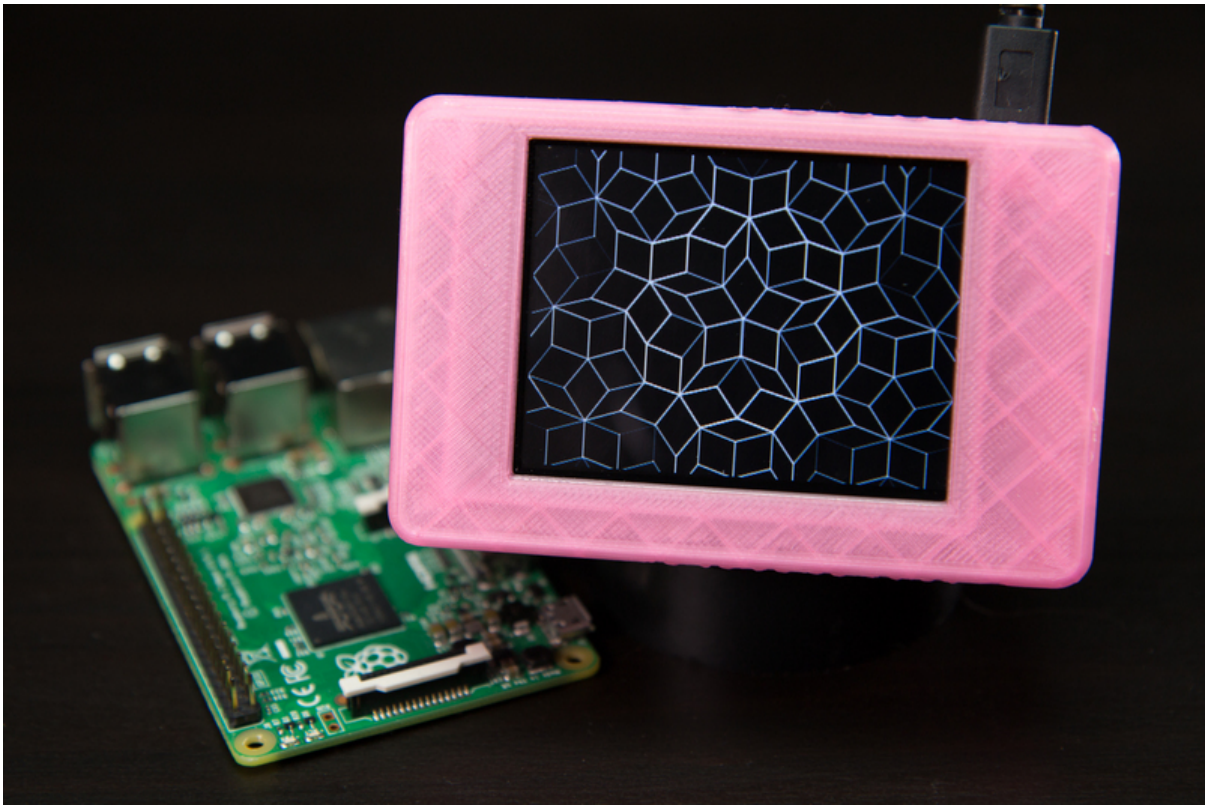
Processing on the Raspberry Pi & PiTFT

Created by Tony DiCola



https://learn.adafruit.com/processing-on-the-raspberry-pi-and-pitft

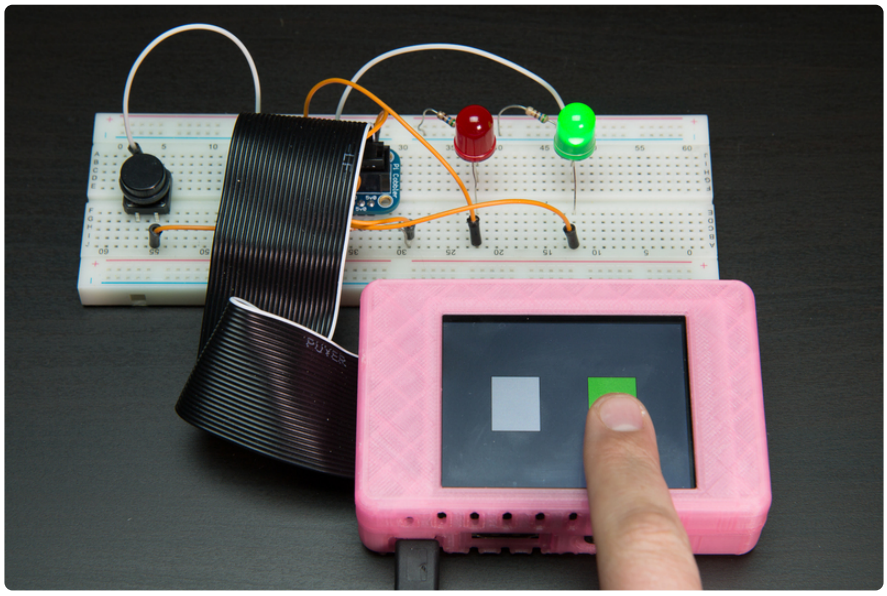
Last updated on 2024-06-03 01:49:06 PM EDT

