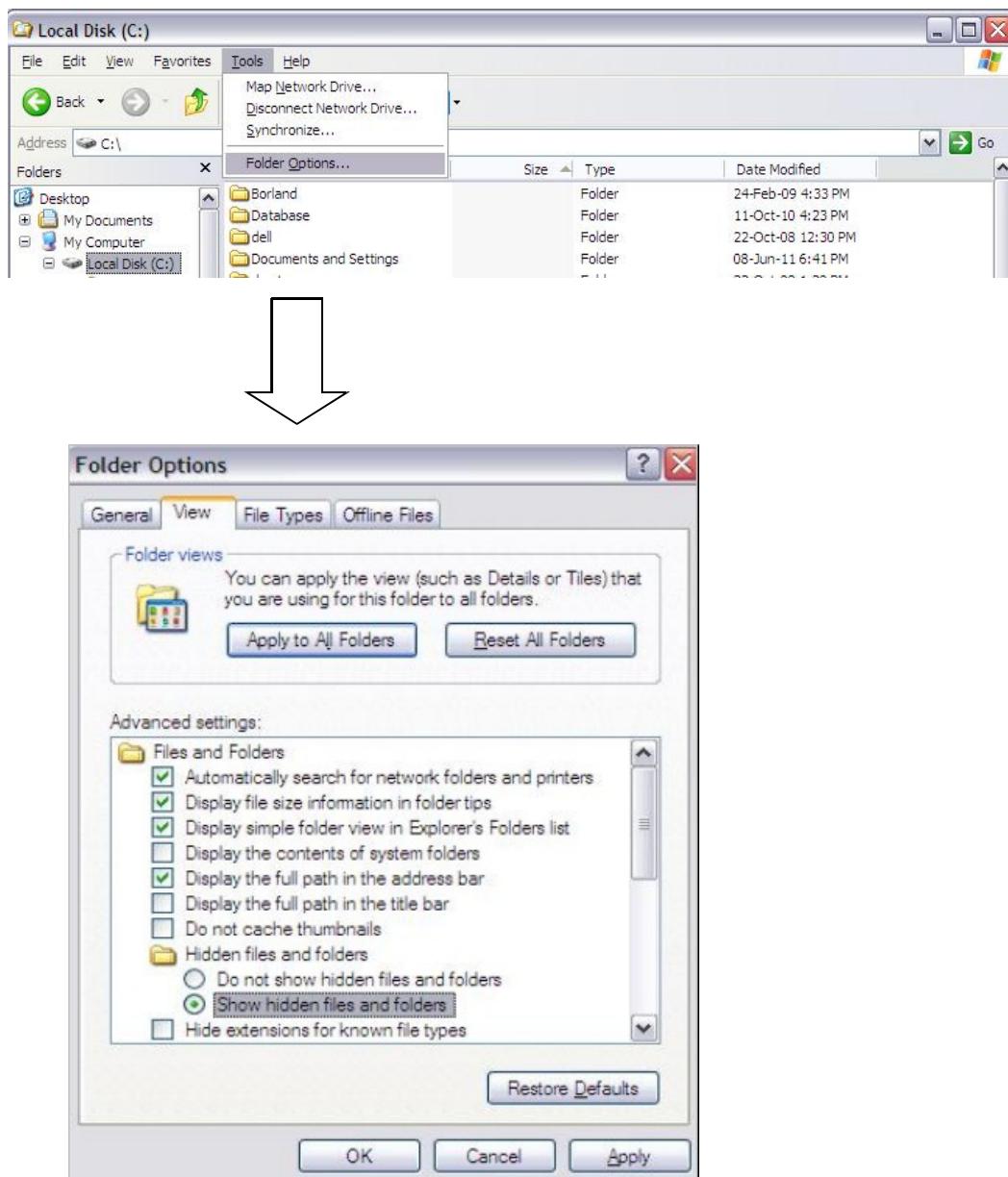
avrdude done.  Thank you.
