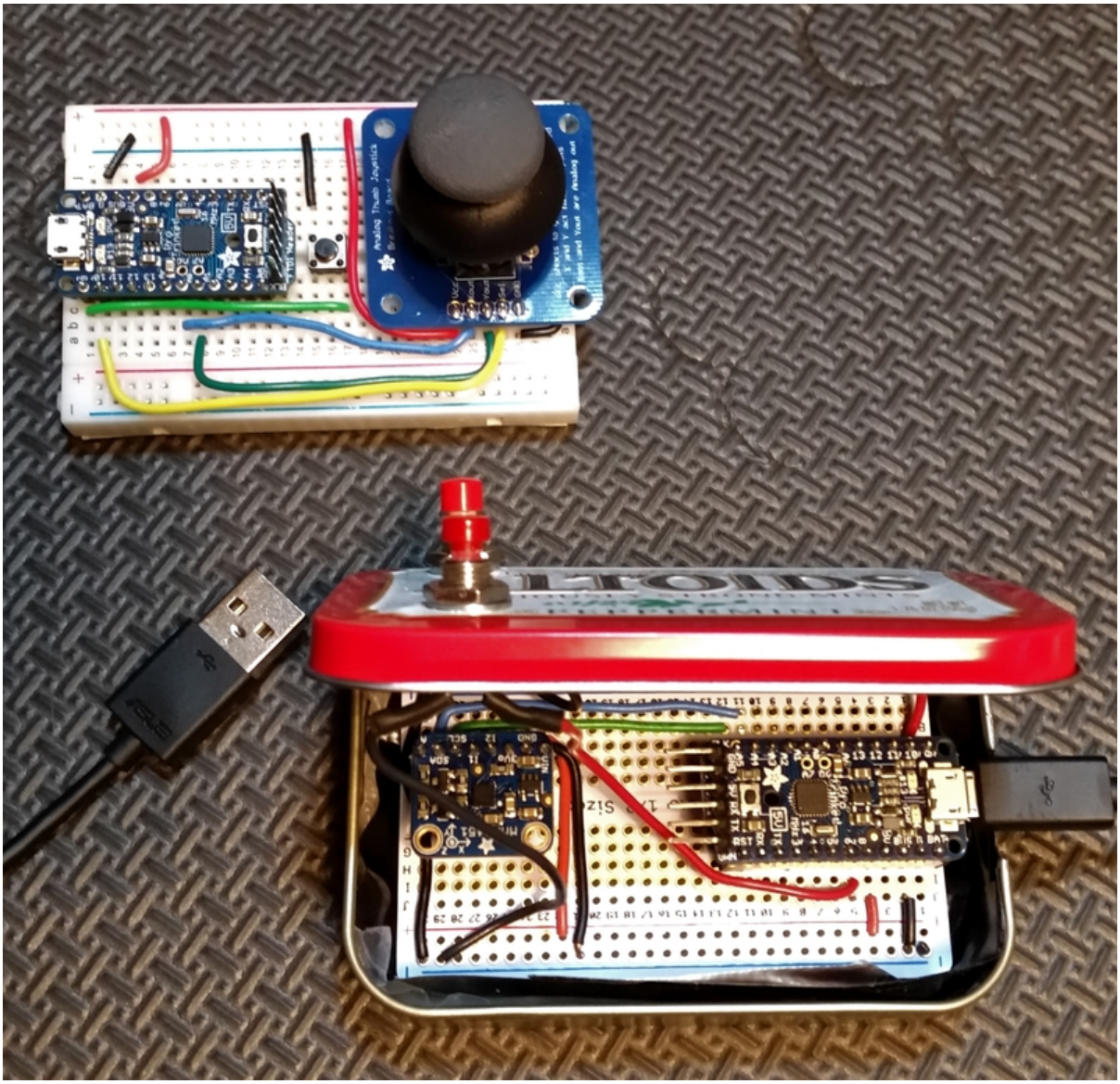
Pro Trinket as a USB HID Mouse

Created by Anne Barela



https://learn.adafruit.com/pro-trinket-usb-hid-mouse

Last updated on 2024-06-03 01:39:31 PM EDT

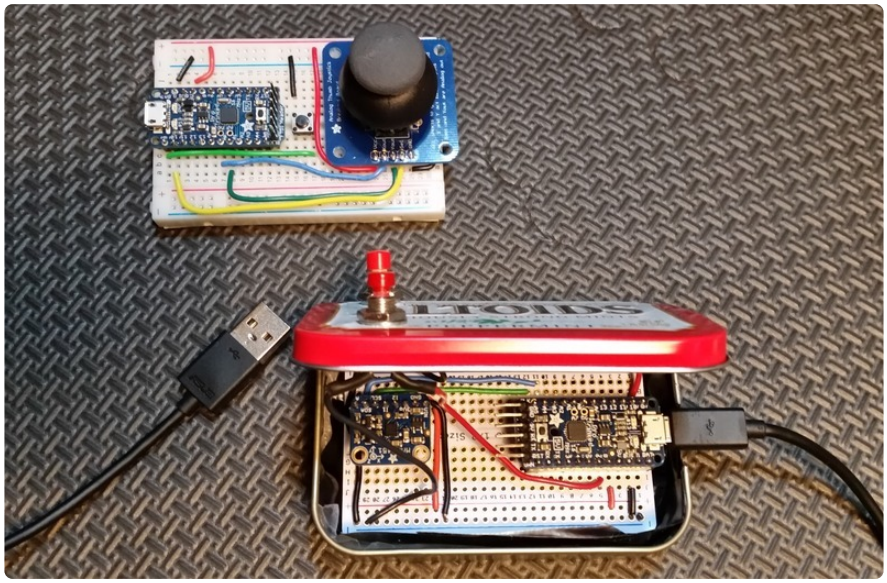
Table of Contents

Overview

This guide has been superseded by other guides which provide similar functionality. Note the methods used in this guide cannot be used with other microcontroller boards.

The Pro Trinket's USB connector is used for uploading sketches, but the connector is not a full USB port due to lack of dedicated hardware. You can, though, use the connection to emulate some basic USB 1.1 devices via an Arduino software library presented in this tutorial. For example, via the USB 1.1 protocol, you