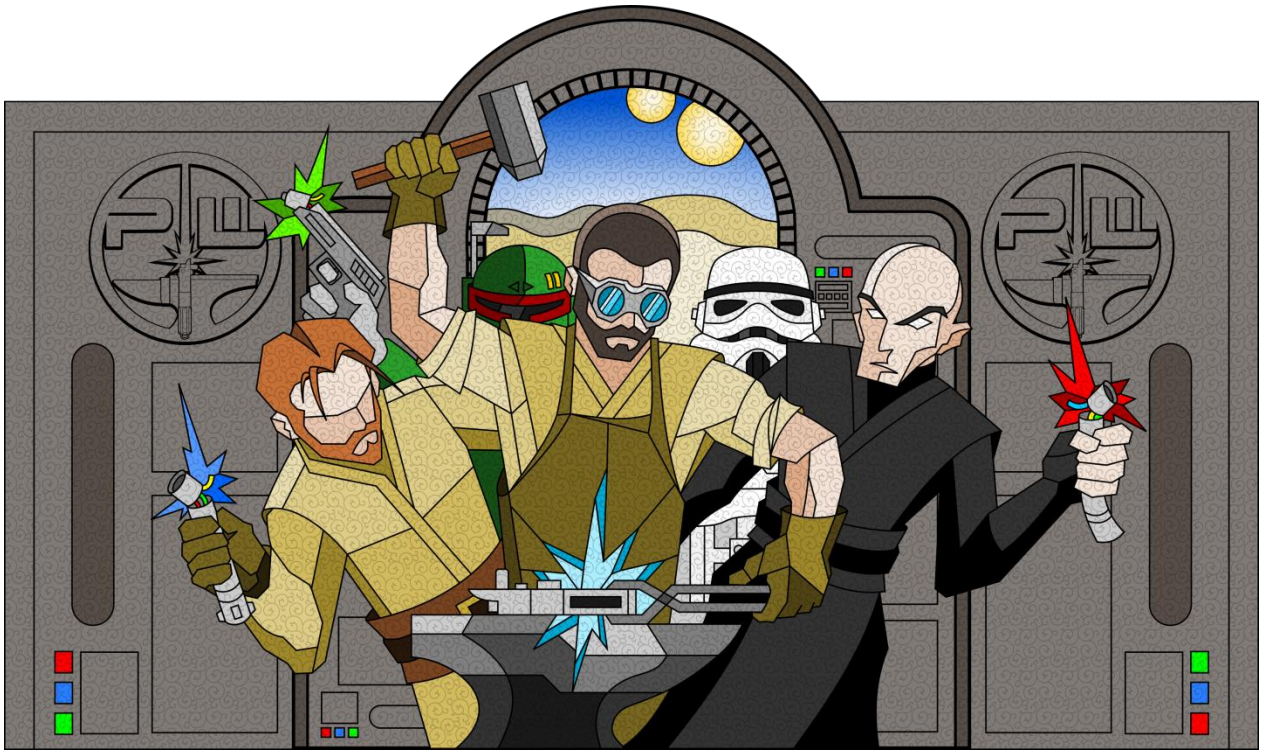
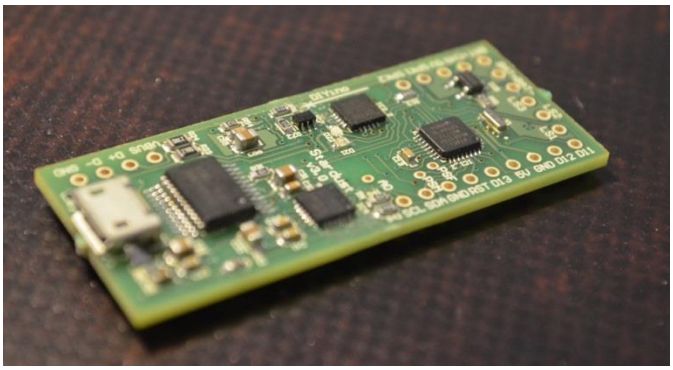


DIYino Stardust v3 User Manual



ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

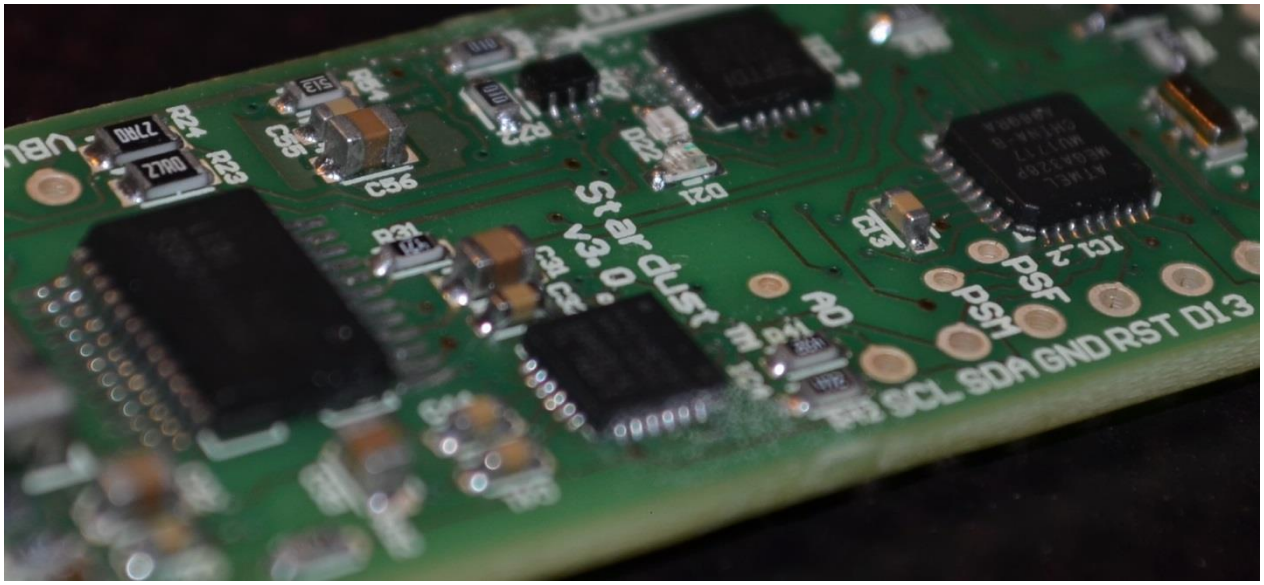
Table of Contents

1 Introduction 3
2 Getting Started 4
2.1 Preparing the board for first usage 4
2.2 A piece of advice on the USB port 5
3 Board Supply Concepts 5
3.1 Standard supply scheme 5
4 Basic Wirings of external components 7
4.1 Wiring buttons and speaker 7
4.2 In-hilt recharge using USB 7
4.3 In-hilt recharge using Recharge Port 8
4.4 Wiring an OLED display (for blaster props) 8
4.5 Charge Status indication LED connection 9
4.6 Copying sound files to the SD-card 9
4.7 Sound font preparation – hum-extension 10
5 Application Examples 10
5.1 High-Power RGB LED setup 11
5.2 Neopixels setup 12
6 Full wiring examples 15
6.1 RGB High-Power LED setup with 3.7V and Single Button 15
6.2 Neopixel setup with programmable kill-key, 3.7V supply and dual charging scheme with USB and Recharge Port 16
7 Technical Specification 16
7.1 Circuit Pinout 16
8 Upload software using Arduino IDE: A Step-by-Step walkthrough 19
9 Generic Saber Code 31
10 Quick-Start Guide to FX-SaberOS 32
11 Related links 32

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

1 INTRODUCTION

DIYino Stardust is an Arduino compatible integrated circuit board for all projects implementing Light/Sound/Motion. Its main field of application is to control lightsaber, blaster and other prop/replica electronics. It is Arduino compatible, i.e. it can be programmed using Arduino compatible IDE's (Arduino IDE, Eclipse etc.)



Board features on a glance:

- Compact size: **21** mm(W)x **51** mm(L)x **5** mm(H)
- Arduino Compatible
- Gapless Wav-audio playback
- Full USB integration (code upload, Li-Ion battery recharge)
- 3W audio amplifier (supports 8Ω/4Ω speakers up to 3W)
- Built in high-end gesture detection sensor (6-axis accelerometer and gyro)
- **K**^{3} Technology: aka. "**K**ill the **K**ill-**K**ey^{tm}": Ultra-low power consumption in sleep/idle mode (0.3mA)
- Includes FTDI USB2Serial chipset for plug-and-play programming with Arduino IDE or similar
- 3 PWM controlled build-in low-side drivers as power extenders with up to 2.4A current capability each
- Input voltage range 3.7V-5V
- Sound files stored on SD-card (*sold separately)

Warning: DIYino Stardust is an electronic board containing parts sensitive to ESD. Final wiring & assembly is under the responsibility of the user with the appropriate tools and ESD protection. If you're not familiar with ESD, please visit : http://en.wikipedia.org/wiki/Electrostatic_discharge
The manufacturer cannot be held responsible for improper use or assembly of the DIYino Stardust board.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

2 GETTING STARTED

2.1 PREPARING THE BOARD FOR FIRST USAGE

After receiving your board, follow the steps outlined below:

1. Plug in your USB cable to the on-board USB port. Please do not ram in the USB connector, see also Chapter 2.2
2. Using a multi-meter measure the voltage between any GND and the 5V signal. The reading should show ~4.2V. This is the peak charging voltage of the USB charger and shows that the board is supplied through the USB and the charger is alive (at this state the battery shall not be connected).
3. Disconnect all power sources from the board.
4. Now your 4Ohm/8Ohm Speaker can be connected between SPK1 and SPK2. Make sure using a multi-meter that you measure the specified impedance of the speaker between its two terminals and no short exists between the terminals and any of the neighboring signals.
5. The board comes delivered with a generic saber code. For more information on its features and wiring, see 9 . Please note that the generic saber code is provided for you to be able to enjoy the saber experience right from the beginning. However if you want to unlock more advanced features, the saber code can be updated from FX-SaberOS, visit the Wiki for more details. Code provided as is, without guarantee.
6. Now that you board is proven to be working fine, you can wire it up fully (of course unpowered) and upload other “sketches” (Arduino slang for application code).

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

2.2 A PIECE OF ADVICE ON THE USB PORT

The delivered USB breakout board can be used to wire up a programming/recharge terminal to be accessible from the exterior of your prop design. This way the electronics can be enclosed and all interaction with the board can be handled via this single USB port. The USB breakout board is also recommended to be used for charging the single cell Li-Ion battery. Why it is so? You can of course use the on-board USB terminal for all these operations as well. However as opposed to the somewhat larger and more robust mini-USB port, the smaller micro-USB port is less mechanically sturdy. It has no guide pins protruding into the PCB to hold it tight in place like the mini-USB, therefore repeated plug-in and –out might dislocate the port and severe the solder joints between the port and the PCB. Therefore please handle the on-board mini-USB port with care (true for all micro-USB port, not only for the one found on the Stardust!) and utilize the USB breakout board for frequent usage.
You can also choose to replace the micro-USB breakout board included with your Stardust with a mini-USB breakout port, for example like the one here: https://www.pololu.com/product/2593

3 BOARD SUPPLY CONCEPTS

Warning: Please note that the DIYino Stardust board does not implement a reverse polarity protection. Reversing the polarity of the supply will lead to board damage!

