adafruit learning system

12mm LED Pixels

Created by Phillip Burgess

Last updated on 2018-08-22 03:30:51 PM UTC

Guide Contents

Project Ideas

These pixels could be used for stuff like...

An LED coffee table (this one was “hand made” some time ago...now you could simply use a long strand of our LED pixels and skip the complicated wiring and driver parts):

A trippy “light bar”:

Signs and displays:

A ball of LEDs:

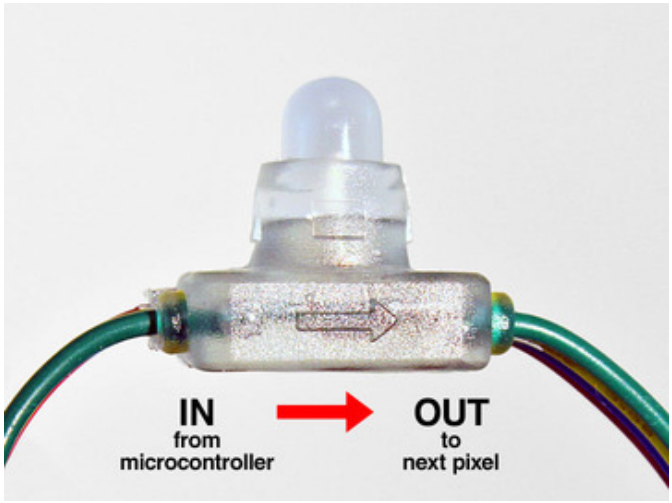
Wiring

EARLIER VERSIONS OF THESE PIXELS USED A DIFFERENT DRIVER CHIP AND/OR WIRING COLORS. ANYTHING PURCHASED AFTER AUGUST 2011 SHOULD MATCH THE FOLLOWING DECRIPTION. We initially sold LEDs in this form factor using the LPD6803 driver chip (BEFORE August 2011). If you are using those pixels, please use the 20mm pixel tutorial (www.ladyada.net/products/pixel20mm), using the wiring and code from that page — nothing further here applies. After switching to the WS2801 chip, the first batch used different wire colors, but is functionally the same as described below.

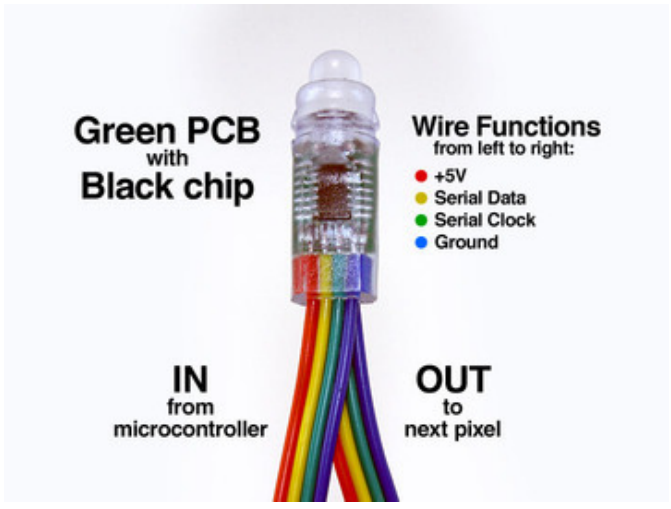
The “magic” of these pixels is that they're digitally controlled...even though there are only two control lines, you can have as many pixels as you’d like in a single long strand, yet each remains independently controllable.

Though it looks like the 4-conductor ribbon cable is continuous, *it isn't!* The pixels have a distinct “in” and “out” side. Data from the microcontroller arrives on the input side, where it’s received by the driver chip. The output side then connects to the input of the next pixel, all the way down the line.

The way to tell input from output is different among the shape and version of the pixels you’re using:



For flat pixels, look for an embossed **arrow** on the side of the plastic capsule. This represents the **direction of data flow**...bits from the microcontroller should arrive at the “tail” of the arrow, *not* the pointy head end.