Table of Contents

Overview

This project will show you how to install and use the Processing programming environment (https://adafru.it/ddm) on a Raspberry Pi and PiTFT. Processing is a fantastic tool to create visual programs, and its latest Processing 3.0 release now has support for the Raspberry Pi. With Processing you can create beautiful artwork, prototype interfaces, and much more without having to be an expert programmer. In fact Processing is targeted at artists, students, makers, and anyone who's new to programming.

One cool new feature in Processing 3.0 is support for controlling hardware. This allows you to add interesting new interactivity to your Processing sketches and even prototype hardware interfaces with ease. On a Raspberry Pi you can control the GPIO (general purpose input/output) pins and do things like blink LEDs, read buttons and much more. For example you can create a sketch that turns on or off LEDs by touching a block on the screen:



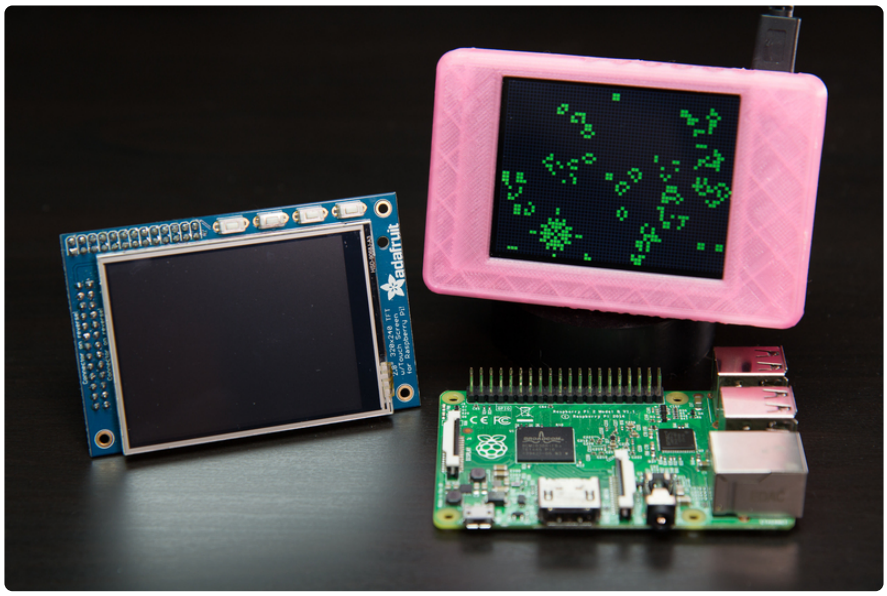
One thing to keep in mind with Processing on the Raspberry Pi is that the Pi hardware is not as fast as a desktop or laptop computer. This means complex Processing sketches with fast animations or processor intensive calculations might not work on the Pi. Keep your sketches simple, like drawing 2D shapes or creating basic user interfaces.

To follow this guide you'll want to somewhat familiar with Processing. Check out the great tutorials on getting started with Processing (https://adafru.it/jrD) if you're completely new to using it. It will also help to check out this introduction to Processing 3.0 (https://adafru.it/jrE) so you can learn about its new features.

You'll also want be familiar with using the Raspberry Pi, like how to load an operating system on it and connect to its command line terminal. Check out this great series of Raspberry Pi learn guides (https://adafru.it/jcW) if you're new to using it.

Continue on to learn about the hardware used in this project.

Hardware



Parts

To follow this guide you'll need a Raspberry Pi (https://adafru.it/dlv). Either the original model (like a Raspberry Pi B+ (http://adafru.it/1914), B (http://adafru.it/998), A+ (http://adafru.it/2266), or A (http://adafru.it/1344)) or the faster Raspberry Pi 2 (http://adafru.it/2358) will work, however I highly recommend using the Raspberry Pi 2. The Pi 2 has more memory, runs a little faster than the original Pi and has multiple CPU cores so it can do other things while running a Processing sketch. However even the Pi 2 is still a somewhat slow computer compared to modern desktops & laptops--complex Processing sketches won't work well on the Pi!

In addition to the Raspberry Pi you'll need a micro SD card (http://adafru.it/1294) that's at least 8 gigabytes in size, and a power supply for the Pi (a 5 volt 2 amp supply (http://adafru.it/1995) is recommended).

You can also use a PiTFT display (https://adafru.it/jrF) to show a Processing sketch. Any of the PiTFT displays will work and you can find them in sizes from 2.2 inches up to 3.5 inches.

