Arduino leonardo manual pdf

Contributors: Jimblom Favorited Favorite 10 Heads up! This is about Pro Micro (ATmega32U4) 5V and 3.3V versions. If you are looking for information about the hardware Qwiic Pro Micro with USB-C connector, be sure to check out the older Qwiic Pro Micro USB-C (ATmega32U4) Hookup Guide. Welcome to the new border arduino compatible boards, which is possible with ATmega32U4. No longer does your Arduino be used with an FTDI cable, or ATmega8U2, or a chip that is the sole purpose of acting as a mediator between your Arduino and your computer. Pro Micro SparkFun Pro Micro [3.3V/8MHz and 5V/16MHz] is a really cool, little development board. It's an Arduino-compatible microcontroller, micro-sized, and it accomplishes with a single chip in the old Arduino Unos, Duemilanoves, and Diecimeillas could never dream of: true USB functionality. DEV-12587 Here at SparkFun, we refuse to leave good enough alone. That's why we're adding to our line-up with Arduino-compatible microc... 17 Favorited Favorite 33 DEV-12640 Here at SparkFun, we refuse to leave good enough alone. That's why we're adding to our line-up with Arduino-compatible microc... 78 Favorited Favorite FioV3 FioV3 This tutorial also applies to Fio v3, which works much like Pro Micro, but adds features like easy XBee linking and LiPo charging. DEV-11520 Fio v3 is a new spin on the Arduino Fio hardware powered by ATmega32U4.Ne only it's small and LiPo-ready, it've... 7 Favorited Favorite 13 On this tutorial This tutorial aims to introduce you to both the hardware and firmware sides of Pro Micro (and Fio v3). We also dedicate a few pages to help install boards on Windows and Mac. Here is a summary of what will be covered: Recommended reading before delving into this tutorial, here are some concepts you should familiarize with. If you're not, consider checking out the related tutorial first. Asynchronous series of communication concepts: packets, signal levels, bodu speeds, UARTs and more! Favorited Favorite 85 What is this Arduino thing anyway? This tutorials dive into what Arduino is and along with Arduino projects and widgets. Favorited Favorite 38 step by step to install and test Arduino software for Windows, Mac and Linux. Favorited Favorite 14 Before we get into installing and using Pro Micro, let's take a quick look at the board – check its raw materials, output, and other hardware quirks. Pinout All Pro Micro I/O and power pins are broken up to two, parallel headers. Some spikes have power inputs or output, other spikes are dedicated to I/O spikes. In addition, I/O pins may have special abilities, such as analog input. Here is a map of which pin is, where, and what specific hardware features it can have: Delving a little further, in which pins do what... Power Pins There Are Different Power and Energy Related Dissolves: RAW is an unregulated voltage input for Pro Micro. If the board is powered via USB, the voltage in this pin will be about 4.8V (USB's 5V minus schottkey diodes drop). On the other hand, if the board is powered externally using this pin, the suitable voltage can be up to 12V. VCC is the voltage supplied on board the ATmega32U4. This voltage will depend on whether you are using the 3.3V/8MHz Pro Micro or 5V/16MHz version, it will be either 3.3V or 5V respectively. This voltage is regulated by the voltage applied to the raw pin. If the board is powered by a RAW pin (or USB), this pin can be used as an output for feeding other devices. You can use RST to restart Pro Micro. This pin is pulled high 10k &amp;amp; Om; resistor on the board and is active low, so it is connected to the ground to start the reset. Pro Micro will remain turned off until the reset line is pulled back to high again. GND, of course, is the total ground voltage (0V reference) for the system. I/O Pins Pro Micro's I/O pins – 18 all – are multi-talented. Each pin can be used as a digital input or output to flash light houses or reading buttons. These spikes are referenced by the Arduino IDE using an entire value between 0 and 21. (A0-A3 pins can be referenced digitally using their analog or digital pin number). The nine pins feature an analog digital converter (ADCs) and can be used as analog raw materials. They are useful for reading potentiometers or other analog devices using the analogRead([pin]) function. There are five pins with pulse width modulation (PWM) functionality that allow you to create an analog output shape using the analogwrite ([pin], [value]) function. These pins are indicated on the board with a weak, white circle around them. There are hardware UART (serial), I2C, and SPI pins available as well. They can be used to interact with digital devices such as serial LCD, XBees, IMU and other series sensors. Pro Micro has five external breaks that allow you to immediately start the function when the pin goes either high or low (or both). If you stop with a break-enabled contact, you need to know the specific break activated by the contact: pin 3 cards to stop 0 (INT0), pin 2 is interrupted 1 (INT1), pin 0 is 2 . (INT2), 1 stops 3 (INT3) and 7 pin is 4 termination (INT6). On-Board LED There are three LEDs for Pro Micro. One red indicator indicates whether there is a power supply. The other two light mines help you specify when data is transferred via USB. A yellow indicator represents USB data that enters (RX) Pro Micro, and a green indicator indicates that the USB data will end (TX). 