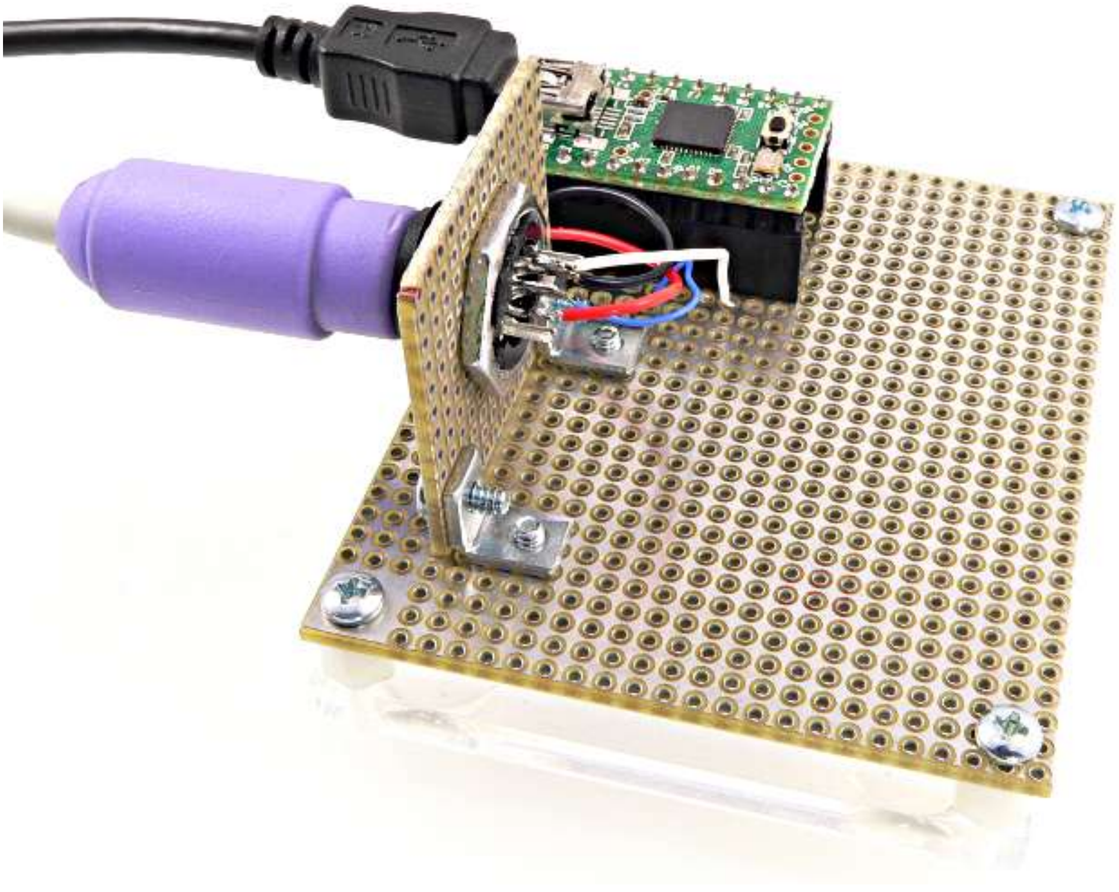
# PS2Keyboard Library

PS2Keyboard allows you to use a keyboard for user input.

**Download**: Included with the Teensyduino Installer
Latest Developments on Github

# Hardware Requirements

PS2 Keyboards are easy to interface, requiring only 5 volt power and 2 signals.