3.1 STANDARD SUPPLY SCHEME

The DIYino Stardust board shall be supplied directly from a single Li-Ion battery cell. Please connect the positive terminal of the battery to the BAT+ and the negative terminal of the battery to the BAT- to supply the board directly from the battery.
Figure 1 shows wiring of the battery to supply the board. If the USB cable is plugged in, i.e. as during sketch upload, the circuit will be supplied both by the battery and by the USB port, with any surplus current used to charge the battery. If there is no battery connected to the board, it can be supplied via USB through the USB charger circuitry, providing ~4.2V to the board with a maximum charging current of 250mA. Please note that if you connect a 2W 8/4Ohm speaker to the board and supply it only via USB (i.e. no battery connected), depending on the set volume the USB might not be able to supply the powerful on-board audio amp and you might hear a static noise from the speaker, as well as the board resetting due to under-voltage. Therefore it is recommended to connect the battery to the board if you want to use the speaker at full volume.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

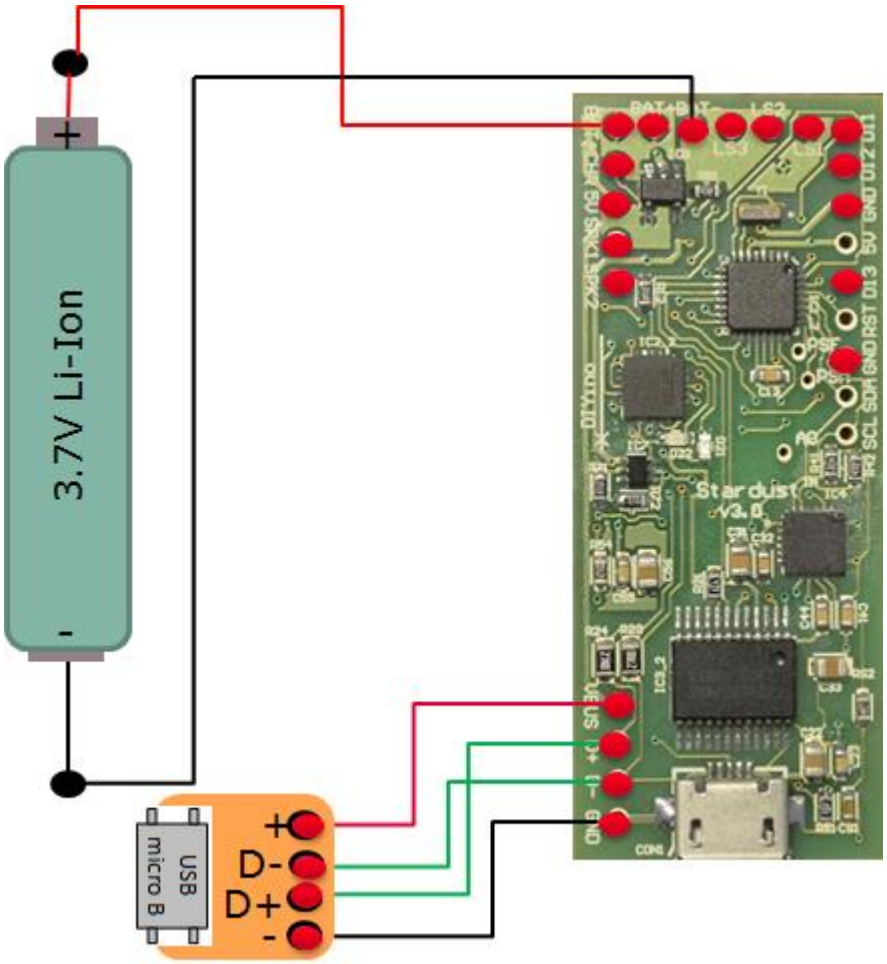


FIGURE 1: STANDARD SUPPLY WIRING

Warning: before connecting any Li-Ion battery to the DIYino Stardust, please ensure that your selected battery complies to the charging characteristic of the USB on-board charger (CCCV with 250mA average charge current at 4.2V charge voltage). In case of doubt please consult your battery vendor. The board manufacturer of the DIYino boards cannot be held liable for any injury or damage due to incompatibility of the used battery with the on-board Li-Ion charger.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

4 Basic Wirings of external components

4.1 Wiring buttons and speaker

Figure 2 shows wiring of the switches and that of the speaker.
The speaker has to be connected between the SPK1 and SPK2 terminals/pins of the board. 4Ω/8Ω speaker can be used, up to 3W output power. It does not matter which terminal of the speaker you connect to which pin. The SPK1 and SPK2 signals are furthermore connected on-board to the ADC input pins A6 and A7 respectively, which can be used to sample the audio file played. This feature finds widespread use to control the flicker of the LEDs in-sync with the sound file being played (**SoundTrack`ER**[tm])
Switches – latching or momentary, although mostly momentary switches are supported by Arduino libraries – are connected with one terminal connected to GND and the other terminal to a digital I/O.

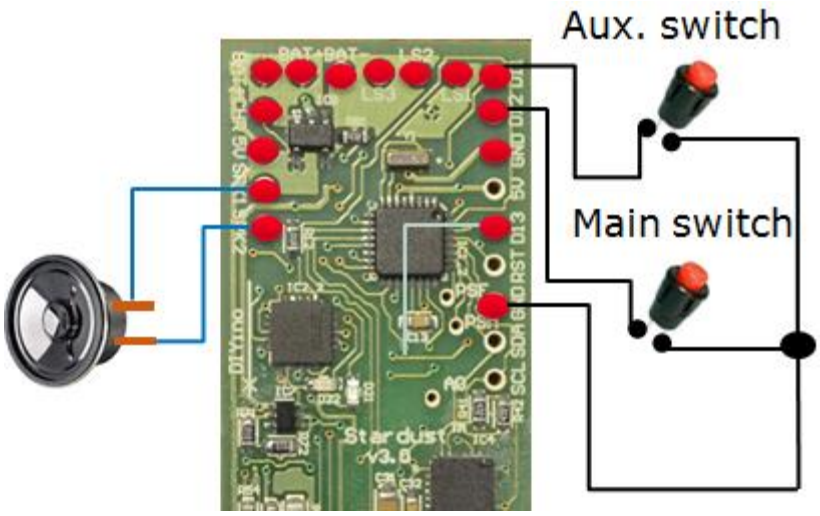


Figure 2: wiring switches and the speaker

4.2 In-hilt recharge using USB

It is very convenient if the battery does not have to be removed from the hilt every time it needs recharging. Therefore so called in-hilt recharge ports found a wide-spread use in saber hilts together with rechargeable batteries (most common type being the 3.7V type 18650). The DIYino Stardust has an integrated Li-Ion USB charger module which can be used to charge the battery directly from USB, thus eliminating the need to include a bulky recharge port in the hilt. To ease recharging and interaction with the board like sketch, a USB breakout board can be used to connect the USB signals to a port in the hilt. Connection of an external USB breakout-board is depicted in Figure 3 .

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

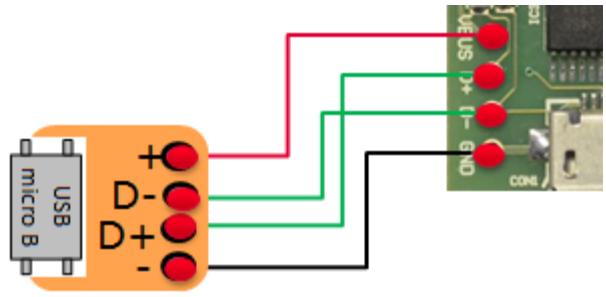


FIGURE 3: IN-HILT RECHARGE USING AN USB BREAKOUT-BOARD

4.3 IN-HILT RECHARGE USING RECHARGE PORT

To make the Stardust board compatible with hilt and chassis design having a so called recharge port (see for example link), the V3 is designed for dual-charging, i.e. the single Li-Ion cell can be charged either via USB or via Recharge Port, making it ideally suited to different circumstances like charging at home or quick recharge in the car or on an event/Con. Wiring of the 2.1mm recharge port can be seen on Figure 4, a full wiring diagram using the dual-charging scheme can be seen on Figure 10

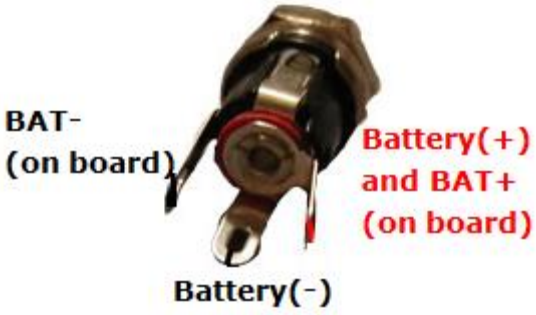


FIGURE 4: IN-HILT RECHARGE USING A 2.1MM RECHARGE PORT

4.4 WIRING AN OLED DISPLAY (FOR BLASTER PROPS)

If you want to add a small display to your prop project - especially popular with sci-fi blaster, but you can include one for a custom MP3 player, armor gadgets etc. – there is a good selection of OLED displays in different display sizes using the SSD1306 driver IC.
The wiring of such an OLED using I2C communication can be seen on Figure 5 . You need to supply the OLD display with 3.3V (some OLED displays have in-built 3.3V LDO, in which case you can also use the 5V pin to supply them) and use the Atmega328P's I2C bus signals A4(SDA)/A5(SCL) to communicate with the display.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

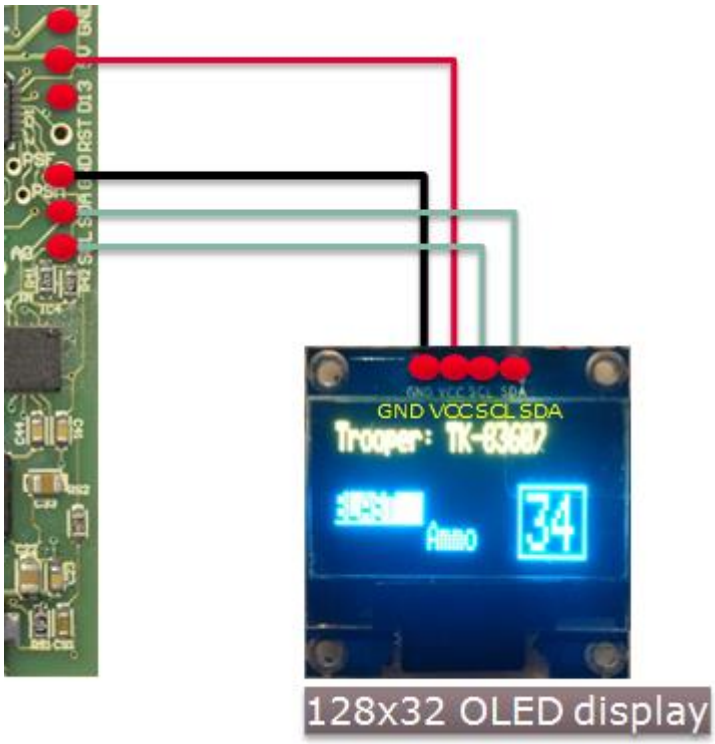


FIGURE 5: WIRING OF AN OLED DISPLAY USING THE SD1306 OLED CONTROLLED (128X32)

4.5 CHARGE STATUS INDICATION LED CONNECTION

The on-board USB charger includes a charge status indication signal which can be accessed on the board via the CHR signal. The signal remains logic low during charging and changes to logic high when the battery is fully charged. This signal can be used to connect a charge indication LED via a proper resistor to light up when charging completes. Connect the LED anode (+) to the CHR signal (via a resistor if needed) and the cathode (-) of the LED to board GND. Alternatively you can connect the CHR signal to the aux. signal A0 and read the charge status voltage via digitalRead() command in the C++ sketch. This makes it possible to implement more sophisticated charge completion functions i.e. notifying via a unique sound, going back to sleep mode until charging finishes etc.

4.6 COPYING SOUND FILES TO THE SD-CARD

On DIYino Stardust sound files can be stored and played back from the SD-card connected to the MP3 chipset.
Sound files stored on the SD-Card will be indexed and accessed according to their physical copy order to the storage media. Therefore some simple rules apply to define SD-card content.

1. Format your SD-card always before changing any file on it
2. Select all the files from this folder and **"Drag and Drop"** them to your SD-card. **NO COPY AND PASTE !!!** : We need to have this file copied in the same order as their filename order. On Microsoft Windows, Copy/paste produce an anarchic copy order, but Drag and Drop produce an ordered copy...

You can organize your files in folders to keep a better overview of the content. All what counts to the MP3/Wav chip is the physical order of the files on the SD-card. You can even drag-and-drop whole

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

folders, but inside the folder you need to establish a defined copy order by preceding files names with a numbering for instance:

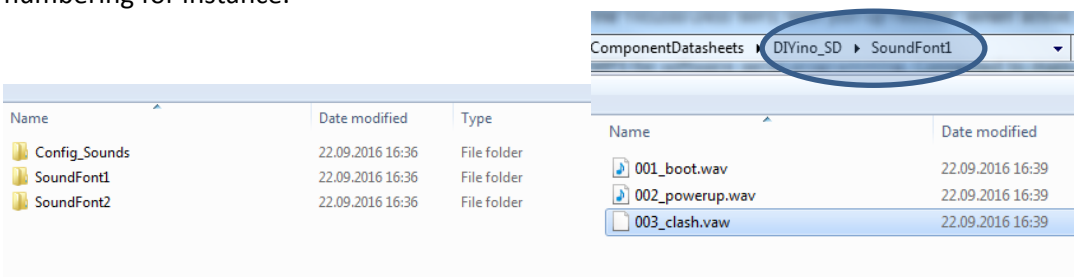


FIGURE 6: COPYING FILES TO THE SD-CARD

4.7 SOUND FONT PREPARATION – HUM-EXTENSION

The DIYino Stardust is capable of gapless playback of sound files in WAV format, i.e. it can change from one sound track to another without a gap in the audio. This feature is mandatory for lightsaber electronics where a seamless transition is expected between the different sound files (like hum, clash, swings etc.).
While the MP3/WAV chips is capable of gapless playback, especially when it comes to relaunching the background noise of a lightsaber (so called “hum”), it cannot be done without spending a significant amount of effort and code space. For this reason a more pragmatic approach to hum-relaunch is implemented in most of the open source lightsaber software which simply relaunches the hum at certain time intervals. In order not to “loose” the hum if the saber is idling, each sound file (except lockup and power off) has to be extended with a bit of hum sound. This is called the hum-extension. Modifying an existing sound font file is easy by simply copy+paste a chunk of hum sound after the end of the sound file. This can be done with freewares (Audacity for instance). A more advanced method is developed by Jakesoft – the father of Arduino lightsabers – in form of an automatic conversion program which can be downloaded here: http://forum.arduino.cc/index.php?action=dlattach;topic=361566.0;attach=192631

5 APPLICATION EXAMPLES

Warning : High-power LEDs (such as the Luxeon, Cree etc.) and strings/stripes of LEDs (such as LED strings composed of many single LEDs or neopixel LED moduls such as WS2812B) are extremely bright. Especially High-power LEDs are considered “class 2 lasers”! You should neither look directly to the beam nor point someone with it when the light source is not diffused/blocked, just like a powerful lamp or flashlight. Manufacturer of the DIYino Stardust board could not be held responsible for any injury resulting from the use of high-power or other type of LEDs/LED modules. To avoid injuries and retina damage due to the high brightness of LEDs, always use protective googles or other means to avoid looking directly into the light source and also take care to protect others (like children) from being able to look directly into the light source.

