Animated Scrolling "Mario Clouds" TFT Jewelry

Created by Phillip Burgess

https://learn.adafruit.com/scrolling-mario-clouds-tft-jewelry

Last updated on 2023-08-29 02:38:57 PM EDT

Table of Contents

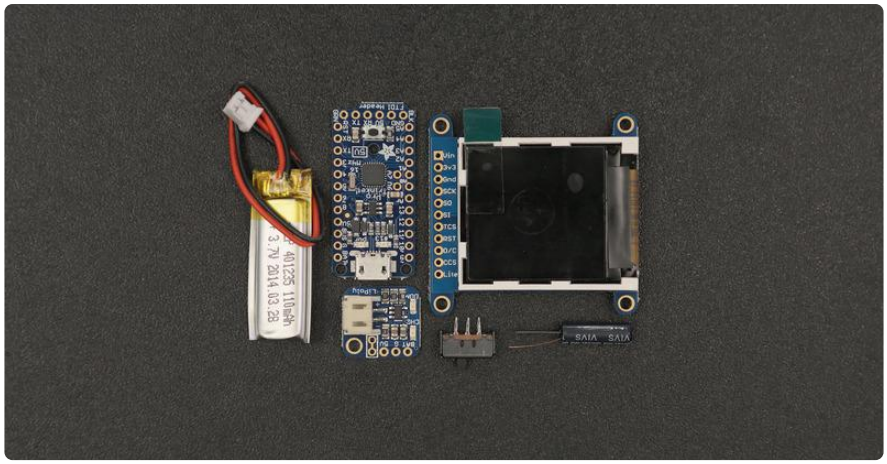
Overview

It's 8-bit retro Mario Clouds! It's so tiny and cute, you can wear it! In this project, we're making Cory Arcangel-inspired Super Mario jewelry using 3D printing and DIY electronics.

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/LiIon Backpack ()
- 1.44" Color TFT LCD Display with MicroSD Card ()



Parts

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

- Pro Trinket (http://adafru.it/2000)
- 1.44" Color TFT LCD Display with MicroSD Card breakout ()
- Trinket LiIon/LiPoly Backpack Add-On ()
- 100mAh battery ()
- 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 (http://adafru.it/2080)
- 3D Printer ()
- Panavise Jr. ()
- Helping Third Hand ()

Code

Our cloud sketch depends on the Adafruit_GFX, Adafruit_BusIO and ST7735 libraries. Library installation is a common sticking point for beginners...our All About Arduino Libraries () guide explains how this is done.

Click to download Adafruit_GFX library for Arduino

Click to download Adafruit_ST7735 library for Arduino.

Click to download Adafruit_BusIO library for Arduino

After installing the libraries, restart the Arduino IDE.

The code 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.

