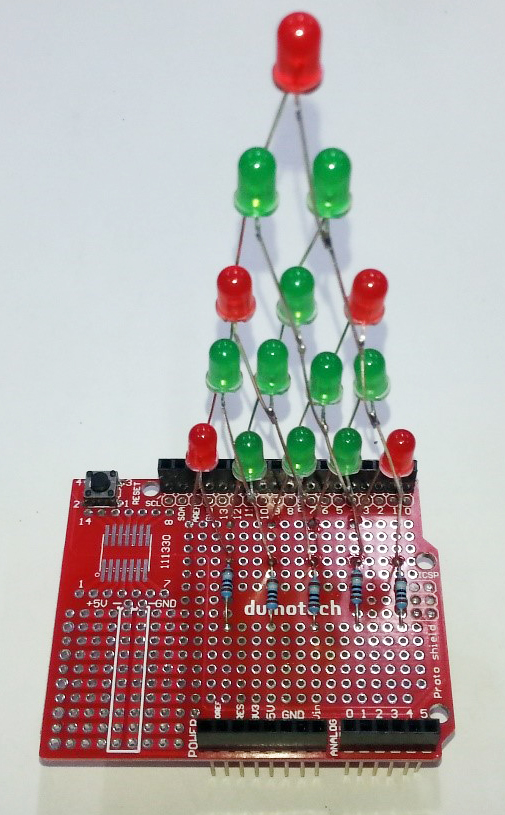
Duinotech LED Christmas Tree

Here’s a good project for a beginner, which can add a bit of Christmas spirit to your workbench, and also teach you how multiple LED’s can be controlled from a handful of microcontroller pins. Being Arduino based, the LED Christmas Tree can also be customized to your liking. There is some soldering required for this project, so make sure you have an adult’s assistance when building the tree. A set of needle-nose pliers will also come in handy.



## Shopping List:

1 x XC4410 Uno Main Board (a Leonardo will work fine too).

1 x XC4482 Prototyping Shield

1 x RR0554 Pack of 8 180 Ohm Resistors

5 x ZD0150 5mm Red LED

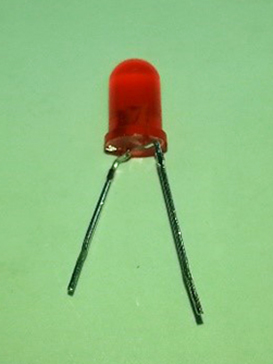
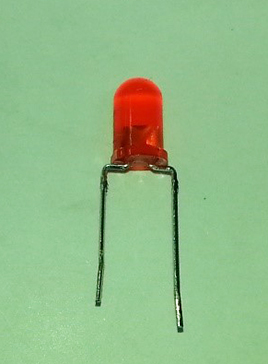
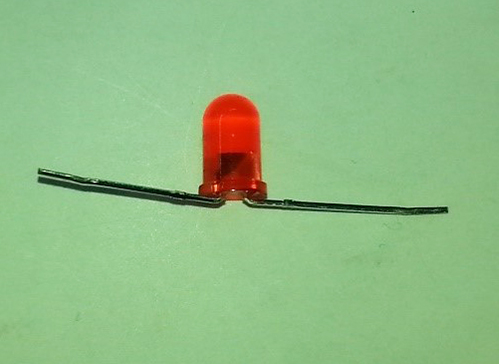
10 x ZD0170 5mm Green LED

You’ll also need about a metre of insulated wire- if you don’t have any spare, our WH3010 will work.

## Construction:

The construction of the tree of LED’s might look a bit tricky, but if you following the steps carefully, you should have no trouble. We’ll start by carefully bending the legs of the LED’s. The legs don’t like being bent and straightened and bent again, in case they break off.

