Este proyecto es una variación del cubo de leds 3x3x3 el cual, en lugar de ser un cubo como tal, está abierto y aplicado en un árbol de navidad a modo de luces decorativas.

Los elementos necesarios para su fabricación son:

- Arduino UNO (se ha utilizado una duemilanove con un ATMEGA328 con el bootloader del UNO).
- 27 leds (9 blancos, 9 rojos y 9 verdes)
- 9 resistencias de 220Ω
- 3 resistencias de 1kΩ
- 3 transistores NPN
- pila de 9V
- Un árbol de Navidad, en mi caso, un miniarbol.

El proceso de montaje la decoración es similar al cubo tradicional, aunque con algunas diferencias, Se suelda en 3 lineas y luego se unen entre ellas con cables, para poder envolver el árbol apropiadamente. Aquí tenemos el esquema:

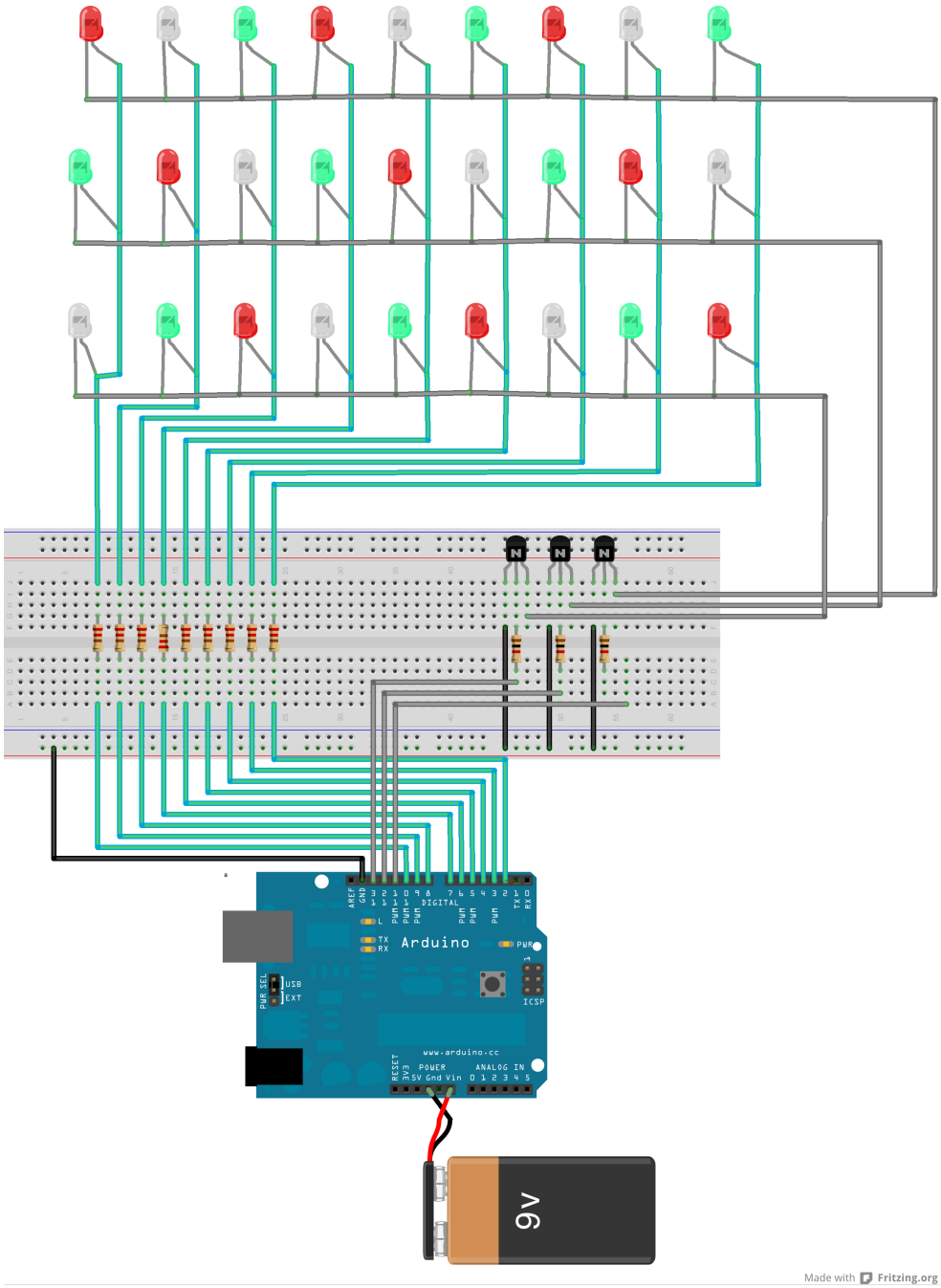


Fig. 1: Esquema del montaje

Para la programación del cubo, se puede realizar pensando que es una matriz de leds, pero hay que tener en cuenta que la posición de cada led ($led[m][n]$) no puede ser programada trivialmente como tal, cada led de la columna n se verá afectado por los leds de la fila m.
Por esta razón, se utiliza el efecto POV (Persistance Of Vision). Este efecto selecciona individualmente cada led y lo enciende, quedando su brillo momentáneamente en la retina, con lo que se aprovecha para encender varios leds y dar sensación de haber varios leds a la vez encendidos en las distintas columnas y filas.