In order to draw the clouds scrolling nice and smoothly, we write directly to the display buffer on the TFT rather than going thru the nice Adafruit_GFX helper library.

```
// Scrolling cloud pendant for Adafruit Pro Trinket and ST7735R display.
// Inspired by Cory Arcangel's "Super Mario Clouds."
// Triggered with vibration switch between digital pins 3 and 4.

// This is NOT a good learning example for the Adafruit_GFX library!
// To achieve fast frame rates, the code does horrible irresponsible
// things, bypassing the GFX lib and issuing commands &amp;amp; data directly
// to the LCD driver.  It's hardcoded for specific control pins, not
// portable to other displays, and other such crimes.
// Look at the ST7735R library examples for better role models.

// As part of the optimization strategy, everything's drawn sideways;
// clouds scroll "up," not across.  Mount TFT rotated to compensate.

#include &amp;lt;avr/sleep.h&amp;gt;
#include &amp;lt;avr/power.h&amp;gt;
#include &amp;lt;SPI.h&amp;gt;
#include &amp;lt;Adafruit_GFX.h&amp;gt;
#include &amp;lt;Adafruit_ST7735.h&amp;gt;
#include "clouds.h"

#define TFT_CS   10 // Chip select line for TFT  DO NOT CHANGE
#define TFT_DC    8 // Data/command line for TFT DO NOT CHANGE
#define TFT_RST   6 // TFT Reset pin
#define BACKLIGHT 9 // TFT "Lite" pin
#define EXTRAGND  4 // Extra ground pin for vibration switch
                    // Other leg of vibe switch MUST go to pin 3!

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

#define N_CLOUDS 6

struct {          // Cloud structure:
  uint8_t column; // Screen position 0-3
  uint8_t reps;   // Number of middle tiles in cloud (0-3)
  int16_t y;      // Top edge of cloud * 16 (subpixel pos)
  int16_t endy;   // Reset when Y reaches this value
  int16_t prev;   // Pixel position on prior frame
} cloud[N_CLOUDS];

const uint8_t PROGMEM          // For each of the 4 cloud columns...
  xpos[] = { 4, 36, 68, 100 }, // X pixel coordinate
  inc[]  = { 7, 10, 13,  16 }; // Vertical subpixel speed (16 = 1px)
  // The pseudo-parallax scrolling isn't canon, but adds flair(tm).

uint32_t startTime;

void setup(void) {
  randomSeed(analogRead(2));           // Seed randomness from unused input
  DDRB  = DDRC  = DDRD  = 0x00;        // Set all pins to inputs and
  PORTB = PORTC = PORTD = 0xFF;        // enable pullups (for power saving)
  pinMode(BACKLIGHT, OUTPUT);
  digitalWrite(BACKLIGHT, LOW);        // Backlight off
  pinMode(EXTRAGND, OUTPUT);           // Set one pin low to provide a handy
  digitalWrite(EXTRAGND, LOW);         // ground point for vibration switch
```

```
  tft.initR(INITR_144GREENTAB);        // Init 1.44" "green tab" screen
  SPI.setClockDivider(SPI_CLOCK_DIV2); // Force 8 MHz SPI for faster refresh
  // Default rotation (0) is used.  Rotation 3 would be cooler (breakout board
  // would hang symmetrically from mounting holes), but experiencing glitches/
  // artifacts when using non-default rotations on this screen; possible
  // MADCTL/CASET/RASET register strangeness to be resolved in library.
  // So the screen must be mounted with the control pins on the left.
  // tft.setRotation(3);
  tft.fillScreen(0x6C3F);              // Sky background
  // Cloud columns are primed to nonsense value so comparisons in
  // randomize() don't have trouble with non-initialized clouds.
  for(uint8_t i=0; i&lt;N_CLOUDS; i++) cloud[i].column = 255;
  for(uint8_t i=0; i&lt;N_CLOUDS; i++) randomize(i, false);

  // 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)

  digitalWrite(BACKLIGHT, HIGH); // Backlight on
  startTime = millis();
}

void loop() {
  uint32_t t = millis();
  int16_t  y;
  uint8_t  i, x, r;

  for(i=0; i&lt;N_CLOUDS; i++) {                          // For each cloud...
    if((y = (cloud[i].y / 16)) != cloud[i].prev) {      // Has it moved?
      if(y &lt; 128) {                                    // Is it on screen yet?
        x = pgm_read_byte(&amp;xpos[cloud[i].column]);      // Horiz pos from table
        // Address window = blit destination rectangle...
        tft.setAddrWindow(x, (y &gt; 0) ? y : 0, x+23, 127);
        // Access ST7735 control pins directly...dirty hack...
        PORTB |=  _BV(0);                               // Data mode
        PORTB &amp;= ~_BV(2);                               // Chip select
