adafruit learning system

Mouse and Keyboard Control Using TinyUSB and BLE

Created by Chris Young

Last updated on 2020-05-13 04:55:51 PM EDT

Overview

With the introduction of the Arduino Leonardo and other ATMega32u4 based boards, Arduino introduced 3 new libraries **HID.h**, **Mouse.h** and **Keyboard.h** which allowed you to emulate a mouse or keyboard connected by USB to your computer. HID stands for "Human Interface Device" and refers to not only mouse and keyboard but other devices such as touchpads and game controllers. This opened up a world of possibilities especially for assistive technology applications for the disabled who need alternative ways to operate a computer.

This capability was extended when boards featuring the M0 (SAMD21 based) and M4 (SAMD51 based) systems became available. They were also able to perform mouse and keyboard emulation using the Arduino API originally developed for the Leonardo.

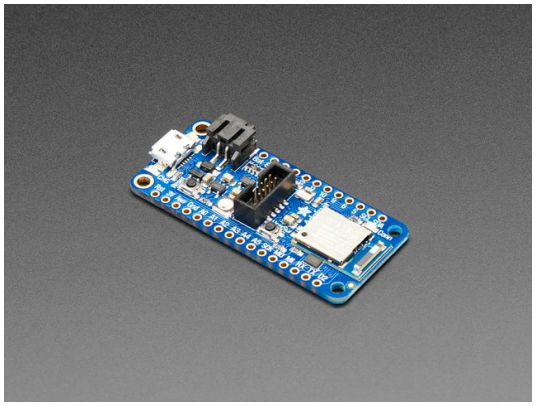
Newer boards and microprocessors have migrated to a new type of USB interface called TinyUSB. It is likely that most future boards will take advantage benefits of the TinyUSB platform. Additionally many new boards provide Bluetooth BLE capability that can emulate Bluetooth mouse and keyboard devices. Both TinyUSB and Adafruit Bluefruit libraries have powerful capabilities for emulating mouse and keyboard. However these APIs are not fully compatible with the old traditional Arduino **Mouse.h** and **Keyboard.h** APIs. Lots of legacy code has been written using these older Arduino APIs.

In this tutorial, 2 new libraries are presented which convert the traditional Arduino APIs into calls to the newer TinyUSB and Bluefruit commands. Although TinyUSB and Bluefruit HID interfaces provide capability not available in the traditional Arduino **Mouse.h** and **Keyboard.h** interfaces, these new conversion libraries will let you run legacy code and will be a stepping stone for new users to become familiar with mouse and keyboard emulation using a simpler interface. Using these simpler libraries does not require as much direct knowledge of USB and BLE protocols.

In the next section we will take a brief look at the traditional Arduino mouse and keyboard classes.

Parts

The parts may be any microcontroller compatible with TinyUSB. For Bluetooth use, this currently requires a nRF52840 microcontroller-based board. The following board is representable of the board that will work. See the TinyUSB GitHub repo (https://adafru.it/FFF) for compatible microcontrollers.



Adafruit Feather nRF52840 Express

$24.95
IN STOCK

Add To Cart

USB cable - USB A to Micro-B

$2.95
IN STOCK

Add To Cart

Understanding Arduino Mouse and Keyboard Classes

To use the original Arduino system to emulate a mouse or keyboard you would simply include the HID library followed by either the mouse library or keyboard library or both like this.

```
#include <HID.h>         //include the basic HID infrastructure first
#include <Mouse.h>
#include <Keyboard.h>
```

This would automatically create 2 global objects: one called `Mouse` and the other `Keyboard`. Here are the details of the public methods defined in **Mouse.h.**

```
#define MOUSE_LEFT 1
#define MOUSE_RIGHT 2
#define MOUSE_MIDDLE 4
#define MOUSE_ALL (MOUSE_LEFT | MOUSE_RIGHT | MOUSE_MIDDLE)

class Mouse_
{
public:
  Mouse_(void);
  void begin(void);
  void end(void);
  void click(uint8_t b = MOUSE_LEFT);
  void move(signed char x, signed char y, signed char wheel = 0);
  void press(uint8_t b = MOUSE_LEFT);   // press LEFT by default
  void release(uint8_t b = MOUSE_LEFT); // release LEFT by default
  bool isPressed(uint8_t b = MOUSE_LEFT); // check LEFT by default
};
extern Mouse_ Mouse;
```

The public methods are pretty much self-explanatory. The `click` method briefly presents and releases a mouse button. The default button if none is specified is the left button. The `move` method moves the mouse in the `x` (horizontal) and `y` (vertical) direction and an optional third parameter allows you to move the mouse wheel up or down. If you only want to move in one direction then the other parameters can be zero. The `press` and `release` methods allow you to press and hold a particular mouse button and only release it later. There is also a boolean method that allows you to test whether or not a particular button is currently pressed.

There is a `begin` and `end` method however close inspection of the actual code shows that they don't really do anything. But it's still a good idea to put a `Mouse.begin();` in your setup section of your sketch.

Although "Mouse.begin();" is necessary with the original Arduino libraries, you MUST do a call to the begin method when using our conversion libraries.

Here are the details of the public methods of the `Keyboard` class.

```
class Keyboard_ : public Print
{
public:
  Keyboard_(void);
  void begin(void);
  void end(void);
  size_t write(uint8_t k);
  size_t write(const uint8_t *buffer, size_t size);
  size_t press(uint8_t k);
  size_t release(uint8_t k);
  void releaseAll(void);
};
extern Keyboard_ Keyboard;
```

Again the methods are pretty much self-explanatory. There are 2 versions of the `write` method. One writes a single character while the other allows you to pass a pointer to a buffer full of characters with a length value. The `press` and `release` methods allow you to press and hold a particular key and then release it later. There is also a `releaseAll` method to release all pressed keys.

