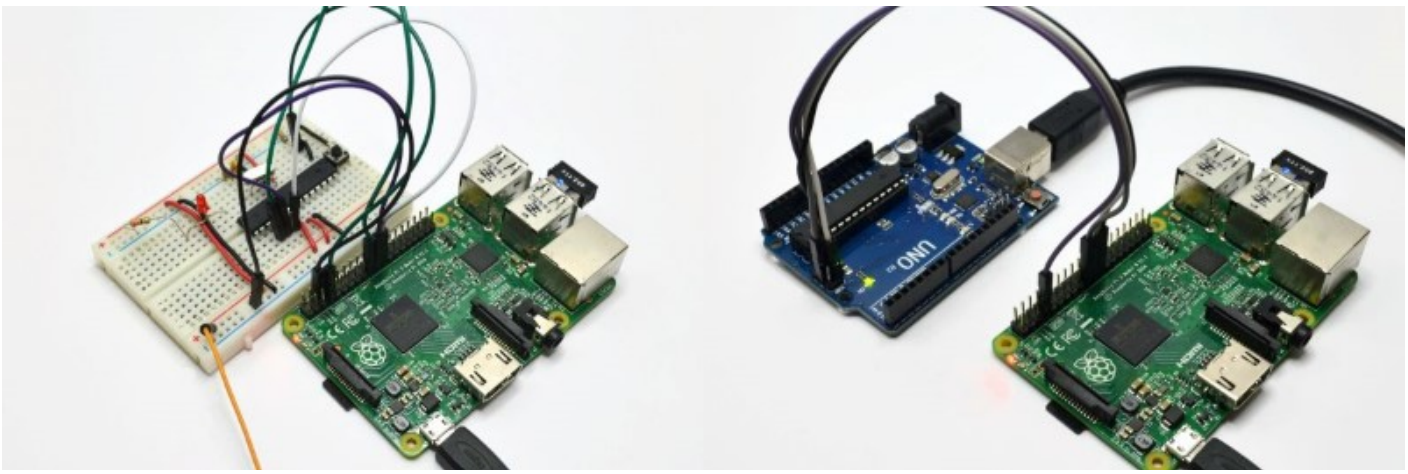


Arduino, Raspberry Pi

HOW TO PROGRAM AN AVR/ARDUINO USING THE RASPBERRY PI GPIO

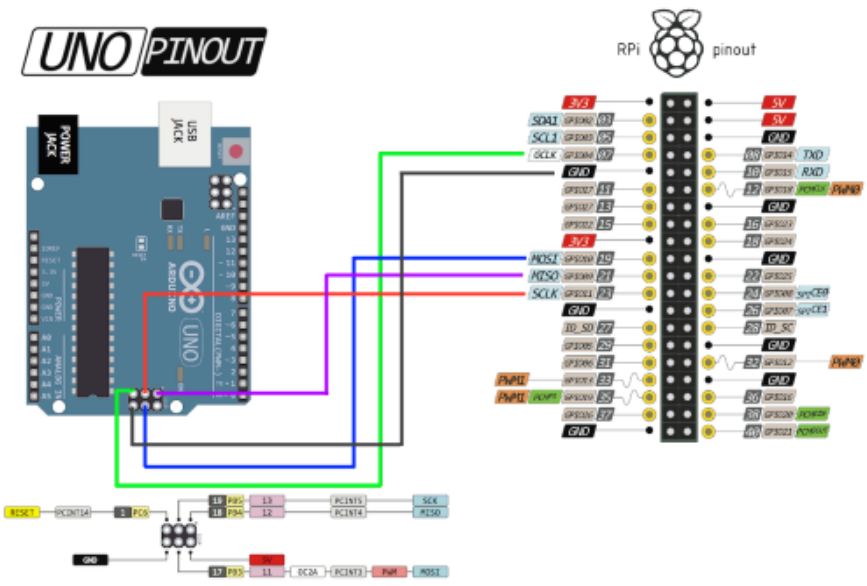
| Mark Williams | 17 Comments

In this tutorial I am going to show you how to program an AVR(ATmega328) and an Arduino UNO using the GPIO on the Raspberry Pi.