In this Chapter the most common lighting options of saber designs will be discussed with application notes and wiring diagrams showing how these options can be interfaced to the DIYino Stardust board.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

Caution: in case of a High-Power LED or LED-string setups, you can use the LS pins to adjust the current flowing through the LEDs using PWM control (Pulse Width Modulation) of the transistor gates. The transistors which connect the cathode (-) of the LEDs to the GND can act as voltage controlled variable resistors, thus limiting the current through the LED. For those LEDs having a Vth above the battery voltage, this intrinsic current limiting is sufficient to ensure no overvoltage/overcurrent to the LED, while still offering the full dynamic range of brightness control. But in case the LED has a Vth below (or even far below) the nominal battery voltage (i.e. for red/amber/yellow LEDs), depending on the electrical characteristic of the LED, this intrinsic limiting might not be enough to ensure no overvoltage to the LED, which in turn can lead to damage or degradation of the life time of the LED. Even if the limiting through the transistors is deemed sufficient, during debug the PWM level can be set accidentally to a level which causes overvoltage. Last but not least, if only a small portion of the available PWM range can be used to control the brightness, it can lead to less smooth color blending. Therefore in doubt please include a limiting series resistor, which can be calculated using the following formula:

$$Rseries = \frac{Vbattery - Vth_LED}{I_LED}$$

Example: suppose you use a Red LED with Vth_LED=2V and you want the current to be I_LED=700mA, you use a Vbattery=3.7V, type 18650 battery. The value of the resistor you need to wire in series to the LED is:

$$\frac{3{,}7V - 2{,}0V}{0{,}7A} = 2.4\Omega$$

Please check the power rating of the resistor you intend to use.

5.1 High-Power RGB LED setup

Figure 7 depicts connection of a High-Power LED module to the DIYino Stardust. It depicts a 3-color LED module consisting of 3 LED dies in colors Red, Green and Blue. The same wiring can be extended to all different variants of HP-LEDs, like a single die HP-LED which can be controlled with a single Low-side driver. Please include appropriate resistors (see previous section) for the individual LED dice, as depicted on Figure 7.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

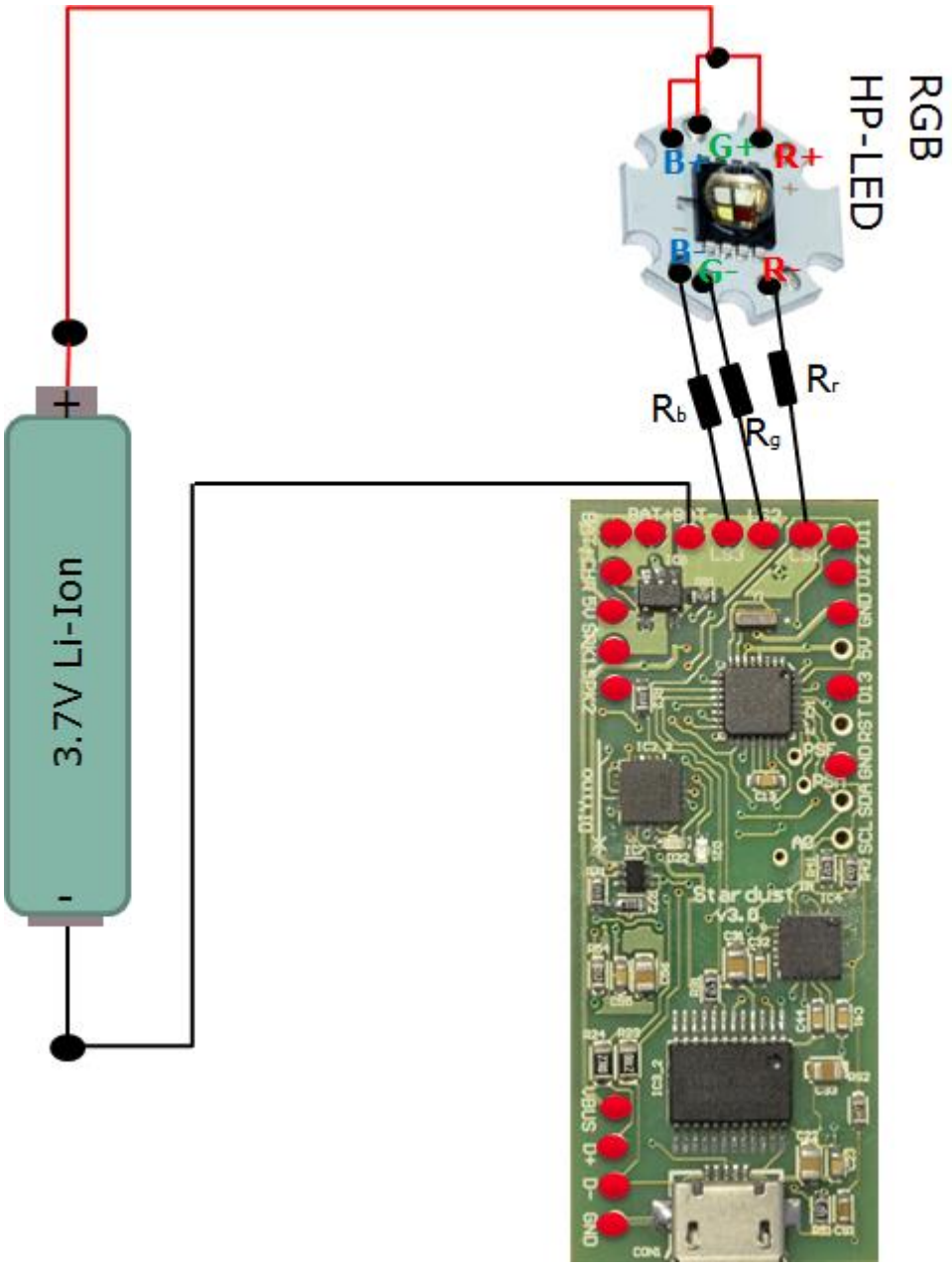


FIGURE 7: WIRING OF A HIGH-POWER LED MODULE (RGB)

Please note that that maximum DC current that can be switched by the individual LS pins is 2.4A. If the HP-LED used in the design involves LED dies with a max current above this rating, you must connect the cathode of the LED die to multiple of these LS pins and ensure they are controlled with the same PWM setting to avoid violating the maximum rating of the transistors.

5.2 NEOPIXELS SETUP

"NeoPixel" is Adafruit's brand for individually-addressable RGB color pixels and strips based on the **WS2812**, **WS2811** and **SK6812** LED/drivers, using a single-wire protocol. Commonly used neopixel

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

stripes are composed of individual LED segments connected together to form a ladder similar to LED-strings. The stripes can be cut at any joint and multiple striped can be connected together at these joints as well.
Neopixel LED modules integrate RGB LEDs and a control circuit which uses PWM (Pulse Width Modulation) to control the brightness of each die individually. For that purpose each LED module has a shift register composed of 24-bits, 8-bits belonging to each of the colors Red, Green and Blue. The shift registers are connected between the DI (Data-In) and DO (Data-Out) pins of the individual segments. In a stripe configuration one segment shift register is connected serially to the shift register of the next segment. If a blade has a stripe with 100LEDs, it means during programming 100x24=2400 bits of data have to be transmitted to the stripe using neopixels own serial protocol to fill up all PWM registers, which in turn determine the brightness of the LEDs. This takes only a few us, so programming can happen so fast, that transitions seem smooth to the eye.
A neopixel stripe has only 3+1 signals:
5V: supply of the stripe
DI: Data-In for the single-line serial protocol
GND: Ground or negative of the stripe
DO: Data-out, this signal has to be used only of you want to connect several stripes together which are not continuous (back-to-back stripes or think about the cross guard of a Kylo Ren style saber)
The specification of the neopixel LED segments defines 5V as nominal voltage for the stripes, however the module can work with a much lower voltage as well. A lower voltage is even necessary to minimize power loss during operation, because any excess voltage above the voltage threshold of the used LEDs (Red ~2V, Green and Blue ~3.5V) is “wasted” over protection circuits in the control logic. Therefore a voltage source around the LED Vth is ideal to power neopixel strings.
For a complete characterization of neopixel brightness and current consumption please see LINK
Neopixels chips consume considerable power even when all the LEDs are switched off (all 0’s). This static current consumption amounts to 1mA per LED. You can quickly calculate what this means to your battery life time if you use – let’s say – 60LED/m type of neopixels, back to back, in a 80cm blade. There will be 100 LEDs in your blade, drawing 100mA even if all LEDs are switched off. It will even discharge the best battery in less than one day. It is not hard to predict, that with the introduction of the 144LED/m type neopixel stripes, some blades will include much more than 100 LEDs, which will lead to an even quicker discharging of the battery. Killing power to the circuitry using a kill-key is a good method to lengthen the shelf life of your saber, but quite annoying during a show if you have to keep plugging in/out the kill key. Therefore a unique method was invented using the existing DIYino Stardust architecture. If the GND pin of the neopixels stripe is connected to the LS pins instead of the GND of the battery/DIYino Stardust, the transistors of the LS pins can be used to cut power to the neopixels stripe. If the blade is activated, the transistors have to be fully switched on (using the digitalWrite function) to connect the GND of the neopixels to the Battery(-), therefore powering the stripe. If the blade is retracted/switched off, the transistors have to be fully switched off in order to avoid the static current consumption of the stripes discharging the battery. Please take note that the restrictions as to the maximum current capability of the DIYino Stardust board apply also here (max 6A!!!).

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

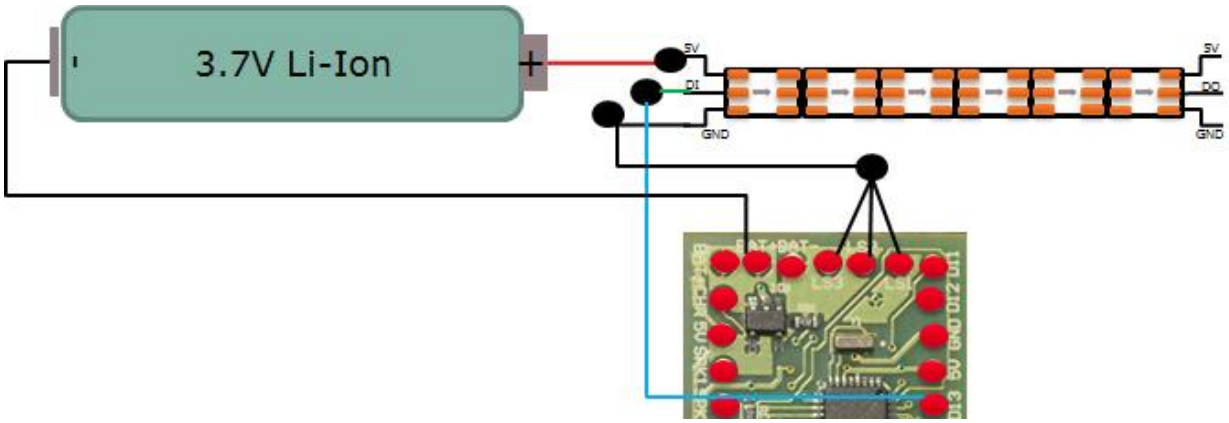


FIGURE 8: NEOPIXELS WIRING USING 3.7V VOLTAGE SOURCE, "PROGRAMMABLE KILL-KEY"

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

6 Full wiring examples

6.1 RGB High-Power LED setup with 3.7V and Single Button

The Figure 9 shows the full wiring diagram of a HP-LED saber using a single 3.7V Li-Ion battery as supply, using a single main button. The HP-LED module is directly supplied from the Battery+ (note: for a more compact wiring schematics, resistors to the LED dice are not shown, but might needed to be included, see 5.1)

