Setting up the Raspberry Pi to program the ATtiny84

Audience

This document has been prepared for people with a working knowledge of the Raspberry Pi and hardware experience at the level of connecting an external device to the Raspberry Pi GPIO interface. It assumes some familiarity with the "C" programming language in order to create program code that can be loaded into an ATtiny84 processor.

Objective

The document describes how to set up the hardware and software environments so that programs can be developed on a Raspberry Pi and transferred to an ATtiny84. While the objective is to create a development environment it is necessary to demonstrate the environment working in practice and to this end a simple "blink" program is used as an example in order to make an LED attached to the ATtiny84 flash on and off. The intention is that the end user should be able to develop programs for the ATtiny84 to work with whatever peripheral hardware the program design requires.

Outline

There are two main components for the software environment. These are:

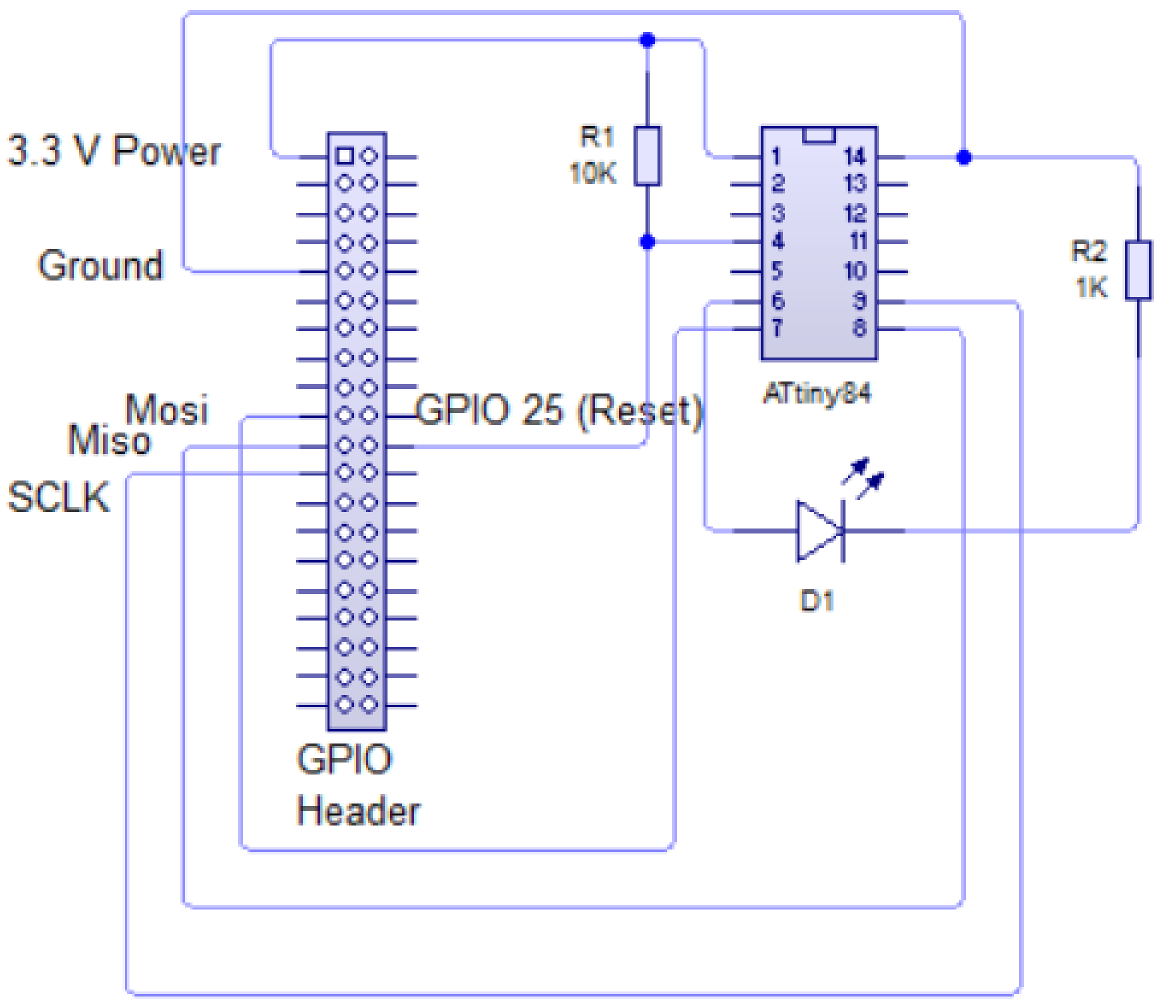
- the Arduino IDE which is used to enter program code and compile it into the binary file needed by the ATtiny84
- the Avrdude software that enables the binary file to be loaded into the ATtiny84 using the SPI communication interface.

The hardware consists of a way of mounting the ATtiny84, either on a breadboard or by means of a ZIF socket. There are six wire connections that have to be made between the GPIO and the ATtiny84. In addition, a pull-up resistor is needed on the ATtiny84 reset pin and an LED and series resistor must be provided to show the demonstration program working.

Pre-requisites

The instructions that follow have been tested on a Raspberry Pi 2 Model B and a Raspberry Pi 3 Model B. In either case the SD card must be set up to use the Jessie version of Raspbian.

The ATtiny84 must be connected to the Raspberry Pi as shown below before continuing with the software installation.



Explanation:

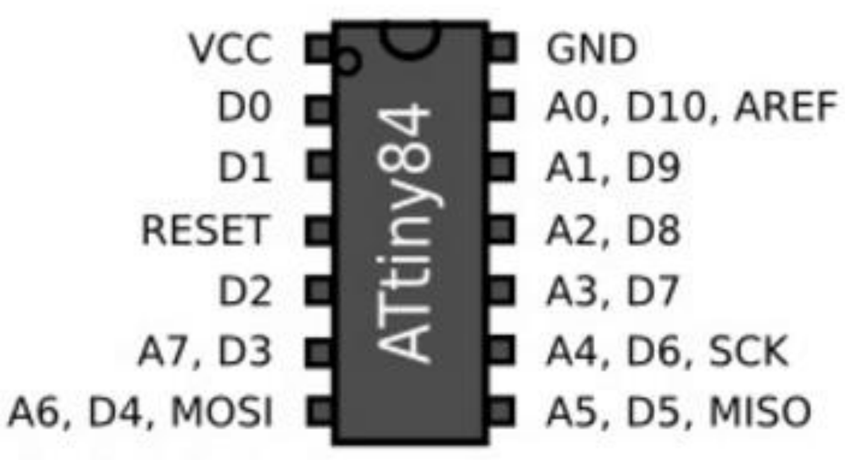
The ATtiny must be powered from the 3.3 volt connection on the Raspberry Pi. Any higher voltage would damage the Raspberry Pi.

Communication is via the SPI interface that uses three lines, Mosi (Master out, slave in), Miso (Master in, slave out) and SCLK (Synchronising clock). The relevant pins on the GPIO header are connected to the equivalent ones on the ATtiny84.

To program the ATtiny84 the avrdude interface must be able to reset the chip. This is accomplished using GPIO 25 as this is the default used by avrdude. This line is pulled up to supply via a 10k resistor in order to avoid spurious resets.

