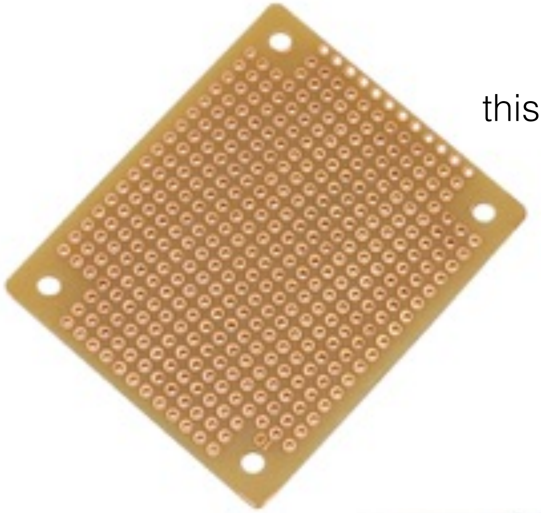
# Part list for your node

these are “barrel jacks”
we will use them to connect
power (in) and solenoid (out).

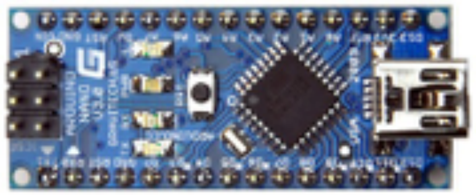


this is a “perforated board”
where we will fix our
project together.

this is a capacitor
we will use it to make
the radio less “noisy”.
the white stripe is  “negative” or ground.

on the left, a red LED light
used to indicate data traffic
we will use it together with
the resistor bellow.

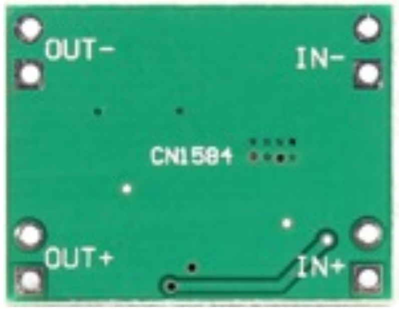
this is an Arduino nano,
our central processor.



this is a “radio module” we will
connect it to the Arduino to
talk to the node wirelessly.

(top)

this is a power converter
it transforms 12v inputs
to 3.3v used by the Arduino.
bottom side shows
how to connect it.