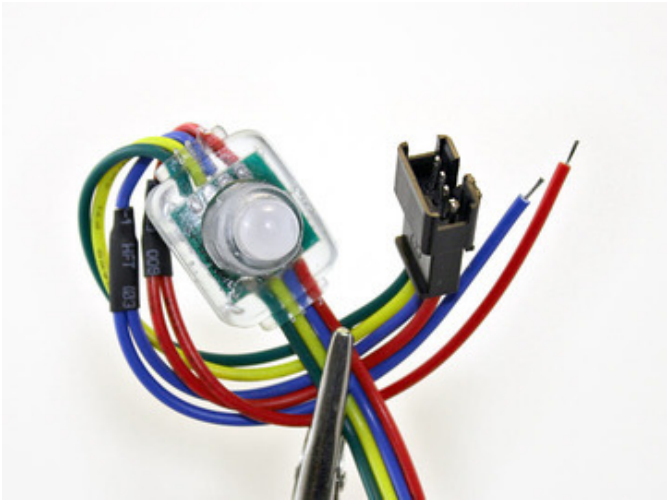


For bullet pixels with a **green circuit board** inside, look for the rectangular black chip on one side of the board. With this chip facing you, the set of **wires in front are the input** — data arrives here — and the wires underneath are the output to the next pixel.

For older bullet pixels with a **white circuit board** inside, look for the "**INPUT**" label printed on the board. With this side facing you, the set of **wires in front are the input** — data arrives here — and the wires underneath are the output to the next pixel.

Our LED pixels now include plugs for joining multiple strands and to simplify connection to a microcontroller:

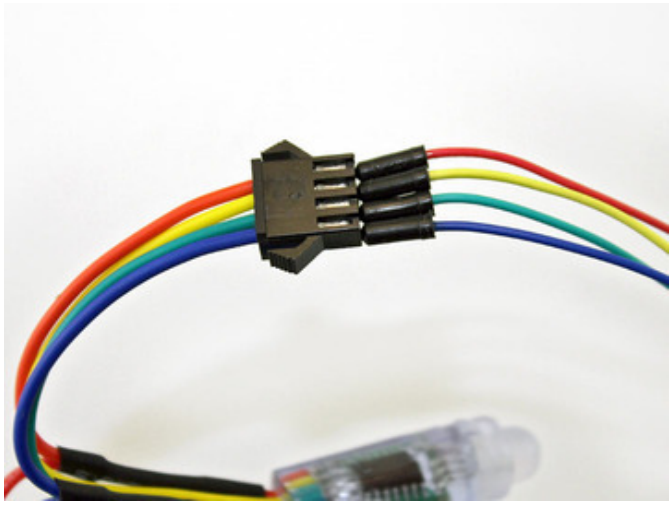
Your pixels will have one end with a plug - these tend to be the input side on **bullet** shaped pixels but we've found that isn't true 100% of the time so don't rely on this as a way of determine which end is the input



Your pixels will have one end with a receptacle - these tend to be the input side on **flat** shaped pixels but we've found that isn't true 100% of the time so don't rely on this as a way of determine which end is the input

Mating connectors for both are available in the Adafruit shop. These have a housing at one end and four tinned wires at the opposite end...the wires can be soldered directly to your circuit, or strip a little more insulation to press these into Arduino pin sockets.

Luckily, we have them as a pair (http://adafru.it/578) so no matter what pixels you end up with, you'll have one for input and one for output



There's one more trick up our sleeve: if you don't have a mating connector available, breadboarding jumper wires can be pressed into the connector - here we're **assuming that is the input side!**

