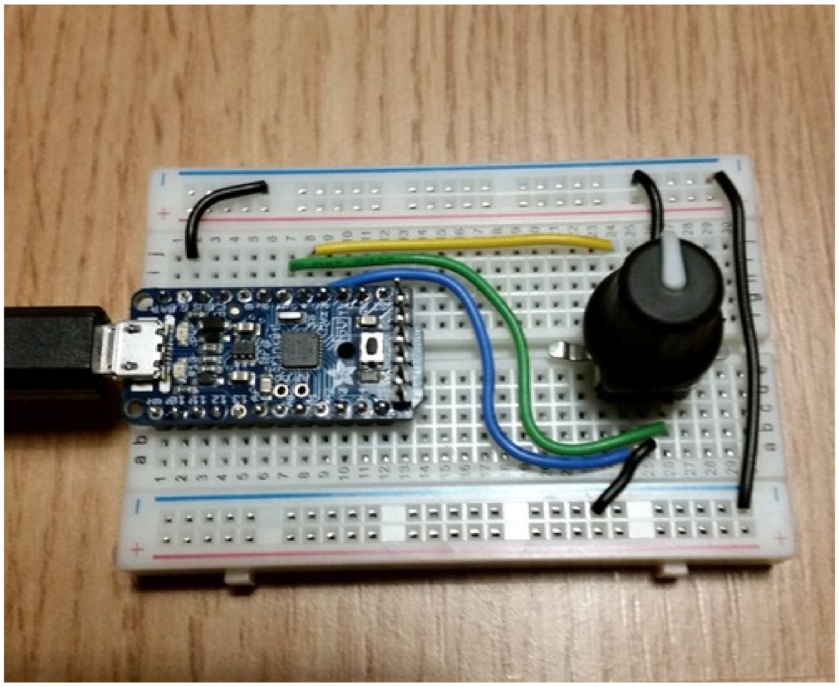
adafruit learning system

Pro Trinket Rotary Encoder

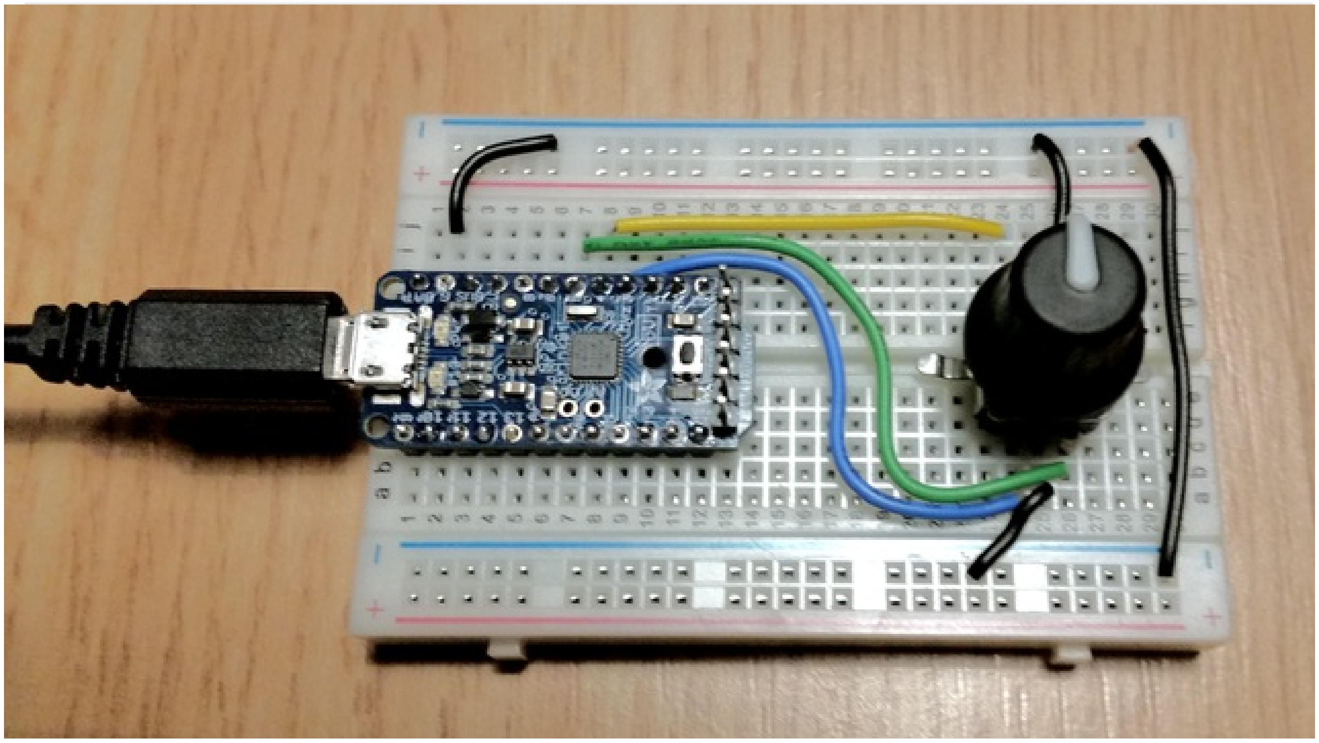
Created by Mike Barela



Last updated on 2015-01-28 03:30:11 PM EST

Guide Contents

Overview



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.

Previous Pro Trinket HID tutorials:

- Pro Trinket as a USB Mouse (http://adafru.it/eo5)
- Pro Trinket as a USB Keyboard (http://adafru.it/emo)

Pro Trinket can emulate a both a computer mouse and keyboard at the same time. The code is a bit larger but adds the functionality of the previous two libraries.

This tutorial will show use of the library using a rotary encoder switch as a digital volume knob and mute switch for multimedia PCs.

Software Library

The Pro Trinket HID library (http://adafru.it/eo6) is available on GitHub. (http://adafru.it/enf) You can download the code from the link below. See the tutorial All About Arduino Libraries (http://adafru.it/egP) on installing the code into your Arduino Integrated Development Environment (IDE) for your use.

Download the Pro Trinket USB Mouse and Keyboard HID Library

http://adafru.it/eo7

The functions available in the library are shown below. All of the definitions are in the ProTrinketHidCombo.h file in the library. There are definitions for the common PC keyboard keys including control and shift variants. Special to this library are multimedia and special keys.

