# With a Game Controller and Arduino

By **Ian Buckley** / March 1, 2018 01-03-2018 / 8 minutes

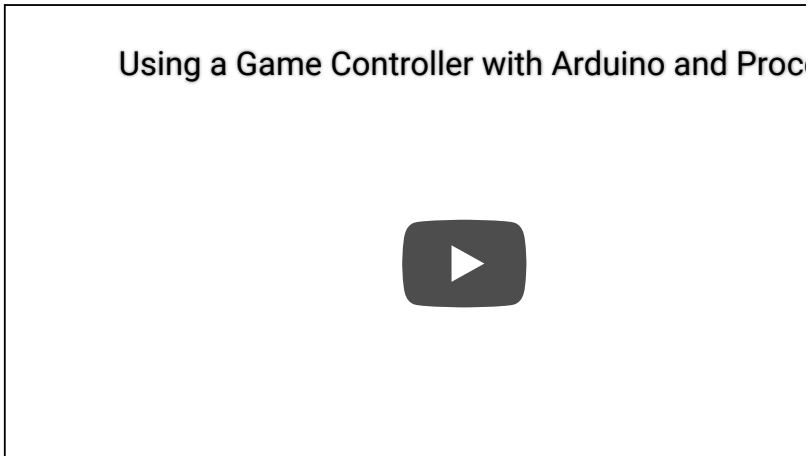
Ian Buckley
**107** articles

Ian Buckley started out with a degree in Music composition, before devoting his time to DIY tech and coding. He now works as freelance journalist, performer an video producer living in Berlin, Germany. When he's not writing o on stage, he's tinkering with DIY electronics or code in the hope of becoming a mad scientist.



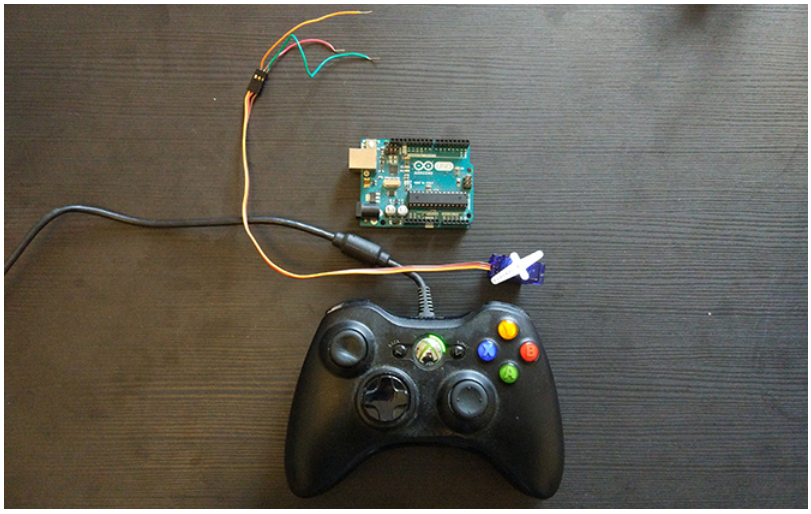
Arduinos and **similar compatible boards** are one of the go to devices for DIY tinkerers everywhere. Whether you are a beginner who is **just getting started with Arduino** or someone who has already been putting them to use in your life, they provide a platform for countless awesome projects.

Today we will be exploring a creative way to control a servo using Processing, and an Xbox360 controller. If you are already pretty game development savvy, you might be interested in our **Custom Game Controller tutorial**, which uses Unity.




This tutorial will assume a little prior knowledge, if this is your first foray into Arduino fiddling, you might find our **Arduino guide** useful here. Similarly, if this is your first time using Java it can be a little confusing. While Processing

## What You Need



1 x Arduino. We are using an UNO today.

