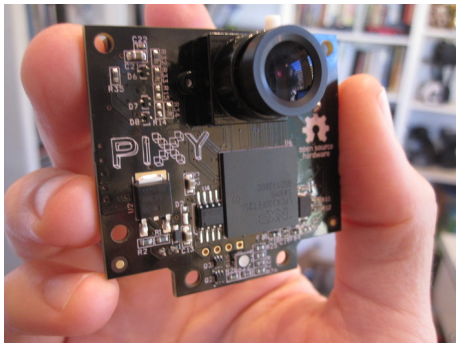
Introducing Pixy

- Small, fast, easy-to-use, low-cost, readily-available vision system
- Learns to detect objects that you teach it
- Outputs what it detects 50 times per second
- Connects to Arduino with included cable. Also works with Raspberry Pi, BeagleBone and similar controllers
- All libraries for Arduino, Raspberry Pi, etc. are provided
- C/C++ and Python are supported
- Communicates via one of several interfaces: SPI, I2C, UART, USB or analog/digital output
- Configuration utility runs on Windows, MacOS and Linux
- All software/firmare is open-source GNU-licensed
- All hardware documentation including schematics, bill of materials, PCB layout, etc. are provided
- Can be used for all kinds of cool projects



How Pixy got started

Pixy (CMUcam5) is a partnership between the Carnegie Mellon Robotics Institute and Charmed Labs. Pixy comes from a long line of CMUcams, but Pixy got its real start as a Kickstarter campaign. It first started shipping in March of 2014, but it's already become **the most popular vision system in history**! Pixy is funded exclusively through sales, so thank you for making Pixy a success! You can watch the original Kickstarter video below -- it's a good introduction!

Vision as a Sensor

If you want your robot to perform a task such as picking up an object, chasing a ball, locating a charging station, etc., and you want a single sensor to help accomplish all of these tasks, then **vision** is your sensor. Vision (image) sensors are useful because they are so flexible. With the right algorithm, an image sensor can sense or detect practically anything. But there are two drawbacks with image sensors: 1) they output lots of data, dozens of megabytes per second, and 2) processing this amount of data can overwhelm many processors. And if the processor can keep up with the data, much of its processing power won't be available for other tasks.

Pixy addresses these problems by pairing a powerful dedicated processor with the image sensor. Pixy processes images from the image sensor and only sends the useful information (e.g. purple dinosaur detected at x=54, y=103) to your microcontroller. And it does this at frame rate (50 Hz). The information is available through one of several interfaces: UART serial, SPI, I2C, USB, or digital/analog output. So your Arduino or other microcontroller can easily talk with Pixy and still have plenty of CPU available for other tasks.

It's possible to hook up multiple Pixys to your microcontroller -- for example, a robot with 4 Pixys and 360 degrees of

sensing. Or use Pixy without a microcontroller and use the digital or analog outputs to trigger events, switches, servos, etc.

Controller support

Pixy can easily connect to lots of different controllers because it supports several interface options (UART serial, SPI, I2C, USB, or digital/analog output), but Pixy began its life talking to Arduinos. Over the last several months we've added support for Arduino Due, Raspberry Pi and BeagleBone Black. Software libraries are provided for all of these platforms so you can get up and running quickly. Additionally, we've added a Python API if you're using a Linux-based controller (e.g. Raspberry Pi, BeagleBone).

Purple dinosaurs (and other things)

Pixy uses a color-based filtering algorithm to detect objects. Color-based filtering methods are popular because they are fast, efficient, and relatively robust. Most of us are familiar with RGB (red, green, and blue) to represent colors. Pixy calculates the color (hue) and saturation of each RGB pixel from the image sensor and uses these as the primary filtering parameters. The hue of an object remains largely unchanged with changes in lighting and exposure. Changes in lighting and exposure can have a frustrating effect on color filtering algorithms, causing them to break. Pixy's filtering algorithm is robust when it comes to lighting and exposure changes.

Seven color signatures

Pixy remembers up to 7 different color signatures, which means that if you have 7 different objects with unique colors, Pixy's color filtering algorithm will have no problem identifying them. If you need more than seven, you can use color codes (see below).

Hundreds of objects

Pixy can find literally hundreds of objects at a time. It uses a connected components algorithm to determine where one object begins and another ends. Pixy then compiles the sizes and locations of each object and reports them through one of its interfaces (e.g. SPI).

50 frames per second

What does "50 frames per second" mean? In short, it means Pixy is fast. Pixy processes an entire 640x400 image frame every 1/50th of a second (20 milliseconds). This means that you get a complete update of all detected objects' positions every 20 ms. At this rate, tracking the path of falling/bouncing ball is possible. (A ball traveling at 30 mph moves less than a foot in 20 ms.)

