South Berkeley Electronics

Electronic Snowflake Model I

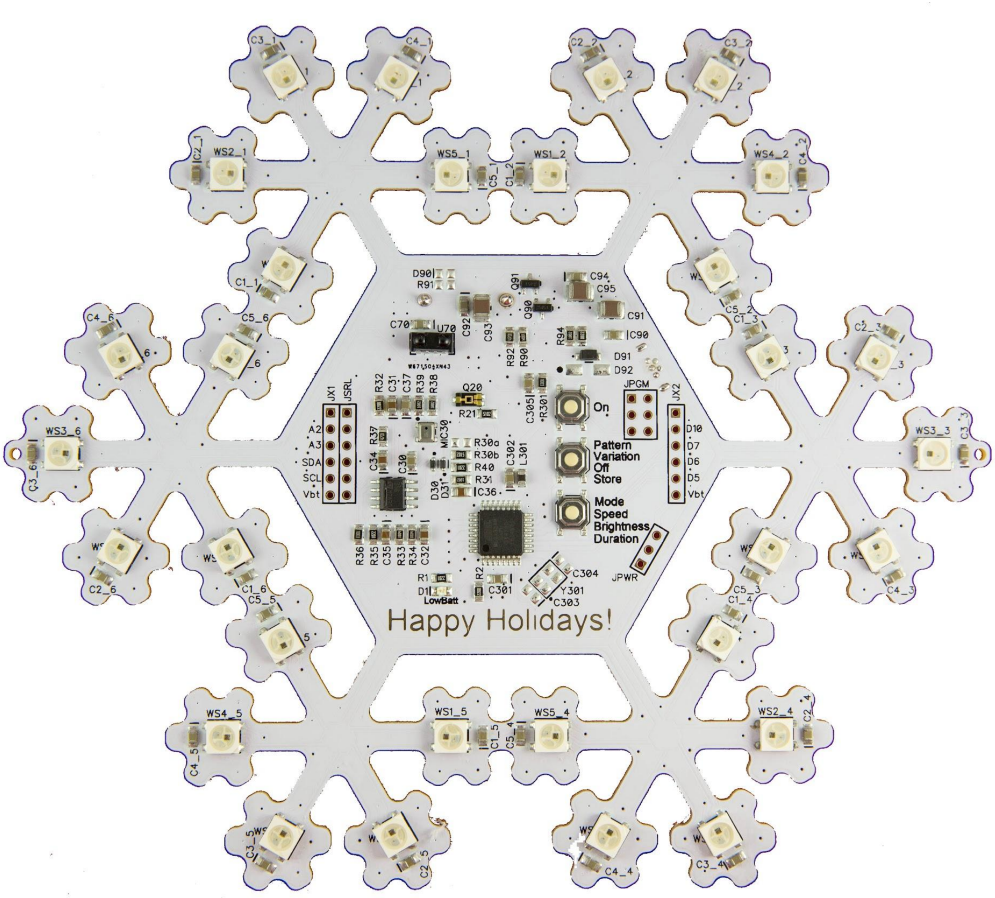
Manual Rev 0.090 November, 2020

[Warranty]

Congratulations on your purchase of the South Berkeley Electronics Snowflake! We hope that you enjoy your new toy and that it will provide you years of nerdy entertainment.

What is it?

The electronic snowflake was conceived as a Christmas tree ornament for people with electronics enthusiasts in their family. We thought it would be fun to make a circuit board in the shape of a snowflake with colored lights.



However, we might have gotten a little carried away. After all:

- shouldn't a light-up snowflake automatically adjust to ambient lighting?
- and wouldn't it be cool if it responded to sound?
- and if you put it high in a tree, wouldn't it be nice to be able to control it remotely?
- and finally, since it is based on the same technology as the popular Arduino, shouldn't it be easy to reprogram?

The answer was **YES**, of course it should do those things!

Precautions

The snowflake consists of electronic components soldered onto a bare circuit board. We think this looks pretty cool. The board and components are lead-free, but the snowflake should not be used where small children might put it in his or her mouth. Furthermore, the item is somewhat delicate, particularly the USB power connector on the reverse side, so please be gentle when handling.

