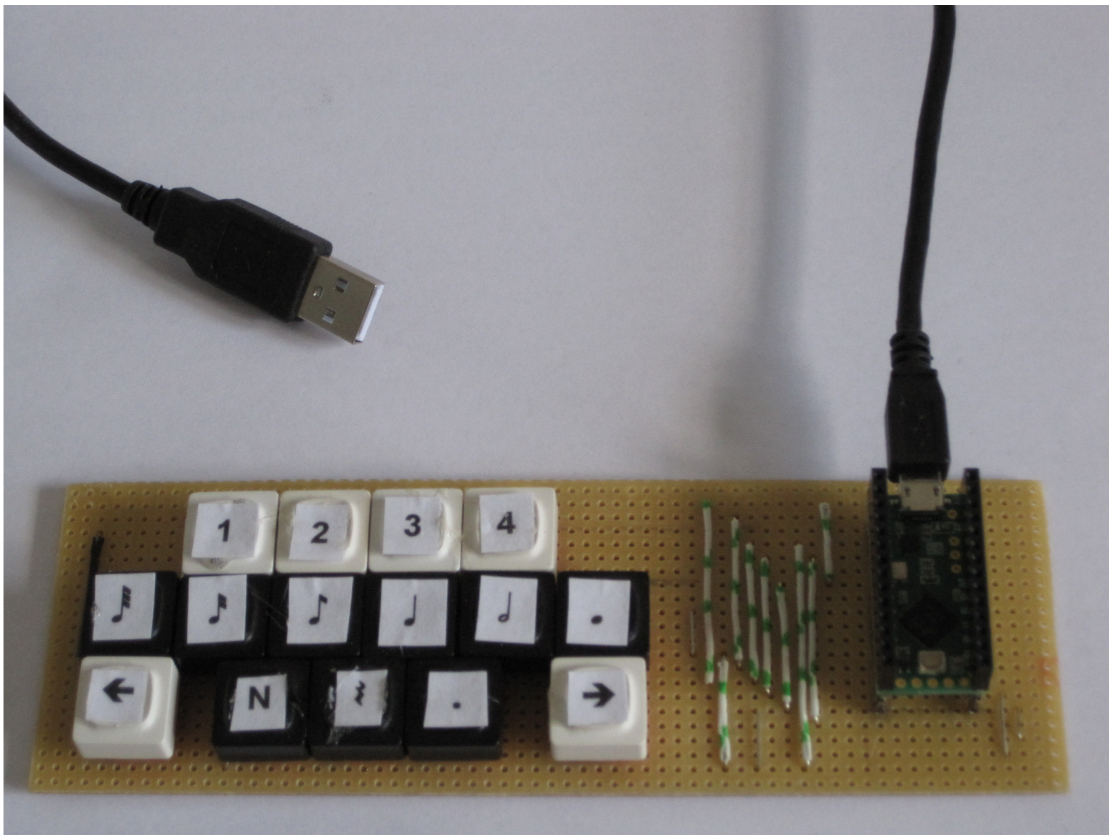
MuseScore Keypad

Introduction

Welcome to the MuseScore Keypad.

The aim of this hardware plugin is to produce a keypad that will help you to enter notes into MuseScore. The idea is to use a MIDI keyboard with one hand to enter the notes, and to use the MuseScore Keypad with the other hand to select note lengths, enter rests etc. This will avoid most of the need to constantly switch to using your mouse.
If you are a drummer, you may wish to investigate the keyboard shortcuts for the various drums and add those to your MuseScore Keypad. Perhaps your keypad layout would match the layout of a standard drum kit.



In the photo above, the keys labelled 1 – 4 are to select voices 1 to 4. “N” is to toggle note entry mode and “.” is the augmentation dot for entering dotted notes.

At the moment the MuseScore Keypad is just a working prototype. I was hoping to have some nice keys with pictures of crotchets, minims etc. but Maplin do not sell keyboard switches with removable tops any more. It would be ideal if the keyswitch for the crotchet (quarter note) had a raised bump, just like the F and J keys have on a standard keyboard, to make the key easier to find without looking down.

No drivers are required. The MuseScore Keypad emulates a standard keyboard, so it works with Windows and should be fine with Mac and Linux.

No external power supply is needed. The MuseScore Keypad is powered by the USB port.

I have tested the prototype with MuseScore 2.0.1 on Windows Vista.

Hardware

The hardware is very simple. All you need is:

- A Teensy LC (I bought one from HobbyTronics http://www.hobbytronics.co.uk/teensy-boards/teensy-lc).
- Some PCB-mounting keyboard switches.
- A USB A to micro B USB cable.
- A bit of stripboard, solder and some wire.

Development Software

You will need to download and install the following (free) software to build the code and upload it to the Teensy LC:

- Arduino programming software (arduino-1.6.5-r2-windows).
- TeensyDuino plugin for the Arduino programming software.
- The Teensy loader.