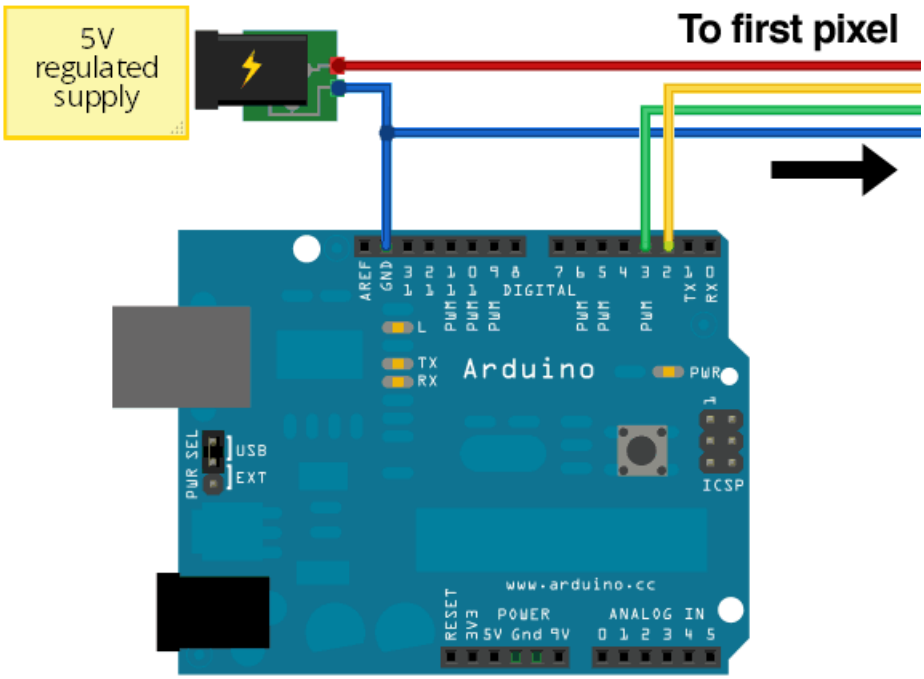
The extra red and blue wires (the exposed ones not inside a connector) are for power — we'll explain that in the next section.

Connecting to Arduino

To use our example code for Arduino, connect the **yellow** wire (serial data) to Arduino **pin 2** and the **green** wire (serial clock) to **pin 3**. The software can be configured to use other pins, but we recommend using this arrangement when starting out, so that everything is tested in a known configuration. The **blue** wire (ground) should be connected to any of the Arduino **GND** pins.

Why do the bullet and flat pixels use different "genders" for the connecting plugs? Why can't I mix and match strands?

This is intentional, to avoid wiring mishaps. Although the two pixel types have the same electrical specifications and use the same data protocol, the order of the wires is different between the two. Joining them directly would mix signal and power wires improperly, potentially damaging the pixels.


5V
regulated
supply
To first pixel
Arduino
DIGITAL
POWER
ANALOG IN
www.arduino.cc
ICSP
PWR SEL
USB
EXT

Power

When running a lot of LEDs, it's important to keep track of power usage. Individual LEDs don't get very hot or use tons of power, but they add up fast!

Each single 12mm RGB LED pixel can draw up to 60mA from a 5V supply. That means a strand of 25 can use up to 1.5 Amps. That's a peak rate, which assumes that all the LEDs are on at full brightness. If most of the LEDs are kept dim or off (as when animating patterns), the power usage can be 1/3 this or less.

As shown in the previous wiring diagram, connect ground to both your power supply and microcontroller. Then connect the 5V line from the power supply to the red wire on the LED strand.

Never, NEVER connect more than 5 Volts to the pixels! This will permanently damage them!

We suggest a nice switching supply for driving LED pixels:

Our 5 Volt, 2 Amp power supply (http://adafru.it/276) is ideal for one strand of pixels.

For larger projects using multiple strands of pixels, our 5 Volt 10 Amp power supply (http://adafru.it/658) is good for up to 6 strands (150 pixels total).

The female DC power adapter (http://adafru.it/368) mates with either of the above power supplies. Screw terminals clamp down on the power leads at the end of an LED strand, so there's no soldering required.

Note the embossed polarity markings. Connect the **red wire** to the + terminal and the **blue wire** to the - terminal.

Pixel strands have these power wires at *both* ends. You can use *either* side. But don't leave the wires exposed at the opposite end! Insulate them with heat shrink or tape, or trim the exposed wire flush with the insulation.

Tips for powering pixel strands:

- When linking multiple strands together, power should be split and applied to *each* strand. If you try to power too many LEDs from just one end of the strand, they'll start to "brown out" the further they are from the power supply.
- Strands can be powered from *either* end — "input" and "output" doesn't apply to power, only the data signals from the microcontroller.
- If the 10 Amp power supply isn't large enough for your project, a slightly modified ATX computer power supply (https://adafru.it/aHR) can provide 30 Amps to power upwards of 500 pixels!
- Generally speaking, you should not try to power an LED strand from the Arduino's 5V line. This is okay if just a few pixels are lit, but is not adequate for driving a full strand.
- For a standalone application (not USB connected to a computer), you can power the Arduino from the same regulated 5V supply as the LEDs — connect to the 5V pin on the Arduino, *not* Vin, and don't use the DC jack on the Arduino.
- Remember to insulate or trim any unused, exposed power wires!