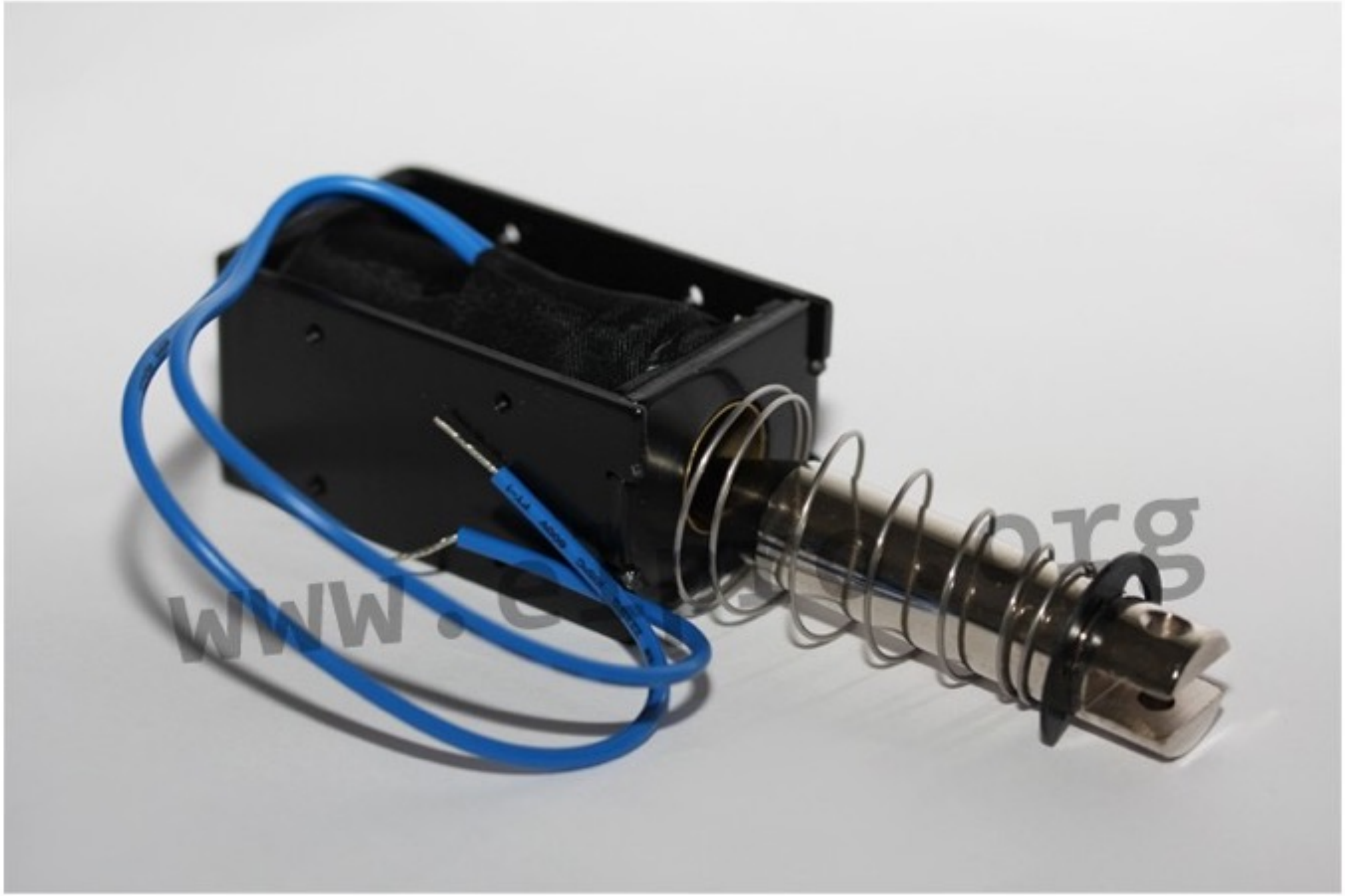


(bottom)

this is a “relay”. you can think
of it as a switch. we will use it to
switch the solenoid on and off.

# Part list for your node (cont.)

This is a “solenoid”, it will be our **“actuator”**.
This means that it is the part that will let our node interact with the physical world.



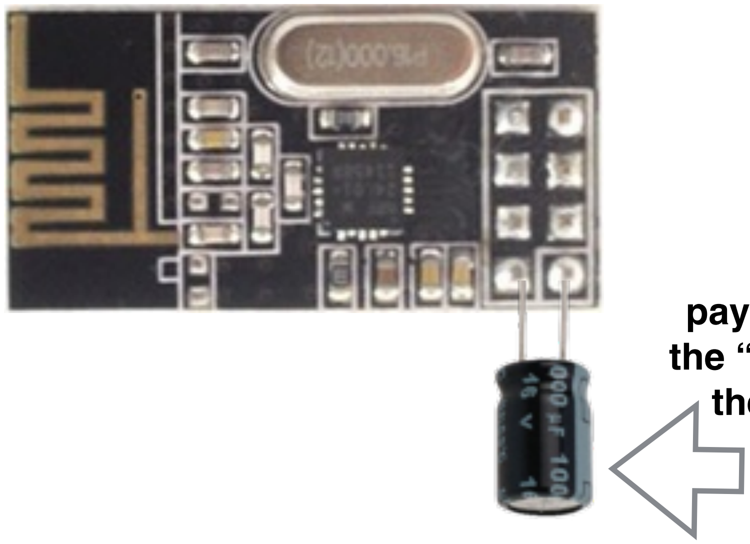
There are other “actuators” that you can attach to your node depending on the kind of interaction you want to have with the outside world. For this workshop we will use a **solenoid**.

# Preparing the radio module

1.

