Animated Flying Toaster OLED Jewelry

Created by Phillip Burgess

https://learn.adafruit.com/animated-flying-toaster-oled-jewelry

Last updated on 2023-08-29 02:38:56 PM EDT
## Table of Contents

### Overview
- Prerequisite Guides
- Parts
- Tools & Supplies

### 3D Printing
- FDM 3D Printing
- Slicing Software
- PLA or ABS Material

### Code
- The Old Way...
- The Code

### Assembly
- Cut Traces
- Add slide switch
- Measure Trinket wires
- Bend wires into shape
- Bend wires tips
- Vibration Sensor
- Bend Vibration sensor pins
- Add Battery
- Insert Circuit into case
- Insert Slide Switch
- Mount monitor
- Back cover
- Wear it!
Overview

The flying toaster screen saver from After Dark makes great inspiration for a retro project using 3d printing, and DIY electronics. In this project we'll show you how to make very cute and adorable electronic jewelry.

You can get the parts to build this project from the shop on adafruit dot com. The adafruit pro trinket is a micro controller perfect for miniture projects and wearables. Combined with the lipoly backpack, you can power and recharge this circuit over micro USB.

Prerequisite Guides

We recommend walking-through the following tutorials below before starting this project. These guides will help you get familiar with the components and get setup and configured with the Arduino IDE and libraries.

- Introducing Pro Trinket()
- Adafruit Pro Trinket LiPoly/Lilon Backpack()
- Monochrome OLED Breakouts()

Parts

We have all the lovely components and tools to build this project. Be sure to check out the features products on the right sidebar.

- Pro Trinket (http://adafruit.it/2000) 5V 16MHz version
- OLED Display()
• Trinket Lilon/LiPoly Backpack Add-On ()
• Fast Vibration Sensor Switch (Easy to trigger) ()
• Breadboard-friendly SPDT Slide Switch (http://adafru.it/805)

Tools & Supplies

You'll need a couple of hand tools and accessories to assist you in the build.

• Solder Iron (http://adafru.it/1204)  +  Solder (http://adafru.it/734)
• Silicone Wire (http://adafru.it/1877)
• PLA Filament ()
• 3D Printer ()

3D Printing

FDM 3D Printing

These parts are optimized to print with desktop 3D Printers capable of printing in ABS or PLA material with a minimum build area of 100mm x 100mm x 90mm. The two parts are designed to print without any support material.
Slicing Software

The recommend settings above should work with most slicing software. However, you are encouraged to use your own settings since 3D printers and slicing software will vary from printer to printer.

PLA or ABS Material

We recommend using PLA material for an easier print with high quality. The tolerance has been tested with PLA filament but should also work with ABS. The parts do not require any support material or a raft.
Code

Using the OLED display with Arduino sketches requires that two libraries be installed: Adafruit_SSD1306, which handles the low-level communication with the hardware, and Adafruit_GFX, which builds atop this to add graphics functions like lines, circles and text.

In recent versions of the Arduino IDE software (1.6.2 and later), this is most easily done through the Arduino Library Manager, which you’ll find in the “Sketch” menu: Sketch→Include Library→Manage Libraries...

Enter “ssd1306” in the search field, locate the Adafruit SSD1306 library and select “Install” (or “Upgrade” if you have an older version). Then repeat the same for “gfx” and the Adafruit GFX library, “busio” and the Adafruit BusIO library (newer Arduino IDE releases will handle this dependency automatically when installing GFX).

The Old Way...

If you’re using an earlier version of the Arduino IDE software and need to install libraries manually, or if you’d like to download the library source code to look inside, both of these can be found on Github.

- Adafruit_SSD1306 library repository ()
- Adafruit_GFX library repository ()
- Adafruit_BusIO library repository ()

Or here are links directly to the ZIP files:
After uncompressing, the folders should be renamed (if necessary) to Adafruit_SSD1306, Adafruit_GFX and Adafruit_BusIO. The first should contain the files Adafruit_SSD1306.cpp and Adafruit_SSD1306.h (plus a few extra files). Sometimes unZIPping creates a nested Adafruit_SSD1306 folder within another folder…you don’t want that. The second should contain Adafruit_GFX.cpp and Adafruit_GFX.h (plus some extra files) and so forth.

Place these folders in your (home)/Documents/Arduino/Libraries folder. You may need to create the libraries subfolder if it’s your first library. Then restart the Arduino IDE.