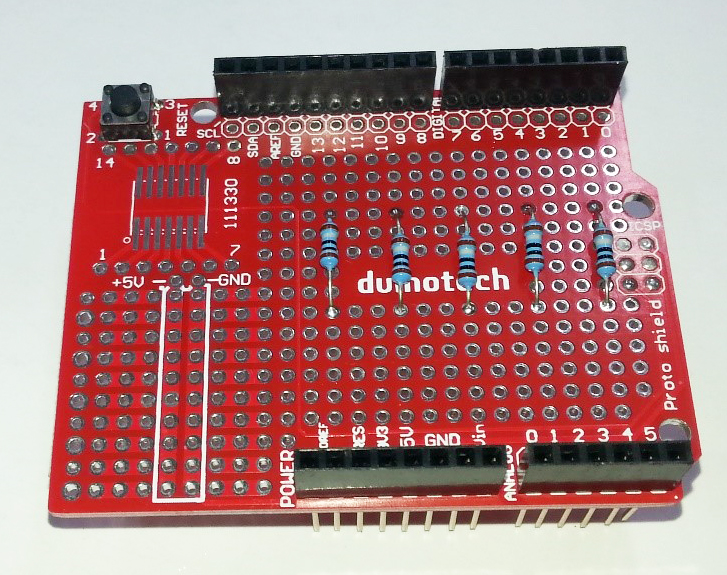
Start by bending the legs 90 degrees so that they point straight out the sides (1), then about 3mm further along the legs, bend the legs straight down (2). We want to make sure that the legs run outside the main body of the LED. Finally, bend the left leg towards you be about 10 degrees and bend the right leg back by 10 degrees. The result should be like (3).



(1) (2) (3)

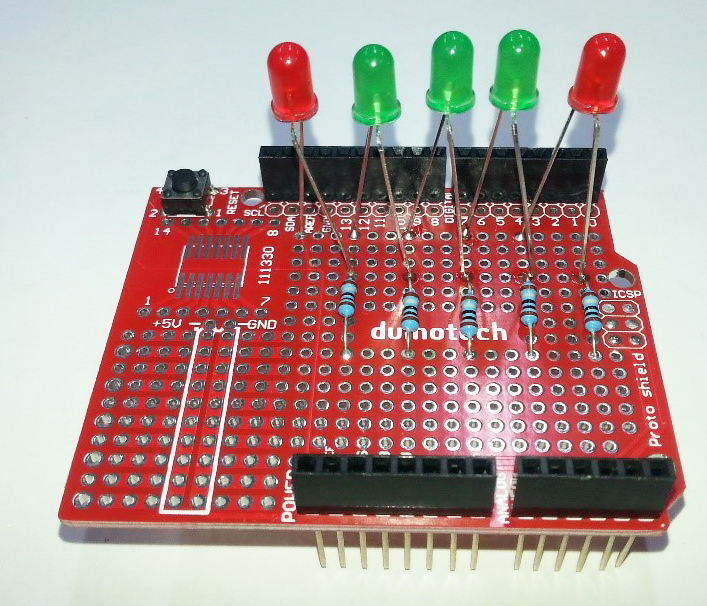
You’ll notice that one leg is longer than the other. This is normal, and is used to differentiate the cathode (negative terminal, and it also has a flat on the body of the LED near it) from the anode (or positive terminal). We’ll need to know this later so our LED’s go onto the board the correct way.

The next step is to fit the resistors to the prototyping board- we’ll do this because they’ll sit lower than the LED’s, and they’ll be hard to get to if the LED’s went in first.