If the character you want to print is a standard printable ASCII character, you can simply send it using a character literal for example

```
Keyboard.write('A');
```

will write a capital letter "A" as if that key had been pressed. It takes care of upper and lowercase for you. However for non-printable characters you need to use special values for example

```
Keyboard.write(KEY_UP_ARROW);
```

will emulate the pressing of the up arrow key on your keyboard. A complete list of these values can be found on the Arduino.cc website at the link below and they are also in our libraries that we will be introducing later in the tutorial.

Arduino Keyboard Modifiers (https://adafru.it/KGF)

For example if you want to do do a control-c you would do

```
Keyboard.press(KEY_LEFT_CTRL);
Keyboard.write('c');
Keyboard.release(KEY_LEFT_CTRL);
```

In addition to the methods listed above, note that this class is also connected to the Print class. That means you can also use statements like

```
Keyboard.print("This is a bunch of text");
Keyboard.println("This text has a new line at the end");
Keyboard.print("Here's a number:");
Keyboard.println(My_Value,DEC);
```

There is a "begin" and "end" method however close inspection of the actual code shows that they don't really do anything. But it's still a good idea to put a `Keyboard.begin();` in your setup section of your sketch.

Although "Keyboard.begin();" is necessary with the original Arduino libraries, you MUST do a call to the begin method when using our conversion libraries.

More information about the Arduino libraries can be found in these links.

- Arduino Mouse Library (https://adafru.it/KHa)
- Arduino Keyboard Library (https://adafru.it/KHb)
- Arduino Serial.print(...) method also available as Keyboard.print(...) (https://adafru.it/KHc)

NOTE: The **Mouse.h** and **Keyboard.h** source code which we have quoted above contains the following copyright notice.

```
/*
  Copyright (c) 2015, Arduino LLC
  Original code (pre-library): Copyright (c) 2011, Peter Barrett

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/
```

Installing the Libraries

This tutorial describes 2 similar libraries but you will likely only need one of them for your application. The libraries are

- TinyUSB Mouse and Keyboard (https://adafru.it/KHd)
- BLE52 Mouse and Keyboard (https://adafru.it/KHe)

These libraries are not available through the Arduino IDE library manager. You will have to download and install them yourself. Click on either of the links above to visit the GitHub pages for these libraries or click on the buttons below for a direct link to download the zip files.

https://adafru.it/KHf

https://adafru.it/KHA

Those buttons will download a file such as **TinyUSB_Mouse_and_Keyboard-master.zip**. You should unzip the files into a folder without the word "-master" after it. Copy the folder into your Documents/Arduino/libraries folder.

For more information on how to manually install libraries into the Arduino IDE, check out the following learning guide.

https://adafru.it/m3e

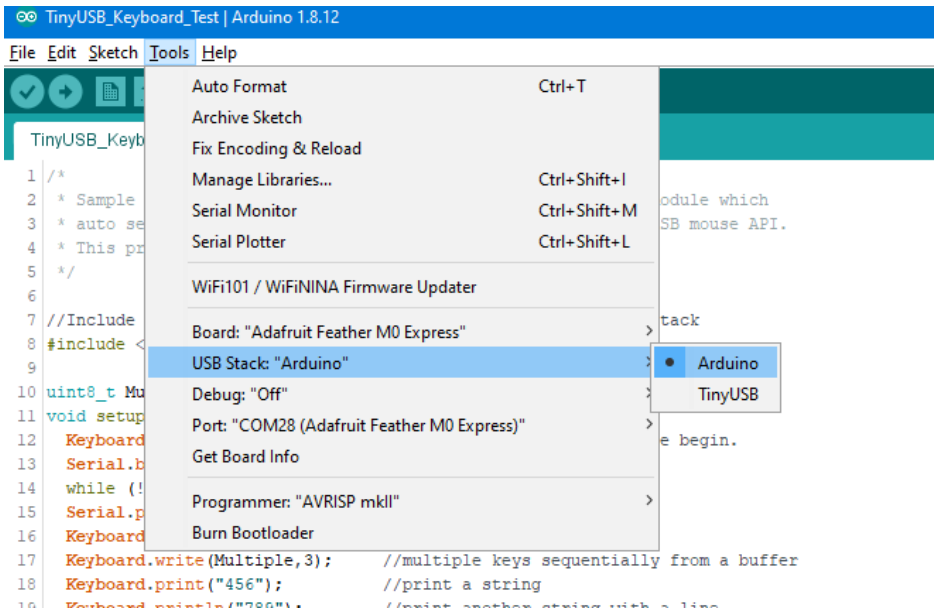
Each of the libraries has an **examples** folder that contains the sample code we will be using in the rest of the tutorial.

What is TinyUSB and Why Do I Need It?

There are two different methods of transmitting USB data between your microcontroller board and your PC. They are called "USB Stacks". They are layers of code that handle all of the protocols for transmitting data whether you are using it to upload your program to the board, receiving data via the serial monitor or serial plotter, or talking back to your computer emulating a mouse, keyboard or other device.