The mouse part of the library differs from the Pro Trinket HID mouse library in the moveMouse function provides movement and mouse clicks but does not emulate the scroll wheel. The arguments are the x and y values to move the mouse and whether the buttons are to be pressed - the predefined values for mouse presses are also in the .h file.

```
// Starts the USB driver, place in the Arduino IDE setup routine:
void TrinketHidCombo.begin(); // start the USB device engine and enumerate

TrinketHidCombo.isConnected(); // checks if USB is connected, 0 if not connected

// this (or "press" something) must be called at least once every 10ms
TrinketHidCombo.poll();  // this (or "press" something) must be called at least once every 10ms

// presses up to 6 keys, and modifiers (modifiers are keys like shift, CTRL, etc)
TrinketHidCombo.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3, 
TrinketHidCombo.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3, 
TrinketHidCombo.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3, 
TrinketHidCombo.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3);
TrinketHidCombo.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2);
TrinketHidCombo.pressKey(uint8_t modifiers, uint8_t keycode1);

// presses a list of keys
TrinketHidCombo.pressKeys(uint8_t modifiers, uint8_t* keycodes, uint8_t sz);

// type out a single ASCII character
TrinketHidCombo.typeChar(uint8_t ascii);

// Press special keyboard key types
TrinketHidCombo.pressMultimediaKey(uint8_t key);
TrinketHidCombo.pressSystemCtrlKey(uint8_t key);

// returns the state of the three LEDs on a keyboard (caps/num/scroll lock)
TrinketHidCombo.getLEDstate();

// inherit from "Print", these two write functions are implemented
```