3.3V or 5V? 8MHz or 16MHz? Pro Micros comes in two flavors that vary depending on the system voltage and frequency of operation. The standard 5V Pro Micro works at 16MHz and is very comparable to the Arduino Leonardo, while the 3.3V version of Pro Micro works speed (must remain in the safe operating area at the lowest voltage) -- 8MHz. For example, if you have 3.3V Pro Micro, there is no interface to it with anything that outputs 5V. Don't forget which version you have! We will need to distinguish two when we upload the code Arduino. If you are not sure which version you have, check the back corner of the board. Check one of two boxes to indicate the operating voltage. Board Dimensions Pro Micro is 1.30 x0.70. Note that the USB connector adds a small bit of length to the board so the total length is about 1.37. As Power Pro Micro's main function is its innate USB functionality, the most common way to power it is by using USB. In this setup, the 5V Pro Micro will be powered directly from the USB bus and the 3.3V Pro Micro will regulate the 5V supply coming from the USB down. You can connect a second end of a USB cable to your computer, USB hub, or USB wall adapter, which can (in most cases) provide more power. Alternatively, if Pro Micro lives in the wild, out of reach of USB cables, it can be operated by either RAW or VCC pins. Raw pin at the correct working voltage (5V or 3,3V). To be safe, it must not be greater than 12V and should be at least 1V more than pro micro operating voltage (e.g. &gt;6V per 5V Pro Micro). Pro Micro powered by RAW tap, with a four-series AA battery pack. If you power Pro Micro through a VCC pin, keep in mind that this signal is unregulated. Use this option only if you have a clean, regulated 3.3V or 5V power to connect to it. How, exactly, you power your project depends on you and your project requirements. If you make something battery powered, you can choose a 3.3V Pro Micro that could be powered by a LiPo battery or a couple of alkaline. Be sure to check out the following tutorials for more information. Behind the battery used portable electronic device basics: LiPo, NiMH, coin cells, and alkali. Favorited Favorite 42 Tutorial to help clarify the energy requirements of your project. Favorited Favorite 58 On this page we check the hardware side of fio v3, looking at the pinout, layout and schematic board. Fio v3 is like an elongated Pro Micro. On the one hand, it's shape and pinouts are similar to this ATmega32U4 brother. The other end of Fio v3 is what makes it unique: the footprint of the XBee on the bottom, and the LiPo charging chain on top. Pinout All Fio v3's pins are broken on both sides of the board. Some spikes have power inputs or output, other spikes are dedicated to I/O spikes. In addition, I/O spikes may have special capabilities such as analog or series input/output. Here is a map of which pin is, where, and what special options it can be: Heads up! The pins indicated in the figure are correct. However, the labels printed with the mesa stamp on the physical board are incorrect. D17 with name on board is Arduino pin 14/MISO. D14 on the board is Arduino pin 17/SS/ RXLED. The pins called 3.3V are distinguished by the ATmega32U4 operating voltage source. While the board is powered through a white JST connector or USB, this voltage is adjusted to 3.3V. These pins can be used as outputs to supply 3.3V to other devices. You can use the RST pin to restart the Fio. This pin is pulled high with a 10kΩ resistor on board and is active-low, so it is connected to the ground to start the reset. The Fio will remain off until the reset line is pulled back to high again. I/O Pins Many fio's I/O pins are multi-talented. Each pin can be used as a digital input or output to flash light houses or reading buttons. These spikes are referenced by the Arduino IDE using an entire value between 0 and 23. (A0-A10 pins can be referenced digitally using their analog pin number or digital contact number). Eleven pins feature an analog digital converter (ADCs) and can be used as analogue raw materials. They are useful for reading potentiometers or other analog devices using the analogRead([pin]) function. There are six pins with pulse width modulation (PWM) functionality that allow you to form an analog output shape using the analogWrite ([pin], [value]) function. These pins are indicated on the board with a weak white circle around the pin. Also available are hardware UART (serial), I2C and SPI pins available. They can be used to interact with digital devices such as serial LCD, IMU and other series sensors. Fio v3 has five external breaks that allow you to instantly trigger the function when the pin goes either high or low (or both). If you stop with a break-enabled contact, you need to know the specific break activated by the contact: pin 3 cards to stop 0 (INT0), pin 2 is interrupted 1 (INT1), pin 0 is 2 . (INT2), 1 stops 3 (INT3) and 7 pin is 4 termination (INT6). Board LED There are various LEDs for the Fio, the simplest of which is a red power indicator. Two LEDs down – titled RX and TX – help indicate when data is transferred to and from Fio via USB. A blue LED displays USB data coming in (RX) pro micro, and a yellow indicator indicates that the USB data will go out (TX). There are three LEDs connected to the XBee interface, in particular: stat, RSSI, and associate. The red LED named ON is connected to the XBee tap 13 - DIO9 - which is set by default to indicate the ON/OFF status of the XBee module. RSSI LED connects to XBee pin 6 (PWM0), which by default indicates rssi (received signal strength) - LED is a stronger signal received. Finally, the ASO LED connects to the XBee pin 15, which flashes when the module is connected. Finally, there's a yellow LED called CHG that indicates whether the attached lithium polymer battery charges. If the battery is not connected to fio, the LED will be in an undefined position and will most likely be illuminated. As Power Fio v3 The recommended power supply for Fio v3 is any single cell lithium polymer (LiPo) battery. The nominal voltage of these batteries is 3.7 V, ideal for 3.3 V fio power. LiPos is awesome because they're rechargeable and still pack a lot of power into a tiny room. Any of our single-cell LiPos with JST terminators can directly connect to the Fio onboard JST connector. As a (stationary) alternative to batteries, Fio can be operated directly through a USB