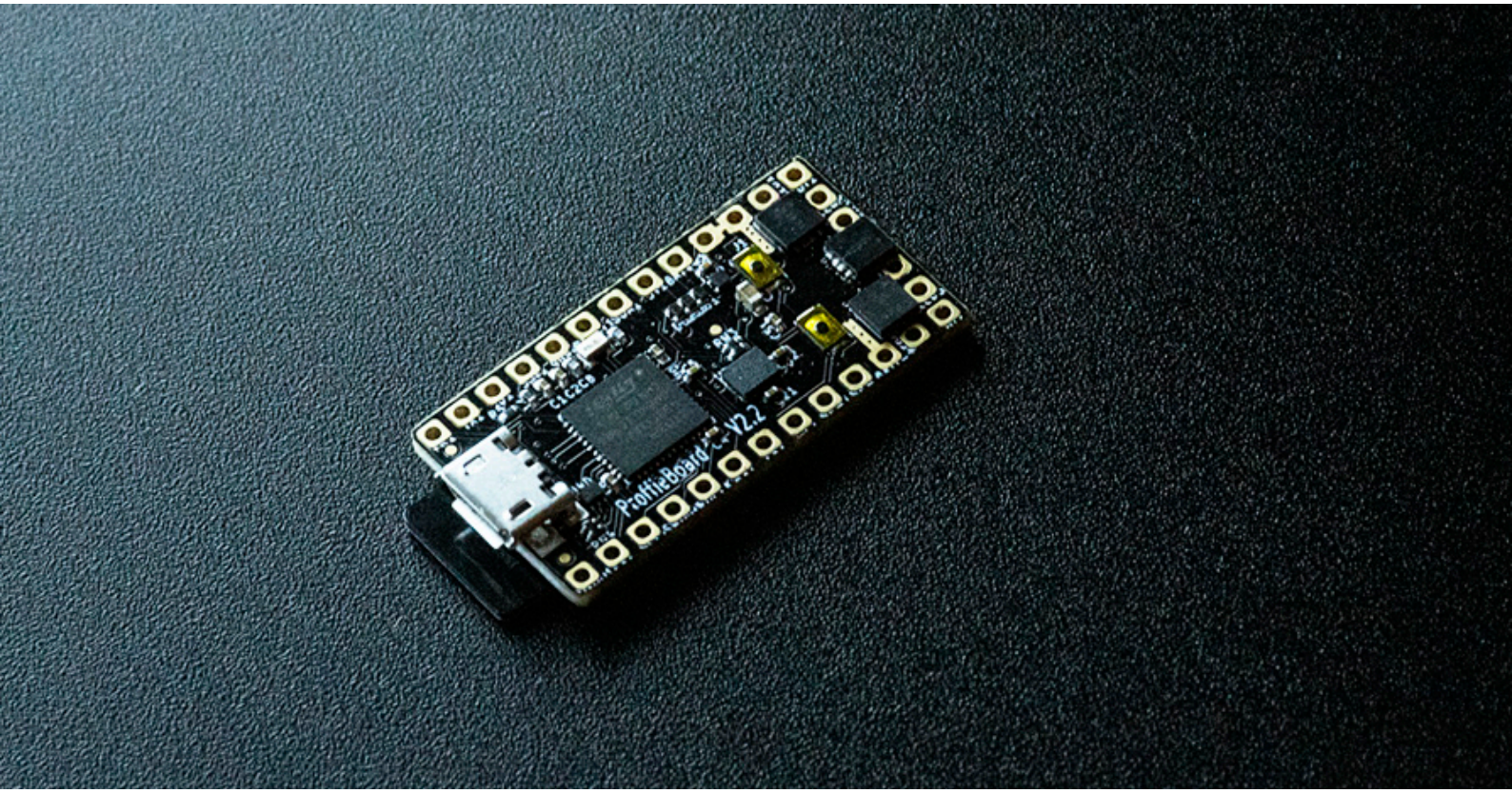
ProffieBoard v2

Open-Source advanced saber sound board



User Manual

by Dmitry Shtok and Fredrik Hubinette

2020

Contents

updated:
26.06.2021

INTRODUCTION

It really just started with a trip to Disneyland. I was really just disappointed with the cheap plastic lightsabers they had available. I had hoped to pick something more display-worthy, or at least in the "toys for grownups" category, but did not find anything. So when I got home, I went and ordered an FX "black series" Luke lightsaber, which looks quite nice, but the sound, light and interactivity was still pretty disappointing.