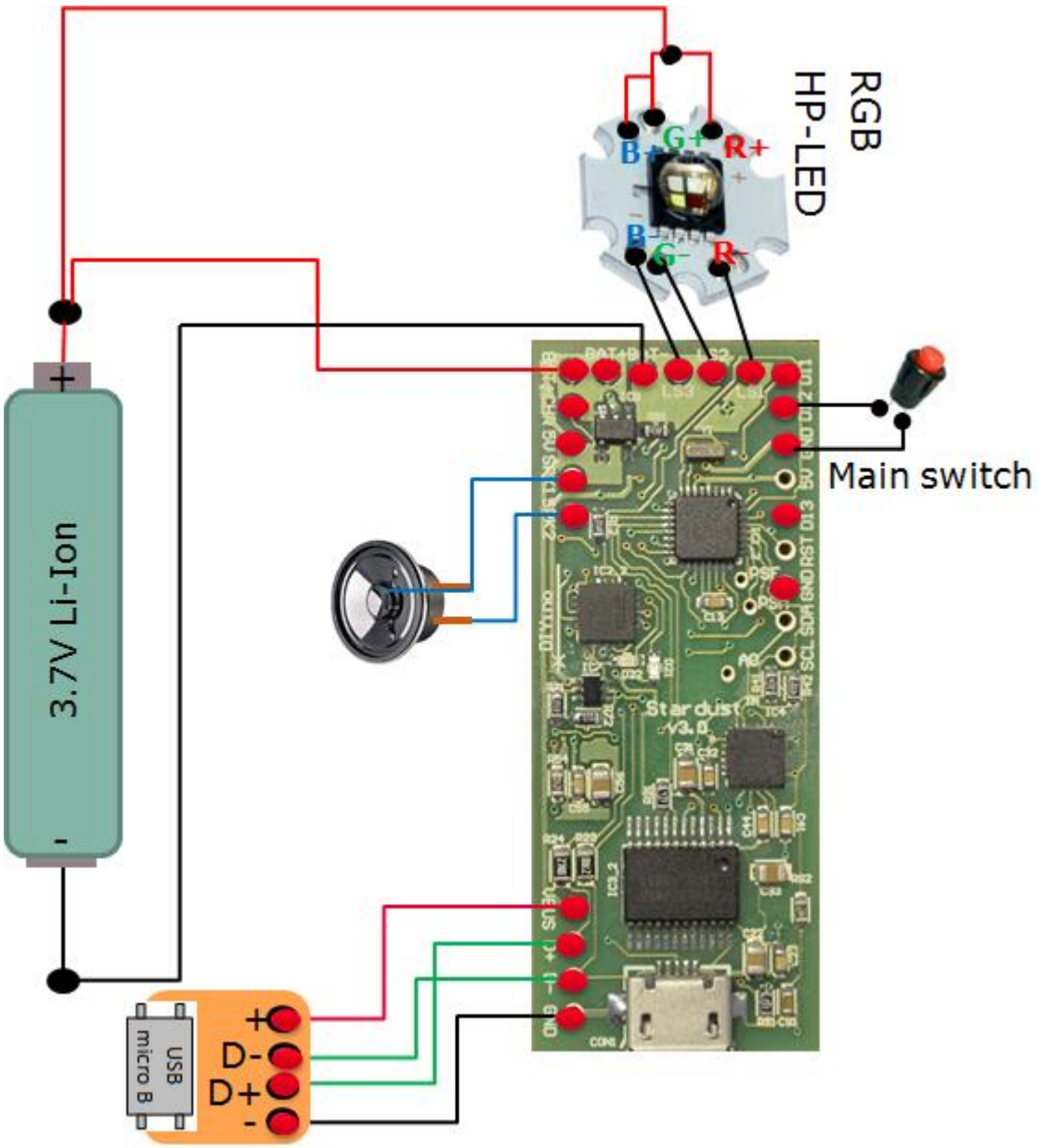


Figure 9: full wiring diagram of a High-Power LED (RGB) saber

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

6.2 NEOPIXEL SETUP WITH PROGRAMMABLE KILL-KEY, 3.7V SUPPLY AND DUAL CHARGING SCHEME WITH USB AND RECHARGE PORT

On Figure 10 the full saber wiring diagram can be seen. It includes

- an in-hilt USB recharge port
- an in-hilt 21mm recharge port
- 3.7V Li-Ion battery as voltage source (for instance 18650)
- A neopixels LED-stripe of (theoretically ☺) any length
- Programmable neopixels stripe kill-key setup
- A main and an aux. switches
- Speaker

When connecting the programmable neopixels stripe kill-key, you have to calculate max. current consumption of your LED stripe and connect the LSx pins accordingly, i.e. if you anticipate 2A-3A's, connect 3 of the LS pins to the GND terminal of the stripe. Up to max. 6A over all LSx pins due to limited heat dissipation on the board!!!

All considerations in the Chapters describing the individual parts of this circuit diagram apply here as well. Please read them carefully.

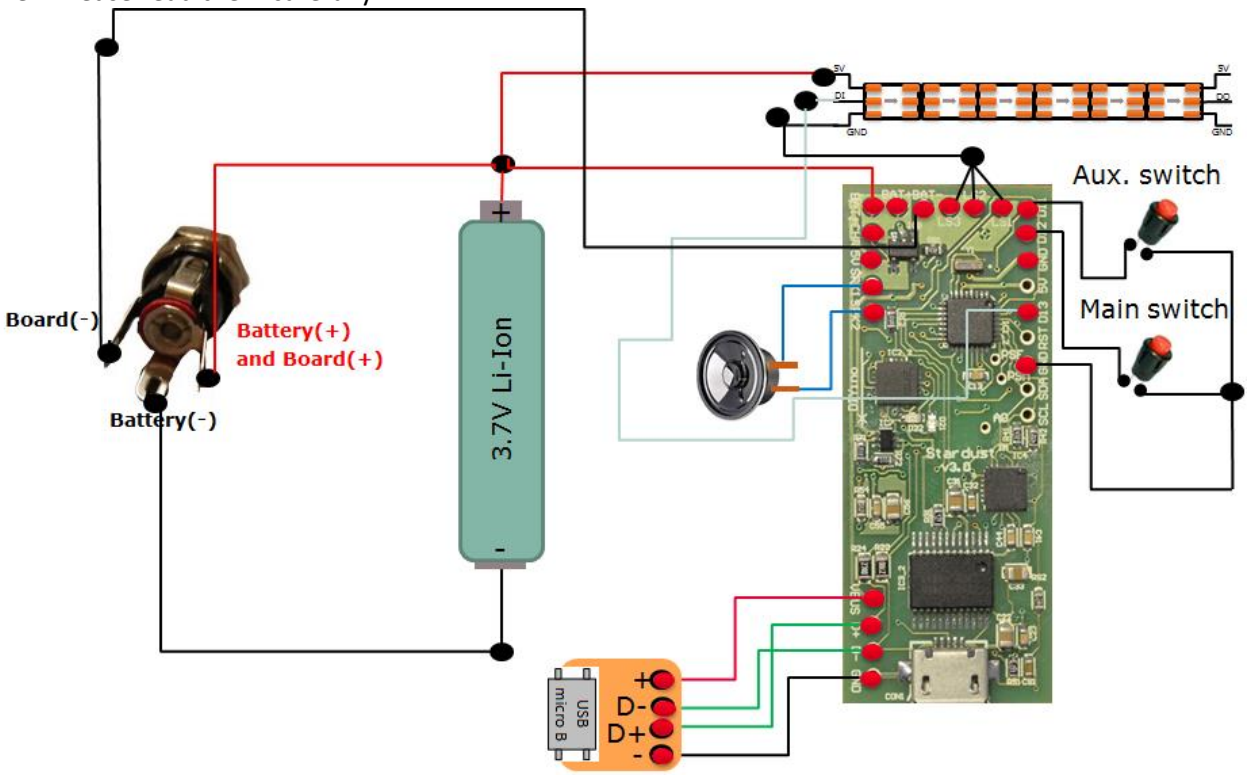


FIGURE 10: FULL WIRING DIAGRAM OF A NEOPIXEL STRIPE BASED SABER WITH DUAL-CHARGING

7 TECHNICAL SPECIFICATION

7.1 CIRCUIT PINOUT

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

Pin	Functionality	Comment
BAT+ (2x)	Positive terminal of a single Li-Ion 3.7V battery cell.	2 breakout signals
BAT-	Negative terminal of a single Li-Ion 3.7V battery cell.	
LS1	Drain of the Low-Side switch 1	HP-LED: connect cathode of Red die here (via resistor if needed)
LS2	Drain of the Low-Side switch 2	HP-LED: connect cathode of Green die here (via resistor if needed)
LS3	Drain of the Low-Side switch 3	HP-LED: connect cathode of Blue die here (via resistor if needed)
CHR	Charge completed signal of the USB charger circuit.	Can be connected to the anode of an LED (via resistor if appropriate) to light up when battery is fully charged over USB.
B	The „B“-point!!!	It can be used to implement the *USB Kill-Key*^{tm} Feature
5V (2x)	Legacy Arduino name of board logic supply.	2 breakout signals.
RST	Reset pin of the Atmega328P, with pull-up to 5V.	Pin used for burning bootloader.
D11	D11 digital I/O of Atmega328P, PWM capable, connected to Gate of LS6 Low-Side driver, with 100kΩ pull-down to GND. MOSI pin for ICP.	Pin used for burning bootloader.
D12	D12 digital I/O of Atmega328P. MISO pin for ICP.	Pin used for burning bootloader.
D13	D13 digital I/O of Atmega328P.	Pin used for burning bootloader.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

	SCK pin for ICP.	
GND (3x)	Board(-) or GND. Connected to GND plane of the PCB.	Pin used for burning bootloader.
SPK1	Speaker terminal 1. Connect to speaker directly. Other speaker terminal to SPK2.	Connected to A6 of the Atmega328 for *Sound-Track'ER*^{tm}
SPK2	Speaker terminal 2. Connect to speaker directly. Other speaker terminal to SPK1.	Connected to A7 of the Atmega328 for *Sound-Track'ER*^{tm}
GND	Board(-) or GND. Connected to GND plane of the PCB.	Connected to BAT- via star-point. Digital GND, do not connect to battery negative terminal. 3 breakout signals.
SDA	A4 digital I/O of Atmega328P with input 10-bit ADC.	SDA I2C signal for communication with MPU6050 or an external device using I2C protocol (i.e. OLED display).
SCL	A5 digital I/O of Atmega328P with input 10-bit ADC.	SDA I2C signal for communication with MPU6050 or an external device using I2C protocol (i.e. OLED display).
PSF	FTDI power switch	Pulled to GND. Can be connected to 5V signal to disable FTDI chipset.
PSM	MP3 power switch	Pulled to GND. Can be connected to 5V signal to disable MP3 chipset and Audio Amp.
A0	A0 digital I/O of Atmega328P with input 10-bit ADC.	Auxiliary signal, can be used as an additional ADC input or as accent light signal
VBUS	USB supply voltage	5V voltage input from USB bus or USB charger. Breakout signal for

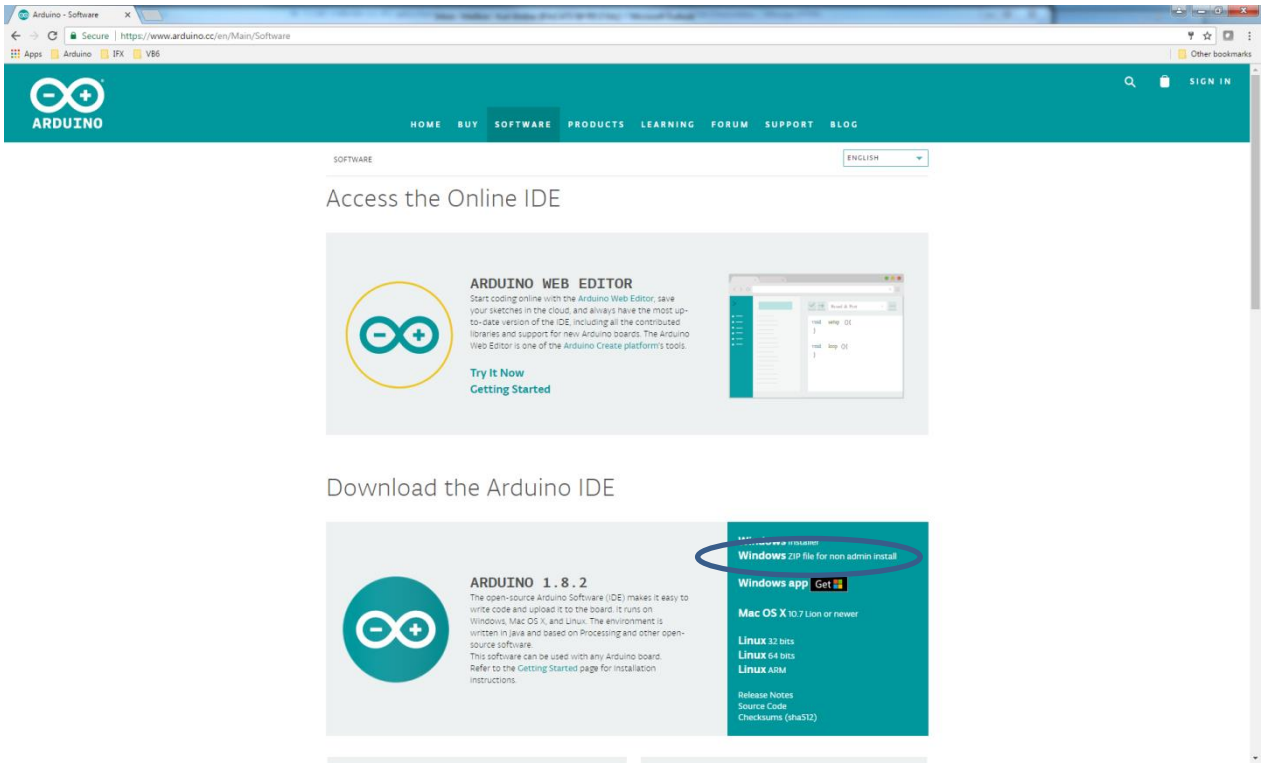
ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

		external USB connector hook-up
D+	USB positive	Breakout signal for external USB connector hook-up
D-	USB negative	Breakout signal for external USB connector hook-up