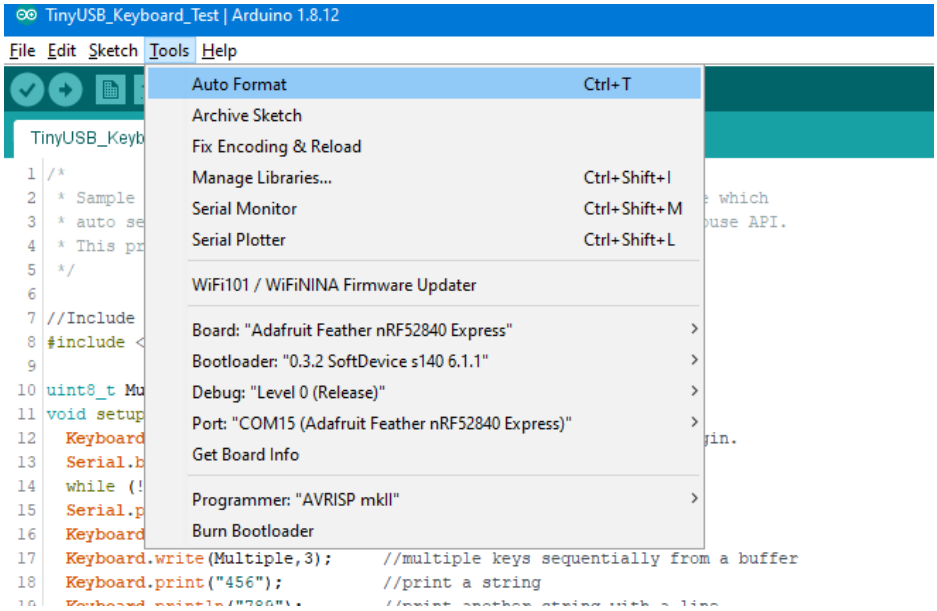
Traditional Arduino 8-bit boards all use the original Arduino stack. Newer boards such as M0 and M4 based on the SAMD21 and SAMD51 have the option of using either the Arduino stack or a different version called the TinyUSB Stack. Still other boards such as the nRF52840 based boards use only TinyUSB and it is likely that upcoming boards such as the ESP32-S2 will continue to use only TinyUSB. This is primarily because TinyUSB is the underlying architecture for implementing CircuitPython on these boards.

If you are using an M0 or M4 board you select which stack you want to use in the Tools menu of the Arduino IDE as shown below.



The image shows the tools menu of the Arduino IDE and we have selected an Adafruit Feather M0 Express board. Here you have a choice between using the Arduino stack or the TinyUSB stack.

However in the image below, we have configured for an Adafruit Feather nRF52840 Express (https://adafru.it/DLQ). As you can see there is no "USB Stack" option. What you cannot see is that this particular board only uses TinyUSB. If you try to #include it will not find the proper library because is not supported under TinyUSB.



For the M0 and M4 boards you can simply choose to select the Arduino stack and there's no problem. However the TinyUSB stack also has many other features that might be useful to you. Among them are the ability to use WebUSB and to use the onboard flash chip of your board as a mass storage device. This essentially turns your Feather board into a flash drive where you can drag-and-drop files. We will not be covering those capabilities in this tutorial. Of course if you're using the nRF52840 based systems don't have a choice. You have to use TinyUSB.

Using the TinyUSB Mouse and Keyboard Library

As mentioned previously, the traditional way to control mouse or keyboard is the following include files.

```
#include <HID.h>
#include <Mouse.h>
#include <Keyboard.h>
```

You should erase those lines and replace them with

```
#include <TinyUSB_Mouse_and_Keyboard.h>
```

This will automatically detect whether you are using the Arduino Stack or the TinyUSB Stack. If you are using the original Arduino Stack it will simply do the necessary include files for you. And if you're using TinyUSB Stack it will instead use its own code that works exactly like the originals. Note that there is no way to separate Mouse and Keyboard inclusion in our system. It was much easier to implement both at once rather than implementing them separately because of the way TinyUSB implements its HID functions. Theoretically when you compile your code if you did not make any reference to Keyboard and only to Mouse the linking loader will eliminate the Keyboard code. And vice versa if you use only Keyboard and not Mouse. Combining these into a single library saved us a lot of headaches.

If you have existing code that uses **Mouse.h** or **Keyboard.h** or both you should make the changes noted above and give it a try.

WARNING: If you did not do a Mouse.begin() or Keyboard.begin() in your code, you MUST add that in your setup section. The original mouse and keyboard did not require it but we do.

If you are using an M0 or M4 based board, you will have to set the Tools->USB Stack to "TinyUSB". In fact try switching back and forth between the two stacks and recompiling. You should see the same results using either stack. If you are using the nRF52840 processor, you do not need to select the TinyUSB option.

When to begin the Keyboard or Mouse

While developing and testing this library, we discovered that occasionally it makes a difference when you call the `Mouse.begin()` or `Keyboard.begin()` methods relative to the `Serial.begin(...)` method. Sometimes your computer would get confused as to how your USB was operating. Was it a mouse? Was it a keyboard? Was it a serial device? We had inconsistent results. Our best results came if you did your `Mouse.begin()` and/or `Keyboard.begin()` before doing `Serial.begin(...)`. No such restriction is necessary when using the BLE52 version of the library. It only affects the TinyUSB version.

In the next section, we will describe the BLE52 library followed by a series of three demonstration examples.

BLE52 Mouse and Keyboard Usage

Understanding Bluetooth BLE HID

The other library we used in this tutorial relied on communication over a USB cable between your Arduino device that was emulating mouse or keyboard and your PC, tablet or other similar device. However many PCs, tablets and other devices allow you to connect a mouse or keyboard using Bluetooth Low Energy or BLE for short. Establishing a Bluetooth link between your board and another device is pretty complicated programming. This library attempts to not only translate the traditional Arduino Mouse and Keyboard API calls into a BLE call, it also isolates you from all of the overhead necessary to establish and maintain a Bluetooth link.

Currently this library is called `BLE52_Mouse_and_Keyboard` and has only been tested using boards based on nRF52840 processors. It may work on older Adafruit Bluefruit BLE devices but we have not yet had the opportunity to test it on those platforms. So for now we are assuming you're using a board that has a nRF52840 processor.