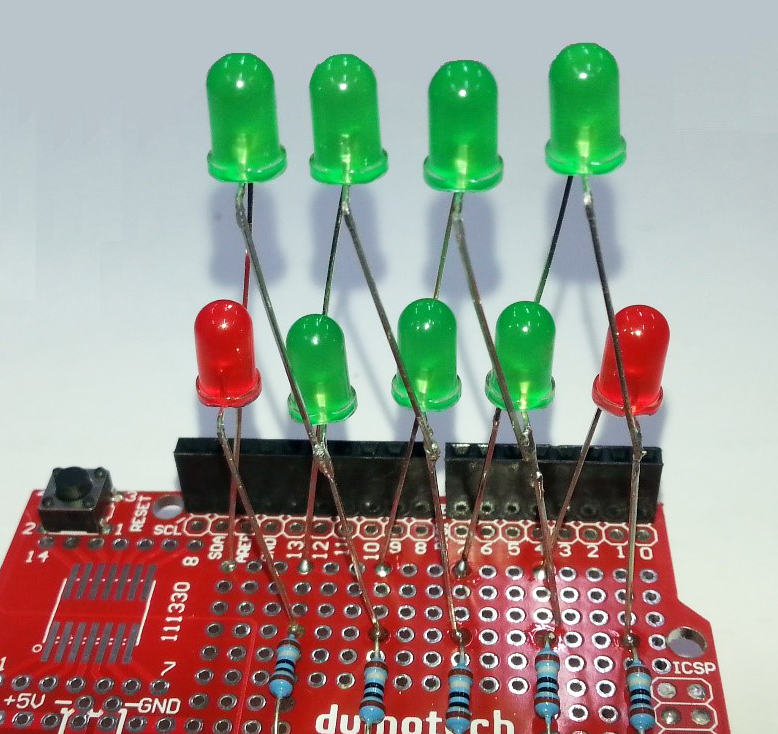


Notice that one end of the resistors is at the fourth row of holes in line with the large hole at the end of the prototyping board, and they are spaced three holes apart in line with D1, D4, D7, between D9 and D10, and between D12 and D13. The other end of the resistor is in the sixth row of holes away from the analog pins.

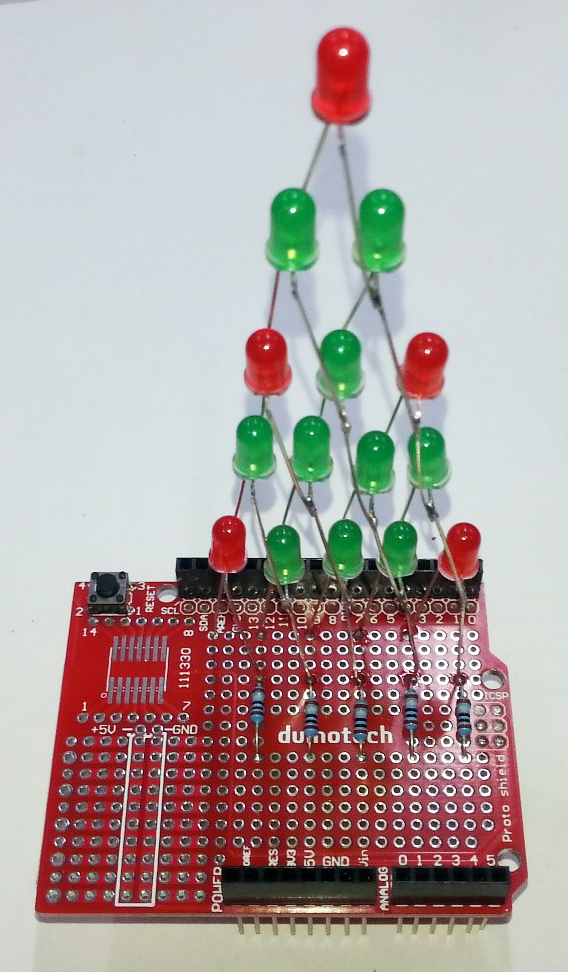
Then we mount the LED’s. The short leg of the LED gets mounted on the end of the resistor leg- this gives it a bit more height. Then the other leg of the LED gets soldered to a hole on the protoboard three holes over, and three holes across. For most of the LED’s, this means their anode (the long leg) is actually in line with the next resistor over. This should give us the legs of the LED’s running clear of either side of the bodies. Note the colours too- we’ve done a red LED at the end and three green LED’s in the middle. Keep following the pattern from the above photo as we build up.