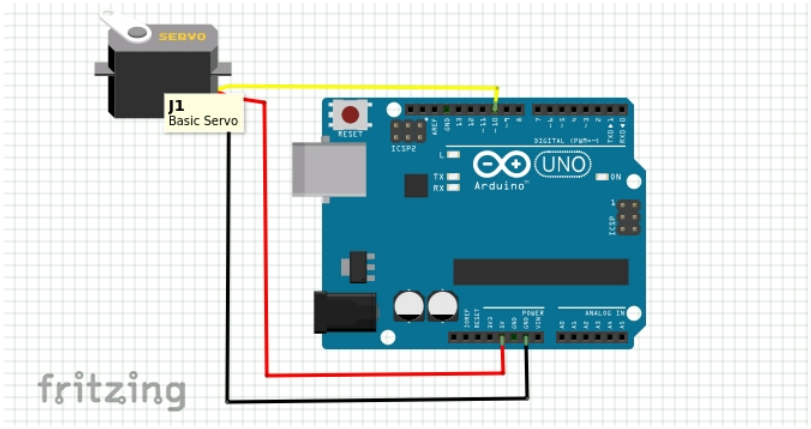
1 x hobby servo. Anything that will work with the Arduino pins.

1 x wired Xbox360 controller. Although this will technically work with almost any controller.

Several hookup wires.

In addition to these things, you'll also need to download **Processing** and the **Arduino IDE** from their respective websites.

## Preparing the Arduino




PRODUCT REVIEWS , SMART HOME
Monoprice Strata H The Best Budget Rol Vacuum?

DIY
8 Fun Rugged Raspb Pi Projects to Build the Outdoors

MAC , ENTERTAINMENT
6 Tips to Improve Yc Mac Gaming Experie

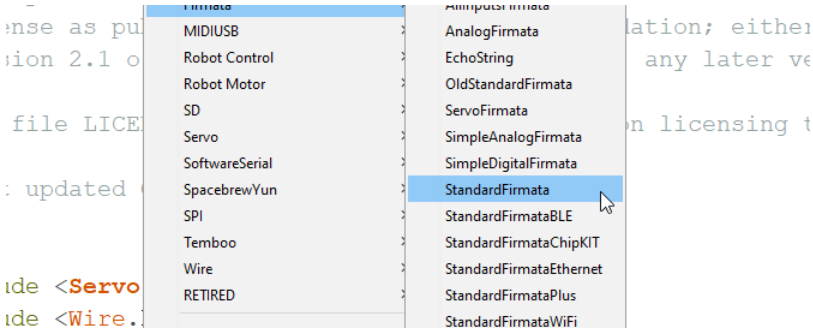
## Latest Videos


Geeks Try Drawing Tech Logo TGTS S2E07


How to Send Emails from an E Spreadsheet Using VBA Scrip

and **brown** or **black** attach to the **GND** pin. The data line, which is usually **yellow** or **orange**, attaches to **pin 10**.

Check your wiring and connect the Arduino to the computer. Open up the Arduino IDE.




Open up the StandardFirmata sketch located at **File > Examples > Firmata > StandardFirmata**. This sketch sets up the board for external control over the serial port, and is the same one we used in our article on **controlling Arduino with Python**. Upload the sketch to the board.

### How to Program and Control an Arduino With Python

Sadly, it is impossible to directly program an Arduino in Python, but you can control it over USB using a Python program. Here's how.

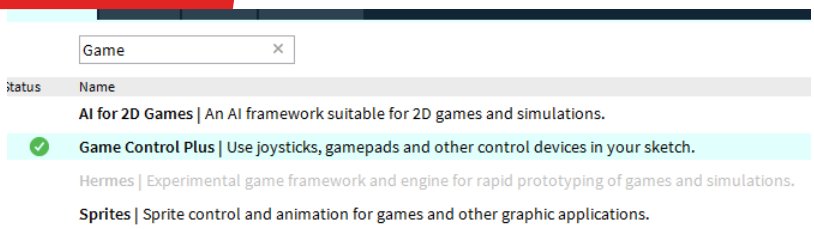
READ MORE

If the upload fails, check you've selected your correct board and port details in the **Tools** menu.

Our Arduino is ready to go!

## Setting Up Processing

Open up processing, you'll be greeted with a blank sketch. Before we do anything here we will need to install a few libraries. Head to the **Sketch** menu and select **Import Library > Add Library**. This will bring up the **Contribution**






We need to install three libraries to make this work. First up is the **Game Control Plus** library. This is what will allow us to use our game controller with Processing. Use the search window to find it, and click install in the bottom right hand corner. Game Control Plus needs another library installed for its configuration tool, so let's get it now. Search for the **G4P** library and install it too.