In this section, we will try running one of the sample programs and pairing it with your PC, phone, or other device capable of using Bluetooth mouse or keyboard. We will show you how to do this pairing on a Windows 10 computer but the process should be straightforward on other operating systems or platforms.

Using the BLE52 Mouse and Keyboard Library

As previously mentioned, the traditional way to control a mouse or keyboard is the following include files.

```
#include <HID.h>
#include <Mouse.h>
#include <Keyboard.h>
```

You should erase those lines and replace them with

```
#include <BLE52_Mouse_and_Keyboard.h>
```

Note that there is no way to separate Mouse and Keyboard inclusion in our system. It was much easier to implement both at once rather than implementing them separately. Your board will present itself over Bluetooth as a single Bluetooth device with both mouse and keyboard capabilities. It is possible that if you do not use Keyboard for example and only Mouse that the linking loader will eliminate the Keyboard code from your final compile and vice versa if you use keyboard and not mouse. However the nRF52840 has plenty of program memory and it shouldn't hurt if we have to waste a little bit on applications that use mouse or keyboard but not both.

If you have existing code that uses **Mouse.h** or **Keyboard.h** or both you should makes the changes noted above and give it a try.

WARNING: If you did not do a Mouse.begin() or Keyboard.begin() in your code, you MUST add that in your setup section. The original mouse and keyboard did not require it but we do.

There is one more modification you will have to add to your code. You need to test to make sure that your nRF52840 board has been properly paired over Bluetooth with your PC or whatever device to which you are connecting. To facilitate this we have added an extra method not available in the standard **Mouse.h** and **Keyboard.h** classes. We have

added

```
bool Mouse.isConnected(void);
bool Keyboard.isConnected(void);
```

Below is a code excerpt from the **BLE52_Pairing_Test** sample program available in the **examples** folder of the library. It illustrates how to use the `isConnected()` method.

```
void setup() {
  Serial.begin(115200);
  while ( !Serial ) delay(10);   // for nrf52840 with native usb
  Serial.println("Bluefruit52 HID Mouse Example");
  Serial.println("Go to your phone's Bluetooth settings to pair your device");
  Serial.println("then open an application that accepts keyboard input");

  Mouse.begin();     //Unlike the standard Mouse.h you MUST use the "Mouse.begin();" method
  Serial.print("Attempting to connect");
  uint8_t i=0;
  while(! Mouse.isConnected()) {
    Serial.print("."); delay(100);
    if((++i)>50) {
      i=0; Serial.println();
    }
  };
  delay(1000);//just in case
  Serial.println("\nConnection successful");
```

After printing out an opening message, we do a `Mouse.begin()`. This attempts to initialize a BLE connection with your device. We then go into a while loop in which we test to see if `Mouse.isConnected()`. It prints out a series of dots on your serial monitor until they connection is established. When the connection is finally established, we break out of the while statement and report the successful connection. We then continue with the rest of the program.

If we were using a keyboard example we would use the `Keyboard.isConnected()` method instead.

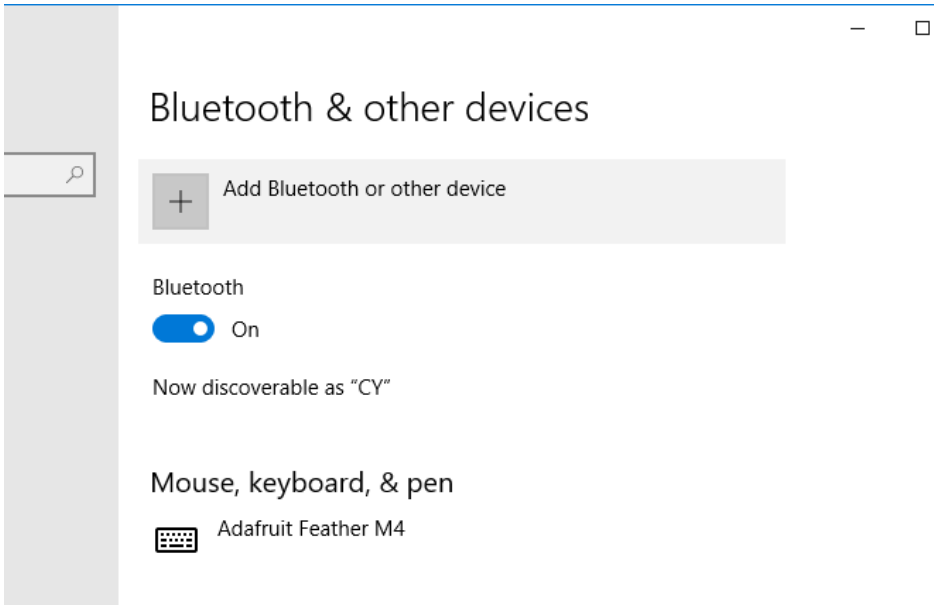
If your program uses both mouse and keyboard, you only need to check one or the other. Your nRF52840 appears to be a single device supporting both mouse and keyboard functions and so if one is connected, everything is connected.

Note if you are using mouse you must do a `Mouse.begin()` to initialize the connection and similarly `Keyboard.begin()` if you're using keyboard alone. If you are using both mouse and keyboard, technically you don't have to do `begin` for both but we suggest you do. At some point you might reuse this program for just one or the other function and forget to do the begin for the proper device. Internally we make sure that if you use both begins it only tries to initialize once.