can build low speed USB Human Interface Devices (or HID). Examples of HIDs are keyboards, mice, joysticks, gamepads, etc.

Using the Pro Trinket as a USB keyboard is demonstrated in an earlier tutorial here (https://adafru.it/rD3). Pro Trinket can emulate a computer mouse with two examples to demonstrate translatating movement into on-screen cursor changes.



Software Library

The Pro Trinket Mouse library is available on GitHub. (https://adafru.it/enf) You can download the code from the link below. See the tutorial All About Arduino Libraries (https://adafru.it/dNR) on installing the code into your Arduino Integrated Development Environment (IDE) for your use.

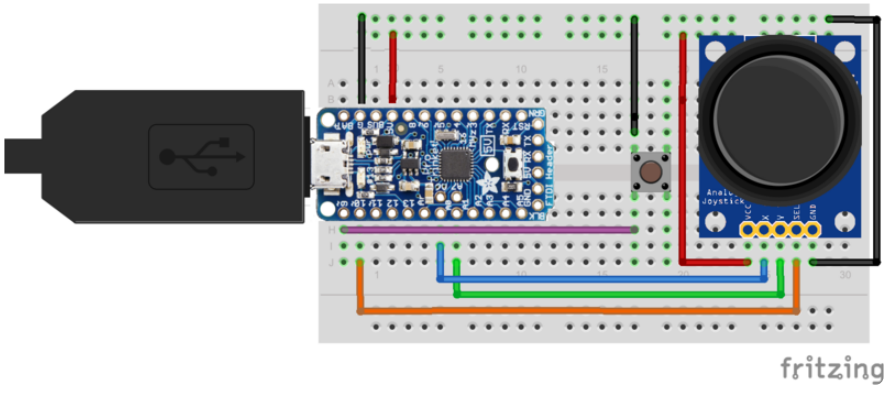
Download the Pro Trinket USB Mouse Library