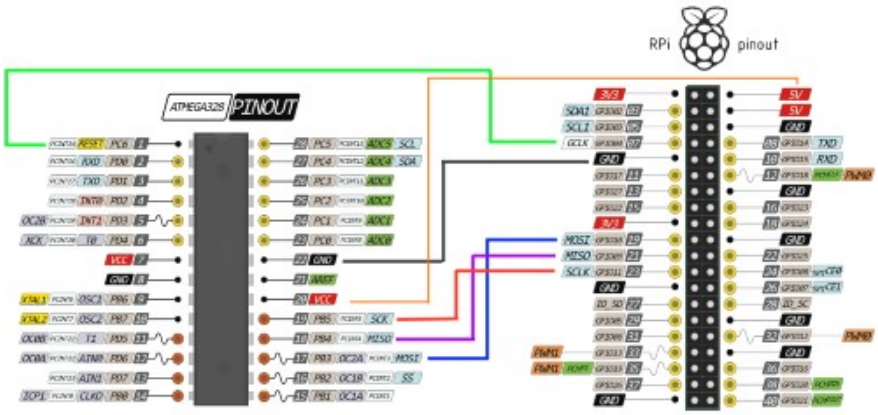
Adding an Arduino or an AVR to your projects will give you much greater flexibility.

Hook up the Raspberry Pi to the Arduino UNO or AVR.

The image below shows how to connect a Raspberry Pi 2 and an Arduino UNO. ***click the image to make it larger***



The image below shows how to hook up an ATmega328 (DIP package) to a Raspberry Pi 2. ***click the image to make it larger***



Install avrdude

Once you have your devices hooked up, it is time to install avrdude. Avrdude is an AVR programmer for Linux, which allows us to use the GPIO pins on the Raspberry Pi to program an AVR or Arduino.
First you need to install some prerequisites

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get install bison flex -y
```

Now install avrdude from source;

```
pi@raspberrypi ~ $ wget http://download.savannah.gnu.org/releases/avrdude/avrdude-6.2.tar.gz
pi@raspberrypi ~ $ tar xfv avrdude-6.2.tar.gz
pi@raspberrypi ~ $ cd avrdude-6.2/
```

Enable linuxgpio in the avrdude source, make and then install;

```
pi@raspberrypi avrdude-6.2/~ $ ./configure - -enable-linuxgpio
pi@raspberrypi avrdude-6.2/~ $ make
pi@raspberrypi avrdude-6.2/~ $ sudo make install
```

No we need to tell avrdude to use the GPIO and we need to let it know what GPIO pins to use. This can be done by editing the config file;

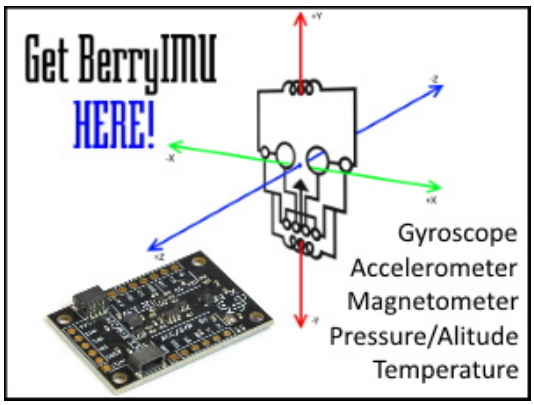
```
pi@raspberrypi avrdude-6.2/~ $ sudo nano /usr/local/etc/avrdude.conf
```

Look for “linuxgpio” and you should see this section;

```
#programmer
#  id    = "linuxgpio";
#  desc  = "Use the Linux sysfs interface to bitbang GPIO lines";
#  type  = "linuxgpio";
#  reset = ?;
#  sck   = ?;
#  mosi  = ?;
#  miso  = ?;
#;
```

And change it to this;