Code

Installation

Our Arduino library takes care of all the communication and color conversion required to run these pixels. You can find our Arduino library source repository at the repository on GitHub (https://adafru.it/aHV).

To install it, click the button below to download the compressed ZIP file then install it. This guide (https://adafru.it/aYM) will help you with the install process if you have never installed an Arduino library.

https://adafru.it/cO6

https://adafru.it/cO6

After installing the Adafruit_WS2801 library, restart the Arduino IDE. You should now be able to access the sample code by navigating through menus in this order: File→Sketchbook→Libraries→Adafruit_WS2801→strandtest

Using the Library

Let's look through the strandtest example code. To use the library in an Arduino sketch, you'll first need to globally declare a WS2801 object to talk to the strip. It is invoked with three variables: the number of pixels and the data and clock pins:

```
int dataPin = 2;
int clockPin = 3;

// Set the first variable to the NUMBER of pixels. 25 = 25 pixels in a row
Adafruit_WS2801 strip = Adafruit_WS2801(25, dataPin, clockPin);
```

Next, we initialize the strip in the setup() procedure:

```
void setup() {

  strip.begin();

  // Update LED contents, to start they are all 'off'
  strip.show();
}
```

begin() initializes the library, while show() refreshes the displayed colors of the LEDs. You'll need to call show() after changing any pixel colors to see this reflected in the LEDs. This way you can change the entire strip at one time (it takes the same amount of time to change one pixel as it does for an entire strip).

Let's look inside an example function, colorWipe(). This creates a 'chase' sequence that fills the strip with a color. It is basically a loop that increments through every pixel (which you can query with the numPixels() function) and sets the color of each (incremented with i) to the value passed (c — colors are expressed as a 32-bit variable type, though only the bottom 24 bits are used). The strip output is then updated with show(). Finally there is some delay (otherwise this would happen instantly).

Below that is a helper function that converts a color from separate 8-bit red, green and blue values into a combined 24-bit value (suitable for passing to colorWipe()). The brightness range is from 0 (off) to 255 (max brightness).

```
// fill the dots one after the other with said color
// good for testing purposes
void colorWipe(uint32_t c, uint8_t wait) {
  int i;

  for (i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
  }
}

/* Helper functions */
// Create a 24 bit color value from R,G,B
uint32_t Color(byte r, byte g, byte b)
{
  uint32_t c;
  c = r;
  c <<= 8;
  c |= g;
  c <<= 8;
  c |= b;
  return c;
}
```

For example, in the loop() function we call colorWipe(Color(255, 0, 0), 50) which will fill the strand with full-brightness red light, pausing about 50 milliseconds between pixels.

```
  colorWipe(Color(255, 0, 0), 50);  // red fill
  colorWipe(Color(0, 255, 0), 50);  // green fill
  colorWipe(Color(0, 0, 255), 50);  // blue fill
```

Troubleshooting

How do I install the LEDs (physically)?

It's pretty easy! Simply drill a 12mm hole into any material up to 1.5mm/0.06" thick. Then push the LED bulb first into the hole. It takes a little wiggling but there are four flanges molded in so that you can 'push' them thru and the flanges will keep the LED pixel in place

Many support issues arise from eager users getting ahead of themselves, changing the code and wiring before confirming that all the pieces work in the standard configuration. We recommend always starting out with the examples as shown. Use the pinouts and wiring exactly as in the tutorial, and run the stock, unmodified “strandtest” example sketch. Only then should you start switching things around.

Here are the most common issues and solutions...

The pixels are wired and powered exactly as in the tutorial, the sketch compiles and uploads successfully, but nothing happens.