If you need a bigger display you can use the Pi Foundation's 7" TFT display (http://adafru.it/2718), a HDMI display (https://adafru.it/jsb) or DPI display (using the DPI TFT kippah (http://adafru.it/2454)), or just plug in a computer or television to the Pi's HDMI output.

If you'd like to control hardware GPIO pins on the Pi from Processing you might want a Pi Cobbler GPIO breakout cable (http://adafru.it/2029) and a few switches, LEDs, and resistors to experiment with. All of these parts are available in this handy Raspberry Pi 2 starter kit (http://adafru.it/2380) (which you can also get without the Pi 2 (http://adafru.it/2126) if you already have one).

Don't forget you can put the Pi in a nice case (https://adafru.it/dV5) to keep everything neat & tidy. Or you could 3D print a fancy case for a 3.5" PiTFT (https://adafru.it/jsc) (shown in the photos for this project), Pi Foundation TFT (https://adafru.it/jsd), and more!

Raspberry Pi Setup

To use Processing on the Pi you'll want to use the latest Raspbian operating system. Currently there are two version of Raspbian, an older version called Wheezy (based on Debian Wheezy) and the latest version called Jessie (based on Debian Jessie). I recommend using the newest Jessie release, but you can still use the older Wheezy release if necessary. Grab the official Raspbian release from this page (https://adafru.it/fQi) and burn it to a SD card and install on the Pi (https://adafru.it/jd0).

If you're using a PiTFT be sure to instead follow the directions on the appropriate PiTFT guide to assemble and setup the Pi. I highly recommend using the easy install image from the guide to get started easily:

- PiTFT 3.5" Guide (https://adafru.it/jse)
- PiTFT 3.2" Guide (https://adafru.it/dDK)
- PiTFT 2.8" Capacitive Touch Guide (https://adafru.it/jsf)
- PiTFT 2.8" Resistive Touch Guide (https://adafru.it/dDK)
- PiTFT 2.4" Guide (https://adafru.it/jsA)
- PiTFT 2.2" Guide (https://adafru.it/jsB)

You'll also want to make sure your Pi has access to the internet from either a wired or wireless networking adapter. See this guide on setting up wired and wireless networking on a Raspberry Pi (https://adafru.it/jd2) if you need help.

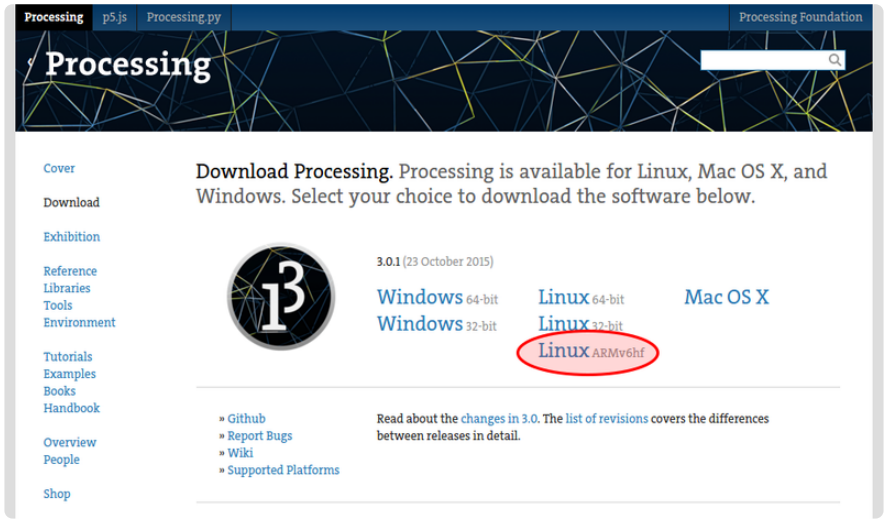
Once your Pi is setup and connected to the internet then continue on to learn how to install Processing on the Pi.

Processing

Installation

With the latest Processing version 3.0 there are premade binaries you can download and easily install. There's no need to install Java or other dependencies as they're now included in Processing.

From the Processing download page (https://adafru.it/jsC) pick the **Linux ARMv6hf** release (highlighted below):



(if you're curious the ARMv6hf name stands for the ARMv6 architecture with a hardware floating point unit, i.e. the architecture of the Raspberry Pi)

Once downloaded copy the .tgz to the Pi. You can actually download the file directly to the Pi by connecting to the Pi's command line terminal and running:

```
cd ~
wget http://download.processing.org/processing-3.0.1-linux-armv6hf.tgz
```

This will download the 3.0.1 release to the home directory for the Pi user. Note that version 3.0.1 is currently the latest version of Processing, but be sure to check the Processing website (https://adafru.it/ddm) to see if a later version is available and download it instead of 3.0.1.

Once the file is on the Pi, run the following command to extract it:

```
tar xvfz processing-3.0.1-linux-armv6hf.tgz
```

This will extract Processing to a processing-3.0.1 subdirectory. That's all you need to do to install Processing on the Pi!