```
programmer
  id    = "linuxgpio";
  desc  = "Use the Linux sysfs interface to bitbang GPIO lines";
  type  = "linuxgpio";
  reset = 4;
  sck   = 11;
  mosi  = 10;
  miso  = 9;
;
```



If everything is hooked up correctly you should now be about to communicate between the Raspberry Pi and Arduino/AVR.

Time for a quick test;

```
pi@raspberrypi avrdude-6.2/~ $ sudo avrdude -c linuxgpio -p atmega328p -v
```

-c Specify the programer type
-p Part Number. Use atmega328p for the Arduino UNO
-v Verbose output
Below is from a successful test;

```
avrdude: Version 6.2, compiled on Mar  9 2016 at 11:41:53
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch
         System wide configuration file is "/usr/local/etc/avrdude.conf"
         User configuration file is "/root/.avrduderc"
         User configuration file does not exist or is not a regular file, skipping
         Using Port                    : unknown
         Using Programmer              : linuxgpio
         AVR Part                      : ATmega328P
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
         BS2                           : PC2
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :
                                  Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65    20     4    0 no       1024    4      0  3600  3600 0xff 0xff
```

```
            flash         65     6   128     0 yes      32768   128     256  4500  4500 0xff 0xff
            lfuse          0     0     0     0 no           1     0       0  4500  4500 0x00 0x00
            hfuse          0     0     0     0 no           1     0       0  4500  4500 0x00 0x00
            efuse          0     0     0     0 no           1     0       0  4500  4500 0x00 0x00
            lock           0     0     0     0 no           1     0       0  4500  4500 0x00 0x00
            calibration    0     0     0     0 no           1     0       0     0     0 0x00 0x00
            signature      0     0     0     0 no           3     0       0     0     0 0x00 0x00
          Programmer Type : linuxgpio
          Description     : Use the Linux sysfs interface to bitbang GPIO lines
          Pin assignment  : /sys/class/gpio/gpio{n}
            RESET    =  4
            SCK      =  11
            MOSI     =  10
            MISO     =  9
avrdude: AVR device initialized and ready to accept instructions
Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: safemode: hfuse reads as DE
avrdude: safemode: efuse reads as FD
avrdude: safemode: hfuse reads as DE
avrdude: safemode: efuse reads as FD
avrdude: safemode: Fuses OK (E:FD, H:DE, L:FF)
avrdude done.  Thank you.
```

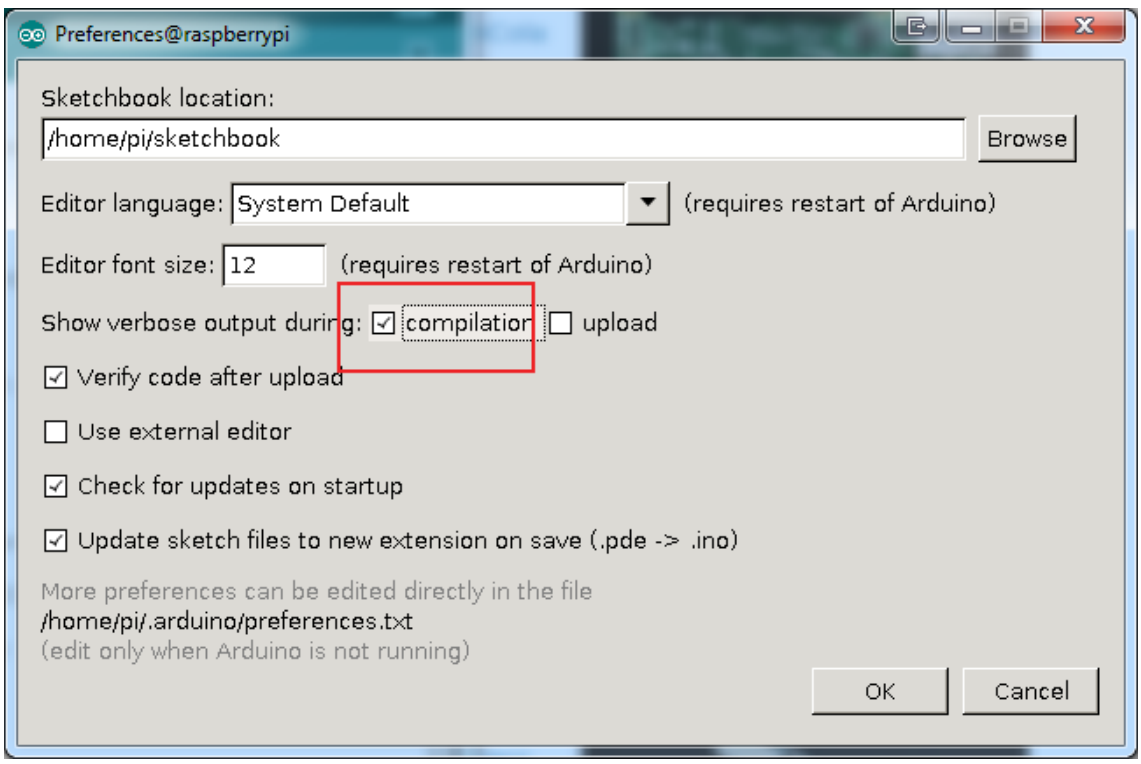
Blink Example using Arduino IDE

It's time to upload our first sketch.

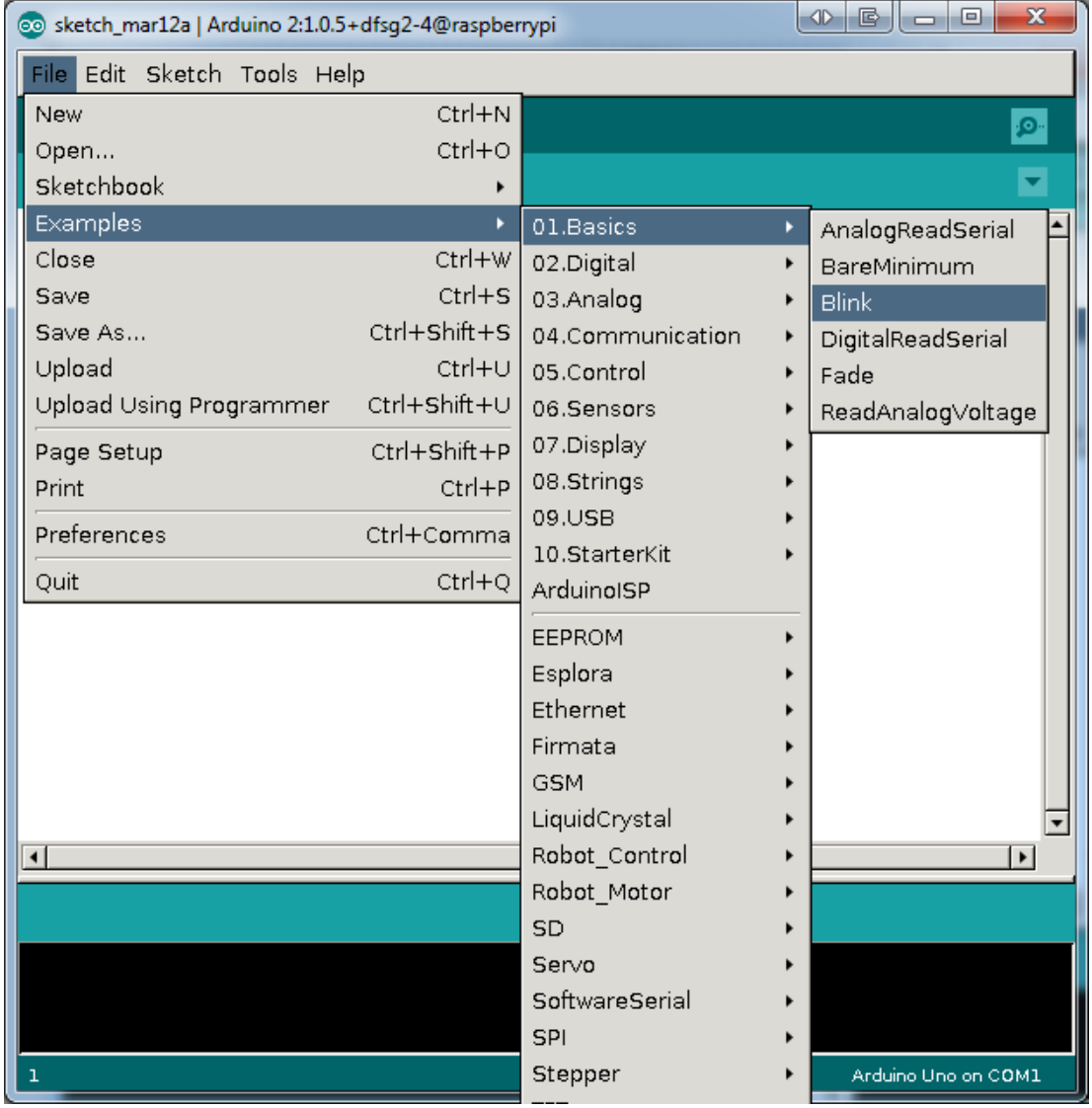
First, install the Arduino IDE;

