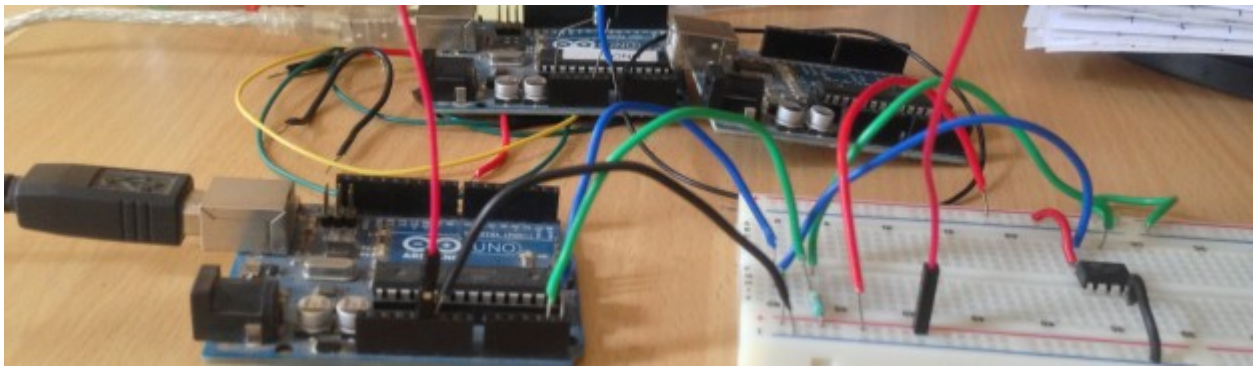
Hardware Fun

Adding FUN to Hardware Programming and Electronics

Compiling Arduino sketches using Makefile



Tweet

One of the main reasons why Arduino is very popular with beginners is that it is completely self-contained which makes it very easy to use. Once you have an Arduino board, all you have to do is to download the software and within minutes you can make your first led blink. The software has everything you need right from the IDE, to the compiler and even serial monitor to communicate with Arduino, all within one single package.

While the Arduino IDE is very good for beginners, you might start to feel its limitations, once you start working with it regularly. I use vim for everything and always wished writing Arduino sketches in vim.

I configured Arduino IDE, to use an external editor and started using vim to write code and then Arduino IDE to compile and upload. While this worked for sometime, I started to feel the need for a proper makefile so that I can do everything from vim itself.

Makefile for Arduino

My search landed me to a makefile created by Martin Oldfield. I started using it regularly and also contributed back my patches. Over a period of time, Martin lost interest in the project and now I took ownership of the project and maintain it.

The makefile is quite mature now and in most cases, you can use it to replace the Arduino IDE. There are still some corner cases, but I guess you may not hit them in your day to day use. Also the other advantage of using the makefile is that, you can even program directly using AVR C or assembly, which is not quite easy to do with the Arduino IDE.

Installation

The makefile requires the Arduino software to be installed. So if you have not installed it before, then you have to install it first.

There are three ways by which you can get the makefile

- Install through package
- Do a checkout from github
- Download zip file from github

Install through package

Packages are available for Debian, Ubuntu and FreeBSD. The package name is `arduino-mk`. If you prefer to install a package, rather than checking out code from github, then you can install the package, if you are using Debian, Ubuntu or FreeBSD.

I also have plans to have a package for homebrew. I will post an update once the package is available for homebrew.

Do a checkout from github

The makefile is hosted in github. You can directly checkout from github. The advantage of this method is that it is every easy to get updates, since the project is currently under heavy development.

Download zip file from github

If you are not comfortable with git or don't want to do a checkout, then you can also download the zip file from github.

Install dependencies

The Makefile delegates resetting the board to a short Perl program. You'll need to install `Device::SerialPort` and `YAML` library.

On Debian or Ubuntu:

```
apt-get install libdevice-serialport-perl
apt-get install libyaml-perl
```

On Fedora:

```
yum install perl-Device-SerialPort
yum install perl-YAML
```

On Mac using MacPorts:

```
sudo port install p5-device-serialport
sudo port install p5-YAML
```

and use `/opt/local/bin/perl5` instead of `/usr/bin/perl`

On other systems:

```
cpanm Device::SerialPort
cpanm YAML
```

Setup

Instead of copying the makefile to every sketch folder, you can just place the downloaded makefile separately in a common location and then create a small child makefile for every sketch.

Global variables

Once you have copied the makefile to a common location, or installed it through a package, you need to declare the following global variables. You can either declare them in your child makefile or set them as environmental variables.

- `ARDUINO_DIR` – Directory where Arduino is installed
- `ARDMK_DIR` – Directory where you have copied the makefile
- `AVR_TOOLS_DIR` – Directory where avr tools are installed

I have the following setup in my `~/.bashrc` file

```
export ARDUINO_DIR=/home/sudar/apps/arduino-1.0.5
export ARDMK_DIR=/home/sudar/Dropbox/code/Arduino-Makefile
export AVR_TOOLS_DIR=/usr
```

Per sketch variables

After the global settings, you will need to specify the following variables for each sketch

- `BOARD_TAG` – The Arduino board that you are using. By default `Uno` is used
- `ARDUINO_PORT` – The serial port where Arduino is connected
- `ARDUINO_LIBS` – Space separated set of libraries that are used by your sketch

Usage

Compiling programs

To compile your programs you just have to invoke the command `make`.