At this point I started to think about how I would make a lightsaber. I had already done things with neopixels before, so that was kind of a no-brainer for making a better blade, but I really wanted to do was to make the sound react fluidly to motion.

At this point I joined a bunch of forums and came across the NEC and Plecter boards, but there didn't seem to be a way to alter how they produced sounds, so I picked up a teensy and a PJRC prop shield and started building from there.

The Teensy 3.2 + PJRC prop + SD card reader + voltage booster + FETs I ended up with, was fairly large. Luckily, the Graflex lightsabers are also fairly large, so I purchased a Graflex 2.1 and barely managed to squeeze everything in there.

Around this time, I got kind of stuck with how to synthesize all the sounds a lightsaber makes, so I decided to implement support for Plecter and NEC sound fonts to get the saber I built make some sounds. There are some amazing sound fonts out there, but even so, the interactivity I craved was still missing.

Since I didn't really have a good idea for how to make that interactivity happen, I took on a different challenge instead: Make it smaller. For the TeensySaber V2, I decided to try to make my own circuit board. That meant integrating some components from the prop shield, the sd card reader, the voltage booster and the FETs into a single board. To make things interesting, I bought a Korbanth OWK, which has an inner diameter of 7/8 inches, and my goal was to fit everything in there. It took a while to do, but the result was the TeensySaber V2 board. The V2 fits really great inside an OWK, without cutting into the inner chassis parts, and was generally a great success, but the sound quality wasn't as good as I wanted it to be, so eventually I designed he TeensySaber V3, which is mostly the same as the V2, but uses a digital 3W amplifier.

As I was working on the TeensySaber V3, this guy Thexter showed up on a couple of forums, with some great videos showing off an algorithm for better swing sounds. Since this was what I wanted all along, I couldn't wait until he provided a description of his algorithm so that I could implement it. Lucky for me, he didn't mind describing his algorithm, so I implemented it. My implementation never really sounded as good as his videos though, but that's probably because I'm not really a font designer. Later, Thexter came back with an improved version, which is what we now call "SmoothSwing V2".

With SmoothSwing V3, TeensySaber V3 was getting some attention from people, but a lot of people still thought it was too big, since it's made out of two boards sandwiched together. The sandwiching also creates extra work for installers and extra complications for hobbyists, so it was time to try to put everything together into one board.

At first, I was thinking of using the same components that make up a Teensy to make the all-in-one board, but it turned out to be complicated and expensive. Instead I found another board called a "Butterfly", which had nearly identical capabilities and an already functional arduino plugin. Even better, the Butterfly was 100% open source (the teensy is only *mostly* open source).

I spent most of the Christmas vacation last year designing the Proffieboard, and it took another couple of months of testing to get a working prototype, but it's been a lot of fun.

- Fredrik Hubinette

2016

Read full interview on SaberSourcing:

[Proffieboard lightsaber controller developer Fredrik Hubinette interview]

FEATURES

Specifications and features:

– Dimensions: 17.8x33.2x4mm (+3.2mm with micro USB port and micro SD card)
– 100% Open-Source, you may add any feature you like (GPLv3)
– Power supply: 2.6-4.5 Volts, up to 10A per LED output 1-6; single Li-Ion 3.6-3.7V (low 2.6V, full 4.2V) battery recommended
– 1.3mA current drain in Idle Sleep mode = 3.5-4 months of shelf time if everything is setup properly (No Deep Sleep yet)
– Speaker: 4 ohm or 8 ohm, 2-5W (recommended)
– Unlimited amount of sound banks/fonts, supports regular (Plecter, NEC) and "Smoothswing" sound fonts
– Sound FX (WAV sound files): boot, blaster deflect, lockup, hum, swing, clash, drag, font, force, ingnition, retraction and more
– Light FX: blade flickering, pulsing, flash on clash, drag, stab, blaster deflect, lockup and other
– Music tracks (WAV sound files) playback in idle mode and saber sound effects background
– Micro SD card: 4-32Gb Class 4-10 by SanDisk brand recommended
– Support for remote control via bluetooth (with external bluetooth module addon)
– Speedy 32-bit processor for advanced features like sound filters, synthesizing and mp3 playback
– 3 Watts 5V sound amplifier, 16-bit digital output
– 4 Data signal outputs for neopixel LEDs/strips control
– Sample rate is 44kHz (default), 22kHz and 11kHz are supported and upsampled to 44kHz automatically
– Gapless playback, with 2.5ms cross-fade when you interrupt one sample to go to another
– Polyphonic playback, currently configured for up to 5 simultaneous samples
– "Smoothswing" motion-to-sound algorithm support
– PL9823 (RGB), WS2812B (RGB), SK6812 (RGB, WWA, RGBW) Neopixel support
– 1/2/3/4-color LED stars (Tri-Cree and Quad (also RGBW) LED modules)
– Segmented (6 segments + Flash string) classic string blades support
– Multi-blade support for dual and crossguard setups
– Blade LED type, Presets and Blade Styles selection by different values of a resistor (Blade ID functions)
– Crystal chamber support
– Power-level indicator with neopixel blade
– OLED PLI and FONT, animations display
– Real-time "Blade Detection"
– Real-time Color Changing, IR receiving support (since ProffieOS 3.9 firmware update)
– 3.3V (≈250mA available) pad for powering satellite devices like Bluetooth module or OLED display
– sound files upload to SD card via USB cable directly from PC (Mass Storage support)
– POV (persistance of vision) mode support
– Accent LEDs support (also implemented as additional "blades")
– Spoken error and low battery messages
– Easy and free firmware updates by user

Demonstration videos:

Link to the demonstration video by K-Sith
Link to the demonstration video by Megtooth Sith Sabers
Link to the demonstration video by Zimmer Labs
Link to the demonstration video by ShtokCustomWorx

1 HELPFUL LINKS

WHERE TO BUY

ProffieBoards:

TheSaberArmory (KR-sabers) UK store
JQ-sabers UK store
Korbanth USA store
ShtokCustomWorx RUS store – ENG
SaberBay Etsy USA store
Electronics123 USA store
Artekit EU store
ShtokCustomWorx в VK – RUS

Other parts links:

RGB Neopixel strips (they are SK6812, though sellers list them as WS2812b)
WWA (White/White/Amber) Neopixel strips SK6812 Source 1
WWA (White/White/Amber) Neopixel strips SK6812 Source 2
Individual Neopixel LEDs
Neopixel strips/connectors/other supplies (UK) – TheSaberArmory
Tri-Cree high power LEDs (Canada/USA) – TheCustomSaberShop
Tri-Cree high power LEDs (UK) – TheSaberArmory
Various Accent LEDs (UK) – TheSaberArmory
Various Batteries (UK) – TheSaberArmory
Protected KeepPower 18650 10A 3500mAh battery
Protected KeepPower 18650 15A 3120mAh battery
Unprotected Vapcell 21700 15A 5000mAh battery – requires external PCM
Unprotected KeepPower 26650 15A 6000mAh battery – requires external PCM
15A Protection Circuit Module (PCM) (aliexpress)
18650 Protected Battery holder
High Power 1.3mm Recharge Port
Recharge Ports (UK) – TheSaberArmory
High Power Kill Switch
Various Switches (UK) – TheSaberArmory
SCW NPXL™ pixelblade pogo connector
GX16 Neopixel/string blade connectors
Various speakers (UK) – TheSaberArmory
Various speakers (UK) – JQ-sabers
Various speakers (Canada/USA) – TheCustomSaberShop
3W speakers
FSC-BT630 bluetooth module (USA store)
FSC-BT909 bluetooth module (USA store)
FSC-BT630 bluetooth module (UK store)
FSC-BT909 bluetooth module (UK store)