Finally, we need the **Arduino (firmata)** library. You guessed it, search for it, and click install. With these things installed we are ready to get on with testing that everything will work. We are working with Windows 10 today, but processing is available for most platforms, including Raspberry Pi. Imagine the possibilities!

## Testing the Arduino

Before we dive into creating a custom sketch, let's test the Arduino and Servo with Processing. Open **File > Examples**, and select **ArduinoServo** from the **Contributed Libraries/Arduino (firmata)** folder. We'll use this to test our servo, but first we may need to change a couple of things.

Scroll down through the sketch and find this line:

```
println(Arduino.list());
```

If it is commented out, remove the two slashes before **println(Arduino.list());**, and save the sketch. Run it, by clicking the play icon, and keep an eye on the console at

```
COM1 COM3 COM8
```

In my case, my Arduino was on COM 8, which was the third port listed here. This is important as the code in the line below has an **Array** whose value determines which COM port to use.

```
println(Arduino.list());

// Modify this line, by changing the "0" to the index of the serial
// port corresponding to your Arduino board (as it appears in the list
// printed by the line above).
arduino = new Arduino(this, Arduino.list()[0], 57600);
```

We need to change this to reflect our COM port. For me, it was the third position, or index number 2:

```
arduino = new Arduino(this, Arduino.list()[2], 5
```

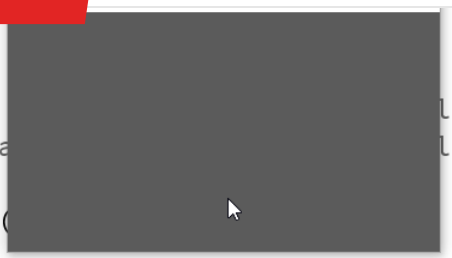
We need to make a couple of other small changes to this code to test it. Scroll down to where the Arduino pins are setup and comment out one of the lines here. Change the other one to **Pin 10**.

```
//arduino.pinMode(4, Arduino.SERVO);
arduino.pinMode(10, Arduino.SERVO);
```

We need to do the same thing in the **Draw()** method:

```
arduino.servoWrite(10, constrain(mouseX / 2, 0,
// arduino.servoWrite(4, constrain(180 - mouseX
```

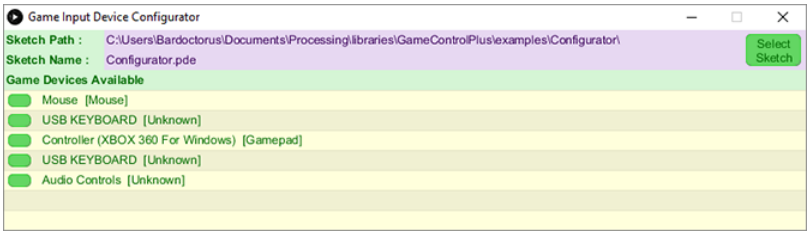
Save the sketch, and run it. You should be able to move your servo by moving your mouse back and forth across the window the program generates.



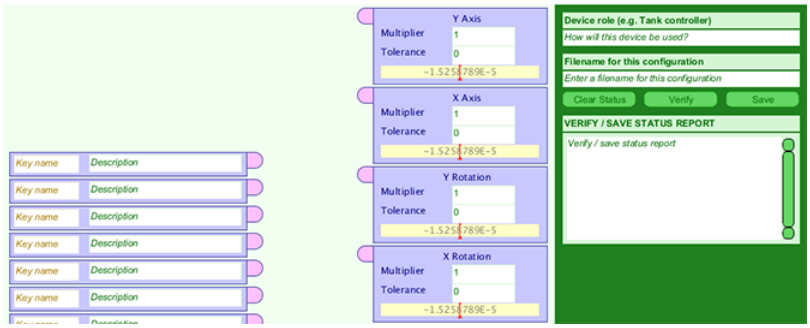

If it doesn't work for you, check your Servo wiring, and check you have the right array position for your COM port. Once you know the Arduino is talking nicely with Processing, it's time to move on.