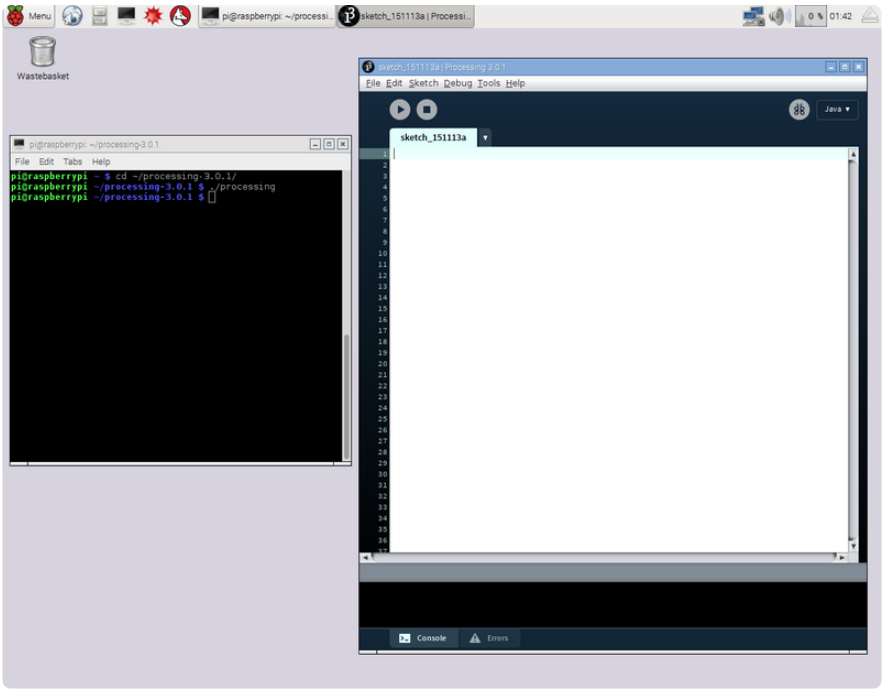
Usage

To run processing you'll need to use the Pi's graphical desktop environment. The easiest way to run the Pi's desktop is to enable the Pi to boot directly to the desktop instead of a text console. Use the raspi-config tool and enable the **Boot to Desktop mode** (https://adafru.it/jsD) (specifically the boot to desktop and autologin as Pi user option on newer Raspbian versions), then reboot the Pi. You should see the Pi's graphical desktop environment displayed on the Pi's HDMI output.

Inside the graphical environment open the **Terminal** application and run the following commands to change to the directory Processing was downloaded (in the previous section) and run it:

```
cd ~/processing-3.0.1/
./processing
```

Processing's splash screen should pop up, then after a few moments the IDE will load:



Now you can use Processing just like using it on your computer. Type in or load a sketch, click the run button, and watch your sketch execute on the Raspberry Pi.

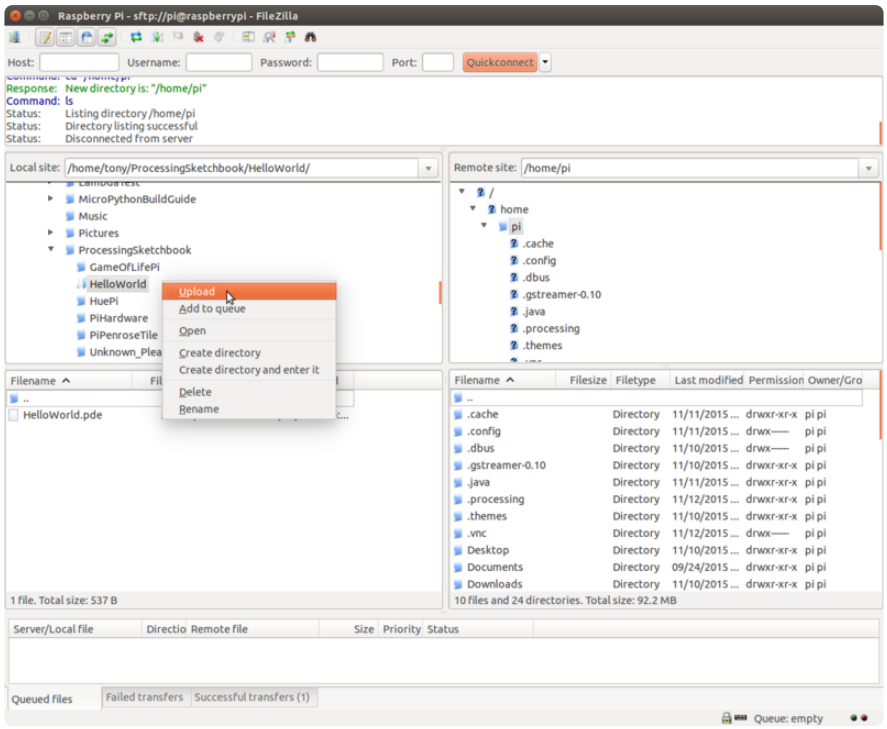
Running a Sketch Without the Editor

In the previous section you saw how to run Processing and use its editor to create and run a sketch. However you might find it more useful to run a sketch directly on the Pi without using the editor. This is great for small displays like the PiTFT where using the Processing editor isn't easy. You can also use this to run a sketch from a command terminal outside the graphic environment (like when connected to the Pi using SSH (https://adafru.it/jsE)).