- Double-check all wiring. Are the clock and data wires swapped? Is ground connected to the Arduino?
- Confirm the Arduino is connected to the input end of the strand.
- Check power supply polarity and voltage. Are + and – swapped? If you have a multimeter, confirm 5V DC output (±10%) from the power supply.
- Are the power wires at the opposite end of the strand insulated or trimmed? They should not be left exposed where they might make contact with metal, or each other.
- Is the correct board type selected in the Arduino Tools→Board menu?

A few LEDs randomly turn on when power is applied, but then nothing happens.

The power supply is probably OK. Check for any of the following:

- Double-check all wiring. Are the clock and data wires swapped? Is ground connected to the Arduino?
- Confirm the Arduino is connected to the input end of the strand.
- Is the correct board type selected in the Arduino Tools→Board menu?
- Did the strandtest code successfully compile and upload?

Only the first few LEDs respond. The rest remain off or flicker randomly.

- Confirm that the number of LEDs in the Adafruit_WS2801() constructor match the number of LEDs in the strand (both will be 25 if using the strandtest example and a single strand of LEDs).
- Inside each pixel there’s a small circuit board. Give the first bad pixel (and the one immediately before it) a firm squeeze where the ribbon cable joins the board — it may simply be a dodgy connection. If that works, you can either cut out the offending pixel and join the two sub-strands, or arrange for a replacement strand if new.

The LEDs flicker randomly, not the way they’re programmed to.

Are the clock and data wires swapped? Is ground connected to the Arduino?

“White” LEDs are showing pink instead.

- This can happen when trying to power too long of a strand from one end. Voltage will drop along the length of the strand and the furthest pixels will “brown out.” Connect power to *every* 25 pixel strand.

Sketch will not compile: error message is “Adafruit_WS2801 does not name a type”

- Confirm the library is unzipped prior to installation.
- Confirm the library is properly named and located. The folder should be called Adafruit_WS2801, and placed

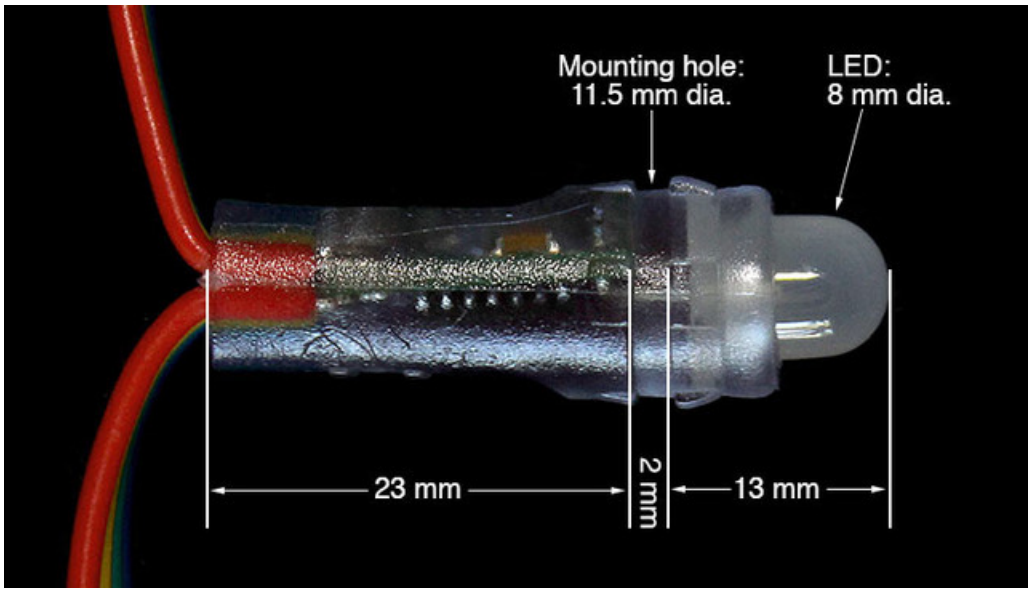
inside your personal Documents/Arduino/Libraries folder — *not* inside the Arduino application folder!
- After installation, the Arduino IDE needs to be restarted for new libraries to be used.