The output is pretty verbose and will list down the configurations that the makefile is using and from where it got the values.

All the build files will be created under a subdirectory in your sketch folder.

If there were any errors then they will be displayed with line numbers, which you can correct.

Uploading programs

To upload the compiled program to your Arduino, just plug your Arduino and run the command `make upload`.

The program will be recompiled if needed and will be uploaded to your Arduino board.

Opening Serial monitor

The makefile can also be used to check the serial output from your Arduino. To do that just run the command `make monitor`.

This command will open a serial connection to your Arduino using `screen`.

The makefile tries to auto detect the baud rate. If it is not able to detect it properly, then you can manually set it using the variable `MONITOR_BAUDRATE`.

Advanced Usage

In addition to the above typical workflows, the makefile can also be used to do the following advanced stuff. I will write detailed guide for each of these use cases when I get some free time. Meanwhile you can also checkout some of the sample makefiles in the examples folder at github.

- Compiling plain AVR C programs

- Program using Arduino as ISP
- Generate assembly and symbol files
- Program using alternate Arduino core (like ATtiny or Arduino alternate cores)

Related projects

If you are using Vim as your editor for compiling Arduino programs then you might be interested in some of my following projects as well.

- Vim syntax files for Arduino
- Vim snippets for Arduino
- Arduino Extra Core – Additional cores for non Arduino AVR microcontrollers like ATmega 16

Do check them out as well 🙂

This entry was posted in Tutorials and tagged Arduino, makefile on June 23, 2013 [http://hardwarefun.com/tutorials/compiling-arduino-sketches-using-makefile] .

---

84 thoughts on “Compiling Arduino sketches using Makefile”

K
July 1, 2013 at 1:27 AM

Dude, really glad there’s someone to continue taking care of arduino-mk.

Just wanted to say thanks – I’m sitting here with a Leonardo Nano and I thought I’d have to start tweaking to get it to work, but now I can just use your repo. Cheers!

Sudar Post author
July 1, 2013 at 2:19 PM

You are welcome 🙂

BTW do let me know if Leonardo Nano works with this makefile.

Pingback: Use Arduino to program non-Arduino AVR microcontrollerHardware Fun | Hardware Fun

Pingback: Use Arduino code in non-Arduino AVR microcontrollersHardware Fun | Hardware Fun

michalkocer
July 30, 2013 at 12:23 PM

Awesome work, thank you. Easy and straight forward 🙂 Keep it rolling 😉

Sudar Post author
July 30, 2013 at 2:48 PM

Nice to know that you like this 🙂

Jay
August 1, 2013 at 12:29 PM

Sudar, firstly – great work, so please do keep it up.

Actually, I started looking for latest version of Arduino-mk after I accidentally spotted it in Debian package list. However the first link that came up with this other version:
http://ed.am/dev/make/arduino-mk

Wondering if it would be possible to do a compare and contrast of the two ? I am sure there are more alternatives, but the other was, I think also has a link-back from arduino.cc !

Also, have you considered something like picocom (little brother of minicom), for the serial communication. The other (ed.am) Arduino-mk was reportedly used by someone in conjunction with picocom. The benefit of picocom, as I understand is the ability to have bi-directional communication over serial port. Very useful tool for the poor-man's debugging technique using serial.writes and serial.read (especially to emulate break-points, in a way). From what I understood, the Perl device-serial is just one way lane, or perhaps, I am mistaken !

Sudar Post author
August 3, 2013 at 2:21 PM

Nice to know that you like this.

This is the first time I am hearing about the *other* arduino-mk. The link that you shared is currently down. I will have a detailed look at it latter, once the link is up.

Meanwhile, the Perl device-serial is used only for resetting the board not for communicating with the board over serial. There is build in support for doing serial communication with Arduino. Checkout the *monitor* target in the makefile.

Also the serial command can be easily overridden. So, in theory my makefile supports picocom (and other programs) without any change to it.

Scott Davis
August 20, 2013 at 7:57 AM

Thanks, man! Easy as pie.

Sergey
September 18, 2013 at 11:12 AM

Hi Sudar,

Thank you for your job. That is really gorgeous! One proposal:
If you run make not from vim directly, but from command line, a nice to have feature is colorgcc (https://github.com/colorgcc/colorgcc). It adds a color output to gcc messages. It is extremely easy to set up. All what have to be done is:

1.
in ~/.colorgccrc add

avr-g++: /Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-g++
avr-gcc: /Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-gcc
color-avr-g++: /Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-g++
color-avr-gcc: /Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-gcc

2.
put soft links to colorgcc.pl for avr-g++ and avr-gcc from somewhere in your PATH, i.e.

```
PATH=${HOME}/bin:${PATH}

ln -s path_to_colorgcc.pl ~/bin/avr-gcc
ln -s path_to_colorgcc.pl ~/bin/avr-g++
```

3.
alter your project Makefile to look like:

```
BOARD_TAG = uno
MONITOR_PORT = /dev/cu.usb*
ARDUINO_LIBS =

include $(ARDMK_DIR)/Arduino.mk

#
# Override Arduino.mk defaults to point to my soft links to
# colorgcc. Colorgcc reads the config from ~/.colorgccrc
#
CC = avr-gcc
CXX = avr-g++
```

4.
and you will get a colored output in console for avr-gcc.

