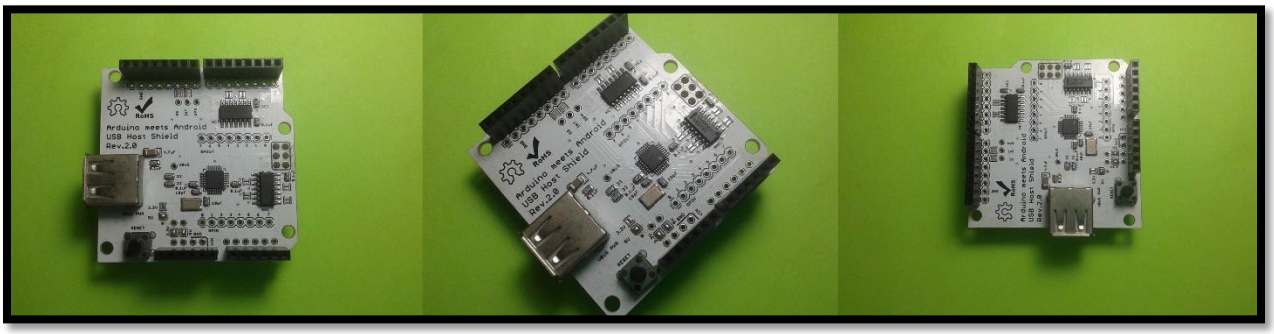
USB HOST SHIELD



Product Description:

This is revision 2.0 of USB Host Shield. Thanks to new interface layout it is now compatible with more Arduinos - not only UNO and Duemilanove, but also big Mega and Mega 2560 work with Standard variant of this shield out of the box. No more SPI re-wiring and code modifications - just solder included stackable connectors (2x3 ICSP connector's female side should be facing down), plug and play!

Product Specification:

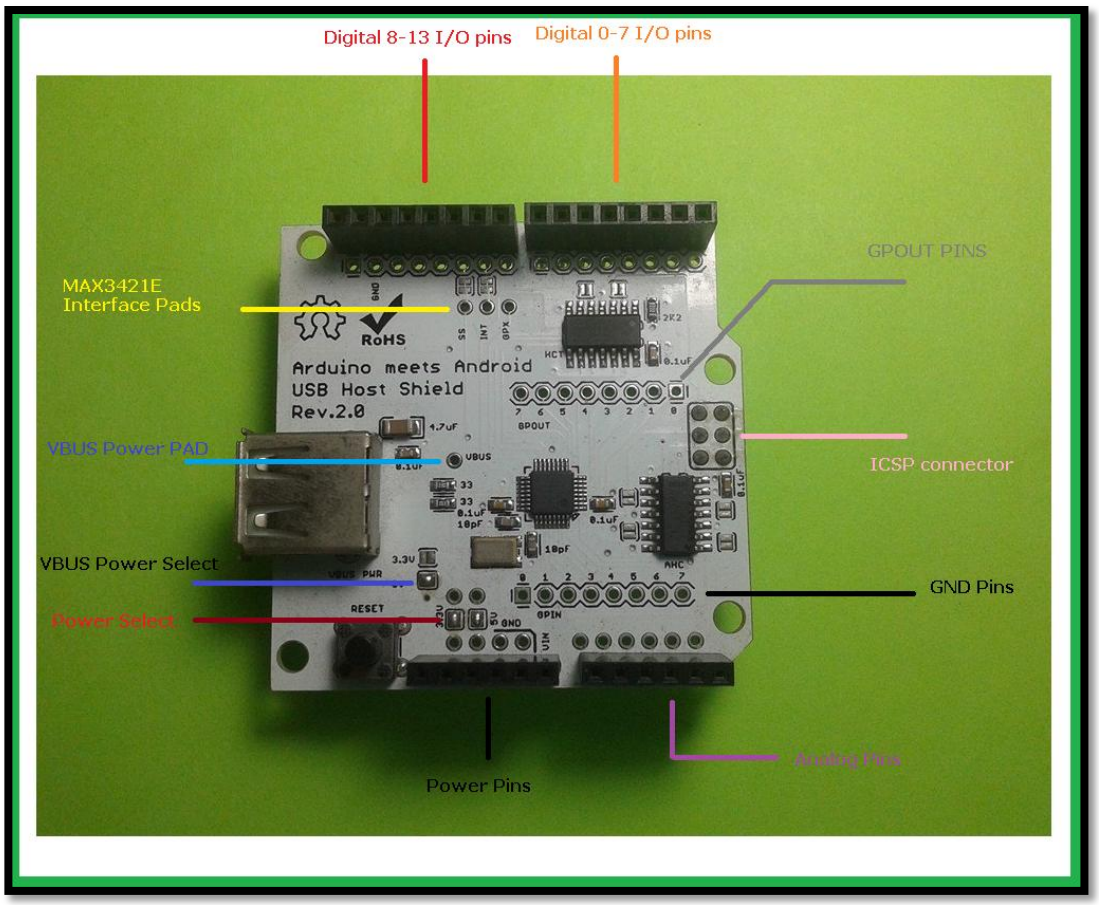
- This shield will work with standard (dual 5/3.3V) and 3.3V-only (for example, Arduino Pro) boards.
- Arduino clones with standard connector layout, including ICSP connector, should work, however only BlackWidow has been tested so far.
- If your Arduino clone doesn't follow classic layout, take a look at previous revision of the shield.

