


Arduino + Max/MSP/Jitter = Maxuino

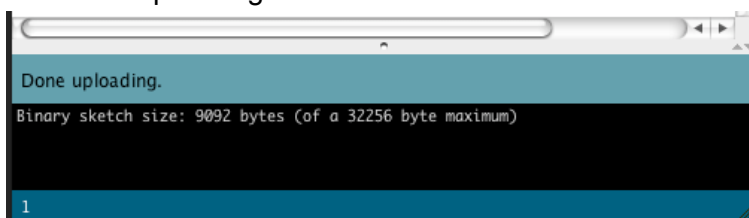
Maxuino is a set of components that allows Max/MSP/Jitter (MMJ) to communicate with an Arduino microcontroller. This gives MMJ access to the Arduino's pins, allowing it to read from the digital and analog pins to sense physical input and to write to Arduino's digital pins to turn things off and on (and, using pulse width modulation, put out analog voltages).

Install the necessary software

Setup takes a few steps. The process is outlined here, and then I'll give details. This assumes that you already have Max/MSP/Jitter installed.

1. [Download and install the Arduino software](#)
 2. [Download and install Maxuino](#)
 3. [Download and instal CNMAT's huge MMJ package](#)
-

- Download and install the Arduino software.
 - Go to <http://www.arduino.cc/> and click on the "Downloads" link at the top.
 - Download the appropriate version for your operating system.
 - Install the Arduino Software, including the drivers if you have an older (Deumilanove or Decimella) board. Be sure to follow all of the instructions exactly as written in the following guides
 - For Windows, use this guide: <http://arduino.cc/en/Guide/Windows>
 - For Mac, see use guide: <http://arduino.cc/en/Guide/MacOSX>
 - Once the software is loaded, open the Blink example as described in the above linked guides. Insert an LED with its longer pin going into the Arduino's digital pin 13 and the shorter pin going into GND. Upload the Blink program to the board by clicking on the upload button  or pressing command-U (Mac) or control-U (Windows).
 - If the LED starts blinking, then Arduino is functioning properly.
 - Now we can upload Firmata to the Arduino board (Firmata is a program that makes Arduino's pins accessible to other applications, like Max/MSP/Jitter, Processing, etc.)
 - Select File--> Examples -->Firmata -->Standard Firmata
 - Upload this program to the Arduino board. (Remember, you have to have already selected the proper Serial Port and board type.) Look for the blinking tiny LEDs on the board. That's a good sign. At the bottom of the Arduino window, watch for "Done uploading."



- Download and install Maxuino.
 - Go to <http://www.maxuino.org> and click on Downloads.
 - Download the release (currently as of October 2011, v.009) and unzip it.
 - Make a new folder called Maxuino and copy into it the maxuino.maxpat file and the

- entire “support” folder from the Maxuino download.
- Open a Finder or Explorer window and navigate to the Max5 folder (inside of the Applications folder in Mac OSX). Inside that folder, go into the “patches” folder. Drag the Maxuino folder that you just made into the “patches” folder.
 - Mac OSX will ask you to authenticate in order to change the contents of a folder inside of Applications. Just enter your User and Password info.
 - Download and install CNMAT’s huge MMJ package. (CNMAT is the Center for New Music and Audio Technology at the University of California - Berkeley. This collection of Max bits has lots of useful parts. We’ll install everything in case parts of it are useful in the future for our work.)
 - Go to <http://cnmat.berkeley.edu/downloads>.
 - Download the appropriate “Everything” package for your operating system.
 - Unzip the folder that downloads.
 - Place that entire folder into the same “patches” folder where you put the Maxuino files above.
 - You’ll have to authenticate again here.
-

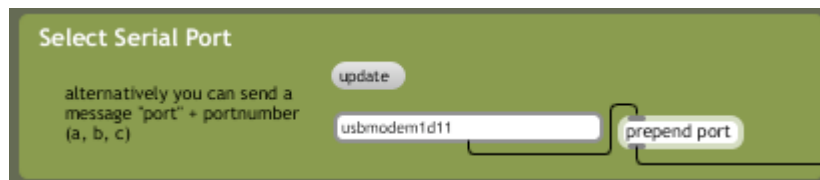
Getting Started with Maxuino in Max

From the Maxuino download folder, open the patch called maxuino.help.maxpat. This Max patch walks us through how to access and use the Arduino pins.