connector. Using the LiPo Charger Fio v3 is a LiPo charging management chain (built around the MCP73831) built on what handles the signal conditioning required to safely charge a single-celled LiPo battery. To use the charging chain, you obviously need a single cell LiPo battery attached to the Fio. Then connect the board via USB, so the charging chain has the primary voltage source to charge the battery. The CHG LED indicates the charging status of the battery. If it's turned on, the battery still is charged. When the CHG LED goes out, the battery is fully charged. The charging circuit is programmed to charge the battery 500mA, so to be safe, the battery must not be less than 500mAH capacity. Connecting XBee XBee-footprint connectors at the bottom of fio v3 is what makes it so unique. This product is designed to provide a simple interface between Arduino and XBee, such as some of the XBee pins coming wired up to ATmega32U4. Most importantly, the series interfaces of both devices are wired – XBee's DOUT pin is connected to ATmega32U4's RX, and the DIN is connected to the TX. XBee's are controlled and configured using the serial interface. To learn more about XBee's check out their datasheet and various tutorials to help get started with these awesomely simple wireless transmissions of the day. Getting Pro Micro or Fio v3 set up on your computer and in your Arduino environment can be difficult. Follow this page step by step using the driver setup and Arduino-enabling process. Step 1 for installing a Windows driver: Download the driver Before you add a board, start with downloading the drivers. Check GitHub Repository for the latest files. The same driver file works for both Pro Micro and Fio v3. Both Fio and Pro Micro drivers are signed for Windows users. You can download them directly via the link below. Unzip that zip file, and don't forget where you are its contents. In this zip file, you should find .inf and .cat files that contain all the information Windows needs to install the Pro Micro driver. Sparkfun.inf manager and sparkfun.cat digitally signed catalog file will be found ... Arduino_Boards-master/sparkfun/avr/signed_driver . Step 2: Plug in Pro Micro/Fio v3 When you originally plug in the board, the Installation device driver software bubble notification should pop up in the lower right corner of the taskbar. After green dot circles in the gray box several times, you'll probably get a sad bubble like this: Never fear! Windows just doesn't know where to find our drivers. Note: Some users are having problems connecting pro micro to a USB 3.0 port. If you're having problems on usb 3.0 ports, try switching to use a USB 2.0 port. From here, the easiest way to install a driver is device manager. To open Device Manager, click the Start button, and then open Control Panel. In Control Panel, click System and Maintenance, and then open Device Manager. Or, you can open the Run Prompt (Windows key +R) and type devmgmt.msc and click OK. In Device Manager, expand the Other Devices tree, where you need to find a USB IO board with a yellow warning sign above its icon. Right-click on the USB IO Board and select Update Driver Software.... This should spawn Update Driver Software – USB IO Board window. Step 4: When searching for driver in the first window that comes up, click Browse my computer driver software. In the next window, click Browse.... to search for the driver you just downloaded. It should be a folder called Arduino_Boards-master, marked in step 1 in the subdirectory. After you select the Driver folder, click OK, and then select Next. In the warning dialog, it is safe to select the Install this driver software option. After you have viewed the progress bar beam several times, you should get happy windows has successfully updated the driver software window. And Device Manager should have a new entry in SparkFun Pro Micro (COM##) (or SparkFun Fio V3 (COM##) if you have one of them) under the Port tree. Take note of which COM port your Pro Micro was assigned to. We will need it soon. Installing the Arduino Addon We are still not fully ready for the Arduino, but this is the last stretch. Before you can use ProMicro with Arduino IDE, you will need to install the ship's (.brd) files in Fio/Pro Micro so the Arduino IDE will know how to communicate with your board. Using the board leader with the release of Arduino 1.6.4, adding third-party boards to the Arduino IDE is easy to achieve with the board leader. If you are using an earlier version of Arduino (1.6.3 or we recommend upgrading now. As always, you can download the latest version of Arduino from arduino.cc. To get started, you will need to specify an Arduino IDE board manager custom URL. Open Arduino, then go to Preferences (File &gt; Preferences). Then, at the bottom of the window, paste the following URL into the Advanced Board Manager URLs text box: You can add multiple URLs by clicking the window icon and pasting into one LINE of THE URL. Click OK. Then, open Board Manager by clicking Tools, and then point to the Board Selection tab, and then click Panel Manager. Look for sparkfun in the board manager. You should see the SparkFun AVR Boards package appear. Click Install, wait a moment, and you must install any .brd files that you want to install by specifying the Installed blue that is printed next to the package. Now you should be able to upload the code to several Products compatible with SparkFun Arduino, including Fio and Pro Micro. Manually install .brd files If you are using an earlier version of the Arduino IDE and do not have access to the Board Manager, you will need to install .brd files for the old-fashioned type. To get started, download this zip folder and unzip its contents into the hardware directory of your Arduino sketchbook. Note: These Arduino addon files only work with Arduino 1.5 and up. If you're using an older version of Arduino, either update (and get some cool new