Arduino

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It is intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Contents

[hide]

Installation

- Install **`arduino`** and **`arduino-docs`** for its offline documentation.
- Add yourself to the `uucp` and `lock` **groups** (more information in the **#Accessing serial** section).
- You may need to **load** the `cdc_acm` module.

AVR Boards

To use AVR boards such as the Arduino Uno you can install **`arduino-avr-core`** optionally to use archlinux upstream avr-gcc instead of the bundled older avr-core. If you still want to use the older arduino-core you need to **install it in the board manager**. You can always switch between the different cores in the "Tools>Board" menu.

Pinoccio Scout

Pinoccio Scouts can also be programmed using the Arduino IDE. Instructions can be found **here**. Alternative you can install **`arduino-pinoccio`**AUR from the **AUR**.

Intel Galileo

To use the Intel Galileo boards with Archlinux install the Arduino IDE and download the Galileo tools package via "Tools->Board->Boards Manager". To fix the installation you can follow **this github post**.

On Arm7 devices

See **here** for a work around.

RedBear Duo

You might need to install `perl-archive-zip` or you'll get an error about missing crc32.

Configuration

Accessing serial

The arduino board communicates with the computer via a serial connection or a serial over USB connection. So the user needs read/write access to the serial device file. **Udev** creates files in `/dev/tts/` owned by group `uucp` so adding the user to the `uucp` group gives the required read/write access. If you are planning to use the default Java IDE, add your user to the lock group for `/var/lock/lockdev` access.

```
# gpasswd -a $USER uucp
# gpasswd -a $USER lock
```

Note: You will have to logout and login again for this to take effect.

Before uploading to the Arduino, be sure to set the correct serial port, board, and processor from the Tools menu.

stty

Preparing:

```
# stty -F /dev/ttyACM0 cs8 9600 ignbrk -brkint -imaxbel -opost -onlcr -isig -
icanon -iexten -echo -echoe -echok -echoctl -echoke noflsh -ixon -crtscts
```

Sending commands through Terminal without new line after command

```
# echo -n "Hello World" > /dev/ttyACM0
```

Note: As autoreset on serial connection is activated by default on most boards, you need to disable this feature if you want to communicate directly with your board with the last command instead of a terminal emulator (arduino IDE, screen, picocom...). If you have a Leonardo board, you are not concerned by this, because it does not autoreset. If you have an Uno board, connect a 10 µF capacitor between the RESET and GND pins. If you have another board, connect a 120 ohms resistor between the RESET and 5V pins.
See **http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection** for more details.

Reading what your Arduino has to tell you

```
$ cat /dev/ttyACM0
```

Arduino-Builder

You can also build Arduino sketches with the `arduino-builder` command line tool. In order to use the provided `arduino-avr-core` with upstream `avr-gcc` and `avrdude` you need to create a small settings file:

```
build.options.json

{
    "fqbn": "archlinux-arduino:avr:uno",
    "hardwareFolders": "/usr/share/arduino/hardware",
    "toolsFolders": "/usr/bin"
}
```

Compile a sketch with:

```
$ arduino-builder -build-options-file build.options.json blink.ino
```

Or pass all options via command line:

```
$ arduino-builder -fqbn archlinux-arduino:avr:uno -hardware /usr/share/arduin
o/hardware -tools /usr/bin blink.ino
```

Alternatives for IDE

ArduIDE

ArduIDE is a Qt-based IDE for Arduino. `arduide-git`AUR is available in the **AUR**.

gnoduino

`gnoduino`AUR is an implementation of original Arduino IDE for GNOME available in the **AUR**. The original Arduino IDE software is written in Java. This is a Python implementation and it is targeted at GNOME but will work on xfce4 and other WM. Its purpose is to be light, while maintaining compatibility with the original Arduino IDE. The source editor is based on gtksourceview.

If you prefer working from terminal, below there are some other options to choose from.

Arduino-CMake

Using **arduino-cmake** and **CMake** you can build Arduino firmware from the command line using multiple build systems. CMake lets you generate the build system that fits your needs, using the tools you like. It can generate any type of build system, from simple Makefiles, to complete projects for Eclipse, Visual Studio, XCode, etc.

Requirements: `cmake`, `arduino`, `avr-gcc`, `avr-binutils`, `avr-libc`, `avrdude`.

Ino

Ino is a command line toolkit for working with arduino hardware. `ino`AUR is available in the **AUR**.

Makefile

Note: Update 2015-03-23. Due to recent changes in Arduino ≥v1.5, many old Makefiles do not work without some modification. A simple Makefile for Arduino version 1.5+ can be found **on GitHub**.

Instead of using the Arduino IDE it is possible to use another editor and a Makefile.

Set up a directory to program your Arduino and copy the Makefile into this directory. A copy of the Makefile can be obtained from `/usr/share/arduino/hardware/cores/arduino/Makefile`

You will have to modify this a little bit to reflect your settings. The makefile should be pretty self explanatory. Here are some lines you may have to edit.

```
PORT = usually /dev/ttyUSBx, where x is the usb serial port your arduino is p
lugged into
TARGET = your sketch's name
ARDUINO = /usr/share/arduino/lib/targets/arduino
```

Depending on which library functions you call in your sketch, you may need to compile parts of the library. To do that you need to edit your SRC and CXXSRC to include the required libraries.

Now you should be able to `make && make upload` to your board to execute your sketch.

Arduino-mk

arduino-mkAUR is another alternative Makefile approach. It allows users to have a local Makefile that includes Arduino.mk. See **project page** for usage.