Hope, that makes sense

Sudar Post author
September 18, 2013 at 12:30 PM

@Sergey,

Thanks for mentioning colorgcc. I didn't know about it before and it seems to be extremely useful. I will add notes about integrating the makefile with colorgcc

One small issue though is that, I see there are a couple of open issues and pull requests for it in github, but no one seems to be maintaining it anymore.

Sergey
September 18, 2013 at 12:38 PM

I didn't pay an attention to that. I've been using colorgcc for 3 years, may be, and I have never had any issue big enough to turn me off using it. It is extremely nice to have when you're in the console. There is another one, called colorsvn if you like SVN. Git, which is better, of course, has embedded color support.

Sudar Post author
September 18, 2013 at 12:42 PM

Good to know that it is stable. I am have just added it to the todo list https://github.com/sudar/Arduino-Makefile/issues/119

BTW, can you share the link to colorsvn. Thanks.

Sergey
September 18, 2013 at 12:48 PM

Thanks for adding a TODO.

colorsvn project homepage is http://colorsvn.tigris.org/
and it took me a while to find a download section 🙂
http://colorsvn.tigris.org/servlets/ProjectDocumentList

Sudar Post author

September 18, 2013 at 2:34 PM

Thanks for the link. Will check it out.

Jay
September 18, 2013 at 6:49 PM

Then there is ‘colormake’ as well 🙂
https://github.com/pagekite/Colormake

Sudar Post author
September 19, 2013 at 8:18 PM

Thanks Jay. Looks interesting 🙂

Pingback: Arduino Makefile hits v1.0.0Hardware Fun | Hardware Fun

Pingback: Use Arduino as an ISP programmer to program non-Arduino AVR microcontrollersHardware Fun | Hardware Fun

Pingback: Arduino Makefile v1.1.0 releasedHardware Fun | Hardware Fun

Dougie
January 6, 2014 at 1:34 PM

How does this differ from inotool.org ?

Sudar Post author
January 6, 2014 at 7:11 PM

Both are very similar, but there are some fine differences.

- Ino abstracts out most of the makefile, therefore it is quite easy, but if you need to tweak it, then it is not possible.
- This makefile can be used to compile just plain AVR C, but that is not possible by Ino.
- Ino doesn't (as far as my understanding) support custom core, but this makefile supports it.

In short, both do similar things, but this makefile gives you more control, while ino is very easy to get started.

Gareth Moffatt
August 21, 2017 at 4:34 AM

One difference I discovered was that after several hours of hacking, I still couldn't get ino to work, but arduino-mk worked almost out of the package!

I don't know if it is still being supported, but I'm impressed so far.

Gareth

lotsarats
March 15, 2014 at 7:28 PM

installed this with apt-get install arduino-mk with debain wheezy 7 and i received arduino.mk version 0.8 by martin.

your "installation through package" appears to be incorrect as your not updating the official repositories

---

Peter
April 2, 2014 at 11:16 AM

Hello Sudar,

I started to play with the Makefile and I like it! Especially the plain BlinkInAVRC example. I would like to upload the BlinkInAVRC on my Arduino micro, but I failed. I edited the BOARD_TAG and the MCU
BOARD_TAG = atmega32u4
MCU = atmega32u4
F_CPU = 16000000L

but I do have to do something with the programmer I guess. Can you give some advise?

Cheers,
Peter

Sudar Post author
April 2, 2014 at 11:37 AM

Can you post your full makefile and the exact error message that you are getting?

Peter
April 2, 2014 at 2:22 PM

Hi Sudar,

here it comes. Just as a comment, it works when I use leonardo as BOARD_TAG but then all the Arduino software is compiled and linked as well, I do only want to compile the pure blink.c

Thanks for the help,
Peter

Makefile:

This sample Makefile, explains how you can compile plain AVR C file.
#
Arduino Make file. Refer to https://github.com/sudar/Arduino-Makefile

NO_CORE = Yes

```
#BOARD_TAG = atmega16
#MCU = atmega16
BOARD_TAG = atmega32u4

MCU = atmega32u4
F_CPU = 16000000L

ISP_PROG = stk500v1
AVRDUDE_ISP_BAUDRATE = 19200

include ../../Arduino.mk

# !!! Important. You have to use make isploadto upload when using ISP programmer
————————————————————————————-
```