The instructions for downloading and installing the code are here:
https://www.pjrc.com/teensy/td_download.html
https://www.pjrc.com/teensy/loader.html

Getting Started

Remember that the Teensy LC is a static-sensitive piece of electronics, so it can be damaged by the static charge that builds up on our bodies. To reduce the likelihood of damage:

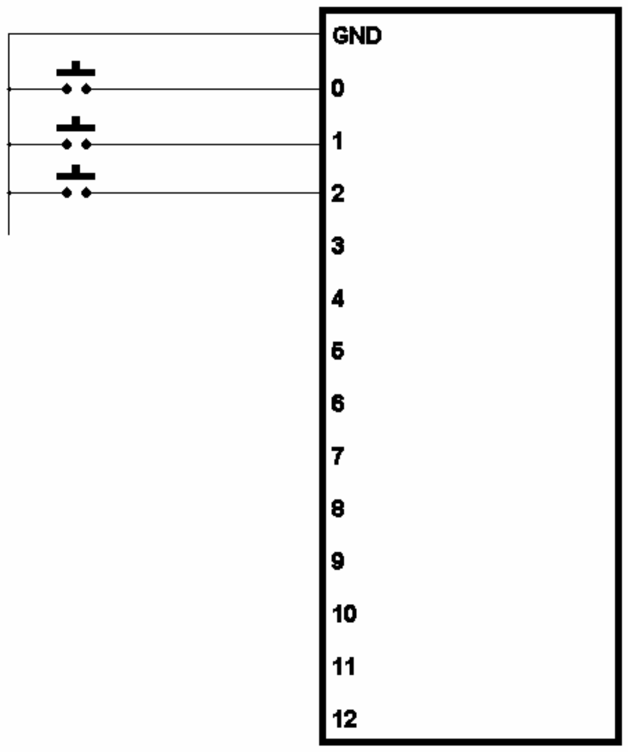
- Wear an earthing strap.
- Handle the Teensy LC as little as possible, and only by the edges.

Before you rush off to compile the code, start by plugging your Teensy LC into a USB port. The LED on the Teensy LC board should flash.

Press and release the small button on the board and it will stop flashing. Unplug it now and read on.

MuseScore Keypad Hardware

To build the hardware, all you need to do is connect one side of your keypad switches to the GND connection on the Teensy, and the other to the connections labelled 0 to 14. There is no need to use pull-up resistors.



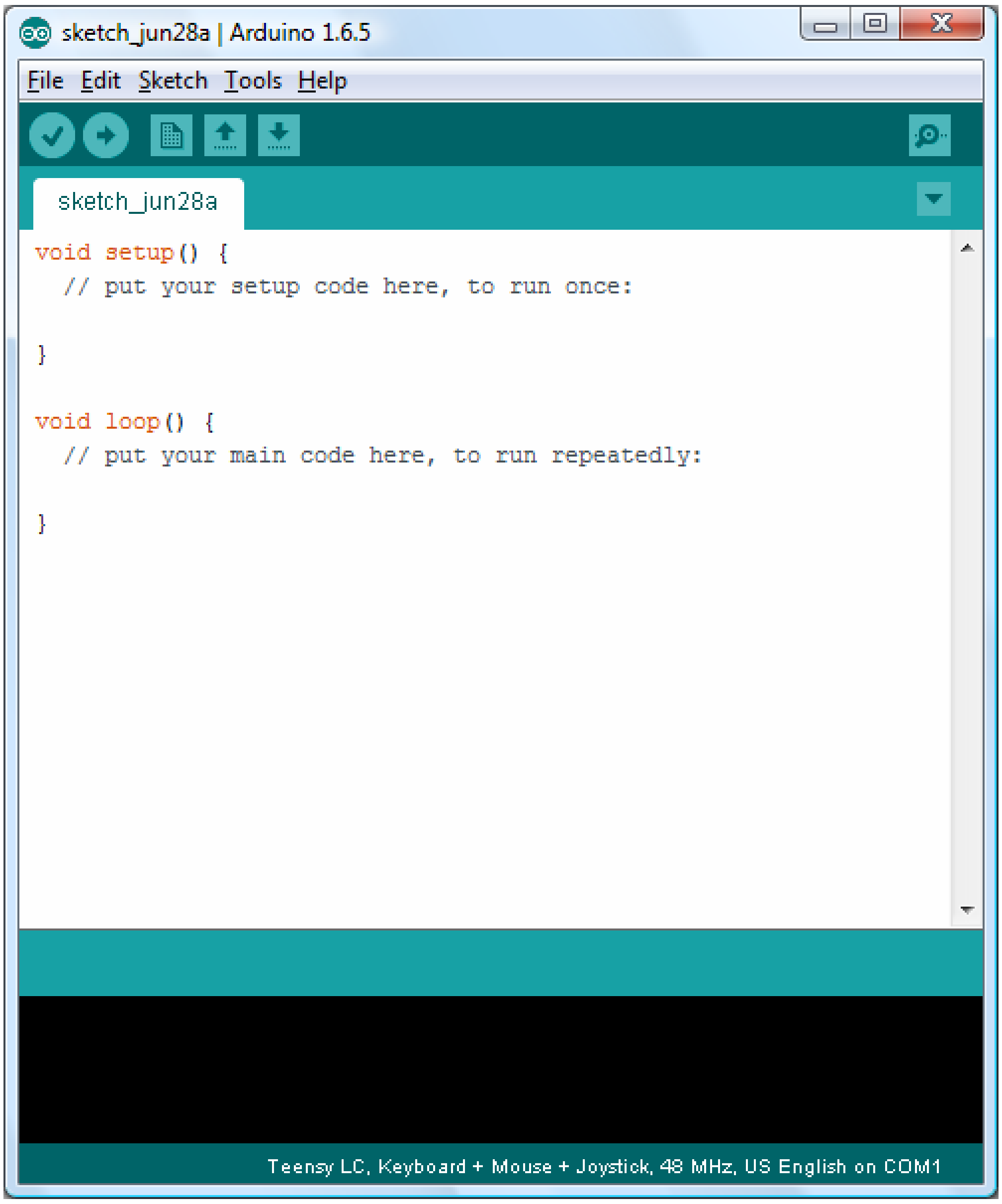
MuseScore Keypad Software

The MuseScore Keypad works just like a normal USB keyboard, so it can use the keyboard shortcuts that are configured into MuseScore.

The TeensyDuino plugin software comes with some example code that just needs a slight change before loading into the Teensy LC.