## Configuring the Controller

The Game Control Plus library we are using comes with a powerful configuration too. Make sure your controller is plugged in, open the **Configurator** example project, and run it. You will get a menu like this:

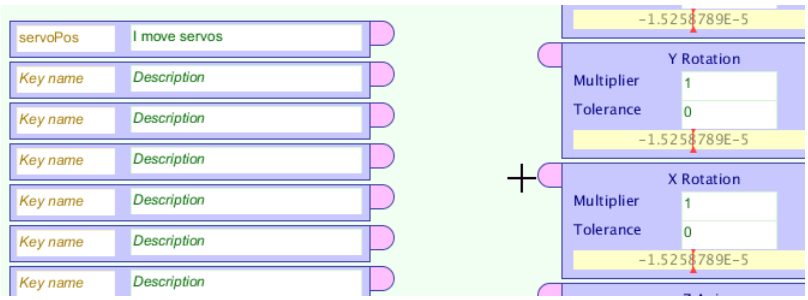



Click on your controller name, and a much larger configuration window will pop up.



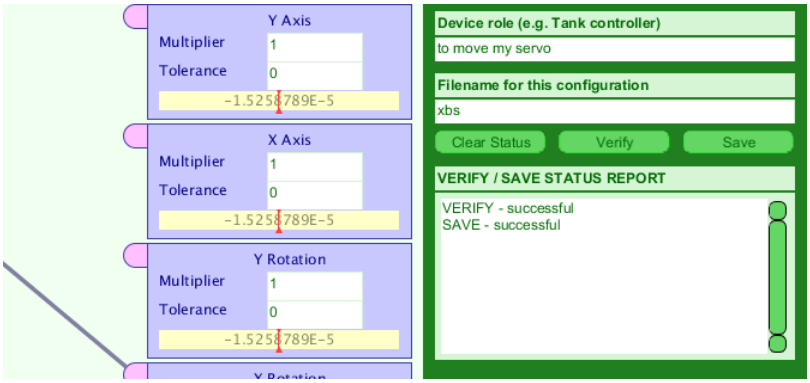

This may look fairly daunting, but it's designed to be as simple as possible. On the left side fill out the first key with

In the box next to it you can give a brief description of what it does. Now pick up your controller and move the stick you wish to use with your servo. A little experimentation shows that the right thumbstick corresponds with the X Rotation box. Drag a line between the **servoPos** variable, and this box.




Now we need to save our configuration as a data file. In the top right of the window, fill out the **Device role** field and the **Filename** field.

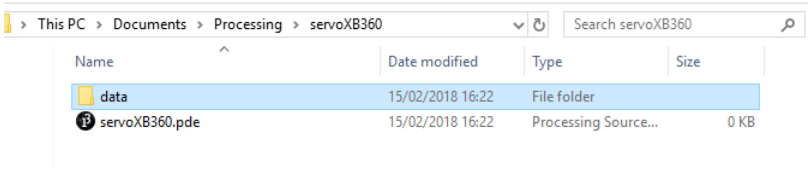
The filename is important, as you'll be using it in your code. I'm keeping it simple by calling it **xbs**. Click **Verify** then **Save**. This writes a file with instructions for our controller which we can use later.




## Preparing the Custom Sketch Folder

Let's set up our working folder. Open up a blank processing sketch, and save it under whatever name you like. This will create a directory for it in the save location.

and copy the folder labelled **data**. This folder contains the configuration file we just created. Navigate to the directory of your newly saved blank sketch, and paste the data folder.




## Creating the Custom Sketch

Now everything is in place and we can start making a sketch using our two libraries. We'll go through this step by step, but you can **download the full sketch and data folder** if you want to jump ahead. Note that you may still need to modify the code to reflect your Arduino COM ports.