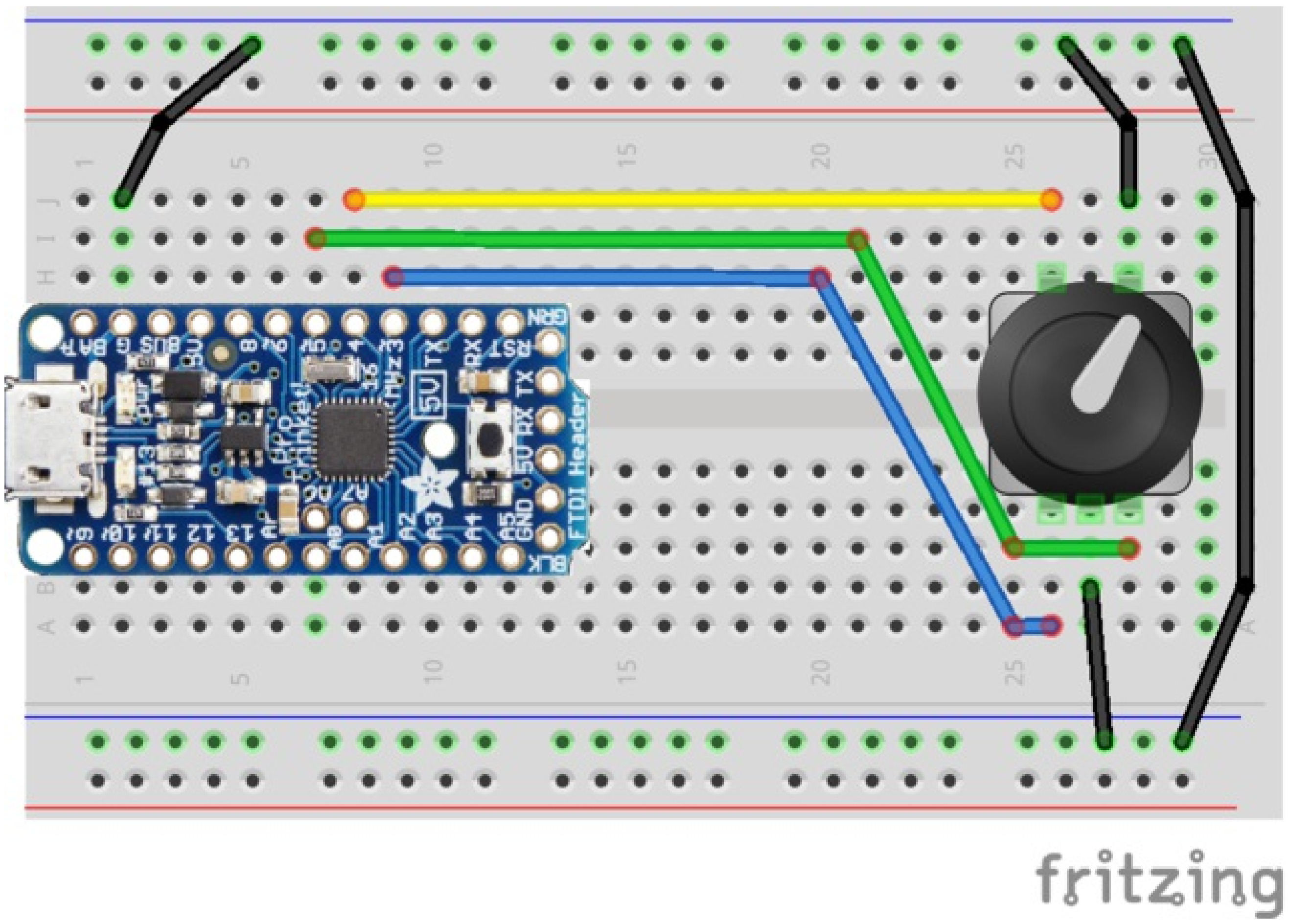
```
size_t val = TrinketHidCombo.write(uint8_t);
using Print::write;
// other "print" and "println" functions are automatically available
using Print::print;
using Print::println;

// helps translate ASCII characters into keycode and modifier combinations,
// while taking into account whether or not caps lock is on
ASCII_to_keycode(uint8_t ascii, uint8_t ledState, uint8_t* modifier, uint8_t* keycode);

// Here is the mouse function. This is a bit different from the Pro Trinket mouse library
//     as this does not have the scroll wheel argument.
//  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)
//  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 TrinketHidCombo.mouseMove(signed char x, signed char y, uint8_t buttonMask); // makes a mou
// Examples:
TrinketHidCombo.mouseMove(5,-5,0);  // Move 5 pixels up, 5 left from current location