You can modify the code so that pushing a button on the MuseScore Keypad will use one of the MuseScore keyboard shortcuts. For example, you can modify the code to write “5” and so select a crotchet (quarter note) for the next note that you enter.
You can write “0” to enter a rest.

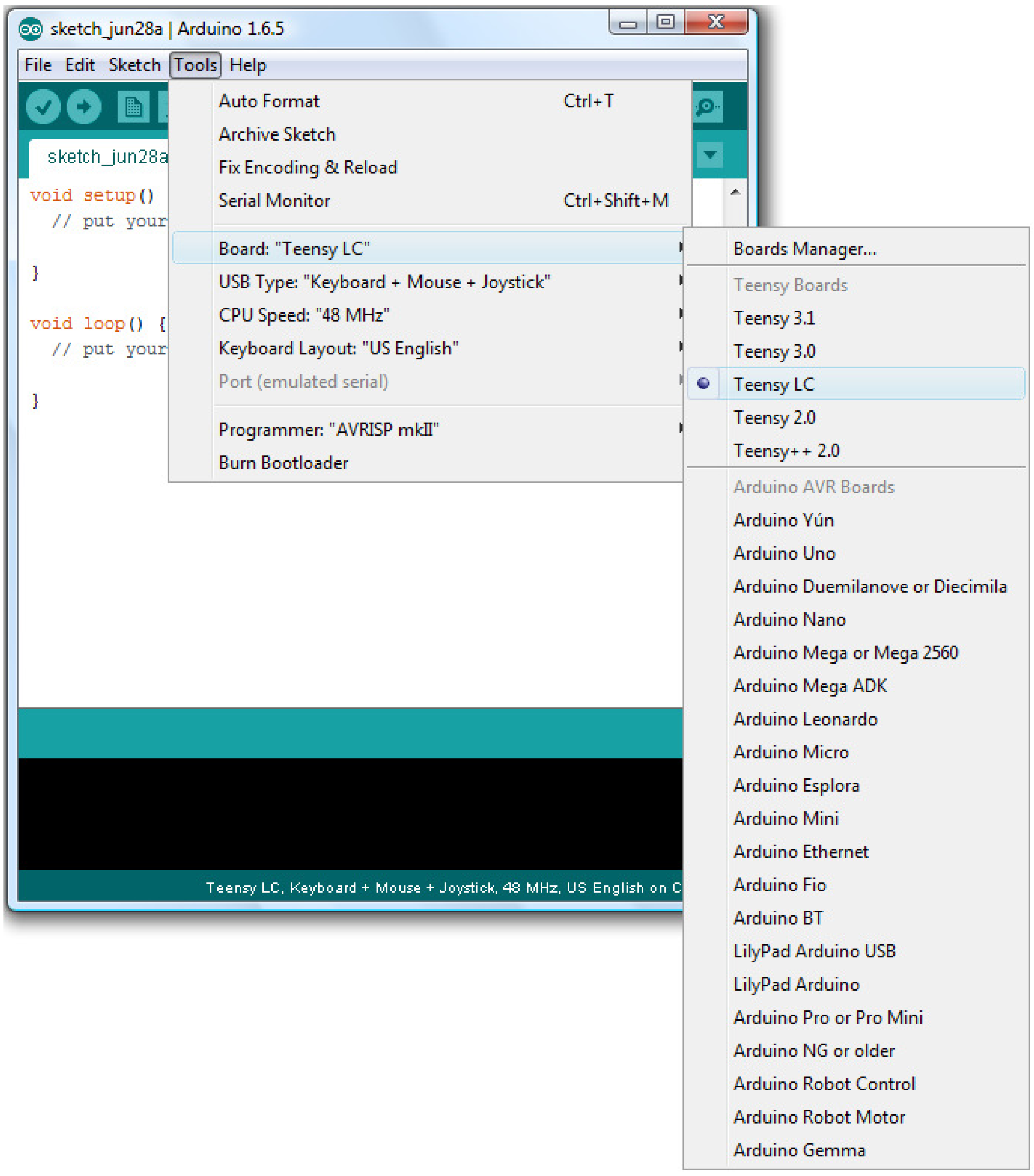
Start the Arduino programming software:



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Teensy LC, Keyboard + Mouse + Joystick, 48 MHz, US English on COM1

From the Tools menu, select Board and “Teensy LC”:



sketch_jun28a | Arduino 1.6.5

File Edit Sketch Tools Help

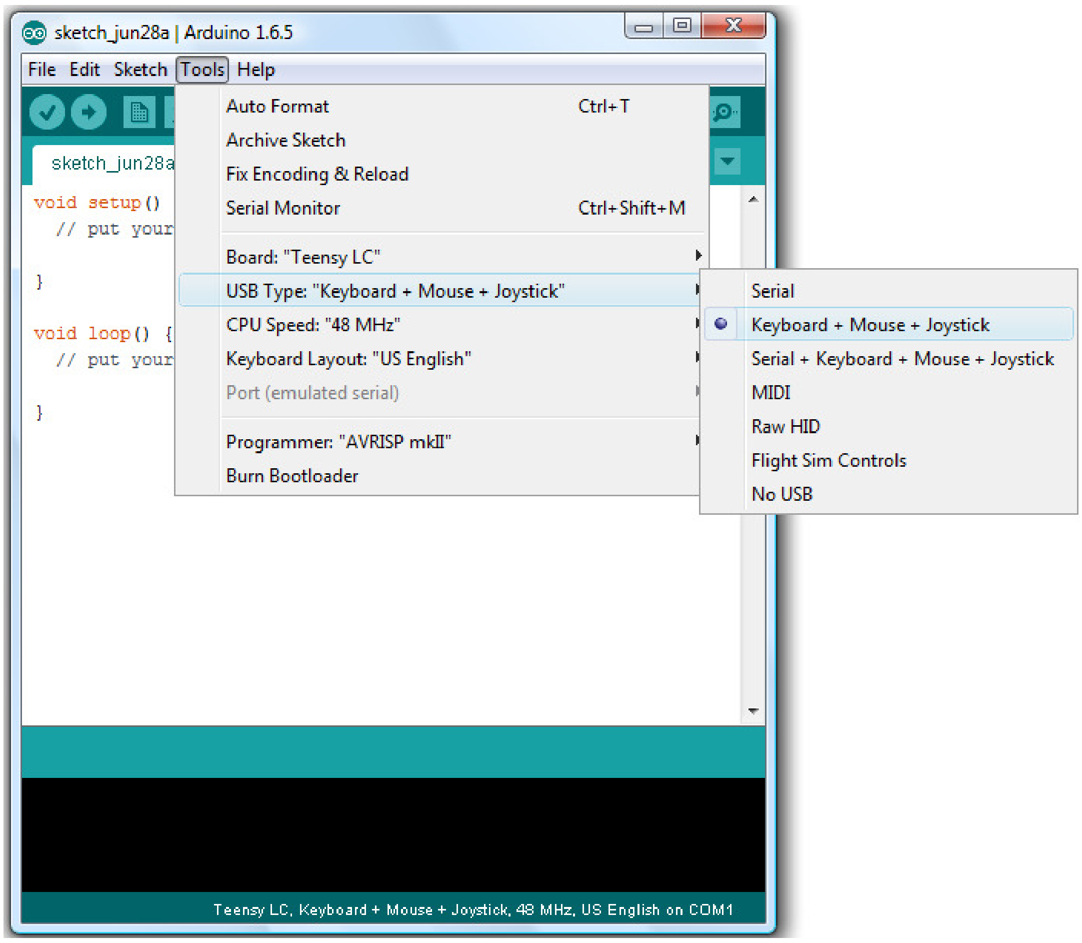
```
void setup()
  // put your
}

void loop() {
  // put your
}
```

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Teensy LC"
USB Type: "Keyboard + Mouse + Joystick"
CPU Speed: "48 MHz"
Keyboard Layout: "US English"
Port (emulated serial)
Programmer: "AVRISP mkII"
Burn Bootloader

Boards Manager...
Teensy Boards
Teensy 3.1
Teensy 3.0
● Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino AVR Boards
Arduino Yún
Arduino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma

Teensy LC, Keyboard + Mouse + Joystick, 48 MHz, US English on C

From the Tools menu again, select USB Type and “Keyboard + Mouse + Joystick”