Basic Operation

Operation of the snowflake is simple, if a bit quirky. There are three buttons that control several functions. Because of all the functions possible, the buttons each can do more than one thing. The time you hold down the button determines what command you issue.

“On” button

Okay, we lied a bit. This one is simple.

function	duration	Meaning
Turn on	Any duration	This just turns the unit on if it is not already and resets it.

“Pattern” Button

function	duration	Meaning
pattern	Quick press	Advances the unit through the next light pattern.
variation	1-2 seconds	Advances to the next variation of the current light pattern. Some light patterns respond to variation changes, some do not, so this command doesn’t *always* do something.
off	5-6 seconds	Turns the snowflake off. (Note: on battery power, off is really off. On USB power, the unit is in a low-power state so that the remote control can still wake it.
store	> 10 seconds	Stores all the current settings to non-volatile memory as your “favorite” (preferred default). When you reset or next use the “on” button,the snowflake will start in

		the same configuration it was in when you issued the store.

“Mode” Button

function	duration	Meaning
mode	Quick press	Switch the sound mode. There are three sound modes: - No sound response (sound mode off) - “VU” meter mode - Flash to rhythm mode
speed	1-2 seconds	Adjusts the update speed of the light pattern. The pattern update speeds can be adjusted from very fast to very slow in 9 steps. The default is a moderate speed.
brightness	5-6 seconds	Adjusts the overall brightness of the LEDs. Note that the LEDs will always respond to ambient lighting conditions. This setting essentially adjusts the “top” of the output brightness range.
duration	> 10 seconds	Adjust the amount of time the unit stays on. The default is 30 minutes.

What all the modes and settings do

Patterns

The most basic setting of the snowflake is the “pattern”. This is the effect or animation that is currently being displayed. Patterns vary from spinning colors to cylon sweeps, and all manner of random flashing in between.

As of the writing of this manual, there are thirteen patterns, plus a special “settings” pattern, which I will describe below.

Pattern Names													
Chaser	sparkle	rainbow	sparse	snake	leaves	flash	cylon	solid	minicircle	inching	pulse	fade	lines

“Settings” Pattern

One of the patterns available on the snowflake is not much fun, but it is helpful. I call it the “settings” pattern. In this pattern you can see the current setting of the speed, turn-off delay, and maximum brightness. Three LEDs will be lit continuously. The blue one tells the speed setting. It rotates around the LEDs on one “arm” of the snowflake as you advance the setting. The magenta one does the same for the turn-off delay, and the teal one does the same for the brightness setting.

If, in this mode, you make very long press on the second button to “store” your settings, they will all be stored, including the current pattern. That will be a bit boring, as the settings pattern itself will become the default, which you probably do not want. However, if you want to store the speed, shutdown delay, etc, you can certainly just do that in this pattern, then switch to another pattern and issue the “store” again, and that pattern will be stored without overwriting the other setting.

Variations

Some patterns also have variations. These are slight variation on the pattern, using the same overall code. For example, what colors to use in a “rainbow” pattern, which way to spin, and how many lights to twinkle.

The variations do not have names. You’ll just have to play around a bit to see what they do.

Speed

This determines how fast the animation runs. There are nine speed settings. Each step goes slower, until it wraps around and goes fast. The default speed is a middle value.

Update periods								
70 ms* (default)	100 ms	200 ms	500 ms	1 s	5 s	5 ms	20 ms	40 ms

Mode

This determines if the snowflake will respond to sound. There are three sound modes. The first is off -- no sound response. The second mode causes the snowflake to act like a sort of “VU Meter”, lighting more LEDs the louder the sound. The third mode is a mode where it will enable all the LEDs together, according to the general sound level.

One thing to understand about the sound modes is that they work *independently* of the pattern. Think of the sound mode as acting like a mask overlaid on the pattern that would otherwise be lit. If the pattern has few LEDs lit to begin with, it might be hard to tell that the sound mode is working at all. Therefore, sound mode is the most fun with patterns that would have had most of the LEDs lit without sound mode enabled.

