Heart CKT II

Project in a Box

Heart CKT II

Developed by Stazia “WigginWannabe” Tronboll and Colin “MrSwirlyEyes” Keef

Last Updated: 2016-09-21

Revision 1.0

PLEASE REPORT ANY FEEDBACK/ERRORS/COMMENTS/QUESTIONS/CONCERNS to us at: ProfSwirlyEyes@gmail.com
All and any inquiries are very much appreciated!

Phuong Truong | Professor Truong Nguyen

Introduction

In this project, you will create a simple LED lightshow – the name Heart was given because it was originally a Valentine's Day CKT, but the design is up to you (I.e. Star, Letters, etc). Although 11 different controllable pins are available, we used 10 in this design. This shouldn't limit you. Feel free to combine two or more LED's to each pin, and create any image you want.

The Heart CKT II is an introduction to EAGLE, PIC's (Programmable Integrated Chips), Arduino IDE, and Arduino C programming. First you will design the schematic in EAGLE and breadboard it, then choose whether to design a PCB board or solder the circuit. When the hardware is ready, you will learn how to prepare an Atmel ATTiny chip for use, and program it to perform a pattern with the LEDs.

This combines creativity with a powerful underlying concept that has far reaching applications and potential. PICs allow us to complete and sustain Arduino projects without permanently dedicating an Arduino.

Customized Design

We encourage you to create your own image, but there are some restrictions! PCBs generally take a long time to fabricate and ship, and in the case of mistakes on PCB designs, they are very hard – if not impossible – to salvage. Because of this, if you want to design a PCB for **Challenge #2**, you should probably stick to the heart shape. With that said, you are welcome to design a personal board (and order it for yourself!).

Customized designs may use more than 10 LEDs – you know your design and the parts you need for it. When providing instructions, we will refer to the 10 LEDs used in the heart shape and trust you to adapt the guidelines to suit your design.

Learning Objective

- Understand the Arduino UNO and its various features
- Simple programming in Arduino C/C++
- Using Arduino UNO as ISP to program ATTiny chips
- Understand the benefits of the ATTiny chips and how to use them
- Using ATTiny in a simple program
- PCB design using an ECAD program
- Soldering

Required Materials for Entire Project

Item	REF	Vendor	Qty	Price	Total	Note
Arduino UNO		Sparkfun	1	24.95	24.95	
Breadboard		Sparkfun	1	4.95	4.95	
ATTiny84	**U1**	Sparkfun	1	2.95	2.95	Can alternatively get an ATTiny44
14-pin DIP Socket		Sparkfun	1	0.50	0.50	
LED	**LED1 – LED10**	Sparkfun	10	0.95	9.50	Color(s) of choice
Resistor Kit	**R1-R10**	Sparkfun	1	7.95	7.95	Value may vary; need ~10
Slide Switch	**S1**	Sparkfun	1	0.75	0.75	3 pins
3V BATT connector	**BATTERY_1**	Sparkfun	1	1.50	1.50	
3V coin cell battery		Sparkfun	1	1.95	1.95	
Heart II v5 PCB		OSH Park	1	27.10	20.10	Pack of 3; Price may vary; 2 extra PCB at end
			Grand Total		82.10	Does not include shipping

Required Tools for Entire Project

- Arduino UNO + USB A (male) to USB B (male) cable
- Computer + Arduino IDE + ATTiny boards
- Computer + PCB CAD program (i.e. EAGLE , KiCAD or Altium's CircuitMaker)
- Internet connection
- Hook-up wire (20-gauge) or male-to-male jumper wire
- Wire cutter
- Soldering Iron + solder
- File (recommended)
- Tape (recommended)
- Needle nose pliers (recommended)
- Digital Multimeter (recommended)
- MrSwirlyEyes' HEART CKT II PCB board from GitHub (EAGLE .sch and .brd files) (optional)

Challenge #1: Designing the Schematic

In this challenge we introduce you to PCB design using EAGLE (you may use another program if you wish). We take the approach of designing the PCB first to take on a different perspective than breadboard prototyping first. Firstly, it is arguably easier to do a custom design when you can play around on the board design instead of breadboarding (especially if you want to do interesting shapes, since you have to conform to the breadboard). You may, in **Challenge #2**, decide to change your design, which is fine (and common).