features) or download an older version of Addon. Where is your Arduino sketchbook? Well, by default, it should be an Arduino folder in your home directory, but to check again, you can go to File &gt; Preferences in the Arduino and check the Sketchbook location in the text box. Just make sure you close all Arduino windows when you're done. Once you've unzipped this folder to the hardware folder in your Arduino sketchbook (you may actually have to create a hardware folder), your directory structure looks something like this: this directory structure is critical – it should look something like Arduino/hardware/[manufacturer]/[architecture], in this case [the manufacturer] is a sparkfun, and [architecture] is AVR. There's a lot going on in that addon, but one of the most important files is the boards.txt that will add some new entries to your Tools &gt; Board menu. To re-check if board definitions have been added to The Arduino, open Arduino, and check under the Tools &gt; Board menu. There must be some new entires for SparkFun Pro Micro, SparkFun FioV3, Qduino Mini, and other 32U4-based boards. Notice there are two options Pro Micro - 8MHz and 16MHz. It is very important that you choose the Pro Micro option that matches your ship's voltage and speed. It must be listed under Tools &gt; I don't know which ship you're on? Check the bottom of the board where you should find either 5V or 3.3V optional. You should also see the Pro Micro COM port on the Tools &gt; Ports menu. Select it and go to the Example 1 page where we'll upload our first part of the code. If you're using Mac or Linux, follow these steps to make pro Micro (or Fio v3) ready to work on your pc. We're not going to name names here, but installing Pro Micro on Mac OS X and Linux is much easier than on other OS's... Following these strands is crucial to getting your Pro Micro backed in your Arduino environment! Board Installation Using Board Manager With the release of Arduino 1.6.4, adding third-party boards to the Arduino IDE is easy to achieve with the board leader. If you're using an earlier version of Arduino (1.6.3 or earlier), we recommend that you upgrade now. As always, you can download the latest version of Arduino from arduino.cc. To get started, you will need to specify an Arduino IDE board manager custom URL. Open Arduino, then go to Preferences (File &gt; Preferences). Then, at the bottom of the window, paste the following URL into the Advanced Board Manager URLs text box: You can add multiple URLs by clicking the window icon and pasting into one LINE of THE URL. Click OK. Then, open Board Manager by clicking Tools, and then point to the Board Selection tab, and then click Panel Manager. Look for sparkfun in the board manager. You should see the SparkFun AVR Boards package appear. Click Install, wait a moment, and you must install any .brd files that you want to install by specifying the Installed blue that is printed next to the package. Now you should be able to upload the code to several Products compatible with SparkFun Arduino, including Fio and Pro Micro. Manually install .brd files If you are using an earlier version of the Arduino IDE and do not have access to the Board Manager, you will need to install .brd files for the old-fashioned type. When you initially connect Pro Micro to a Mac, it will display the Keyboard Setup Assistant window. This stems from Pro Micro's ability to resemble a HID USB device (such as keyboards and mice) – Mac thinks your Pro Micro is a human input device (which it might be! but not yet). There is nothing to configure in this window, so just click the big red, X to close it. That's all there is to it! Your Pro Micro CDC (communication device class) part (the part that processes USB-to-serial conversions) must be automatically installed on your computer. Installing The Arduino Addon To use Pro Micro or Fio v3 in your Arduino IDE, you need to add some ship definition files to it. That's what we'll do in this section. by downloading pro micro add-in files. Note: These Arduino addon files only work with Arduino 1.5 and up. If you're using an older version of Arduino, either update (and get some cool new features) or download an older version of Addon. With this download, follow these steps to enable Pro Micro in your Arduino environment: addon files are delivered to a zip folder, so you need to get the files first. Find your Arduino sketchbook folder. If you don't know where it is, you can find your sketchbook by looking at the preferences dialog in your Arduino IDE. If you don't have one, create a folder in a sketch book called Hardware. Copy the sparkfun folder that was archived in the first step of the Hardware folder. Your directory structure should look something like .../Arduino/hardware/sparkfun/avr. Restart the Arduino, and see Tools &gt; Panel. You should see some new features including SparkFun Pro Micro, SparkFun Fio V3, and Qduino Mini. If the boards are visible, select the option that matches your board. If you have Pro Micro, make sure you choose the correct operating speed and voltage under Tools &gt; Processor! At this point, select the COM port that lists Pro Micro/FioV3/Qduino. Then head over to the next page where we upload our first sketch! The Arduino standard Blink sketch will have no visible effect on pro micro – there's no LED on the pin 13. In fact, the only LED on board is the power indicator, and the RX/TX is flashing. Unlike other Arduino boards, though, we can control the RX/TX LED in our sketch. So let's get flashing! Upload RX/TX Blinky, Hello World Sketch Copy and paste the code below arduino IDE. Be sure to select the correct ship and COM port that your respective ship listed. Finally, upload [1] it into your Pro Micro language: c/*Pro Micro Test Code by: Nathan Seidle modified by: Jim Lindblom SparkFun Electronics date: September 16, 2013 license: Public Domain - please use this code however you want. It is provided