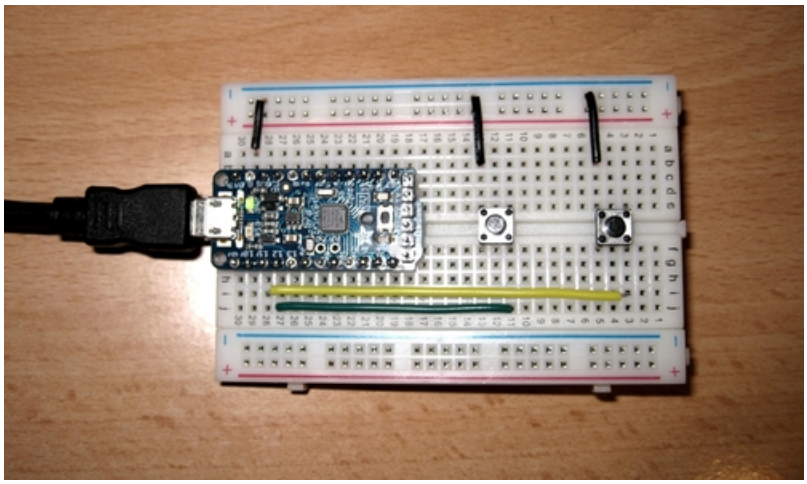
adafruit learning system

Pro Trinket Keyboard

Created by Mike Barela



Last updated on 2018-08-22 03:45:52 PM UTC

Guide Contents

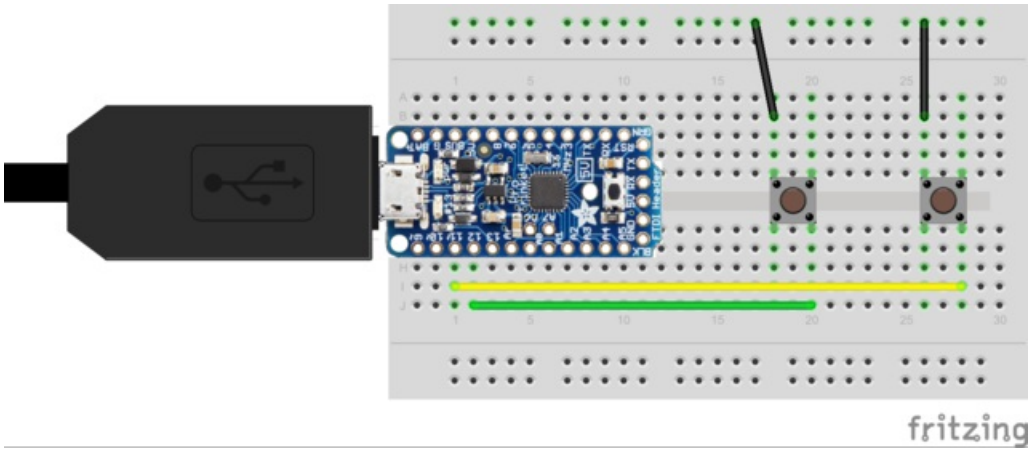
Overview

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.

The easiest and most useful device is a HID keyboard. Pro Trinket can emulate a computer keyboard, one which may be customized by users.

Circuit Wiring

The simplest example is a Pro Trinket programmed to send characters when a trigger event happens. For triggering your own key sequences, you may add button(s) as shown in the Fritzing diagram below:



With version 1.1 of the library updated January 5, 2015, the code works with Pro Trinket 5V 16 MHz and Pro Trinket 3V 12 MHz. Just be sure to select the correct board in the Arduino IDE Tools -> Board list. If you do not see Pro Trinket as a board, add the boards as listed in the Pro Trinket tutorial (https://adafru.it/e8N).

Library

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.

https://adafru.it/ejY

https://adafru.it/ejY

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 next page to see how they are used.