Learning Objective

- PCB design using an ECAD program

Required Materials

- Arduino UNO + USB
- Breadboard
- ATTiny84 [**U1**]
- 14-pin DIP socket
- LED [**LED1-LED10**]
- Resistor [**R1-R10**] ***values needed may vary**
- Slide switch [**SW**]
- 3V Battery connector
- 3V coin cell battery

Required Tools

- Hook-up wire (20-gauge) or male-to-male jumper wire
- Computer + PCB CAD program (i.e. EAGLE , KiCAD or Altium's CircuitMaker)
- Internet connection

Regardless of whether you choose to solder or design a PCB, it is useful to design the schematic in EAGLE. Thus, challenge one is to design a schematic with the following properties:

1. Place ten LEDs, each with a resistor. If you are pursuing a custom design, use the appropriate number of LEDs and their series/parallel relationships. Make sure they form at most 11 groups, and that each group of LEDs has a resistor.
2. Connect the LED-resistor pairs/groups each to an I/O pin on Atmel's ATTiny84 chip. Use the ATTiny84 datasheet (link in **Resources & References**) to determine which pins are good options.
3. Use a switch to provide 3V to the ATTiny84

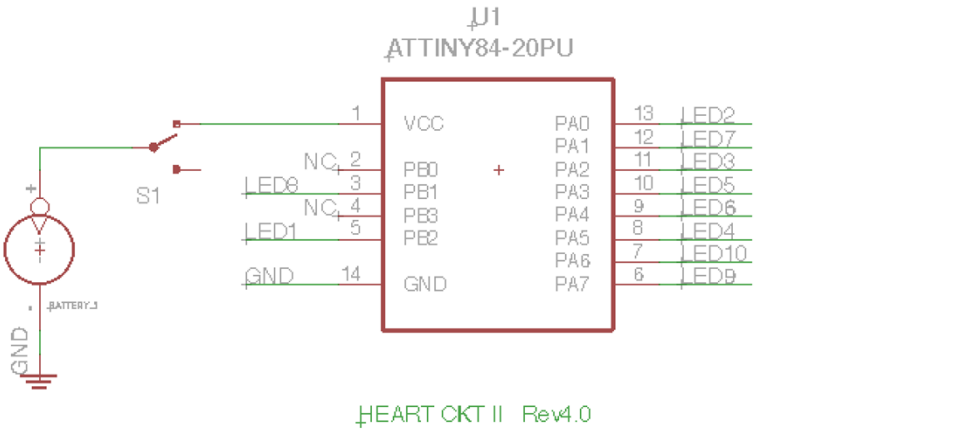
Before beginning, read the first tutorial found in the **Resources & References** section at the back. It will teach you how to design a schematic with EAGLE. Similarly **IEEE/HKN/TBP PCB Design Workshop** has great resources for learning EAGLE PCB design.