For Arduino 1.5, try the following local Makefile (because Arduino 1.5's library directory structure is slightly different):

```
ARDUINO_DIR = /usr/share/arduino
ARDMK_DIR = /usr/share/arduino
AVR_TOOLS_DIR = /usr
ARDUINO_CORE_PATH = /usr/share/arduino/hardware/arduino/avr/cores/arduino
BOARDS_TXT = /usr/share/arduino/hardware/arduino/avr/boards.txt
ARDUINO_VAR_PATH = /usr/share/arduino/hardware/arduino/avr/variants
BOOTLOADER_PARENT = /usr/share/arduino/hardware/arduino/avr/bootloaders


BOARD_TAG    = uno
ARDUINO_LIBS =

include /usr/share/arduino/Arduino.mk
```

In some cases you could need to install **avr-libc** and **avrdude**.

Scons

Using **scons** together with **arscons** it is very easy to use to compile and upload Arduino projects from the command line. Scons is based on python and you will need python-pyserial to use the serial interface. Install **python-pyserial** and **scons**.

That will get the dependencies you need too. You will also need Arduino itself so install it as described above. Create project directory (eg. test), then create a arduino project file in your new directory. Use the same name as the directory and add .ino (eg. test.ino). Get the **SConstruct** script from arscons and put it in your directory. Have a peek in it and, if necessary, edit it. It is a python script. Edit your project as you please, then run

```
$ scons                  # This will build the project
$ scons upload           # This will upload the project to your Arduino
```

PlatformIO

PlatformIO is a python tool to build and upload sketches for multiple Hardware Platforms, at the moment of writing these are Arduino/AVR based boards, TI MSP430 and TI TM4C12x Boards. In the near future the author plans to add a library function that allows to search and include libraries directly from GitHub.

Installation

Install the `platformio`AUR or `platformio-git`AUR package.

Usage

```
$ platformio platforms install atmelavr
$ platformio init
$ vim platformio.ini
#
# Atmel AVR based board + Arduino Wiring Framework
#
[env:ArduinoMega2560]
platform = atmelavr
framework = arduino
board = megaatmega2560
upload_port = /dev/ttyACM0
targets = upload
$ platformio run
```

Emacs

It is possible to configure **Emacs** as IDE.

Install the package `emacs-arduino-mode-git`AUR from the **AUR** in order to enable the `arduino-mode` in emacs.

Add to the init script:

```
~/.emacs
```

```
;; arduino-mode
(require 'cl)
(autoload 'arduino-mode "arduino-mode" "Arduino editing mode." t)
(add-to-list 'auto-mode-alist '("\.ino$" . arduino-mode))
```

You can compile and upload the sketches using `Arduino-mk` (see above) with `M-x compile` `make upload`.

Main resource: **here**.

Troubleshooting

Version 1.6

Some older 3rd party tools may only work with Arduino 1.0 (`arduino10`AUR). Some of the tools may partially work for Arduino version 1.6 (`arduino`) and after. Check the version if the tools do not work. Note that some newer boards do not work with the old Arduino IDE.

Consistent naming of Arduino devices

If you have more than one arduino you may have noticed that they names /dev/ttyUSB[0-9] are assigned in the order of connection. In the IDE this is not so much of a problem, but when you have programmed your own software to communicate with an arduino project in the background this can be annoying. Use the following udev rules to assign static symlinks to your arduino's:

```
/etc/udev/rules.d/52-arduino.rules

SUBSYSTEMS=="usb", KERNEL=="ttyUSB[0-9]*", ATTRS{idVendor}=="0403", ATTRS{idP
roduct}=="6001", SYMLINK+="sensors/ftdi_%s{serial}"
```

Your arduino's will be available under names like "/dev/sensors/ftdi_A700dzaF". If you want you can also assign more meaningfull names to several devices like this:

```
/etc/udev/rules.d/52-arduino.rules

SUBSYSTEMS=="usb", KERNEL=="ttyUSB[0-9]*", ATTRS{idVendor}=="0403", ATTRS{idP
roduct}=="6001", ATTRS{serial}=="A700dzaF", SYMLINK+="arduino/nano"
```

which will create a symlink in /dev/arduino/nano to the device with the specified serialnumber. You do need to unplug and replug your arduino for this to take effect or run

```
udevadm trigger
```

Common `idVendor`/`idProduct` pairs can be found in `hardware/arduino/avr/boards.txt` in the distribution package. Note that some of them (notably **FTDI** ones) are not unique to the Arduino platform. Using `serial` attribute is a good way to distinguish between various devices.

Error opening serial port

You may see the serial port initially when the IDE starts, but the TX/RX leds do nothing when uploading. You may have previously changed the baudrate in the serial monitor to something it does not like. Edit ~/.arduino/preferences.txt so that serial.debug_rate is a different speed, like 115200.

Working with Uno/Mega2560

The Arduino Uno and Mega2560 have an onboard USB interface (an Atmel 8U2) that accepts serial data, so they are accessed through /dev/ttyACM0 created by the cdc-acm kernel module when it is plugged in.

The 8U2 firmware may need an update to ease serial communications. See **[1]** for more details and reply #11 for a fix. The original arduino bbs, where you can find an image explaining how to get your Uno into DFU, is now in a read-only state. If you do not have an account to view the image, see **[2]**.

You can perform a general function test of the Uno by putting it in loopback mode and typing characters into the arduino serial monitor at 115200 baud. It should echo the characters back to you. To put it in loopback, short pins 0 -> 1 on the digital side and either hold the reset button or short the GND -> RESET pins while you type.

Application not resizing with WM, menus immediately closing

see **Java#Applications not resizing with WM, menus immediately closing**

See also

- **https://bbs.archlinux.org/viewtopic.php?pid=295312**
- **https://bbs.archlinux.org/viewtopic.php?pid=981348**

- http://answers.ros.org/question/9097/how-can-i-get-a-unique-device-path-for-my-arduinoftdi-device/