as a learning tool. This code is designed to show you how to control SparkFun ProMicro's TX and RX LED within the sketch. It also helps explain the difference between Serial.print() and Serial1.print(). */ int RXLED = 17; RX LED is set in Arduino pin // Note: The TX LED was not so lucky, we will need to use a predetermined // macro (TXLED1, TXLED0) to control it. (We could use the same macro RX LED too – RXLED1, // and RXLED0.) voided setup() (pinMode(RXLED,OUTPUT); // Set RX LED as output // TX LED is set as output behind the scenes Serial.begin(9600); // This pipe on a series of monitors Serial.println(Initialize Serial Monitor); Series1.begin(9600); This is UART, pipe sensors attached to the ship Serial1.println (Initialize Serial Hardware UART Pins); } voided loop ( ) (in the world!); Print Hello World on Serial Monitor Serial1.println (Hello! Does anyone hear me?); Print Hello! over UART digitalWrite (RXLED, LOW); set the RX LED ON TXLED0; TX LED is not associated with a normally controlled contact, so you need a macro, turn the LED OFF delay(1000); wait for a second digitalWrite (RXLED, HIGH); set RX LED OFF TXLED1; TX LED macro to turn on LED on delay (1000); wait a second } With the code uploaded you should see RX and TX LEDs make turns flashing and turn off every second. You can also open the Arduino IDE series monitor (set to 9600 bps) and see each programmer's favorite two-word phrase. Understanding Sketch RX LED RX LED is linked to Arduino's pin 17. You can control it just like you do any other digital pin. Set it as OUTPUT and digitalWrite([pin], [level]) to HIGH to turn off LEDs, or LOW to turn on LEDs. Here is part of the code highlighted. language:c int RXLED = 17; RX LED is defined in the Arduino pin void setup() (pinMode(RXLED, OUTPUT); / / Set RX LED as output } . . . digitalWrite(RXLED, LOW); turn on RX LED ON digitalWrite(RXLED, HIGH); set RX LED OFF TX LED TX LED was not provided as an Arduino defined pin, unfortunately, so you need to use a couple of macros to control it. TXLED1 turns on the LED, and TXLED0 turns off the LED. Here is part of the code highlighted. language:c TXLED0; TX LED is not associated with a normally controlled contact, so macros are necessary, turn the LED OFF TXLED1; TX LED macros to turn on LED on Serial Monitor (Serial) and Hardware Serial UART (Serial1) In this sketch, you will also notice a couple of Serial Initialization Statements: Language: c Serial.begin (9600); This pipe with serial monitor Serial.println (Initialize Serial Monitor); Series1.begin(9600); This is UART, pipe sensors attached to the ship Serial1.println (Initialize Serial Hardware UART Pins); That 1 makes a huge difference. Think of Pro Micro, which has two separate series ports. One free 1 is for communication with and from your COMPUTER via USB; this is what the serial monitor shows. Serial1 port is a bonafide, hardware UART where your Pro Micro can talk to any batch of serial-enabled hardware. If you open a Serial Monitor, you should only see Hello world! Printed. Hello! Can anyone hear me? is sent out via the hardware UART, where there may be nothing listening. It begs an age-old question: If Pro Micro says Hello!' over the hardware series port, and nothing is there to hear it, does Pro Micro really say anything? Why does my board re-enumerate each upload? Pro Micro emulates the virtual serial port to communicate in series. In fact, it resembles two different series ports - one bootloader, and one sketch. Since bootloader and sketch run Only one of these serial ports is visible at the same time. When you click Upload with Arduino IDE, Pro Micro resets itself and starts its bootloader program. (Bootloader is a low-level program for Pro Micro that allows self-programming through serial.) For our operating system, the bootloader looks like a completely different device, so it gets its serial port number. While Pro Micro is being programmed, the bootloader serial port will be open. When the sketch is uploaded, the bootloader will close, the serial port will close, and the regular Pro Micro serial port will open. What it all limits to is the fact that you have to be patient with Pro Micros. Every time you upload a new sketch, your OS will have to work with your driver's magic before you can open the COM port. This may take a few seconds after the code upload is complete. [1] Note to Windows users: The first time uploading a sketch, it may fail and give you an error. On top of that, Windows will appear that the familiar Device driver software was not successfully installed in the notification. Don't let it worry you too much. If you receive an error message, wait about a minute and try uploading again. Hopefully the upload will succeed a second time, but if it continues to fail, check out how to enter the bootloader section of the FAQ. Windows needs to install the same driver that we've already installed on the Pro Micro boot loader, but it's not able to get everything set before the bootloader exits. So far Pro Micro's most revolutionary feature (as far as Arduinos go) is its true USB functionality. Pro Micro can be programmed to resemble any USB device you could imagine. You can even program it works just like a mouse, keyboard or other HID class USB device. What is HID? This is one of many defined classes of USB devices. Each USB device is assigned a class that determines what its overall purpose is. There are load classes – printers, hubs, speakers, and webcams mention some, but in this example we are emulating HID – Human Interface Device. ATmega32U4 takes care of the USB hardware barrier, but we've still got to reset the firmware one. Time for some example code! We broke the HID example in two parts. Example 2a: USB Keyboard Made Simple Example 2b: USB Mouse Functionality Example 2a: USB Keyboards Made Simple Resemble a USB Keyboard When We Are Using a Keyboard Class. Here are some features available to us with this