The functions in the library for interacting with the USB port and sending text are shown below. You can refer to the examples on the following pages to see how they are used.

```
// Starts the USB driver, place in the Arduino IDE setup routine:
void TrinketMouse.begin();

// Here is the do-all function. Pass the following values:
//  x - the number of pixels to move in the horizontal direction (-127 to 127)
//      Note: this is in relation to the current mouse position, not absolute
//      screen location
//  y - the pixels to move in the vertical direction (-127 to 127)
//  wheel - amount to tell the PC the mouse wheel has been spun (-127 to 127)
//  buttonMask - the following values may be ANDed to press mouse buttons:
//      MOUSEBTN_LEFT_MASK      0x01  // Left button
//      MOUSEBTN_RIGHT_MASK     0x02  // Right buttom
//      MOUSEBTN_MIDDLE_MASK    0x04  // Middle Button
void TrinketMouse.move(signed char x, signed char y, signed char wheel, uint8_t
buttonMask); // makes a mouse movement, must be called at least once every 10ms,
even if no movement
// Examples:
TrinketMouse.move(5,-5,0,0);  // Move 5 pixels up, 5 left from current location
TrinketMouse.move(0,0,0,MOUSEBTN_LEFT_MASK); // Click left mouse button

// This function checks if USB is connected, returns 0 if not connected
char TrinketMouse.isConnected();
```

The code works for both the Pro Trinket 5V running at 16 MHz and the Pro Trinket 3V running at 12 MHz.

Example: Joystick Mouse



This circuit uses an Adafruit joystick breakout to read movements to control the mouse. The joystick has a pushbutton switch built-in that simulates the mouse left-click. An external pushbutton (shown on the breadboard) enables and disables the mouse control which is useful when you want the project to not read mouse control (if you have two mice, it allows the other mouse to resume full control). One push enables, another disables.

The mouse uses some exponential math to give non-linear movement - the harder you lean on the joystick, the faster it will go. Also the switch is debounced to prevent spurious mouse clicks.

```
/*
 Joystick Mouse Control

 Controls a PC mouse from a joystick on an Adafruit Pro Trinket.
 Uses the joystick pushbutton to click the left mouse button

 Hardware:
 * 2-axis joystick connected to pins A0 and A1 with pushbutton on D10
 * Pushbutton enable/disable the mouse entirely on D9 (optional)

 The mouse movement is always relative.

 The sketch assumes that the joystick resting values are around the
 middle of the range, but that they vary within a threshold.

 WARNING:  When you use the new mouse, the Arduino takes
 over your mouse!  Make sure you have control before you use the project.
 This sketch includes a pushbutton to toggle the mouse on and off.

Based on software on arduino.cc by Tom Igoe placed in the public domain

Version 1.0  Initial version for Adafruit Pro Trinket by Mike Barela

 */
#include &lt;ProTrinketMouse.h&gt;  // include mouse library for Pro Trinket (3V or
5V)

// set pin numbers for switch, joystick axes, and LED
const int switchPin   =  9;   // switch to turn on and off mouse control
const int mouseButton = 10;   // input pin for the mouse pushButton
const int xAxis  = 1;         // joystick X axis to A1
const int yAxis  = 0;         // joystick Y axis to A0
const int ledPin = 13;        // Mouse control LED

// parameters for reading the joystick
int range = 12;               // output range of X or Y movement (zero to range)
int responseDelay = 5;        // response delay of the mouse, in ms
int threshold = range/4;      // resting threshold
int center = range/2;         // resting position value
const float powerValue = 1.4; // for exponential behavior, 1 &lt; value &lt; 2

boolean mouseIsActive = false;      // whether or not to control the mouse
int lastSwitchState = LOW;          // previous switch state
boolean mouseButtonPressed = false; // whether or not mouse button pressed
int  lastReading = 1;       // last joystick/mouse button reading
long debounceTime = 0;      // last time the mouse button was toggled
long debounce = 50;         // debounce time, increase if the mouse clicks rapidly

void setup() {
  pinMode(switchPin,   INPUT_PULLUP);   // the switch pin
  pinMode(mouseButton, INPUT_PULLUP);   // mouse button on joystick
  pinMode(ledPin, OUTPUT);              // the LED pin
  TrinketMouse.begin();                 // initialize the mouse library
}

void loop() {
  int switchState;  // State of the mouse enable/disable button
  int buttonState;  // State of the mouse left button switch on joystick
  int xReading, yReading; // readings of the joystick movements
  int buttonReading;      // reading of the joystick (left mouse) button

  switchState = digitalRead(switchPin);  // read the mouse disable switch
```

```
  // if it's changed and it's high, toggle the mouse state
  if (switchState != lastSwitchState) {
    if (switchState == HIGH) {
      mouseIsActive = !mouseIsActive;
//      digitalWrite(ledPin, mouseIsActive);   // toggle LED to indicate mouse state
    }
  }
  lastSwitchState = switchState;    // save switch state for next comparison

  // read and scale the two joystick readings, one for each axis
  xReading = readAxis(xAxis);
  yReading = readAxis(yAxis);

  // This code gives the mouse a nonlinear acceleration
  //   These 8 lines may be commented out to have linear acceleration
  if(xReading &gt; 0)
     xReading =  (int)pow(powerValue,xReading);
  else if(xReading &lt; 0)
     xReading = -(int)pow(powerValue,-xReading);

  if(yReading &gt; 0)
     yReading =  (int)pow(powerValue,yReading);
  else if(yReading &lt; 0)
     yReading = -(int)pow(powerValue,-yReading);  // end nonlinear acceleration code

  // Read the joystick button as the left mouse button.  Debounce per
  //   Ladyada code at https://learn.adafruit.com/tilt-sensor/using-a-tilt-sensor
  buttonReading = digitalRead(mouseButton);  // read the mouse left button (push
joystick)
  if(buttonReading != lastReading) {         // switch changed
     debounceTime = millis();                // reset debounce timer
  }
  if((millis() - debounceTime) &gt; debounce) {
     buttonState = buttonReading;
     if(buttonState == LOW) {
        mouseButtonPressed = true;
     }
     else {
        mouseButtonPressed = false;
     }
  }
  lastReading = buttonReading;
  digitalWrite(ledPin, mouseButtonPressed);  // toggle LED to indicate button state

  // if the mouse control state is active, move the mouse:
  if (mouseIsActive) {
      if (mouseButtonPressed) {  // if joystick pressed down, indicate that too
         TrinketMouse.move(xReading, yReading, 0, MOUSEBTN_LEFT_MASK);
      }
      else {
         TrinketMouse.move(xReading, yReading, 0, 0);  // move, no mouse button
press
      }
  }
  delay(responseDelay);  // wait between mouse readings
}

// Reads a joystick axis (0 or 1 for x or y) and scales the
//  analog input range to a range from 0 to &lt;range&gt;
int readAxis(int thisAxis) {
  int reading = analogRead(thisAxis);  // read the analog input

  // map the reading from the analog input range to the output range
  reading = map(reading, 0, 1023, 0, range);

  // if the output reading is outside from the rest position threshold, use it
  int distance = center - reading;

  if (abs(distance) &lt; threshold) { // if distance not to threshold, no move
```

```
    distance = 0;                  // prevents tiny jitters due to readings
  }
  return distance;   // return the distance for this axis
}
```