Sound Modes		
Normal, not sound sensitive (default)	Sound sensitive “VU meter”	Sound sensitive “beat flash”

Brightness

The snowflake is equipped with an ambient light sensor. This is nice, as the snowflake will auto-dim in a dark room. When you use the brightness adjustment you are actually adjusting the maximum brightest that a pattern can use. Patterns that vary LED brightness still do so, but in a range determined by this setting.

Brightnesses				
Medium (default)	medium-bright	bright	brightest	dim

Duration

The snowflake turns itself off after a preset period, which you can control. The settings are as below:

On Times								
30 min (default)	1 hr	2 hr	3 hr	6 hr	12 hr	21 hr	5 min	15 min

Remote Control Operation

The snowflake can be operated by the infra-red remote control provided just as easily as from the front panel buttons. The same functions are available, but they are mapped to buttons as follows:

Button	Meaning
1, ▸	Go to next pattern
◂	Go to previous speed
2	Go to next variation
3	Save the current state as the power-on default
4	Increment the mode
5, ▼	Go to next speed
▲	Go to previous speed
6	Increment the brightness
7, ✱	Turn off, or if it's already off, back on[†]
9, #	Increment turn-off delay
0	Increment "automatic mode" (changes patterns on it own)

[†]Hitting "7" or "✱" to turn off the snowflake will have different behavior depending on whether the snowflake is running on batteries of USB power. **If on batteries, it will turn off and you will *not* be able to turn it back on again via the remote.** This is a feature designed to keep from running the batteries completely down. When "off", but powered via USB, the unit keeps the IR receiver powered so that it can work. On batteries, the IR receiver is also powered down, so it cannot be used to wake the unit!

Note: Depending on how you ordered the snowflake, the remote may not have come with a battery. This is because of complexities in shipping regulations for lithium batteries. If you did not get a battery, contact dave@southberkeleyelectronics, and we'll send you one!

Power

The snowflake can be powered either of two ways:

- by three AAA batteries
- by a USB power supply with a micro-USB connector

To use battery power, simply insert ***fresh*** batteries into the rear holder, observing the polarity marked. You can also power the snowflake from just about any USB charger with a micro-USB connector. Just plug it into the plug on the back. This can be nice if you want to hang the snowflake in a tree and you already have an AC cord in there. The device will use the USB power if is available, otherwise it will revert to the batteries. There is no need to remove the batteries while plugged into USB.

> *Note*: The micro-USB connector is more delicate than we'd like. Use care when removing a power supply from the connector.

Low Batteries

If the batteries are getting low, the unit might malfunction and do strange things. To avoid this, the voltage is monitored. If it gets too low, the unit will turn off the main LEDs and flash a small, red LED marked "LowBatt" for about 20 sceonds, then turn off.

> *Note***:** "sound mode" (described further below) really likes fresh batteries. Partly depleted batteries that do not trigger the low battery warning may still cause the sound sensitive mode not to work. The solution is USB power or a fresh set of batteries.

Turning "Off"

Turn-odd behavior is slightly different depending on whether the snowflake is running on batteries or USB:

- When running on batteries and it turns off (either because you turned it off, or it turned itself off after the pre-set time), it will turn *completely* off in order to minimize battery drain. ***It cannot be turned back on via the remote control.***

- When has been running from USB power, turning the snowflake off will put the unit into a low-power state that keeps the infrared receiver powered up. This uses just a tiny bit of power, but allows the remote control to turn the unit back on.

Storage & Battery Life

When the device is in the “off” state, battery draw will be quite low, but it is not zero. (For those playing at home: it’s about 85 µA, mostly through D91/2, R93, R94, I think.) For the period a Christmas tree is up, this should not drain your batteries very much, but over a year in storage, it surely will. Nobody likes leaky and dead batteries, so I ***strongly*** recommend that you remove the batteries between holidays, in order to avoid leaky ickiness and potential damage to the battery holder.