        y = blit(tile_a, y);                            // Blit first tile of cloud
        for(r=0; r&lt;cloud[i].reps; r++) y = blit(tile_b, y); // Middle tiles
        blit(tile_c, y);                                // Last tile
        PORTB |= _BV(2);                                // Chip deselect
      }
      cloud[i].prev = y;                                // Record new position
    }
    cloud[i].y -= pgm_read_byte(&amp;inc[cloud[i].column]); // Move cloud (subpixel)
    if(cloud[i].y &lt; cloud[i].endy) randomize(i, true);  // Regenerate?
  }

  if((t - startTime) &gt;= 15000L) {                       // If 15 seconds
elapsed...
    digitalWrite(BACKLIGHT, LOW);                       // Backlight off
    PORTB &amp;= ~(_BV(0) | _BV(2));                        // Command + chip select
    for(SPDR = ST7735_SLPIN; !(SPSR &amp; _BV(SPIF)););     // Issue command
    PORTB |= _BV(2);                                    // Chip deselect
    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.
```

```
    PORTB &amp;= ~(_BV(0) | _BV(2));                         // Command + chip select
    for(SPDR = ST7735_SLPOUT; !(SPSR &amp; _BV(SPIF)););    // Screen on
    PORTB |= _BV(2);                                    // Chip deselect
    delay(120);                                         // Must pause after wake
    digitalWrite(BACKLIGHT, HIGH);                      // Backlight on
    startTime = millis();
  } else if((t = (millis() - t)) &lt; 30) {
    delay(30 - t);                                      // Regular-ish frame timing
  }
}

// Copy data from tile array to screen, w/some clipping.
// Address window must already be set in calling function.
int16_t blit(const uint8_t *ptr,    // -&gt; tile data (in clouds.h)
                   int16_t  y) {    // Position (topmost row) on screen
  uint8_t h = pgm_read_byte(ptr++); // Tile height
  int16_t r = y + h;                // Return value = pos. of next tile
  if((y &lt; 128) &amp;&amp; (y &gt; -h)) {       // Ignore tile clipped fully off
screen
    if(y &lt; 0) {                     // Tile clipped partially off top
      ptr += y * -48;               //  Move data pointer to first visible row
      h   += y;                     //  Reduce blit height
    } else if(r &gt; 128) {            // Tile clipped partially off bottom
      h = 128 - y;                  //  Reduce blit height
    }
    uint16_t count = h * 48;        // Number of bytes to transfer
    uint8_t  c;                     // Temp byte storage
    SPDR = pgm_read_byte(ptr++);    // Issue first byte
    while(--count) {                // Do loop control during SPI out
      c = pgm_read_byte(ptr++);     // Fetch next byte during SPI out
      while(!(SPSR &amp; _BV(SPIF)));   // Wait for SPI completion
      SPDR = c;                     // Issue next byte
    }
    while(!(SPSR &amp; _BV(SPIF)));     // Wait for last byte out
  }
  return r;                         // Starting pos. of next tile
}

// Randomize values for one cloud, making sure it doesn't overlap others
void randomize(uint8_t i, boolean offRight) {
  uint8_t j, tries = 0;
  int     iy1, iy2, jy1, jy2, maxy = 2048;

  cloud[i].reps = random(4);     // # tiles repeated in middle (0-3)
  do {
    cloud[i].column = random(4); // Randomize position...
    cloud[i].y      = 16 * (offRight ? (128 + random(64)) :
                      random((2 + cloud[i].reps) * -16, 192));
    iy1 = cloud[i].y;                               // Top of cloud i
    iy2 = iy1 + (2 + cloud[i].reps) * 256 + 15;   // Bottom of cloud i
    for(j=0; j&lt;N_CLOUDS; j++) { // Then test if it overlaps other clouds...
      if((i == j) || (cloud[i].column != cloud[j].column)) continue;
      jy1 = cloud[j].y;                             // Top of cloud j
      jy2 = jy1 + (2 + cloud[j].reps) * 256 + 15; // Bottom of cloud j
      if(jy2 &gt; maxy) maxy = jy2;                   // Track lowest cloud
      if((jy1 &lt;= iy2) &amp;&amp; (jy2 &gt;= iy1)) break;     // Overlap!
    }
  } while((j &lt; N_CLOUDS) &amp;&amp; (++tries &lt; 5));       // Retry until no
overlap
  if(tries &gt; 4) cloud[i].y = maxy + 16;           // Give up; move to bottom
  cloud[i].endy = (2 + cloud[i].reps) * -256 - 31;
  cloud[i].prev = -30000;
}

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 “clouds.h,” then copy and paste the following into it:

```
// Cloud bitmaps for scrolling pendant project.

#include &lt;Arduino.h&gt;

// Tiles are stored in a 'raw' 16-bit color format (5/6/5 R/G/B);
// no color table, no run-length encoding.  This keeps the blitting
// function simple and quick.  Each tile requires nearly 800 bytes of
// program space, but it's OK as there's not much else going on.

//        Hi   Lo
#define K 0x00,0x00 // Black
#define W 0xFF,0xFF // White
#define B 0x6C,0x3F // Blue (sky)
#define G 0x04,0xBE // Gray (cloud shadow, alt blue really)

// First byte is height of tile in pixels.

const uint8_t PROGMEM
 tile_a[] = { 16, // Left cloud tile
  B,B,B,B,B,B,B,B,K,K,B,B,B,B,B,B,B,B,B,B,B,B,
  B,B,B,B,B,B,B,B,K,W,W,K,B,B,B,B,B,B,B,B,B,B,
  B,B,B,B,B,B,B,K,W,W,W,K,B,B,B,B,B,B,B,B,B,B,
  B,B,B,B,B,B,K,W,W,W,W,W,K,B,B,B,B,B,B,B,B,B,
  B,B,B,B,K,K,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,B,
  B,B,B,K,W,W,W,G,W,W,W,W,W,W,K,B,B,B,B,B,B,B,
  B,B,B,K,W,W,G,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,
  B,B,B,K,W,G,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,
  B,B,K,W,W,G,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,
  B,K,W,W,W,G,G,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,
  B,K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,K,K,K,B,B,B,
  K,W,W,W,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,G,W,W,W,W,K,B,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,G,W,W,W,W,K,B,
  B,K,W,W,W,G,G,W,W,W,W,W,W,W,W,W,G,W,W,W,W,W,K,
  B,B,K,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K },
 tile_b[] = { 16, // Middle (possibly repeating) cloud tile
  B,K,W,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K,
  B,K,W,W,G,G,W,G,W,W,W,W,W,W,W,W,G,W,W,W,W,K,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,G,W,W,W,W,K,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K,K,B,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,B,B,
  K,W,W,W,G,W,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,B,B,
  B,K,W,W,W,G,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,B,B,
  B,K,W,W,W,W,W,W,W,W,W,W,W,W,W,K,K,K,B,B,B,B,B,
  B,B,K,W,W,G,W,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,
  B,K,W,W,W,G,G,W,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,
  B,K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,K,K,K,B,B,B,
  K,W,W,W,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,G,W,W,W,W,K,B,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,G,W,W,W,W,K,B,
  B,K,W,W,G,G,W,W,W,W,W,W,W,W,W,G,W,W,W,W,W,K,
  B,B,K,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K },
 tile_c[] = { 17, // Right cloud tile
  B,K,W,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K,
  B,K,W,W,G,G,W,G,W,W,W,W,W,W,W,W,G,W,W,W,W,K,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,G,W,W,W,W,K,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,W,K,K,B,B,
  K,W,W,G,G,W,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,B,B,
  K,W,W,W,G,W,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,B,B,
  B,K,W,W,W,G,W,W,W,W,W,W,W,W,W,W,W,W,K,B,B,B,B,
  B,K,W,W,W,W,W,W,W,W,W,W,W,W,W,K,K,K,B,B,B,B,B,
  B,B,K,W,W,W,W,W,W,W,W,W,W,W,W,K,K,B,B,B,B,B,B,
```