The output of the commandline:

```
– [COMPUTED] BOOTLOADER_PARENT = /usr/share/arduino/hardware/arduino/bootloaders (from
ARDUINO_DIR)
——————————-
mkdir -p build-atmega32u4

Maximum flash memory of atmega32u4 is not specified. Make sure the size of build-
atmega32u4/BlinkInAVRC.hex is less than atmega32u4\'s flash memory

make reset
make[1]: Entering directory `/ubuntu/home/holterma/data/src/Arduino-Makefile.git/examples/BlinkInAVRC'
/ubuntu/home/holterma/data/src/Arduino-Makefile.git/bin/ard-reset-arduino /dev/ttyACM3
TIOCMBIC(21527) ioctl failed: Datenübergabe unterbrochen (broken pipe) at
/ubuntu/home/holterma/data/src/Arduino-Makefile.git/bin/ard-reset-arduino line 66
dtr_active(0) ioctl: Datenübergabe unterbrochen (broken pipe)
could not restore from dtr on
make[1]: Leaving directory `/ubuntu/home/holterma/data/src/Arduino-Makefile.git/examples/BlinkInAVRC'
make do_upload
make[1]: Entering directory `/ubuntu/home/holterma/data/src/Arduino-Makefile.git/examples/BlinkInAVRC'
/usr/bin/avrdude -q -V -D -p atmega32u4 -c -b -P /dev/ttyACM3 \
-U flash:w:build-atmega32u4/BlinkInAVRC.hex:i

avrdude: Can't find programmer id "-b"
```

Sudar Post author

April 2, 2014 at 2:29 PM

For Arduino Micro, you should use `BOARD_TAG=micro`

Peter
April 2, 2014 at 3:00 PM

Its working with this Makefile,

best,
Peter

```
# This sample Makefile, explains how you can compile plain AVR C file.
#
# Arduino Make file. Refer to https://github.com/sudar/Arduino-Makefile

NO_CORE = Yes

#BOARD_TAG = atmega16
#MCU = atmega16
BOARD_TAG = micro
MCU = atmega32u4
F_CPU = 16000000L

#ISP_PROG = stk500v1
#AVRDUDE_ISP_BAUDRATE = 19200

AVRDUDE_ARD_PROGRAMMER = avr109
AVRDUDE_ARD_BAUDRATE = 57600

include ../../Arduino.mk

# !!! Important. You have to use make ispload to upload when using ISP programmer
```

Sudar Post author

April 2, 2014 at 3:38 PM

Glad to know that you got it working.

Pingback: Arduino Makefile updatesHardware Fun | Hardware Fun

Xavier
May 10, 2014 at 1:02 AM

Thank you for your work on this project: it worked during my first try on my side!

And to make the documentation even better: you say in this page that the makefile delegates board resetting to a Perl script while it seems that it's a Python script that does this. Maybe this page is not up to date? Moreover, the Arch Linux package to install for pySerial is "python-pyserial" so the command to install it is:
pacman -S python-pyserial

Thanks again!

Sudar Post author
May 10, 2014 at 10:27 AM

Nice to know that you like the project.

Yes, this page is slightly out of date. I was updating the github readme file, but didn't updated this post. Will update this post as well.

Pingback: Arduino Makefile v1.3.4 releasedHardware Fun | Hardware Fun

Gerhard
November 9, 2014 at 3:30 PM

Thank you for this great addition. It already helped me a lot.

One thing I couldn't figure out how to compile release code. I have a couple of '#ifdef DEBUG' scattered in my code, but I want to remove these sections from the compiled hex for release by just adding something to the command line. Is this already possible?

Gerhard
November 9, 2014 at 8:16 PM

I found out myself. Always use a clean working directory... 😉

$ make clean && CXXFLAGS_STD=-DDEBUG make upload
$ make clean && make upload

Sudar Post author
November 9, 2014 at 8:33 PM

Glad that you found the solution 🙂

Pingback: Fix Arduino Uno Error Compiling Windows XP, Vista, 7, 8 [Solved]

Pingback: Fix Arduino Error Compiling Windows XP, Vista, 7, 8 [Solved]

Pingback: Standalone Arduino Makefile for compiling sketches with AVR-GCC - TEKK

Alex
February 2, 2015 at 4:46 AM

Is there a way to specify avr-gcc options in the makefile? I'm doing a paper on how optimization effects different things like code size and speed.

Sudar Post author
February 2, 2015 at 7:42 PM

Yes it is possible. Check out https://github.com/sudar/Arduino-Makefile/blob/master/arduino-mk-vars.md#cflags