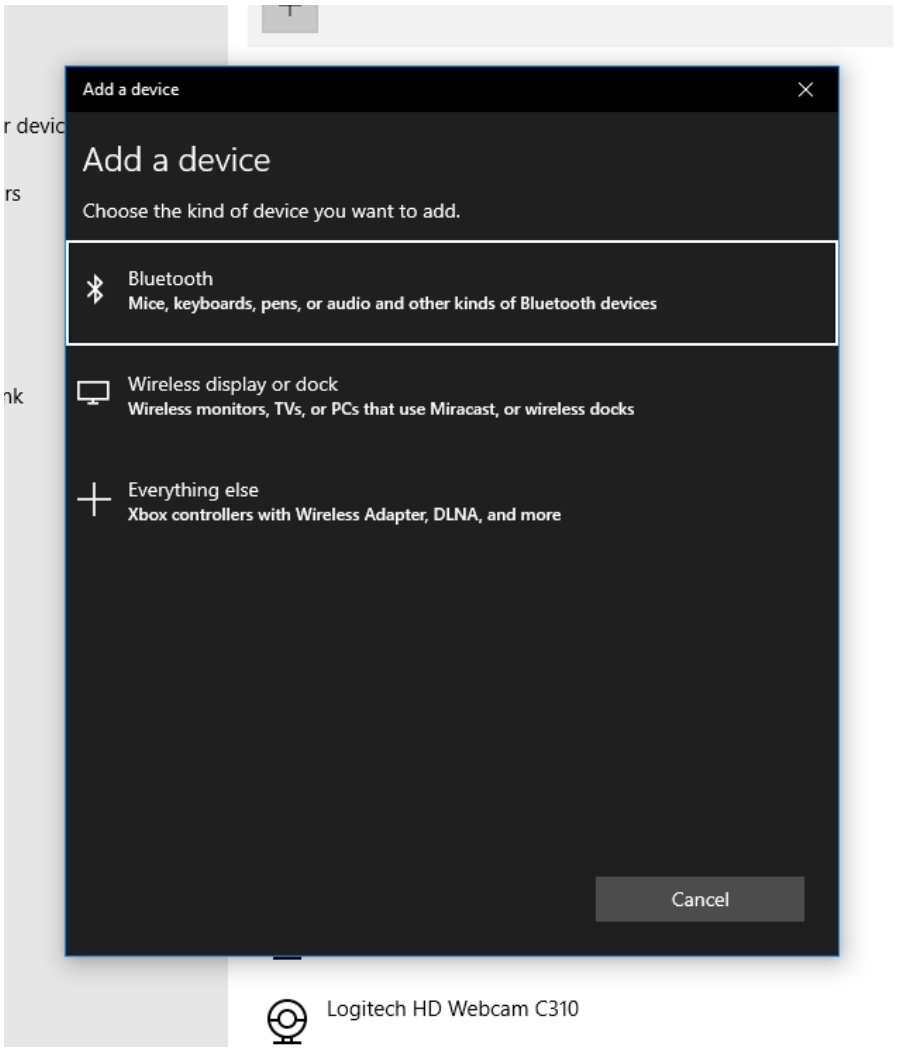
Pairing to a Windows 10 PC

There is a sample program in the **examples** folder titled **BLE52_Pairing_Test**. The same code in this program is also at the top of all of the other BLE sample programs. Configure your IDE for your nRF52840 and the proper port for uploading. Open your serial monitor and compile and upload the program.

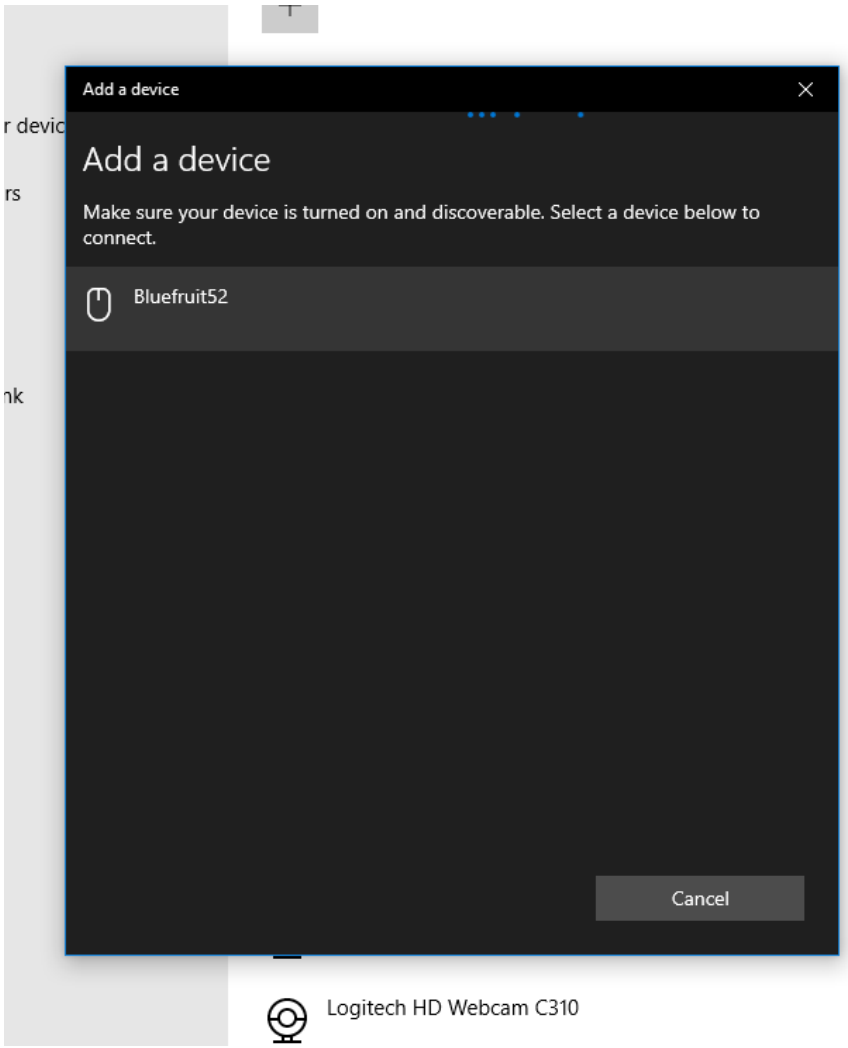
Then on your Windows 10 PC click on Start and then open your "Settings" page. Then click on "Devices. (Bluetooth, printers, mouse)" you will then see a page similar to this.