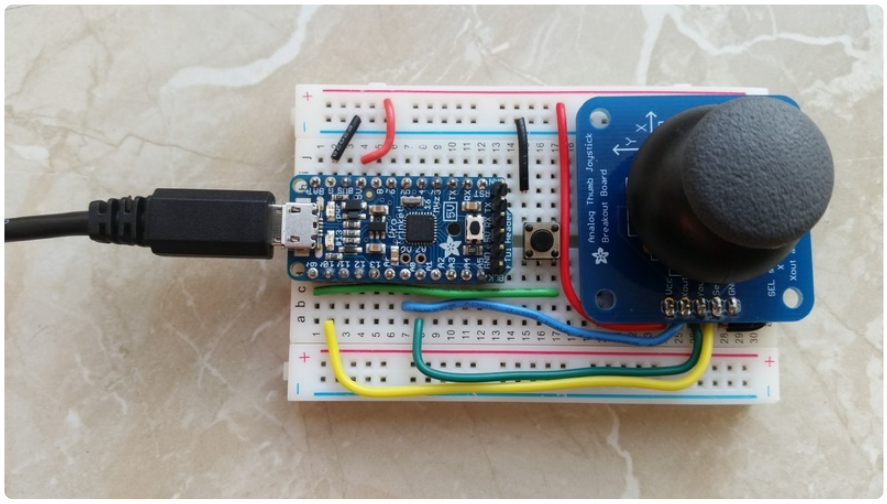
Use

The single button enables and disables the mouse. The joystick moves the mouse cursor in all 4 directions with acceleration for bigger movements. The joystick push down button is the mouse left button. The Pro Trinket onboard red LED indicates the left button press but can be changed to the on/off state by changing the two lines with a digitalWrite to ledPin.