```

7.2 Arduino Software

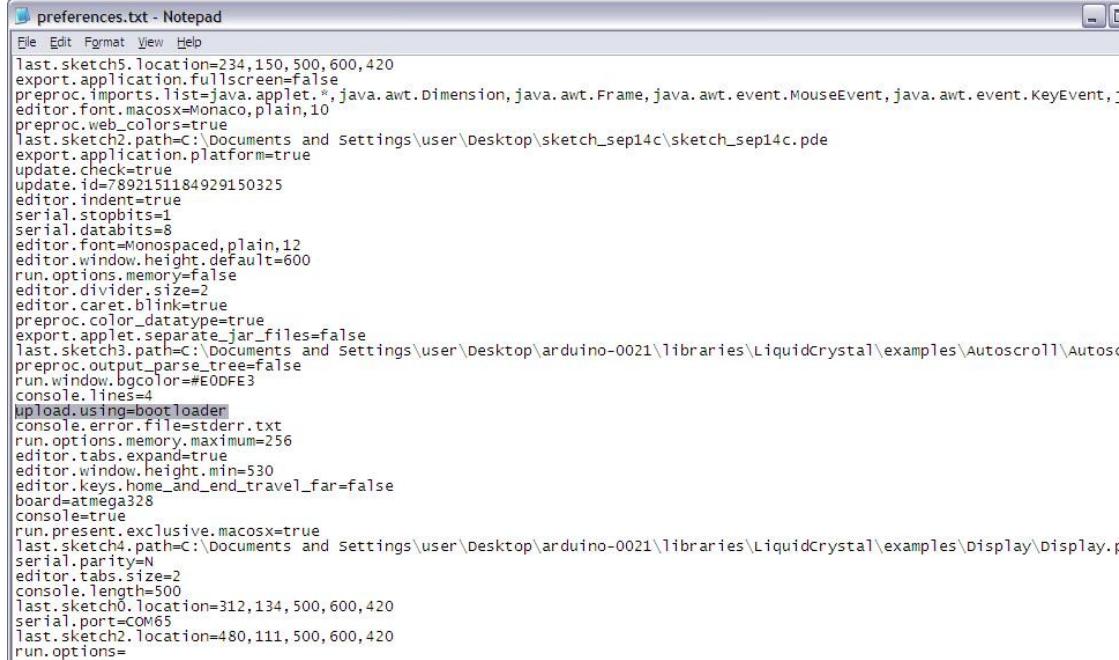
By modifying Arduino Software, we are able to program the Arduino board using AVR USBasp without pre-program bootloader in microcontroller. Besides that, we are also able to program the microcontroller without using Arduino board. For Arduino software, you can download it on this website: <http://arduino.cc/en/Main/Software>

Steps of modification

1. After you have installed the Arduino software to your PC, do not open the software first.
2. Goto C: ,click Tools > Folder Options > View , and tick the column which show “show hidden files and folder” then click apply and ok.



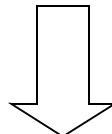
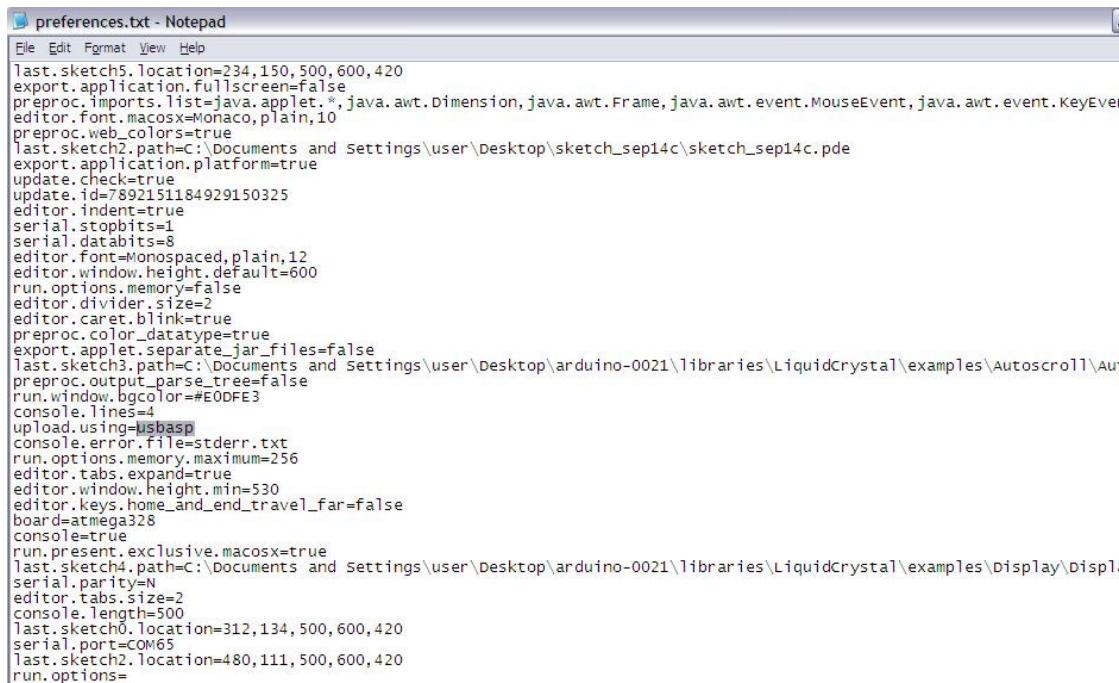
3. Go to C:\Documents and Settings\"user"\Application Data\Arduino
4. Open the file preferences.txt
5. Find the sentence upload.using=bootloader, and change it to upload.using=usbasp



```

preferences.txt - Notepad
File Edit Format View Help
last.sketch5.location=234,150,500,600,420
export.application.fullscreen=false
preproc.imports.list=java.applet.*,java.awt.Dimension,java.awt.Frame,java.awt.event.MouseEvent,java.awt.event.KeyEvent,*
editor.font.macosx=Monaco,plain,10
preproc.web_colors=true
last.sketch2.path=C:\Documents and Settings\user\Desktop\sketch_sep14c\sketch_sep14c.pde
export.application.platform=true
update.check=true
update.id=7892151184929150325