Begin by importing all the libraries we will need:

```
import processing.serial.*;
import net.java.games.input.*;
import org.gamecontrolplus.*;
import org.gamecontrolplus.gui.*;
import cc.arduino.*;
import org.firmata.*;
```

We also need to declare our **ControlDevice**, **I/O**, and **Arduino**, along with a **float** to hold values from our thumbstick:

```
ControlDevice cont;
ControlIO control;
Arduino arduino;
float thumb;
```

Our **setup()** method creates a small window, an instance of the controller, and matches the device with our config file. This is where it is important to get the filename of our configuration data file correct:

```
    control = ControlIO.getInstance(this);
    cont = control.getMatchedDevice("xbs");

    if (cont == null) {
      println("not today chump"); // write better
      System.exit(-1);
    }
    // println(Arduino.list());
    arduino = new Arduino(this, Arduino.list()[2],
    arduino.pinMode(10, Arduino.SERVO);
  }
```

We also check if there is no applicable controller at this stage, and quit out of the program if needs be. While the window created with **size()** is not needed, it will give us some feedback later as to whether we are getting useful values from our controller. We also initialise our Arduino and pin here just like we did while testing.

Now we create a little method to grab the input value from our controller, and map it to values our servo will be able to use:

```
  public void getUserInput() {
    thumb = map(cont.getSlider("servoPos").getValu
  }
```

This one line of code uses our data file to get our named control **servoPos**, which is linked to the right thumbstick of the controller, and read values from it. It then maps the values and stores the value in our **thumb** float variable.

Right now this code never gets called, we'll fix that now.

```
  void draw() {
    getUserInput();
    background(thumb,100,255);
```

The **draw()** is similar to the **loop()** method in the Arduino IDE. Every frame, it calls the **getUserInput()** method and updates the **thumb** value. It uses this value to change the red value of the **background()** giving us a visual indicator of the change in value. It then writes this value to the servo using the **arduino.servoWrite()** function. Note that we have to cast **thumb** as an integer value as the servoWrite function takes two integers (pin number, and angle) as its arguments.

Check your code for errors, save it, and click run. After a slight delay to initialise the Arduino, it should look like this:

## Control With Game Controller and Arduino: Finished!

This project was in many ways quite in depth for the **inexperienced coder**, despite the fantastic libraries available to help us. What it represents is a new way to think about controlling robots, and any other devices you build.

This project would go along perfectly with our guide on building a **Laser Turret**, giving you full control over it. You could set up a Piezo buzzer like in our **Simple Arduino Alarm** tutorial and use your controller to change the pitch of the buzzer or color of the lights.

Or you could, well, build a massive robot and take over the earth. As long as you had a USB cord long enough!

Explore more about: **Arduino**, **Game Controller**, **Robotics**.



1 COMMENT WRITE A COMMENT

Scroll down for the next article

Beginner to Arduino for Days with ESP8266

Can Make Right Now 6 Best Arduino Alternative Microcontrollers Arduino Projects for Beginners Connect Arduino to Android

# Nano Replace the Raspberry Pi?

By **Ian Buckley** / April 12, 2019 12-04-2019 / 6 minutes

Ian Buckley
**107** articles

Ian Buckley started out with a degree in Music composition, before devoting his time to DIY tech and coding. He now works as freelance journalist, performer an video producer living in Berlin, Germany. When he's not writing o on stage, he's tinkering with DIY electronics or code in the hope of becoming a mad scientist.



It's an exciting time for small factor computing. As if the Raspberry Pi wasn't enough of an all purpose machine, more powerful boards capable of incredible feats keep appearing.

The Jetson Nano from Nvidia is a recent addition to the ranks of super powerful machine learning enabled boards. What makes it special? Should you buy one? What is the Nvidia Jetson Nano all about?

## What Is the Nvidia Jetson Nano?

NVIDIA's Jetson Nano machine learning development board

The Jetson Nano is a Single Board Computer (SBC) around the size of a Raspberry Pi, and aimed at AI and machine learning. Seemingly a direct competitor to the Google Coral

Nvidia are leveraging their prowess for graphics processing power for these small computers, using parallel neural networks to process multiple videos and sensors simultaneously.

While all three Jetson boards aim to be accessible to all, the Nano is for both hobby and professional developers. The dev kit comprises two parts—a baseboard for connectivity, and a System On Module (SOM) for the actual processing units.

## What Is System on Module?

The Jetson Nano System on Module

System on Module refers to any development board which has all system-critical parts in a removable module. The Nano features a 260-pin edge connector to attach it to a baseboard for development.

Once development is over, the SOM can be removed and added into an embedded system with custom inputs, and a new SOM connects to the baseboard for further development.

If all of this sounds a little familiar, it is!