class: Keyboard.write(char) - This feature will send one character over USB. The passed character can be any standard, printable, ASCII-defined character: 0-9, a-z, A-Z, space, symbols, etc. Here is an example of a line of code: Keyboard.write(z)// It will send one z character to your computer. Keyboard.print (string) - If you need to make a series of Keyboard.write()'s, consider it only one Keyboard.print (). It works similarly to Serial.print() – it's a series of characters and it'll send that stream of characters via USB. Keyboard.println(string) is also defined if you want newline/linefeed to close the string. Example: Keyboard.print(Hello, world); It will send your computer to H followed by e, followed by ... Keyboard.press (byte) and Keyboard.release (byte) give you more precise control over keypresses. They're doing exactly what you'd expect. One pushes the button down, the other releases the button. Make sure you release all the buttons by pressing, otherwise you will encounter some wiggyness on your computer. That's all. You don't need to include any libraries or anything, just refer to any of these features. Here is an example sketch to try it out: language: c/*HID KeyBoard Example: Jim Lindblom Date: 1/12/2012 License: MIT License - Feel free to use this code for any purpose. No restrictions. Just keep this license if you continue to use this code in your future endeavors! Reuse and share. This is a very simplified code that allows you to send z with a short push-button. * / #include &lt;keyboard.h&gt;int buttonPin = 9; Set button on any pin void setup() (pinMode (buttonPin, INPUT); // Set button as input digitalWrite (buttonPin, HIGH); // Pull button high Keyboard.begin(); // Init keyboard emulation } void loop() (if (digitalRead(buttonPin) == 0) // if button goes low { Keyboard.write ('z'); // send ' send 'z'); Send '' z' to your computer using Keyboard HID delay (1000); Delay so no kajillion z's } } In this sketch, joining the pin 9 on the ground will make the Pro Micro spit out z character. If you have a simple, fleeting button handy, tie one end to pin 9 and the other to the ground. Otherwise, just use the wire for short 9 to GND. After the Keyboard.write() or Keyboard.print() function has been made by Pro Micro, your computer will have to decide what to do with it. What your computer does with this character or character string depends entirely on what program is running at that time. If you have a text editor open and active, it will print it out there. Or, you can try using the text box below to check. SparkFun Pro Micro test area example 2b: USB mouse functionality that covers about half of the USB HID library. How about we add the mouse to the mix now? The USB HID mouse requires a few more features, but it's still crazy. There are five features by Arduino's HID class that can be used to implement the mouse: Mouse.move (x, y, wheel) tells the computer to move the mouse to a certain number of pixels in either x, y and/or wheel axis. Each variable can be any value from -128 to +127, with negative numbers moving the cursor down/ left, positive numbers right/up. Mouse.press(b) sends down click the button or button. Button(s) will remain &lt;/Keyboard.h&gt; &lt;/Keyboard.h&gt; until you call Mouse.release(b). The B variable is one byte, each bit representing a different button. It can be set equal to any of the following, or or (|) you can click multiple buttons at the same time: MOUSE_LEFT - Left mouse button MOUSE_RIGHT - Right mouse button MOUSE_MIDDLE - Middle mouse button MOUSE_ALL - All three mouse buttons Mouse.click (b) sends down click (press) followed immediately by the uplift (release) button(s) b. For example, to click the left and right buttons at the same time, try the following:Mouse.click(MOUSE_LEFT | MOUSE_RIGHT); Press and release the left mouse button Here are some example code to show off these features: language: c/ * HID joystick Mouse Example: Jim Lindblom date: 1/12/2012 license: MIT License - Feel free to use this code for any purpose. No restrictions. Just keep this license if you continue to use this code in your future endeavors! Reuse and share. This is a very simplistic code that allows you to convert the SparkFun Thumb joystick (into a HID Mouse. The select button for the joystick is set as left click of the mouse. */ #include &lt;Mouse.h&gt;int horzPin = A0; Analog output horizontal joystick pins int vertPin = A1; Analog output vertical joystick pin int selPin = 9; select the button pin joystick int vertZero, horzZero; Maintains the original value of each axis, &lt;= about 500 int mouseClickFlag = 0; int invertMouse = 1; Invert joystick based on orientation int invertMouse = -1; Noninverted joystick based on orientation void setup() { pinMode(horzPin, INPUT); // Set both analog pins as inputs pinMode(vertPin, INPUT); pinMode(selPin, INPUT); // set button select pin as input digitalWrite(selPin, HIGH); // Pull button select pin high delay(1000); // short delay to let outputs settle vertZero = analogRead(vertPin); // get the initial values horzZero = analogRead(horzPin); // Joystick should be in neutral position when reading these Mouse.begin(); //Init mouse emulation } void loop() { vertValue = analogRead(vertPin) - vertZero; // read vertical offset horzValue = analogRead(horzPin) - horzZero; read horizontal offset if (vertValue != 0) Mouse.move(0,(invertMouse* (vertValue/sensitivity)), 0); // move mouse on y axis if (horzValue != 0) Mouse.move((invertMouse* (horz 0,0); // move mouse on x axis if ((digitalRead(selPin) == 0) &amp;&amp; (!mouseClickFlag)) // if the joystick button is pressed { mouseClickFlag = 1; Mouse.press(MOUSE_LEFT); click the left button down } else if ((digitalRead(selPin)) &amp;&amp; (mouseClickFlag)) // if the joystick button is not pressed { mouseClickFlag = 0; Mouse.release(MOUSE_LEFT); release the left button } about= 500= int= mouseclickflag=0; int= invertmouse=1; invert= joystick= based= on= orientation= int= noninverted= joystick= based= on= orientation= void= setup()= {= pinmode(horzpin,= input);= set= both= analog= pins= as= inputs= pinmode(vertpin,= input);= pinmode(selpin,= input);= set= button= select= pin= as= input= digitalwrite(selpin,= high);= pull= button= select= pin= high= delay(1000);= short= delay= to= let= outputs= settle= vertzero=analogRead(vertPin); get= the= initial= values= horzzero=analogRead(horzPin); joystick= should= be= in= neutral= position= when= reading= these= mouse.begin();= init= mouse= emulation= }= void= loop()= {= vertvalue=analogRead(vertPin) -= vertzero;= read= vertical= offset= horzvalue=analogRead(horzPin) -= horzzero;= read= horizontal= offset= if= (vertvalue= !