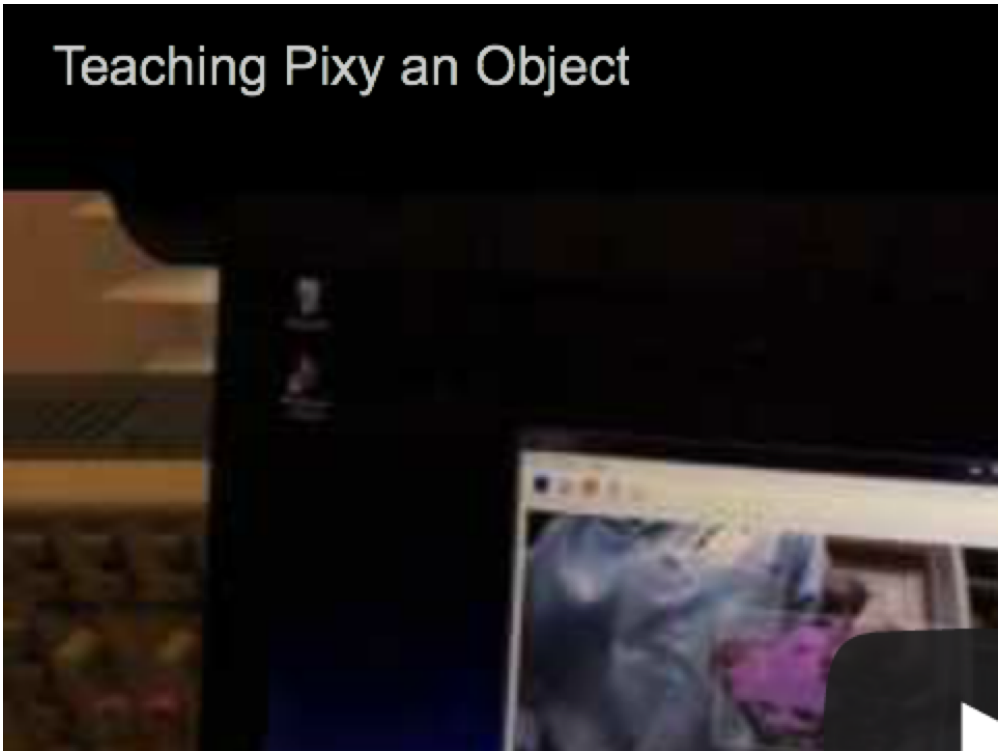
Teach it the objects you're interested in

Pixy is unique because you can physically teach it what you are interested in sensing. Purple dinosaur? Place the dinosaur in front of Pixy and press the button. Orange ball? Place the ball in front of Pixy and press the button. It's

easy, and it's fast.

More specifically, you teach Pixy by holding the object in front of its lens while holding down the button located on top. While doing this, the RGB LED under the lens provides feedback regarding which object it is looking at directly. For example, the LED turns orange when an orange ball is placed directly in front of Pixy. Release the button and Pixy generates a statistical model of the colors contained in the object and stores them in flash. It will then use this statistical model to find objects with similar color signatures in its frame from then on.

Pixy can learn seven color signatures, numbered 1-7. Color signature 1 is the default signature. To teach Pixy the other signatures (2-7) requires a simple button pressing sequence.

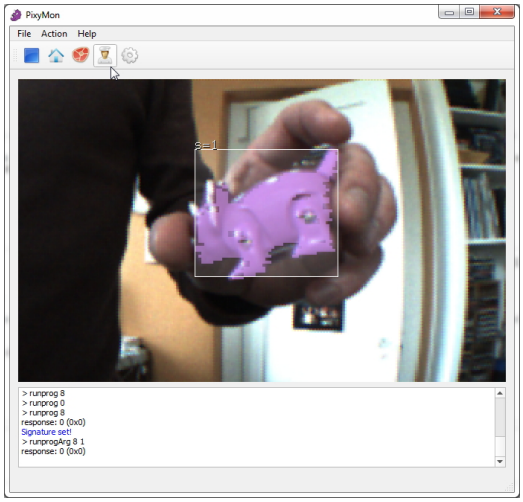


PixyMon lets you see what Pixy sees

PixyMon is an application that runs on Windows, MacOs and Linux. It allows you to see what Pixy sees, either as raw or processed video. It also allows you to configure your Pixy, set the output port and manage color signatures. PixyMon communicates with Pixy over a standard mini USB cable.