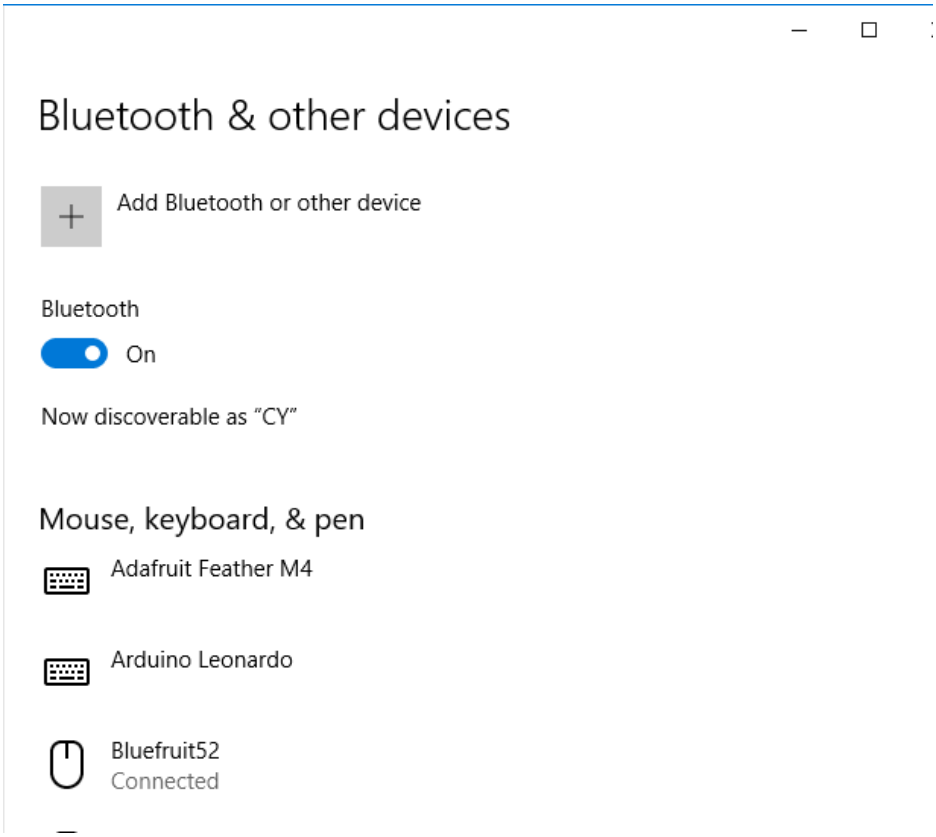
You should click on the + in front of "Add Bluetooth or other device". You will then see something like this.



Click on the Bluetooth option and after a few seconds you should see.



Your nRF52840 board will appear as "Bluefruit52". Later we will show you how you can use a different name. Click on it and then it should connect. When the pop-up closes you will now see the device listed among your available connected devices. Although it shows a mouse icon, it will have both mouse and keyboard capabilities.



Other Information About Pairing

While your attempt to pair the nRF52840 is in progress, you will see the following on your serial monitor.

```
Bluefruit52 HID Mouse Example
Go to your phone's Bluetooth settings to pair your device
then open an application that accepts keyboard input
Attempting to connect.....................................................
.....................................................
.....................................................
.....................................................
.....................................................
.....................................................
.....................................................
.....................................
Connection successful
```

Once you have done this process one time, you probably will not need to pair your device again. You will probably only see 4 or 5 dots appear after the attempting to connect message and then you will successfully connect.

As mentioned before, the default name for your nRF52840 will be "Bluefruit52". However if you want to use a different more personal name to help identify your gadget you can specify the name in your begin method. For example

```
Mouse.begin("MyDevice");
```

Note that you cannot name the mouse and keyboard two different things because they appear as a single multipurpose device.

We suggest you now try all three of the other sample programs for Mouse, Keyboard, and a combination of Mouse and Keyboard.