```
B,K,W,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,B,B,
B,K,W,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,B,B,
B,B,K,W,W,W,W,W,W,W,W,W,K,B,B,B,B,B,B,B,B,
B,B,K,W,W,W,W,W,W,W,W,K,K,K,B,B,B,B,B,B,B,B,
B,B,B,K,W,W,W,K,W,W,W,K,B,B,B,B,B,B,B,B,B,B,
B,B,B,K,W,W,K,B,K,W,W,W,K,B,B,B,B,B,B,B,B,B,
B,B,B,B,B,K,B,B,B,K,K,K,B,B,B,B,B,B,B,B,B,B,
B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B };
```

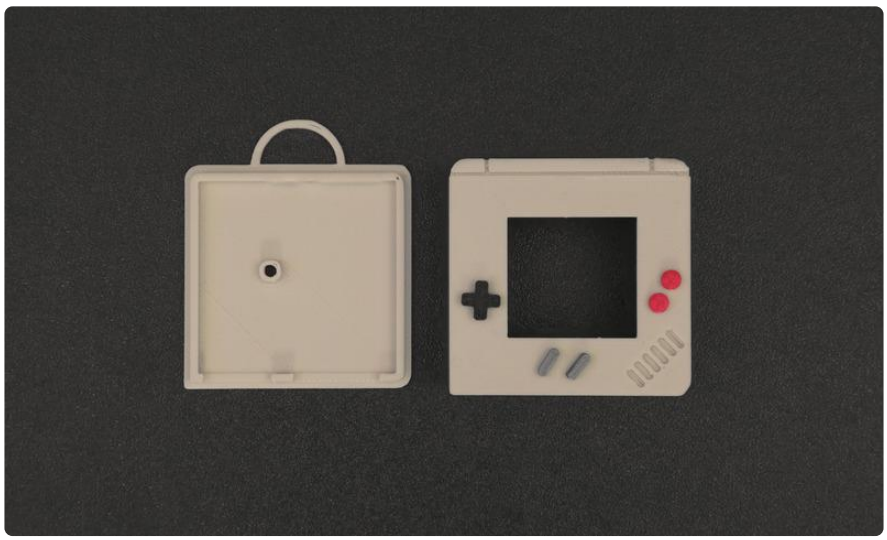
3D Printing

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 or raft.

Printing Tiny Buttons

The buttons are rather small and should be printed separately. These can be painted or printed in different colored filament. We recommend printing these pieces with slo w feed and travel rates (about 45/50mms).



Edit 3D files

Download STLs

cloudcase.stl cloudcover.stl cloudABbtn.stl cloudDiabtn.stl cloudSelbtn.stl	235c 10% infill 90 feed 120 travel no supports or rafts	Printing all five parts should take about one hour to print.

Slicing Software

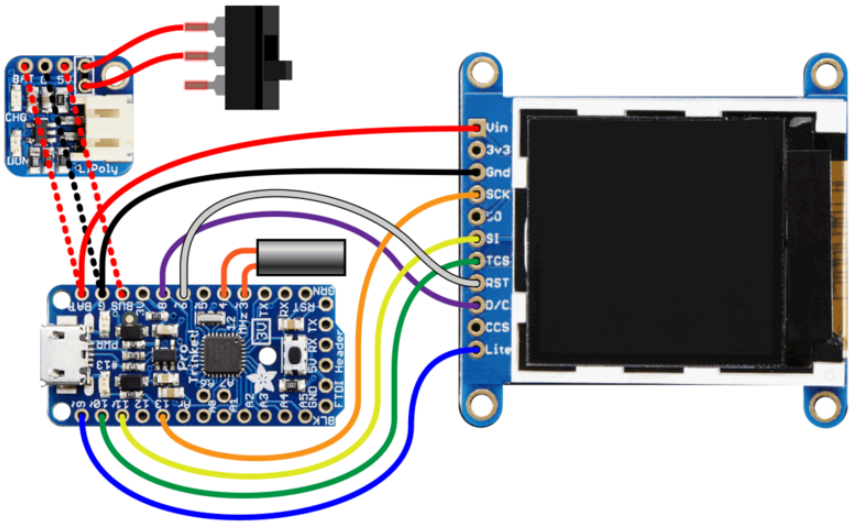