Technical Stuff for Haxx0rs

This board comes ready to use as a fun Christmas ornament or toy, and you can enjoy it just like that. But if you want to write code to make a new cool pattern or just want to hack on it a bit, you can do that, too!

The board comes configured with the Arduino Bootloader already installed, and has an FTDI-style serial port. (You'll need to install a header.) So you can just hook up a serial cable and the Arduino IDE and reprogram to your heart's content.

> *Note*: The USB port does not support serial programming. You need to use the FTDI serial header.

We don't have the resources to provide a complete Arduino tutorial here, but if you are already familiar with the Arduino ecosystem, the technical details you'll need to know are below. These should be sufficient if you are already familiar with the Arduino family and just need to know what is "special" about this board.

Arduino Setup

We did all the development for this project using the Arduino 1.8.9 environment under Linux. We suspect, but cannot guarantee that any 1.8.x+ version should work on any platform that runs Arduino.

In order to program this board you will need to take a few additional setup steps.

Step 0:

If you intend to use a *hardware programmer* (ICSP) with this board and the Arduino IDE. You'll want to first solder in the 2x3 ICSP header and then create an entry in `boards.txt` that matches this device.This file is usually found in:

```
<install>/arduino-1.8.9/hardware/arduino/avr/boards.txt
```

This is so that the IDE understands it is dealing with an 8 MHz board using the internal oscillator.

If you are going to do only serial uploads using the bootloader, you can skip this. But if you plan to use the ICSP programmer and/or need to burn the bootloader again, this is important.

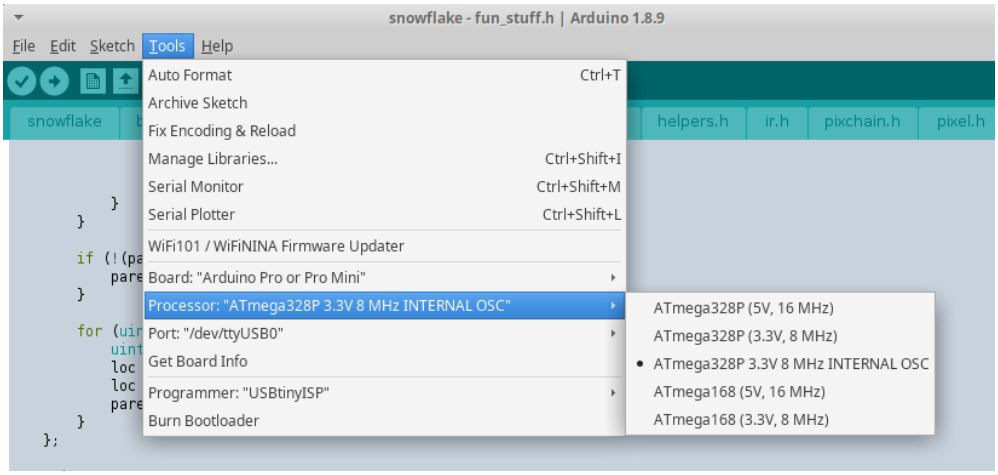
I simply took the entry for an 8 MHz Arduino Pro Mini and made the fuse changes to use the internal oscillator:

The additional lines are as below:

```
## RAW 328p internal clock
## ------------------------------------------------
pro.menu.cpu.8MHzatmega328int =ATmega328P 3.3V 8 MHz INTERNAL OSC

pro.menu.cpu.8MHzatmega328int.upload.maximum_size=30720
pro.menu.cpu.8MHzatmega328int.upload.maximum_data_size=2048
pro.menu.cpu.8MHzatmega328int.upload.speed=57600
pro.menu.cpu.8MHzatmega328int.bootloader.low_fuses=0xE2
pro.menu.cpu.8MHzatmega328int.bootloader.high_fuses=0xD8
pro.menu.cpu.8MHzatmega328int.bootloader.extended_fuses=0xFF
pro.menu.cpu.8MHzatmega328int.bootloader.file=atmega/ATmegaBOOT_168_atmega328_pro_8M
Hz.hex
pro.menu.cpu.8MHzatmega328int.build.mcu=atmega328p
pro.menu.cpu.8MHzatmega328int.build.f_cpu=8000000L
```

After making this change, restart the Arduino IDE and choose “ATmega328P 3.3V 8 MHz INTERNAL OSC” from the board menu:



Alternatively, if you only plan on using the normal Arduino upload over a serial cable, then you will want to solder in a 6x1 header for JSRL, and you can skip editing `boards.txt` and just choose Atmega328P (3.3V 8 MHz), which is already in the default `boards.txt`.