```
pi@raspberrypi ~ $ sudo apt-get install arduino
```

Open the arduino IDE and then we need to make one change before we do anything else. Go the **File** menu and select **Preferences.** In the preference dialog box, place a tick in the 'Show verbose output during compilation'



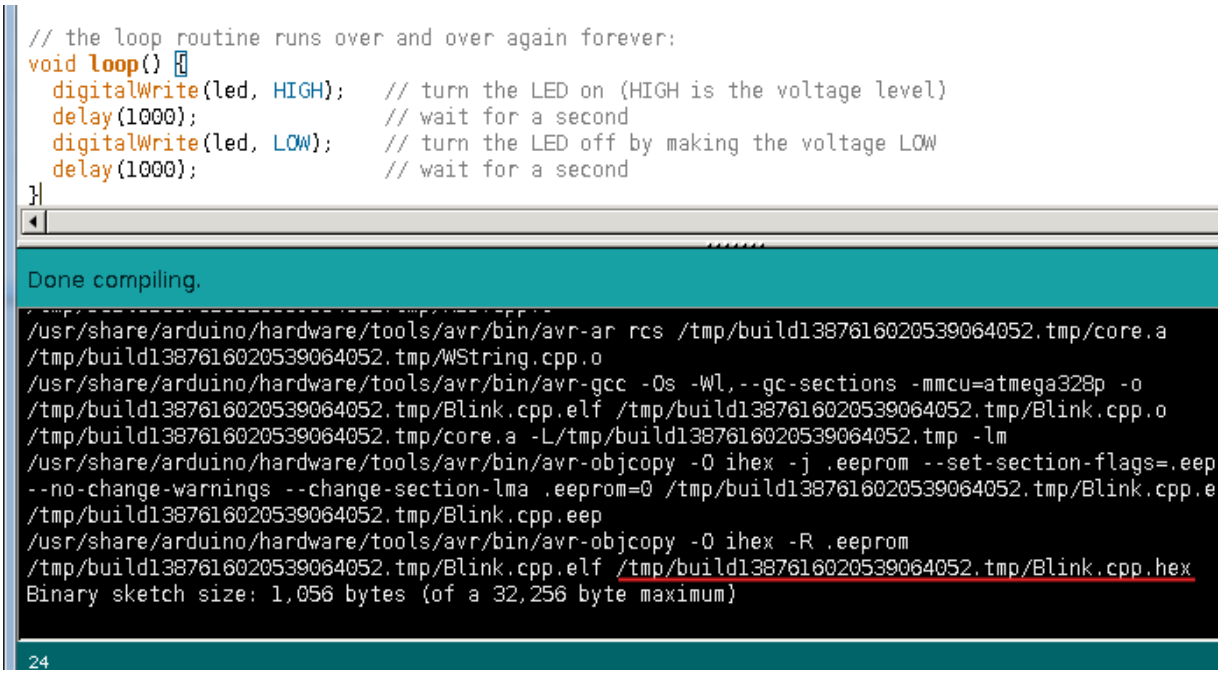
Now we can load the Blink sketch from the examples folder. The will cause the LED on pin 13 to blink.



Now compile the sketch.

As we have enabled verbose output we can now see what the IDE is doing when it compiles and where it puts the files.

The file that we need to upload to the Arduino/AVR is the **.hex** file, I have underlined this in the example below.



We will use avrdude to upload the file;

