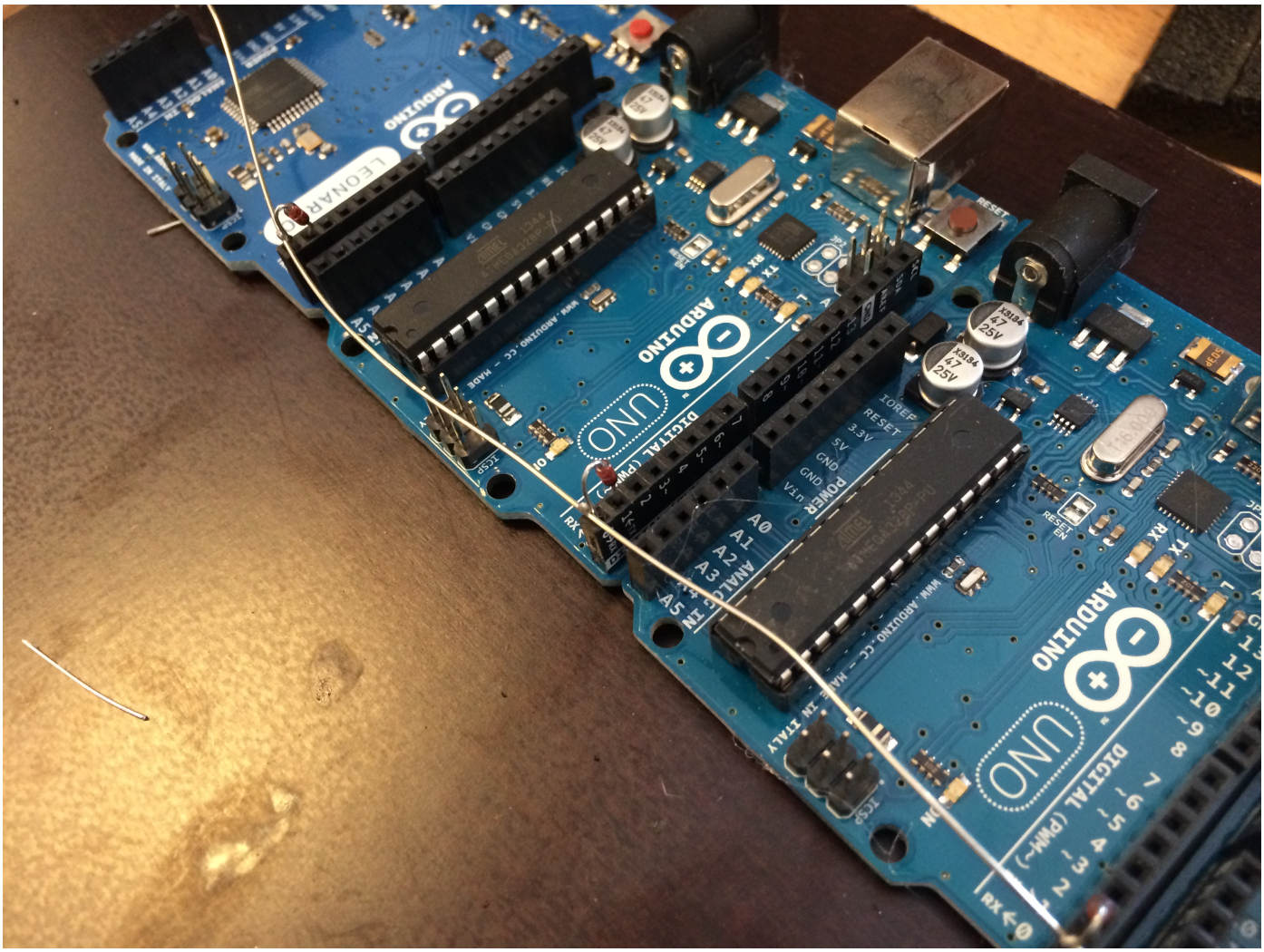
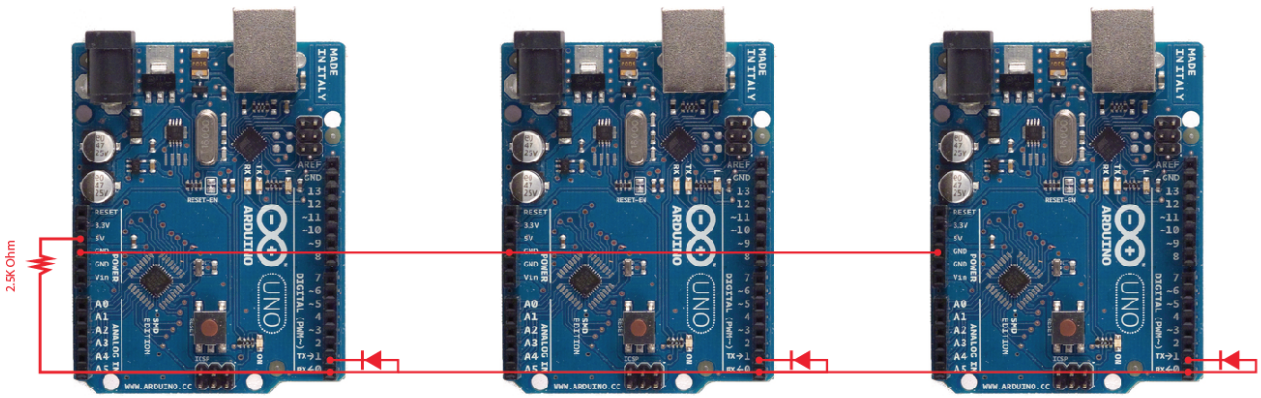


Connecting more than two Arduino boards can be a challenging task. To solve this problem we have created a little solution that simplifies this problem. It makes sure that the different Arduino board will not send data at the same time. It can be considered a simple IP protocol where each node has its own id.

It simply connects all the Serial ports together in a network and one Arduino Board acts as master telling the different slaves when they are allowed to speak.

Wiring



The RX pins on each board should be connected together and connected to 5V with a 2.5k ohm resistor in between. The TX pins on each board should be connected to the same wire with an diode limiting current from TX to the wire.

Remember to connect ground between the Arduinos.

Software

Download the software here (https://github.com/madshobye/multicom)

Copy the folder "easytransfer" into your Arduino library folder (located documents/Arduino/libraries).

Setup the software

Open the example in under file/example/easytransfer/multicom*basic*example

You need to give each board an ID and you need to tell each board the amount of nodes on the network:

```
#define NODEID 2 // the id of current arduino board
#define NUMNODES 2 // inclusive master node
```

One node has to be master:

```
#define MASTERNODEID 0
```

Use the software

The multicom_receive() is called everytime a package has arrived for a specific node:

```
void multicom_receive()
{

}
```

Use the struct mydata to read the value.

The talk is called everytime the currrent node is allowed to communicate on the network:

```
void multicom_talk()
{

  multicom_send(MASTERNODEID,5,testValue++);

}
```

Use multicom*send(ID, PARAM,VALUE) to send data. You can send as much data as you want to. As long as you are within multicom*talk you have reserved the line.

Software serial VS Hardware Serial

The major limitation with this solution is that it reserves the hardware serial to the communication. It is therefore not possible to upload code or communicate with the computer individually. This can be solved in two ways:

Either one can use a Arduino with multiple hardware serial ports. This could be an Leonardo (use Serial1 instead of Serial). Or one can use the software serial library (the communication system i compatible with it).

If you use the latter then you should make sure that you do not have other interrupts running. This happens when you use libraries that are time critical to interface with external hardware (DMX, Accellerometers etc.).