This is the same setup as the **Google Coral Dev board**, which is a similar size, and also aimed at embedded machine learning for hobbyists and professionals alike!

**Is the Google Coral Dev Board Better Than a Raspberry Pi?**

Heralding a new era in accessible hobbyist boards, just what is Google's Coral Dev Board? And can it replace your Raspberry Pi?

READ MORE

Designed Headphor
For Everyone

## Related Articles

DIY , ENTERTAINMENT

Build Your Own Ama
Alexa Smart Speake
With a Raspberry Pi

PROGRAMMING , DIY

Why Coding for
Raspberry Pi Is Way
Better With Code-O

PROGRAMMING , DIY

How to Get Started
Rust on Raspberry P

## Latest Videos


The Soliom Security Cam Ru
Completely on Solar Powe

Nvidia has packed a lot into the Jetson Nano:

**SOM**:

- CPU: Quad-core ARM® Cortex-A57 MPCore processor
- GPU: Nvidia Maxwell™ architecture with 128 Nvidia CUDA cores
- RAM: 4 GB 64-bit LPDDR4
- Storage: 16 GB eMMC 5.1 Flash
- Video: 4k @ 30fps encoding, 4k @ 60fps decoding
- Camera: 12 lanes (3×4 or 4×2) MIPI CSI-2 DPHY 1.1 (1.5 Gbps)
- Connectivity: Gigabit Ethernet
- Display: HDMI 2.0 or DP1.2 | eDP 1.4 | DSI (1 x2) 2 simultaneous
- PCIE/USB: 1 x1/2/4 PCIE, 1x USB 3.0, 3x USB 2.0
- I/O: 1x SDIO / 2x SPI / 6x I2C / 2x I2S / GPIOs
- Dimensions: 69.6 mm x 45 mm

**Baseboard**:

- USB: 4x USB 3.0, USB 2.0 Micro-B
- Camera: 1x MIPI CSI-2 DPHY lanes (Raspberry Pi camera compatible)
- LAN: Gigabit Ethernet, M.2 Key E
- Storage: microSD slot
- Display: HDMI 2.0 and eDP 1.4
- Other I/O: GPIO, I2C, I2S, SPI, UART

## What Can It Do?

DeepStream on Jetson Nano

It will come as a shock to nobody that Nvidia has produced a board suited well to visual tasks. Object recognition is a key focus here, and the Visionworks SDK has many potential applications in this field.

Rather than use a separate processing unit for machine learning tasks, the Jetson Nano uses a Maxwell GPU with 128 CUDA cores for the heavy lifting.

The Jetson Inference project features demos of a pre-trained neural network performing high-performance multiple object recognition in a variety of environments. Feature tracking, image stabilization, motion prediction, and multi-source simultaneous feed processing are all featured in the available demo packages.

Perhaps most impressive is the DeepStream technology featured in the above video. Running live analytics on eight simultaneous 1080p streams at 30fps on a small single board computer is incredible, and shows the potential power of the Nano's hardware.

## What Will It Be Used For?

Given its prowess for video analysis and small form factor, the Jetson Nano will almost certainly shine in robotics and autonomous vehicles. Many of the demos show these applications in action.

Given its power and size, it'll also likely work in embedded systems which rely on facial and object recognition.

For Hobbyists like us? It seems to be a perfect blend of powerful machine learning possibilities in a factor familiar

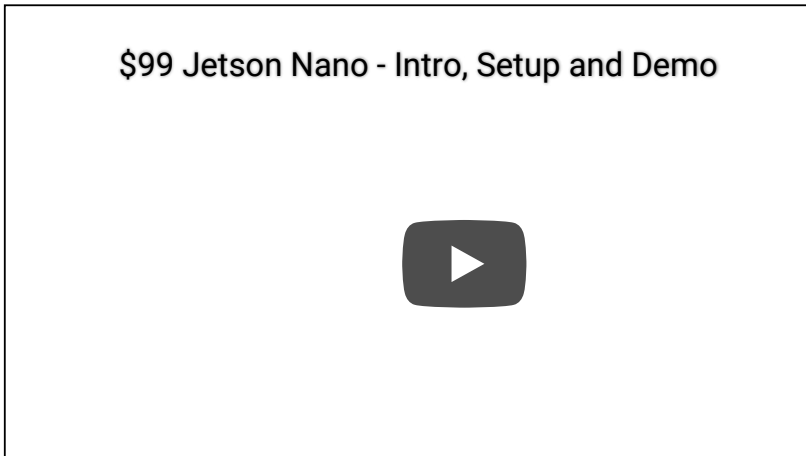
Raspberry Pi, the Jetson Nano is much more suited to the task.