Mouse Example

In the **examples** folder of the TinyUSB_Mouse_and_Keyboard library, you will find 3 sample programs. The BLE52_Mouse_and_Keyboard library also has three similar files. There are only minor differences between the two. The BLE52 version will attempt to pair your nRF52840 with your PC or other device. See the previous section on how to do the pairing. Once it is paired you should not need to pair it again and you can run on the examples with no further effort. The TinyUSB mouse example looks like this.

```
/*
 * Sample program demonstrating TinyUSB_Mouse_and_Keyboard.h module which
 * auto selects between standard Arduino Mouse.h API and TinyUSB mouse API.
 * This program tests the mouse portion alone.
 */

//Include this module whether using Arduino stack or TinyUSB stack
#include <TinyUSB_Mouse_and_Keyboard.h>

void Status (void) {  //prints out mouse button status
  Serial.print("Mouse Buttons: Left:"); Serial.print(Mouse.isPressed(MOUSE_LEFT),DEC);
  Serial.print(" Right:"); Serial.print(Mouse.isPressed(MOUSE_RIGHT),DEC);
  Serial.print(" Middle:"); Serial.println(Mouse.isPressed(MOUSE_MIDDLE),DEC);
}

void setup() {
  Mouse.begin();  //Unlike Arduino Mouse.h, you must use "begin" method.
  Serial.begin(115200);
  while (! Serial)delay (1);
  Serial.println("USB mouse test");
  Mouse.click(MOUSE_LEFT);delay(1000);  //do a click
  Mouse.move(50,0);delay(1000);         //movie in all four directions
  Mouse.move(0,50);delay(1000);
  Mouse.move(-50,0);delay(1000);
  Mouse.move(0,-50);delay(1000);
  Mouse.move(0,0,-5);delay(1000);       //scroll down and up
  Mouse.move(0,0,5);delay(1000);
  Mouse.press(MOUSE_LEFT);delay(1000);  //hold the left button
  Mouse.move(50,0,0); delay(1000);      //drag
  Mouse.release(MOUSE_LEFT);delay(1000);//release the button
  Mouse.click(MOUSE_RIGHT);delay(1000); //try a right click
  Mouse.move(-30,-30); delay(1000);     //move away from the right-click menu that popped up
  Mouse.click(MOUSE_LEFT);delay(1000);  //do a left click to clear the pop-up
  Status();                             //print the buttons status
  Mouse.press(MOUSE_LEFT); delay(1000); //press each button, print status, then release all
  Status();
  Mouse.press(MOUSE_RIGHT); delay(1000);
  Status();
  Mouse.press(MOUSE_MIDDLE); delay(1000);
  Status();
  Mouse.release(MOUSE_ALL); delay(1000);
  Status();
  Mouse.move(-30,-30); delay(1000);     //move slightly to clear the pop-up
  Mouse.click(MOUSE_LEFT);delay(1000);  //clear of the pop-up
  Serial.println("Test completed");
};
void loop() {
}
```

Note that all of the code is in the `setup` section and there is nothing in the `loop`. This code will run just once each time you upload it.

Configure the Arduino IDE for your particular board and port to upload the code. If you are using an M0 or M4 based board be sure to select the Tools->USB Stack->TinyUSB option in your Arduino IDE.

If you are using USB connection on the nRF52840 or using the BLE52 version on the nRF52840 you will not have to select a different stack.

Open your serial monitor and compile and upload the code. While it is compiling, place your mouse over some of the text of the program in the Arduino IDE. During the test we will be moving the mouse around and occasionally dragging it to highlight some text. (Don't worry, we won't delete anything).

This sample program makes use of all of the methods and features of the Mouse class.

First we move the mouse 50 units in four directions with a one second delay between each movement. First 50 to the right, then down 50, left 50, and back up 50 returning to our original location. Note that X values increase to the right and Y values increase downwards.

It will move the scroll wheel up and down five units. Depending on the window size of your Arduino IDE and the amount of text you have on the screen you may or may not be able to see the scrolling action.

It does a left button press and leaves it held in place. It then moves 50 units to the right in a drag movement. It then releases the left mouse button. That should highlight whatever text was beneath it.

Next we are going to try a right click operation. That should pop up a context menu on your Arduino IDE. We then move out of the way and do another left click to clear the pop up menu. Now we will try pressing and holding all three mouse buttons and then releasing all of them. When we press the left button we get another pop-up and we clear that and the test is complete.

If you are using an M0 or M4 board you might want to try compiling the same code using the Arduino Stack. It should operate the same way using the traditional Arduino **Mouse.h** library.