=0) mouse.move(0,= (invertmouse= *= (vertvalue= sensitivity)),= 0);= move= mouse= on= y= axis= if= (horzvalue= !=0) mouse.move((invertmouse= *= (horzvalue= sensitivity)),= 0,= 0);= move= mouse= on= x= axis= if= ((digitalread(selpin)== 0)= &amp;&amp;= (!mouseclickflag))= if= the= joystick= button= is= pressed= {= mouseclickflag=1; mouse.press(mouse_left);= click= the= left= button= down= }= else= if= ((digitalread(selpin))= &amp;&amp;= (mouseclickflag))= if= the= joystick= button= is= not= pressed= {= mouseclickflag=0; mouse.release(mouse_left);= release= the= left= button= }=&gt;&lt;/= about 500 int mouseClickFlag = 0; //int invertMouse = 1; //Invert joystick based on orientation int invertMouse = -1; //Noninverted joystick based on orientation void setup() { pinMode(horzPin, INPUT); // Set both analog pins as inputs pinMode(vertPin, INPUT); pinMode(selPin, INPUT); // set button select pin as input digitalWrite(selPin, HIGH); // Pull button select pin high delay(1000); // short delay to let outputs settle vertZero = analogRead(vertPin); // get the initial values horzZero = analogRead(horzPin); // Joystick should be in neutral position when reading these Mouse.begin(); //Init mouse emulation } void loop() { vertValue = analogRead(vertPin) - vertZero; // read vertical offset horzValue = analogRead(horzPin) - horzZero; // read horizontal offset if (vertValue != 0) Mouse.move(0, (invertMouse * (vertValue / sensitivity)), 0); // move mouse on y axis if (horzValue != 0) Mouse.move((invertMouse * (horzValue / sensitivity)), 0, 0); // move mouse on x axis if ((digitalRead(selPin) == 0) &amp;&amp; (!mouseClickFlag)) // if the joystick button is pressed { mouseClickFlag = 1; Mouse.press(MOUSE_LEFT); // click the left button down } else if ((digitalRead(selPin)) &amp;&amp; (mouseClickFlag)) // if the joystick button is not pressed { mouseClickFlag = 0; Mouse.release(MOUSE_LEFT); // release the left button } &gt; parasti ap 512 int vertValue, horzValue; // Veikali katras ass konst int jutības analogā izvade = 200; // Augstākas peles jutība = lēnāka pele, jābūt&lt;/Mouse.h&gt; jābūt&lt;/Mouse.h&gt; This sketch is set to use an analog joystick connected to analog pins A0 and A1 to move the mouse cursor. Loop() this code continuously monitors the horizontal and vertical analog value joystick and sends the Mouse.move() command based on what it sounds. This will move the mouse in steps, depending on what sensitivity variable is set to. If the sensitivity is set to 2, the cursor will be moved in 1 or 2 pixel steps. Depending on the orientation of the joystick, there is also the option to trim the mouse based on how V indicates when adjusting the invertouse. You use the selection switch joystick to control the left mouse click. Note that this code uses Mouse.press() and Mouse.release() instead of just one Peli.click(). It requires a little more coding, but it allows you to do things like drag-and-drop, double-click, etc. For more HID example code, see Arduino supplied examples under File &gt; 09.usb &gt; Examples. Or check out Arduino's reference language with USB for more information. On this page, you'll find troubleshooting tips and faqs. Here's a directory of questions covered: Serial Port doesn't show up Tools &gt; Board Menu Pro Micro can be a finicky little thing. There are some strings of events that can lead to its serial port being removed from The Arduino IDE's Serial Port selection menu. If you don't see your Pro Micro serial port, try these steps: close all Arduino windows. (Don't forget to save!) Disconnect Pro Micro from your computer. Wait a few seconds for the device to disconnect. Connect Pro Micro. Open the Arduino back up, check the Serial Ports menu again. Reset to Bootloader We ship Pro Micro with a modified version of the Arduino Leonardo bootloader, with one great improvement. When Leonardo (or any device that uses a stock bootloader) is externally reset, it goes back to the bootloader... and waiting there for eight seconds before it starts running sketch. For some embedded projects, waiting eight seconds before running the program is not acceptable, so we modified the bootloader execution time. Arduino Leonardo Leonardo bootloader for reset functionality. When Pro Micro is externally reset (pulling the RST pin low), it will only briefly (&lt;750ms) start the bootloader before continuing to the sketch. If you need to run the bootloader longer, resetting twice quickly will get Pro Micro to enter bootloader mode for eight seconds. Pro Micro and Fio v3 Pro Micro, Fio v3 and any Backsmega32U4-based board reset functionality. Press reset twice, quickly to enter bootloader mode. Resetting Pro Micro in particular can be tricky because there is no reset button. RST pins are connected to the ground to start the reset. This can be done with a small piece of wire or an externally attached button. Why would you need to enter Mode. Glad you asked ... How to revive Bricked Pro Micro Incorporating all USB tasks into a single chip is an awesome feature that makes Pro Micro and boards like this truly unique. But it also puts more stress on one chip, and if something goes wrong with that chip, the board becomes almost unusable. It's