sketch_jun28a | Arduino 1.6.5

File Edit Sketch Tools Help

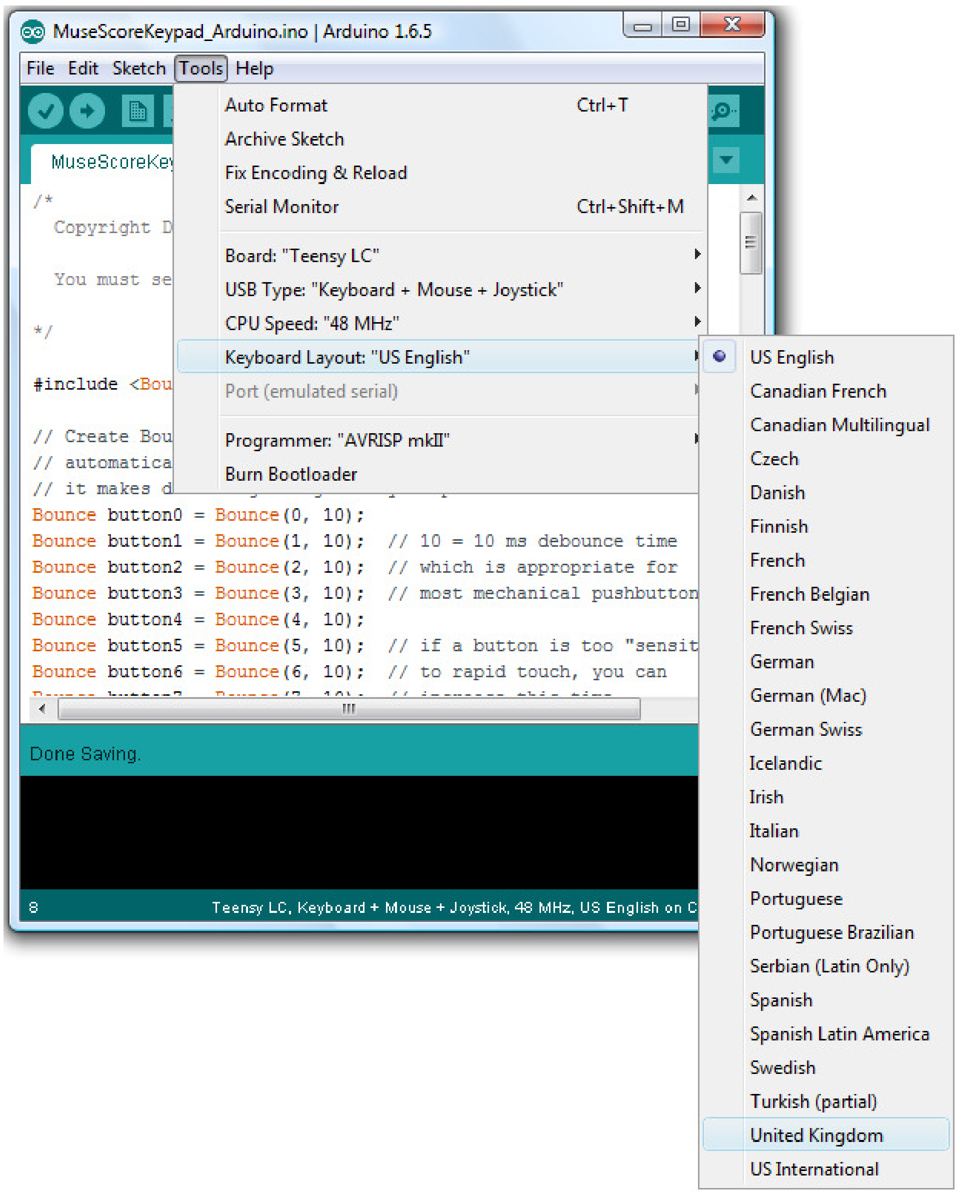
```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Teensy LC"
USB Type: "Keyboard + Mouse + Joystick"
CPU Speed: "48 MHz"
Keyboard Layout: "US English"
Port (emulated serial)
Programmer: "AVRISP mkII"
Burn Bootloader

Serial
● Keyboard + Mouse + Joystick
Serial + Keyboard + Mouse + Joystick
MIDI
Raw HID
Flight Sim Controls
No USB

Teensy LC, Keyboard + Mouse + Joystick, 48 MHz, US English on COM1

And again from the Tools menu, select Keyboard Layout and “United Kingdom”.

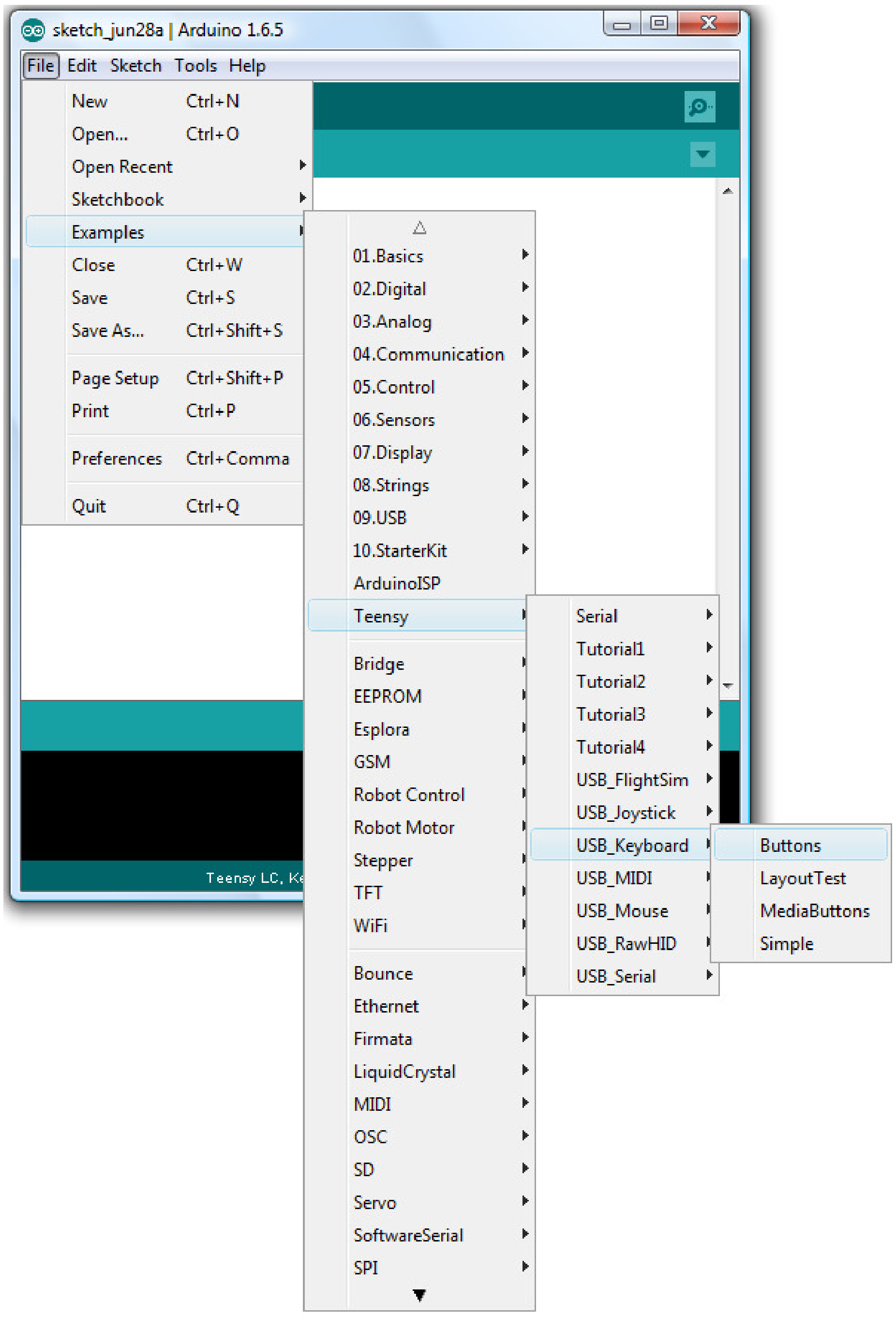