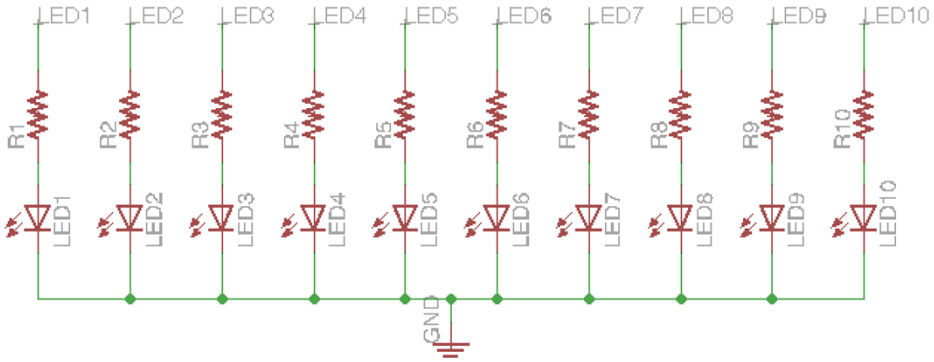
To make the schematic, you will need specific parts that aren't present in Eagle's default libraries. Follow this guide to add libraries to your machine: **How to Install and Setup EAGLE**. We also would like to point you to a previous project of ours: **MrSwirlyEyes' ATTiny Programmer** which can be used to program the ATTiny84 in this project and goes into some detail on PCB design if you are unfamiliar with either of these two.

Note: More details on ATTiny84 in **Challenge #3**

CKT Element	REF	Library	Device
ATTiny84	**U1**	Atmel_By_element14_Batch_1-00	ATTINY84-20PU
Battery	**BATTERY_1**	Itead_BP_110825	BATTERY_1236_C
Switch	**S1**	SparkFun-Electromechanical	SWITCH-SPDTPTH (SWITCH-SPDT)
LED	**LED1-LED10**	Led	LED5MM (LED)
Resistor	**R1-R10**	Rcl	R-US_0207/7 (R-US_)

Now you're ready. Make a schematic that fulfills the project specifications listed at the beginning of this challenge. You may use our schematic as a correctness guide, but try to solo it as much as possible.





Once the schematic is done, prepare a breadboard model of your design. This is a good sanity check for your schematic, and we suggest breadboarding every design before progressing to more permanent builds!

Challenge #2: Building the Circuit

The point of this challenge is to train you in designing PCB boards or soldering. Soldering the board is cheaper and you can have it done within a few hours. If you choose the PCB route, you train the valuable skill of using EAGLE and the final result looks cleaner, but printing is risky (especially the first few times) if there are mistakes – the board is generally useless.

Learning Objective

- PCB design using an ECAD program
- Soldering

Required Materials

- Arduino UNO + USB
- Breadboard
- ATTiny84 [**U1**]
- 14-pin DIP socket
- LED [**LED1-LED10**]
- Resistor [**R1-R10**] ***values needed may vary**
- Slide switch [**SW**]
- 3V Battery connector
- 3V coin cell battery

Required Tools

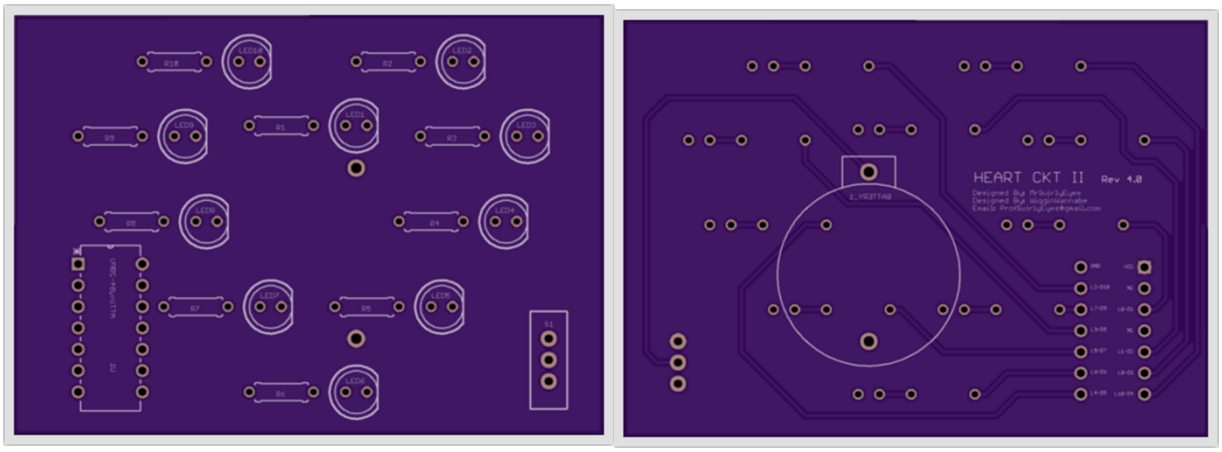
- Computer + PCB CAD program (i.e. EAGLE , KiCAD or Altium's CircuitMaker)
- Internet connection
- Wire cutter
- Soldering Iron + solder
- File (recommended)
- Tape (recommended)
- Needle nose pliers (recommended)
- Digital Multimeter (recommended)