**Get Started With Image Recognition Using TensorFlow and Raspberry Pi**

Want to get to grips with image recognition? Thanks to Tensorflow and a Raspberry Pi, you can get started right away.

READ MORE

## What Else Can the Jetson Nano Do?




The Jetson Nano runs Ubuntu, though a specialized OS image is available from Nvidia featuring software specific to the platform. While the primary focus of the board is machine learning, this is Nvidia so you'd expect some graphical wizardry to be going on too.

You won't be disappointed. Demos showing particle systems, real-time fractal rendering, and an array of visual effects would only until recently have been found on flagship desktop graphics cards.

Given that its video encoding is rated for 4k @ 30fps, and decoding at 60fps, it is safe to assume the Nano will be perfect for video applications too.

It is tough to say which is the better board between the Google Coral Dev board and the Jetson Nano at this stage.

Google's TensorFlow neural network is a dominant force in the field of machine learning. It would follow that Google's own Edge TPU coprocessor might work better for applications of TensorFlow Lite.

On the other hand, Nvidia has already shown an impressive array of machine learning based demos for the Jetson Nano. This, alongside the impressive graphics the Nano is capable of make it a real competitor.

## How Much Does Jetson Nano Cost?

Price is another aspect we haven't covered yet. The Google Coral Dev board retails at $149.99 while the Jetson Nano is only $99. Unless the Coral Dev board can bring something unique to the table, hobbyists and small developers might find the extra $50 a hard stretch to justify.

There is currently no price for the SOM alone for either board, but I would imagine for most hobby developers this won't be quite as important. From a commercial point of view, the performance/price contrast is going to be what makes the critical difference between the Jetson Nano and the Coral Dev board.

The Jetson Nano is available from Nvidia directly along with third party sellers.

**Buy**: **Jetson Nano direct from Nvidia**

## Could It Replace My Raspberry Pi?

While the Google Coral Dev board is powerful, it doesn't stack up to the Raspberry Pi in some ways. The Raspberry

**Using a Raspberry Pi as a Desktop PC: 7 Things I Learned After a Week**

Can a modest Raspberry Pi replace a desktop PC? I spent seven days writing and editing on the Pi, with interesting results.

READ MORE

Sure, the Coral Dev board is powerful, but their own documents advise against attaching a mouse and keyboard. The Coral's custom OS is for SSH connections primarily. It is, however, likely capable of sustaining any variation of Linux. This puts it right back up there as a direct Pi competitor

There is a problem though. If you want a board for learning machine learning, but one that can also perform other daily tasks, why would you buy the Coral Dev Board?

The Jetson Nano supports a display port, and as previously mentioned has impressive video examples straight out of the box. The custom Ubuntu desktop will be familiar to many and the cheaper price point will make it an attractive prospect for many, even those uninterested in machine learning.

## AI for Everyone

At this stage, it is hard to say which will be the better board. It's also unknown which will be more accessible to home developers. I look forward to spending time with both the Coral Dev and Jetson Nano boards to get a definitive answer!

It's an exciting time to be tinkering with SBCs! If you are new to it and want a place to start, get a Raspberry Pi and follow our **ultimate getting started guide**!

Whether you're a current Pi owner who wants to learn more or a potential owner of this credit-card size device, this isn't a guide you want to miss.

READ MORE

Explore more about: **Google TensorFlow**, **Jetson Nano**, **Machine Learning**, **Raspberry Pi**.



1 COMMENT WRITE A COMMENT

Scroll down for the next article

**Outdoor Security Camera Review**

**Best Ergonomic Vertical Mice**

**Cover Letter Using Canva**

**Windows 10 the Fast Way**

**Perfect Mac on Apple's Website**

**Secure Than Dumb Phones**

**File From Windows Explorer**

**iPhone, iPad, or Mac?**