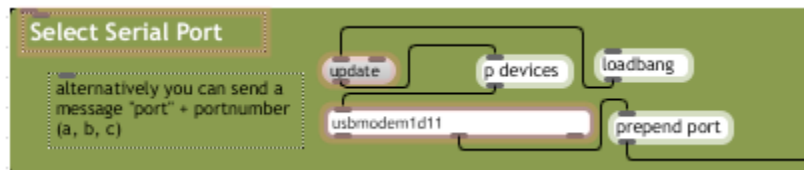
This patch is built around two custom objects. The first is the [maxuino] object, which does the actual communication with the Arduino. The second is the [maxuino-gui] object, which gives us an interface to the Arduino’s pins. (“gui” is short for graphical user interface.) You can double-click on either of them to see what’s inside.

The [maxuino-gui] object has 54 digital pins represented (it starts at 0, not 1, so they’re numbered from 0 to 53), which is far more than your Arduino Uno boards have. This is because another Arduino, the Mega, has that many digital pins. Likewise, the Uno only has 6 analog pins, but [maxuino-gui] has 16 so it can interface with the Mega also. We’ll only use as many pins as our boards have.

1. [Set the proper Serial Port](#)
 2. [Set the pin mode for the pins we’ll be using](#)
 3. Digital out
 4. Digital in
 5. Analog in
 6. PWM (pulse width modulation)
- Set the proper Serial Port.
 - Max needs to know which port to communicate through. The first part of maxuino.help.maxpat allows you to select the serial port.




Click on the “update” button to see a current list of attached devices. Then you can click on the drop-down list to select the one that starts with usbmodem... Notice that the word “port” is prepended to the name of the port and this is sent directly to the right inlet of the [maxuino] object. To better understand what is going on here, we can unlock the patcher to see hidden objects:



There is a custom patcher object called “devices” which loads the name of all of the devices attached to the computer. (You can open that patcher to see what’s in it, but for now we can just use it and copy/paste it if we need to use it.)

- **Set the pin modes**

Once the correct Serial Port is selected we can initialize the Arduino's pins for what we want them to do. In order to use a pin, we have to set its "mode," which determines whether it will be a digital in, digital out, analog in, pwm (pulse width modulation, which approximates an analog out), or servo. The digital pins (0-13 on the Arduino Uno) can be digital in, digital out, or PWM. The analog pins can only be analog in. (We'll talk about servos later.)

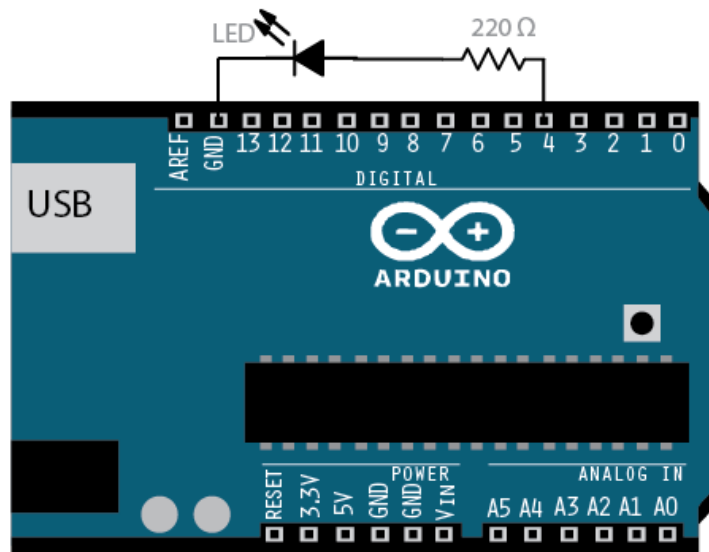
- Look in the Set Pin Mode section of the patch. This patch uses a [sprintf] object and [prepend] combined with a radiogroup and a number box to generate a message. What we care about is the message that is generated at the bottom left. To set the pin mode, make a message. Put the pin number between forward-slashes and then type "mode" and a number between 0 and 4. 0=Digital in, 1=Digital out, 2=Analog in, 3=PWM out, and 4=Servo out. They used radio buttons in this patch to select a number, but you can do it directly with a number in your message object or with a number box.  This sets pin 4 as a digital in. (/8/mode 1) would set pin 8 as a digital out.
- Maxuino automatically recognizes if you're setting a pin as an analog in and applies that to the appropriate analog pin. So if I send it a message (/3/mode 2) it knows that digital pin 3 can't be an analog in, so it will set analog pin 3 as an analog in (which, for an analog pin, turns it on so it's sending out its input). On the other hand, if I send it a message (/3/mode 0) it knows that I can only be referring to digital pin 3, so it sets it as a digital in.