The keyboard's clock signal must connect to an interrupt pin. The data signal may connect to any pin, but do not use the pin with an LED connected to ground because the LED current will interfere with the data signal.

| Board | Data Pin | Clock Pin (IRQ) |
|---|---|---|
| Teensy 3.6<br>Teensy 3.5<br>Teensy 3.2<br>Teensy 3.1<br>Teensy 3.0 | Any except 13 & analog-only pins | Any except 13 & analog-only pins |

- Ethernet
- Bounce
- Keypad
- ⊞ **Audio**
- Encoder
- Ping
- CapacitiveSensor
- FreqCount
- FreqMeasure
- Servo
- PulsePosition
- Stepper
- AccelStepper
- FrequencyTimer2
- Tlc5940
- SoftPWM
- ShiftPWM
- Time
- TimeAlarms
- DS1307RTC
- Metro
- TimerOne
- MsTimer2
- EEPROM

⊞ **Reference**

| | | |
|---|---|---|
| Teensy LC | Any except 13 | 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 20, 21, 22, 23 |
| Teensy 2.0 | Any, except 11 | 5, 6, 7, 8 |
| Teensy 1.0 | Any | 0, 1, 2, 3, 4, 6, 7, 16 |
| Teensy++ 2.0 | Any, except 6 | 0, 1, 2, 3, 18, 19, 36, 37 |
| Teensy++ 1.0 | Any | 0, 1, 2, 3, 18, 19, 36, 37 |

**Caution**: PS2 Keyboards contain pullup resistors to 5 volts which can cause damage to 3.3V only boards like Teensy LC and Teensy 3.6. Use of a level shifter circuit is needed to safely use boards which are not 5 volt tolerant.

## Basic Usage

PS2Keyboard keyboard;

Create the keyboard object. Even though you could create multiple objects, only a single PS2 keyboard is supported by this library.

keyboard.begin(DataPin, IRQpin)

Begin the receiving keystrokes. DataPin and IRQpin are the pin numbers where you connected the PS2 keyboard's Data and Clock signals.

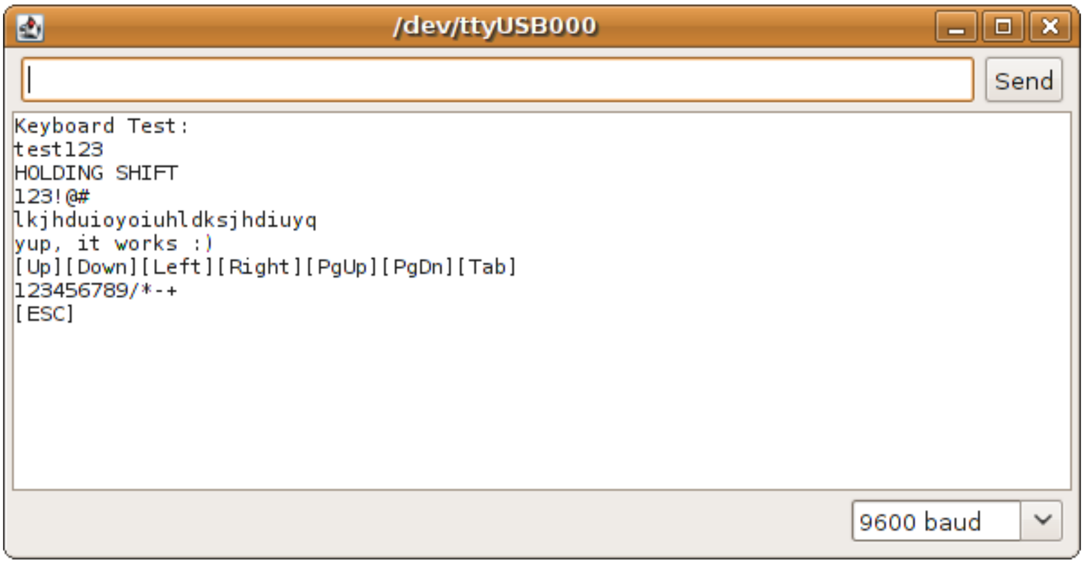
keyboard.available()

Check if a keystroke has been received. Returns true if at least one keystroke.

keyboard.read()

Read the next keystroke. -1 is returned if no keystrokes have been received. Keystrokes are returned as ASCII characters. Special keys are mapped to control characters.