The example Blink sketch is configured to operate an LED on logical output 7 which is physical pin 6 on the chip. A current limiting resistor must be included in circuit. Any value from about 220 ohms to 1k ohms should suffice.

The ATtiny84 pin functions are shown here:



Raspberry Pi Environment

The following instructions assume that you are using the Raspberry Pi GUI. That is, you have a display, keyboard and mouse connected to the Raspberry Pi. You must also have a network connection giving internet access to the Raspberry Pi.

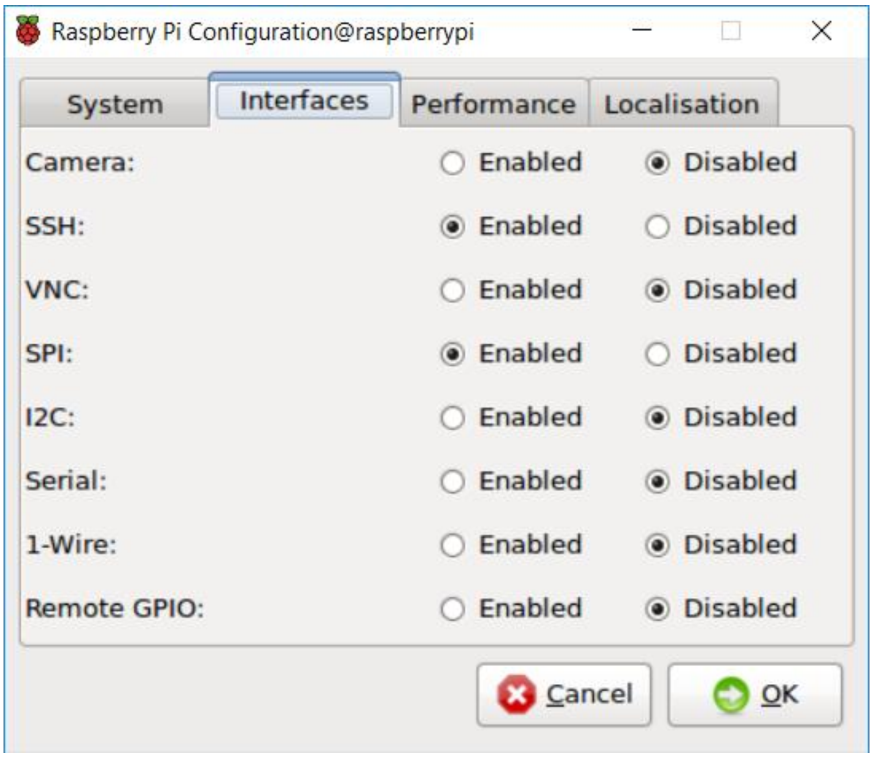
It is possible to perform the setup with the Raspberry Pi running “headless” (using ssh to connect to another computer and without display, keyboard or mouse). However, it is necessary to run an lxsession instance in order to be able to copy and paste urls as described later.

Step 1 – Enable SPI

Go to the Raspberry Pi menu and select Preferences and then Raspberry Pi Configuration.

Select the Interfaces tab and enable SPI.

OK the dialogue.


Raspberry Pi Configuration@raspberrypi

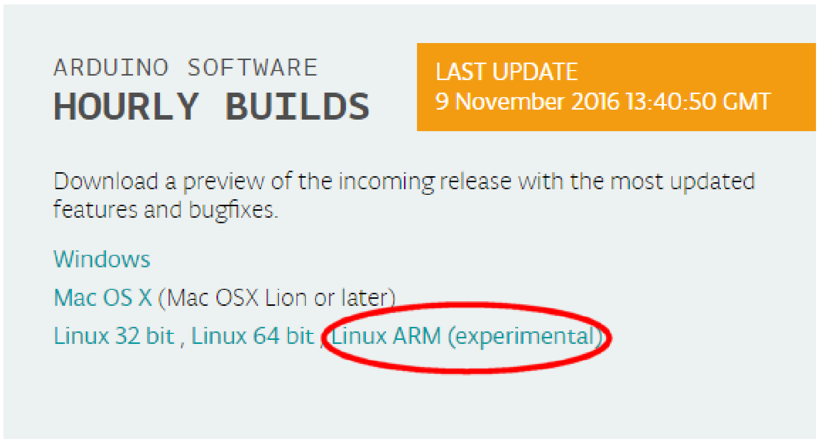
Camera:	○ Enabled	● Disabled
SSH:	● Enabled	○ Disabled
VNC:	○ Enabled	● Disabled
SPI:	● Enabled	○ Disabled
I2C:	○ Enabled	● Disabled
Serial:	○ Enabled	● Disabled
1-Wire:	○ Enabled	● Disabled
Remote GPIO:	○ Enabled	● Disabled

Step 2 - Arduino IDE

2.1 Open the web browser on your Pi and enter the following URL

https://www.arduino.cc/en/Main/Software

2.2 Go down to the Hourly Builds section and click on Linux ARM as shown below



ARDUINO SOFTWARE
HOURLY BUILDS

LAST UPDATE
9 November 2016 13:40:50 GMT

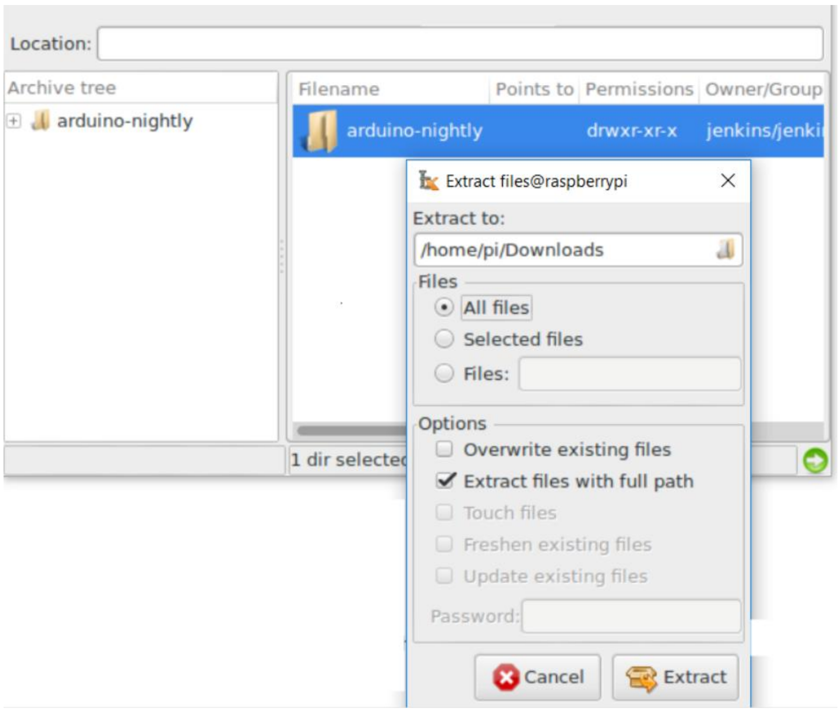
Download a preview of the incoming release with the most updated features and bugfixes.