The recommend settings above should work with most slicing software. However, you are encouraged to use your own settings since 3D printers and slicing software will vary from printer to printer.

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.

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. This makes for a compact USB-rechargeable design

The vibration sensor switch connects to Pro Trinket pins 3 and 4. We use this to turn on when the necklace is 'tapped'

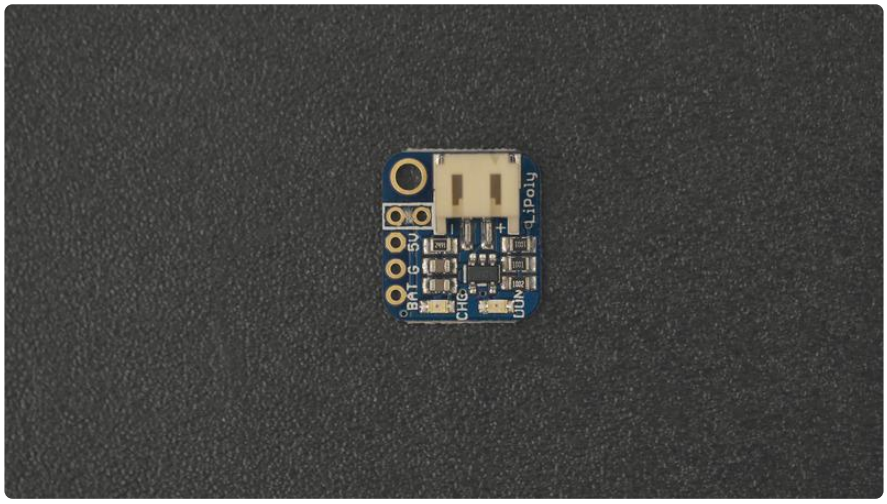
The remaining connections are:

Pro Trinket Pin	TFT Display Pin
6	RST
8	D/C
9	Lite
10	TCS
11	SI
13	SCK
G	Gnd
BAT+	Vin

However, the Pro Trinket and display are installed back to back, so the routing won't exactly follow the diagram above...one’s rotated 90 degrees with respect to the other. Also, much shorter wire lengths can be used! Here’s the process...

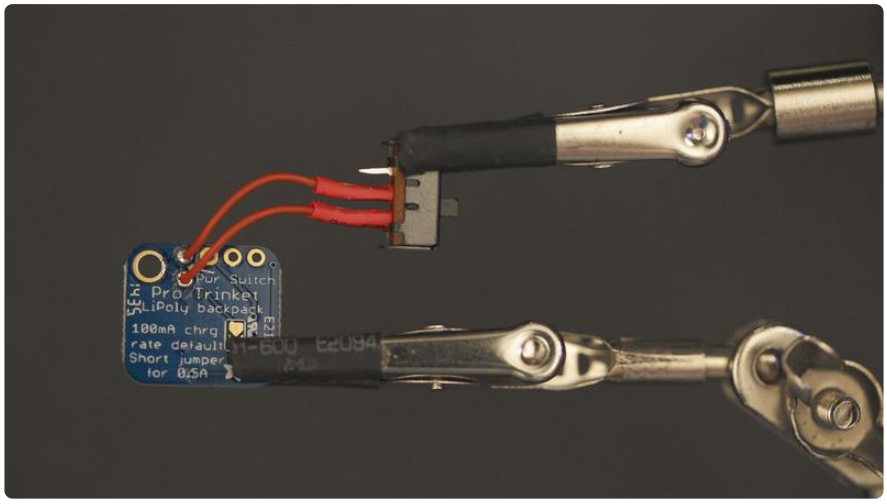
Lilon / LiPoly Backpack

Prepare the circuit to work with an on-off slide switch by carefully cutting the trace between the two 0.1" holes with a box around them (the battery output line)