```
pi@raspberrypi ~ $ sudo avrdude -c linuxgpio -p atmega328p -v -U
flash:w:/tmp/build1387616020539064052.tmp/Blink.cpp.hex:i
```

Blink example without Arduino IDE

You can create and compile sketches without the Arduino IDE by using the Arduino Make file. This allows you to edit the code in your favorite text editor and compilation is a lot faster. This is the method I use.
You still need to install the Arduino IDE as shown above.

Install Arduino makefile

```
pi@raspberrypi ~ $ sudo apt-get install arduino-mk
```

We will now create another blink example however we will use the arduino Makefile to compile our code.

```
pi@raspberrypi ~ $ mkdir blink
pi@raspberrypi ~ $ cd blink
pi@raspberrypi blink/~ $ nano blink.ino
```

Copy in entire contents from the blink.ino example.

Now we need to create a Makefile that will reference the Arduino Makefile.

```
pi@raspberrypi blink/~ $ nano Makefile
```

And add

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

Then compile the blink.ino example

```
pi@raspberrypi blink/~ $ make
```

Make will create another folder called `build-uno` This is where it places the .hex file which we need to upload to the Arduino/AVR.
To upload this hex file;

```
pi@raspberrypi blink/~ $ sudo avrdude -c linuxgpio -p atmega328p -v -U flash:w:build-uno/blink.hex:i
```

I put both make and the avrdude commands in the one line, so it will compile the code and then upload right aftwards.

```
pi@raspberrypi blink/~ $ make && sudo avrdude -c linuxgpio -p atmega328p -v -U flash:w:build-uno/blink.hex:i
```

You will notice that it defaults to the board UNO. This can be changed in the Makefile we created. Here you would enter anything different from the defaults.
The below make file would be used for an ATmega1284p running at 20Mhz