Step 1:

If you want to use the IR receiver, you will need an IR library. I've been using IRLib2. Follow the instructions on their site: https://github.com/cyborg5/IRLib2. Other similar libraries will also probably work. Use what you like.

Example Sketch and Schematics

A full schematic, some additional documentation, and a sample firmware are all available at https://github.com/djacobow/snowflake.

PCB layout / gerbers and the complete sketch are not available.We want to make this project as open as possible, but it is not an open-source product. The shape is our own intellectual property and we want to have some minimal protection against *commercial* copying. If you would like to make this board yourself, or need to reprogram that original sketch after overwriting it, we will send you the necessary files on an individual basis and the condition that you do not share them with others. Simply contact dave@southberkeleyelectronics.com.

Headers and connections

The board has several headers to enable you to use it in your own projects. Several, but not all Arduino pins are brought out to headers. This makes it more limited than a real Arduino but infinitely more flexible than your average holiday decoration.

You will notice that the headers are not populated with pins or sockets. For most people who will not hack their board, this makes the board more comfortable in the hand. Any ordinary 0.1" pins/sockets can be used. If you order directly from me on Tindie, I will include these by request, but you'll have to solder them yourself.

Headers provided for hacking use

The following header locations make it easy to use some of the Atmega328p pins for your own projects/purposes. Note that some of these pins are also shared with peripherals on the board.

Header Connection	Purpose / Description
JPGM	A standard Atmel 6-pin ICSP connector. If you have a hardware programmer such as an AVRISP, you can program the Atmega328p directly with this header.

<table>
<tr><td></td><td>This connector does not come with a pin header installed; you will have to solder one in.

You can use this for programming, but of course, you can use it to access the SPI signals, too. D10, traditionally used as an SPI “SS” pin is also available on JX2.</td></tr>
<tr><td>JX1</td><td>This header provides access to four signals, plus power and ground.

The top pin is ground, followed by “A2”, “A3”, “SDA”, and “SCL”. The bottom pin is power. This will be about 4-4.5 V if running off batteries, or 5V if running from USB power.

Note: that A2 and A3 are shared with the snowfake’s power controls (see reserved pin table below). You can use these to sense whether the snowflake is on or not. You probably do not want to use these for general purpose IO.)</td></tr>
<tr><td>JSRL</td><td>This is the “standard” 6 pin serial connection as featured in the “original” FTDI cables. You can plug such a cable in directly and access the serials lines for debug or programming.

Notes:

The pin order matches a real FTDI cable. Black is the square pin at the top near the header marking, green at the bottom.

<table>
<tr><th>Cable</th><th>Header</th><th>Color</th></tr>
<tr><td>ground</td><td>ground</td><td>black</td></tr>
<tr><td>cts/</td><td>No connection</td><td>brown</td></tr>
<tr><td>Vcc</td><td>No connection</td><td>red</td></tr>
<tr><td>Tx from cable</td><td>Rx to board</td><td>orange</td></tr>
<tr><td>Rx to cable</td><td>Tx from board</td><td>yellow</td></tr>
<tr><td>dtr/</td><td>dtr/ (reset)</td><td>green</td></tr>
</table>

CTS/ is not connected, which is just like any other Arduino board. Vcc is also not connected. This was an intentional choice to avoid having the serial cable try to power the board. Instead, power the board through batteries the USB connector at the back, or JPWR.

Cheap CH340 adapters work as well as a real FTDI connector.