Keyboard Example

In the **examples** folder of the TinyUSB_Mouse_and_Keyboard library you will find 3 sample programs. The BLE52_Mouse_and_Keyboard library also has three similar files. There are only minor differences between the two. The BLE52 version will attempt to pair your nRF52840 with your PC or other device. See the earlier section on how to do the pairing. Once it is paired you should not need to pair it again and you can run on the examples with no further effort. The TinyUSB keyboard example looks like this.

```
/*
 * Sample program demonstrating TinyUSB_Mouse_and_Keyboard.h module which
 * auto selects between standard Arduino Mouse.h API and TinyUSB mouse API.
 * This program tests the keyboard portion alone.
 */

//Include this module whether using Arduino stack or TinyUSB stack
#include <TinyUSB_Mouse_and_Keyboard.h>

uint8_t Multiple[3]= {'1','2','3'};
void setup() {
  Keyboard.begin();   //Unlike Arduino Keyboard.h, you must use begin.
  Serial.begin(115200);
  while (! Serial)delay (1);
  Serial.println("USB keyboard test");
  Keyboard.write('a');            //press and release 'a' key
  Keyboard.write(Multiple,3);     //multiple keys sequentially from a buffer
  Keyboard.print("456");          //print a string
  Keyboard.println("789");        //print another string with a line
  Keyboard.press(KEY_LEFT_SHIFT); //hold down the shift
  Keyboard.println("a uppercase sentence"); //this will be all caps
  Keyboard.release(KEY_LEFT_SHIFT);//release the shift
  Keyboard.println ("back to lowercase");
  Keyboard.press(KEY_LEFT_SHIFT); //press shift
  Keyboard.println("1234");       //some text
  Keyboard.releaseAll();          //release all
  Keyboard.println("1234");       //not shifted
  Keyboard.print("A mistake");  //will attempt to erase this
  delay(1000);
  Keyboard.press(KEY_LEFT_CTRL);  //will attempt control-z
  Keyboard.write('z');
  Keyboard.releaseAll();         //release the control key
  Serial.println("USB keyboard test completed");
};
void loop() {
}

/*
 * Click below before uploading and it will type characters in this comment
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

Like the previous Mouse example, all of the code is in the `setup` section and there is nothing in the `loop`. This code will run just once each time you upload it.

Configure the Arduino IDE for your particular board and port to upload the code. If you are using an M0 or M4 based board be sure to select the Tools->USB Stack->TinyUSB option in your Arduino IDE.

If you are using USB connection on the nRF52840 or using the BLE52 version on the nRF52840 you will not have to change the USB Stack.

Open your serial monitor and compile and upload the code. While it is compiling, click your mouse somewhere in the comment at the bottom of the program. The program will then type some text into that comment and it won't hurt anything else. When you are finished you can delete the text.

This sample program makes use of all of the methods and features of the `Keyboard` class.

First it uses the `Keyboard.write('a')` command to press and release the "a" key. Then it uses the buffered version of the `write` method to write the three characters "123". Next it uses the more versatile `Keyboard.print(...)` method to type the characters "456". Finally it uses `Keyboard.println(...)` method to type "789" and then it adds a new line at the end.

Next we press and hold the left shift key and type a sentence using `println(...)`. As you will see, even though the contents of that print statement is all in lowercase, the results come out in uppercase. We then use `Keyboard.release(KEY_LEFT_SHIFT)` to release the shift that we held down. When we use another `println(...)` To verify that we are back to lowercase. Note that this is a shift operation and not a caps lock. In the next series of commands we print "1234" with the shift key held and it comes out "!@#$".

In our last test we print out the words "A mistake" and then one second later we hold down the left control key, write a "z" and then release all pressed keys. The control-z does an undo operation in this text editor as is common for most text editors. That concludes our test.

If you are using an M0 or M4 board you might want to try compiling the same code using the Arduino Stack. It should operate the same way using the traditional Arduino Keyboard.h library.

Mouse and Keyboard Combined Example

In the **examples** folder of the TinyUSB_Mouse_and_Keyboard library you will find 3 sample programs. The BLE52_Mouse_and_Keyboard library also has three similar files. There are only minor differences between the two. The BLE52 version will attempt to pair your nRF52840 with your PC or other device. See the earlier section on how to do the pairing. Once it is paired you should not need to pair it again and you can run on the examples with no further effort. This TinyUSB example combines both mouse and keyboard capabilities. It looks like this.

```
/*
 * Sample program demonstrating TinyUSB_Mouse_and_Keyboard.h module which
 * auto selects between standard Arduino Mouse.h API and TinyUSB mouse API.
 * This program tests the mouse and keyboard portions together.
 */

//Include these modules whether using Arduino stack or TinyUSB stack
#include <TinyUSB_Mouse_and_Keyboard.h>

void setup() {
  //Don't really need much Serial output but we wanted to make sure it doesn't interfere
  Keyboard.begin(); //Unlike Arduino Keyboard.h and Mouse.h,
  Mouse.begin();    //    you must use "begin" method for both mouse and keyboard.
  Serial.begin(115200);
  while (! Serial)delay (1);
  Serial.println("USB mouse and keyboard test");
  Keyboard.println("The first line of text");   //write some text
  Keyboard.println("The second line of text");
  Serial.println("Wrote some text.");
  delay(2000);
  Mouse.click();
  Keyboard.press(KEY_LEFT_SHIFT);   //hold the shift key
  Mouse.move(100,0);                //move the mouse
  Mouse.click();                    //click again
  Keyboard.releaseAll();            //release the shift
  Serial.println("Highlighted the first line.");
  delay(2000);
  Keyboard.press(KEY_LEFT_CTRL);    //do a control-x
  Keyboard.write('x');              //press and release 'x' key
  Keyboard.releaseAll();            //release the control
  Serial.println("Cut the text");
  delay(2000);
  Mouse.move(0, 20);                //move the mouse down a bit
  Mouse.click();
  delay(2000);
  Keyboard.press(KEY_LEFT_CTRL);    //do a control-v
  Keyboard.write('v');              //press and release 'v' key
  Keyboard.releaseAll();            //release the control
  Serial.println("Pasted the text");
  Serial.println("Test completed.");
};
void loop() {
}

/*
 * Click just after the "*" on the on the next line before uploading
 *
 *
 *
 *
 *
 *
 *
 */
```

As mentioned previously, you should do both `Mouse.begin()` and `Keyboard.begin()` even though only one of them were actually run. They can be placed in either order. It's just a good idea to highlight the fact that this particular program uses both mouse and keyboard features.

Like the previous two examples, all of the code is in the `setup` section and there is nothing in the `loop`. This code will run just once each time you upload it.

Configure the IDE for your particular board and port to upload the code. If you are using an M0 or M4 based board be sure to select the Tools->USB Stack->TinyUSB option in your Arduino IDE.

If you are using USB connection on the nRF52840 or using the BLE52 version on the nRF52840 you will not have to configure the USB Stack.

Open your serial monitor and compile and upload the code. While it is compiling, click your mouse just after the "*" in the second line of the comment. We will then type some text on the next two lines. Then we will be positioned at the start of the first line. The program press and hold the shift key and move (but not drag) the mouse to the right. It will then left click the mouse which will select the text. This verifies that we can do things such as shift click in combination. The program will use a control-x to cut the text. We move the mouse down a ways and then then it below the second line of text. This completes our test.

If you're using an M0 or M4 board you might want to try compiling the same code using the Arduino Stack. It should work the same way using the traditional Arduino **Keyboard.h** and **Mouse.h** libraries.

Notice that approximately line 31 of the BLE52 version of this program we only check for `Mouse.isConnected()`. We previously mentioned that your board presents itself as a single USB or BLE device with both mouse and keyboard capabilities so you only need to check if one is connected.