3D-printed chassis links:

ShtokCustomWorx on Shapeways
GOTH-3Designs on Shapeways

1 HELPFUL LINKS

Tutorials and instructions

Video tutorials by Megtooth Sith Sabers:

Video tutorials by Megtooth Sith Sabers on youtube
LED Resistor Calculator

For more information please check these links:

ProffieBoard v2.2 sound board instructions
"Blade style sharing" - here you can find and share custom blade styles
Fett263 Blade Style Library
Web Blade Style Editor

Here you can buy legacy and SmoothSwing sound fonts to use with Proffieboard:
– SaberFont.com
– Kyberphonic Fonts

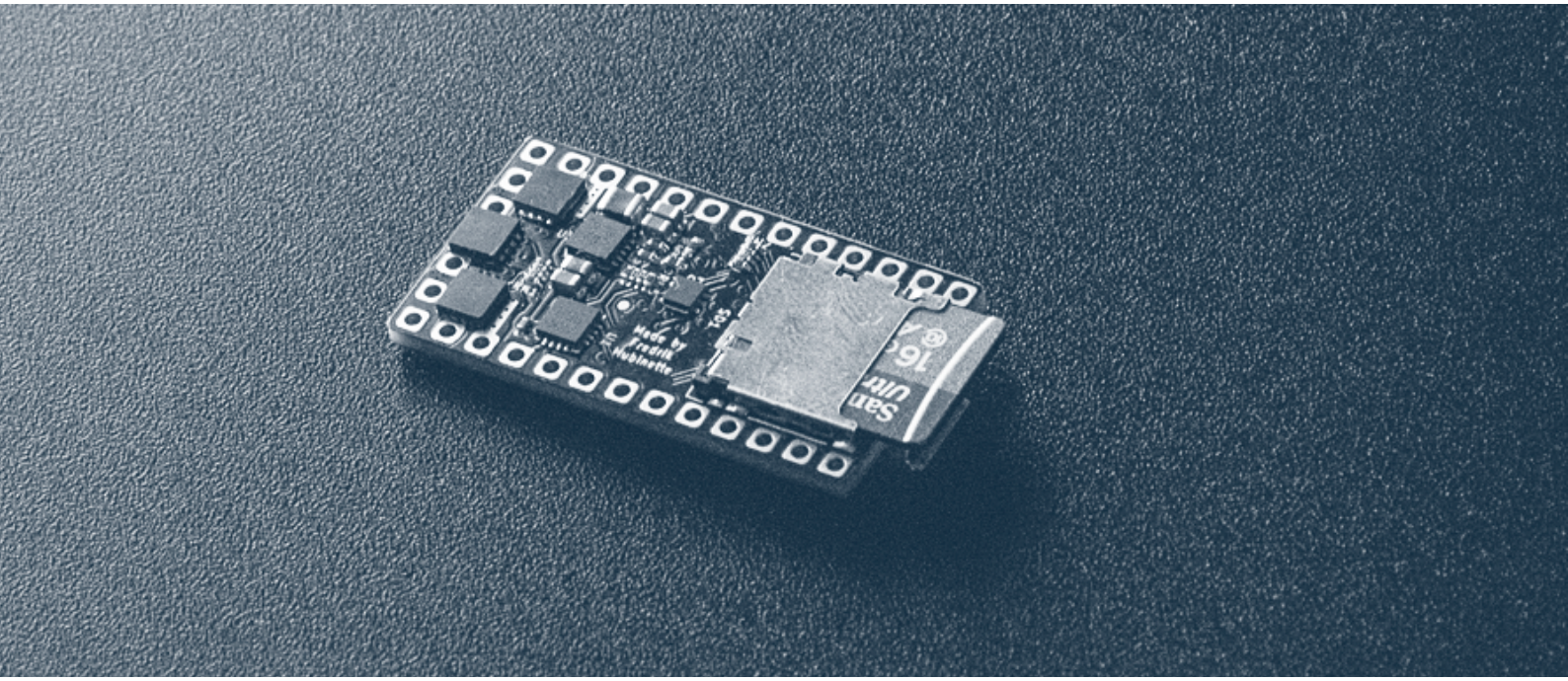
ProffieOS/ProffieBoard/TeensySaber wiki on GitHub
Profezzorn's Lab on The Crucible forums
Profezzorn's Lab on FX-sabers forums
Ask your questions, share your builds in a facebook group