Because the internal oscillator of an Atmega is not particularly</td></tr>
</table>

	precise and these boards have no crystal, you may have trouble using the serial port at very high speeds. If you get garbage from your serial port, consider lowering the speed to 38400 or even 19200. I have not seen any boards that do not work at 19200. Note that the arduino bootloader is installed but if you program with an ICSP programmer on JPGM, the bootloader will be overwritten, so serial programming will not work again until you re-burn the bootloader. It is this step that really requires that modified `boards.txt` as described above.
JX2	Provides ground at the top, and Vcc at the bottom, and access to pins D10, D7, D6, and D5.
JUSB	This connector is only for 5V power over USB.
JPWR	A 3 pin header with access to always on power, switched power, and ground

Clock

The snowflake is configured to use the internal 8 MHz of the Atmeg328p, and the clock divider is set to divide by 1, so the Atmega runs at 8 MHz. This is adequate for most purposes, but if you want to run at 16 or 20 MHz, or if you want a more accurate clock, pads for Y301, C303, and C304 are available to use an external crystal and its associated capacitors. You will also obviously have to set the appropriate fuse bits on the Atmega to enable this. We have tested this functionality and it works.

> *Be forewarned*: if you set the fuse to use an external crystal and such a crystal is not present or damaged, you will have a difficult time coaxing the Atmega to respond at all!

Reserved IO Pins

Reserved Pins and Their Use
The following pins of the Atmega328p are connected to hardware on the board, so you generally cannot use them for any purpose other than what it says they do here. This table will help you program the board.

Pin	header	Use

A3	JX1	This pin controls power to the IR receiver. It is connected directly to the VCC pin of the IR module.
		
		To turn the IR module on, I suggest:
		`pinMode(A2,OUTPUT);`
		`digitalWrite(A2,HIGH);`
		
		To turn it off, I suggest:
		`digitalWrite(A2,LOW);`
		`pinMode(A2,INPUT);`
		
		For reference, the IR receiver is a TSOP77238 or TSOP77438.
		https://www.vishay.com/docs/82471/tsop772.pdf

A2	JX1	This pin controls power to most of the circuit: LEDs, microphone, opamp, associated resistors. These all would draw quiescent power if they could not be turned off. The use of this pin is inverted. Driving it LOW turns the power **on**. Driving it HIGH or turning the pin to an input turns the power **off**.
A1	n/a	The ambient light sensor is connected to this pin. Brighter light causes this pin to be pulled higher. You can read it directly with analogRead(). *Do not set this pin for use as an output*; you may damage the phototransistor. The light sensor is a TEMT6000X01 pulled up through a 51k resistor. http://www.vishay.com/docs/81579/temt6000.pdf
A0	n/a	The sound detection circuit is connected to this pin. The sound detection circuit uses an operational amplifier configured as a precision rectifier followed by a low-pass filter to convert sound into a loudness envelope. You can read this pin with `analogRead(A0)` to get an overall loudness sense. Because of limited gain in the analog stuff, you will generally not get full scale readings. The microphone is from Knowles: https://www.mouser.com/datasheet/2/218/knowles_03312017_SPW0442HR5H-1%20Datasheet%20Rev%20E-1173676.pdf. The opamp is a generic LM358 equivalent.
D2	n/a	This is the input from the TSSOP77x38 IR receiver module. The IR transmitter included with the kit uses 32-bit NEC codes. However, I have tried this detector with various 38 kHz remotes, and they all seemed to work. However, the included sketch only

		understands NEC and Sony numeric codes.
D3	n/a	This pin is connected to the LowBatt LED. Driving it high turns the LED on. You could also use this as a general-purpose debug LED.
D4	n/a	This pin drives the data-input of the LED chain. The LEDs are like the Adafruit NeoPixels. They are World Semi WS2812C's: http://www.world-semi.com/Certifications/details-116-4.html. You can use the NeoPixel Library (https://github.com/adafruit/Adafruit_NeoPixel) to drive these pixels. In my sketch I do not use the Adafruit code directly. Instead, I cribbed what I needed from their library into my own simpler code.
D8	n/a	This is connected to the mode button. This button is wired to pull the pin down, and there is no external pullup resistor. For it to work, you need to configure the pin as INPUT_PULLUP.
D9	n/a	This is the pattern button. This button is wired to pull the pin down, and there is no external pullup resistor. For it to work, you must configure the pin as INPUT_PULLUP.