Windows
Mac OS X (Mac OSX Lion or later)
Linux 32 bit , Linux 64 bit , Linux ARM (experimental)

2.3 This downloads a tarred archive to your Downloads directory.

2.4 Open File Manager, go to Downloads and right click on the tar archive arduino-nightly-linuxarm.tar.tz

2.5 Select Archiver. This opens a dialogue that says "Opening archive please wait". It takes quite a long time but then you see the file name appear on the right of the dialogue. Click the Extract Files icon and select "All files".



Filename	Points to	Permissions	Owner/Group
arduino-nightly		drwxr-xr-x	jenkins/jenki

Extract files@raspberrypi

Extract to: /home/pi/Downloads

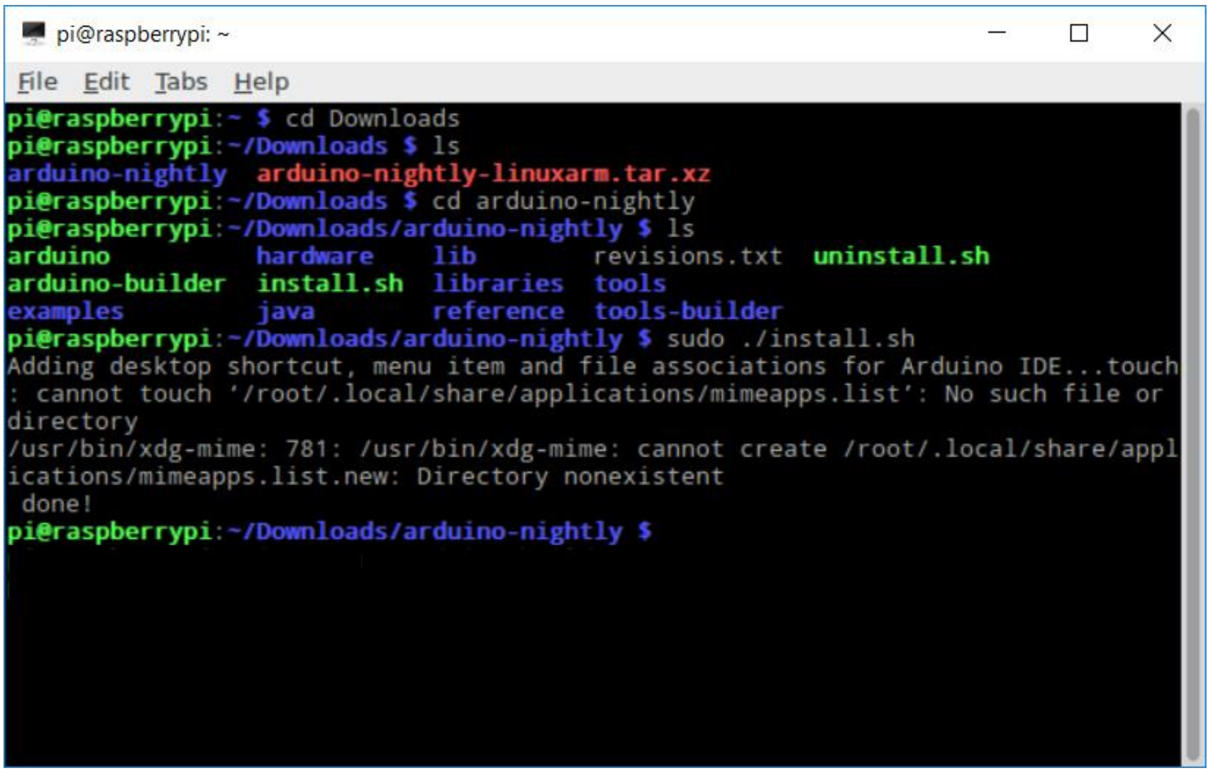
Files
- [✓ All files]
- Selected files
- Files:

Options
- Overwrite existing files
- [✓] Extract files with full path
- Touch files
- Freshen existing files
- Update existing files

Password:

2.6 Click on "Extract" and wait for it to complete.

2.7 When it has done, bring up an instance of Terminal and navigate to the directory containing the expanded archive then run the install script from sudo (see screenshot below).

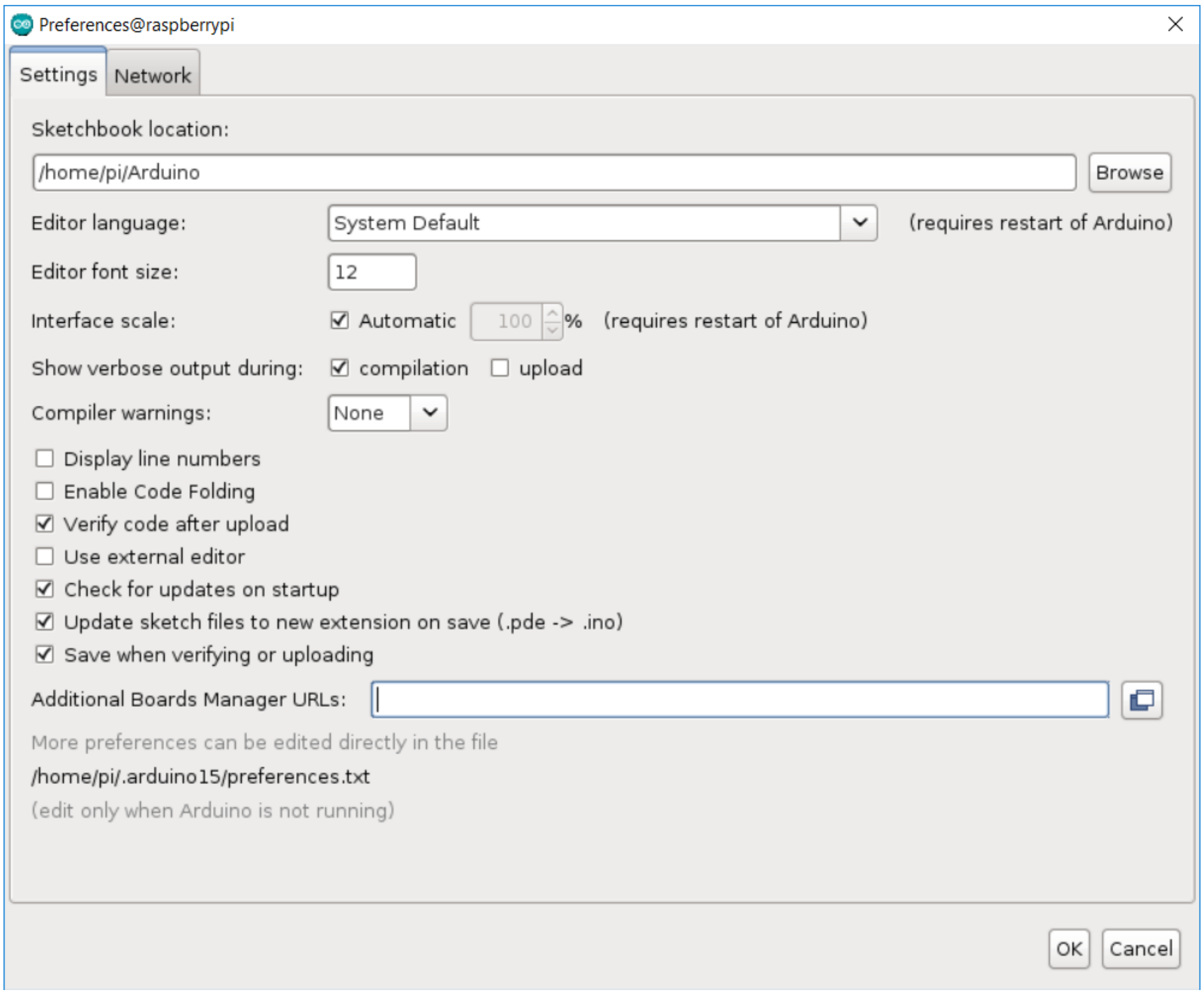


```
pi@raspberrypi:~ $ cd Downloads
pi@raspberrypi:~/Downloads $ ls
arduino-nightly  arduino-nightly-linuxarm.tar.xz
pi@raspberrypi:~/Downloads $ cd arduino-nightly
pi@raspberrypi:~/Downloads/arduino-nightly $ ls
arduino          hardware    lib         revisions.txt  uninstall.sh
arduino-builder  install.sh  libraries   tools
examples         java        reference   tools-builder
pi@raspberrypi:~/Downloads/arduino-nightly $ sudo ./install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE...touch
: cannot touch '/root/.local/share/applications/mimeapps.list': No such file or
directory
/usr/bin/xdg-mime: 781: /usr/bin/xdg-mime: cannot create /root/.local/share/appl
ications/mimeapps.list.new: Directory nonexistent
 done!
pi@raspberrypi:~/Downloads/arduino-nightly $
```