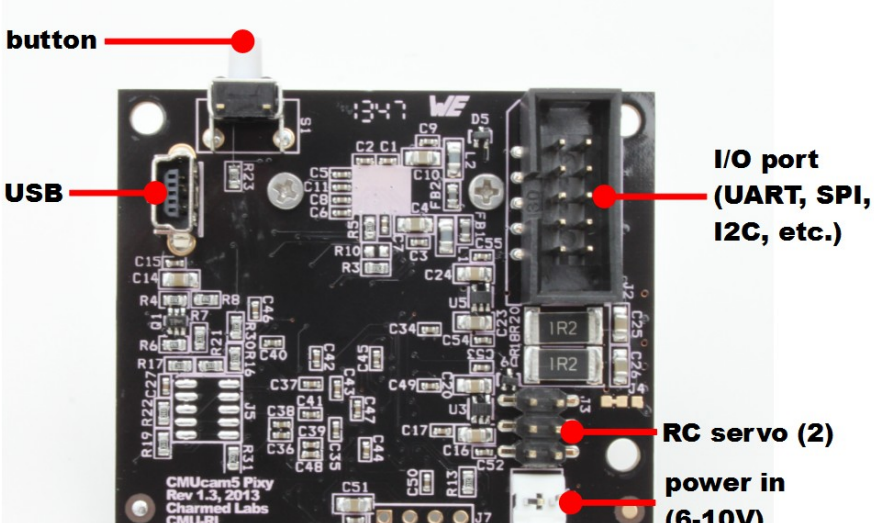
PixyMon is great for debugging your application. You can plug a USB cable into the back of Pixy and run PixyMon and then see what Pixy sees while it is hooked to your Arduino or other microcontroller -- no need to unplug anything

and then see what Pixy sees while it is hooked to your Arduino or other microcontroller -- no need to unplug anything. PixyMon is open source, like everything else.



CMUcam5 Pixy - Technical specs

- Processor: NXP LPC4330, 204 MHz, dual core
- Image sensor: Omnivision OV9715, 1/4", 1280x800
- Lens field-of-view: 75 degrees horizontal, 47 degrees vertical
- Lens type: standard M12 (several different types available)
- Power consumption: 140 mA typical
- Power input: USB input (5V) or unregulated input (6V to 10V)
- RAM: 264K bytes
- Flash: 1M bytes
- Available data outputs: UART serial, SPI, I2C, USB, digital, analog
- Dimensions: 2.1" x 2.0" x 1.4
- Weight: 27 grams



What's a "color code"?

A color code (CC) is two or more color blocks placed close together. Pixy can detect and decode CCs and present them as special objects. CCs are useful if you have lots of objects you want to detect and identify (i.e. more than could be detected with the seven separate color signatures alone.)

A color code scheme with 2 blocks and 4 different colors can differentiate up to 12 unique objects. CCs with 3, 4 and 5 blocks (and more colors) are possible and can allow for many, many more unique objects. (In fact, thousands of unique codes are possible by using CCs with 5 blocks and 6 colors.)

Why Color Codes?

CCs are useful if you have lots of objects you want to detect and identify, more than could be detected with the seven separate color signatures alone. CCs also improve detection accuracy by decreasing false detections. That is, there is a low probability that specific colors will occur both in a specific order and close together. The drawback is that you need to place a CC on each object you're interested in detecting. Often the object you're interested in (yellow ball, purple toy) has a unique color signature and CCs aren't needed. Objects with CCs and objects without CCs can be used side-by-side with no problems, so you are free to use CCs for some objects and not others.

CCs give you an accurate angle estimate of the object (in addition to the position and size). This is a computational "freebie" that some applications may find useful. The angle estimate, decoded CCs, regular objects and all of their positions and sizes are provided at 50 frames per second.

CCs might be particularly useful for helping a robot navigate. For example, an indoor environment with CCs uniquely identifying each doorway and hallway would be both low-cost and robust.

Face detection

Face detection will be released soon in a new version of firmware -- stay tuned! When it's released, all Pixy owners will thereby have a new sensor (a face detector!) with no new hardware necessary. We'll continue to improve and add new capabilities to Pixy as long as, you know, people buy it.

Teach Pixy an Object

Teaching Pixy an object is super easy, but first let's talk about which objects will work well with Pixy. Pixy uses a hue-based color filtering algorithm to detect objects. Since Pixy uses hue (color), the object needs to have a distinct hue. Here are some objects that are good because they have good, distinct hues.

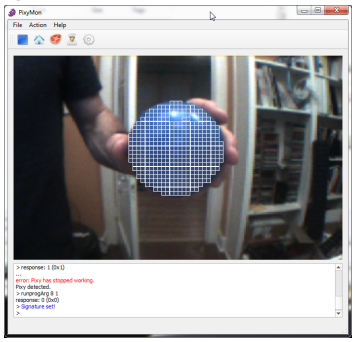
Here are some bad objects because either there is no hue (black, white or gray) or the hue is not distinct.