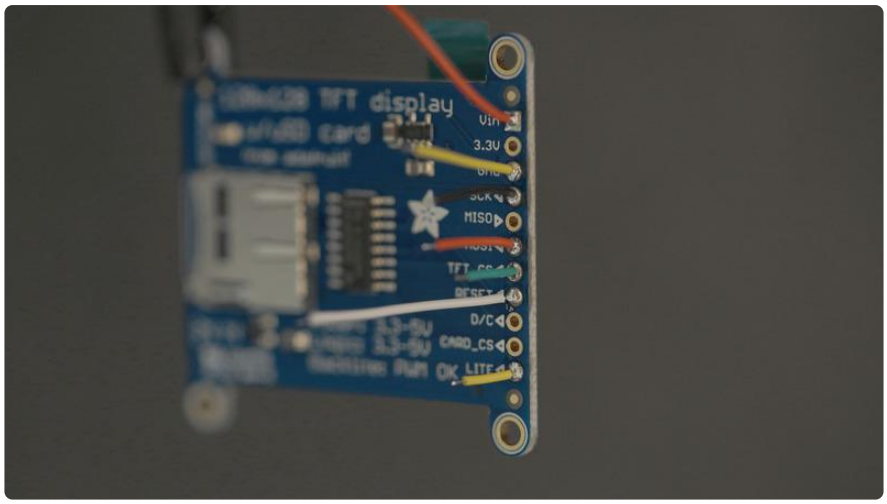
Add the On/Off Switch

Once that trace is cut, you can solder two wires from the power-switch pads to a switch (like a slide switch (), or a pushbutton (http://adafru.it/1683) one, for example)



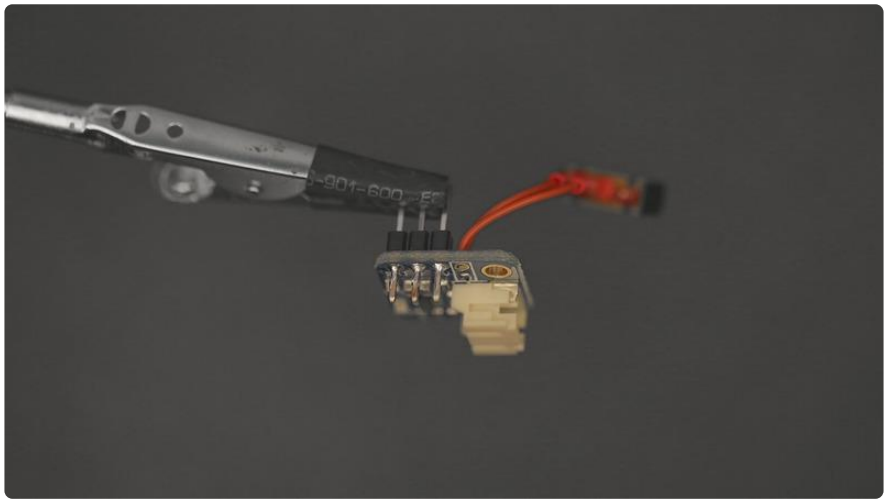
Measure wires

Align the Pro Trinket up with the display and appropriately cut each wire so that it can reach each through-hole.



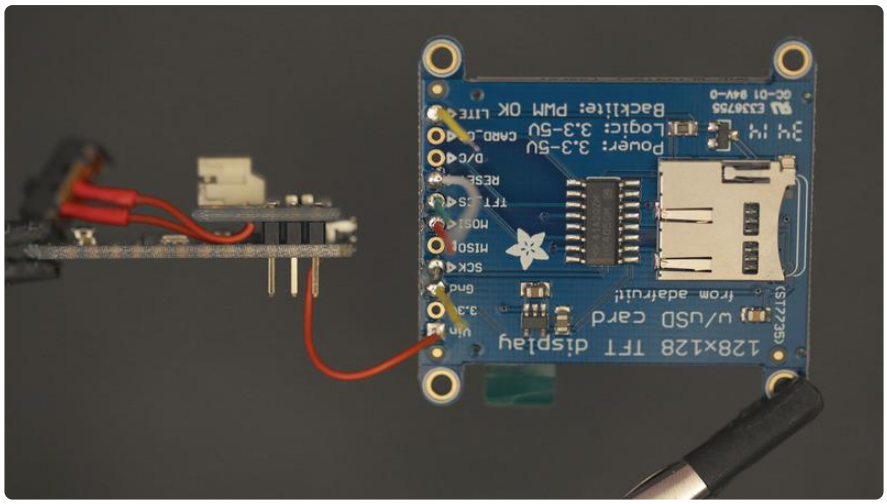
LiPoly backpack headers

Solder the included headers onto the LiPoly backpack.



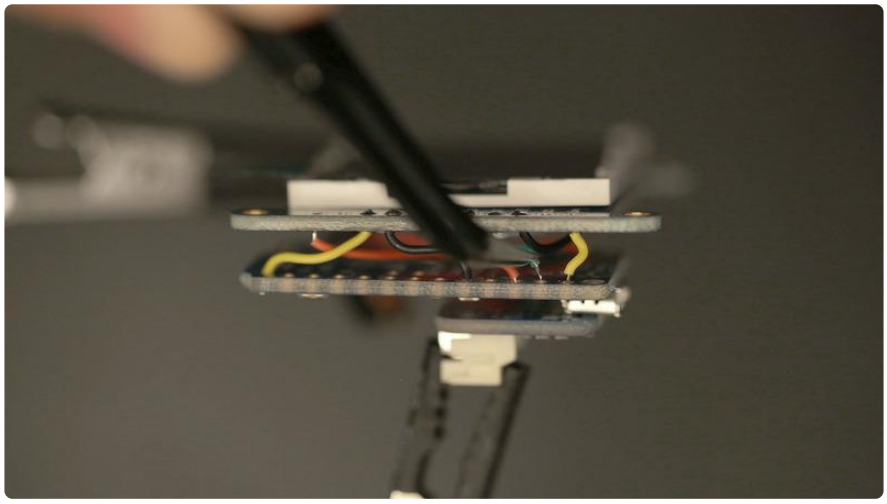
Display power

Connect the VIN pin on the display to Bat pin on the LiPoly backpack and solder the LiPoly header pins into the Pro Trinket.



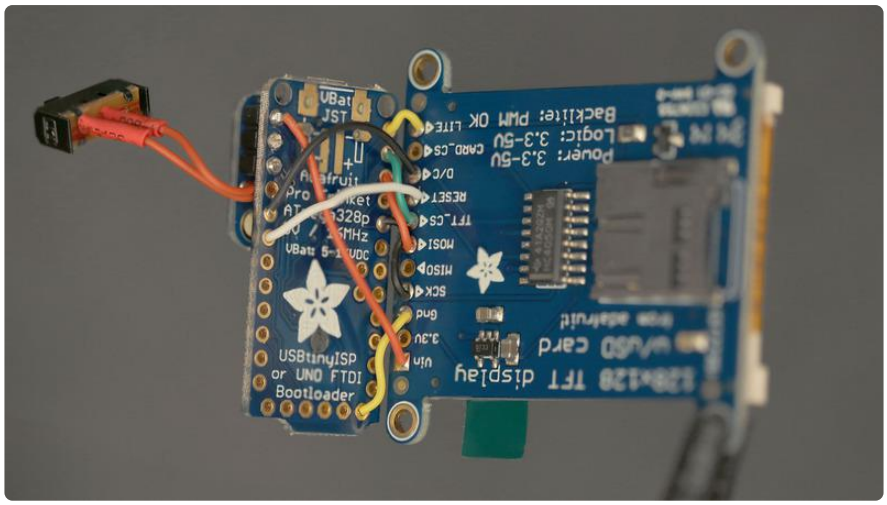
Sandwich Pro Trinket

Use two third-helping hands and tweezers to align each wire and through-hole.



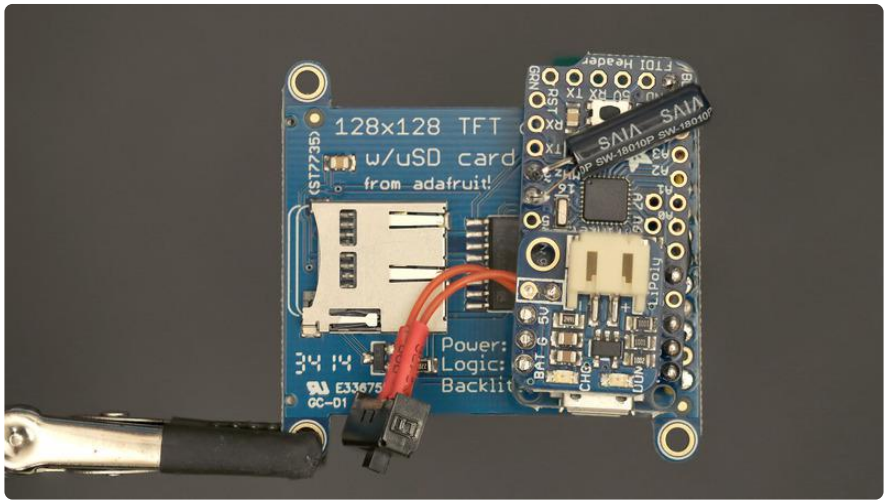
Solder display and Pro Trinket

Follow the circuit diagram and double check each connection before soldering.



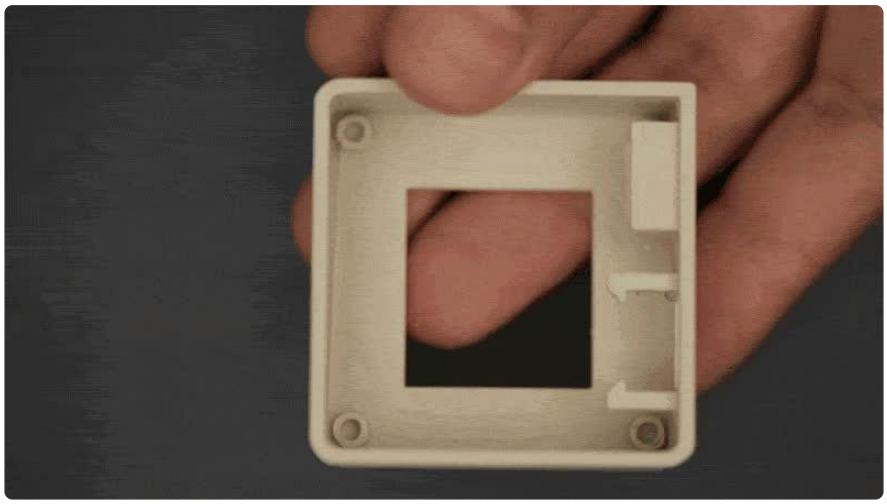