We also have a great tutorial on Arduino library installation here: http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

The Code

As written, this will work on an Adafruit Pro Trinket or an Arduino Uno. It makes reference to some specific pins and hardware features (sleep, interrupts, etc.) that may not work on other boards without some code changes and deeper understanding of the hardware.

```c
// Animated pendant for Adafruit Pro Trinket and SSD1306 OLED display,
// inspired by the After Dark “Flying Toasters” screensaver.
// Triggered with vibration switch between digital pins 3 and 4.

#include <avr/sleep.h>
#include <avr/power.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "bitmaps.h" // Toaster graphics data is in this header file

#define EXTRAGND   4  // Extra ground pin for vibration switch
#define OLED_DC    6  // OLED control pins are configurable.
#define OLED_CS    5  // These are different from other SSD1306 examples
#define OLED_RESET 8  // because the Pro Trinket has no pin 2 or 7.
// Hardware SPI for Data &amp; Clk -- pins 11 &amp; 13 on Uno or Pro Trinket.
Adafruit_SSD1306 display(128, 64, &amp;SPI, OLED_DC, OLED_RESET, OLED_CS);

#define N_FLYERS   5  // Number of flying things

struct Flyer {       // Array of flying things
  int16_t x, y;      // Top-left position * 16 (for subpixel pos updates)
};
```
```c
int8_t depth;     // Stacking order is also speed, 12-24 subpixels/frame
uint8_t frame;     // Animation frame; Toasters cycle 0-3, Toast=255
}
flyer[N_FLYERS];

uint32_t startTime;

void setup() {

    randomSeed(analogRead(2));           // Seed random from unused analog input
    DDRB  = DDRC  = DDRD  = 0x00;        // Set all pins to inputs and
    PORTB = PORTC = PORTD = 0xFF;        // enable pullups (for power saving)
    pinMode(EXTRAGND, OUTPUT);           // Set one pin low to provide a handy
    digitalWrite(EXTRAGND, LOW);         // ground point for vibration switch
    display.begin(SSD1306_SWITCHCAPVCC); // Init screen
    display.clearDisplay();
    for(uint8_t i=0; i<N_FLYERS; i++) {  // Randomize initial flyer states
        flyer[i].x     = (-32 + random(160)) * 16;
        flyer[i].y     = (-32 + random(96)) * 16;
        flyer[i].frame = random(3) ? random(4) : 255; // 66% toaster, else toast
        flyer[i].depth = 10 + random(16);             // Speed / stacking order
    }
    qsort(flyer, N_FLYERS, sizeof(struct Flyer), compare); // Sort depths

    // AVR peripherals that aren't used by this code are disabled to further
    // conserve power, and may take certain Arduino functionality with them.
    // If you adapt this code to other projects, may need to re-enable some.
    power_adc_disable();    // Disable ADC (no analogRead())
    power_twi_disable();    // Disable I2C (no Wire library)
    power_usart0_disable(); // Disable UART (no Serial)
    power_timer1_disable();
    power_timer2_disable();

    EICRA = _BV(ISC11); // Falling edge of INT1 (pin 3) generates an interrupt
    EIMSK = _BV(INT1);  // Enable interrupt (vibration switch wakes from sleep)
    startTime = millis();
}

void loop() {

    uint8_t i, f;
    int16_t x, y;
    boolean resort = false;     // By default, don't re-sort depths

    display.display();          // Update screen to show current positions
    display.clearDisplay();     // Start drawing next frame

    for(i=0; i<N_FLYERS; i++) { // For each flyer...

        // First draw each item...
        f = (flyer[i].frame == 255) ? 4 : (flyer[i].frame++ & 3); // Frame #
        x = flyer[i].x / 16;
        y = flyer[i].y / 16;
        display.drawBitmap(x, y, (const uint8_t *)pgm_read_word(&mask[f]), 32, 32,
                            BLACK);
        display.drawBitmap(x, y, (const uint8_t *)pgm_read_word(&img[f]), 32, 32,
                            WHITE);

        // Then update position, checking if item moved off screen...
        flyer[i].x -= flyer[i].depth * 2; // Update position based on depth,
        flyer[i].y += flyer[i].depth;     // for a sort of pseudo-parallax effect.
        if((flyer[i].y &gt;= (64*16)) || (flyer[i].x &lt;= (-32*16))) { // Off screen?
            if(random(7) &lt; 5) {         // Pick random edge; 0-4 = top
                flyer[i].x = random(160) * 16;
                flyer[i].y = -32           * 16;
            } else {                    // 5-6 = right
                flyer[i].x = 128          * 16;
                flyer[i].y = random(64)   * 16;
            }
        }
        flyer[i].frame = random(3) ? random(4) : 255; // 66% toaster, else toast
    }

    // AVR peripherals that aren't used by this code are disabled to further
    // conserve power, and may take certain Arduino functionality with them.
    // If you adapt this code to other projects, may need to re-enable some.
    power_adc_disable();    // Disable ADC (no analogRead())
    power_twi_disable();    // Disable I2C (no Wire library)
    power_usart0_disable(); // Disable UART (no Serial)
    power_timer1_disable();
    power_timer2_disable();

    EICRA = _BV(ISC11); // Falling edge of INT1 (pin 3) generates an interrupt
    EIMSK = _BV(INT1);  // Enable interrupt (vibration switch wakes from sleep)
    startTime = millis();
}
```