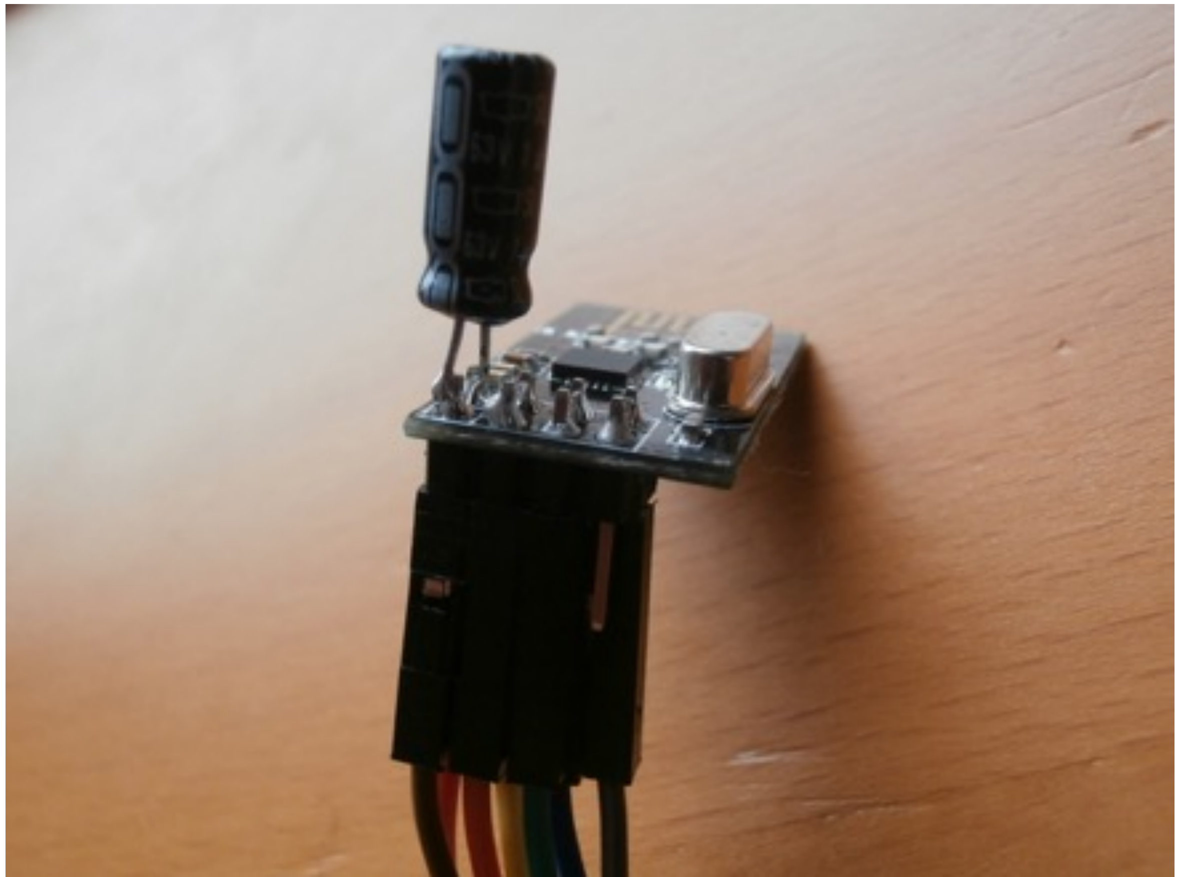
clip legs to same length
to prepare for soldering

2.



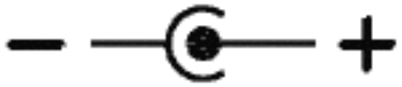
**pay attention to the white stripe on the “-“ side of the capacitor. It marks the leg that will be connected to ground (pin 1).**

3. Solder the capacitor carefully,
making a bridge between pins 1 and 2.



# Preparing the power circuit

notice: every time we use a pin connector we will use the convention “center positive”.