- **Digital out**

When a pin is set as a digital out, it can be turned on (+5 volts) or off (0 volts). This can be used directly, such as to turn on or off a LED, or to control a relay, which can switch on and off much bigger things.

- Look in the Digital Output section of the maxuino.help patcher. (or if you're looking at an older version, double click on the patcher object)
- As with setting pin modes, they're using a few objects to generate the message. In the message box you see the syntax for writing to a digital pin. The message has the pin number between forward slashes, followed by "digitalWrite", then a space and either 0 (off, or low) or 1 (on, or high). So, the message to set pin 3 high (on) would be (/3/digitalWrite 1) or (/3/digOut 1).
If you use a \$1 you can turn on and off the digital outs using the toggles. NOTE: In some instances (maybe older versions of Maxuino? I'm not sure) I have found that "digitalWrite" doesn't work and instead you need to use "digOut". To see that this is working, we can make a circuit like this:

Circuit for blinking an LED

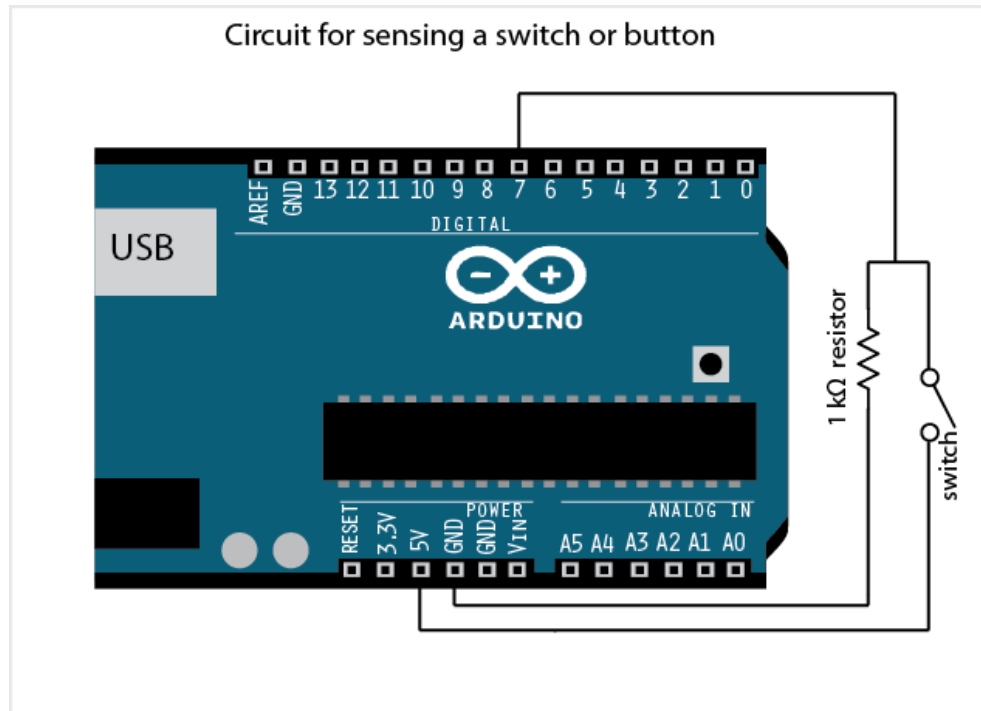


If you connect it to the Digital Out pin that you're controlling, you should see the LED turn on and off as you click the toggle.