TrinketHidCombo.move(0,0,MOUSEBTN_LEFT_MASK); // Click left mouse button

// See the file ProTrinketHidCombo.h in the library for all the possible
//    predefined key codes.  Some special codes available:
//
// multimedia keys
// MMKEY_VOL_UP       0xE9
// MMKEY_VOL_DOWN   0xEA
// MMKEY_SCAN_NEXT_TRACK 0xB5
// MMKEY_SCAN_PREV_TRACK 0xB6
// MMKEY_STOP    0xB7
// MMKEY_PLAYPAUSE   0xCD
// MMKEY_MUTE    0xE2
// MMKEY_BASSBOOST   0xE5
// MMKEY_LOUDNESS   0xE7
// MMKEY_KB_EXECUTE      0x74
// MMKEY_KB_HELP   0x75
// MMKEY_KB_MENU   0x76
// MMKEY_KB_SELECT   0x77
// MMKEY_KB_STOP   0x78
// MMKEY_KB_AGAIN   0x79
// MMKEY_KB_UNDO   0x7A
// MMKEY_KB_CUT       0x7B
// MMKEY_KB_COPY   0x7C
// MMKEY_KB_PASTE   0x7D
// MMKEY_KB_FIND   0x7E
//
// system control keys
// SYSCTRLKEY_POWER      0x01
// SYSCTRLKEY_SLEEP      0x02
// SYSCTRLKEY_WAKE   0x03
```