PCB

1. Read the tutorial on board design found in the **second link in Resources & References**. Also look at the board hints below for guidance, but only *after* reading the tutorial.
2. Have fun! Design your board, paying close attention to what side of the board your parts are on.
3. Find a friend to check your board.
4. We stress taking your time in designing your board. Really – you don't want to mess this up! When you're certain that you're ready…
5. Build it! Gather the materials in the list below, and solder them in – follow the instructions for soldering, with the exception that you don't need to solder connections. Just secure the parts into their place.

Board Hints

1. We recommend setting your grid to 0.1 inches, with 0.025 inches alt
2. It is safe to have routing lines 1mm away from each other. It results in boards as seen on the right side of the right image, below. The dark spaces are non-conducting. Don't get closer than 1mm though!
3. When you ratsnest with ground planes, as described in the tutorial, the ground routing goes away.
4. We don't recommend the autorouter! It is often not as clean, and could produce errors depending on the DRC rules of your fabricator.



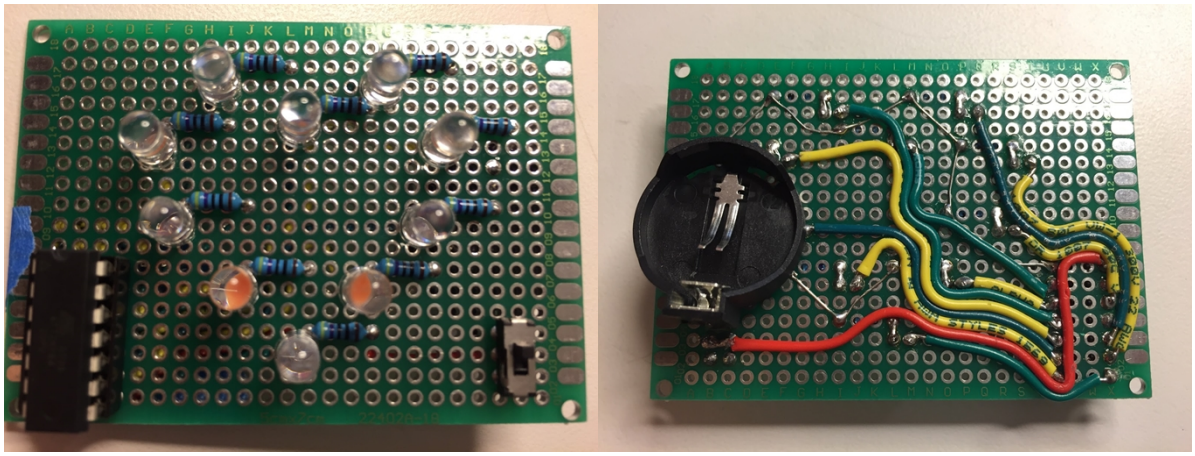
Board Readiness Checklist – if you pass these, you are probably finished with your board

- ☐ Run **ratsnest**. You should see **"Nothing to Do"** in the lower left of the screen. If there are any airwires, find and route them!
- ☐ Run DRC error checks using the rules of your fabricator. We use OSH Park, and their dru file can be found at https://oshpark.com/guidelines. Stopmask errors are okay - they just mean that there is some overlap between silkscreen text and solder pads. Know that in a stopmask error, the text will be sacrificed in favor of the solder pad. Keep a special eye out for the three errors mentioned in the tutorial. If you see any other errors, Google it or ask us at ProfSwirlyEyes@gmail.com
- ☐ Find a friend to check routing and connections!