editor.indent=true
serial.stopbits=1
serial.databits=8
editor.font=Monospaced,plain,12
editor.window.height.default=600
run.options.memory=false
editor.divider.size=2
editor.caret.blink=true
preproc.color_datatype=true
export.appletp.separate_jar_files=false
last.sketch3.path=C:\Documents and Settings\user\Desktop\arduino-0021\libraries\LiquidCrystal\examples\Autoscroll\Autoscroll
preproc.output_parse_tree=false
run.window.bgcolor=#E0DFE3
console.lines=4
upload.using=bootloader
console.error.file=stderr.txt
run.options.memory.maximum=256
editor.tabs.expand=true
editor.window.height.min=530
editor.keys.home_and_end_travel_far=false
board=atmega328
console=true
run.present.exclusive.macosx=true
last.sketch4.path=C:\Documents and Settings\user\Desktop\arduino-0021\libraries\LiquidCrystal\examples\display\display
serial.parity=N
editor.tabs.size=2
console.length=500
last.sketch0.location=312,134,500,600,420
serial.port=COM65
last.sketch2.location=480,111,500,600,420
run.options=

```

```

preferences.txt - Notepad
File Edit Format View Help
last.sketch5.location=234,150,500,600,420
export.application.fullscreen=false
preproc.imports.list=java.applet.*,java.awt.Dimension,java.awt.Frame,java.awt.event.MouseEvent,java.awt.event.KeyEvent,*
editor.font.macosx=Monaco,plain,10
preproc.web_colors=true
last.sketch2.path=C:\Documents and Settings\user\Desktop\sketch_sep14c\sketch_sep14c.pde
