# Arduino Project – Documentation

# Lightning Snowflake

Jakab-Gyik Sarolta and Simon Katalin

Students at Technical University of Cluj-Napoca, Computer Science, Group 30413

Email: Jakab.Jo.Sarolta@utcluj.didatec.ro; Simon.Zo.Katalin@utcluj.didatec.ro

2020.12.20.

**Abstract:** This article presents the Lightning Snowflake Arduino based project, using the Arduino Nano development board. By coming in contact with the Touch sensor, it lights up in different with different animations, creating a visual effect of a snowflake.

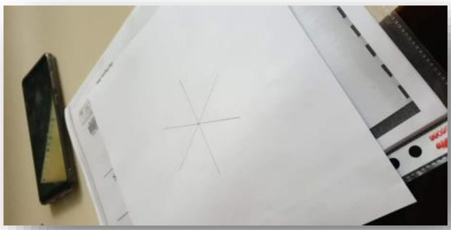
## Table of Contents

## Components and Supplies

- Arduino NANO R3 processor
- 30 LEDs – SMD LEDs used for the core part and simple LEDs used for the branches and leaves
- Resistors
- Brass rods
- Soldering iron, solder
- Pliers

## Construction

We drew a regular hexagon with an 8 cm long diagonal between two different nodes. That will be the size of the inner hexagon with the touch sensor. The outer hexagon with the LEDs will have a 10 cm long diagonal.



We soldered the brass iron to this hexagon shape and solder the leaf LEDs to the top of the branches. Then the other LEDs that are grouped 2 by 2 were soldered to the middle of the branch wires.

Again 2 by 2 we soldered to the core the SMD LEDs.

Then by soldering the resistors to the LEDs we connected them to the Arduino inputs as labelled on the schematic.

## Touch Sensor

Then we add the capacitive touch sensor. It is used to interact with the Arduino snowflake and create different animations every time we touch it.

## Code

We used a software PMW library which allowed us to use all the pins as if they were hardware PWM: https://github.com/bhagman/SoftPWM

Briefly:

Loop()-everything displayed is defined inside this function, which loops

Various animations:

- fireworkAnimation();
- inoutAnimation();
- shinyAnimation();
- circleAnimation();
- loopAnimation();
- waveAnimation();
- fadingAnimation();
- outinAnimation();

```
void inoutAnimation(byte gvalue)

{

 for (int i = 0; i < 5; i++) {

 SoftPWMSet(innerLedPins[i], gvalue);

 delay(100);}
for (int i = 0; i < 5; i++) {
 SoftPWMSet(middleLedPins[i], gvalue);
 delay(100);}
 for (int i = 0; i < 5; i++) {
 SoftPWMSet(edgeLedPins[i], gvalue);
 delay(100);}
 _fill(5);
 }
```

_fill()-all the leds are light up with the same intensity

SoftPWMSet(innerLedPins[i], gvalue);

-the inner leds light up

SoftPWMSet(middleLedPins[i], gvalue);

-the middle leds light up

SoftPWMSet(edgeLedPins[i], gvalue);