Alex
February 8, 2015 at 5:52 AM

Okay I'm having problems…
————————-
Arduino.mk Configuration:
– [USER] ARDUINO_DIR = /usr/share/arduino
– [AUTODETECTED] ARDUINO_VERSION = 105
– [USER] AVR_TOOLS_DIR = /usr
– [COMPUTED] ARDUINO_LIB_PATH = /usr/share/arduino/libraries (from ARDUINO_DIR)
– [COMPUTED] ARDUINO_VAR_PATH = /usr/share/arduino/hardware/arduino/variants (from ARDUINO_DIR)
– [USER] ARDMK_DIR = /usr/share/arduino/
– [COMPUTED] ARDMK_PATH = /usr/share/arduino//bin (relative to ARDMK_DIR)
– [AUTODETECTED] ARDUINO_SKETCHBOOK = /home/abferm/sketchbook (in arduino preferences file)
– [USER] USER_LIB_PATH = /home/abferm/Documents/code-optimization/Arduino/BlinkSizeL1/lib
– [USER] BOARD_TAG = uno
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
– [USER] MONITOR_BAUDRATE = 115200
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
– [AUTODETECTED] Size utility: AVR-aware for enhanced output
————————-
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
make: /usr/share/arduino//bin/ard-parse-boards: Command not found
/usr/bin/avr-g++ -x c++ -include Arduino.h -MMD -c -mmcu=atmega328p -DF_CPU=16000000L -DARDUINO=105 -I. -I/usr/share/arduino/hardware/arduino/cores/arduino -I/usr/share/arduino/hardware/arduino/variants/ -g -Wall -ffunction-sections -fdata-sections -pedantic -Wall -Wextra -fno-exceptions BlinkSizeL1.ino -o /home/abferm/Documents/code-optimization/Arduino/BlinkSizeL1/bin/uno/BlinkSizeL1/BlinkSizeL1.o
In file included from :0:0:
/usr/share/arduino/hardware/arduino/cores/arduino/Arduino.h:198:14: warning: anonymous variadic macros were introduced in C99 [-Wvariadic-macros]
/usr/share/arduino/hardware/arduino/cores/arduino/Arduino.h:213:26: fatal error: pins_arduino.h: No such file or directory
compilation terminated.

Alex
February 9, 2015 at 9:21 AM

Sorry for the trouble. I solved it myself. FYI if you install using apt the ARDMK_DIR should be /usr/bin…

Sudar Post author
February 11, 2015 at 11:53 AM

Glad to know that you got it resolved yourself.

Havard
February 8, 2015 at 5:06 AM

In Ubuntu 14.10, the recipe for getting started is the following:

1. $ sudo apt-get install arduino-mk

2. Get your development user into the ‘dialout’ group:
$ sudo gedit /etc/group
and append your username to the comma separated list of usernames on the dialout line:
dialout:x:20:

3. Log out, log in, to get the group to take effect

4. make an empty directory you want to program in.

5. create this makefile (for the uno):
~/empty_dir_for_an_arduino_project $ cat Makefile
ARDMK_DIR = /usr/share/arduino
BOARD_TAG = uno
include $(ARDMK_DIR)/Arduino.mk

6. make this code file:
$ cat main.cc
#include

int main(void) {
init();
const int led_pin = 12;
pinMode(led_pin, OUTPUT);
for (bool state = true; ; state = !state) {
digitalWrite(led_pin, state ? HIGH : LOW);
delay(500);
}
}

7: make && make upload

Havard
February 8, 2015 at 5:16 AM

The included file in the code is supposed to be

#include Arduino.h

Where Arduino.h inside less than and greater than signs like this:
https://github.com/ondras/arduino/blob/master/serial-remote/Remote.cc#L3

Mike Schwager
February 21, 2015 at 12:47 AM

This is really nice. I was going to cobble together my own Makefile when I bumped into this.

I love the instructions for installing into Fedora 20. This is how software should be: Complete, few bugs or issues, works out of the box, helps me get my work done. I can't believe how small my local Makefile is. Great work.

Thanks!

Sudar Post author
February 21, 2015 at 9:17 AM

Hello Mike,

I am very glad to know that you like the makefile 🙂

Tom Freudenberg
March 13, 2015 at 7:57 PM

Hi,

I am using an UDOO board and was looking for a suitable solution for having an environment to compile and flash the Arduino Due directly on my UDOO boards. I created a script which does all the work and prepared a full debian package environment. The package is suitable only for UDOO boards but the script “tools/udoo-arduino-cli” may help others too.

https://github.com/TomFreudenberg/udoo-arduino-cli

Cheers,
Tom

Pingback: Makefiles (and arduino-core?) | Project Echo

Brian
March 23, 2015 at 10:14 AM

Hi,

Thanks for the work on Arduino-MK.

I’m having trouble with the ATtinyBlink example.
I think i have everything set up but I get this error:

```
mkdir -p build-attiny85-8
/usr/local/CrossPack-AVR-20131216/bin/avr-g++ -x c++ -include Arduino.h -MMD -c -mmcu= -DF_CPU= -
DARDUINO=105 -D__PROG_TYPES_COMPAT__ -
I/Applications/Arduino.app/Contents/Resources/Java/hardware/arduino//cores/arduino -
I/Users/brian/sketchbook/hardware/attiny/variants/ -Wall -ffunction-sections -fdata-sections -Os -fno-
exceptions ATtinyBlink.ino -o build-attiny85-8/ATtinyBlink.o
avr-g++: error: missing argument to '-mmcu='
make: *** [build-attiny85-8/ATtinyBlink.o] Error 1
```

I'm on Mac OS X 10.10. I have the attiny folder in the sketchbook/hardware dir.

Any ideas?

Thanks

Sudar Post author
March 23, 2015 at 11:20 AM

Hello Brian,

I think there is some problem in the configuration.

Kindly post your makefile and output at https://github.com/sudar/Arduino-Makefile/issues and I will have someone to take a look at it.

Brian
March 24, 2015 at 10:18 AM

Thanks! I just submitted issue #327

moni
July 7, 2015 at 3:33 PM

Hi friends,
i am moni from delhi wanted to say thanks I'd have to start tweaking to get it to work, but now I can just use your repo. Cheers!