export.application.platform=true
update.check=true
update.id=7892151184929150325
editor.indent=true
serial.stopbits=1
serial.databits=8
editor.font=Monospaced,plain,12
editor.window.height.default=600
run.options.memory=false
editor.divider.size=2
editor.caret.blink=true
preproc.color_datatype=true
export.appletp.separate_jar_files=false
last.sketch3.path=C:\Documents and Settings\user\Desktop\arduino-0021\libraries\LiquidCrystal\examples\Autoscroll\Autoscroll
preproc.output_parse_tree=false
run.window.bgcolor=#E0DFE3
console.lines=4
upload.using=usbasp
console.error.file=stderr.txt
run.options.memory.maximum=256
editor.tabs.expand=true
editor.window.height.min=530
editor.keys.home_and_end_travel_far=false
board=atmega328
console=true
run.present.exclusive.macosx=true
last.sketch4.path=C:\Documents and Settings\user\Desktop\arduino-0021\libraries\LiquidCrystal\examples\display\display
serial.parity=N
editor.tabs.size=2
console.length=500
last.sketch0.location=312,134,500,600,420
serial.port=COM65
last.sketch2.location=480,111,500,600,420
run.options=

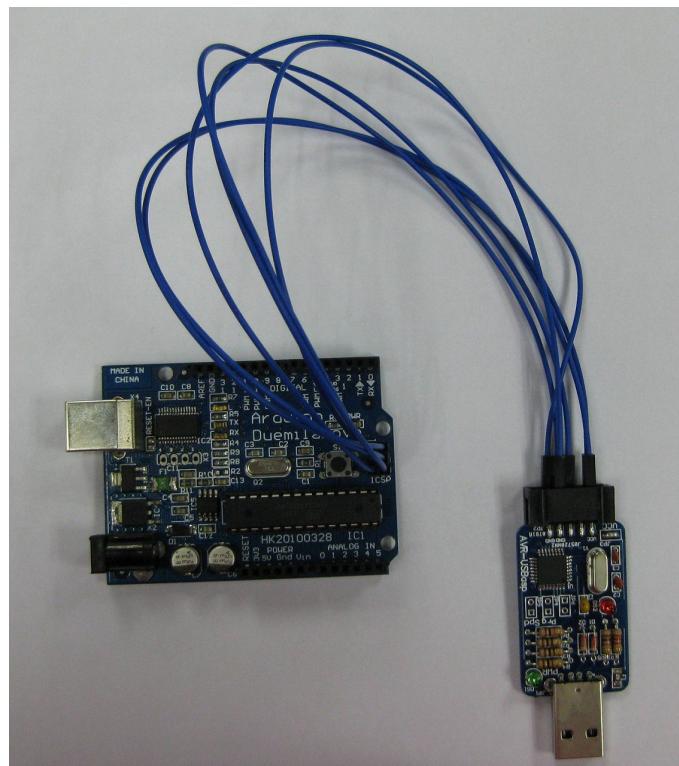
```

6. After replace it, save and close it.
7. Next, open the folder of the software Arduino which you have installed at the beginning.
8. Open Folder ...\\hardware\\arduino
9. Open the text file programmers.txt
10. Add another 2 lines in the below of the last sentence in the text file :