Keeping these guidelines in mind, choose an object to teach Pixy. (First, apply power to Pixy via battery or USB cable if you haven't already. When you power up Pixy, it will go through a series of LED flashes. Wait for the LED to turn off before teaching Pixy an object.) Now, hold down the button on top of Pixy. After about 1 second, the LED will turn on - first white, then red, then other colors - but when it turns red, release the button.

When you release the button, Pixy will enter what's called "light pipe" mode, where the LED color is the color of the object that Pixy has "locked" onto. Pixy will lock onto objects in the center of its video frame, so hold the object directly in front of Pixy, between 6 and 20 inches from the lens.

Pixy uses a region growing algorithm to try to determine which pixels are part of your object and which pixels are part of the background. Using these pixels, Pixy will try to create a statistical model of your object so it can detect it reliably under different lighting conditions. Use the LED color as feedback to determine if Pixy has a good lock on the object, and use the following guidelines to judge:

1. When Pixy has locked onto your object, the LED color should match the color of your object.
2. The brighter the LED, the better the lock. Move the object a little and see if you can maximize the LED brightness.
3. Look at the region-growing grid in the PixyMon window and see how the size of the grid corresponds to the LED color and brightness. A grid that covers more of your object is better than a grid that covers only some of your object and results in a brighter LED. The picture below shows a grid that covers most of the object (good).



Learning when Pixy has a decent lock by looking at the LED might take a little practice, but once you get the feel of it, you can teach Pixy new objects reliably **without needing to use PixyMon**!

When you are satisfied, press and release the button on Pixy, like you'd click your mouse. The LED will flash a couple times indicating that Pixy has now "learned" your object. It will now start tracking your object.

The video below is a good short "how-to" regarding teaching Pixy objects.

Multiple Signatures

Pixy can learn up to seven color signatures. We can teach Pixy the 1st color signature by releasing the button when the LED turns red. If we continue to hold down the button, the LED will turn orange, yellow, etc., indicating the remaining color signatures. Here are the signatures in order:

1. Red
2. Orange
3. Yellow
4. Green
5. Cyan (light blue)
6. Blue
7. Violet

The color signature number is determined by *when* you release the button. Release the button when the LED is yellow and you're teaching signature 3. Release the button when the LED is blue and you're teaching signature 6. These colors *are not related to the hue of the object*. The colors are used only to indicate the signature number. So, for example, signature 1 can be a yellow object, even though signature 1 is indicated by a red LED, and signature 2 could be a pink object even though signature 2 is indicated by an orange LED.

After you teach Pixy a signature, it saves the signature in flash, so when you power cycle your Pixy, it will remember the signatures you taught it and continue to track objects that match these signatures.

If you accidentally find yourself teaching signature 2 when you meant to teach signature 1 (i.e. you released the button when it was orange instead of red), for example, just hold down the button until the LED turns off. This is how you tell Pixy to cancel teach mode. You can then start over by holding the button down again.

White Balance

White Balance

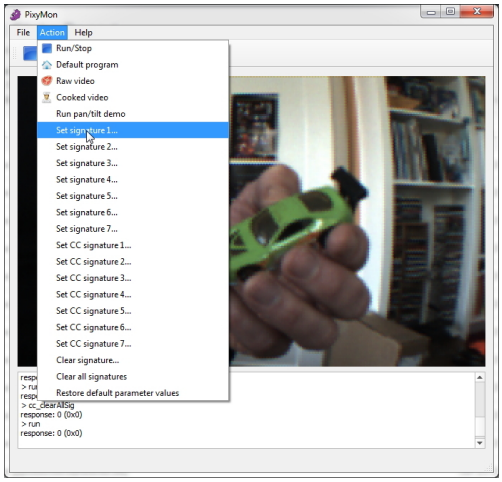
Some types of lighting (such as incandescent) have a reddish hue and others (such as fluorescent) have a bluish hue. The lighting can affect your color signatures. For example, if you teach an object under incandescent lighting and move into a room with fluorescent lighting, the color signatures will likely no longer work as well. You can either re-teach all signatures or you can adjust the white balance.

When you first apply power to Pixy, it will spend the first 5 or so seconds determining the correct white balance to use. It will then disable automatic white balance. If you wish to readjust the white balance, hold down the button until the LED turns white and release. It happens quickly, so be prepared! Pixy is now in automatic white balance mode. You can hold a white sheet of paper in front of Pixy so Pixy can adjust the white balance. It only takes 2 or 3 seconds to adjust the white balance, after which you can press and release the button (like a mouse click). The LED will flash, indicating success, and now Pixy is "rebalanced".