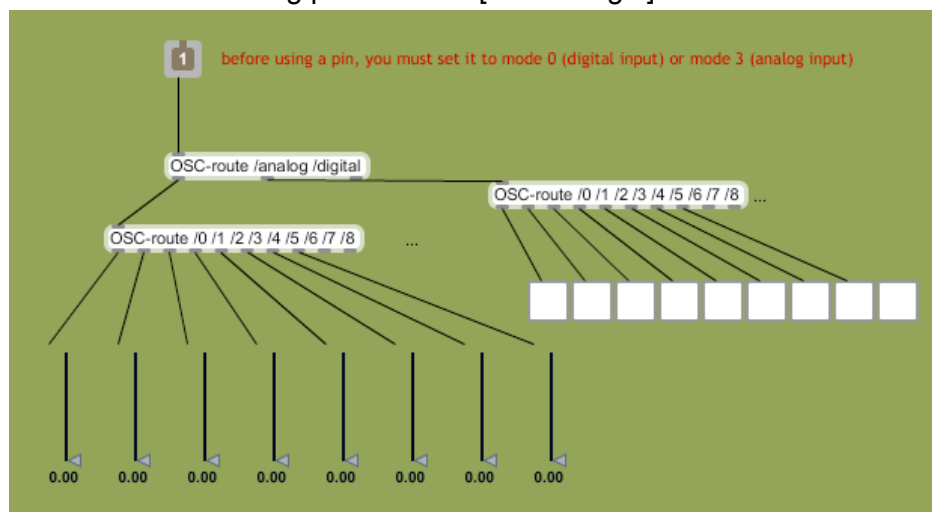
- **Analog and Digital In**

Double-click on the Analog-Digital-Inputs patcher object or look in the section of the patcher that says “Analog/Digital In”. In order to read the Arduino’s pins, maxuino requires you to use the OSC-route object to route the messages. It’s probably easiest to just copy and paste from the help patch:

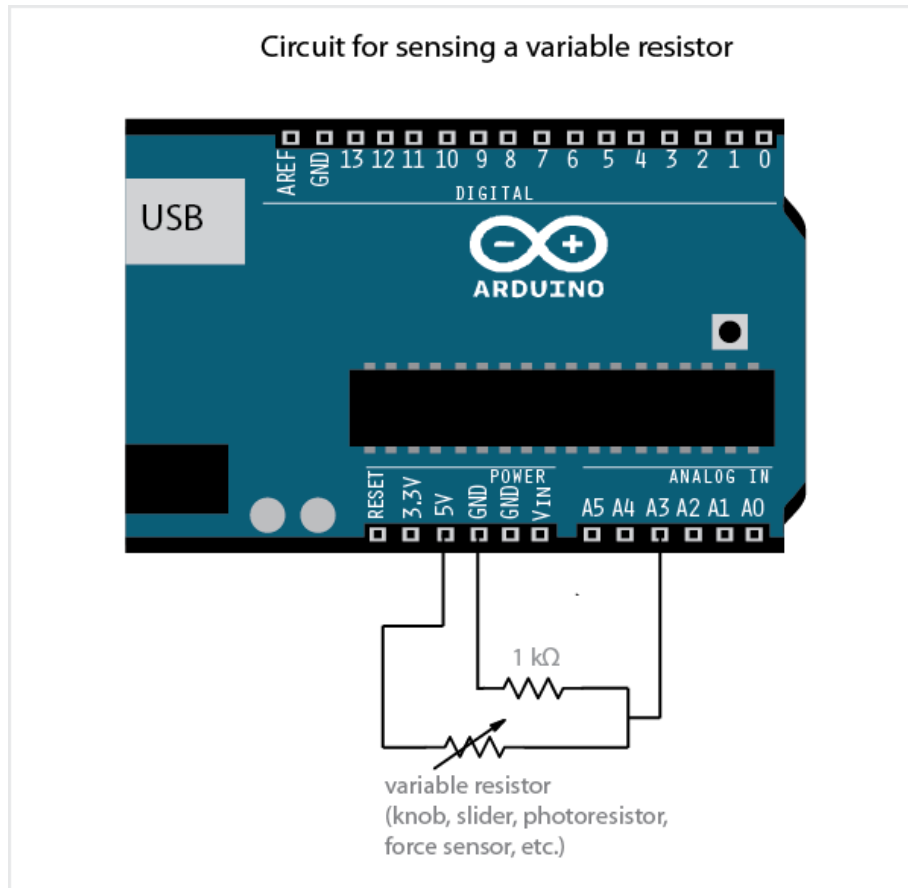
- **Digital in** checks to see if there is or isn’t voltage at the pin. Put another way, digital in lets us check a switch, button, etc. to see if it’s off or on.
- Just as with digital out, we need to set the pin mode, but this time to digital in.
- The circuit is a bit more complex for this:



- As you flip the switch or push the button the toggle for that pin will turn on and off.
- **Analog in**
The Analog-Digital-Inputs patcher also has analog ins. Analog ins measure how much voltage is coming in at a pin (0 to 5 volts).
 - Turn an analog pin on in the [maxuino-gui].



- Then, build this circuit and connect it to whatever pin you have set as an analog in.



- You should see the slider object for that pin moving up and down with the values coming from your variable resistor.