---

© Adafruit Industries  Page 8 of 19
flyer[i].depth = 10 + random(16);
resort = true;
}
// If any items were 'rebooted' to new position, re-sort all depths
if(resort) qsort(flyer, N_FLYERS, sizeof(struct Flyer), compare);

if((millis() - startTime) &gt;= 15000L) {   // If 15 seconds elapsed...
display.ssd1306_command(SSD1306_DISPLAYOFF);   // Screen off
power_spi_disable();   // Disable remaining periphs
power_timer0_disable();
set_sleep_mode(SLEEP_MODE_PWR_DOWN);   // Deepest sleep
sleep_mode();
// Execution resumes here on wake.
power_spi_enable();   // Re-enable SPI
power_timer0_enable();   // and millis(), etc.
display.ssd1306_command(SSD1306_DISPLAYON);   // Main screen turn on
startTime = millis();   // Save wake time
}

// Flyer depth comparison function for qsort()
static int compare(const void *a, const void *b) {
   return ((struct Flyer *)a)-&gt;depth - ((struct Flyer *)b)-&gt;depth;
}
ISR(INT1_vect) {} // Vibration switch wakeup interrupt

The graphics data is stored in a separate source file. Use the “new tab” button (near top right of Arduino editor window), name the new file “bitmaps.h,” then copy and paste the following into it:

#include &lt;Arduino.h&gt;

const uint8_t PROGMEM toastermask0[] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00,
0x00, 0x20, 0x00, 0x00, 0x00, 0x88, 0x00, 0x00,
0x00, 0x10, 0x00, 0x00, 0x00, 0x43, 0x08, 0x00,
0x00, 0x1F, 0x80, 0x00, 0x00, 0x7C, 0x3F, 0x80,
0x01, 0xF1, 0xF8, 0xF0, 0x07, 0xC7, 0xC7, 0x00,
0x0F, 0x1F, 0x08, 0x2B, 0x1E, 0x7C, 0x11, 0x00,
0x04, 0xF0, 0x20, 0x56, 0x01, 0xE4, 0x22, 0x00,
0x78, 0x48, 0x40, 0xA8, 0x4E, 0x10, 0x44, 0x00,
0x53, 0x90, 0x81, 0x00, 0x5C, 0xA0, 0x90, 0x58,
0x7C, 0xA1, 0x07, 0x90, 0x74, 0xA1, 0x21, 0x38,
0x7F, 0xB5, 0x0E, 0x30, 0x77, 0xB0, 0x90, 0x78,
0x7F, 0xB4, 0x07, 0xF0, 0x77, 0xBB, 0x03, 0xE0,
0x7F, 0xBC, 0x0F, 0xC0, 0x77, 0xBF, 0xBF, 0x00,
0x7F, 0xBF, 0xFC, 0x00, 0x3F, 0xBF, 0xF0, 0x00,
0x1F, 0xBF, 0xC0, 0x00, 0x07, 0xBE, 0x00, 0x09,
0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
toastermask1[] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
toastermask2[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x03, 0x0F, 0x08, 0x00,
  0x00, 0x1F, 0x80, 0x00, 0x00, 0x07, 0x7C, 0x00,
  0x01, 0xF1, 0xF8, 0x00, 0x07, 0xC7, 0xC1, 0xE0,
  0x0F, 0x1F, 0x06, 0x10, 0x1E, 0x7C, 0x00, 0x00,
  0x04, 0xF0, 0x00, 0x00, 0x61, 0xE4, 0x00, 0x08,
  0x7B, 0x48, 0x00, 0x10, 0x4E, 0x10, 0x00, 0x18,
  0x53, 0x90, 0x60, 0x70, 0x5C, 0xA0, 0x9F, 0xC8,
  0x7C, 0xA1, 0x04, 0x92, 0x74, 0xA1, 0x09, 0x24,
  0x7F, 0xB5, 0x02, 0x48, 0x77, 0xB0, 0x80, 0x10,
  0x7F, 0xB4, 0x41, 0x00, 0x77, 0xBB, 0x20, 0x40,
  0x7F, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0xB0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
},