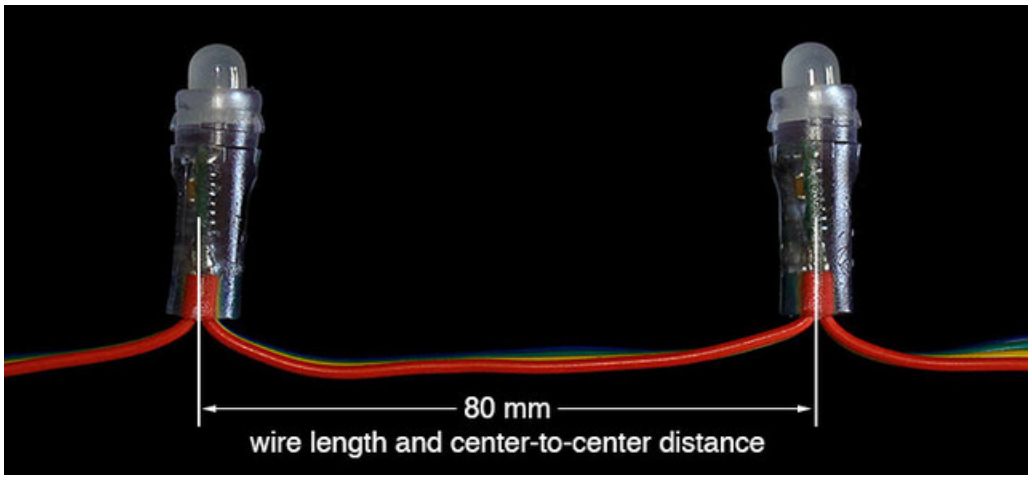
Dimensions

There's a fair amount of variance between pixels due to the rubbery, waterproof coating. The figures given here are just guidelines, rounded up to the next whole millimeter (except for the mounting hole diameter). Allow yourself some extra "wiggle room" for spacing when planning an installation.

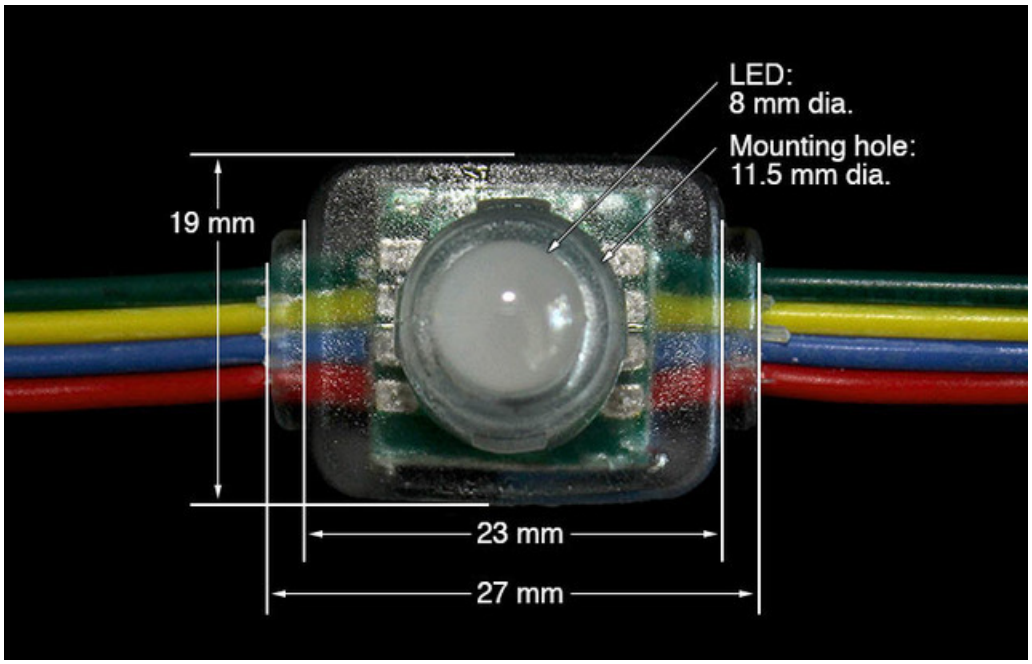
For mounting holes, 11.5 millimeters is a good guideline, or might go as narrow as 11 mm for a really firm grip, though at that size pixels will be hard to remove and the mounting flange may take some abuse in the process.