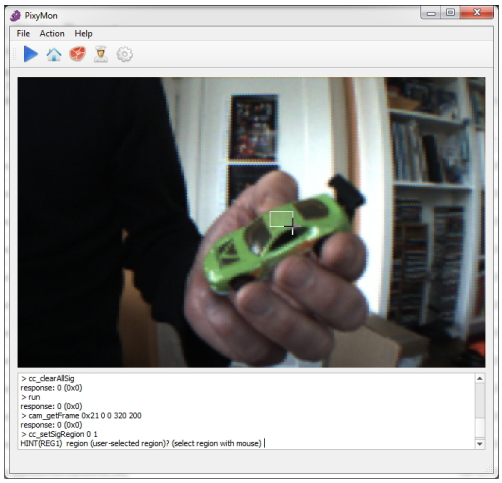
Teaching through PixyMon

You can also teach Pixy an object through PixyMon. This may be useful if the object you want to teach is small, or if you want more control over which pixels are used for teaching. Begin by plugging in the USB cable between Pixy and your computer and running PixyMon.

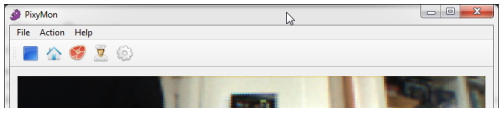
Now hold the object you want to teach in front of Pixy and select **Action➜Set signature 1** from the pulldown menu.



Using the mouse, click and drag to select the region you want Pixy to use to learn the object.

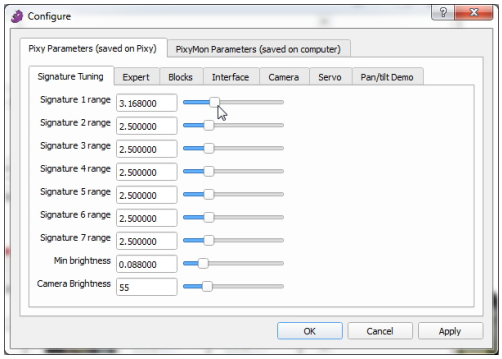


That's it! After you select the region, Pixy will "learn" the object and automatically go into "cooked" video mode so you can verify how well your color signature is working.

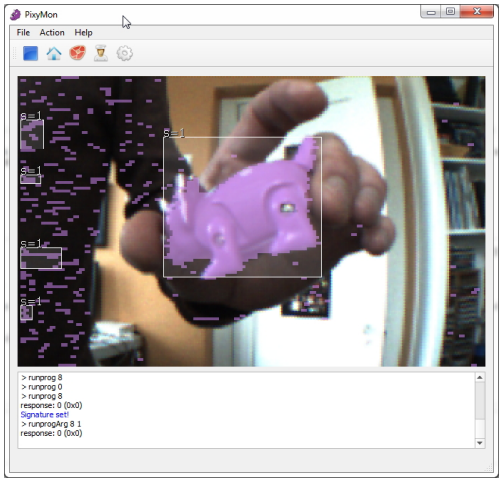


Signature tuning

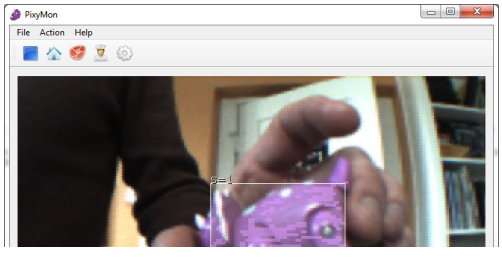
Sometimes the color signatures you teach Pixy need to be "tweaked" -- that is, you are getting some false positives (Pixy is detecting objects that aren't the objects that you intended) or false negatives (Pixy isn't detecting the object you just taught it, or it's detecting the object intermittently.) You can tweak things by bringing up the Configure dialog (click on the gear icon or select **File➔Configure**). Select the **Signature Tuning** pane under **Pixy Parameters**.

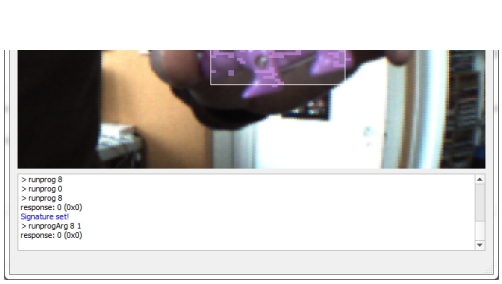


Use the slider for **Signature 1 range** to adjust the inclusiveness of signature 1 (assuming it's signature 1 you're wanting to adjust). Slide it to the left if you want to be less inclusive (i.e. if you're seeing false positives, like the picture below):