Thank you,

Sudar Post author
July 12, 2015 at 11:49 AM

Glad to know that you like it 🙂

Alex
July 12, 2015 at 7:40 AM

Hey guys, I'm having some issues. I switched from the debian version to directly pulling the git repo, and am having issues now that I didn't have before.

————————-
Arduino.mk Configuration:
– [AUTODETECTED] CURRENT_OS = LINUX
– [USER] ARDUINO_DIR = /usr/share/arduino
– [USER] ARDMK_DIR = ../../Arduino-Makefile
– [AUTODETECTED] ARDUINO_VERSION = 105
– [DEFAULT] ARCHITECTURE =
– [DEFAULT] VENDOR = arduino
– [DEFAULT] ARDUINO_SKETCHBOOK =
– [USER] AVR_TOOLS_DIR = /usr
– [COMPUTED] ARDUINO_LIB_PATH = /usr/share/arduino/libraries (from ARDUINO_DIR)
– [COMPUTED] ARDUINO_VAR_PATH = /usr/share/arduino/hardware/arduino//variants (from ARDUINO_DIR)
– [COMPUTED] BOARDS_TXT = /usr/share/arduino/hardware/arduino//boards.txt (from ARDUINO_DIR)
– [USER] USER_LIB_PATH = `pwd`/lib
– [DEFAULT] PRE_BUILD_HOOK = pre-build-hook.sh
– [USER] BOARD_TAG = uno
– [COMPUTED] CORE = arduino (from build.core)
– [COMPUTED] VARIANT = standard (from build.variant)
– [USER] OBJDIR = `pwd`/bin/uno/Code
– [COMPUTED] ARDUINO_CORE_PATH = /usr/share/arduino/hardware/arduino//cores/arduino (from

ARDUINO_DIR, BOARD_TAG and boards.txt)
– [USER] MONITOR_BAUDRATE = 115200
– [DEFAULT] OPTIMIZATION_LEVEL = s
– [DEFAULT] MCU_FLAG_NAME = mmcu
– [USER] CFLAGS_STD = -std=gnu11
– [USER] CXXFLAGS_STD = -std=gnu++11
– [COMPUTED] DEVICE_PATH = /dev/ttyACM0 (from MONITOR_PORT)
– [DEFAULT] FORCE_MONITOR_PORT =
– [AUTODETECTED] Size utility: AVR-aware for enhanced output
– [COMPUTED] BOOTLOADER_PARENT = /usr/share/arduino/hardware/arduino//bootloaders (from ARDUINO_DIR)
– [COMPUTED] ARDMK_VERSION = 1.5
– [COMPUTED] CC_VERSION = 4.8.1 (avr-gcc)
———————-
make: *** No rule to make target ‘elseif.s’, needed by ‘``pwd`/bin/uno/Code/elseif.o’. Stop.

Sudar Post author
July 12, 2015 at 11:49 AM

Can you post it as an issue in github? I (or some collaborator) will address it.

Simon
September 21, 2015 at 9:31 PM

shelling out to `pwd` is probably a bad idea, have you tried hardcoding those paths, or using an environment variable e.g. $(HOME)/blah

Thomas Kilian
September 21, 2015 at 8:49 PM

Hi, this is really great! Thanks a lot for sharing!

I have 2 remarks
1) above you use ARDUINO_PORT but according to the docs this is legacy and should be MONITOR_PORT.
2) In the sample Makefile you use
On OS X:
ARDUINO_DIR = /Applications/Arduino.app/Contents/Resources/Java
but the Resources is wrong. It must be
ARDUINO_DIR = /Applications/Arduino.app/Contents/Java

Thomas Kilian
September 22, 2015 at 7:26 PM

Me again. I'm trying to adapt the makefile to compile/upload for my Nano but it wont do. It complains that avr-g++: error: missing argument to '-mmcu='. Anything I can do about it?

Sudar Post author
September 24, 2015 at 5:45 PM

Create a new issue in github and specify your entire Makefile and the full error that you are getting.

Thomas Kilian
September 24, 2015 at 6:11 PM

Well. It turned out the error was sitting in front of the keyboard. I used 368 instead of 328 (mixing with 168) as sub-type. Make is nice but not so helpful when it comes to error messages. I can live with that.

Sudar Post author
September 24, 2015 at 6:22 PM

Glad to know that you got it working 🙂

Pingback: Arduino Development; There's a Makefile for That | Hackaday

Pingback: Arduino Development; There's a Makefile for That | Ad Pub

Pingback: Arduino Development; There's a Makefile for That | Dinesh Ram Kali.

Peter Ata
October 14, 2015 at 10:14 AM

Thanks for this package. The arduino ide is excellent for getting started but after a while I felt constrained by it. I much prefer to have an editor window (editor of my choice) and a terminal window with tabs for compiling and running a terminal.

The Arduino-Makefile package worked first time with a uno. When I tried to upload a file to a mega I received a "missing argument to '-mmcu='." message. After poking around I discovered that I had two copies of boards.txt. The "mega" data in one file was incomplete. After some editing the mega upload worked perfectly.

The problem was because of the way I had installed arduinio – version 1.6.5 from the arduino website and an earlier version from the Mageia repositories. I removed the package from the repository and now have only one copy.