Note that the error messages are normal and can be ignored.

2.8 When this has finished, look on the Pi Menu under Programming where Arduino IDE should be at the top. Click it to start Arduino.

2.9 On the Arduino screen, go to File / Preferences and select "Show verbose output during compilation"



Sketchbook location: /home/pi/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☑ compilation ☐ upload

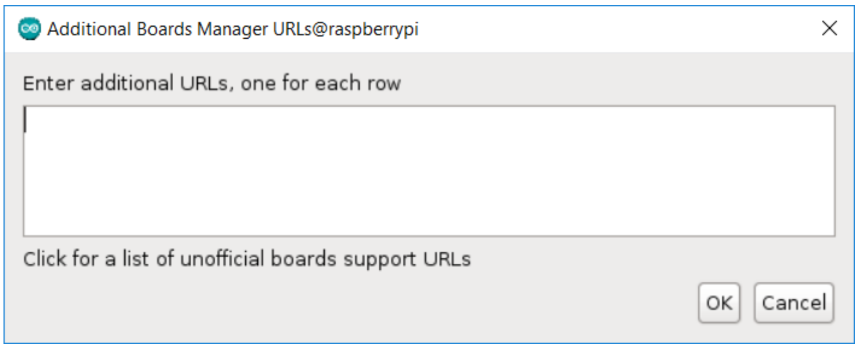
Compiler warnings: None

☐ Display line numbers
☐ Enable Code Folding
☑ Verify code after upload
☐ Use external editor
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

Additional Boards Manager URLs:

More preferences can be edited directly in the file
/home/pi/.arduino15/preferences.txt
(edit only when Arduino is not running)

2.10 Click the button to the right of the field labelled “Additional Boards Manager URLs”. This opens the dialogue shown below.



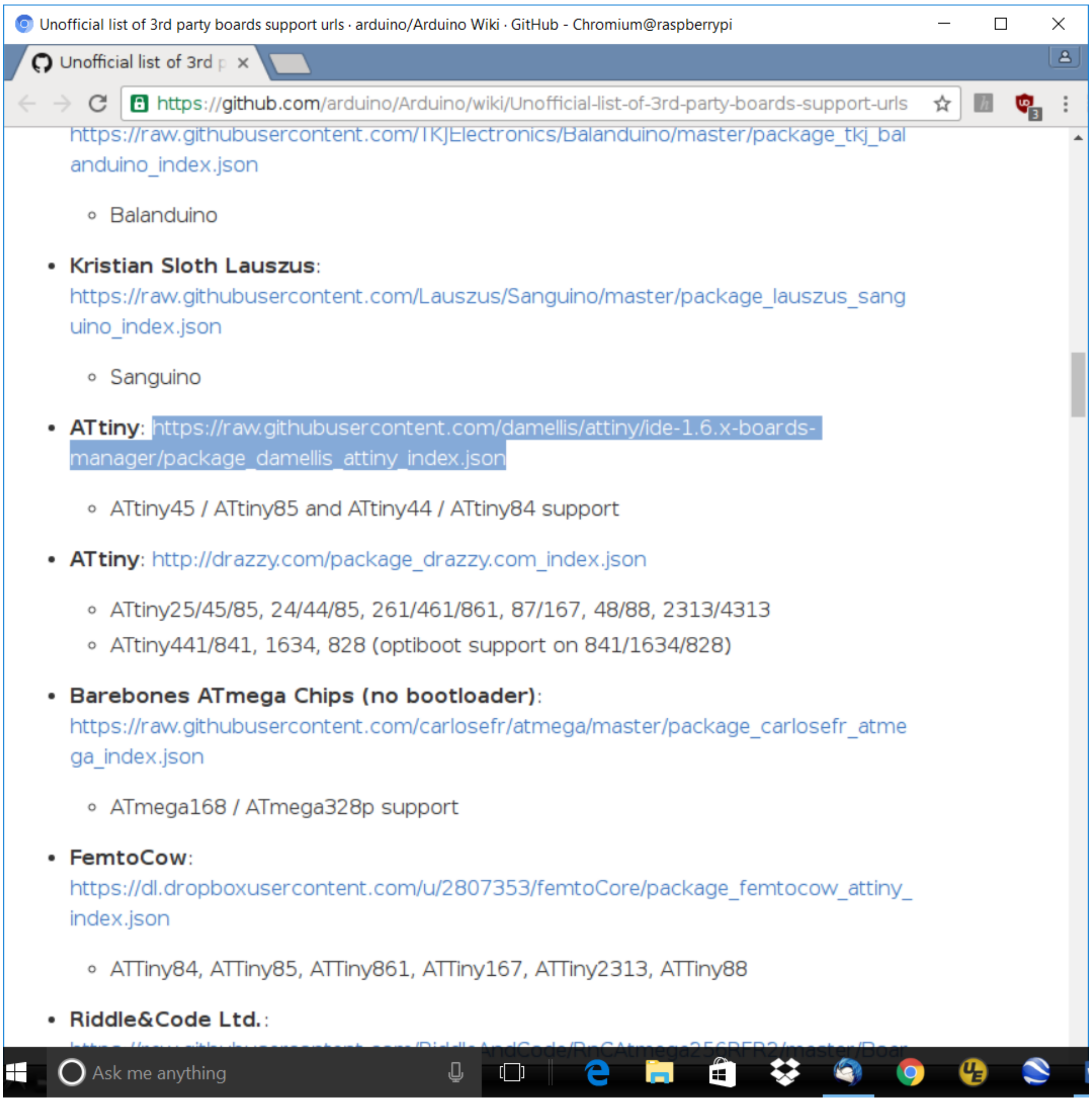
Enter additional URLs, one for each row

Click for a list of unofficial boards support URLs

2.11 Click on the words “Click for a list of unofficial boards support URLs” and this opens the Raspeberry Pi web browser and takes you to