Soldering

1. Ensure you have all of the materials in the list below. Choose 10 LEDs of any color(s) available
2. Use a breadboard to set up a testing station for varying resistances with LEDs. This is simply a resistor in series with an LED, supplied 3V. For each of your chosen colors, determine a resistance value that makes the LED an acceptable brightness.
3. Lay out the parts on the protoboard. This step is artistic and up to you, though keep in mind that there has to be space for the battery on the back, and – depending on how you made your schematic – the correct polarity of the battery has to reach the switch.
4. Now it's a matter of soldering however seems best to you. We suggest securing the LEDs first, then tie them to the chip and switch. Hook in the battery last

Item	REF	Vendor	Qty	Price	Total	Note
Arduino UNO		Sparkfun	1	24.95	24.95	
Breadboard		Sparkfun	1	4.95	4.95	
ATTiny84	**U1**	Sparkfun	1	2.95	2.95	Can alternatively get an ATTiny44
14-pin DIP Socket		Sparkfun	1	0.50	0.50	
LED	**LED1 – LED10**	Sparkfun	10	0.95	9.50	Color(s) of choice
Resistor Kit	**R1-R10**	Sparkfun	1	7.95	7.95	Value may vary; need ~10
Slide Switch	**S1**	Sparkfun	1	0.75	0.75	3 pins
3V BATT connector	**BATTERY_1**	Sparkfun	1	1.50	1.50	
3V coin cell battery		Sparkfun	1	1.95	1.95	
Heart II v5 PCB		OSH Park	1	27.10	20.10	Pack of 3; Price may vary; 2 extra PCB at end
			Grand Total		82.10	Does not include shipping



Ta-da! This is an example – yours need not look identical.

In the next challenge, we will test it with a preprogrammed Atmel chip – if all is well, you will be able to program your own chip!

Challenge #3: Programming the ATTiny84

Pull out an Arduino and your laptop; it's time to make that circuit do stuff. Using the Arduino IDE, we are going to configure an Arduino into an ISP (In System Programmer). This allows us to write normal programs, but instead of uploading them into Arduino we upload them to an ATTiny. This is a powerful (and awesome) concept. Now instead of keeping an Arduino tied to a particular project, you can program a chip to take its place, and use the Arduino as a prototype for your next project.

Learning Objective

- Understand the Arduino UNO and its various features
- Simple programming in Arduino C/C++
- Using Arduino UNO as ISP to program ATTiny chips
- Understand the benefits of the ATTiny chips and how to use them
- Using ATTiny in a simple program

Required Materials

- Arduino UNO + USB A (male) to USB B (male) cable
- Your circuit from the last challenge
- ATTiny Programmer (optional)
    - See **Resources & References** for details about MrSwirlyEyes' ATTiny Programmer
- ATTiny84-PU chip

Required Tools

- Arduino UNO + USB A (male) to USB B (male) cable
- Computer w/ Arduino IDE installed
    - Follow this document to install Arduino: Installing Arduino
- Download the ATTiny library
    - Follow this document to install a library: Installing Arduino by following the optional part:
        - **[Optional] Adding 3rd party boards to Arduino**
    - **Note:** There are many different ATTiny libraries, we particularly like this one because it has all the ATTiny's we use, namely: ATTiny85, ATTiny84, ATTiny2313