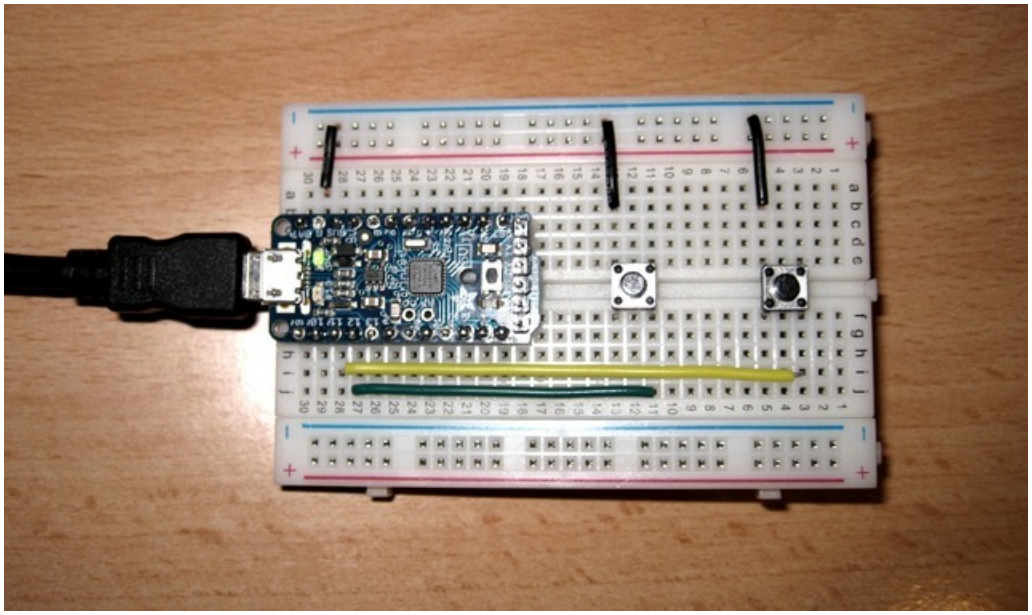
```
Trinket_Keyboard.begin(); // starts the USB driver, causes re-enumeration

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

// this (or "press" something) must be called at least once every 10ms
Trinket_Keyboard.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)
Trinket_Keyboard.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3, uint8_
Trinket_Keyboard.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3, uint8_
Trinket_Keyboard.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3, uint8_
Trinket_Keyboard.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2, uint8_t keycode3);
Trinket_Keyboard.pressKey(uint8_t modifiers, uint8_t keycode1, uint8_t keycode2);
Trinket_Keyboard.pressKey(uint8_t modifiers, uint8_t keycode1);

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

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

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

// inherit from "Print", these two write functions are implemented
size_t val = Trinket_Keyboard.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);
```

A list of keys and keyboard modifiers is in the file ProTrinketKeyboard.h.

Design your code so you call a keypress or the poll function once every 10 milliseconds. This allows the connection to the USB host (PC) to "stay alive". Placing the call in the Arduino Loop function is usually sufficient but if you have any code in the body of the loop that waits, include poll calls in those areas also.

Examples

The sketches below are in the examples folder in the library. The first is one for the Fritzing sketch using two pushbuttons for your own keyboard.