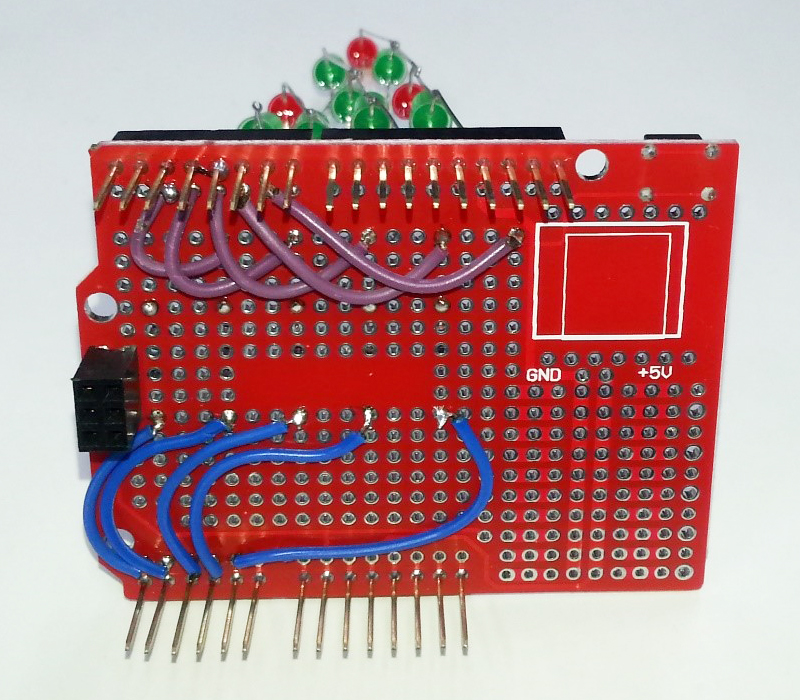
Solder the negative lead of the next row of LED’s to the negative lead of the previous row (we do this first because it’s shorter). Then solder the positive lead of the LED to the positive LED of the adjacent LED below. If you need to, you can now adjust the negative lead.



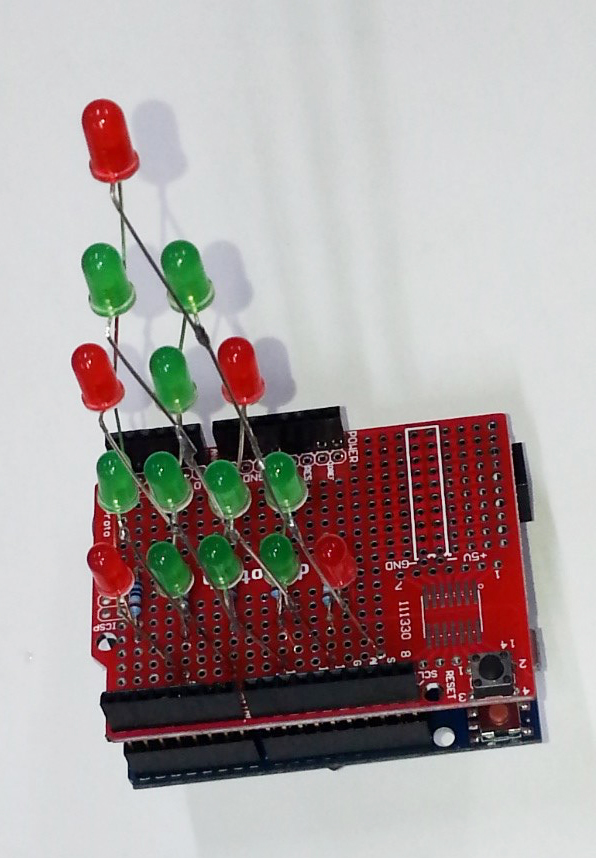
As we keep building up, note that all the negative leads are on the side with the resistors, and all of them slope to the right at roughly the same angle, and all the positive leads are on the other side, sloping the other way. Complete the rows of three, two and one LED’s in a similar fashion, and check against the picture below.



The final wiring step is to connect the LED matrix we have built back to the pins that go to the Uno board. In order, the positive leads go to D2, D3, D4, D5 and D6, while the resistors go to A1, A2, A3, A4 and A5.