8 Upload software using Arduino IDE: A Step-by-Step walkthrough

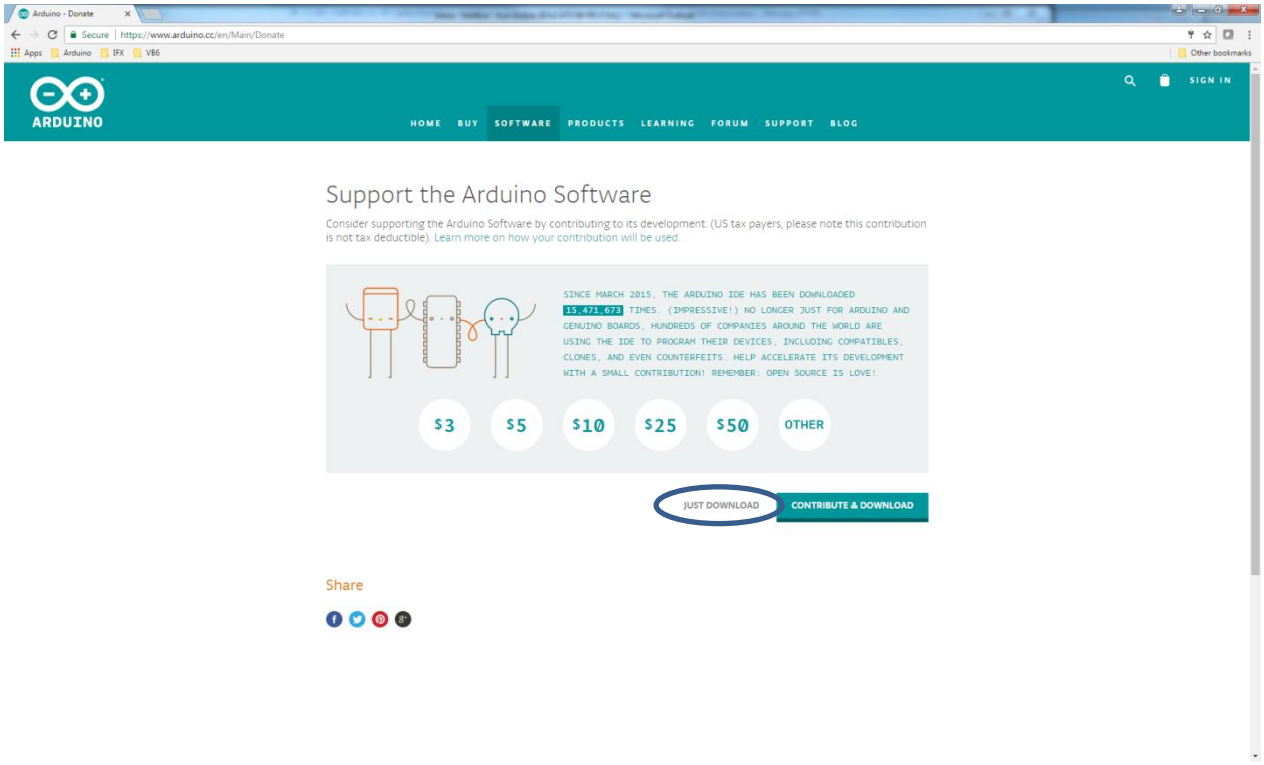
Please visit the Arduino Home Page for instructions how to install the Arduino IDE and upload code using it to your DIYino Stardust board:
https://www.arduino.cc/en/Guide/Windows (instructions are also available for Mac OS X and Linux)
The DIYino Stardust boards are configured as an Arduino/Genuino Uno compatible board, so when choosing your board please select Arduino/Genuino Uno.
To upload open-sorce code to your Stardust board, follow the steps described here:

Step 1: Download Arduino IDE from http://www.arduino.cc. Go to Software and then to Download the Arduino IDE.



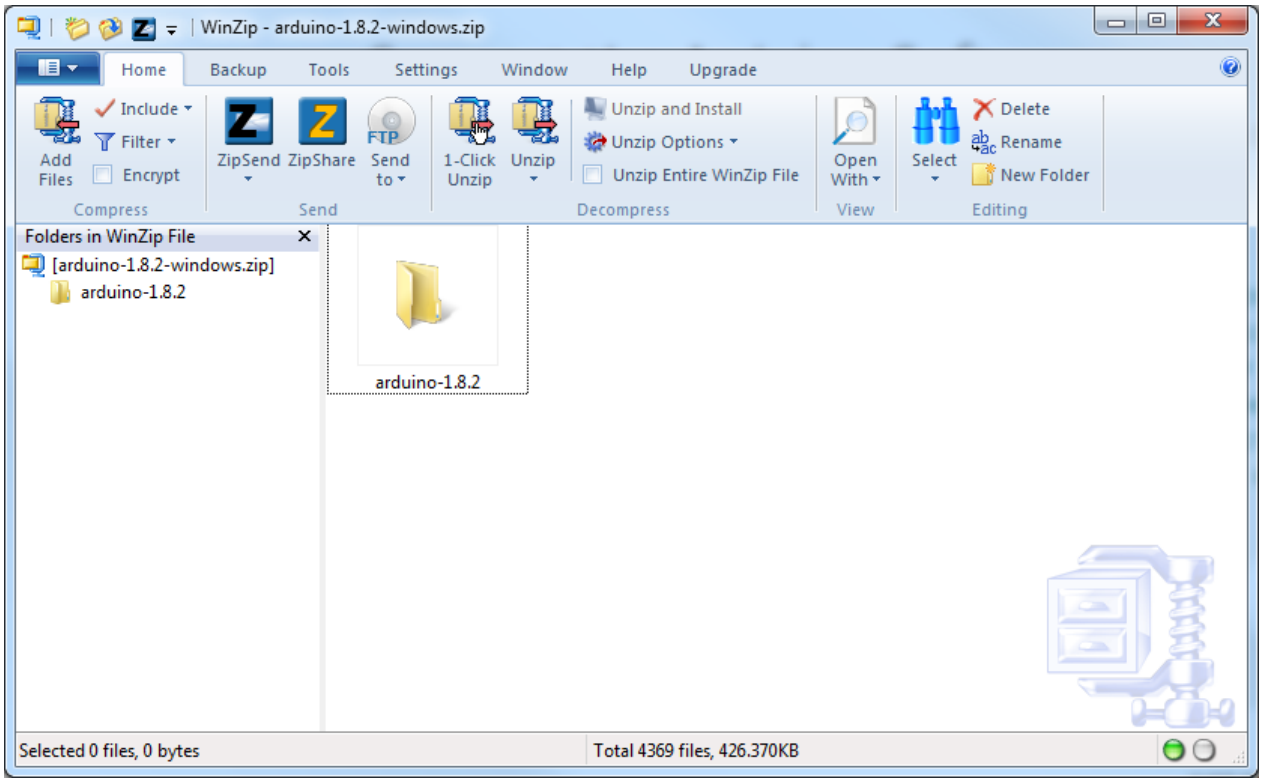
ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

Step 2: From this page you will be redirected to another page where you can donate to the developers of the Arduino IDE. Optional, you can choose not to donate by clicking on Just Download.



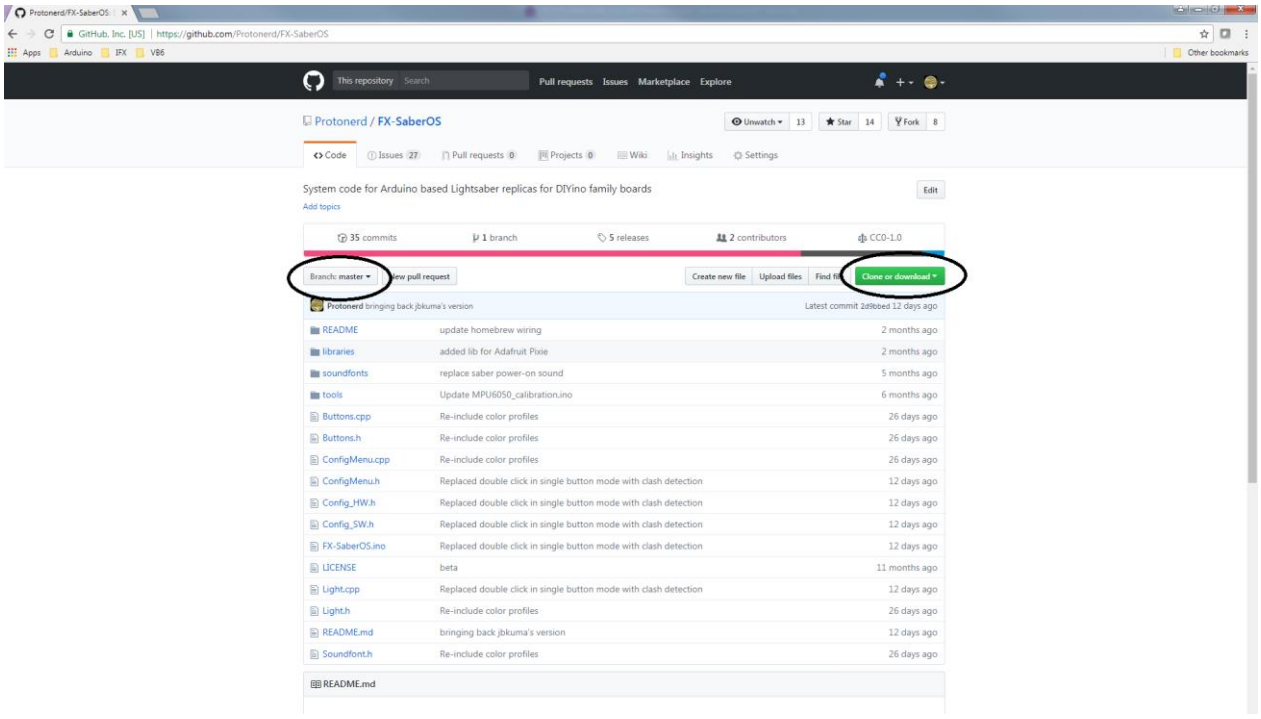
Step 3: The Arduino IDE will be downloaded. When download is completed, unzip the software anywhere you like to your PC.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

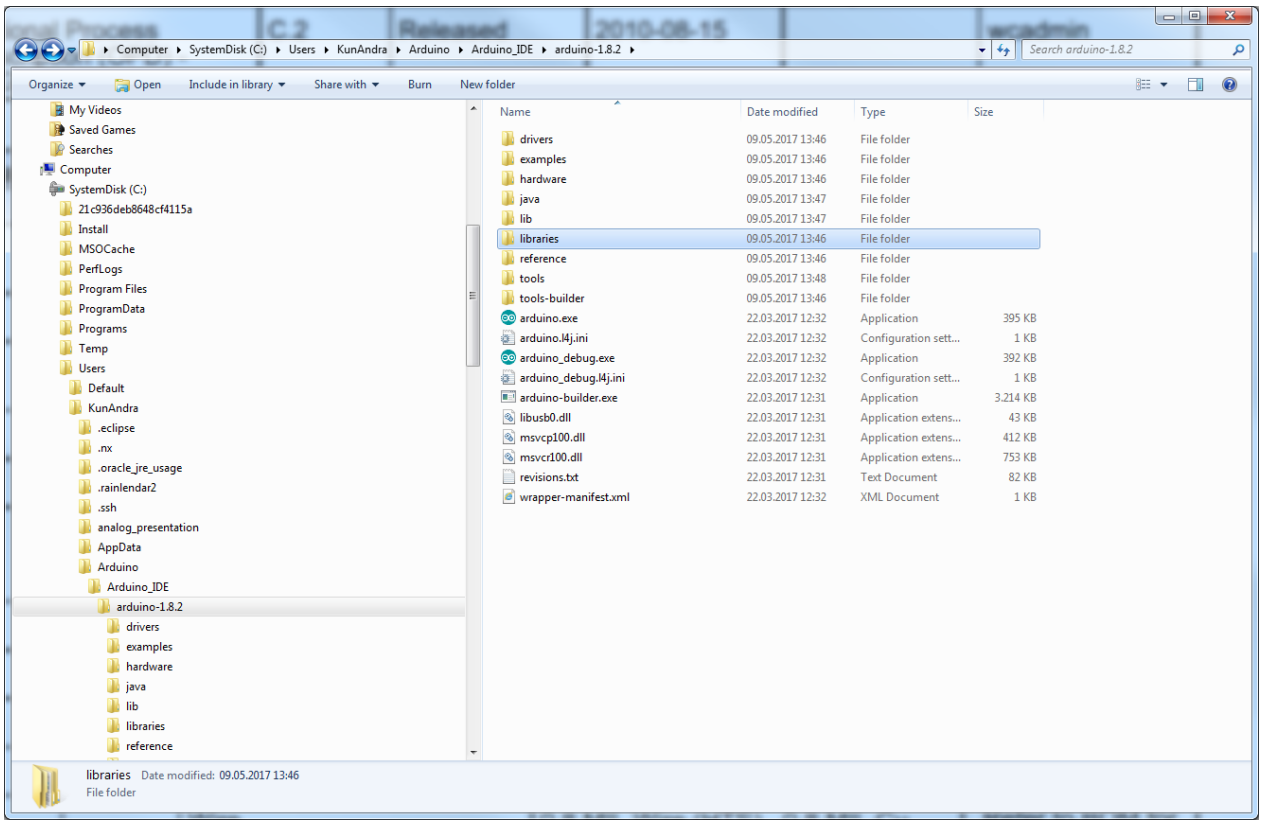


Step 4: Download the lightsaber code of your choice, from example from GitHub: https://github.com/Protonerd/FX-SaberOS