toastmask[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xC0, 0x00,
  0x00, 0x0C, 0x30, 0x00, 0x00, 0x30, 0x0C, 0x00,
  0x00, 0xC2, 0x23, 0x00, 0x07, 0x00, 0x80, 0xC0,
  0x08, 0x25, 0x50, 0x00, 0x10, 0x0B, 0xA8, 0x08,
  0x21, 0x37, 0xF5, 0x04, 0x30, 0x10, 0x0B, 0xA8,
  0x3C, 0x15, 0x52, 0x34, 0x02, 0x00, 0x80, 0xE0,
  0x3F, 0xC2, 0x23, 0x54, 0x1A, 0x00, 0x80, 0xE0,
  0x0D, 0x5C, 0x15, 0x58, 0x03, 0xA8, 0xA6, 0x0E,
  0x00, 0xD5, 0x55, 0x80, 0x00, 0x3A, 0xA6, 0x00,
  0x00, 0xD0, 0x58, 0x00, 0x00, 0x03, 0xA6, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
},

toaster0[] = {
  0x00, 0x30, 0x00, 0x00, 0x00, 0xE8, 0x00, 0x00,
  0x01, 0xD8, 0x00, 0x00, 0x03, 0x74, 0xF0, 0x00,
  0x07, 0xEF, 0xFE, 0x00, 0x07, 0xBC, 0x07, 0x80,
  0x0F, 0xE0, 0x7F, 0xE0, 0x07, 0x83, 0xC0, 0x70,
  0x1E, 0x0E, 0x07, 0x08, 0x18, 0x38, 0x38, 0xFF,
  0x30, 0xE0, 0xF7, 0xD4, 0x61, 0x83, 0xEE, 0xFF,
  0x7B, 0x0F, 0xFD, 0xA8, 0x9E, 0x1B, 0x80, 0xDE,
  0x87, 0xB7, 0xBF, 0x50, 0xB1, 0xEF, 0xBB, 0xF8,
  0x43, 0xF0, 0x20, 0x88, 0x44, 0xFC, 0x10, 0x80,
  0x4B, 0x9F, 0x08, 0x88, 0x44, 0xFC, 0x10, 0x80,
  0x43, 0xF0, 0x20, 0x88, 0x44, 0xFC, 0x10, 0x80,
  0x4E, 0xF0, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x00,
},
toaster1[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x0F, 0x0E, 0x07, 0x0E, 0x07, 0x0E, 0x07,
  0x03, 0x0E, 0x07, 0x0F, 0x07, 0x0E, 0x07, 0x0E,
  0x0E, 0x0F, 0x07, 0x0F, 0x07, 0x0E, 0x07, 0x0E,
  0x30, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
  0x87, 0xB7, 0xBF, 0x50, 0xB1, 0xEF, 0xBB, 0xF8,
  0x43, 0xF0, 0x20, 0x88, 0x44, 0xFC, 0x10, 0x80,
  0x4B, 0x9F, 0x08, 0x88, 0x44, 0xFC, 0x10, 0x80,
  0x43, 0xF0, 0x20, 0x88, 0x44, 0xFC, 0x10, 0x80,
},

©Adafruit Industries
Assembly

This diagram shows the connections to be made:
The LiPoly backpack sits atop the Pro Trinket on the BUS, G and BAT+ pins, and the power jumpers are modified to connect a switch.

The vibration sensor switch connects to Pro Trinket pins 3 and 4.

The remaining connections are:

<table>
<thead>
<tr>
<th>Pro Trinket Pin</th>
<th>OLED Display Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>DC</td>
</tr>
<tr>
<td>6</td>
<td>CS</td>
</tr>
<tr>
<td>8</td>
<td>Rst</td>
</tr>
<tr>
<td>11</td>
<td>Data</td>
</tr>
<tr>
<td>13</td>
<td>Clk</td>
</tr>
<tr>
<td>G</td>
<td>Gnd</td>
</tr>
</tbody>
</table>
However, the Pro Trinket and display are installed back to back, so the routing won't exactly follow the diagram above...one side will be in mirror-image. Also, much shorter wire lengths can be used! Here's the process...

### Cut Traces

Enable the on and off switch by cutting the trace between the power switch.

### Add slide switch

Measure wires for the slide switch and cover the jumper pad so they don't make contact with Trinket.
Measure Trinket wires

Align the monitor on top to measure how long each wire needs to be, add a bit of wiggle room to be safe.

Bend wires into shape

Adjust the wire so they sit in each through hole for monitor.
Bend wires tips

Secure the monitor in place while soldering by bending the wire tips.

Vibration Sensor

Add the vibration sensor between pins 4 and 3
Bend Vibration sensor pins

Use a pair of flat pliers to hold the end of the vibrations pins while bending them across the Trinket as shown.

Add Battery

Connect the battery using a small amount of tac to hold it in place.
Insert Circuit into case

Slide the complete circuit in at an angle, port side first as shown above.

Insert Slide Switch

Wiggle the slide switch into the clips at an angle with a pair of tweezers.
Mount monitor

Use a #4 screw to securely mount the monitor into the case.

Back cover

Align the back cover and use another #4 screw to tight to the back.
Wear it!

Add a split ring and necklace and rock out to retro goodness!