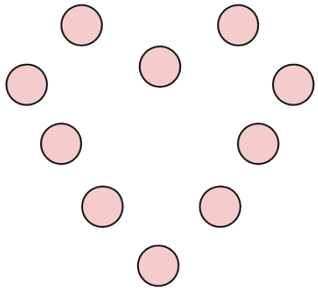
Setup ATTiny

1. Follow MrSwirlyEyes' ATTiny Programmer **Challenge #1** and **#3** to setup your ATTiny84
2. You can either use or make the **ATTiny Programmer** itself or follow the challenges to simply configure your ATTiny84 using a breadboard
    - Make sure to set up Arduino as an **In-System Programmer** and to **burn bootloader** to the ATTIny84 before moving on

Create the Lightshow Sequence

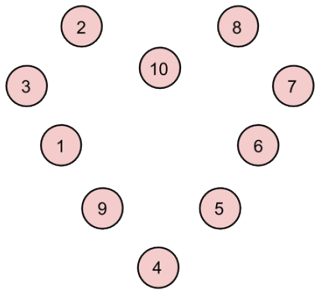
It is time to program a sequence for your lights. When we routed the LED pins to the ATTiny84, we somewhat disregarded which ATTiny84 pins you had to solder the LEDs to. This was for convenience when it comes to making connections. To fix this, we will create a simple program that will essentially map the LED pins, so that you can easily determine which LEDs are connected each particular pin on the ATTiny84. We have provided code for mapping – you are free to jump on ahead and determine your own method for ordering the LEDs and just start coding the lightshow.

1. To map the LEDs, copy the code from MrSwirlyEyes' HEART CKT II (**Heart_II_CKT_LED_mapping.ino**), and upload it. Pop the programmed ATTiny84 into your circuit.
2. Grab some scratch paper, a writing utensil, and draw a figure of your heart (or whatever shape you created), like so:



3. Turn on the circuit and map the LED's. As each LED turns on, put a number starting from 1,2,...,10 into each circle corresponding to the lit LED. After, you should have something that looks like the following.

    Note: Could be any order; it doesn't matter as long as you have mapped all the LED's



4. This process reveals which order the LEDs are attached to the ATTiny. Now use it to program a sequence for your heart! You aren't restricted to digital logic; there are PWM pins that we encourage you to use.

    Note: Be mindful of memory, the ATTiny84 has far less than the ATMEGA328 (the Arduino Uno chip)

Conclusion: Put it Together!

Plug your chip into your circuit, flip the switch, and enjoy the fruits of your labor! You should be comfortable designing schematics in EAGLE, boards in EAGLE, soldering, and programming an ATTiny chip.

If you are interested, you may do MrSwirlyEyes' The Binary Clock project, a more advanced ATTiny application. Also take a look at these extension ideas – perhaps for your final project.

Ideas for Extensions

- Design a library to blackbox sequence code for any shape and sequence. Generalize it to accept the pin mapping and produce the sequence for any circuit using your shape. Consider making patterns that the user can string together in their own order to produce their sequence. Can it support multiple shapes?
- Make an LED sign, with multiple LEDs on each pin and selectable modes. Consider using a 9V battery to support more LEDs per group, and transistors attached to the Atmel pins to act as switches. Perhaps a pushbutton can select between predefined patterns, or a potentiometer can change the speed of the pattern.

Resources & References

This is the official EAGLE tutorial on schematic design. It is very clear and well-written – definitely go through it in preparation for Challenge #1!

https://learn.sparkfun.com/tutorials/using-eagle-schematic

This is the official EAGLE tutorial on board design. Read through it before embarking on Challenge #2, if you choose the PCB route.

https://learn.sparkfun.com/tutorials/using-eagle-board-layout

HKN, IEEE, and TBP's EAGLE PCB design workshop:

IEEE/HKN/TBP PCB Design Workshop

Our ATTiny Programmer project:

MrSwirlyEyes' ATTiny Programmer

This is the datasheet for programmable chip ATTiny84:

http://www.atmel.com/Images/8006S.pdf

Appendix A: Sample Sequence

This is sample code for Challenge 3. The mapping is unique to the circuit it was programmed for, which in this case was a 10-LED star.