To reiterate – my problem had nothing to do with the Arduino-Makefile but was a combination of me, Arduino and Mageia (-;

Again – thanks.

Sudar Post author
October 14, 2015 at 10:32 AM

Hello Peter,

Glad to know that you were able to figure it out 🙂

Mithat
November 4, 2015 at 3:46 PM

The just-released Arduino 1.6.6 includes arduino-builder (command -line tool). How will this influence the development of this project?

Sudar Post author
November 7, 2015 at 4:56 PM

I have not played around with arduino builder yet, so to be honest we don't know yet how it will influence the project 🙂

Nicolas BONOPERA
December 18, 2015 at 12:17 AM

Hi there,

For info, the is a small typo in Arduino.mk that is being hit when wanting to use picocom.
Please find the fix below.

Cheers,
Nico
(diff based on master branch, SHA1 = 9afa0e787cbb1fb2)
diff –git a/Arduino.mk b/Arduino.mk
index 7b1342e..c8cb023 100644
— a/Arduino.mk
+++ b/Arduino.mk
@@ -1504,10 +1504,11 @@ ifeq ($(MONITOR_CMD), 'putty')
else
$(MONITOR_CMD) -serial -sercfg $(MONITOR_BAUDRATE) $(call get_monitor_port)
endif
-else ifeq ($(MONITOR_CMD), picocom)
+else (ifeq ($(MONITOR_CMD), 'picocom')
$(MONITOR_CMD) -b $(MONITOR_BAUDRATE) $(MONITOR_PARAMS) $(call get_monitor_port)
-else
+ else
$(MONITOR_CMD) $(call get_monitor_port) $(MONITOR_BAUDRATE)
+)
endif

Sudar Post author
December 21, 2015 at 12:05 PM

Hello Nicolas,

Thanks for reporting this.

Can you please raise it as an issue in github? https://github.com/sudar/Arduino-Makefile/issues

Thanks.

Rik
December 29, 2015 at 4:55 AM

Good Work, thanks a lot. and it works perfectly! I tried for Arduino Mega 2560.

Suleman
February 16, 2016 at 3:27 PM

Hello Sudar

I want to assign some value to my variable without using arduino IDE,I already upload the sketch and there is some variable with values must be change by changing internet,example port defining and i want this by just plug my device to computer with usb and change it through perl and the value of port must be store in the arduino sketch.

Thanks

Pingback: How to Program an AVR/Arduino using the Raspberry Pi GPIO - ozzmaker.com

mercan

April 10, 2016 at 1:45 AM

Dear Sudar,

Thank you for maintaining the makefile for Arduino. I'm a recent member of the crowd that find it very useful.

I would like to program an attiny85 using my Arduino board. I have seen your attiny example in the github repository. Is this to be uses with a dedicated attiny85 programmer or can I use them to program an attiny85 via the Arduino. If so, would I need to do any modifications?

Thanks a lot in advance for your advice!

---

artisell
October 30, 2016 at 11:43 AM

If you want to download ARD video to MP4, AVI, etc or download ARD audio to MP3, AAC, etc , I highly recommend Allavsoft, it is the best ARD downloader.

Pingback: Programación Arduino | Aprendiendo Arduino

Nicolas
December 6, 2016 at 8:45 PM

Hi Sudar!

Great job! Your project seems to be really great! Does your project support M0 target?

I try to use your project to compile to an Arduino M0. As its cpu is ARM Cortex-M0+ and not an AVR one, I try to use arm-none-eabi toolchain instead of avr. So I defined these following variables in my Makefile:

```
AVR_TOOLS_DIR = $(ARDUINO_DIR)/hardware/tools/gcc-arm-none-eabi-4.8.3-2014q1
TOOLCHAIN_PREFIX = arm-none-eabi
CC_NAME = $(TOOLCHAIN_PREFIX)-gcc
CXX_NAME = $(TOOLCHAIN_PREFIX)-g++
```

OBJCOPY_NAME = $(TOOLCHAIN_PREFIX)-objcopy
OBJDUMP_NAME = $(TOOLCHAIN_PREFIX)-objdump
AR_NAME = $(TOOLCHAIN_PREFIX)-ar
SIZE_NAME = $(TOOLCHAIN_PREFIX)-size
NM_NAME = $(TOOLCHAIN_PREFIX)-nm

I also had to suppress g++ mcu option because I had one error (like “unknown option”).

Then I get many errors… I tried to had include folders but there is still many problems.