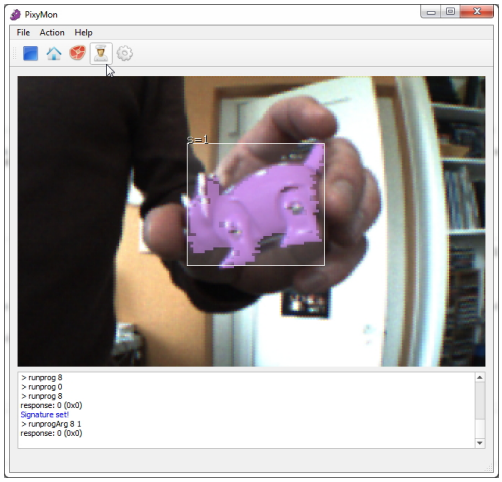


or slide it to the right if you want to be more inclusive (i.e. you're seeing false negatives) or the detection is intermittent or sparse (like in the picture below):





Choose a slider value that provides good strong detection, like the picture below:



You can adjust all seven color signatures this way to maximize detection accuracy for all signatures. Be sure to press **Apply** or **OK** to save the slider ranges! The adjusted values won't be saved if you press **Cancel** or dismiss the dialog.

There are more tips here on how to improve the color signatures.

Powering Pixy

You have some choices when powering Pixy... always good to have choices! We expect that most Pixy users will either power Pixy through the USB cable/connector or though the Arduino cable (I/O connector). These two choices are the simplest, but here are all available power options:

1. Power Pixy through the USB cable/connector (regulated 5V)
2. Power Pixy through the I/O connector (regulated 5V)
3. Power Pixy through the power connector (unregulated 6V to 10V)

Note: you can have both the Arduino cable **and** the USB cable plugged into Pixy simultaneously without any bad things happening. In fact, this is very useful when you want to do a quick check to see what Pixy sees through PixyMon, while Pixy is connected to your Arduino (in situ, so to speak).

For reference, Pixy's typical power consumption is 140mA at 5V.

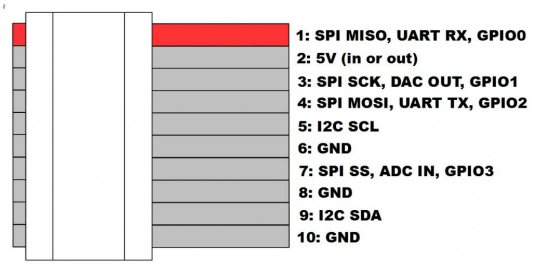
USB cable/connector

When you plug Pixy into your computer via USB cable, Pixy is powered through the regulated 5V provided by the USB port. It is also possible to power small RC servos through the USB cable (like the ones in the pan/tilt mechanism) as long as the USB cable is kept somewhat short (less than 4ft is best).

I/O connector

Pin 2 of the I/O connector can accept 5V for powering Pixy. This is convenient when you want to power Pixy through the Arduino cable (ie, by hooking up the Arduino cable, your Arduino is powering Pixy, assuming your Arduino is suitably powered.) Or you can make your own I/O cable for Pixy communication/power, **but be careful! Pin 2 and the ground pins (pins 6, 8 and 10) are not reverse-polarity protected. Get the power backwards and Pixy is a goner. For good!**

Note also, ribbon cables have poor current-carrying ability. So it's unlikely you can power Pixy through the I/O connector via ribbon cable **and** have enough power left over for controlling RC servos. The voltage will drop too much across the ribbon cable. Nothing bad will happen, if you try this but the servo will likely malfunction.

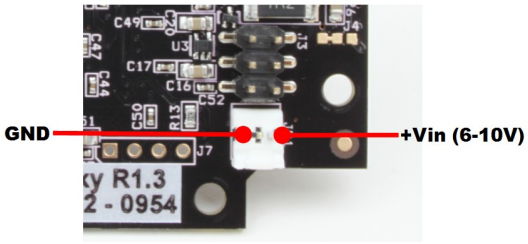


Power connector

Although we totally expect that 90% of Pixy users to use the first two choices (above), we've included a third choice -- an unregulated power input. Because we love it when we feel taken care of, and we've taken care of you, unregulated Pixy power person.

It is recommended that you use this option if you plan on using the pan/tilt unit with Pixy and an Arduino (and no USB cable). In this case, **Pixy is sourcing the power** and powering the Arduino through the Arduino cable, and **that's perfectly fine**, because Pixy can source up to 1.5A of current, which is plenty for itself, the servos and the Arduino. If you attempt to have the **Arduino source the power** and power Pixy and the pan/tilt through the Arduino cable, either the Arduino's power regulator will be overwhelmed, or you'll lose a lot of power through the cable, both of which will mean **the servos won't function** (they'll probably move to one end of their limits and buzz).