First make sure the graphical desktop environment is running. Even though you aren't logging in and running commands in the desktop it still needs to run to show the Processing sketch. Remember you can have the desktop automatically start on boot using the raspi-config command in the previous section (be sure to set the pi to boot and automatically log in).

Next copy your Processing sketch code to the Pi. Remember a Processing sketch includes both a .pde file with the code and the directory that contains the .pde file. For example a Processing sketch called HelloWorld would have a directory called HelloWorld and a file inside that directory called HelloWorld.pde. Copy that directory and the files inside it to the Pi.

You can use a tool like FileZilla to connect to the Pi using SFTP and copy over files from your computer (https://adafru.it/jsF). For example here's a picture of uploading a sketch called HelloWorld from my ProcessingSketchbook folder on my computer to the /home/pi folder on the Pi:



Once the sketch is copied over connect to a command line terminal on the Pi. Run the following command to change to the directory that Processing was installed and use its special processing-java command to run a sketch outside the editor:

```
cd ~/processing-3.0.1/
DISPLAY=:0 ./processing-java --sketch=/home/pi/HelloWorld --present
```

There are a few important parts of the command above:

- **DISPLAY=:0** is setting the DISPLAY environment variable to the value :0 which is the address of the graphical desktop. Programs run inside the graphical desktop will already see this value set, but when you're not running inside the desktop you need to explicitly set the variable like this.
- **./processing-java** is calling the processing-java executable. This is a special command line version of Processing (https://adafru.it/jta) that can run a sketch without using the editor.
- **--sketch=/home/pi/HelloWorld** is a parameter that points to the location of the sketch to run. In this case a HelloWorld sketch that was copied to the /home/pi location on the Pi. You can run a different sketch by changing the location of this parameter.
- **--present** is a parameter that tells Processing to run the sketch and display it on the screen. This needs to be the latest parameter in the call.

Note that it will take a little bit of time for the sketch to start running. To stop the sketch press **Ctrl-c** in the terminal that started it.

Run Sketch Fullscreen

To run a sketch in fullscreen you can use the new fullScreen function (https://adafru.it/jtb) in Processing 3.0. This is especially useful for running a sketch on a small display like the PiTFT. Just replace the call to size() (https://adafru.it/jtc) in the setup function with a call to fullScreen() (https://adafru.it/jtb).

In addition you might want to add a call to the noCursor() (https://adafru.it/jtd) function in the setup to hide the mouse cursor.

PiTFT Limitation

If you're using the PiTFT to display a sketch one thing to be aware of is that Processing's 3D-accelerated renderers are **not** directly supported. Specifically the P2D and P3D renders won't work with the PiTFT. This is because the Pi's graphics processor outputs to the HDMI port and not the PiTFT.

If you're an advanced user you can potentially look at using the fbcp tool to copy the output of the Pi's HDMI port to the PiTFT. See this guide for using fbcp to play 3D accelerated application on the Pi (https://adafru.it/jte) for information on using fbcp. This is a fairly advanced procedure and requires compiling and installing fbcp from its source.

Another option to see 3D-accelerated graphics is to use a small HDMI monitor or television. Remember the Raspberry Pi is not as fast as your desktop computer or laptop so it won't be able to display complex animations or graphics!

Disable Screen Blanking

If you run a Processing sketch for a while without any input you might find the monitor turns off and goes blank. You can prevent this behavior by editing a desktop environment configuration file.