So I don’t know if my usage is not correct or my target is unsupported…

Thank you in advance if you could help me! (and apologize for my awful english…)

Here are my logs:
——————————-
Arduino.mk Configuration:
– [AUTODETECTED] CURRENT_OS = LINUX
– [USER] ARDUINO_DIR = /home/nicolas/arduino-1.7.11-linux64
– [USER] ARDMK_DIR = /home/nicolas/workspace-arduino/Arduino-Makefile
– [AUTODETECTED] ARDUINO_VERSION = 1711
– [USER] ARCHITECTURE = samd
– [DEFAULT] ARDMK_VENDOR = arduino
– [AUTODETECTED] ARDUINO_PREFERENCES_PATH = /home/nicolas/.arduino15/preferences.txt
– [AUTODETECTED] ARDUINO_SKETCHBOOK = /home/nicolas/Arduino (from arduino preferences file)
– [USER] AVR_TOOLS_DIR = /home/nicolas/arduino-1.7.11-linux64/hardware/tools/gcc-arm-none-eabi-4.8.3-2014q1
– [COMPUTED] ARDUINO_LIB_PATH = /home/nicolas/arduino-1.7.11-linux64/libraries (from ARDUINO_DIR)
– [COMPUTED] ARDUINO_PLATFORM_LIB_PATH = /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/libraries (from ARDUINO_DIR)
– [COMPUTED] ARDUINO_VAR_PATH = /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/variants (from ARDUINO_DIR)
– [COMPUTED] BOARDS_TXT = /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/boards.txt (from ARDUINO_DIR)
– [DEFAULT] USER_LIB_PATH = /home/nicolas/Arduino/libraries (in user sketchbook)
– [DEFAULT] PRE_BUILD_HOOK = pre-build-hook.sh
– [USER] BOARD_TAG = M0
– [COMPUTED] CORE = (from build.core)
– [COMPUTED] VARIANT = (from build.variant)
– [COMPUTED] OBJDIR = build-M0 (from BOARD_TAG)
– [DEFAULT] ARDUINO_CORE_PATH = /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino
– [USER] MONITOR_BAUDRATE = 1200
– [DEFAULT] OPTIMIZATION_LEVEL = s

– [DEFAULT] MCU_FLAG_NAME = mmcu
– [DEFAULT] CFLAGS_STD =
– [DEFAULT] CXXFLAGS_STD =
– [COMPUTED] DEVICE_PATH = /dev/ttyACM* (from MONITOR_PORT)
– [DEFAULT] FORCE_MONITOR_PORT =
– [AUTODETECTED] Size utility: Basic (not AVR-aware)
– [COMPUTED] BOOTLOADER_PARENT = /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/bootloaders (from ARDUINO_DIR)
– [COMPUTED] ARDMK_VERSION = 1.5
– [COMPUTED] CC_VERSION = 4.8.3 (arm-none-eabi-gcc)
————————————-
mkdir -p build-M0
/home/nicolas/arduino-1.7.11-linux64/hardware/tools/gcc-arm-none-eabi-4.8.3-2014q1/bin/arm-none-eabi-g++ -x c++ -include Arduino.h -MMD -c -DF_CPU= -DARDUINO=1711 -DARDUINO_ARCH_SAMD -D__PROG_TYPES_COMPAT__ -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/variants/ -Wall -ffunction-sections -fdata-sections -Os -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/sam/system/CMSIS/Device/ATMEL -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/variants/arduino_zero -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/USB -I/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/avr -I/home/nicolas/arduino-1.7.11-linux64/hardware/tools/CMSIS/CMSIS/Include -I/home/nicolas/arduino-1.7.11-linux64/hardware/tools/CMSIS/Device/ATMEL/samd21/include -I/home/nicolas/arduino-1.7.11-linux64/hardware/tools/CMSIS/Device/ATMEL/samd21/include/component -I/home/nicolas/arduino-1.7.11-linux64/hardware/tools/CMSIS/Device/ATMEL/samd21/include/instance -I/home/nicolas/arduino-1.7.11-linux64/hardware/tools/CMSIS/Device/ATMEL/samd21/include/pio -DTC_INST_NUM=3 -DTCC_INST_NUM=3 -fpermissive -fno-exceptions main.ino -o build-M0/main.ino.o
In file included from /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/variants/arduino_zero/variant.h:39:0,
from /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/delay.h:27,
from /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/Arduino.h:70,
from :0:
/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/SERCOM.h:146:16: error: expected ‘)’ before ‘*’ token
SERCOM(Sercom* s) ;
^
In file included from /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/variants/arduino_zero/variant.h:39:0,
from /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/delay.h:27,
from /home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/Arduino.h:70,
from :0:
/home/nicolas/arduino-1.7.11-linux64/hardware/arduino/samd/cores/arduino/SERCOM.h:213:3: error: ‘Sercom’ does not name a type
Sercom* sercom;

—-
Regards.
Nicolas

Kardacian
December 11, 2016 at 5:24 PM

Hello,

I have been trying to install this on OSX latest versions with no luck. Do you have detailed instructions to do a manual install?

Thanks.

DP26
January 24, 2017 at 2:00 PM

Thanks man, you saving my day! 😀

Kardacian
April 12, 2017 at 6:04 PM

Sudar I added functionality to the Arduino IDE to compile and upload to an ESP8266 dev board. Its becoming very popular and I have a few projects that I will be using it with. It all works fine in the IDE. I have tried to get this working with the Arduino-makefile which work great with all my Arduino sketches, but no luck with the esp8266. Is it possible to get this to work? I am not strong with make so I have been doing some guess work without much success.

This is the board manager used to install the esp8266 board
http://arduino.esp8266.com/stable/package_esp8266com_index.json

Module I am using is “NodeMCU 1.0 (ESP-12E Module)”

Thanks

Pingback: Installing an Arduino Wifi Shield