https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls

2.12 Scroll down this list to the first item labelled ATtiny.

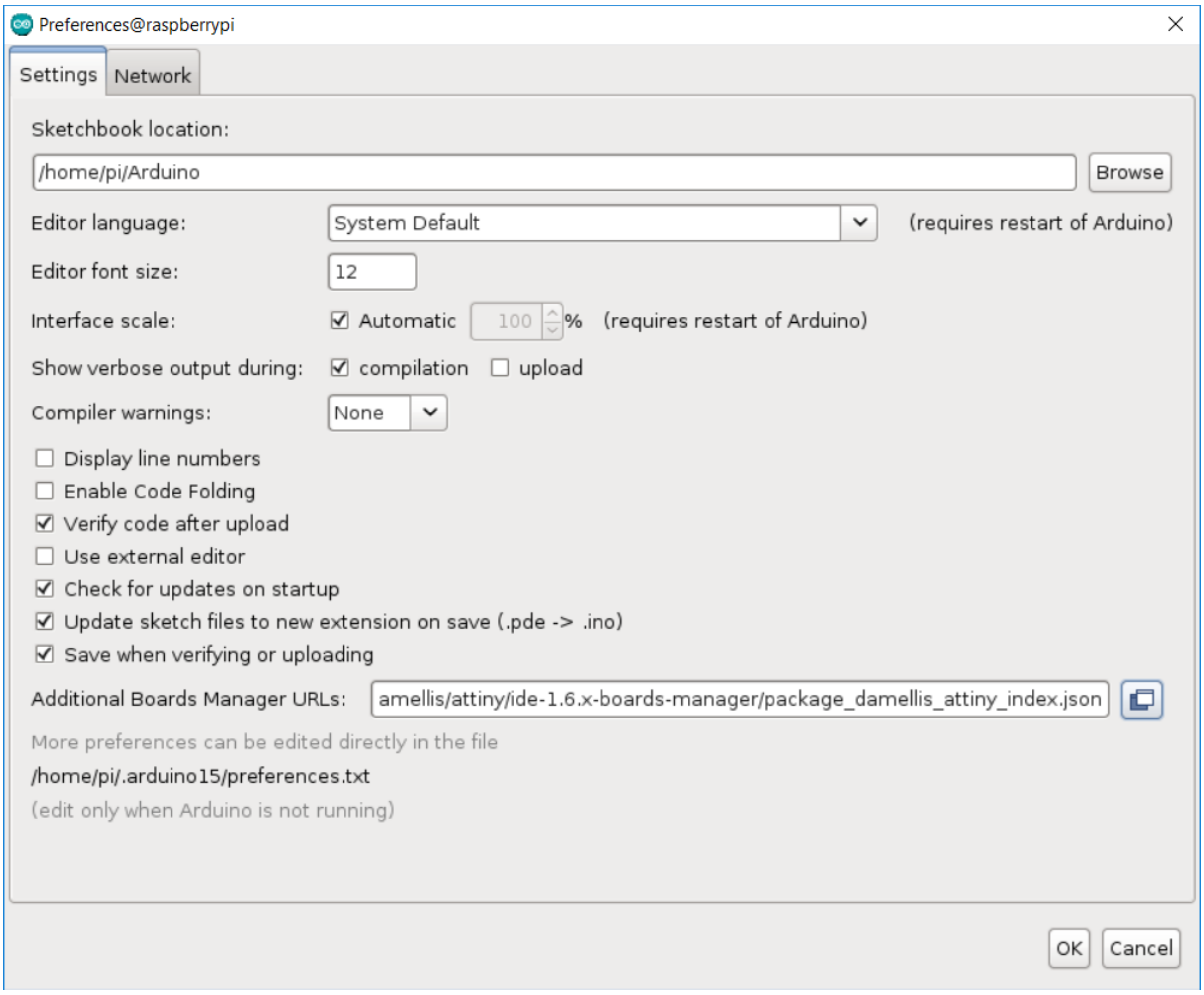

Unofficial list of 3rd party boards support urls · arduino/Arduino Wiki · GitHub - Chromium@raspberrypi
Unofficial list of 3rd p
https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls
Ask me anything

https://raw.githubusercontent.com/TKJElectronics/Balanduino/master/package_tkj_balanduino_index.json

- Balanduino

- **Kristian Sloth Lauszus**: https://raw.githubusercontent.com/Lauszus/Sanguino/master/package_lauszus_sanguino_index.json
  - Sanguino
- **ATtiny**: https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json
  - ATtiny45 / ATtiny85 and ATtiny44 / ATtiny84 support
- **ATtiny**: http://drazzy.com/package_drazzy.com_index.json
  - ATtiny25/45/85, 24/44/85, 261/461/861, 87/167, 48/88, 2313/4313
  - ATtiny441/841, 1634, 828 (optiboot support on 841/1634/828)
- **Barebones ATmega Chips (no bootloader)**: https://raw.githubusercontent.com/carlosefr/atmega/master/package_carlosefr_atmega_index.json
  - ATmega168 / ATmega328p support
- **FemtoCow**: https://dl.dropboxusercontent.com/u/2807353/femtoCore/package_femtocow_attiny_index.json
  - ATTiny84, ATTiny85, ATTiny861, ATTiny167, ATTiny2313, ATTiny88
- **Riddle&Code Ltd.**:

2.13 Copy the url (https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json) and paste it into the dialogue at step 2.10 above. OK that dialogue. The Preference dialogue should now look like this:


Preferences@raspberrypi

Settings | Network

Sketchbook location: /home/pi/Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☑ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers
☐ Enable Code Folding
☑ Verify code after upload
☐ Use external editor
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