The following device classes are currently supported by the shield:

- HID devices, such as keyboards, mice, joysticks, etc.
- game controllers - Sony PS3, Nintendo Wii, Xbox360
- USB to serial converters - FTDI, PL-2303, ACM, as well as certain cell phones and GPS receivers
- ADK-capable Android phones and tables
- Digital cameras - Canon EOS, Powershot, Nikon DSLRs and P&S, as well as generic PTP
- Mass storage devices, such as USB sticks, memory card readers, external hard drives
- Bluetooth dongles

Pin Configuration and System Diagram:

USB Host shields are available in two form factors – full size and Mini. Full size shield is designed to fit on top of "Standard" Arduinos, such as Uno, Duemilanove, Mega 1280/2560, and compatible clones. Full size shield has been designed for ease of use; it has plenty of empty space, features extra pads, solder jumpers and extensive silkscreen markings, simplifying board modification and troubleshooting. Full size shield is recommended for basic prototyping and simple projects. Mini shield main advantages are low size, weight and cost. Ideally, it should be used together with Arduino Pro Mini 3.3V board. It can be mated with other Arduino and non-Arduino MCU boards, but it takes more work. Small size, dense part placement and lack of silkscreen markings make this board more suitable for advanced projects, as well as semi-permanent and permanent installations, when basic functionality and wiring is already confirmed on larger prototype. Generally, modification and troubleshooting of Mini shield board is more difficult. **Full-size shield**USB Host Shield 2.0 exists in 2 configurations – "Standard" and "3.3V". The layout of Standard board is depicted on the right. The board contains Maxim MAX3421E USB host controller, 12MHz crystal, level shifters, resistors, capacitors, Reset button and USB A-type connector. There are also a number of solder pads and jumpers, which are marked with red arrows.



1. **Power Select** 2 solder jumpers marked "5V" and "3.3V". They are used for different power configurations. The configuration shown, when both jumpers are closed, is suitable for official Arduinos, such as UNO, Duemilanove, Mega and Mega 2560. See Power Options section for detailed explanation.
2. **Power pins** are used to connect to power pins of Arduino board. RESET, 3.3V, 5V and GROUND signals from this connector are used.
3. **Analog pins** are not used by the shield. They are provided to simplify mounting and provide pass-through for shields mounted atop of USB Host Shield in a stack.
4. **GPIN pins.** Eight 3.3V general-purpose digital input pins of MAX3421E. They are used primarily to interface with buttons, rotary encoders and such. GPIN pins can also be programmed as a source of MAX3421E interrupt. An example of GPIN use can be seen in digital camera controller project.
5. **ICSP connector** is used by the shield to send/receive data using SPI interface. SCK, MOSI, MISO and RESET signals from this connector are used.
6. **GPOUT pins** are eight 3.3V general-purpose digital output pins of MAX3421E. They can be used for many purposes; I use it to drive HD44780-compatible character LCD, as can be seen in digital camera controller circuit, as well as this keyboard example. Max_LCD library which is part of standard USB Host library software package uses some of GPOUT pins.
7. **Digital I/O pins 0-7**, like already mentioned analog pins are not used by the shield and provided only for convenience.
8. **Digital I/O pins 8-13**. In this group, the shield in its default configuration uses pins 9 and 10 for INT and SS interface signals. However, standard-sized Arduino boards, such as Duemilanove and UNO have SPI signals routed to pins 11-13 in addition to ICSP connector, therefore shields using pins 11-13 combined with standard-sized Arduinos will interfere with SPI. INT and SS signals can be re-assigned to other pins (see below); SPI signals can not.
9. **MAX3421E interface pads**are used to make shield modifications easier. Pads for SS and INT signals are routed to Arduino pins 10 and 9 via solder jumpers. In case pin is taken by other shield an re-routing is necessary, a trace is cut and corresponding pad is connected with another suitable Arduino I/O ping with a wire. To undo the operation, a wire is removed and jumper is closed. See interface modifications section for more information.GPX pin is not used and is available on a separate pad to facilitate further expansion. It can be used as a second interrupt pin of MAX3421E.
10. **VBUS power pad.** This pad is used in advanced power configurations, described in Power Options section.