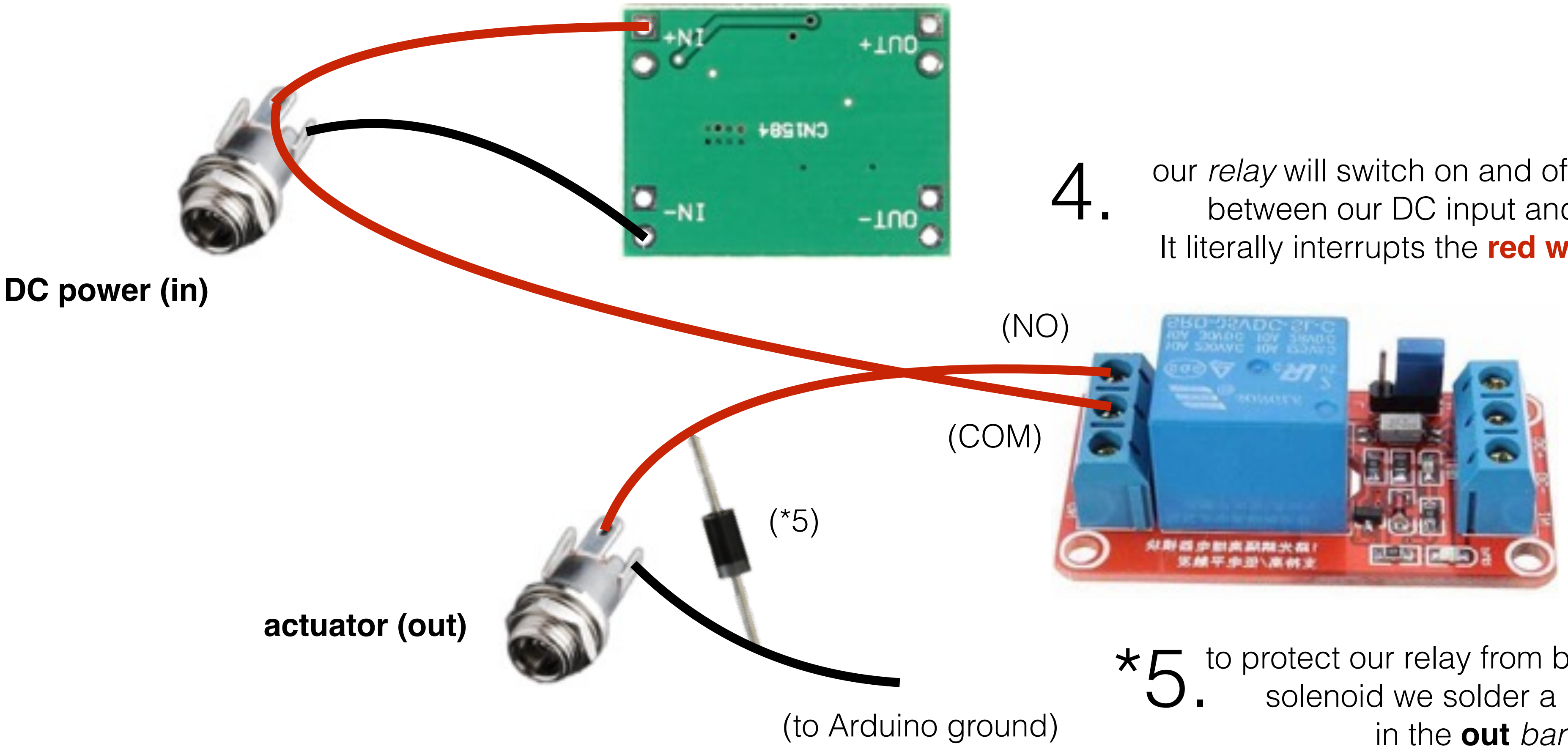


1. use a multimeter to **check** which pin is which in your barrel connector.

2. solder two wires. a **red** one going from the center pin of your barrel connector to the **IN+** of the transformer. And a **black wire** going from the sleeve to the **IN-**.

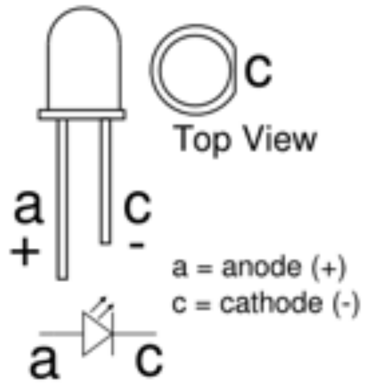
3. Observe how the barrel connectors are wired. Solder a second **red wire** to the *barrel connector*, this will go to the *relay* on step 4.

4. our *relay* will switch on and off the flow of power between our DC input and our solenoid. It literally interrupts the **red wire** from **in** to **out**.



*5. to protect our relay from big current spikes from the solenoid we solder a *diode* across the pins in the **out** *barrel connector*.

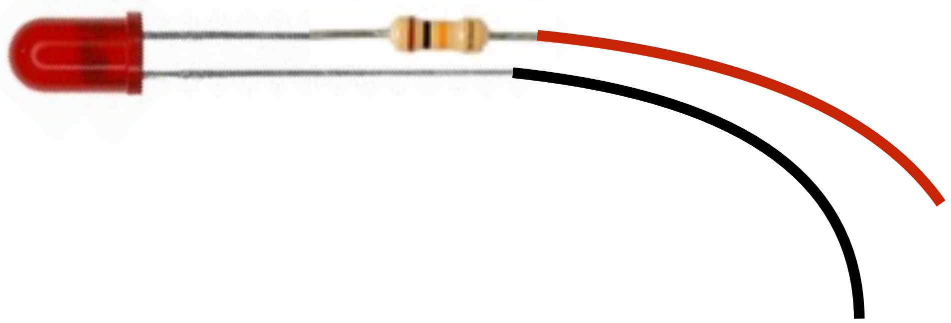
# Preparing the indicator LED



1. cut the legs of your LED to be of equal size

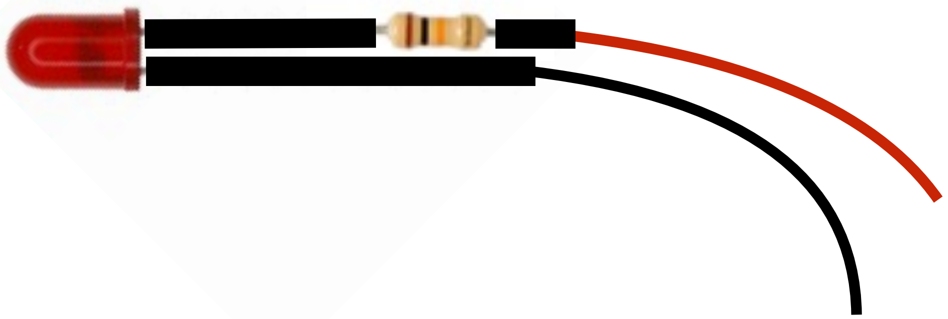
2. solder the 10K resistor to one of the legs of the LED. Let’s attach it to the **+ leg** of our LED.

3. prepare a few slices of ***shrink tubing*** to insulate the legs of your LED.



4. solder two wires that we will use later to connect to the Arduino. The **red wire** goes to the **positive (+)** leg of the LED. The **black wire** goes to the **ground (-)**.