-the edge leds light up

Our code:

```
#include "SoftPWM.h"
#include <ADCTouchSensor.h>

byte edgeLedPins[] = {13, A4, A5, 2, 8, 12};
byte middleLedPins[] = {10, 6, 3, 5, 9, 11};
byte innerLedPins[] = {A2, A3, A1, 4, 7, A1};

ADCTouchSensor touchSensor = ADCTouchSensor(A0, 1);

void setup() {
  Serial.begin(115200);
  SoftPWMBegin();
}
```

```
byte animation = 0;
long touchAt = 0;

void loop() {
  switch (animation) {
    case 0:
     fireworkAnimation();
      inoutAnimation(10);
      inoutAnimation(50);
      inoutAnimation(75);
      inoutAnimation(100);
      break;
    case 1:
     fireworkAnimation();
      shinyAnimation();
      break;
    case 2:
     fireworkAnimation();
      circleAnimation();
      break;
    case 3:
     fireworkAnimation();
      loopAnimation();
      break;
    case 4:
      fireworkAnimation();
      fireworkAnimation();
      fireworkAnimation();
      fireworkAnimation();
      fireworkAnimation();
      fireworkAnimation();
      animation ++;
      break;
     case 5:
      waveAnimation();
      break;
    case 6:
      fadingAnimation();
      break;
    case 8:
      outinAnimation(50);
      outinAnimation(80);
      break;
     case 9:
     _fill(100);
     break;
    default:
     _fill(10);
      break;
  }
  int touchValue = touchSensor.read();
  if (touchAt + 2000 < millis() && touchValue > 1000) {
    touchAt = millis(); // touch down, cold down timeout is 2s
    animation ++;
    _fill(0);
  }
}

void fireworkAnimation() {
  for (int i = 0; i < 4; i++) {
    SoftPWMSet(innerLedPins[i], 100);
    delay(100);
  }
  SoftPWMSet(innerLedPins[4], 100);
  for (int i = 0; i < 6; i++) {
    SoftPWMSet(middleLedPins[i], 255);
  }
  delay(50);
  for (int i = 0; i < 6; i++) {
    SoftPWMSet(innerLedPins[i], 0);
    SoftPWMSet(edgeLedPins[i], 255);
  }
  delay(50);
  for (int i = 0; i < 6; i++) {
    SoftPWMSet(middleLedPins[i], 0);
  }
  delay(50);
  _fill(0);
}

void inoutAnimation(byte gvalue)
{
  for (int i = 0; i < 5; i++) {
  SoftPWMSet(innerLedPins[i], gvalue);
  delay(100);}
 for (int i = 0; i < 5; i++) {
  SoftPWMSet(middleLedPins[i], gvalue);
  delay(100);}
  for (int i = 0; i < 5; i++) {
  SoftPWMSet(edgeLedPins[i], gvalue);
  delay(100);}
  _fill(5);
  }

void outinAnimation(byte gvalue)
{
  for (int i = 0; i < 5; i++) {
  SoftPWMSet(edgeLedPins[i], gvalue);
  delay(250);}
  for (int i = 0; i < 5; i++) {
  SoftPWMSet(middleLedPins[i], gvalue);
  delay(350);}
  for (int i = 0; i < 5; i++) {
  SoftPWMSet(innerLedPins[i], gvalue);
  delay(500);}
  _fill(5);
  }
byte circleState[] = {100, 55, 10};
byte circleStateAnimation[] = {1, 1, 1};

void circleAnimation() {
  for (int i = 0; i < 3; i++) {
    if (circleState[i] >= 100) {
      circleStateAnimation[i] = -1; // dim light
    }
    else if (circleState[i] <= 10) {
      circleStateAnimation[i] = 1; // bright light
    }
    circleState[i] += circleStateAnimation[i];
  }
  for (int i = 0; i < 6; i++) {
    SoftPWMSet(innerLedPins[i], circleState[0]);
    SoftPWMSet(middleLedPins[i], circleState[1]);
```

```
    SoftPWMSet(edgeLedPins[i], circleState[2]);
  }
  delay(5);
}

byte waveState[] = {100, 55, 10, 10, 55, 100};
byte waveStateAnimation[] = {1, 1, 1, -1, -1, -1};

void waveAnimation() {
  for (int i = 0; i < 6; i++) {
    if (waveState[i] >= 100) {
      waveStateAnimation[i] = -1; // dim
    }
    else if (waveState[i] <= 10) {