```
/*
  Copyright D

  You must se

*/

#include <Bou

// Create Bou
// automatica
// it makes d
Bounce button0 = Bounce(0, 10);
Bounce button1 = Bounce(1, 10);  // 10 = 10 ms debounce time
Bounce button2 = Bounce(2, 10);  // which is appropriate for
Bounce button3 = Bounce(3, 10);  // most mechanical pushbutton
Bounce button4 = Bounce(4, 10);
Bounce button5 = Bounce(5, 10);  // if a button is too "sensit
Bounce button6 = Bounce(6, 10);  // to rapid touch, you can
```

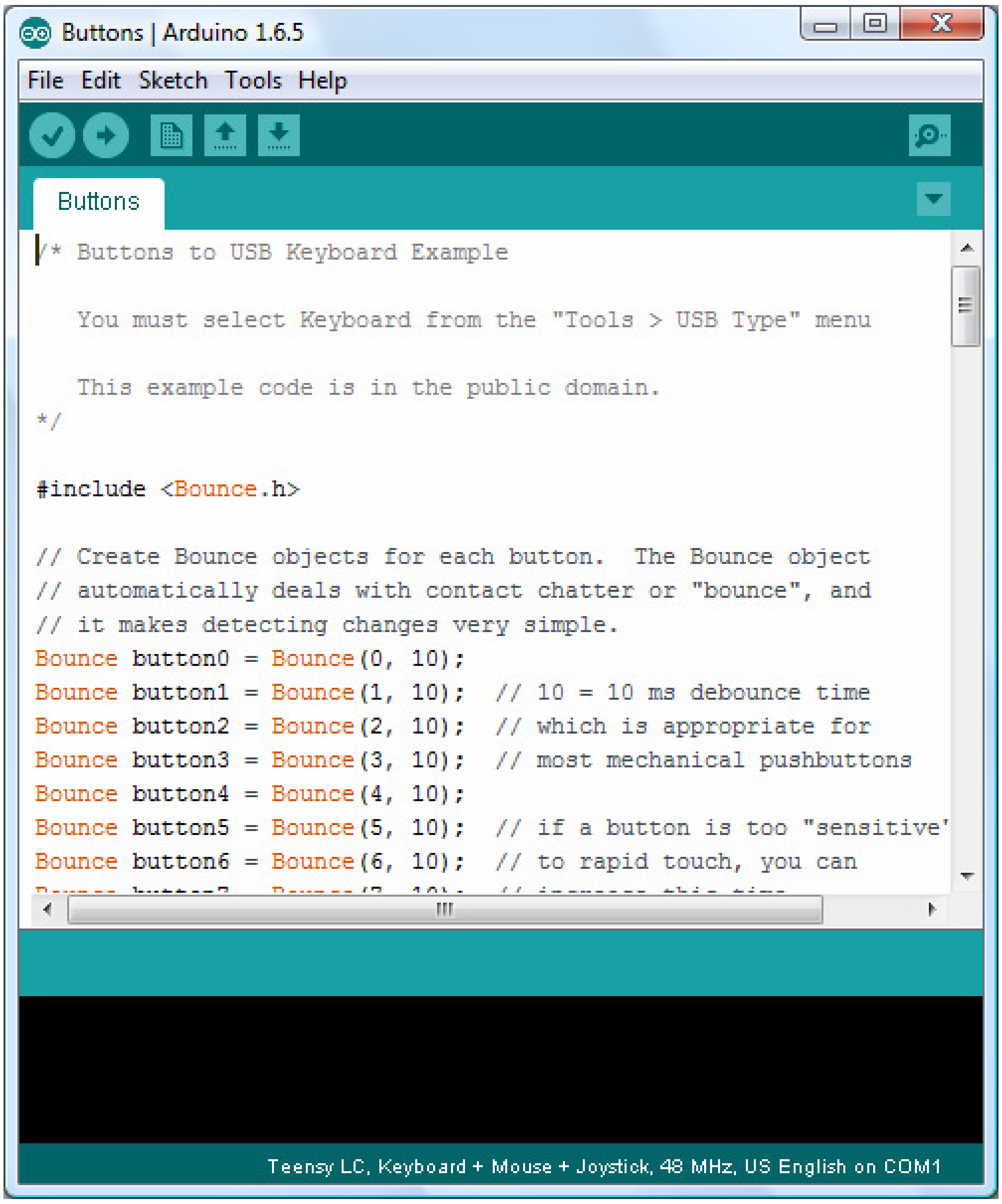
Done Saving.

8 Teensy LC, Keyboard + Mouse + Joystick, 48 MHz, US English on C

Load the demo code by selecting the File menu and then Examples | Teensy | USB_Keyboard | Buttons:



This opens a new window, which displays the example code:



```
/* Buttons to USB Keyboard Example

   You must select Keyboard from the "Tools > USB Type" menu

   This example code is in the public domain.
*/

#include <Bounce.h>

// Create Bounce objects for each button.  The Bounce object
// automatically deals with contact chatter or "bounce", and
// it makes detecting changes very simple.
Bounce button0 = Bounce(0, 10);
Bounce button1 = Bounce(1, 10);  // 10 = 10 ms debounce time
Bounce button2 = Bounce(2, 10);  // which is appropriate for
Bounce button3 = Bounce(3, 10);  // most mechanical pushbuttons
Bounce button4 = Bounce(4, 10);
Bounce button5 = Bounce(5, 10);  // if a button is too "sensitive'
Bounce button6 = Bounce(6, 10);  // to rapid touch, you can
```

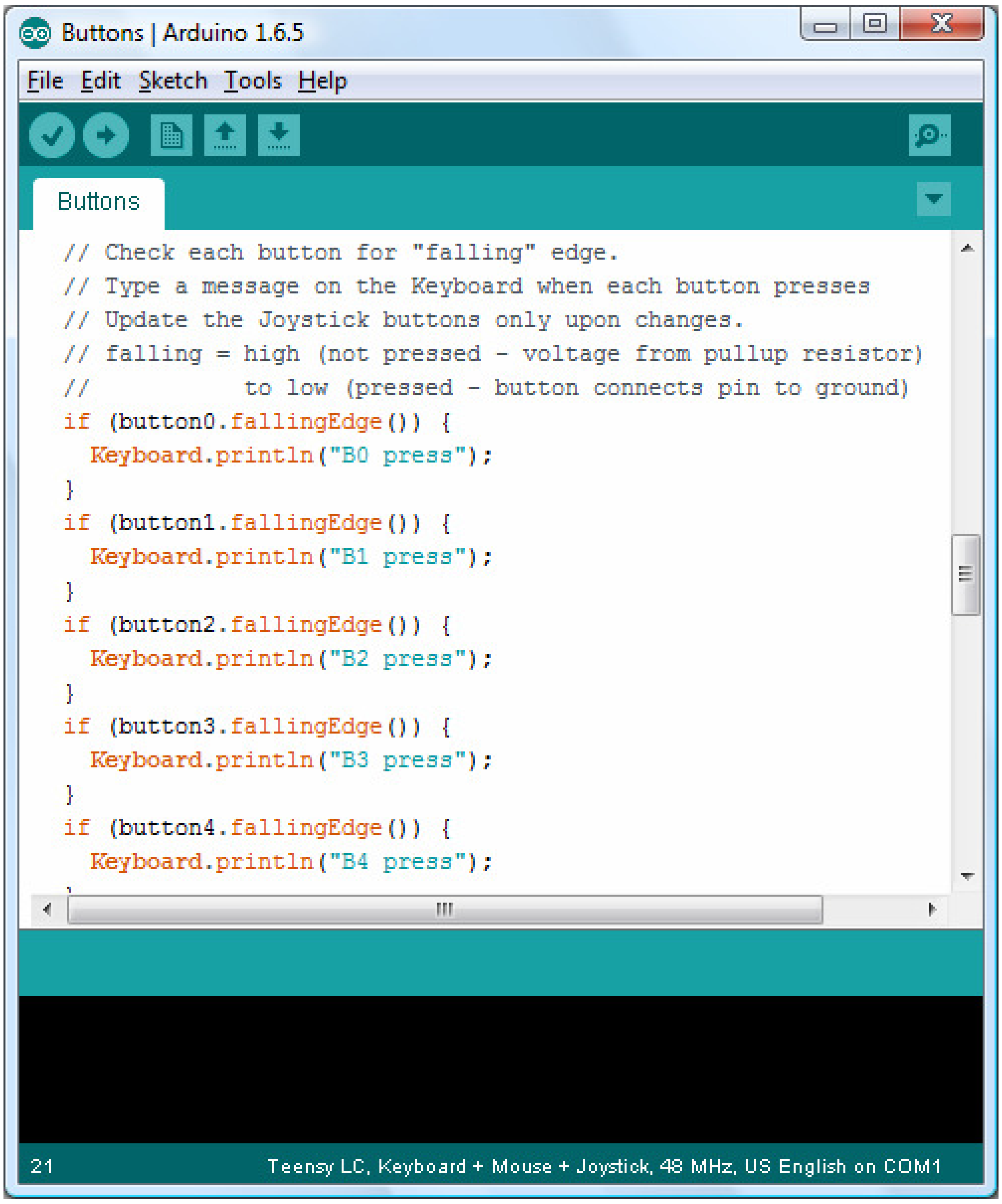
Teensy LC, Keyboard + Mouse + Joystick, 48 MHz, US English on COM1

You can try running this code.
To test, start a text editor or word processor. Make sure it is the active program, then VERY CAREFULLY look on the underside of the Teensy LC and connect any one of the pads labelled 0 to 9 to one of the pads labelled GND. You just need to touch the wire to the pad briefly.

The text editor or word processor will display some text.

MuseScore Keypad Code

If you scroll down, you will find the following short piece of code:



```
// Check each button for "falling" edge.
// Type a message on the Keyboard when each button presses
// Update the Joystick buttons only upon changes.
// falling = high (not pressed - voltage from pullup resistor)
//           to low (pressed - button connects pin to ground)
if (button0.fallingEdge()) {
  Keyboard.println("B0 press");
}
if (button1.fallingEdge()) {
  Keyboard.println("B1 press");
}
if (button2.fallingEdge()) {
  Keyboard.println("B2 press");
}
if (button3.fallingEdge()) {
  Keyboard.println("B3 press");
}
if (button4.fallingEdge()) {
  Keyboard.println("B4 press");
```

```
21          Teensy LC, Keyboard + Mouse + Joystick, 48 MHz, US English on COM1
```

For the MuseScore Keypad, all we need to do is to replace the pieces of text like “B0 press” with something more useful, like the “2” to “7” to set the note lengths, “0” to enter a rest and perhaps a few other keys, depending on how many keyboard switches you have. The more I thought about this, the more keys I thought would be useful.

My suggested mappings are:

Digital Input	Keyboard text	Description
0	0	Enter a rest using the current note length.
1	. (dot)	Toggle the augmentation dot.
2	2	Demi-semi quaver (32nd note).
3	3	Semi-quaver (16th note).
4	4	Quaver (8th note).
5	5	Crotchet (quarter note).
6	6	Minim (half note).
7	7	Semibreve (whole note).
8	n	Toggle note entry mode.
9	Ctrl-Alt-1	Voice 1.
10	Ctrl-Alt-2	Voice 2.
11	Ctrl-Alt-3	Voice 3.
12	Ctrl-Alt-4	Voice 4.
13	Left	Move left.
14	Right	Move right.

Here is the complete software for the suggested mappings. You can cut and paste this into the Arduino programming software.

```
/*
  Copyright Don Horrell 2015. All rights reserved.

  You must select Keyboard from the "Tools > USB Type" menu.