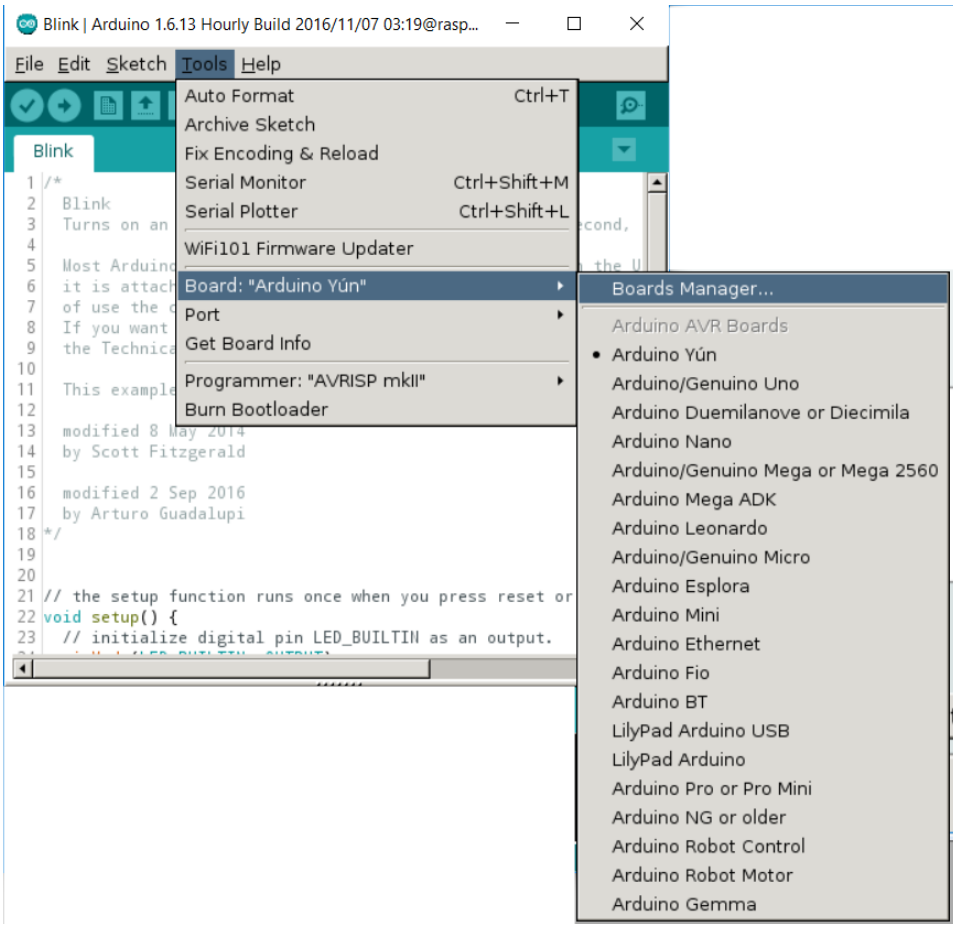
Additional Boards Manager URLs: amellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

More preferences can be edited directly in the file
/home/pi/.arduino15/preferences.txt
(edit only when Arduino is not running)

[OK] [Cancel]

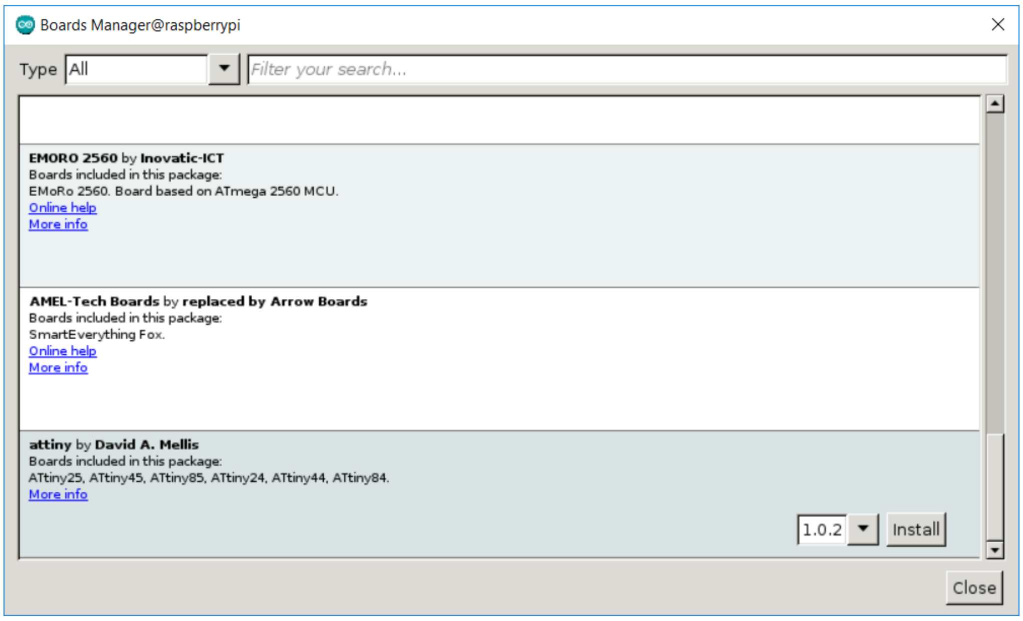
2.14 OK the preferences dialogue

2.15 Next select the Tools Menu and go to Board and then Boards Manager.



```
/*
  Blink
  Turns on an                                          cond,

  Most Arduino                                         the U
  it is attach
  of use the c
  If you want
  the Technica

  This example

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
```

2.16 When the dialogue comes up, scroll to the bottom and select the final entry (ATtiny) as shown below. Click Install and then Close.

