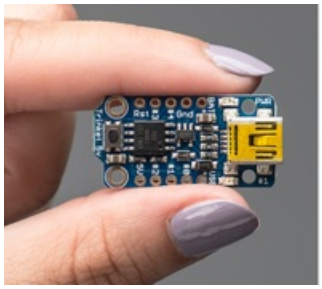
adafruit learning system

Trinket USB Keyboard

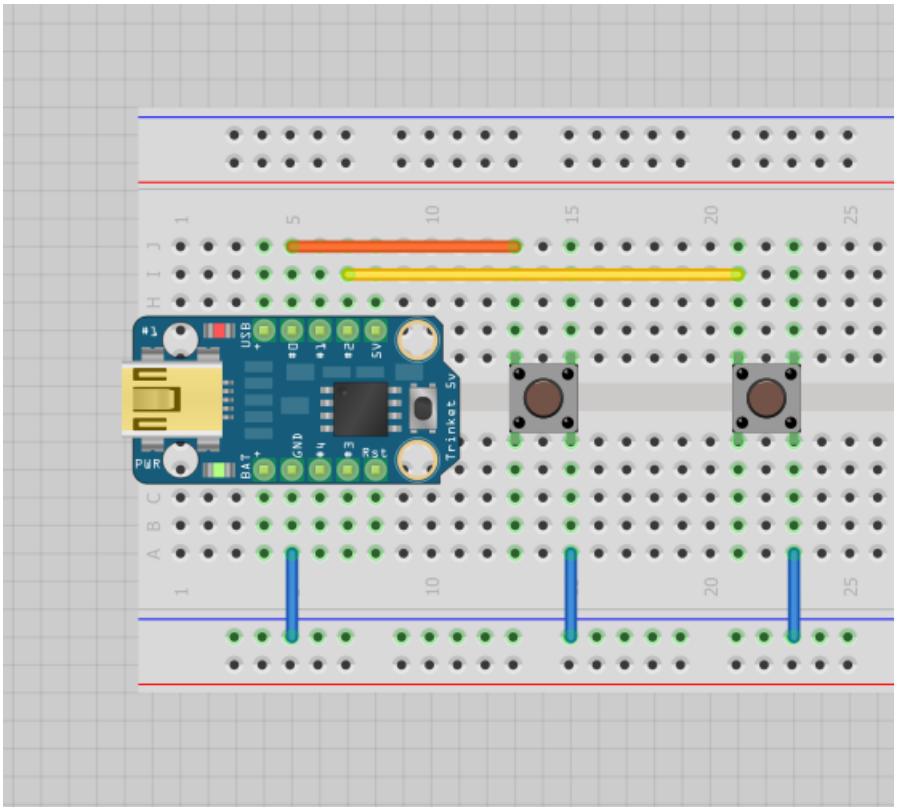
Created by Frank Zhao

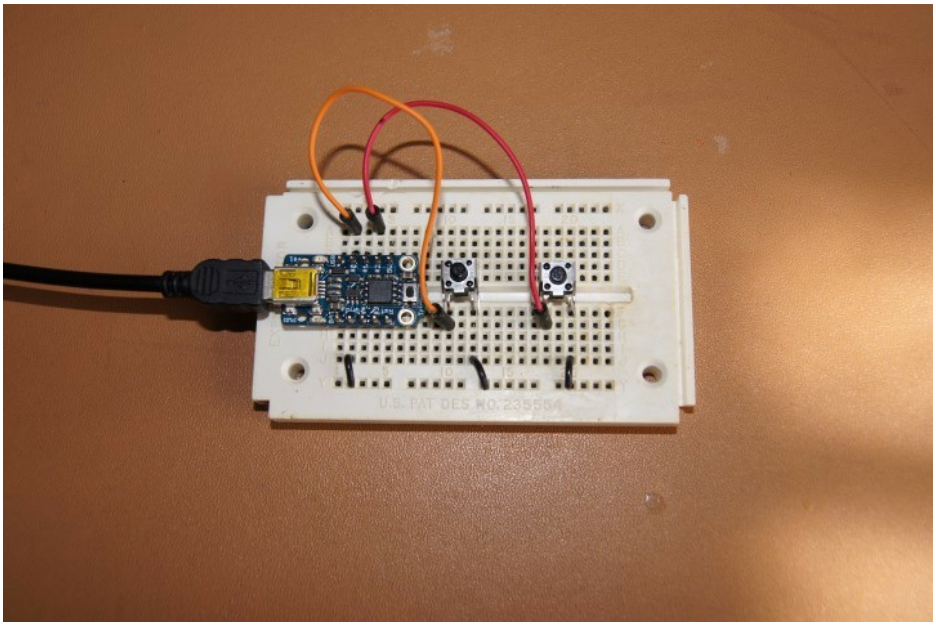


Last updated on 2020-01-23 05:01:51 PM UTC

Wiring

The wiring for the example is really simple. In the simplest terms, you will connect two active-low buttons to pin #0 and pin #2 on the Trinket. Active-low means the other end of the button is connected to ground, such that when the button is pressed, the pin value will read low. We will be using the Trinket's internal pull-up resistors so when the button is not pressed, the pin value will read high.