>>>Get latest ProffieOS firmware here<<<

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

What's needed

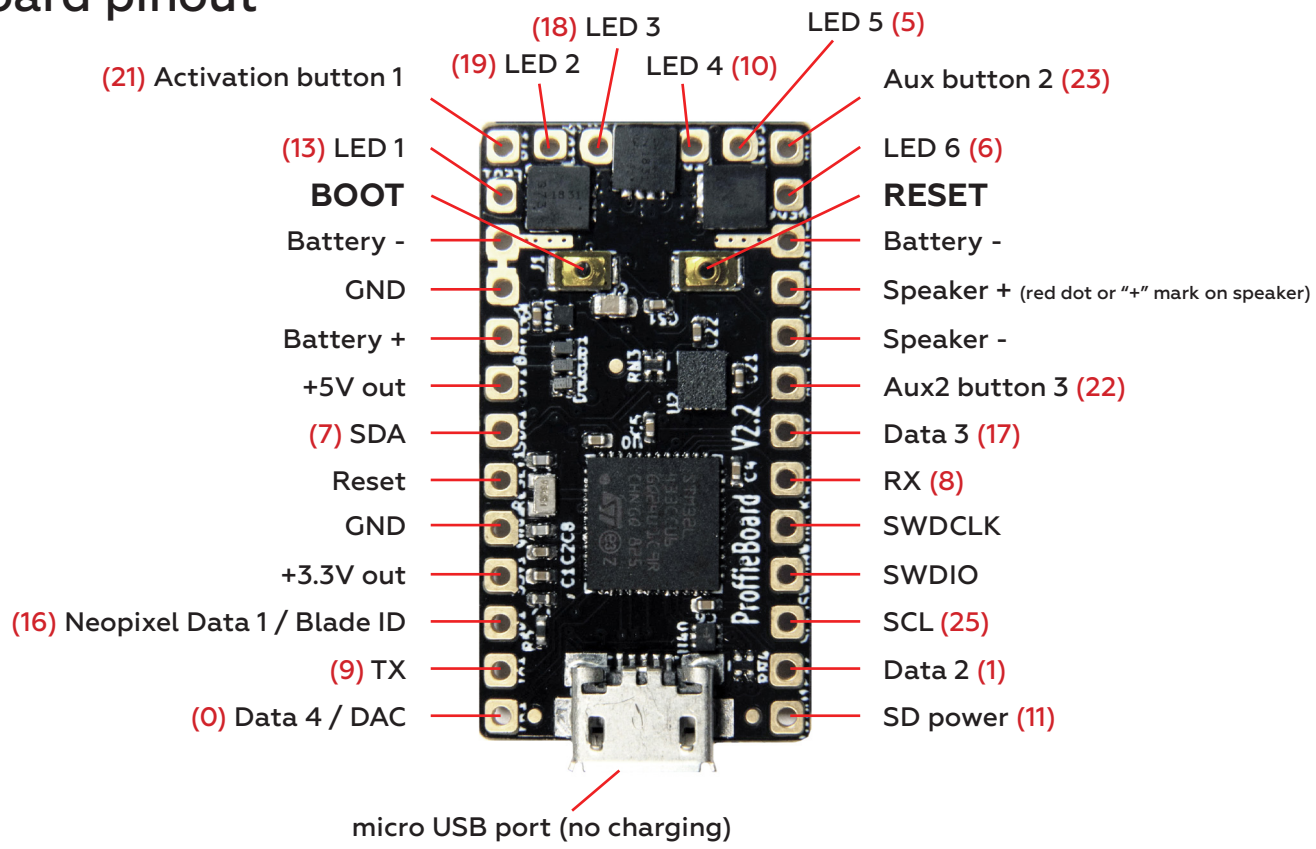


– ProffieBoard v2.2
– micro SD-card (see page 33 for recommendations)
– a USB micro SD-card reader (to load sound files from PC to micro SD card)
– micro USB data transfer cable **(CABLES, THAT SUPPORT ONLY CHARGING, WON'T WORK!)**
– wires of different gauges (34-20 AWG) (PTFE coated copper stranded wires recommended), heat shrink
– ESD safe soldering station, solder wire, flux etc..
– pliers, helping hands etc..
– isopropyl alcohol to clean pads before soldering (helps solder to stick better)
– Digital Multimeter **(VERY USEFUL!)**
– computer running Windows, Linux or Mac OS with internet access
– 3.7V Li-Ion Protected rechargeable battery, switches, recharge port, speaker, LEDs, resistors, chassis etc..
– Smart Li-Ion CC-CV (Constant Current - Constant Voltage mode) battery charger for 3.7V (4.2V) cells
– patience...

2 PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Board pinout



Battery + (BATT+) – 2.6 to 4.5 volt input

Battery - (BATT-) – negative pad for LEDs, connected with GND by default **(make sure to solder the battery Negative wire always to BATT- pad but NOT to GND pad though!)**. Both BATT- pads are internally connected