```
MCU = atmega1284p
F_CPU=20000000L
include /usr/share/arduino/Arduino.mk
```

Read through `/usr/share/arduino/Arduino.mk` for other options that can be set. The file is very well documented.

References

Compiling Arduino sketches using Makefile
Arduino Makefile on github
Arduino Makefile on Hackaday.com
Arduino from the command line #1
Arduino from the command line #2

ARDUINO ATMEGA RASPBERRY PI

17 THOUGHTS ON "HOW TO PROGRAM AN AVR/ARDUINO USING THE RASPBERRY PI GPIO"

John
MARCH 13, 2016 AT 5:44 PM
Hi
In the second image from the top the avr does not seem to have any power connection. Perhaps I misunderstand something…

---

razorvla
JUNE 19, 2016 AT 2:16 AM
Hi,
I did this method and it is working quite well (a little boring but my arduino is broken). Do you know if there is a way to make serial monitor working ? Maybe with TX/RX pins ?

Pingback: Blink LED program with Atmega328P (Arduino Uno) programmed via Raspberry PI. – no title… yet

---

BonPi
DECEMBER 26, 2016 AT 2:54 AM
Thanks this was very helpful…I was trying to write RobotC bootloader to Arduino Mega from Raspberry Pi 2 but avrdude kept showing initialization failed, rc=-1 error. After several hours of Googling, I found that I needed to use BCM pin numbers instead of Physical pin numbers in avrdude config file which are (4, 11, 10, 9). You sample config above was helpful in figuring it out.

Leo Bakker
MAY 1, 2017 AT 4:27 PM
Great write up. Thanks !
Is GND connected to 5v in the hookup image with the UNO ?

★ Mark Williams

MAY 2, 2017 AT 8:54 PM

Well spotted. I have corrected the hookup image

vangelis

DECEMBER 11, 2017 AT 6:36 AM

EXCELLENT FLAWLESS GUIDE!!! thank you for taking time to st up this simple tutorial)))

Peter

JANUARY 4, 2018 AT 12:06 AM

Great guide, got it working in almost no time.

I didn't use avrdude-6.2 as suggested in this guide. It would probably have worked though.

Instead I went directly with the latest, 6.3, which doesn't work, some bug about exporting GPIO 4.

Using the avrdude package (6.1) worked flawlessly.
> apt install avrdude

Oliver

MARCH 20, 2021 AT 1:44 AM

look at:
http://fab.academany.org/2018/labs/fablaboulu/students/mikko-toivonen/Assignments/raspiconfig.html

you have to modify /avrdude-6.3/linuxgpion.c:
by replacing "%ud" with "%u"

good work

SB

APRIL 11, 2018 AT 1:49 PM

I have had this working for a while now but have implemented some extra features on my design to allow gpios to be connected between Pi and AVR. Now when I program my AVR I sometimes get a reboot on the pi. I'm thinking the AVR programming cycle does something to the Pi IO to cause a reboot. Any thoughts ?

Kim SJ

JULY 31, 2018 AT 7:22 PM

You have not worried about the fact that the UNO/ ATmega328 are 5V devices and the Pi is a 3v3. Connecting outputs of the UNO directly to the Pi is likely to damage the Pi. You need a voltage divider:

Uno Out —
|
1k8 resistor
|
— Pi in
|
3k3 resistor

|
GND

The only signal with this problem in your circuit is MISO.
You should also ground SS, to avoid it floating in the breeze.

See https://learn.sparkfun.com/tutorials/voltage-dividers/all for more info.

Richard

JUNE 2, 2019 AT 8:07 PM

Is it not possible to select the newly created programmer that you created with the avrdude. config from the IDE and upload the sketch directly?

bahadir

DECEMBER 26, 2018 AT 7:39 AM

Thanks for the great guide, additionally,
I use the lines in my Makefile for Arduino Pro Mini 328 – 5V/16MHz without any problems:

ARDUINO_DIR = /usr/share/arduino
ARDUINO_PORT = /dev/ttyACM*

USER_LIB_PATH = /home/pi/sketchbook/libraries
BOARD_TAG = pro5v328

include /usr/share/arduino/Arduino.mk

More info in this tutorial: https://youtu.be/qAM2S27FWAI

As Kim SJ says Voltage Dividers is an important thing to be concerned about though.

Peter

APRIL 18, 2019 AT 5:42 PM

Hi, I've got your setup working but I have one question about the choice of pins used.

The pins you use overlap the SPI ports.
Is this a requirement, does the programmer use the SPI functionality?

or

Would this work using any GPIO ports?

Shmolik

AUGUST 21, 2019 AT 10:03 AM

Thanks men , i use 6.3 and i get some but with port 4 .
in 6.2 its work perfect !

David

OCTOBER 16, 2019 AT 12:03 AM

Hello, this tutorial works fine for me. Now I can program a atmega328p without a Arduino board. But now I want to debug mi code. I read about GBD but I don't now if there is a way to debug in real time. Can you give some thread?

Merna Saleh

OCTOBER 11, 2020 AT 5:42 AM

which tool you had used to make the photo show connections of Arduino and raspberry pi