not uncommon for pro Micro's to become bricked and unprogrammable. But in most cases, the brick is reversible! The most common source of Pro Micro brick is uploading code to it with the wrong set board (e.g. programming a 16MHz/5V Pro Micro with a ship set to 8MHz/3.3V). Also make sure that your sketch does not mess with ATmega32U4's PLLCSR registry, or any other registry that determines the USB functionality of ATmega32U4. Pro Micro actually carry the code compiled at the wrong operating speed, but when it tries to re-enumerate, you will be greeted with a statement like this: To revive Pro Micro, you need to find a way to upload a sketch to it with the board set correctly. We can do this with a little help from the bootloader. First, you will need to set the serial port bootloader. But this port is only visible when the board is in bootloader mode, so drag the reset line low twice quickly to call the bootloader reset feature discussed above. Pro Micro's or other devices that don't have a reset button can use a cord to shorten RST to GND twice quickly, or perform a temporary reset button. While Pro Micro has a bootloader change tools &gt; Serial Port menu to the bootloader COM port. Quickly! You only have eight seconds. In Windows, the COM port number of a boot document is typically one number larger than the Pro Micro normal port number. With the set of serial ports, we are ready to re-upload our sketch. But first, check again that the board is set up correctly. Then reset it to the bootloader and quickly upload your sketch. Again, you have to be quick... You only have eight seconds. This can help press Upload keybind - CTRL + U / CMD + U - immediately after the reset. It may take some trying to get the time right. Since the code is compiled first, it can help hit the upload first and then reset. The code starts after the upload, but fails to start after the power cycle we found that ATmega32U4 (such as Pro Micro 3.3V/8MHz) can brown when the output power increase converter. Although the code can run after the upload, the power cycle from the initial current to the draw increase converter is enough to cause the Pro Micro to brown out. Thus causing the sketch does not run. To do this, the user must hide the reset button after the power cycle. FREQUENTLY ASKED QUESTIONS If you have technical difficulties with Pro Micro or Fio v3, see if any of the answers to this FAQ help. If not, please contact our technical Command. What is SRS Pid? The SRS is short for Vender Identification and PID is a short Part Identification. In other words, this number of data determines the device. This is how your computer knows what you've connected, what drivers to use with what COM ports are assigned to it, etc. All local USB devices have a SRS/PID. All The SparkFun ATmega32U4 boards have one vids - 0x1B4F, and they all have a unique PIN. 5V Pro Micros set the requirement for PIN 0x9205 and 0x9206 (one bootloader, one sketch). 3.3V Pro Micros will appear as 0x9203 and 0x9204 for bootloader and sketch, respectively. And Fio v3 is 0xF100 and 0xF101. How do I change the SRS and PID to ATMega32U4 board? Each time you upload the code to the SRS and the PID are uploaded to the device. These values are located on .txt and therefore they will be determined by the board of your choice. Keep in mind that if you choose the wrong board you will get the wrong VID / PID uploaded, which means that the computer can not recognize and the program board. The VID/PID bootloader is part of the bootloader file. To change it, you will need to recompile the bootloader with the new VID/PID, and upload it. Why does my ATMega32U4 board appear twice in Device Manager? Both the bootloader and the sketch have their own SRS/PIN. When you plug the board bootloader starts running for a few seconds and you will see the board appears in Device Manager based on these SRS/PIN. After a few seconds, the sketch will start running and you will see Device Manager disconnect from the bootloader and connect to the sketch. How do I know which COM port to use? When the IDE resets the ship, the COM port is disconnected from the computer. The IDE then searches for a new COM port. This is the port it uses. This is one of those weird things Arduino did to get things working on this chip. How to reinstall Bootloader? Check or reinstall the bootloader tutorial that should work on both ATMega32U4 and ATMega328 boards. If you have the tools to do this, reinstall the bootloader is often easier then try to stay in the bootloader. Since reinstalling the bootloader puts the board back in factory settings it will reset the SRS/PID numbers, allowing your board to work again. Thanks for checking out our Pro Micro and Fio v3 Hookup Guide! If you're looking for more resources associated with these boards, here are some links: Thanks for reading along with our Pro Micro hookup guide! Hopefully now you're fully ready to start using pro micro project of your own. Here are some tutorials that might be worth checking as you continue down the rabbit hole: Send serial data from Arduino to processing back and forth – even at the same time! Favorited Favorite 31 How to collect and use cherry mx switch breakout that allows you to convert a matrix of mechanical switches into a full size keyboard! Favorite 14 LilyPad ProtoSnap Plus is a sewing electronics prototype plate that can be used to learn diagrams and programming with Arduino, then disassemble separately for an interactive fabric or wearable project. Favorited Favorite 1 Create a harmless-looking USB stick with Arduino Pro Micro and 3D printed case that moves the mouse pointer randomly every few seconds. Sure anger your colleagues and friends! Favorited Favorite 5 Feeling Ambitious? Try to create your own custom keyboard! Check out the custom keyboard based on Martin Knobel using Pro Micros and Cherry MX switches: Have a look at these blog posts for inspiration. Inspiration.