GND – ground for electronics except high power LEDs. Note that there are two GND pads on the board that are internally connected

Speaker +/- – hooks up to speaker

Activation 1 / Aux 2 / Aux2 button 3 – hook up to closing buttons, or potentially touch buttons

Neopixel Data 1 / Blade ID – "Blade ID" resistor sense, and neopixel Data 1 output (already has a 470 Ohm resistor on the line)

Data 2, 3 – additional neopixel Data outputs, or free for other purposes

Data 4 – neopixel Data 4 output, free, or audio DAC output

LED 1, 2, 3, 4, 5, 6 – hooks up to negative side of LED (positive side of LED hooks up directly to battery). These pads can handle up to 30 volts

SDA, SCL – these pins are used to wire OLED display or to communicate with the gyro and accelerometer chip

RX, TX – these pins are used for wiring a bluetooth module for wireless control

SWDCLK, SWDIO – can be hooked up to a ST-LINK device and lets you debug programs running on the Proffieboard

SD power – FET-controlled +3.3V pad. Can be used to power down a Bluetooth module and OLED display in low-power mode

+5V – generated by the Proffieboard, normally it's only ON when sound is playing

+3.3V – generated by the Proffieboard for powering OLED display, Bluetooth module or some accent leds (≈250mA available)

BOOT, RESET buttons – buttons to put the Proffieboard in bootloader mode if uploading doesn't work