Commented [CVK1]: Maybe just point to Github?

Commented [SPT2R2]: kk. Put your sequence up and point to it, since all the other references are to your account. Consistency.

```
int base_seq[] = {1, 2, 8, 6, 9, 10, 4, 3, 2, 5, 8, 9, 7, 10, 3, 1};
int upside_triangle[] = {4, 3, 2, 5, 8, 9, 10};
int triangle[] = {7, 10, 3, 1, 2, 8, 6};
int inside[] = {10, 3, 2, 8, 9};
int outside[] = {1, 5, 6, 7, 4};
int led[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int leds = 10;
int base_seq_num = 15;
int upside_triangle_num = 7;
int triangle_num = 7;
int inside_num = 5;
int outside_num = 5;

int delay_long = 500;
int delay_medium = 300;
int delay_short = 100;
int delay_trail = 75;

void setup() {
  for ( int i = 0 ; i < leds ; i++ ) {
    pinMode(led[i], OUTPUT);
  }
}

void trail() {
  for (int i = 0; i < base_seq_num; i++) {
    digitalWrite(base_seq[i],HIGH);
    delay(delay_trail);
    digitalWrite(base_seq[i],LOW);
  }
}

void draw_on() {
  for (int i = 0; i < base_seq_num; i++) {
    digitalWrite(base_seq[i],HIGH);
    delay(delay_medium);
  }
}

void draw_off() {
  for (int i = 0; i < base_seq_num; i++) {
    digitalWrite(base_seq[i],LOW);
    delay(delay_medium);
  }
}
```

```
void star_on() {
  for (int j = 0; j < leds; j++) {
    digitalWrite(led[j],HIGH);
  }
}

void star_off() {
  for (int j = 0; j < leds; j++) {
    digitalWrite(led[j],LOW);
  }
}

void upside_triangle_blink() {
  for (int j = 0; j < upside_triangle_num; j++) {
    digitalWrite(upside_triangle[j],HIGH);
  }
  delay(delay_long);
  for (int k = 0; k < upside_triangle_num; k++) {
    digitalWrite(upside_triangle[k],LOW);
  }
}

void triangle_blink() {
  for (int j = 0; j < triangle_num; j++) {
    digitalWrite(triangle[j],HIGH);
  }
  delay(delay_long);
  for (int k = 0; k < triangle_num; k++) {
    digitalWrite(triangle[k],LOW);
  }
}

void outside_blink() {
  for (int j = 0; j < outside_num; j++) {
    digitalWrite(outside[j],HIGH);
  }
  delay(delay_long);
  for (int k = 0; k < outside_num; k++) {
    digitalWrite(outside[k],LOW);
  }
}

void inside_blink() {
  for (int j = 0; j < inside_num; j++) {
    digitalWrite(inside[j],HIGH);
  }
  delay(delay_long);
  for (int k = 0; k < inside_num; k++) {
    digitalWrite(inside[k],LOW);
  }
}
void loop() {
```

```
  // broadway lights
  for (int l = 0; l < 2; l++) {
    trail();
  }
  delay(delay_short);

  // blink 3 times
  for (int l = 0; l < 3; l++) {
    star_on();
    delay(delay_long);
    star_off();
    delay(delay_long);
  }
  delay(delay_short);

  // broadway again
  for (int l = 0; l < 2; l++) {
    trail();
  }

  // draw on then off
  draw_on();
  draw_off();

  // alternating triangles
  for (int l = 0; l < 2; l++) {
    upside_triangle_blink();
    delay(delay_short);
    triangle_blink();
    delay(delay_short);
  }
  delay(delay_short);

  // alternating inside/outside
  for (int l = 0; l < 2; l++) {
    outside_blink();
    delay(delay_short);
    inside_blink();
    delay(delay_short);
  }
  delay(delay_short);
}
```