Wiring

Solder the joystick to its breakout board and small piece of male header shown in the product page (http://adafru.it/512). Solder headers (included) onto the Pro Trinket. I stripped and cut my own wires from the nice Adafruit hookup wire selection (http://adafru.it/1311) but you can use premade wires (http://adafru.it/153) for prototyping.

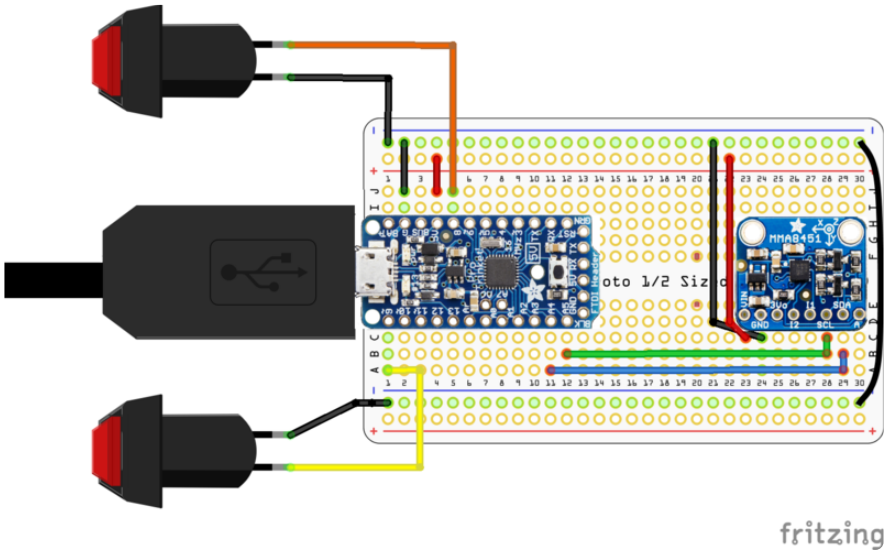
The completed breadboard is shown below. There is a ground wire from the joystick ground to the top ground line that is obscured by the joystick circuit board.



If you would like to make a more permanent project, you can transfer the parts to an Adafruit half Perma-Proto board (http://adafru.it/1609). See the next project, the accelerometer mouse for how using a Perma-Proto looks.

Example: Accelerometer Mouse

The circuit below uses an Adafruit accelerometer breakout to read movements to control the mouse. If you tilt the mouse, it moves the cursor. The momentary buttons serve as the right and left mouse buttons.



The accelerometer (http://adafru.it/2019) samples movement. Keeping the project flat with respect to the ground, tilting the mouse forward and back moves the cursor in the x direction (horizontal) while turning your wrist around the axis of the USB cable moves it in the y direction (vertical). Two external pushbuttons are used for right and left click. This project uses the half Perma-Proto (http://adafru.it/571) which fits in an altoids-sized tin (http://adafru.it/97). Adafruit carries a nice mint tin-sized board (http://adafru.it/723) that would work well also.

Build

With the Perma-Proto board, you can use male header to solder into the board then solder onto the Pro Trinket. I prefer, when possible, to cut some female header into two pieces of 12 to solder onto the board, making a socket for the Pro Trinket. This will make the Pro Trinket sit a bit close to the top of an Altoid tin if you use it as a case. An 8 pin piece of female header may also be soldered onto the circuit board to plug in the accelerometer. Once the headers are on, you can use hookup wire to make the power and signal connections shown. The I2C wires of the sensor are connected to A4 and A5 on the Pro Trinket which are the SDA and SCL lines.

Be sure to insulate the bottom (and maybe the top) of the mint tin so electrical connections are not shorted by the metal case.