      waveStateAnimation[i] = 1; // bright
    }
    waveState[i] += waveStateAnimation[i];
  }
  for (int i = 0; i < 6; i+=2) {
    SoftPWMSet(innerLedPins[i], waveState[0]);
    SoftPWMSet(middleLedPins[i], waveState[1]);
    SoftPWMSet(edgeLedPins[i], waveState[2]);
    SoftPWMSet(innerLedPins[i + 1], waveState[3]);
    SoftPWMSet(middleLedPins[i + 1], waveState[4]);
    SoftPWMSet(edgeLedPins[i + 1], waveState[5]);
  }
  delay(10);
}

byte loopCounter = 0;
byte loopState = 150;
```

```
void loopAnimation() {
  SoftPWMSet(innerLedPins[loopCounter], loopState);
  SoftPWMSet(middleLedPins[loopCounter], loopState);
  SoftPWMSet(edgeLedPins[loopCounter], loopState);

  loopCounter = _nextIndex(loopCounter, 1);
  if (loopCounter == 0) {
    loopState = (loopState == 150 ? 0 : 150);
  }
  delay(100);
}

byte slowOnCounter = 0;
byte slowOnState = 150;

void slowOnAnimation() {
  byte randomLed = random(0, 18);
  if (randomLed < 6) {
    SoftPWMSet(innerLedPins[randomLed], slowOnState);
  }
  else if (randomLed < 12) {
    SoftPWMSet(middleLedPins[randomLed - 6], slowOnState);
  }
  else {
    SoftPWMSet(edgeLedPins[randomLed - 12], slowOnState);
  }
  slowOnCounter ++;
  if (slowOnCounter >= 50) {
    slowOnCounter = 0;
    slowOnState = (slowOnState == 150 ? 0 : 150);
  }
  delay(50);
}

byte shinyState[] = {0, 100, 0, 100, 0, 100};
byte shinyStateAnimation[] = {1, 1, 1, 1, 1, 1};
byte shinyCounter = 0;

void shinyAnimation() {
  for (int i = 0; i < 6; i++) {
    if (shinyState[i] >= 100) {
      shinyStateAnimation[i] = -1; // dim
    }
    else if (shinyState[i] <= 0) {
      shinyStateAnimation[i] = 1; // bright
    }
    shinyState[i] += shinyStateAnimation[i];
    SoftPWMSet(edgeLedPins[i], shinyState[i]);
  }
  shinyCounter ++;
  if (shinyCounter > 10) {
    shinyCounter = 0;
    for (byte r = random(1, 3); r > 0; r--) {
      byte randomLed = random(0, 12);
      if (randomLed < 6) {
        SoftPWMSet(innerLedPins[random(0, 6)], 255);
      }
      else {
        SoftPWMSet(middleLedPins[random(0, 6)], 255);
      }
    }
  }
  else {
    for (int i = 0; i < 6; i++) {
      SoftPWMSet(innerLedPins[i], 20);
      SoftPWMSet(middleLedPins[i], 20);
    }
  }
  delay(30);
}

byte fadingState[] = {0, 100, 0, 100, 0, 100};
byte fadingStateAnimation[] = {1, 1, 1, 1, 1, 1};

void fadingAnimation() {
  for (int i = 0; i < 6; i++) {
    if (fadingState[i] >= 100) {
      fadingStateAnimation[i] = -1; // dim
    }
    else if (fadingState[i] <= 0) {
      fadingStateAnimation[i] = 1; // bright
    }
    fadingState[i] += fadingStateAnimation[i];
    SoftPWMSet(edgeLedPins[i], fadingState[i]);
    SoftPWMSet(middleLedPins[_nextIndex(i, 1)], fadingState[i]);
    SoftPWMSet(innerLedPins[i], 50);
  }
  delay(20);
}

void _fill(byte value) {
  for (int i = 0; i < 6; i++) {
    SoftPWMSet(edgeLedPins[i], value);
    SoftPWMSet(middleLedPins[i], value);
    SoftPWMSet(innerLedPins[i], value);
  }
}

byte _prevIndex(short index, byte step) {
  index -= step;
  while (index < 0) {
    index += 6;
  }
  return index;
}

byte _nextIndex(short index, byte step) {
  index += step;
  while (index > 5) {
    index -= 6;
  }
  return index;
}
```

## References

Jiripraus. CircuitsArduino. Arduinoflake.[online] https://www.instructables.com/Arduinoflake/

Bhagman. SoftPWM. GitHub. [online] https://github.com/bhagman/SoftPWM

Arduino. Arduino Create. [online] https://www.arduino.cc/