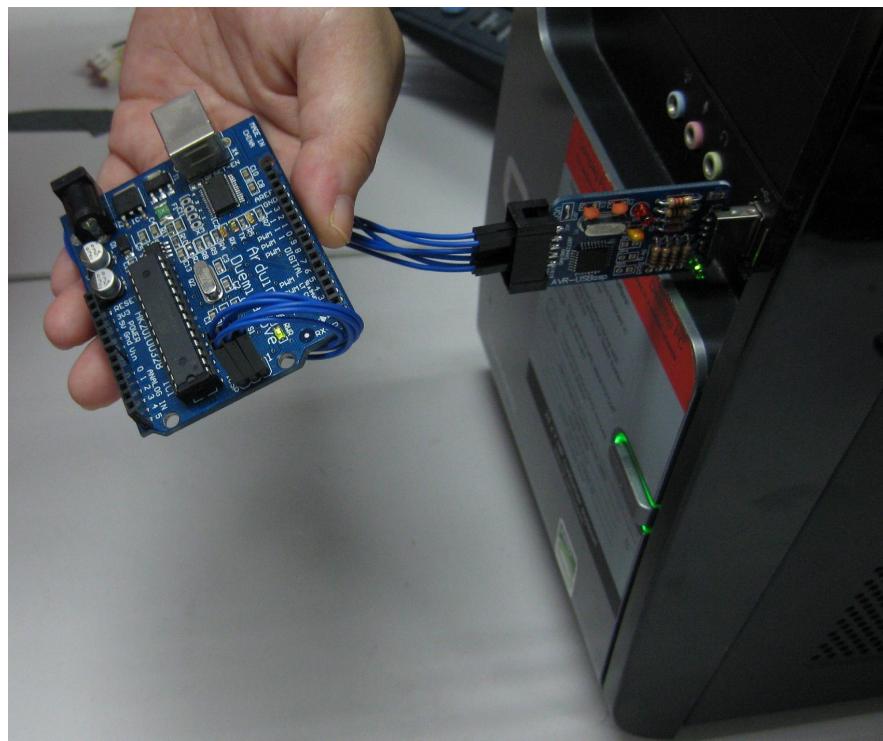
Usbasp.name=USBasp
Usbasp.protocol=usbasp
11. Save and close the text file.
12. The last step is to open the Arduino software Arduino.exe, and you can write your program in it and load it to your microcontroller using AVR USBasp.

Examples below show on how to load the program of LED Blinking to the Arduino Duemilanove board using AVR USBasp programmer.

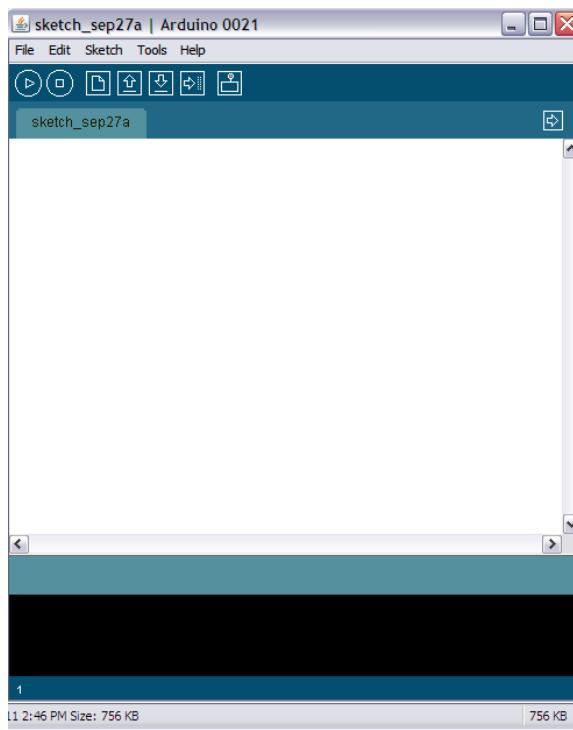
1. Do some wiring on the hardware parts. (Female to female jumper wire which show in the figure below is not come together with the product AVR USBasp, you should buy it if you need it on Cytron Technologies Sdn. Bhd.) The connection below, you can refer page 10, section 6: Example Schematic.