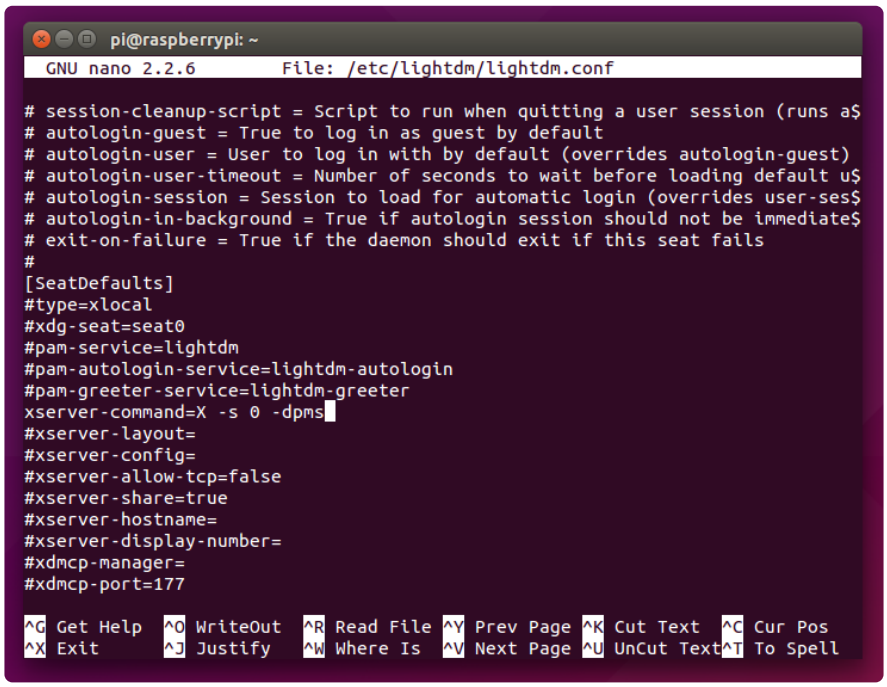
In a command terminal run this command to edit the /etc/lightdm/lightdm.conf configuration file:

```
sudo nano /etc/lightdm/lightdm.conf
```

Then scroll down to the [**SeatDefaults**] line and change the **#xserver-command=X** line beneath it to look like:

```
xserver-command=X -s 0 -dpms
```

The file should look like the following image:

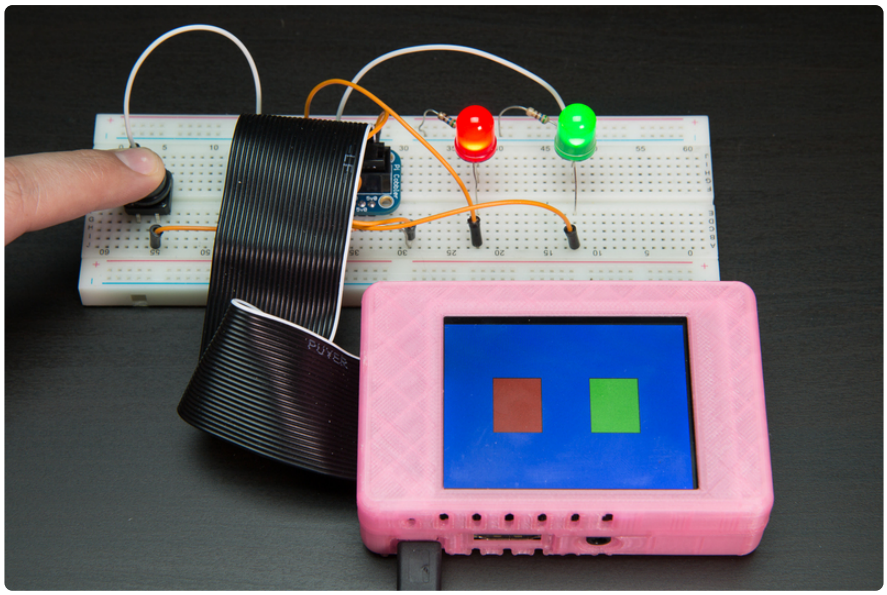


Then save the file by pressing **Ctrl-o** then **enter**, and then **Ctrl-x**. Reboot the Pi and the change should take affect and prevent the monitor or PiTFT from turning off.

Hardware IO

One major new feature in Processing 3.0 is hardware IO (input/output) access on the Raspberry Pi. This means you can control the GPIO (general purpose input/output) pins on the Pi and use them to blink LEDs, read buttons, and more. Combined with the graphic and GUI functionality already in Processing you can prototype and create interesting hardware projects entirely with Processing!

As an example I'll walk through a simple Processing sketch to light LEDs and read a button. The example will display two boxes on the screen. If you click/press inside a box it will light up a LED and change the box color. If you press the button it will change the screen color to blue:

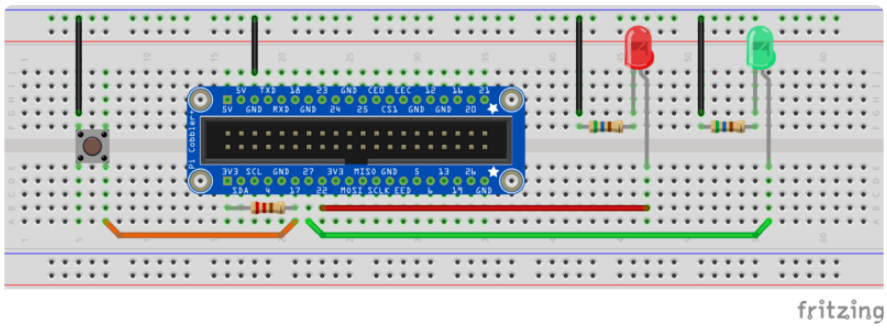


To build the hardware for this sketch you'll need these parts (all of these parts are available in the Raspberry Pi starter kit (http://adafru.it/2380)):