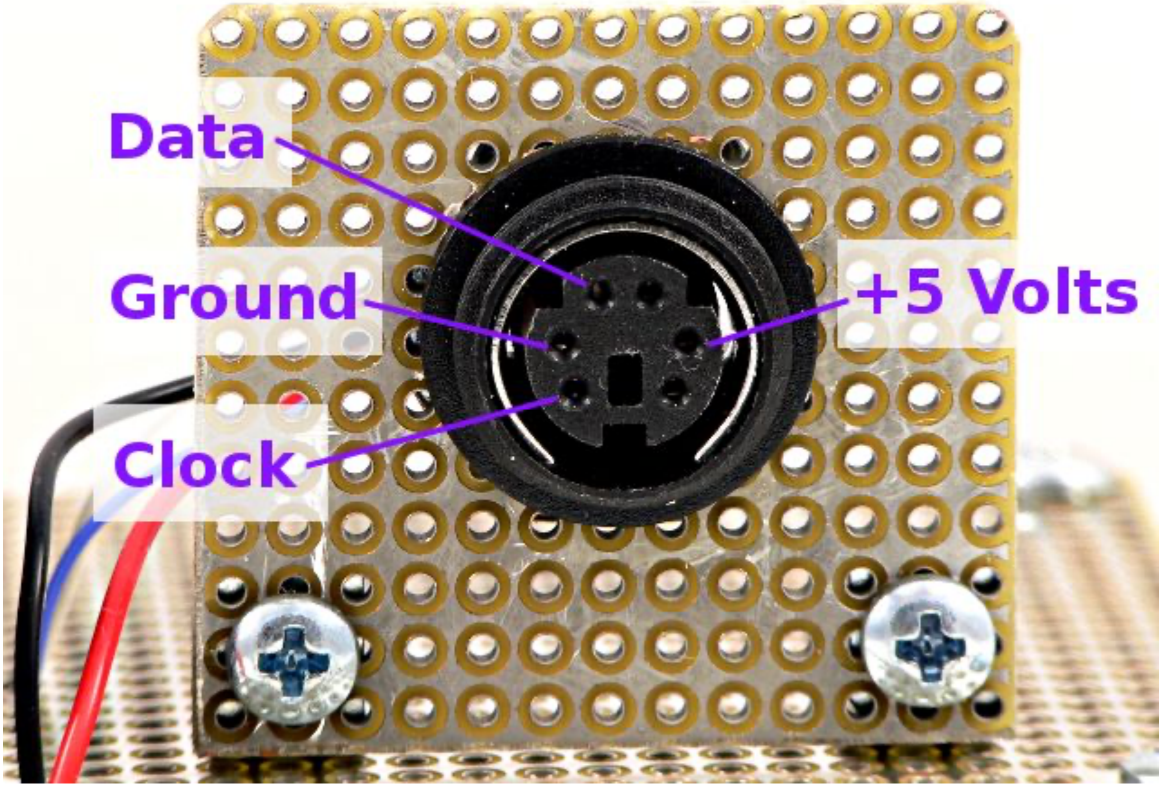
## Example Program



```
Keyboard Test:
test123
HOLDING SHIFT
123!@#
lkjhduioyoiuhldksjhdiuyq
yup, it works :)
[Up][Down][Left][Right][PgUp][PgDn][Tab]
123456789/*-+
[ESC]
```

```
#include <PS2Keyboard.h>

const int DataPin = 8;
const int IRQpin =  5;

PS2Keyboard keyboard;

void setup() {
  delay(1000);
  keyboard.begin(DataPin, IRQpin);
  Serial.begin(9600);
  Serial.println("Keyboard Test:");
}

void loop() {
  if (keyboard.available()) {

    // read the next key
    char c = keyboard.read();

    // check for some of the special keys
    if (c == PS2_ENTER) {
      Serial.println();
    } else if (c == PS2_TAB) {
      Serial.print("[Tab]");
    } else if (c == PS2_ESC) {
      Serial.print("[ESC]");
    } else if (c == PS2_PAGEDOWN) {
      Serial.print("[PgDn]");
    } else if (c == PS2_PAGEUP) {
      Serial.print("[PgUp]");
    } else if (c == PS2_LEFTARROW) {
      Serial.print("[Left]");
    } else if (c == PS2_RIGHTARROW) {
      Serial.print("[Right]");
    } else if (c == PS2_UPARROW) {
      Serial.print("[Up]");
    } else if (c == PS2_DOWNARROW) {
      Serial.print("[Down]");
    } else if (c == PS2_DELETE) {
      Serial.print("[Del]");
    } else {

      // otherwise, just print all normal characters
      Serial.print(c);
    }
  }
}
```