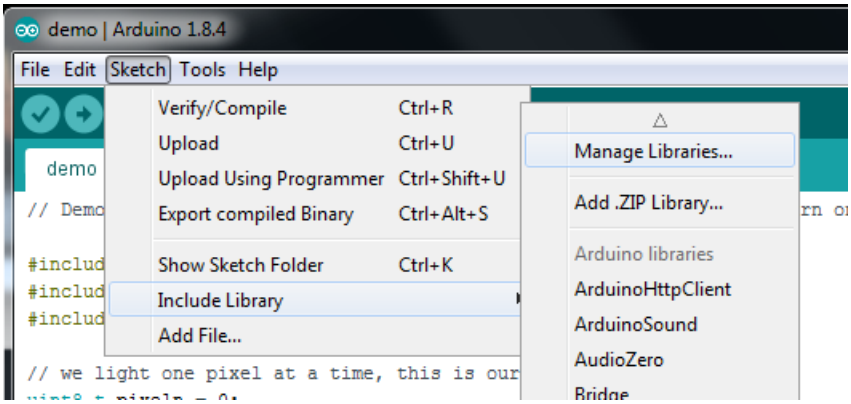


Code

The libraries are located at our github repository https://github.com/adafruit/Adafruit-Trinket-USB/ (https://adafru.it/cJL) . You can also just click the button below to download the ZIP file which contains both the Keyboard and Mouse libraries.

Install these as ordinary Arduino libraries, via the Arduino Library Manager (see http://arduino.cc/en/Guide/Libraries (https://adafru.it/cJM) or our tutorial on Arduino libraries (https://adafru.it/aYM)). Keep in mind that this library is designed specifically for Trinket. (although they can be modified to work with other platforms)

Open up the Arduino Library Manager:



Search for the **Trinket USB Keyboard** library and install it

The example code is really straightforward. The button pins are initialized as inputs with internal pull-up resistors enabled. The USB functionality starts with TrinketKeyboard.begin();, and then the loop will poll the buttons. If the button on pin #0 is pressed, the keyboard will type out 'A' until the button is released. If the button on pin #2 is pressed, the keyboard will type out "Hello World!" until the button is released.

```
/*
TrinketKeyboard example
For Trinket by Adafruit Industries
*/

#include <TrinketKeyboard.h>

#define PIN_BUTTON_CAPITAL_A 0
#define PIN_BUTTON_STRING    2

void setup()
{
  // 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()
{
  // 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.poll();

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

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

If you need some more debouncing, add a delay(3); line after the TrinketKeyboard.poll(); line

Prank Example

Now that you have the example working, you can make a funny prank device. This code will wait a random amount of time, and then randomly hit a key.

```
/*
TrinketKeyboard prank example
For Trinket by Adafruit Industries
*/

#include <TrinketKeyboard.h>

void setup()
{
  // start USB stuff
  TrinketKeyboard.begin();
  while (TrinketKeyboard.isConnected() == 0); // wait until connection
  randomSeed(millis()); // seed the RNG
  // the HOST takes some time before isConnected can return true
  // this amount of time is hard to predict, it depends on how busy the OS is
  // we use this unknown time as the seed
}

void loop()
{
  unsigned long secs_to_wait = random(60, 120); // generate a random amount of time
  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)
}
```

Plug this into your friend's computer and hide it. For extra credit you can modify the code to randomly press the CAPS LOCK key, which will *really* confuse your victim!

Learn More

Trinket is based on V-USB (https://adafru.it/cJO). It is a bit-bang implementation of USB. The generation of signals is done through assembly code, outputting 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.

This is what makes V-USB so cool! Most of the microcontrollers on the market with native USB are big, but V-USB can turn a 8 pin ATtiny85 into an USB device!

To learn more about V-USB, check out the example projects on the V-USB website:
http://www.obdev.at/products/vusb/projects.html (https://adafru.it/cJP)

Trinket can do almost any of those projects, as long as there are enough pins and enough flash memory to use.

Beware however, like I've mentioned in the overview, V-USB is only capable of creating low speed USB 1.1 devices. If you need to create USB 2.0 devices, you need a microcontroller with native USB such as the ATmega32u4 (used in the Arduino Micro & Leonardo), not "bit-banged" USB.