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

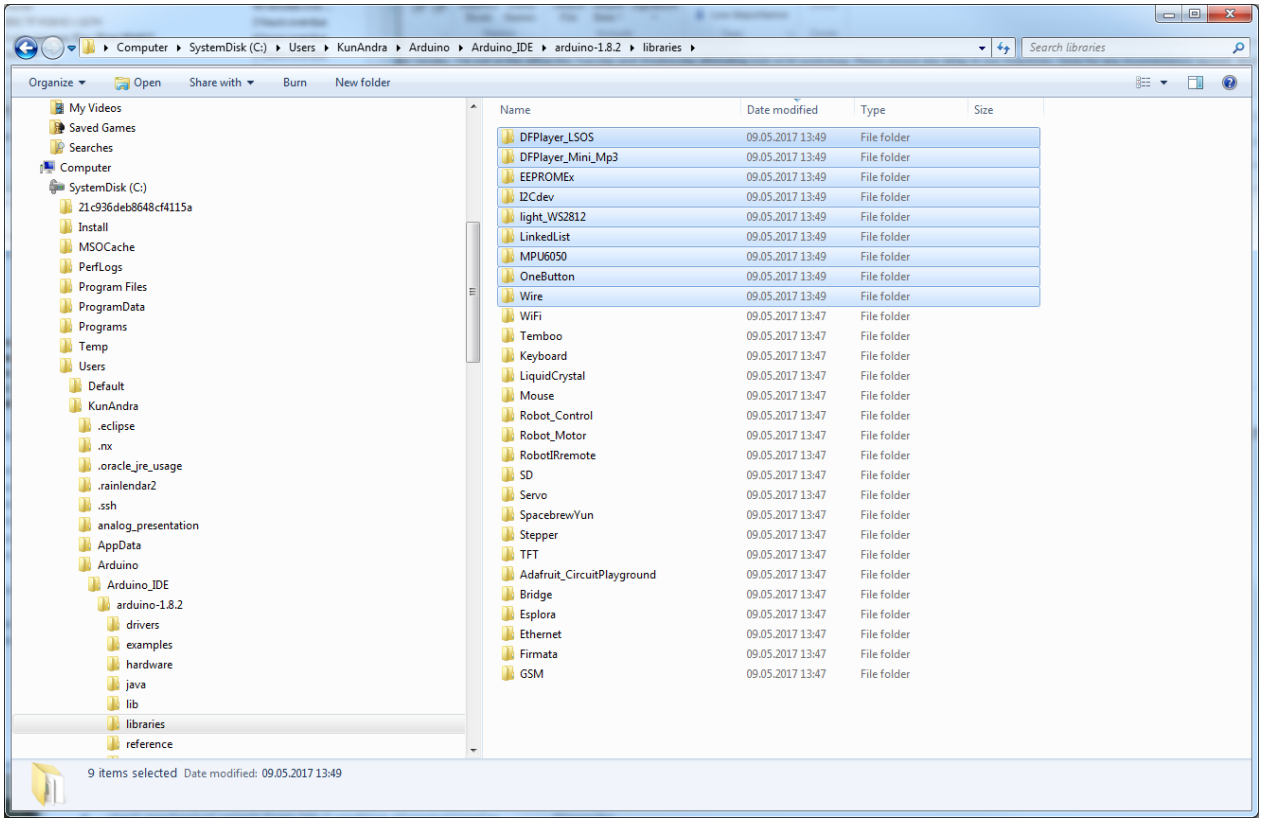


Step 5: Navigate to the folder you unzipped the Arduino IDE and go to the libraries folder.



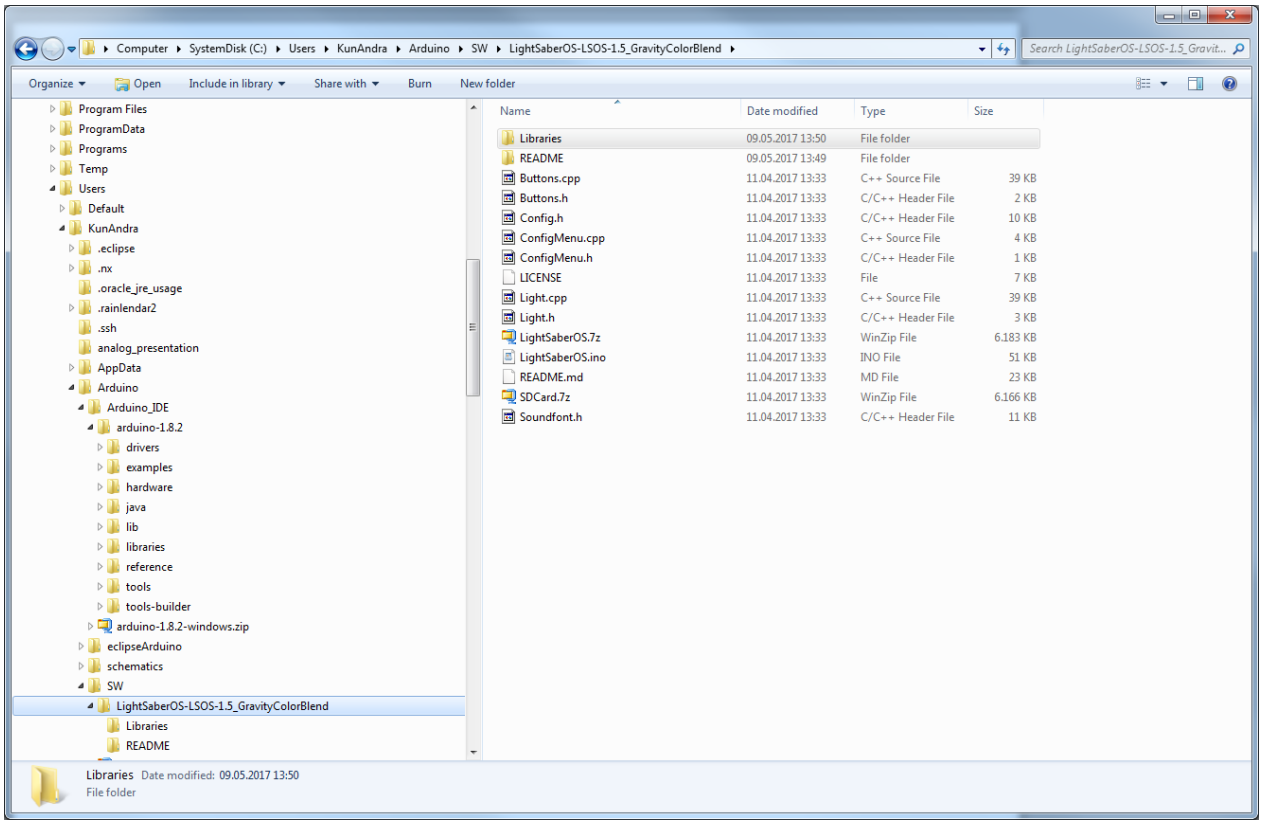
ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

Step 6: This libraries folder contains basic libraries for Arduino. For the IDE to find the libraries necessary to compile your downloaded lightsaber code, you must copy all folders located in libraries folder of the lightsaber software ZIP file to this folder. Sounds more difficult than it really is, look at the next picture which clearly explains what is to be done:



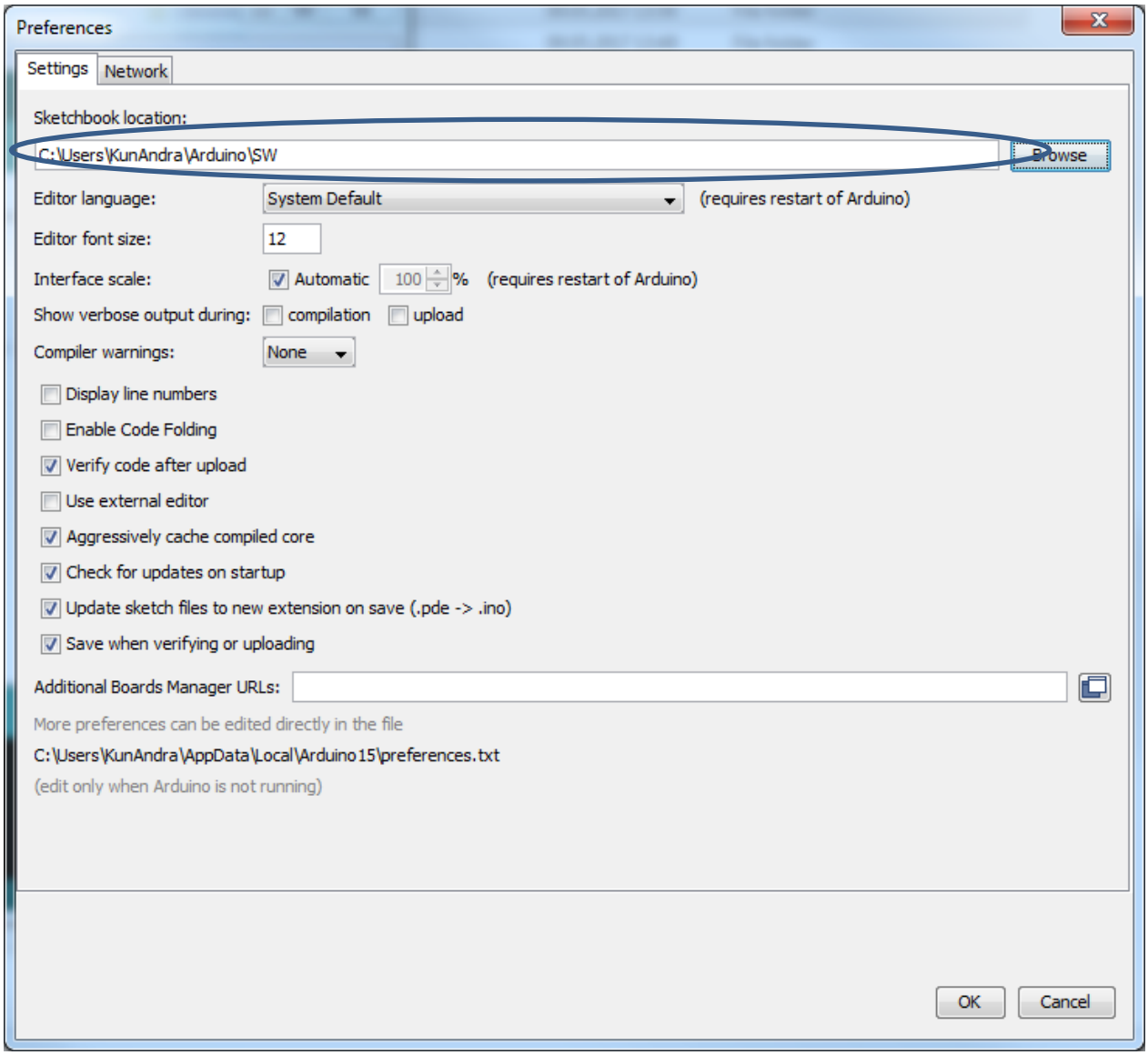
And in the next picture you can see where the libraries of the lightsaber software can be found:

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



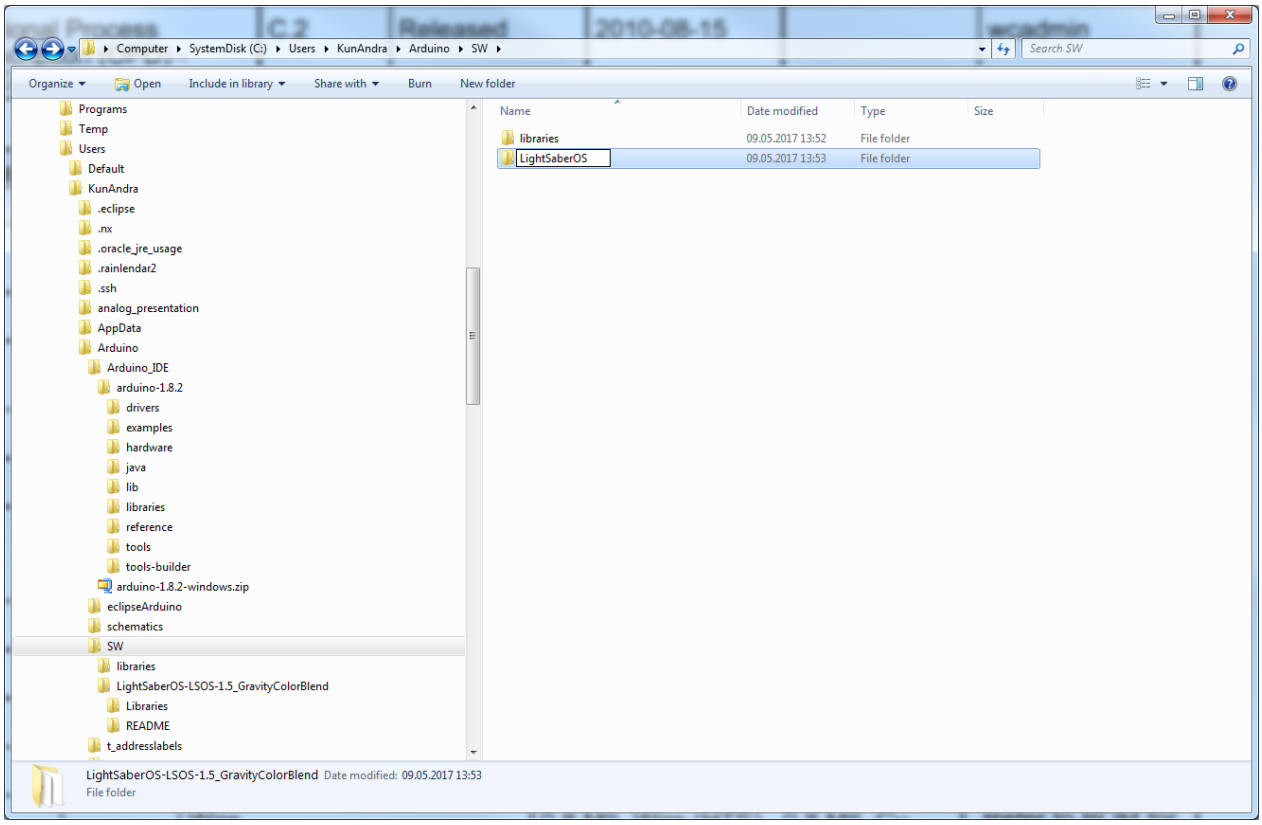
Step 7: Now it's time to start the Arduino IDE. Simply start the arduino.exe. Then go to File/Preferences and specify the folder containing the lightsaber code folder you downloaded (see path in the pic above and compate it to the path indicated under Sketchbook location):

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



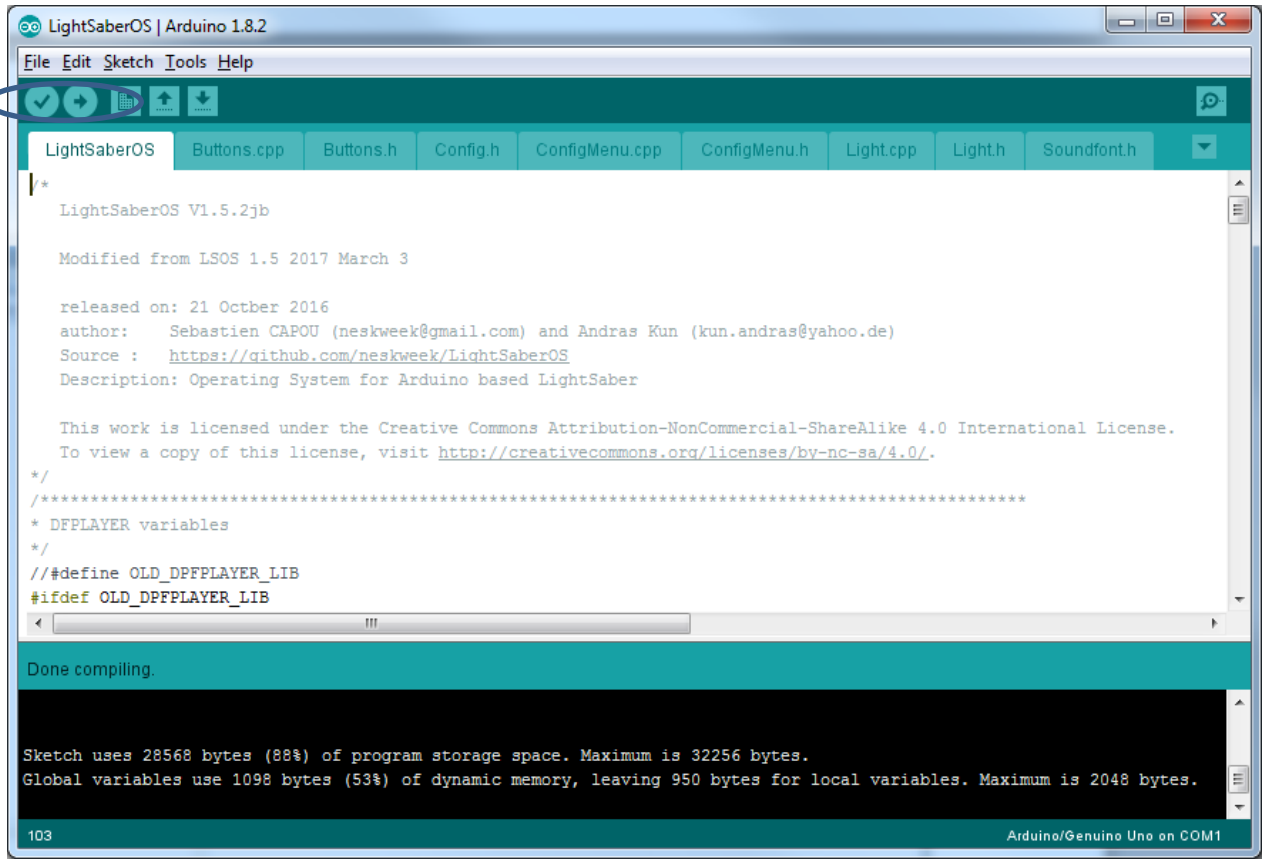
Step 8: It is crucial that the name of the lightsaber code folder be the same as the .ino file in that folder, see in the next picture. I.e. if the ino file name is LightsaberOS.ino, the folder containing the .ino file also must be named LightsaberOS. Rename the folder accordingly.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



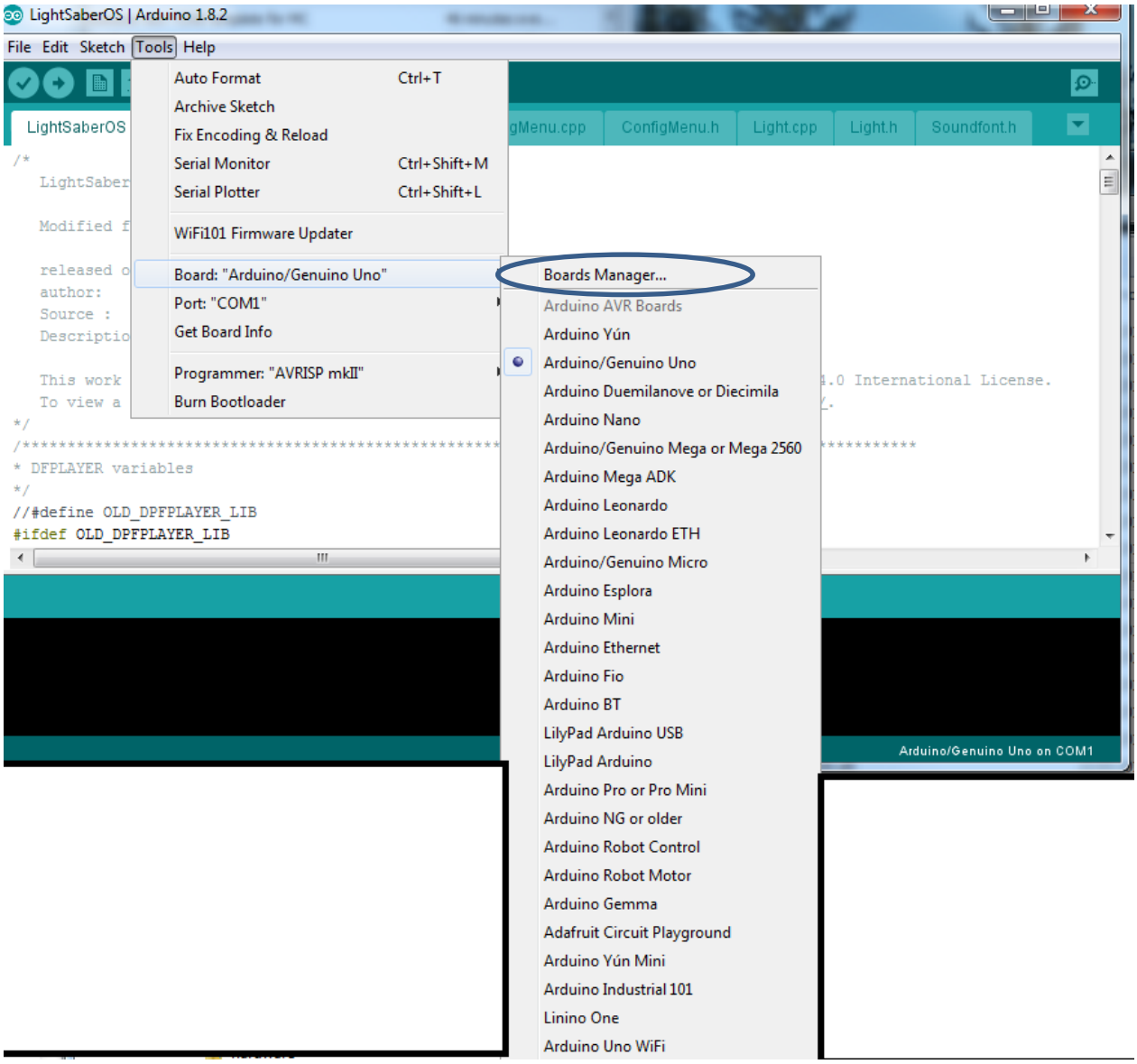
Step 9: Once everything is settled, go to File -> Open -> /../../<*lightsaber code .ino file name*>. See next picture how it should look like once the code is opened. The Check and Arrow buttons can be used to compile or compile/download the code to your Arduino compatible board.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



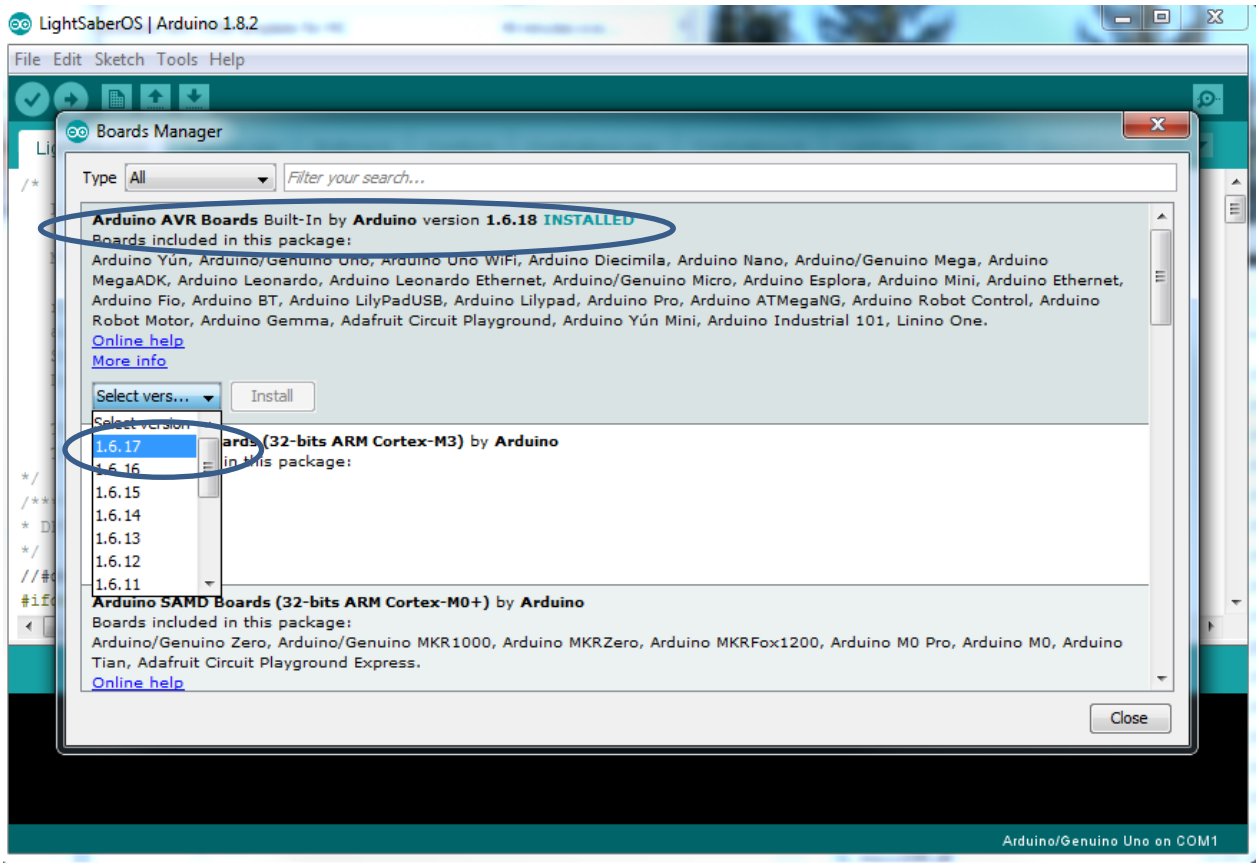
Step 10: Prior to downloading the code to your board, the type of your microcontroller needs to be specified. To do this, enter the Boards Manager as depicted on the following picture:

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



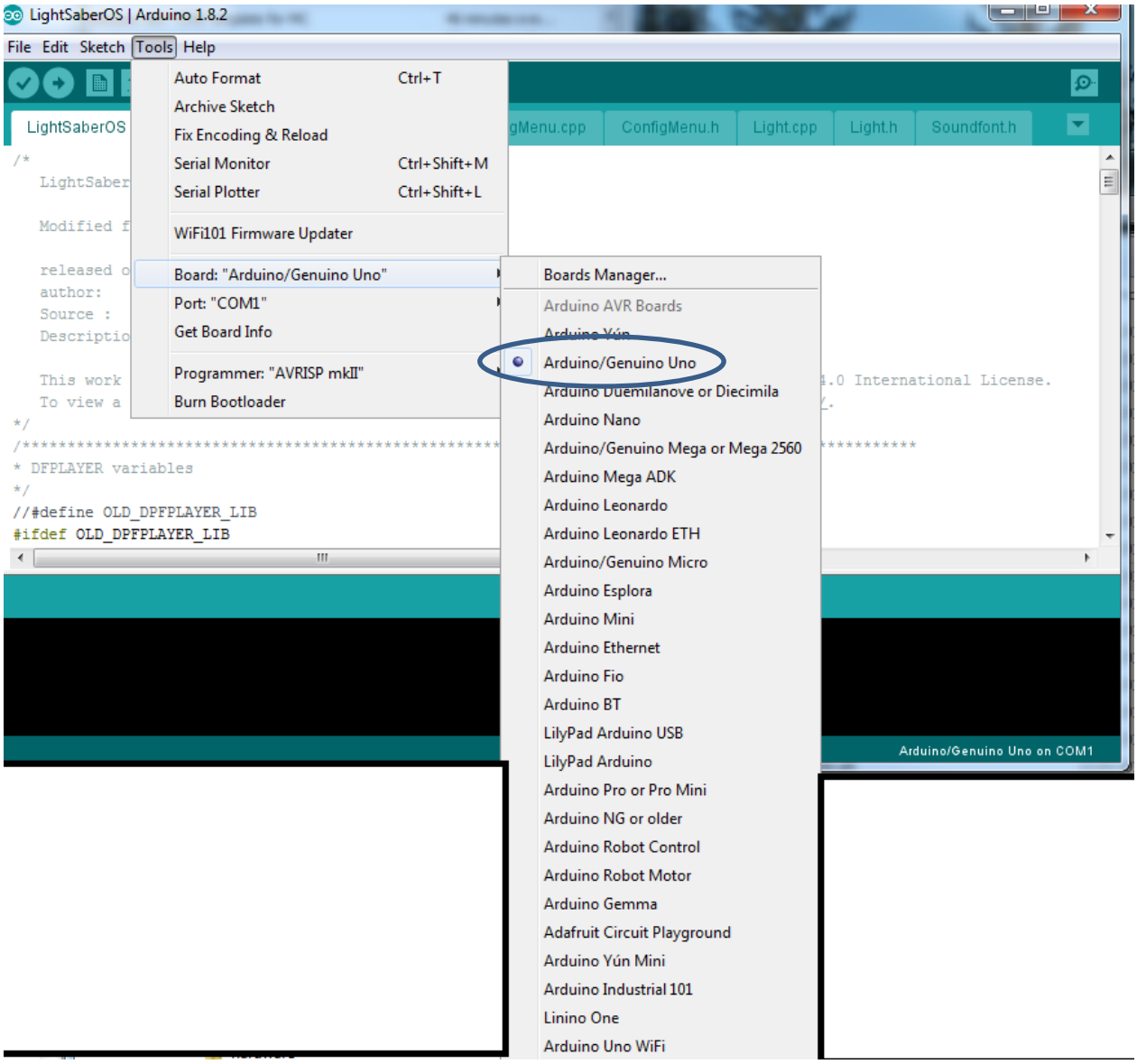
Once the Boards Manager starts, choose Arduino AVR Board, select the version 1.6.17 from the Select Version drop-down meeu and click on install.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



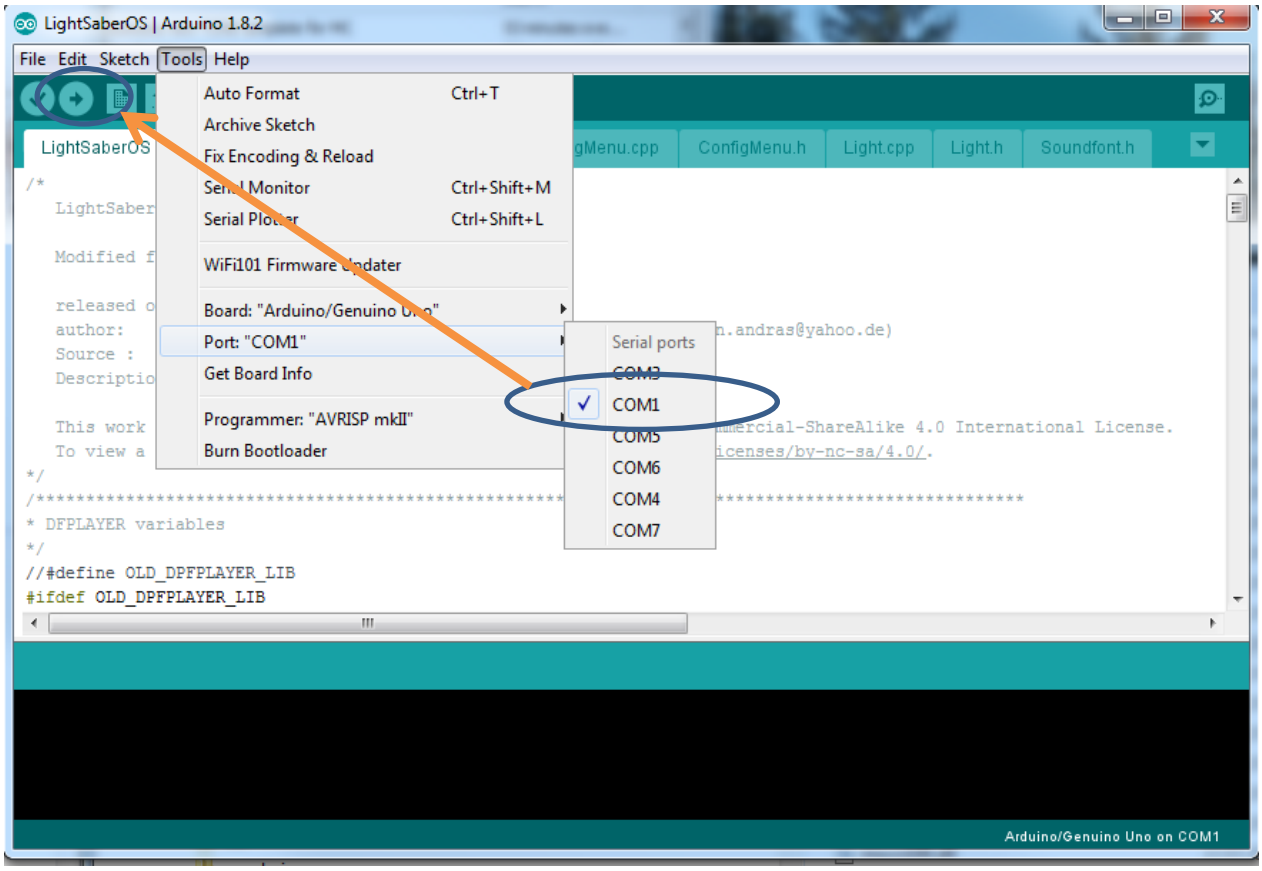
Step 11: Once installation of the Boards Manager is donw, specify your board by selecting Arduino/Genuino Uno from Tools/Boards (DIYino Stardusts boards are all configured as Arduino/Genuino Uno, as well as newer runs of DIYino Prime boards. Some older Runs from the Prime configured the boards as Arduino Nano, since the original Prime was based on the Nano. Later on it was decided to configure the boards unifromly as Uno to benefit from the additional 1.5k code space)

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



Step 12: Connect the Stardust board to your PC using an USB cable. Once done, a new COM port shall appear under Tools/Port, which corresponds to your board. Select this COM port and start uloading code with the Arrow button.

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.



9 Generic Saber Code

All Stardust V3 boards come pre-installed with a generic saber code which implements the most essential saber functions. The goal of the generic saber code is to allow users access to a working saber core right after receiving the board. To unlock the full set of features of the FX-SaberOS, the user can upload the code to replace the generic saber code.

The generic saber code is made to drive an RGB LED module using a single main button (see 5.1) to interact with the board.

After boot the saber starts in `Idle Mode`.

A short press of the main button will activate the saber (`Saber On Mode`) with a smooth ignition, using the main blade color. You can try clashes and swings (default threshold of 1000), the blade flickering is set to AudioTrackER.

Long press in `Saber On Mode` will retract the saber and bring it back to `Idle Mode`.

Long press in idle mode will enter the saber into `Config Mode`.

`Config Mode` consists of a simple set of items:

- Main blade color: choose from 6 preset color profiles by a short press on the main button
- Clash color: choose from 6 preset color profiles by a short press on the main button
- Volume control: short press on main button will increase the volume, after reaching highest value of 31, value will start again at 0 (mute)

ProtoWerkstatt 2018, All rigths reserved
Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.

The generic saber code uses a single sound font. To populate your SD card follow the instructions in 4.6 and copy the Config file folder and a single sound font folder from the sound bundle which can be found here: https://github.com/Protonerd/FX-SaberOS/blob/master/soundfonts/DIYino_Stardust_SPIFlash_image.7z

10 Quick-Start Guide to FX-SaberOS

FX-SaberOS (https://github.com/Protonerd/FX-SaberOS) is the world's most popular open source saber operating system, evolving out of the world's first such system, the LightSaberOS written by Sebastien Capou (neskweek) and Andras Kun (Protonerd). In this Chapter you can learn how to quickly set up the most important parameters of the saber code for a first upload.

First of all you need to upload so called sound fonts to your Stardust board. You can use your own sound fonts (see Chapter **Error! Reference source not found.** for uploading sound files and Chapter 4.7 for preparing your own sound fonts for upload to the Stardust board). There are excellent free-to use sound fonts from Darth PJs (Trinity Force Sabers) on GitHub (https://github.com/Protonerd/FX-SaberOS/blob/master/soundfonts/DIYino_Prime_SDCard_image.7z), the contents of the zip file can be directly moved to the SPI flash following the instructions in Chapter 4.6. Afterwards you can download and open FX-SaberOS in your Arduino IDE.

On the Config_HW.h tab, you can configure your hardware and peripherals:

Make sure you select Stardust as board:

```
//#define DIYINO_PRIME_V1
//#define DIYINO_STARDUST_V2
#define DIYINO_STARDUST_V3
```

You can select between single and 2-buttons modes by commenting out or leaving following line uncommented:

`#define SINGLEBUTTON` -> single button setup

`//#define SINGLEBUTTON` -> 2- buttons setup

Now it's time to define your blade type, leave your blade type uncommented and the other two commented. For instance to select a neopixel blade:

`//#define LEDSTRINGS` -> defaults to 6 segments, DIYino Prime recommended

`//#define STAR_LED` -> defaults to Red to LS1, Green to LS2 and Blue to LS3

`#define PIXELBLADE` -> neopixel blade

And finally if you are using neopixel blade, define the length of your blade in pixels:

```
#define NUMPIXELS 115
```

For basic setup that's all. If you want to understand how the code works, visit the GitHub Wiki. Have fun and May The Force Be With You! Always!

11 Related links

FX-SaberOS: https://github.com/Protonerd/FX-SaberOS

LSOS: https://github.com/neskweek/LightSaberOS

USaber: https://github.com/JakeS0ft/USaber

ProtoWerkstatt 2018, All rigths reserved

Author of this documentation is in no way affiliated, associated, licensed or endorsed by Disney or Lucasfilm Ltd., Industrial Light and Magic or any of their associates. All brands and trademarks listed are the exclusive property of their respective Owners.