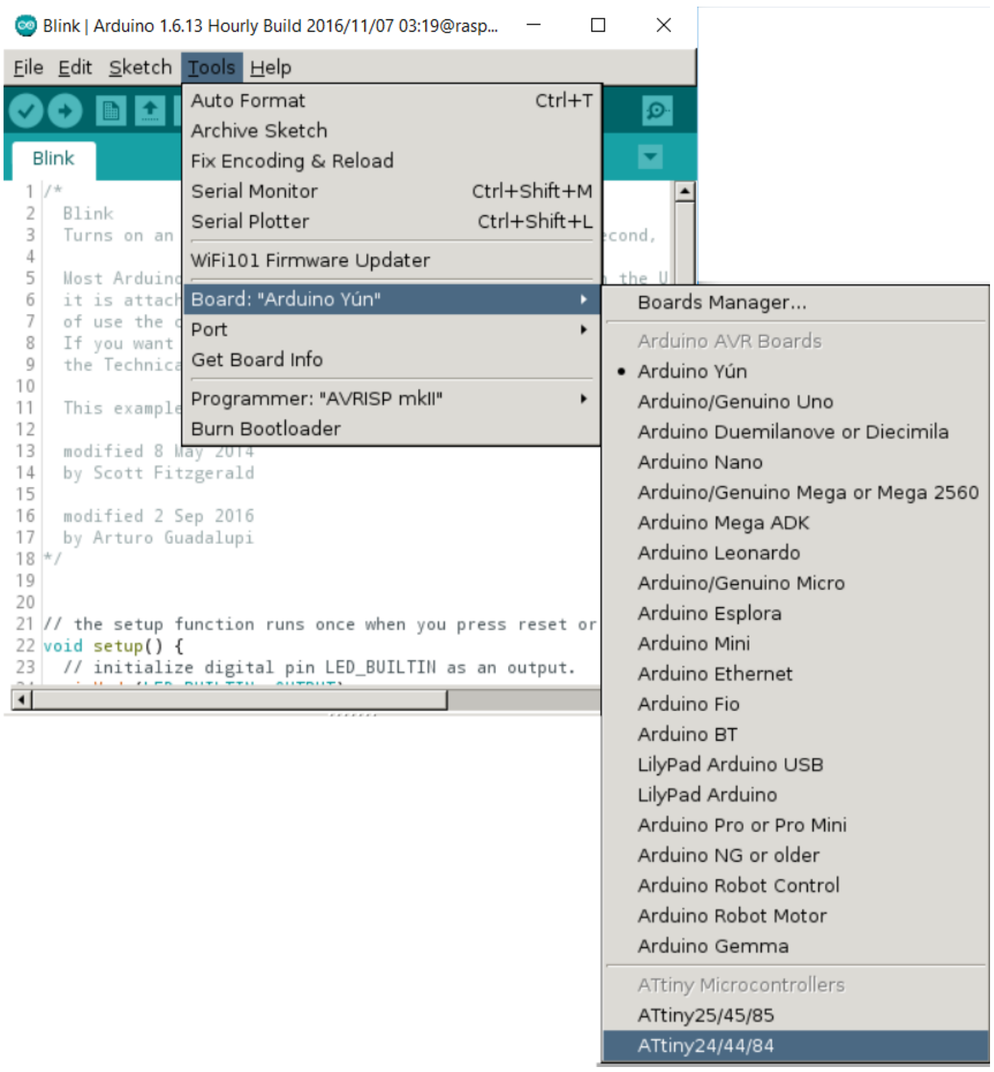


EMORO 2560 by **Inovatic-ICT**
Boards included in this package:
EMoRo 2560. Board based on ATmega 2560 MCU.

AMEL-Tech Boards by **replaced by Arrow Boards**
Boards included in this package:
SmartEverything Fox.

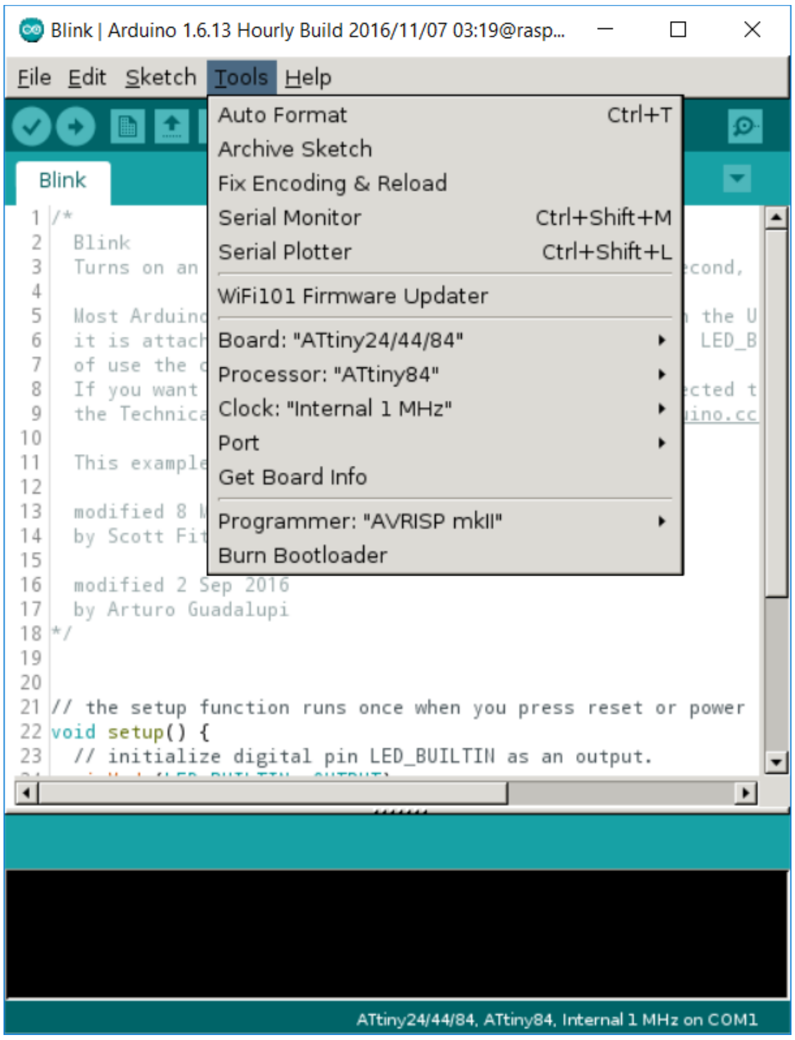
attiny by **David A. Mellis**
Boards included in this package:
ATtiny25, ATtiny45, ATtiny85, ATtiny24, ATtiny44, ATtiny84.

2.17 Go back to the Board menu item and select the ATtiny 24/44/84 entry as shown.



```
/*
  Blink
  Turns on an

  Most Arduino
  it is attach
  of use the c
  If you want
  the Technica

  This example

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
```

2.18 Return to the Board menu again and select the ATtiny84 processor (the clock should be Internal 1 MHz).

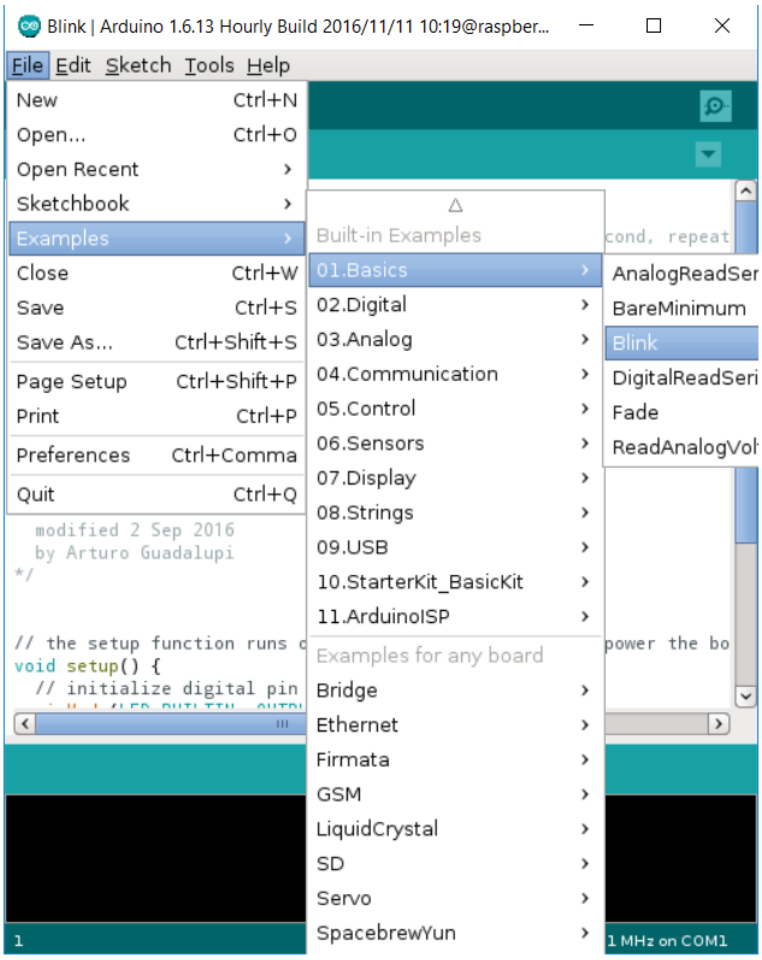


Board: "ATtiny24/44/84"
Processor: "ATtiny84"
Clock: "Internal 1 MHz"
Programmer: "AVRISP mkII"

```
/*
  Blink
  Turns on an                                                  cond,

  Most Arduino                                               the U
  it is attach                                               LED_B
  of use the c
  If you want                                                cted t
  the Technica                                               uino.cc

  This example

  modified 8 M
  by Scott Fit

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or power
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
```