How to test:

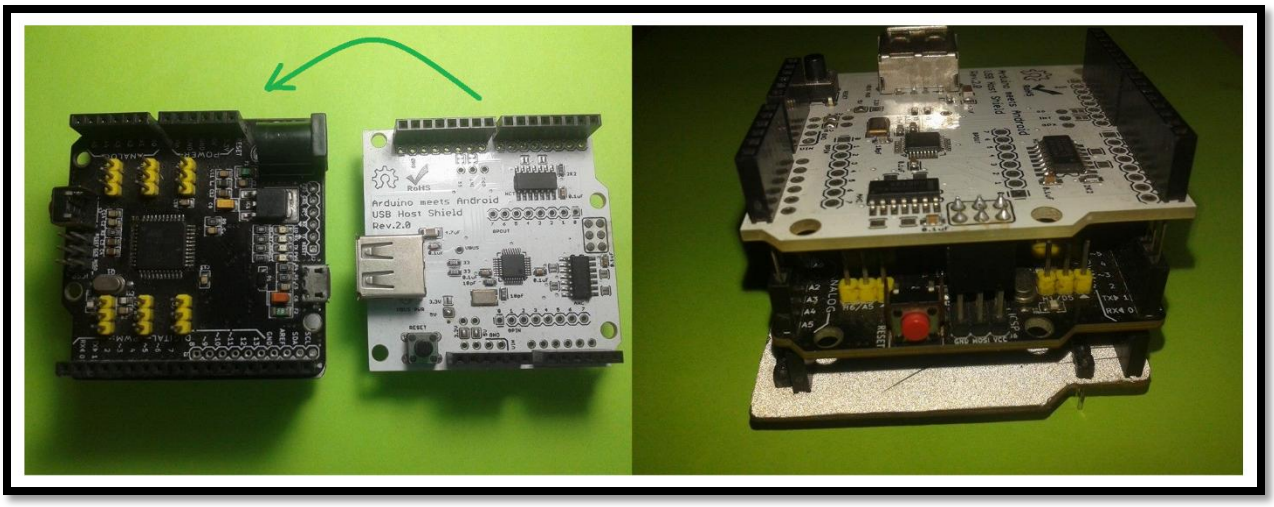
Here's a library for testing the USB host shield:

https://github.com/felis/USB_Host_Shield_2.0

Just download all the files in this link, Extract the Zip file. Rename the folder to USB_Host_Shield_20. Then put it inside the Arduino IDE library folder C:\Users\LeandroPC\Documents\arduino-1.0.3\libraries.

In this Demo we will use the USBHIDBootMouse example and demonstrate how you can get and read data from a USB mouse through the Arduino.

Since we are using a Shield, wiring will not be a problem, Since the USB HOST SHIELD fits directly on top of an Arduino.



Now open the mouse example,

C:\Users\LeandroPC\Documents\arduino-1.0.3\libraries\USB_Host_Shield_20\examples\HID\USBHIDBootMouse

```
#include <hidboot.h>
#include <usbhub.h>
// Satisfy IDE, which only needs to see the include statment in the ino.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif

class MouseRptParser : public MouseReportParser
{
protected:
        virtual void OnMouseMove        (MOUSEINFO *mi);
        virtual void OnLeftButtonUp     (MOUSEINFO *mi);
        virtual void OnLeftButtonDown   (MOUSEINFO *mi);
        virtual void OnRightButtonUp    (MOUSEINFO *mi);
        virtual void OnRightButtonDown  (MOUSEINFO *mi);
        virtual void OnMiddleButtonUp   (MOUSEINFO *mi);
        virtual void OnMiddleButtonDown (MOUSEINFO *mi);
};
void MouseRptParser::OnMouseMove(MOUSEINFO *mi)
{
    Serial.print("dx=");
    Serial.print(mi->dX, DEC);
    Serial.print(" dy=");
    Serial.println(mi->dY, DEC);
};
```

```
void MouseRptParser::OnMouseMove(MOUSEINFO *mi)
{
    Serial.print("dx=");
    Serial.print(mi->dX, DEC);
    Serial.print(" dy=");
    Serial.println(mi->dY, DEC);
};
void MouseRptParser::OnLeftButtonUp     (MOUSEINFO *mi)
{
    Serial.println("L Butt Up");
};
void MouseRptParser::OnLeftButtonDown   (MOUSEINFO *mi)
{
    Serial.println("L Butt Dn");
};
void MouseRptParser::OnRightButtonUp    (MOUSEINFO *mi)
{
    Serial.println("R Butt Up");
};
void MouseRptParser::OnRightButtonDown  (MOUSEINFO *mi)
{
    Serial.println("R Butt Dn");
};
void MouseRptParser::OnMiddleButtonUp   (MOUSEINFO *mi)
{
    Serial.println("M Butt Up");
};
void MouseRptParser::OnMiddleButtonDown (MOUSEINFO *mi)
{
    Serial.println("M Butt Dn");
};
```

```
USB     Usb;
USBHub     Hub(&Usb);
HIDBoot<HID_PROTOCOL_MOUSE>    HidMouse(&Usb);

uint32_t next_time;

MouseRptParser                                  Prs;

void setup()
{
    Serial.begin( 115200 );
    while (!Serial); // Wait for serial port to connect - used on Leonardo,
    Serial.println("Start");

    if (Usb.Init() == -1)
        Serial.println("OSC did not start.");

    delay( 200 );

    next_time = millis() + 5000;

    HidMouse.SetReportParser(0,(HIDReportParser*)&Prs);
}

void loop()
{
  Usb.Task();
}
```

Upload the sketch, plug in the USB mouse to the USB Host shield, open the Arduino Serial Monitor , and you should see the data coming from the mouse.