OK, here are the pinouts and polarity of the power connector:



The power connector is reverse-polarity protected, so nothing bad will happen if you get things backwards. The mating connector is a polarized connector from Molex that comes in two parts:

1. **The housing**, Molex PN 22-01-3027, available from Digikey and Mouser
2. **The crimp pins**, Molex PN 08-50-0113, you'll need at least 2 of them, also available from Digikey and Mouser

It is safe to connect Pixy to a computer via USB while powered using this method.

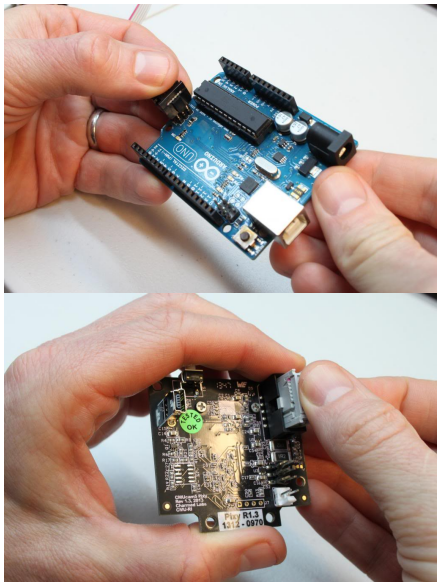
Hooking up Pixy to a Microcontroller (like an Arduino)

Pixy is meant to talk to a microcontroller.

- If you want to hook your Pixy to a **Raspberry Pi**, go here.
- If you want to hook your Pixy to a **BeagleBone Black**, go here.
- If you want to hook your Pixy to something not listed here, check out our porting guide.
- If you want to hook your Pixy to an **Arduino**, keep reading!

Out of the box, Pixy is ready to talk to an Arduino. It sends block information to Arduino at 1 Mbits/second, which means Pixy can send more than 6000 detected objects per second or 135 detected objects per frame (Pixy can process 50 frames per second.)

OK, to get Pixy and Arduino talking to each other, use the supplied Arduino cable to connect Pixy to your Arduino.



Next, download the latest Arduino library "arduino_pixy-x.y.z.zip" here. Bring up the Arduino IDE and import the Pixy library by selecting **Sketch➜Import Library** in the Arduino IDE, and then browsing to the Arduino zip file that you just downloaded.

Next, load the "hello_world" example by selecting it in **File➜Examples➜Pixy**. Upload it and bring up the **Serial Monitor**. You should see messages printed that look similar to this:

```
Detected 1:
  block 0: sig: 1 x: 159 y: 109 width: 61 height: 61
Detected 1:
  block 0: sig: 1 x: 173 y: 114 width: 60 height: 61
Detected 1:
  block 0: sig: 1 x: 146 y: 111 width: 70 height: 65
...
```

Note, this example will only print messages if Pixy is running the "default program" and an object that matches one of its color signatures is visible. This is what PixyMon looks like when Pixy is running the default program and it has detected objects:

Arduino API

Using Pixy with Arduino is really simple. You simply include the SPI and Pixy headers:

```
#include <SPI.h>
#include <Pixy.h>
```

And make a global instance of Pixy by putting this little guy outside your setup() and loop() functions:

```
Pixy pixy;
```

The most important method in the Arduino library is `getBlocks()`, which returns the number of objects Pixy has detected. You can then look in the `pixy.blocks[]` array for information about each detected object (one array member for each detected object.) Each array member (`i`) contains the following fields:

- `pixy.blocks[i].signature` The signature number of the detected object (1-7 for normal signatures)
- `pixy.blocks[i].x` The x location of the center of the detected object (0 to 319)
- `pixy.blocks[i].y` The y location of the center of the detected object (0 to 199)
- `pixy.blocks[i].width` The width of the detected object (1 to 320)
- `pixy.blocks[i].height` The height of the detected object (1 to 200)
- `pixy.blocks[i].angle` The angle of the object detected object if the detected object is a color code.
- `pixy.blocks[i].print()` A member function that prints the detected object information to the serial port

So it's simple to talk to Pixy with your Arduino! For more information on the Arduino Library and API, go here.

Updating Pixy Library for Arduino

Before installing a new version of the Arduino Library, it's recommended that you delete the existing library. To do this, you can go into your C:\Users\<yourname>\Documents\Arduino\libraries (or similar directory, <yourname>/Documents/Arduino in OSX and Linux) and remove the Pixy directory. Then re-run the Arduino IDE.

Mounting Pixy

Pixy was designed to befriend your robot or any other device you've dreamed up. So, with each Pixy we include brackets and fasteners to make mounting Pixy to your device fairly simple.