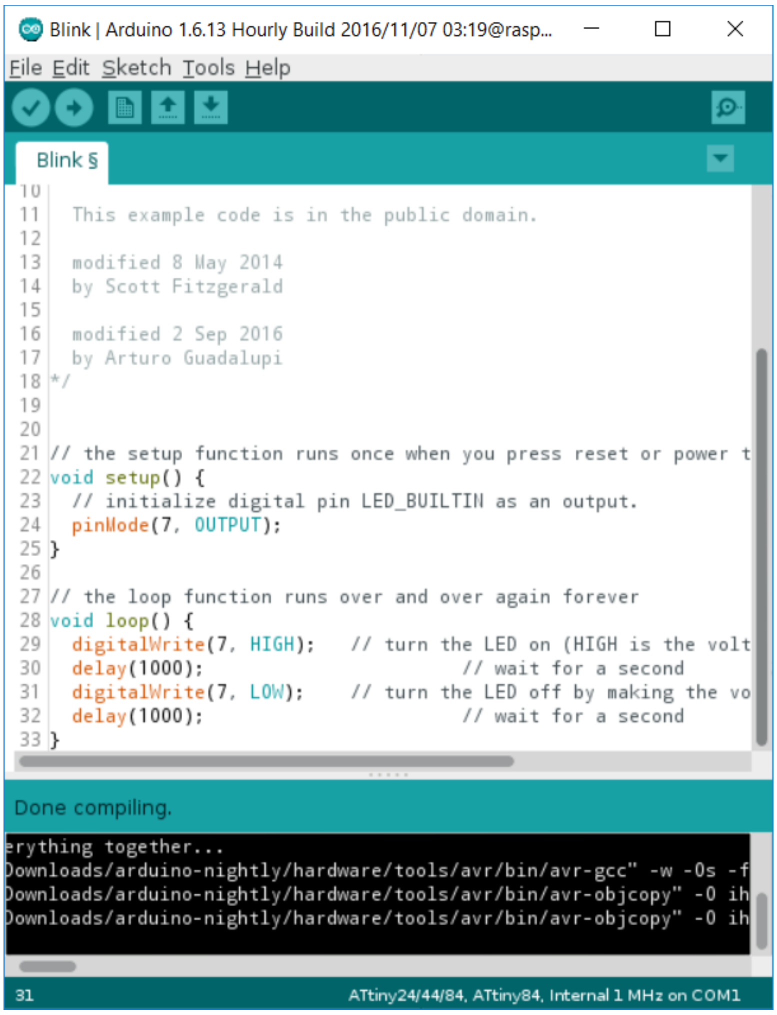
ATtiny24/44/84, ATtiny84, Internal 1 MHz on COM1

2.19 Next load up the example blink sketch by selecting File / Examples / Basics /Blink as shown.



```
  modified 2 Sep 2016
  by Arturo Guadalupi
*/

// the setup function runs o
void setup() {
  // initialize digital pin
```

2.20 Edit the sketch so that it uses output pin 7 like this: (LED_BUILTIN changes to 7 everywhere)


Blink | Arduino 1.6.13 Hourly Build 2016/11/07 03:19@rasp...
File Edit Sketch Tools Help
Blink §

```
  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or power t
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(7, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(7, HIGH);   // turn the LED on (HIGH is the volt
  delay(1000);                       // wait for a second
  digitalWrite(7, LOW);    // turn the LED off by making the vo
  delay(1000);                       // wait for a second
}
```

Done compiling.

```
erything together...
Downloads/arduino-nightly/hardware/tools/avr/bin/avr-gcc" -w -Os -f
Downloads/arduino-nightly/hardware/tools/avr/bin/avr-objcopy" -O ih
Downloads/arduino-nightly/hardware/tools/avr/bin/avr-objcopy" -O ih
```

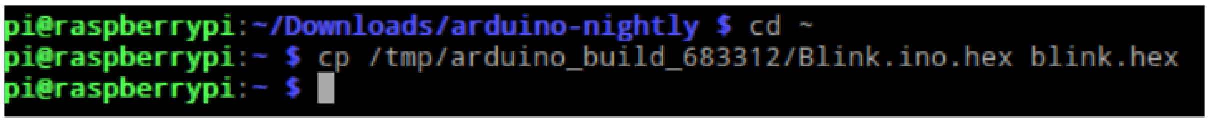
31   ATtiny24/44/84, ATtiny84, Internal 1 MHz on COM1

2.21 Click the Verify button to compile the code. Messages should appear in the area at the bottom of the screen.

2.22 Widen the window so that you can identify and copy the output path. It should be like /tmp/arduino_build_683312/Blink.ino.hex (with a different stamp number).

2.23 Copy this file for later use by entering the following commands in the Terminal window.

cd ~
cp /tmp/arduino_build_683312/Blink.ino.hex blink.hex



```
pi@raspberrypi:~/Downloads/arduino-nightly $ cd ~
pi@raspberrypi:~ $ cp /tmp/arduino_build_683312/Blink.ino.hex blink.hex
pi@raspberrypi:~ $
```

This saves it in the home/pi directory for later use. Note that the Arduino IDE must stay open while you perform this copy because Arduino clears its temporary directory entries on exit.

This completes the Arduino IDE part of the set up.

Step 3 – Install avrdude

3.1 Run the following commands from a Terminal window.

```
sudo apt-get install bison automake autoconf flex git gcc

sudo apt-get install gcc-avr binutils-avr avr-libc

git clone https://github.com/kcuzner/avrdude

cd avrdude/avrdude

./bootstrap && ./configure && sudo make install
```

(Note that during the make there will be various warnings that can be ignored)

3.2 Test the installation by entering

```
sudo avrdude -p t84 -P /dev/spidev0.0 -c linuxspi -b 10000
```

and you should see output like this:

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e930c

avrdude: safemode: Fuses OK (E:FF, H:DF, L:62)

avrdude done.  Thank you.
```

Step 5 – Programming the ATtiny84.

5.1 Run the provided script, burner84.sh. The listing for this script is as follows:

```
#!/bin/bash

# check if spi is loaded
if [ ! -c /dev/spidev0.0 ]
  then
  echo "You need to run the raspi-config program (as root) and enable the SPI module."
  exit
fi

# get name of hex file
echo "Please enter the name of the hex file to load into the chip"
read filename

# check file existence
if [ -f $filename ]
  then
  # file is found, burn it into chip
  sudo /home/pi/avrdude/avrdude/avrdude -c linuxspi -p t84 -P /dev/spidev0.0 -b 10000 -U
flash:w:$filename -U lfuse:w:0x62:m -U hfuse:w:0xDF:m -U efuse:w:0xFF:m
else
  # file not found, warn and exit
  echo "File " $filename "cannot be found"
fi
```

5.2 When requested for the filename, enter the one previously saved (see step 2.18 above)
5.3 When the script completes the LED should start flashing.

End of document.