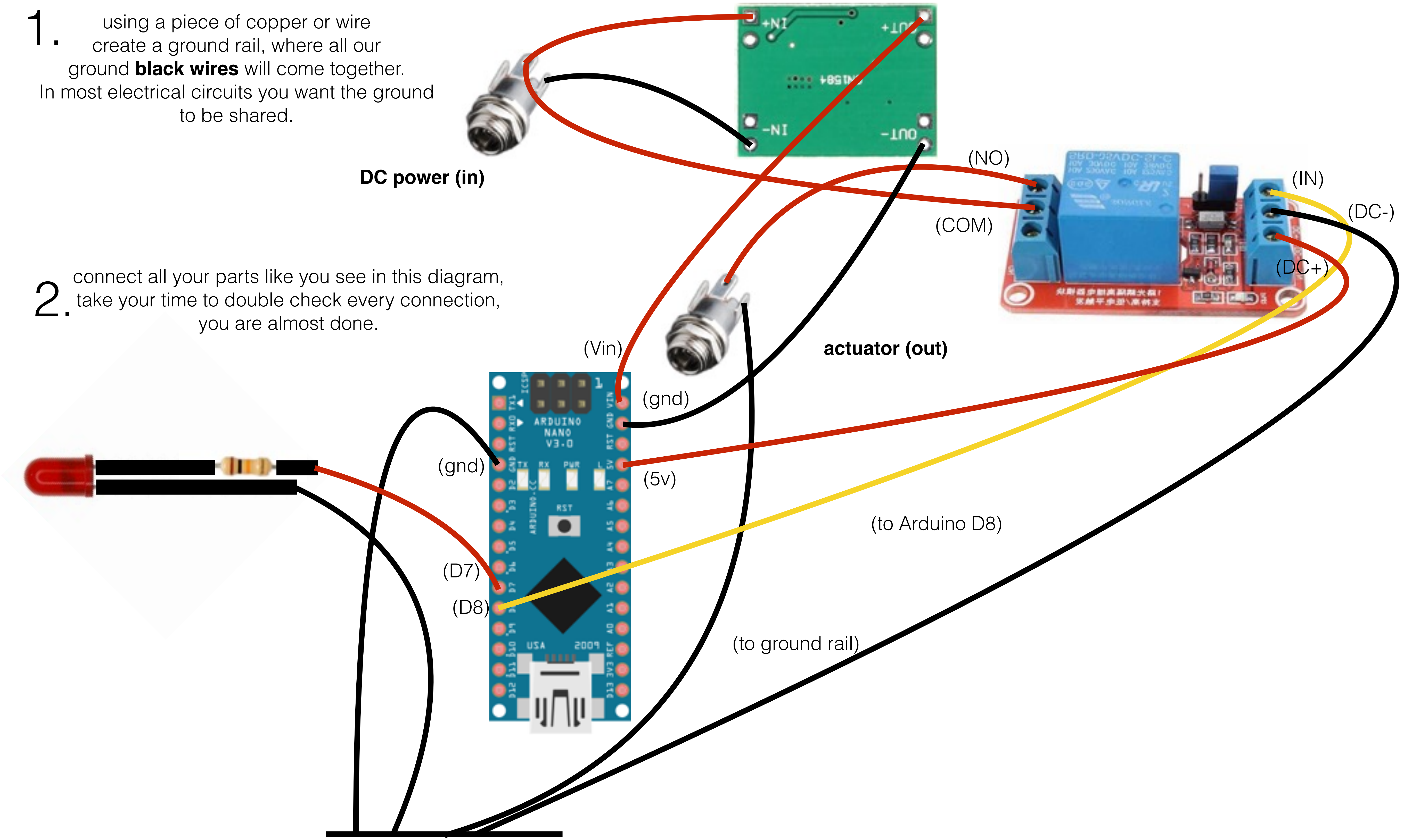
(with shrink tubing protection)



# Connecting your parts to the Arduino

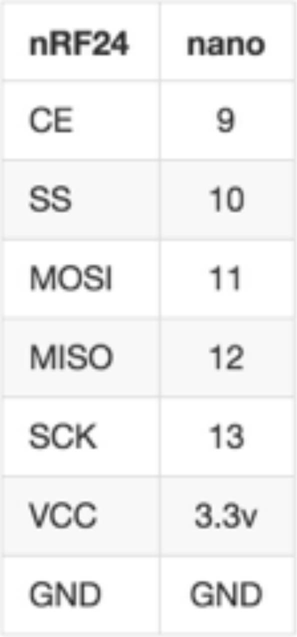
1. using a piece of copper or wire create a ground rail, where all our ground **black wires** will come together. In most electrical circuits you want the ground to be shared.

2. connect all your parts like you see in this diagram, take your time to double check every connection, you are almost done.