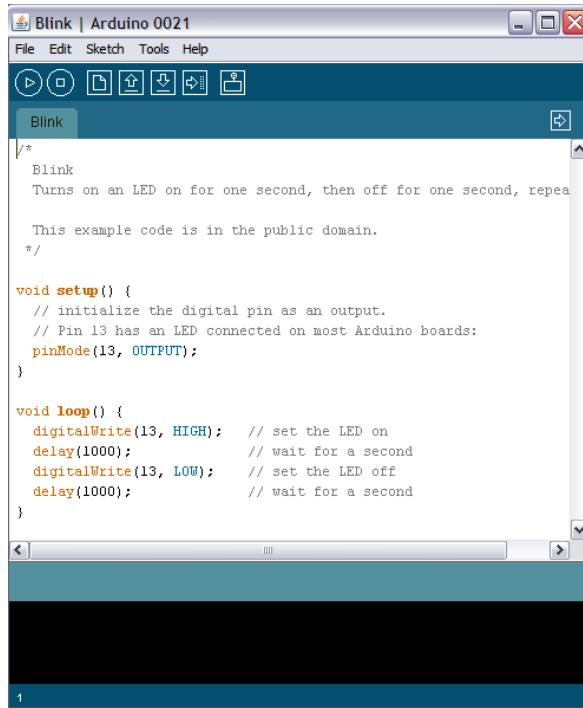
2. Connect the AVR USBasp to the USB port of the PC.



3. Open the Arduino software “Arduino.exe” to load the program in to the Arduino Duemilanove board. By using USBasp programmer, we do not need to load any bootloader program in it, and also, we do not need to choose the COM Port, and directly program into it.



4. Next, we need to call out the example of the source code by click on the Files > Examples > Basics > Blink. Then the source code of LED blinking will pop out in few seconds.



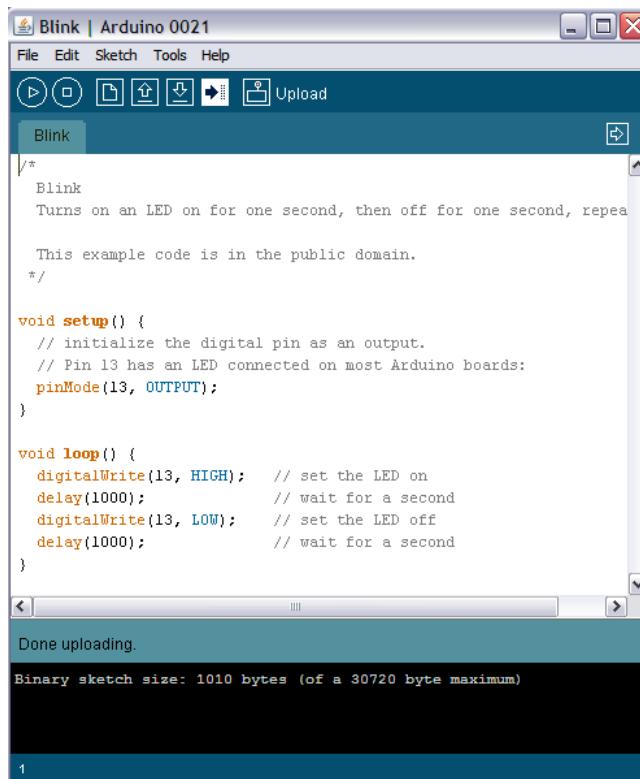
```

Blink | Arduino 0021
File Edit Sketch Tools Help
Upload
Blink
/*
Blink
  Turns on an LED on for one second, then off for one second, repeating.
  This example code is in the public domain.
*/
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);      // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(13, LOW);       // set the LED off
  delay(1000);                // wait for a second
}

```

5. Last step is to click on the Upload button which appears on the page of LED blink's source code to load the program to the microcontroller.