- Breadboard and breadboard jumper wires.
- Red LED.
- Green LED.
- Two 560 ohm 1/4 watt resistors - You can use any resistors in the range of about 300 to 1,000 ohms, but you **must** have them or else you could damage the Pi!
- A momentary push button.
- A 10 kilo-ohm 1/4 watt resistor - This is used as an external 'pull-up' resistor to help read the button. Although the Pi actually supports internal pull-up resistors they aren't available from Processing right now. However you can add your own pull-up like this example will show.
- Although it's not technically required a Pi cobbler (http://adafru.it/2029) will make it much easier to connect components to the Raspberry Pi GPIO pins.

If you're new to using a breadboard check out this great description of how to use them in an Arduino getting started project (https://adafru.it/jtf).

Connect the components on the breadboard in the following way:



- The **top left corner of the button** is connected to the **Pi GND pin**, and the **bottom right corner of the button** is connected to **Pi pin #17**.
- One leg of the **10 kilo-ohm resistor** is connected to **Pi pin #17** and the other leg is connected to the **3.3V pin** on the Pi. This will act as a 'pull-up' that keeps the button input at a high level until the button is pressed and 'pulls' it down to ground.
- The **anode (longer leg) of the red LED** is connected to **Pi pin #22**, and the **cathode (shorter leg) of the red LED** is connected to one leg of a **560 ohm resistor**. The other leg of the resistor is connected to the **Pi GND pin**.
- The **anode (longer leg) of the green LED** is connected to **Pi pin #27**, and the **cathode (shorter leg) of the green LED** is connected to one leg of a **560 ohm resistor**. The other leg of the resistor is connected to the **Pi GND pin**.

Next download the Processing_Pi_GPIO sketch from its home on GitHub (https://adafru.it/EMN) by clicking the button below and extracting the archive:

Download Processing_Pi_GPIO Sketch

https://adafru.it/EMN

Copy this sketch to the Raspberry Pi using a tool like FileZilla (described in the previous page).

Now you can run the sketch using the steps to run outside the editor described in the previous page. Make sure the Pi has booted to the graphical desktop environment, then connect to the Pi's command line terminal.

If you're using the **Raspbian Jessie** operating system which allows access to GPIO as the Pi user run the commands:

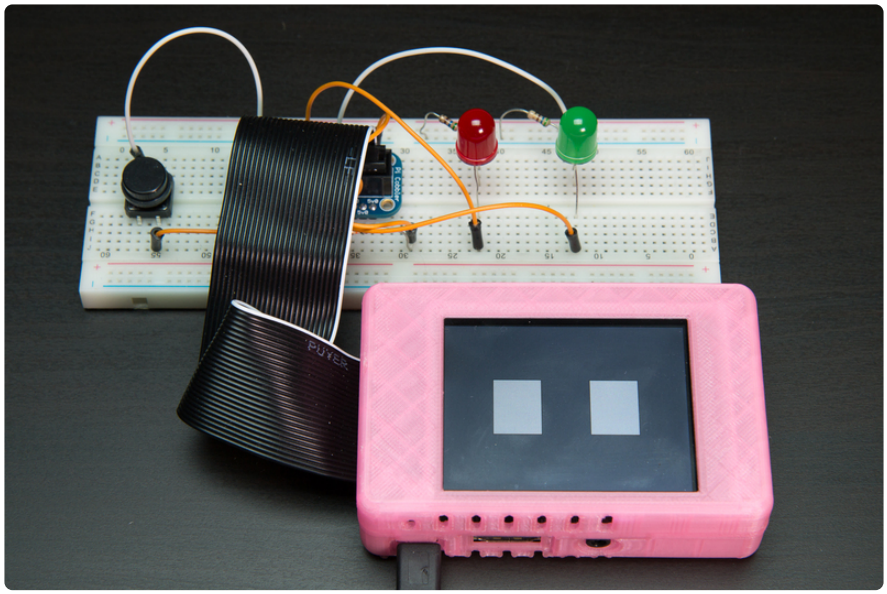
```
cd ~/processing-3.0.1/
DISPLAY=:0 ./processing-java --sketch=/home/pi/Processing_Pi_GPIO --present
```

However if you're using the older **Raspbian Wheezy** operating system which requires a root user to access GPIO you need to run the slightly more complex commands:

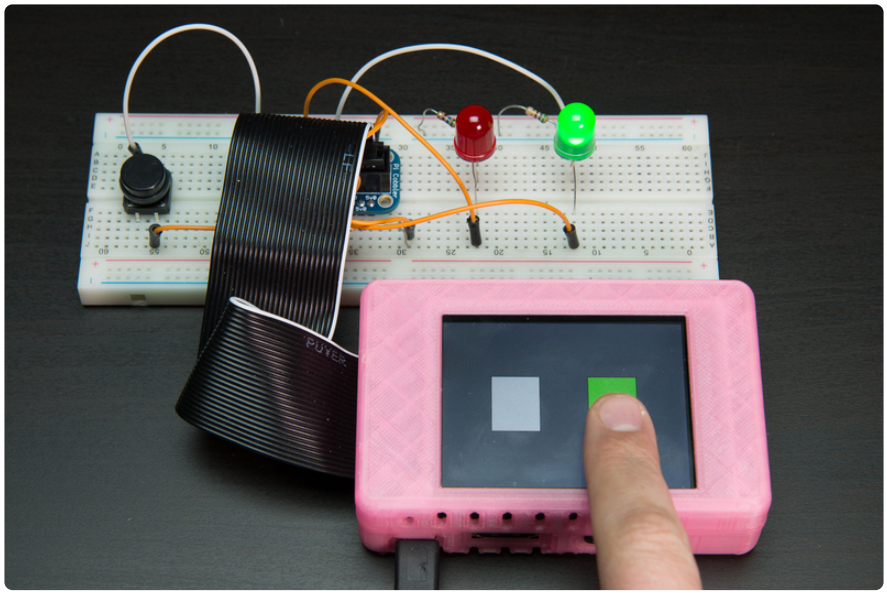
```
cd ~/processing-3.0.1/
DISPLAY=:0 XAUTHORITY=/home/pi/.Xauthority sudo ./processing-java --sketch=/home/pi/
Processing_Pi_GPIO --present
```

This command will run Processing's command line tool as the root user so it can access GPIO, however it also sets a special XAUTHORITY environment variable that's needed to use the desktop environment as root.

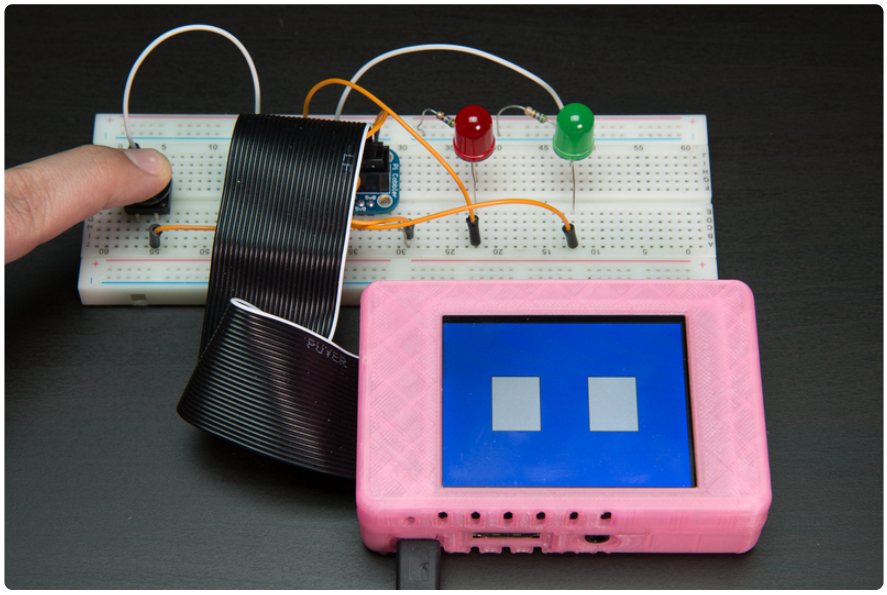
Once the sketch is running it will show two boxes on the screen:



Press a box to turn on or off the LED (the left box controls the red LED and the right box controls the green LED):



And press the momentary button to change the screen color to blue while it's pressed:



If you're curious how the hardware IO access works you can open the sketch in Processing's editor and examine the code (https://adafru.it/jtC). The Processing hardware IO reference (https://adafru.it/jtD) is a great resource to follow to understand the code too.

The sketch uses the following hardware IO functions to control the LEDs and read the button. If you've ever used an Arduino these functions should look very familiar:

- **GPIO.pinMode** - This function is called in the setup of the sketch and configures a pin as either an input or output. The LEDs are outputs and the button is an input. Notice too that each pin is identified by its number on the Pi cobbler (these are the Pi's BCM pin numbers if you're familiar with the Pi's pinout (https://adafru.it/jtE)).
- **GPIO.digitalWrite** - This function will set an output pin to a high (on) or low (off) value. When the LED pins are set to a high value they will turn on, and when set to a low value will turn off.
- **GPIO.digitalRead** - This function will read the value of an input pin. It returns either a high or low value depending on if the pin is at a high (on) value or low (off) value.

That's all there is to controlling the GPIO pins on the Raspberry Pi with Processing!