All that’s left is to plug the shield into the Uno, and the hardware is complete.



## Code:

The code for this project is small- there is a small program which controls the multiplexing (this is the ISR() function). Multiplexing is when the LED’s are rapidly switched on and off in turn- even though it looks like it they might be all on. The rest of the program in the loop function cycles through a series of patterns, which are set by the values in the tree[] array. You can try changing some of these numbers to see how the pattern changes.

There aren’t any special libraries to load or install, it’s simply a case of opening the sketch, selecting the board and serial port, and pressing upload. The tree should spring to life, with the lights forming a pattern. If one LED doesn’t seem to be lighting, it might be the wrong way around. If it’s a whole diagonal row that’s not working, then the wiring for that row might be broken, or might not be going to the correct pin.

## What’s Next:

As mentioned above, you can change the pattern by changing values in the tree[] array. You might want to add a sensor, and have the tree change patterns if a change is detected in the sensor.

If you’re interested in how the multiplexing works, you can change the following line in the treesetup() function to slow down the scanning. Change:

TCCR1B = \_BV(WGM12) | \_BV(CS10);

to

TCCR1B = \_BV(WGM12) | \_BV(CS10) | \_BV(CS12);

This might be clearer to see if you change all the values in the tree[] array to 32767 (so that all the LED’s are activated).

If you aren’t familiar with binary numbers, you might have worked out how the sketch knows which LED to light up. If not, the diagram below shows the values associated with each LED- to generate a pattern, add up the numbers from the diagram where you want the LED lit. You should see that if you add up all the numbers, you get 32767 as we noted above.

16384 top row

4096 8192

512 1024 2048

32 64 128 256

1 2 4 8 16 bottom row

## Sketch:

//15 LED's in a triangle (half of a 5x5 matrix)

// 14

// 12 13

// 9 10 11

// 5 6 7 8

// 1 2 3 4 5

#define MATRIXFREQ 1000L

byte scanrow=0;

const byte apin[15]={2,3,4,5,6,3,4,5,6,4,5,6,5,6,6};

const byte kpin[15]={A5,A4,A3,A2,A1,A5,A4,A3,A2,A5,A4,A3,A5,A4,A5};

unsigned int treeimg=0; //all off

unsigned int tree[]={

16383,30719,32751,30719,16383,32255,32766,32255,16383,327676,

32753,32287,31743,20479,32767,

32753,32287,31743,20479,32767,

32753,32287,31743,20479,32767,

};

void setup() {

treeSetup();

}

void loop() {

static int i=0; //index counter for cycling through patterns

i=(i+1)%(sizeof(tree)/2); //work with however many elements in tree

treeimg=tree[i]; //change output image => actually displaying is done by the ISR

delay(300); //wait a bit

}

void treeSetup(){

for(int i=0;i<15;i++){

pinMode(apin[i],OUTPUT);

pinMode(kpin[i],OUTPUT);

digitalWrite(apin[i],LOW); //anodes off

digitalWrite(kpin[i],HIGH); //cathodes off

}

// Timer 1 set up as a FREQ Hz sample interrupt, only common thing this affects is servo library

TCCR1A = 0;

TCCR1B = \_BV(WGM12) | \_BV(CS10);

TCNT1 = 0;

OCR1A = F\_CPU / MATRIXFREQ;

TIMSK1 = \_BV(OCIE1A);

}

ISR(TIMER1\_COMPA\_vect) { //gets triggered FREQ times/second (works on Leo and Uno)

scanrow=(scanrow+1)%5; //next row

for(int i=0;i<15;i++){

digitalWrite(apin[i],LOW); //anodes off

digitalWrite(kpin[i],HIGH); //cathodes off

}

digitalWrite(apin[scanrow],HIGH); //active anode on

for(int i=0;i<15;i++){

if(apin[i]==apin[scanrow]){ //if it's on the same anode

if((1<<i)&treeimg){ //and LED is supposed to be lit

digitalWrite(kpin[i],LOW); //cathode on

}

}

}

}