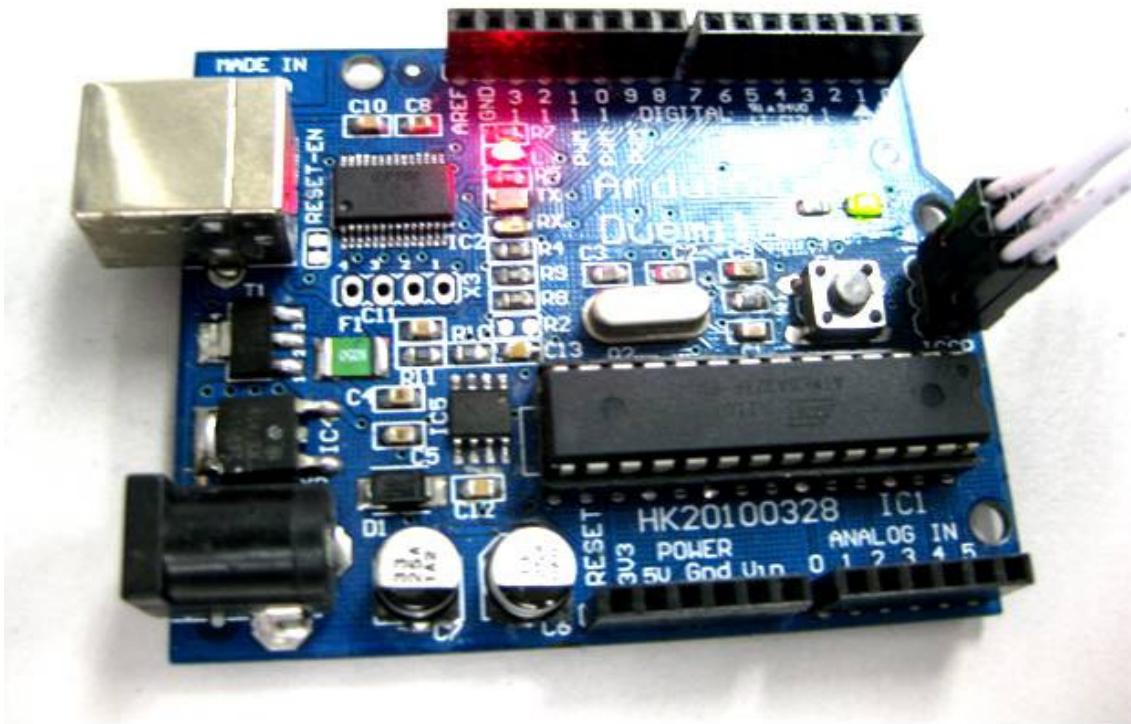
```

Blink | Arduino 0021
File Edit Sketch Tools Help
Upload
Blink
/*
Blink
  Turns on an LED on for one second, then off for one second, repeating.
  This example code is in the public domain.
*/
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);      // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(13, LOW);       // set the LED off
  delay(1000);                // wait for a second
}

```

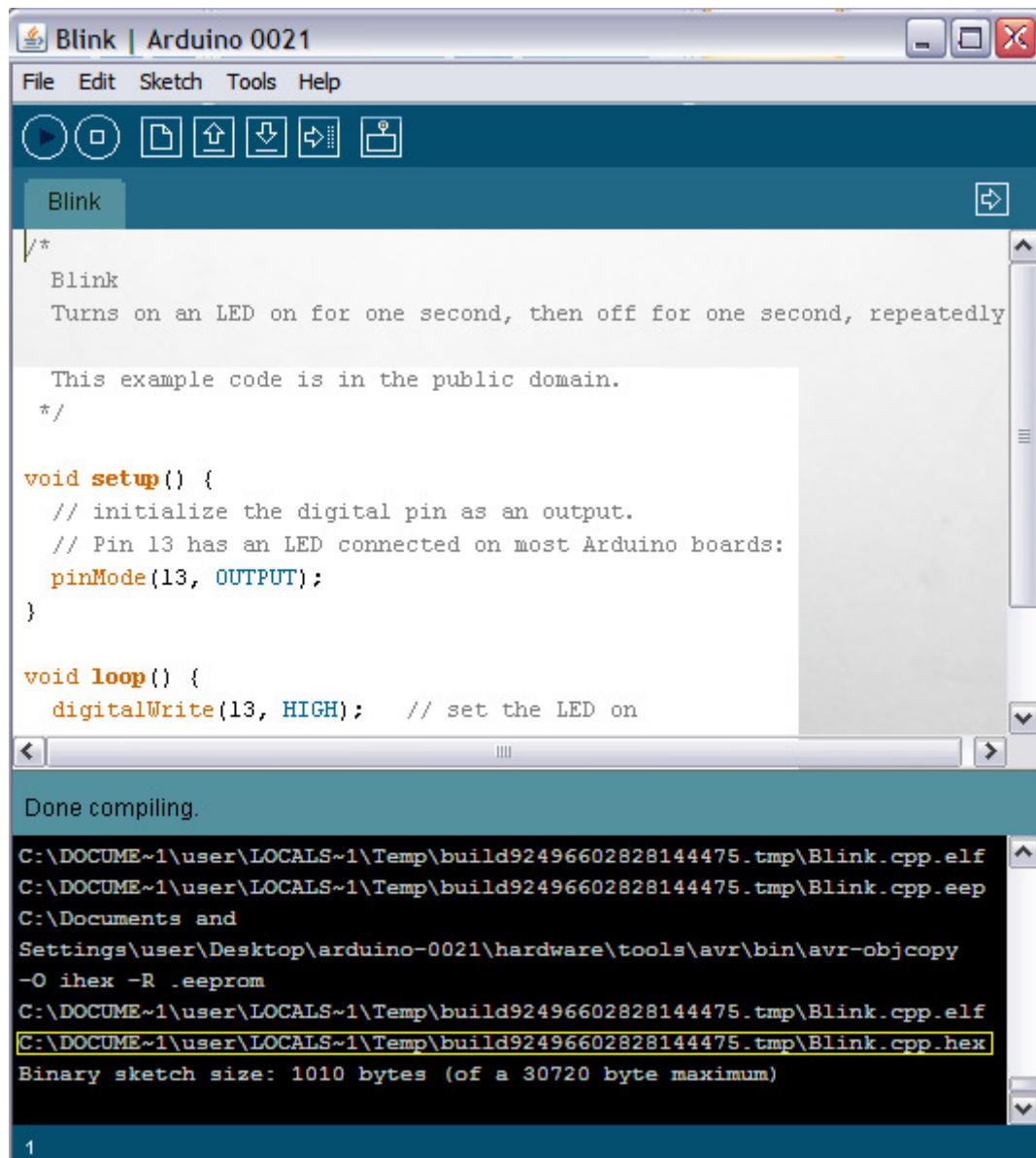
Done uploading.
Binary sketch size: 1010 bytes (of a 30720 byte maximum)



Output of LED Blinking

For your information, the hex file for the Arduino's example source code can be found by follow the steps below. By default, hex file will not be generated in Arduino Software. This allows you to write the code in Arduino Software, but program the AVR microcontroller by others software like AVRdude discussed at section 7.1 which require hex file.

1. Open the Arduino software.
2. Open the example source code (LED Blinking, LCD4bit_mode example and ect.). For this example, we are going to open the LED Blinking Source code.
3. Press “Shift” button on the keyboard and in the same time, click verified button on the LED Blinking source code's page. Then you will see the location of the hex file for the LED Blinking. It is same for every example source code.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Blink | Arduino 0021