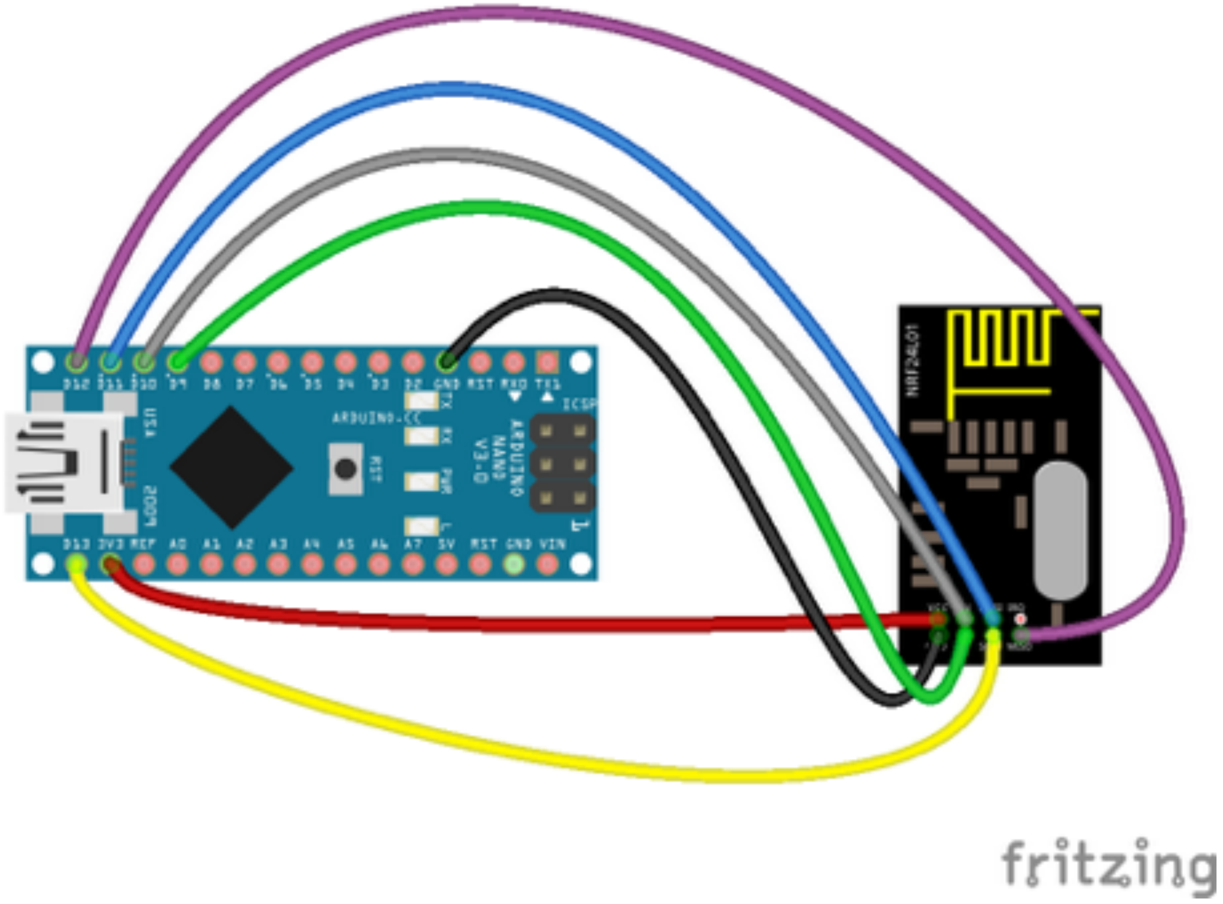


(ground rail, where all the grounds come together)

# Connecting the radio module

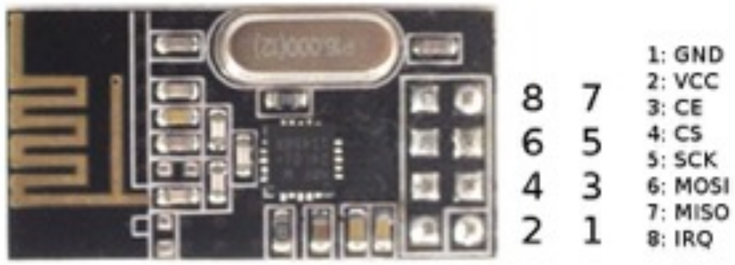


| nRF24 | nano |
|---|---|
| CE | 9 |
| SS | 10 |
| MOSI | 11 |
| MISO | 12 |
| SCK | 13 |
| VCC | 3.3v |
| GND | GND |



1. use jumper wires to form the connections from the radio module to the Arduino. Follow the table above for pin correspondence. **Double check** every connection it is important that these two parts are connected correctly.