*/

#include <Bounce.h>
#define DEBOUNCE_MS 15

// Create Bounce objects for each button.  The Bounce object
// automatically deals with contact chatter or "bounce", and
// it makes detecting changes very simple.
Bounce button0 = Bounce(0, DEBOUNCE_MS);
Bounce button1 = Bounce(1, DEBOUNCE_MS);  // 10 = 10 ms debounce time
Bounce button2 = Bounce(2, DEBOUNCE_MS);  // which is appropriate for
Bounce button3 = Bounce(3, DEBOUNCE_MS);  // most mechanical
pushbuttons
Bounce button4 = Bounce(4, DEBOUNCE_MS);
Bounce button5 = Bounce(5, DEBOUNCE_MS);  // if a button is too
"sensitive"
Bounce button6 = Bounce(6, DEBOUNCE_MS);  // to rapid touch, you can
Bounce button7 = Bounce(7, DEBOUNCE_MS);  // increase this time.
Bounce button8 = Bounce(8, DEBOUNCE_MS);
Bounce button9 = Bounce(9, DEBOUNCE_MS);
Bounce button10 = Bounce(10, DEBOUNCE_MS);
Bounce button11 = Bounce(11, DEBOUNCE_MS);
Bounce button12 = Bounce(12, DEBOUNCE_MS);
Bounce button13 = Bounce(13, DEBOUNCE_MS);
Bounce button14 = Bounce(14, DEBOUNCE_MS);

void setup() {
  // Configure the pins for input mode with pullup resistors.
  // The pushbuttons connect from each pin to ground.  When
  // the button is pressed, the pin reads LOW because the button
  // shorts it to ground.  When released, the pin reads HIGH
  // because the pullup resistor connects to +5 volts inside
  // the chip.  LOW for "on", and HIGH for "off" may seem
  // backwards, but using the on-chip pullup resistors is very
  // convenient.  The scheme is called "active low", and it's
  // very commonly used in electronics... so much that the chip
  // has built-in pullup resistors!
  pinMode(0, INPUT_PULLUP);
  pinMode(1, INPUT_PULLUP);
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);  // Teensy++ LED, may need 1k resistor
pullup
  pinMode(7, INPUT_PULLUP);
  pinMode(8, INPUT_PULLUP);
  pinMode(9, INPUT_PULLUP);
  pinMode(10, INPUT_PULLUP);
  pinMode(11, INPUT_PULLUP);
  pinMode(12, INPUT_PULLUP);
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
}

void loop() {
  // Update all the buttons.  There should not be any long
  // delays in loop(), so this runs repetitively at a rate
  // faster than the buttons could be pressed and released.
  button0.update();
  button1.update();
  button2.update();
  button3.update();
  button4.update();
  button5.update();
  button6.update();
  button7.update();
  button8.update();
  button9.update();
  button10.update();
  button11.update();
  button12.update();
  button13.update();
  button14.update();

  // Check each button for "falling" edge.
  // Type a message on the Keyboard when each button presses
  // Update the Joystick buttons only upon changes.
  // falling = high (not pressed - voltage from pullup resistor)
  //           to low (pressed - button connects pin to ground)
  if (button0.fallingEdge()) {
    Keyboard.print("0");      // Enter a rest of the current note
length.
  }
  if (button1.fallingEdge()) {
    Keyboard.print(".");      // Toggle augmentation dot.
  }
  if (button2.fallingEdge()) {
    Keyboard.print("2");      // Demi-semi quaver (32nd note).
  }
  if (button3.fallingEdge()) {
    Keyboard.print("3");      // Semi quaver (16th note).
  }
  if (button4.fallingEdge()) {
    Keyboard.print("4");      // Quaver (8th note).
  }
  if (button5.fallingEdge()) {
    Keyboard.print("5");      // Crotchet (quarter note).
  }
  if (button6.fallingEdge()) {
    Keyboard.print("6");      // Minim (half note).
  }
  if (button7.fallingEdge()) {
    Keyboard.print("7");      // Semibreve (note).
  }
  if (button8.fallingEdge()) {
    Keyboard.print("n");      // Toggle note entry mode.
  }
  if (button9.fallingEdge()) {
    ctrlAltPlus(KEY_1);         // Voice 1.
  }
  if (button10.fallingEdge()) {
    ctrlAltPlus(KEY_2);         // Voice 2.
  }
  if (button11.fallingEdge()) {
    ctrlAltPlus(KEY_3);         // Voice 3.
  }
  if (button12.fallingEdge()) {
    ctrlAltPlus(KEY_4);         // Voice 4.
  }
  if (button13.fallingEdge()) {
    pressKey(KEY_LEFT);         // Move left.
  }
  if (button14.fallingEdge()) {
    pressKey(KEY_RIGHT);        // Move right.
  }
}

void ctrlAltPlus(int keycode){
  // Press the control, alt then the character.
  Keyboard.set_modifier(MODIFIERKEY_CTRL | MODIFIERKEY_ALT);
  Keyboard.send_now();

  // press DELETE, while CLTR and ALT still held
  Keyboard.set_key1(keycode);
  Keyboard.send_now();

  // release all the keys at the same instant
  Keyboard.set_modifier(0);
  Keyboard.set_key1(0);
  Keyboard.send_now();
}

void pressKey(int keycode){
  Keyboard.set_key1(keycode);
  Keyboard.send_now();
  Keyboard.set_key1(0);
  Keyboard.send_now();
}
```

Save the project, then click on the Upload icon. You may need to press the button on the Teensy – the Arduino software will tell you to do this if it is necessary.

You can test the software by starting a text editor or word processor. Make sure it is the active program, then press the keys on the MuseScore keypad. When you are happy that some of the keys are working, start MuseScore and test that all the keys trigger the correct functions.

When it is all working, enjoy!