# PS2 Connector Pinout



## Non-US Keyboard Layout

The read() function returns ASCII codes, which are translated from the keyboard's raw scan codes. This translation is done using two arrays inside PS2Keyboard.cpp, called scan2ascii_noshift and scan2ascii_shift.

For non-US keyboard layouts, you may need to edit these arrays for proper output.

## Special Keys

PS2 keyboards have many special keys which do not map to ASCII characters. By default, several are mapped to ASCII control characters with similar intended function. Others are disabled, but may be mapped to any byte by editing PS2Keyboard.h.

```
// Every call to read() returns a single byte for each
// keystroke.  These configure what byte will be returned
// for each "special" key.  To ignore a key, use zero.
#define PS2_TAB         9
#define PS2_ENTER       13
#define PS2_BACKSPACE   127
#define PS2_ESC         27
#define PS2_INSERT      0
#define PS2_DELETE      127
#define PS2_HOME        0
#define PS2_END         0
#define PS2_PAGEUP      25
#define PS2_PAGEDOWN    26
#define PS2_UPARROW     11
#define PS2_LEFTARROW   8
#define PS2_DOWNARROW   10
#define PS2_RIGHTARROW  21
#define PS2_F1          0
#define PS2_F2          0
#define PS2_F3          0
#define PS2_F4          0
```

```
#define PS2_F5          0
#define PS2_F6          0
#define PS2_F7          0
#define PS2_F8          0
#define PS2_F9          0
#define PS2_F10         0
#define PS2_F11         0
#define PS2_F12         0
#define PS2_SCROLL      0
```

# Version 2.4

Support Teensy 3.0, Arduino Due, Arduino Leonardo & other boards
French keyboard layout, thanks to David Chochoi

# Version 2.3

Minor bugs fixed.

# Version 2.2

Add support for non-US keyboards. German is the only support (yet), thanks to Rainer Bruch who send a German keyboard for testing! See the example in File > Exampels > PS2Keyboard > International.

# Version 2.1

Version 2.1 adds the following improvements:

- **Resync Timeout**: If the interrupt gets out of sync, after a 1/4 second timeout it will automatically resync. This greatly improves the reliability under less than perfect conditions.
- **Arduino "new-extension" Compatibility**: The latest version of Arduino in development (eventually to become version 1.0) has some incompatible changes. PS2Keyboard now works with it.

# Version 2.0

Version 2.0 adds the following improvements:

- **Scan Code Buffering**: Data captured by the interrupt is stored into a circular buffer, so several keystrokes can be buffered if your sketch is busy and can not call read() between each one. Buffering also retains key break (up) events. Previous versions buffered only a single scan code and discarded all key break codes.
- **Shift Key State**: The state of both shift keys is tracked, so conversion to ASCII produces proper upper and lower case characters. The numbers and other dual character keys also properly respond to shift. Previous versions were limited to only uppercase and non-shift output.
- **ASCII Only Output**: The read() function returns only ASCII characters, or special keys mapped to ASCII control characters. Previous versions returned a mix of ASCII and raw scan codes, with no way to distinguish between them.
- **Efficient Conversion**: Fast table lookup has replaced a slow linear search algorithm.
- **Multi-Board Support**: Arduino Mega, Teensy and Sanguino interrupt pins are supported. Other boards can be easily added, or will work automatically if their core files define certain aliases.

# Details

Other useful pages:

- Original (old) PS2Keyboard library page
- Scan Codes Explained (wikipedia)
- Scan Codes, Set 2 Table
- PS2 Keyboard Protocol (electrical details)
- http://www.arduino.cc/playground/Main/PS2KeyboardExt2
- http://www.practicalarduino.com/projects/ps2-keyboard-or-mouse