The code for using the accelerometer with the Pro Trinket and two momentary buttons as a mouse is below. Be sure you have loaded the Pro Trinket Mouse library from the first page of the tutorial.

```
/**************************************************************************/
/*!
    @file     ProTrinketMouseAccel.ino
    @author   Mike Barela for Adafruit Industries
    @license  BSD (see license.txt)

    This is an example of using the Adafruit Pro Trinket with the Adafruit MMA8451
    Accelerometer breakout board to make a mouse using motion gestures

    Adafruit invests time and resources providing this open source code,
    please support Adafruit and open-source hardware by purchasing
    products from Adafruit!

    @section  HISTORY

    v1.0  - First release 1/25/2015  Mike Barela
*/
/**************************************************************************/

#include &lt;Wire.h&gt;               // I2C Library
#include &lt;Adafruit_MMA8451.h&gt;   // Accelerometer library
#include &lt;Adafruit_Sensor.h&gt;    // Adafruit sensor library
#include &lt;ProTrinketMouse.h&gt;    // Pro Trinket V-USB mouse emulator
#define  DEBUG  0              // Set to 1 for serial console debugging, 0
otherwise

const uint8_t LEFTBUTTON  = 8;  // Left Mouse Button on this Pro Trinket Pin
const uint8_t RIGHTBUTTON = 9;  // Right Mouse Button on Pro Trinket

Adafruit_MMA8451 mma = Adafruit_MMA8451();  // Create an accelerometer object

//Change these values if accelerometer reading are different:
//How far the accerometer is tilted before
//starting to move the mouse:
const int MovementThreshold = 18;

//The average zero acceleration values read
//from the accelerometer for each axis:
const int ZeroXValue = 0;
const int ZeroYValue = 0;
//const int ZeroZValue = 0;

//The maximum (positive) acceleration values read
//from the accelerometer for each axis:
const int MaxXValue = 4096;
const int MaxYValue = 4096;
//const int MaxZValue = 4096;

//The minimum (negative) acceleration values read
//from the accelerometer for each axis:
const int MinXValue = -4096;
const int MinYValue = -4096;
//const int MinZValue = -4096;

//The sign of the mouse movement relative to the acceleration.
//If your cursor is going in the opposite direction you think it
//should go, change the sign for the appropriate axis.
const int XSign = 1;
const int YSign = 1;
//const int ZSign = 1;

//The maximum speed in each axis (x and y)
```