Example: Rotary Encoder Volume Control

A much-requested interface for computers is alternative access to keyboard special function keys.

This project uses the HID library with a rotary encoder switch as a digital volume control for PC sound. The push button connected to the encoder shaft is used as a mute/unmute button.

The wiring is shown below. One side of the switch has two pins, that is the push switch. The side with three contacts is the rotary part of the switch. Circuit power comes from the USB connection.



The code for the volume project is below. Ensure you have installed the HID library. The trickiest part is the poll function must be called every 10 milliseconds so all the other processing needs to be done quickly.

The code for reading the encoder is a bit involved. The switch transitions need to be determined so the direction the switch is being turned can be found. To read things quickly, the sketch uses microcontroller port polling which is much quicker than digitalRead function calls. Direct port access techniques are discussed on arduino.cc.

```
/*************************************************************************/
/*!
    @file     ProTrinketVolumeKnobPlus.ino
    @author   Mike Barela for Adafruit Industries
    @license  MIT

    This is an example of using the Adafruit Pro Trinket with a rotary
    encoder as a USB HID Device.  Turning the knob controls the sound on
    a multimedia computer, pressing the knob mutes/unmutes the sound

    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 1/26/2015  Mike Barela based on code by Frank Zhou
*/
/*************************************************************************/

#include <ProTrinketHidCombo.h>

#define PIN_ENCODER_A      3
#define PIN_ENCODER_B      5
#define TRINKET_PINx       PIND
#define PIN_ENCODER_SWITCH 4

static uint8_t enc_prev_pos   = 0;
static uint8_t enc_flags      = 0;
static char    sw_was_pressed = 0;

void setup()
{
  // set pins as input with internal pull-up resistors enabled
  pinMode(PIN_ENCODER_A, INPUT_PULLUP);
  pinMode(PIN_ENCODER_B, INPUT_PULLUP);
  pinMode(PIN_ENCODER_SWITCH, INPUT_PULLUP);

  TrinketHidCombo.begin(); // start the USB device engine and enumerate

  // get an initial reading on the encoder pins
  if (digitalRead(PIN_ENCODER_A) == LOW) {
    enc_prev_pos |= (1 << 0);
  }
  if (digitalRead(PIN_ENCODER_B) == LOW) {
    enc_prev_pos |= (1 << 1);
  }
}

void loop()
{
  int8_t enc_action = 0; // 1 or -1 if moved, sign is direction

  // note: for better performance, the code will use
  // direct port access techniques
  // http://www.arduino.cc/en/Reference/PortManipulation
  uint8_t enc_cur_pos = 0;
  // read in the encoder state first
  if (bit_is_clear(TRINKET_PINx, PIN_ENCODER_A)) {
    enc_cur_pos |= (1 << 0);
  }
  if (bit_is_clear(TRINKET_PINx, PIN_ENCODER_B)) {
    enc_cur_pos |= (1 << 1);
  }

  // if any rotation at all
  if (enc_cur_pos != enc_prev_pos)
  {
    if (enc_prev_pos == 0x00)
    {
      // this is the first edge
      if (enc_cur_pos == 0x01) {
        enc_flags |= (1 << 0);
      }
      else if (enc_cur_pos == 0x02) {
        enc_flags |= (1 << 1);
      }
    }

    if (enc_cur_pos == 0x03)
    {
      // this is when the encoder is in the middle of a "step"
      enc_flags |= (1 << 4);
    }
    else if (enc_cur_pos == 0x00)
    {
      // this is the final edge
      if (enc_prev_pos == 0x02) {
        enc_flags |= (1 << 2);
      }
      else if (enc_prev_pos == 0x01) {
        enc_flags |= (1 << 3);
      }

      // check the first and last edge
      // or maybe one edge is missing, if missing then require the middle state
      // this will reject bounces and false movements
      if (bit_is_set(enc_flags, 0) && (bit_is_set(enc_flags, 2) || bit_is_set(enc_flags, 4))) {
        enc_action = 1;
      }
      else if (bit_is_set(enc_flags, 2) && (bit_is_set(enc_flags, 0) || bit_is_set(enc_flags, 4))) {
        enc_action = 1;
      }
      else if (bit_is_set(enc_flags, 1) && (bit_is_set(enc_flags, 3) || bit_is_set(enc_flags, 4))) {
        enc_action = -1;
      }
      else if (bit_is_set(enc_flags, 3) && (bit_is_set(enc_flags, 1) || bit_is_set(enc_flags, 4))) {
        enc_action = -1;
      }

      enc_flags = 0; // reset for next time
    }
  }

  enc_prev_pos = enc_cur_pos;

  if (enc_action > 0) {
    TrinketHidCombo.pressMultimediaKey(MMKEY_VOL_UP);  // Clockwise, send multimedia volume up
  }
  else if (enc_action < 0) {
    TrinketHidCombo.pressMultimediaKey(MMKEY_VOL_DOWN); // Counterclockwise, is multimedia volu
  }

  // remember that the switch is active low
  if (bit_is_clear(TRINKET_PINx, PIN_ENCODER_SWITCH))
  {
    if (sw_was_pressed == 0) // only on initial press, so the keystroke is not repeated while the button i
    {
      TrinketHidCombo.pressMultimediaKey(MMKEY_MUTE); // Encoder pushed down, toggle mute or no
      delay(5); // debounce delay
    }
    sw_was_pressed = 1;
  }
  else
  {
    if (sw_was_pressed != 0) {
      delay(5); // debounce delay
    }
    sw_was_pressed = 0;
  }

  TrinketHidCombo.poll(); // check if USB needs anything done, do every 10 ms or so
}
```

Use

Plug the USB cable into the Pro Trinket. You should hear a sound (if using Windows) and note that Windows Update loads the USB device driver.

Start playing music on the computer. As you turn the knob, the volume should go up one way, down the other. Push down on the switch and the sound should be muted, another push, unmute.