## for your reference:
pin layout of radio module

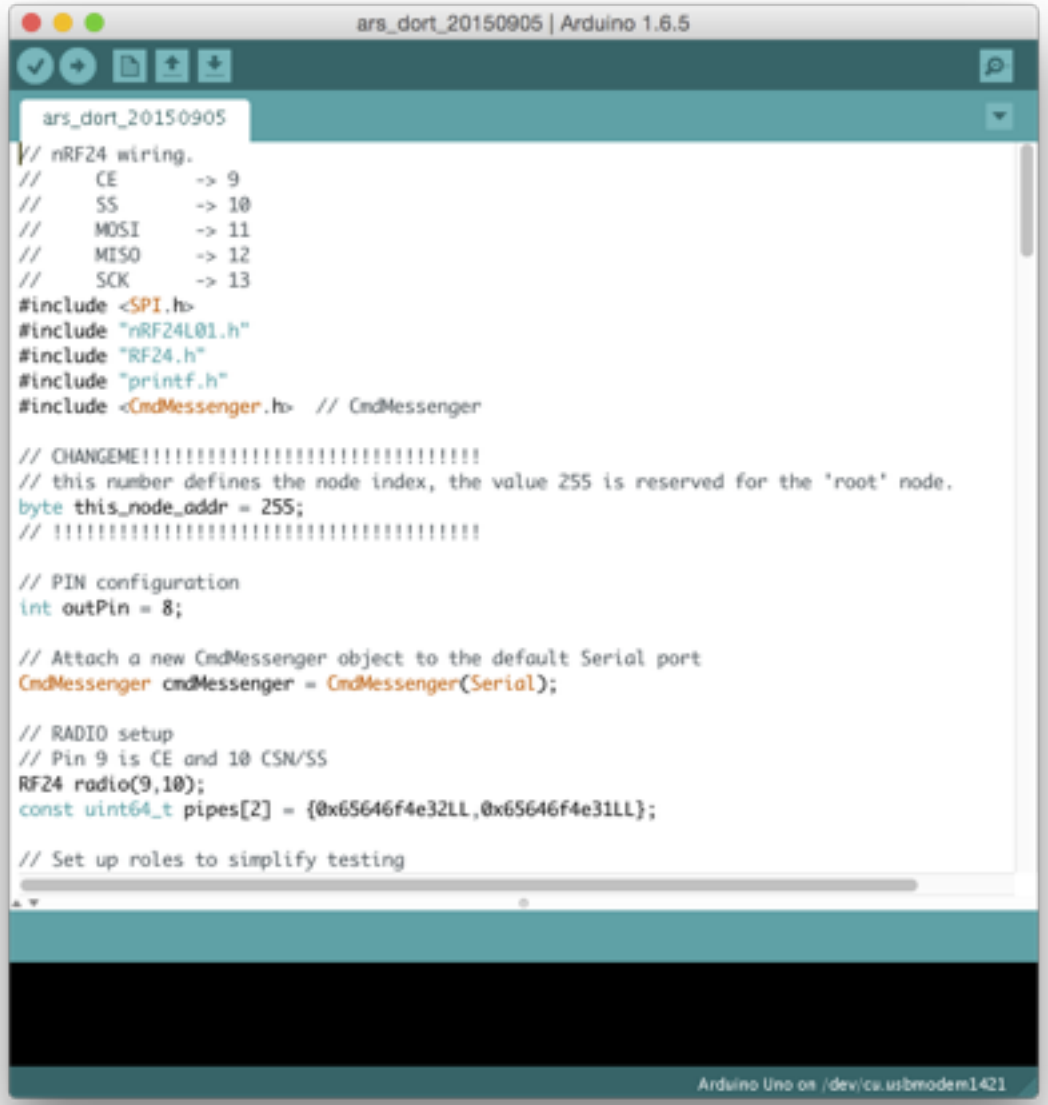


# Loading the firmware into your Arduino

1. Make sure you have the Arduino IDE installed:
http://www.arduino.cc

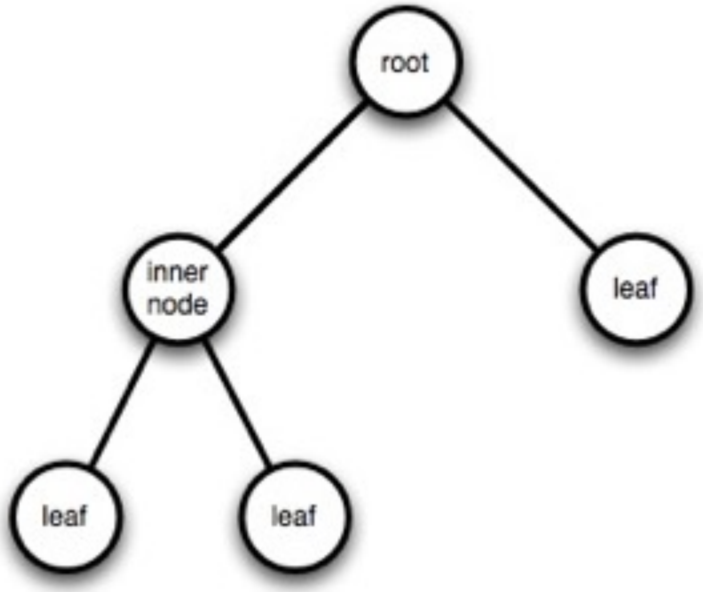
2. Get the software from our Github project:
https://github.com/AR-S/Nodes/