Para ello, Gamaiel Zavala (gzip) publicó una librería para Arduino con licencia MIT, que hace más comprensible su programación. Aquí el código de ejemplo:

```
/*
    ledcube.pde - Example sketch for controlling an LED
    cube.
    Created by Gamaiel Zavala (gzip), 2009-2012
    MIT License. See accompanying LICENSE file for
    terms.
*/

#include <LedCube.h>

#define SIZE 3
#define COLS (SIZE*SIZE)

byte levelPins[SIZE] = {10,11,12};
byte colPins[COLS] = {1,2,3,4,5,6,7,8,9};

LedCube cube(SIZE, levelPins, colPins);

void setup ()
{
}

void loop ()
{
    delay(10);
```

```
// light each light one at a time
for(byte level=0; level<cube.getLevels(); level++)
{
    for(byte col=0; col<cube.getCols(); col++)
    {
        cube.lightPulse(level, col, 100);
    }
}

// light one level at a time, increasing speed
//each time
for(byte d=25; d>2; d-=2)
{
    for(byte l=1; l <= cube.getLevels(); l++)
    {
        cube.lightLevel(l, d);
    }
}

// light each row on each level
for(byte level=1; level<=cube.getLevels(); level++)
{
    for(byte row=1; row<=cube.getLevels()*2; row++)
    {
        cube.lightRow(row, level);
    }
}

// light each plane
for(byte i=3; i; i--)
{
    for(byte row=1; row<=cube.getLevels()*2; row++)
    {
        cube.lightPlane(row, 10*i);
    }
}

// single random light at a time
cube.randomLight(random(25,100),100);
```

```
  // random column drop
  for(byte x=0; x<=15; x++)
  {
      cube.lightDrop(random(0,cube.getCols()),
      random(50,150));
  }

  // circle around cube at a random level
  for(byte x=0; x<=5; x++)
  {
      cube.lightPerimeter(
      random(0,cube.getLevels()), random(1,5),
      random(25,100));
  }

  // light each face
  byte planes[] = {cube.getLevels()+1,
cube.getLevels(),cube.getLevels()*2,1};
  for(byte i=5; i; i--)
  {
      for(byte p=0; p<sizeof(planes); p++)
      {
          cube.lightPlane(planes[p], 5*i);
      }
  }

  // random columns
  cube.randomColumn(25);

  // turn off a single column randomly
  cube.enableBuffer();
  for(byte c=0; c<30; c++)
  {
      cube.fillBuffer();
      cube.invertBuffer();
      cube.randomColumn();
      cube.drawBuffer(7);
  }
  cube.enableBuffer(false);
```

```
  // cols in and out
  for(byte c=1, d=0; c<=10; c++)
  {
      if(c%2 == 0)
      {
          for(d=0; d<20; d++)
          {
              cube.lightColumn(2,1);
              cube.lightColumn(4,1);
              cube.lightColumn(6,1);
              cube.lightColumn(8,1);
          }
      }
      else if(c%4 == 1)
      {
          for(d=0; d<30; d++)
          {
              cube.lightColumn(1,1);
              cube.lightColumn(3,1);
              cube.lightColumn(7,1);
              cube.lightColumn(9,1);
          }
      }
      else
      {
          for(d=0; d<70; d++)
          {
              cube.lightColumn(5,1);
          }
      }
  }

  // diamond and box
  byte diamond[] = {0,4,  1,1, 1,3, 1,4, 1,5,
1,7,   2,4};
  byte box[] = {
      2,0, 2,1, 2,2, 2,3, 2,5, 2,6, 2,7, 2,8,
      1,0, 1,2, 1,6, 1,8,
      0,0, 0,1, 0,2, 0,3, 0,5, 0,6, 0,7, 0,8
```

```
    };
    cube.lightSequence(box, sizeof(box), 200);
    cube.lightSequence(diamond,
    sizeof(diamond), 400);

    // helicopter effect
    byte topSeq[8] = {0,3,6,7,8,5,2,1};
    byte botSeq[8] = {8,5,2,1,0,3,6,7};
    for(byte loops = 0, delay = 50; loops<=8; loops++)
    {
        for(byte s=0; s<8; s++)
        {
            byte seq[] = {2,topSeq[s], 1,4,
          0,botSeq[s]};
            cube.lightSequence(seq, sizeof(seq),delay);
        }
        if(loops < 5) delay-=10;  else delay += 10;
    }

    // turn off one light at a time
    cube.enableBuffer();
    cube.fillBuffer();
    cube.drawBuffer(25);
    for(byte w=0, l, c, max = cube.getNumLights();
    w<max; )
    {
        // lower bound is inclusive, upper is exclusive
        l = random(0, cube.getLevels());
        c = random(0, cube.getCols());

        if(cube.getBufferAt(l,c) == HIGH)
        {
            cube.lightOff(l,c);
            cube.drawBuffer(5);
            w++;
        }
    }
    cube.enableBuffer(false);
}
```

Éste código, junto a la librería se proporciona en la carpeta arbol_navidad, para que no sea necesaria su inclusión en el IDE de Arduino, igualmente se incluye ésta, por si el lector quiere descargarla.
El enlace a la librería `https://github.com/gzip/arduino-ledcube`