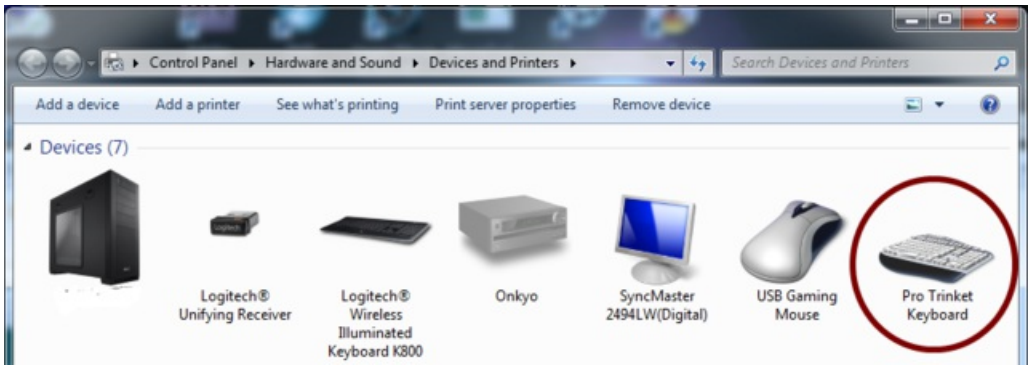


Be sure to select Pro Trinket 5V or Pro Trinket 3V in the Arduino IDE Tools -> Board menu. If you select the original Trinket or another board you will get cryptic compiler errors including "constant not defined".

```
/*
ProTrinketKeyboard example
For Pro Trinket (ATmega328 based Trinket) by Adafruit Industries
Please use library TrinketKeyboard for the ATtiny85 based Trinket
Version 1.0  2015-01-01 Initial Version derived from TrinketKeyBoardExample  Mike Barela
*/

#include <ProTrinketKeyboard.h>  // Ensure the library is installed

// Switches are connected from ground to these defined pins
const int PIN_BUTTON_CAPITAL_A = 12;
const int PIN_BUTTON_STRING    = 11;

void setup()
{
  // Declare button pins as inputs
  pinMode(PIN_BUTTON_CAPITAL_A, INPUT);
  pinMode(PIN_BUTTON_STRING,    INPUT);

  // setting input pins to high means turning on internal pull-up resistors
  digitalWrite(PIN_BUTTON_CAPITAL_A, HIGH);
  digitalWrite(PIN_BUTTON_STRING,    HIGH);
  // remember, the buttons are active-low, they read LOW when they are not pressed

  // start USB stuff
  TrinketKeyboard.begin();
}

void loop()
{
  TrinketKeyboard.poll();
  // the poll function must be called at least once every 10 ms
  // or cause a keystroke
  // if it is not, then the computer may think that the device
  // has stopped working, and give errors

  if (digitalRead(PIN_BUTTON_CAPITAL_A) == LOW)
  {
    TrinketKeyboard.pressKey(KEYCODE_MOD_LEFT_SHIFT, KEYCODE_A);
    // this should type a capital A
    TrinketKeyboard.pressKey(0, 0);
    // this releases the key
  }

  if (digitalRead(PIN_BUTTON_STRING) == LOW)
  {
    // type out a string using the Print class
    TrinketKeyboard.print("Hello World! ");
  }
}
```

Two buttons are placed on a breadboard and connected to Pro Trinket pins 11 and 12. When the first button is pressed, it sends a capital letter A. If the second button is pressed, the string "Hello World! " is sent. You can use these function calls to send letters or strings of your choice.

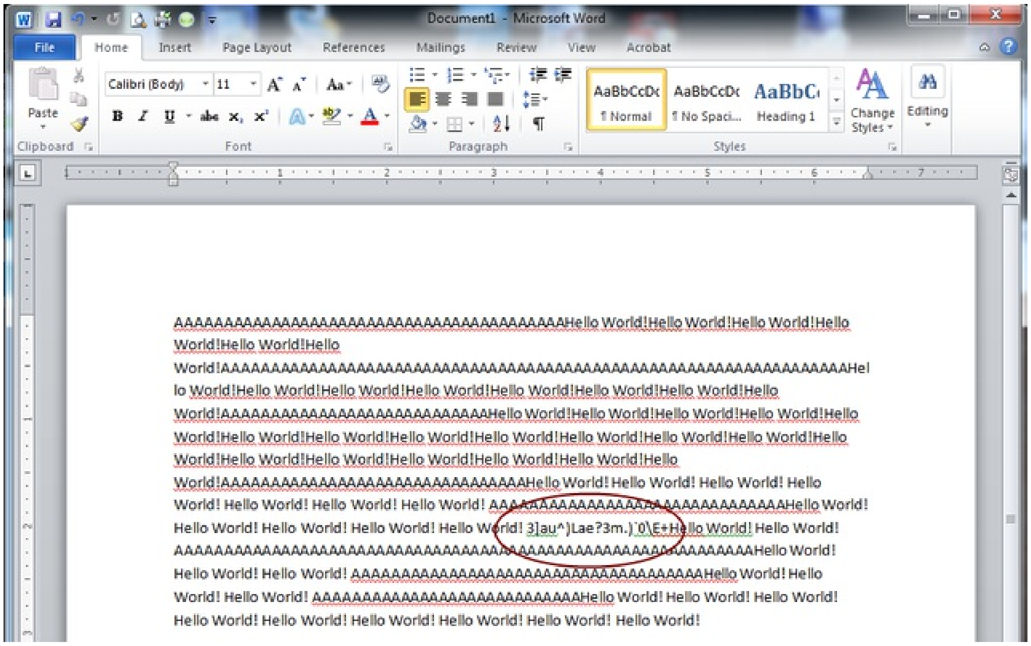
Plug your programmed Pro Trinket into an available USB port. In Windows, the device will show up in the Control Panel Devices and Printers dialog as "Pro Trinket Keyboard" with a keyboard icon as shown below:



The second example is kind of sneaky. It assumes you've plugged the Pro Trinket into a PC and it sends random characters to the open operating system window at random times (from 6 to 60 seconds between characters). Guaranteed to generate a "What the #$%*@!" from the person trying to use their computer.

```
/*
ProTrinketKeyboard prank example

For Pro Trinket (ATmega328P based Trinket) by Adafruit Industries
Please use library TrinketKeyboard for the ATtiny85 based Trinket

Version 1.0  2015-01-01 Initial Version by Mike Barela
*/

#include <ProTrinketKeyboard.h>  // include the Adafruit library

void setup()
{
  TrinketKeyboard.begin();  // initialize USB keyboard code
}

void loop()
{
  unsigned long secs_to_wait = random(6, 60); // wait 6 to 60 seconds between keys
  unsigned long time_stamp = millis();
  while (millis() < (time_stamp + (secs_to_wait * 1000))) // wait the random amount of time
  {
    TrinketKeyboard.poll();
    // the poll function must be called at least once every 10 ms
    // or cause a keystroke
    // if it is not, then the computer may think that the device
    // has stopped working, and give errors
  }
  TrinketKeyboard.typeChar((char)random(33, 122)); // type out a random character (valid ASCII)
}
```

The Microsoft Word screenshot below displays text mostly from the first example (A and Hello World!) but you can see in the circle where the sketch was changed to the random letter sketch.


Document1 - Microsoft Word
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World!Hello World!Hello World!Hello World!Hello World!Hello
World!AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello
World!AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello
World!AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World! Hello World! Hello World! Hello World! Hello World! Hello World! Hello World! AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World! Hello World! Hello World! Hello World! Hello World! 3]au^)Lae?3m.)`0\E+Hello World! Hello World!
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World! Hello World! Hello World! AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World! Hello World! Hello World! AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHello World! Hello World! Hello World! Hello World! Hello World! Hello World! Hello World! Hello World! Hello World!

Notes

This library is very specific to Pro Trinket and its USB connector.

If you wish to use the ATtiny85-based Trinket, see the Trinket Keyboard tutorial (https://adafru.it/rD2). The library for one cannot be used on the other.

Some good news is the library calls for the Pro Trinket Keyboard are the same as the Trinket Keyboard library. If you develop a sketch for either Trinket or Pro Trinket, just link to the appropriate library to make it work (barring changes in hardware pin connections). Pro Trinket has more memory and pins to use,. If you run out of resources on your Trinket project, consider swapping hardware for Pro Trinket!

Note the software can only emulate a USB 1.1 device. USB 1.1 cannot perform functions defined for USB 2 or USB 3 devices like virtual serial ports or mass storage devices.

If you wish to use a different development board than the Pro Trinket, it is suggested you consider the Arduino Micro, Leonardo, or Due. The Arduino IDE has built-in functions for keyboard and mice emulation. For more information, see the official Arduino website (https://adafru.it/ejW) for details.

How is the software implemented?

Trinket and Pro Trinket are based on V-USB (https://adafru.it/cJO). V-USB is a bit-bang implementation of USB. The code outputs 1s and 0s to the USB signal pins with precise timing.

V-USB is designed for AVR microcontrollers without native USB capabilities. USB signals have tight timing specifications that must be met, which is why the timing critical portions of V-USB are written in assembly code. Assembly instructions have predictable execution times and thus it is easier to calculate timing.

The ability to make a USB device without dedicated USB hardware is what makes V-USB so cool! Most of the microcontrollers on the market with native USB are bigger and cost more, but V-USB can turn two data lines on an AVR chip into an USB device!

To learn more about V-USB, check out the example projects (https://adafru.it/ejX) on the V-USB website.

Pro Trinket can do almost any of those projects as the Pro's ATmega 328P most often has enough pins and enough flash memory to use.