3. Select a unique number for your node, so that we can address it individually and change the value of the **this_node_addr** variable.



```
// nRF24 wiring.
//      CE       -> 9
//      SS       -> 10
//      MOSI     -> 11
//      MISO     -> 12
//      SCK      -> 13
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"
#include <CmdMessenger.h>  // CmdMessenger

// CHANGEME!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// this number defines the node index, the value 255 is reserved for the 'root' node.
byte this_node_addr = 255;
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

// PIN configuration
int outPin = 8;

// Attach a new CmdMessenger object to the default Serial port
CmdMessenger cmdMessenger = CmdMessenger(Serial);

// RADIO setup
// Pin 9 is CE and 10 CSN/SS
RF24 radio(9,10);
const uint64_t pipes[2] = {0x65646f4e32LL,0x65646f4e31LL};

// Set up roles to simplify testing
```

A concept of systems topology:
our network of nodes can be represented as a kind of tree with a root node, sending messages to all the leaf nodes.

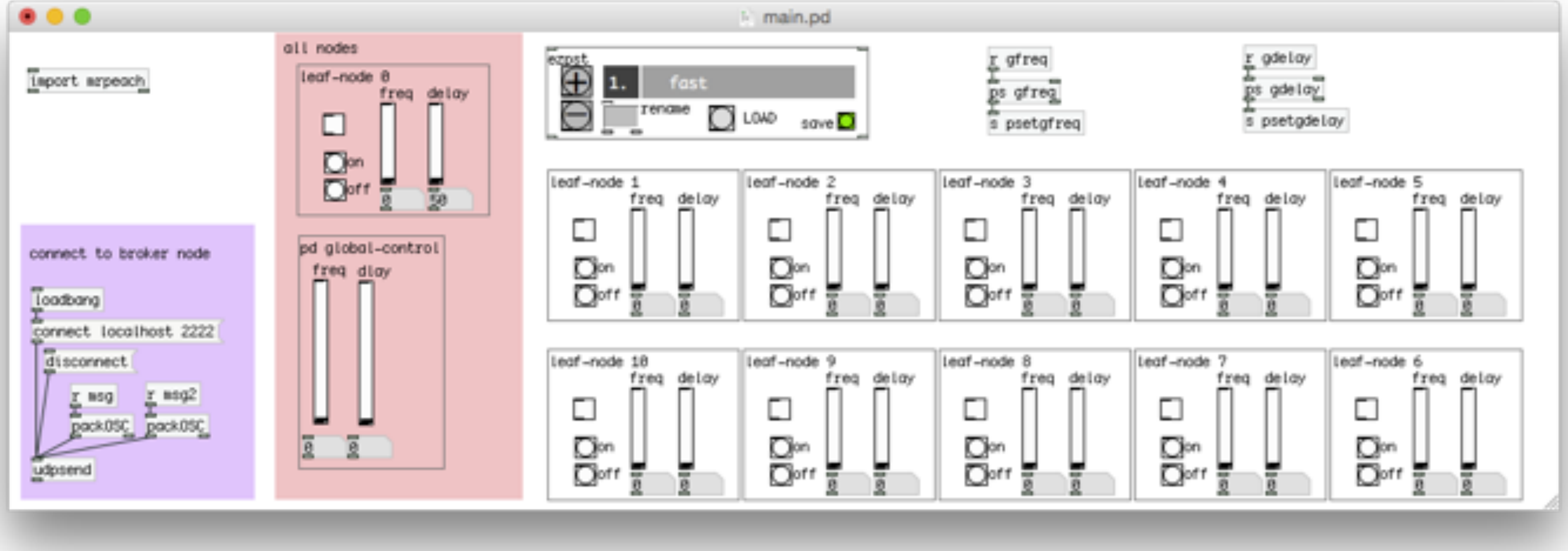


# Testing your node

1. Make sure you have Pure Data installed:
https://puredata.info/

2. Start the broker script **broker.py** from the command line.

3. Load the GUI **.pd file** from Pure Data.


main.pd
import mrpeach
connect to broker node
loadbang
connect localhost 2222
disconnect
r msg
r msg2
packOSC
packOSC
udpsend
all nodes
leaf-node 0
freq
delay
on
off
pd global-control
freq
dlay
ezpst
1.
fast
rename
LOAD
save
r gfreq
ps gfreq
s psetgfreq
r gdelay
ps gdelay
s psetgdelay
leaf-node 1
leaf-node 2
leaf-node 3
leaf-node 4
leaf-node 5
leaf-node 10
leaf-node 9
leaf-node 8
leaf-node 7
leaf-node 6