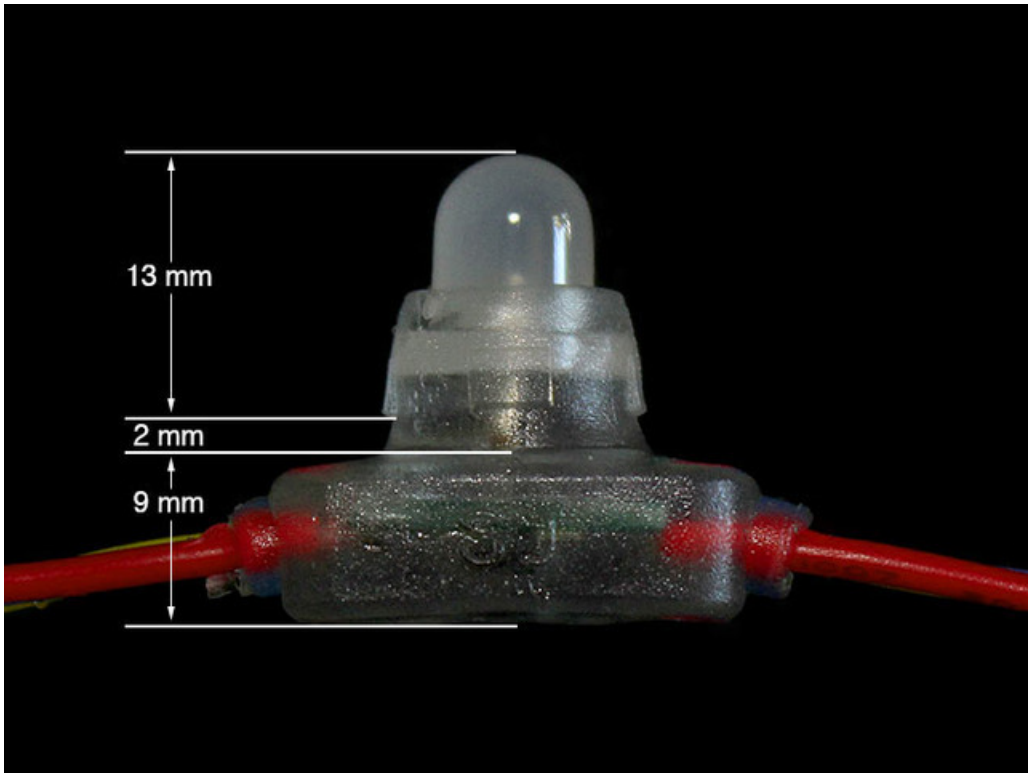
"Bullet" Pixels

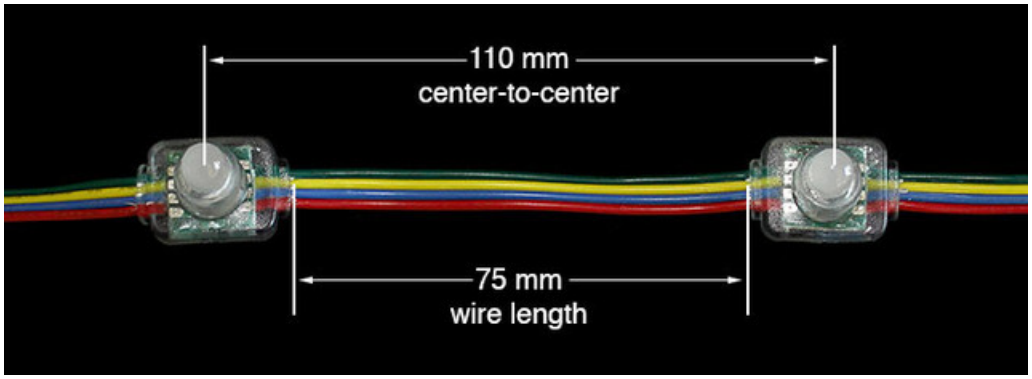




"Square" Pixels


LED:
8 mm dia.
Mounting hole:
11.5 mm dia.
19 mm
23 mm
27 mm


13 mm
2 mm
9 mm


110 mm
center-to-center
75 mm
wire length