Here's what you'll need:

A small phillips screwdriver

Your Pixy

The bag of fasteners that came with your Pixy

Before we continue, take all four right-angle brackets out of the little baggie and examine them. Notice that two of them have **two** threaded holes and two of them have **one** threaded hole. We'll call these two-threaded and one-threaded brackets, respectively. Go ahead and separate these two bracket types to make things easier.

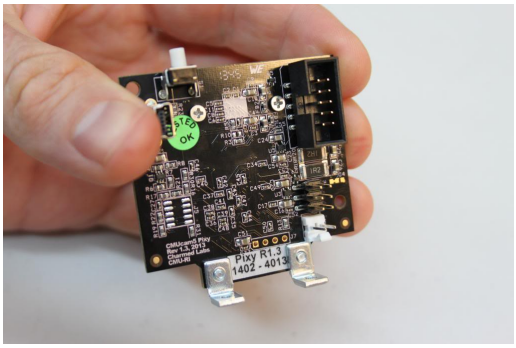
Notice also that the baggie contains two different types of fasteners, three longer fasteners and five shorter fasteners. All of them are the same 4-40 thread.

Right-angle mount

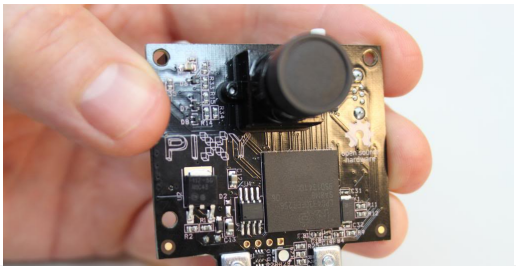
The simplest way to mount Pixy is at a right angle, which is usually all that's required. Take two of the shorter 4-40 fasteners and two of the two-threaded brackets (or one-threaded brackets if you wish) and attach to Pixy as shown.

But first, a word about the two-threaded brackets. If you look really closely at one of these guys, you'll notice that one of its legs is ever-so-slightly longer than the other. Say what? It's true--- I don't know why. But you can best see this by looking at the holes in the bracket. One of the holes in centered in its leg and the other hole is not centered, because its leg is slightly longer. Weird. Just make sure that when you use the two-threaded bracket you keep this in mind, otherwise your Pixy might look a little crooked after you finish mounting. Just a heads-up, to prevent some possible confusion.

Back mount



Front mount

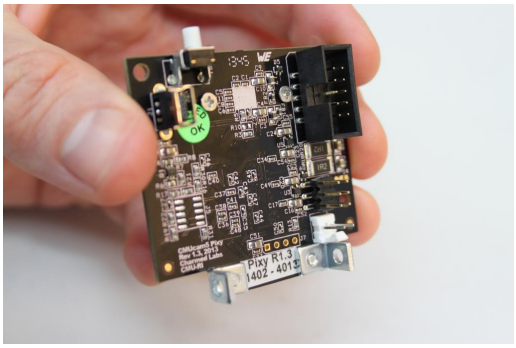


You can then mount Pixy by drilling some holes into your device and using the supplied 4-40 fasteners (long or short). Or you can use self-tapping fasteners (wood or sheet-metal fasteners) and the one-threaded brackets to attach Pixy to your device. Use the handy dandy drill template to make drilling the holes easier.

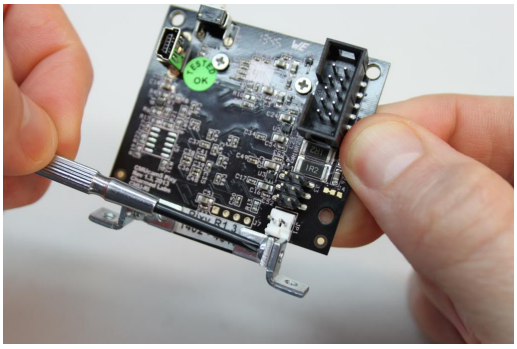
Tilt pivot mount

Sometimes you want to be able to adjust the tilt angle of your sensor so it's looking where you want it to look (down or up). You can do this with a motorized tilt, using a similar mechanism to the pan/tilt. Or you can use a fixed, but adjustable tilt pivot mount, which is what we'll describe now.

Start by attaching the one-threaded brackets to Pixy with the shorter 4-40 fasteners as shown in the picture.



Then attach the two-threaded brackets with two more shorter 4-40 fasteners. Refer to the picture below.



You can then mount Pixy by drilling some holes into your device and using the long 4-40 fasteners. Or you can use self-tapping fasteners that will fit though the holes in the bracket. Use the handy dandy drill template to make drilling the holes easier.

Choose the tilt angle you like best and tighten the fasteners tight. Readjust by loosening/re-tightening.