- Menu Bar:** File, Edit, Sketch, Tools, Help
- Toolbar:** Includes icons for Open, Save, Print, and others.
- Sketch Editor:** Displays the **Blink** sketch code. The code is as follows:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly

  This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);      // set the LED on
```

- Compile Output:** Shows the compilation process and results. The output is:

```
C:\DOCUME~1\user\LOCALS~1\Temp\build92496602828144475.tmp\Blink.cpp.elf
C:\DOCUME~1\user\LOCALS~1\Temp\build92496602828144475.tmp\Blink.cpp.eep
C:\Documents and
Settings\user\Desktop\arduino-0021\hardware\tools\avr\bin\avr-objcopy
-O ihex -R .eeprom
C:\DOCUME~1\user\LOCALS~1\Temp\build92496602828144475.tmp\Blink.cpp.elf
C:\DOCUME~1\user\LOCALS~1\Temp\build92496602828144475.tmp\Blink.cpp.hex
Binary sketch size: 1010 bytes (of a 30720 byte maximum)
```

8. Warranty

- Product warranty is valid for 6 months.
- Warranty only applies to manufacturing defect.
- Damage caused by mis-use is not covered under warranty.
- Warranty does not cover freight cost for both ways.

Prepared by
Cytron Technologies Sdn. Bhd.

19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: +607-521 3178
Fax: +607-521 1861

URL: www.cytron.com.my
Email: support@cytron.com.my
sales@cytron.com.my