Vibration sensor

Insert the vibration sensor into pins 3 and 4. Use flat pliers to grab the ends and bend it over the Pro Trinket. Make sure to avoid covering the reset button.



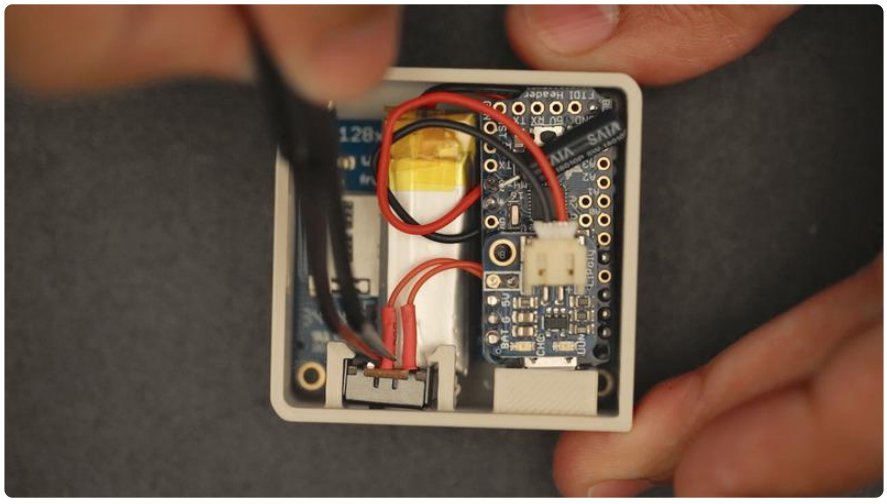
Fit circuit into enclosure

Insert the completed circuit into the printed case at an angle, port side first.



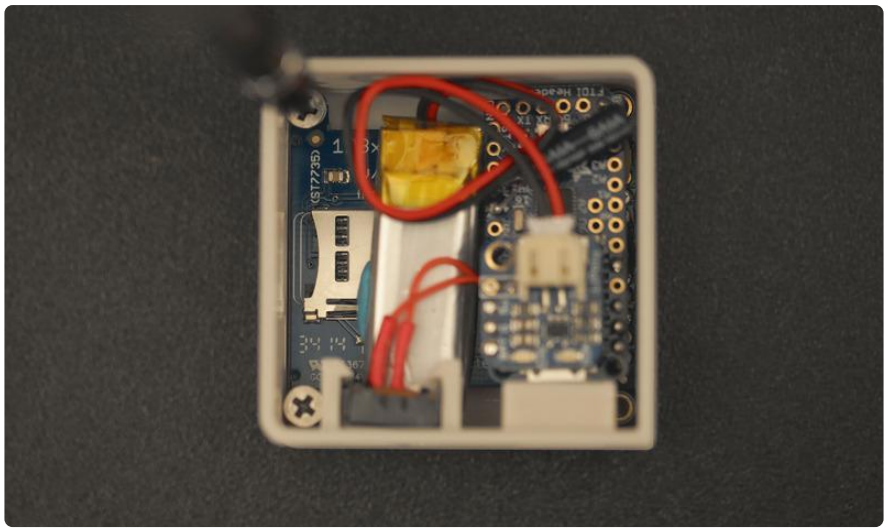
Install Slide Switch

Use tweezers to mount the slide switch into the two clips.



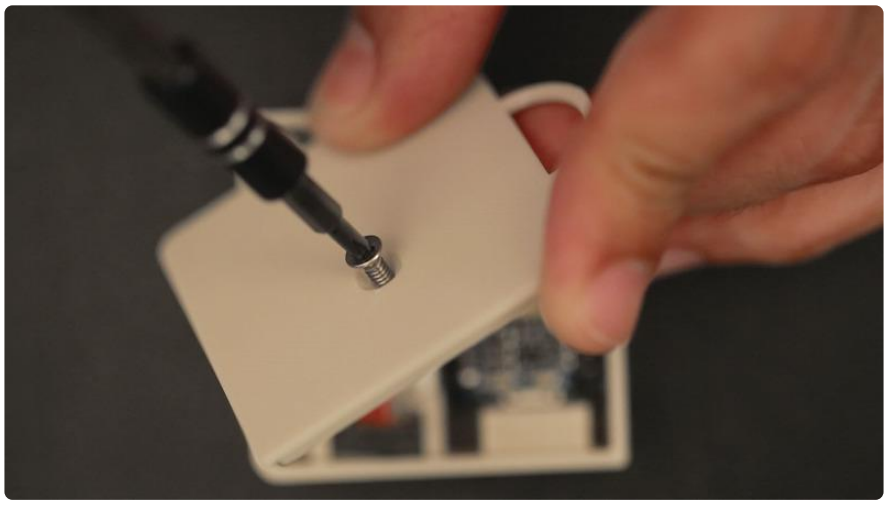
Mount display

Secure the display to the enclosure with two #4-40x 3/8 flat Philips machine screws.



Back Cover

Add one #4-40 x 3/8 flat Philips machine screw into the chamfered hole on the cloud Cover.stl part.



Add buttons

Paint or print each button in a sperate color and glue them on the the front of the clou dCase.stl part.

Make it Wearable

Add a split ring to the top loop of the cover and link it to your favorite necklace.

Wear It

Put it on and show it to your friends! Now you can take your love for 8-bit retroness with you everywhere!