How does this work? / Theory of Operation

The snowflake is actually a very simple device in principle, but it uses some sophisticated components.

The LEDs

Probably the most interesting are the color LEDs themselves. They are WS2812C LEDs from a Chinese company called World Semiconductor.

What's interesting about these parts is that they contain, in one small package, not only separate red, green, and blue emitters, but also a controller chip that is capable of driving those three colors to any of 256 different brightness levels each. The only pins on these LED chips are power, ground, data-in, and data-out. The way you communicate with those chips is by "shifting in" 24 bits (8 bits for each color) one at a time. Once you pause, the LED chips use that data to drive the LEDs at a certain brightness, until you shift in something else. The LED pins also have an output pin, whereby everything you shift in is shifted out, but 24 bits later. This lets you chain a bunch of WS2812Cs up, so that you can still control them with a single pin. In the snowflake, we have 30 such LEDs arranged in a chain, connected to a single pin of the microcontroller chip.

The software running on the microcontroller chip, then, simply needs to determine what colors it wants in all the LEDs, and then shift out 3*8*30 (720) bits of information, every time it wants to redraw the

pattern. Simple! Well, a complicating factor is that there is no “clock” reference to help with synchronizing this serial data, so the timing of the data pulses themselves is critical.

A lot of folks who play with LEDs may recognize the more common WS2812B rather than the WS212C used here. The difference is that the “C” are less bright and consume less current. They’re better for a battery-powered device like this.

The Sound

Of course, the snowflake has some other tricks up its sleeve. It can respond to sound. This is accomplished by way of a very small micromachine (MEMS) microphone (labeled MIC30) on the board. This is similar to the microphones found in phones. It generates a very small signal in response to sound. A dual op-amp chip (U30) amplifies and rectifies this signal in two steps. The rectified (positive-going only) output is then filtered so that the sound is converted into a sort of “loudness” waveform, and that is sent to one of the microcontroller’s input pins where an analog-to-digital converter (ADC) converts it into data which the program can use. A similar process occurs for the light sensor, except no op-amp is required.

The Remote

The unit can respond to infra-red signals from a remote control. For this, an infra-red diode attached to some conditioning electronics (U70) receives IR signals and decodes them into a simple bit stream, which goes to a microprocessor pin for decoding. A single IR frame takes some time to transmit, and that can be longer than the idle time between LED updates. To avoid lost key-presses, LED updates are skipped while an IR code reception is in progress.

Power

The microcontroller on the snowflake is capable of going into a very deep sleep mode where it draws only a few 10’s of nano-amps. Compared to the self-discharge rate of batteries, this is basically nothing. However, other parts on the board are a bit more power hungry. The LEDs draw a few 10’s if micro-amps of current, even when they are not outputting light. Multiplied by 30, this is enough to drain batteries over a few weeks. Also, the opamp in the sound circuit and the infrared remote detector all draw power.

To deal with this, the snowflake has a power transistor that the microcontroller can turn on and off to power up the other devices on the board. The LEDs, opamp, microphone, and a few other sundries are on this controllable power rail. The IR receiver for the remote is powered separately, directly through an output pin of the microcontroller. This lets the snowflake enter a semi-low power mode, where the LEDs are powered down, but the snowflake will still respond to remote IR codes.

One last complexity in managing power is making sure that USB power does not try to “back charge” any batteries. At best, it harms most batteries to try to charge them, and at worse, it could be

dangerous (they could leak or get hot). So, there is a circuit on the snowflake that essentially disconnects the batteries from the rest of the snowflake when USB power is present.

Warranty

This snowflake comes with a limited warranty from South Berkeley Electronics. It covers any defects in material or workmanship under normal use for one full year from the date of purchase. During the one-year warranty period, South Berkeley Electronics will repair or replace the snowflake at no charge. Contact us directly for warranty issues.