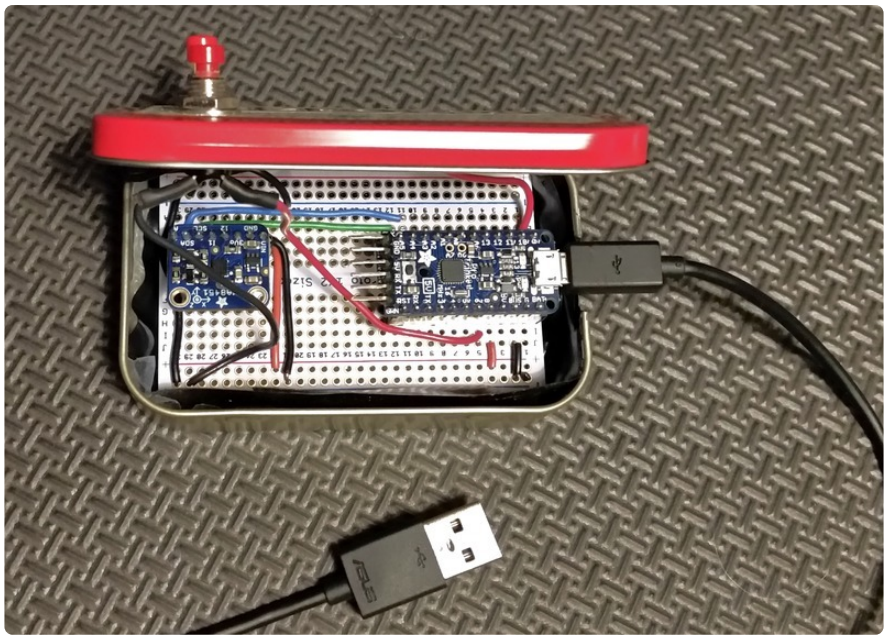
```
//that the cursor should move. Set this to a higher or lower
//number if the cursor does not move fast enough or is too fast.
const int MaxMouseMovement = 50;

//This reduces the 'twitchiness' of the cursor by calling
//a delay function at the end of the main loop.
//There are better way to do this without delaying the whole
//microcontroller, but that is left for another tutorial or project.
const int MouseDelay = 12;

void setup(void) {
#if DEBUG
  Serial.begin(9600);
  Serial.println("Pro Trinket Accelerometer Mouse");
#endif
  if (! mma.begin()) {  // If the accelerometer cannot be found, flash LED
    pinMode(13, OUTPUT);
    while (1) {         // Flash the Pin 13 LED quickly to indicate an error
      digitalWrite(13, HIGH);
      delay(350);
      digitalWrite(13, LOW);
      delay(350);
    }
  }
  mma.setRange(MMA8451_RANGE_2_G);  // 2G Mode is best for hand gestures
  mma.read();                       // get an initial read

  TrinketMouse.begin();               // Initialize mouse library
  pinMode(LEFTBUTTON,  INPUT_PULLUP); // Left and right mouse button pins
initialized
  pinMode(RIGHTBUTTON, INPUT_PULLUP); //   with internal pullup resistors (bring
Low with button)
}

void loop() {

  mma.read();  //   // Read the 'raw' data in 14-bit counts
#if DEBUG
  Serial.print("X:\t"); Serial.print(mma.x);
  Serial.print("\tY:\t"); Serial.print(mma.y);
  Serial.print("\tZ:\t"); Serial.println(mma.z);
#endif

  processAccelerometer(mma.x,mma.y, mma.z);  // Work with the read data

  delay(MouseDelay);  // wait until next reading - was 500 in Adafruit example
}

//Function to process the acclerometer data
//and send mouse movement information via USB
void processAccelerometer(int16_t XReading, int16_t YReading, int16_t ZReading)
{
  //Initialize values for the mouse cursor movement.
  int16_t MouseXMovement = 0;
  int16_t MouseYMovement = 0;

  //Calculate mouse movement
  //If the analog X reading is ouside of the zero threshold...
  if( MovementThreshold &lt; abs( XReading - ZeroXValue ) )
  {
    //...calculate X mouse movement based on how far the X acceleration is from its
zero value.
    MouseXMovement = XSign * ( ( ( (float)( 2 * MaxMouseMovement ) / ( MaxXValue -
MinXValue ) ) * ( XReading - MinXValue ) ) - MaxMouseMovement );
    //it could use some improvement, like making it trigonometric.
  }
  else
  {
    //Within the zero threshold, the cursor does not move in the X.
```

```
    MouseXMovement = 0;
  }

  //If the analog Y reading is ouside of the zero threshold...
  if( MovementThreshold &lt; abs( YReading - ZeroYValue ) )
  {
    //...calculate Y mouse movement based on how far the Y acceleration is from its
zero value.
    MouseYMovement = YSign * ( ( ( (float)( 2 * MaxMouseMovement ) / ( MaxYValue -
MinYValue ) ) * ( YReading - MinYValue ) ) - MaxMouseMovement );
    //it could use some improvement, like making it trigonometric.
  }
  else
  {
    //Within the zero threshold, the cursor does not move in the Y.
    MouseYMovement = 0;
  }

  if(digitalRead(LEFTBUTTON) == LOW) {             // If left button pressed
#if DEBUG
    Serial.println("Left Mouse Button");
#endif
    TrinketMouse.move(0,0,0,MOUSEBTN_LEFT_MASK);   //  tell PC
  }
  else if (digitalRead(RIGHTBUTTON) == LOW) {      // If right button pressed
#if DEBUG
    Serial.println("Right Mouse Button");
#endif
    TrinketMouse.move(0,0,0,MOUSEBTN_RIGHT_MASK);  //  tell PC
  }
  else {
    TrinketMouse.move(MouseXMovement, MouseYMovement, 0, 0);  // otherwise just
move mouse
  }

}
```

Why not use direct accelerometer movements?

You might be tempted to just read the X and Y accelerometer values and use this to map to screen movements rather than tilt. I tried, others also. The thing is that 1) you have to use jerky movements to get accelerations, and 2) "down" will always produce a gravity acceleration of 9.8 meters per second squared. If "down" changes from the Z axis, you'll get measurements in X and Y which make the mouse skew off to the screen edge.

Can the effects of gravity be subtracted out? Not just with an accelerometer. If you get a 9 or 10 degree of freedom (DOF) sensor package, like the ones used for quadcopters and other movable items, there is a math-intensive way to get the movement translated without gravity. See the Adafruit tutorial AHRS for Adafruit's 9-DOF, 10-DOF, LSM9DS0 Breakouts (https://adafru.it/CeO) for details.




The Original Celebrated
ALTOIDS®
CURIOUSLY STRONG MINTS
PEPPERMINT
NET WT 1.76 OZ (50g)