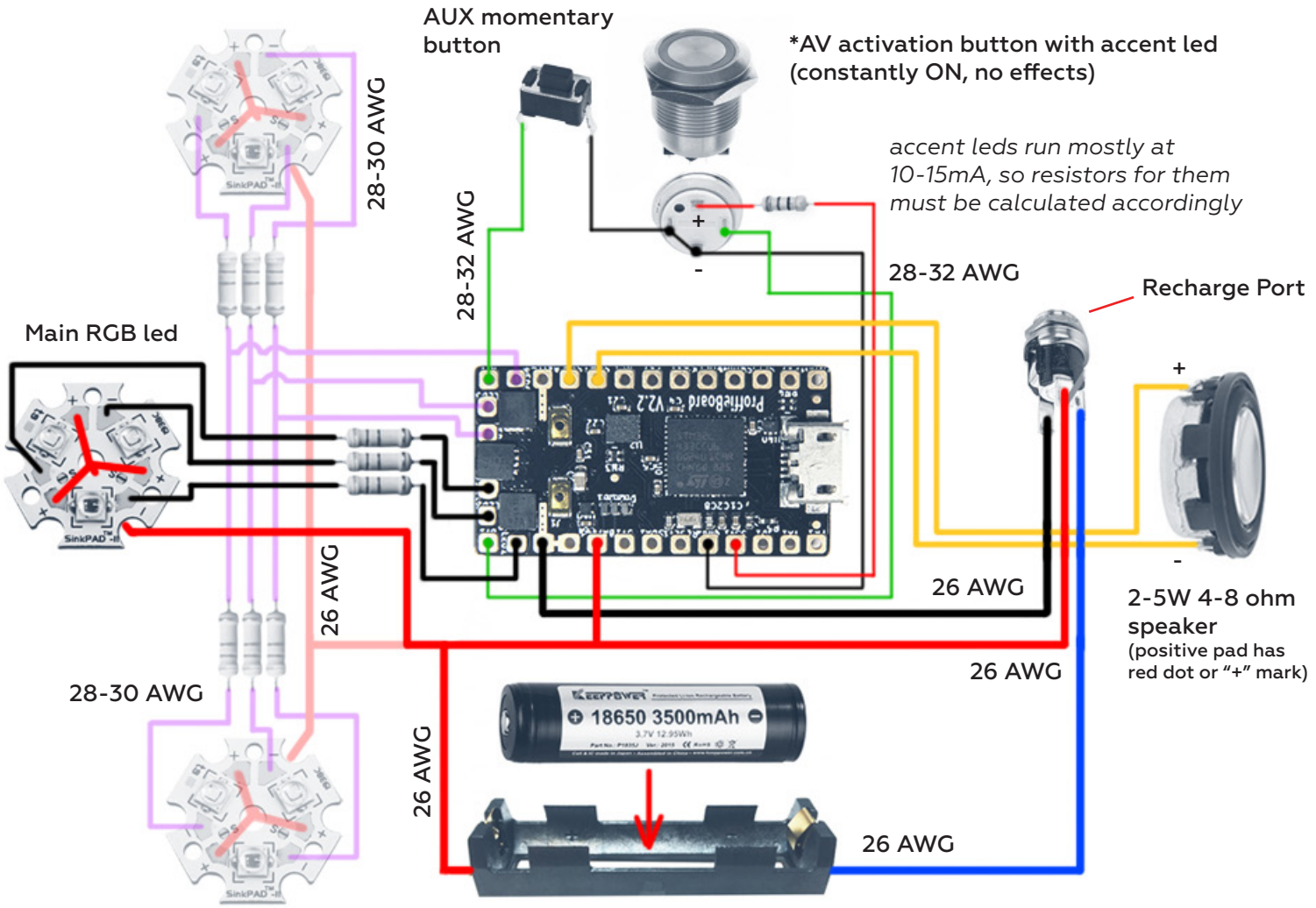
micro USB port – micro USB port used only for firmware upload and can be used for sound files upload to SD card

(THIS PORT ISN'T USED FOR CHARGING THE BATTERY, IT'S ONLY FOR DATA TRANSFER!)

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Basic Tri-Cree wiring diagram (In-hilt LED)



*

In case no additional high power leds are needed, LED channels 4, 5, 6 can be used for 3 controllable (programmable for different effects) accent leds. So AV switch led can be wired to one of these channels. Accent leds also can be wired to Data pads 1-4, please see "Accent LEDs wiring and setup" page.

[LED Resistor Calculator]

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Basic Tri-Cree wiring (In-hilt LED) “config.h” file setup

Use a given or build your wiring diagram on **THIS PAGE**, copy the code to some default *config.h* file and **Save** it under new Name in ...*ProffieOS**config* folder. Follow the instructions on page 29-30 to upload it to the board.

```
“proffieboard_v2_config.h”
NUM_BLADES 2
NUM_BUTTONS 2
VOLUME 1000
CLASH_THRESHOLD_G 1.0
StyleNormalPtr<CYAN, WHITE, 300, 800>()
StyleNormalPtr<CYAN, WHITE, 300, 800>()

CreeXPE2RedTemplate<1000>,
CreeXPE2GreenTemplate<0>,
CreeXPE2BlueTemplate<240>,
NoLED
```

where *1000* is 1 Ohm resistor, *0* is no resistor, *240* is 0.24 Ohm resistor, *NoLED* – no 4th led used

— **ProffieBoard v2 config setup**
— **number of “blades” used**
— **number of buttons used** *(1-3)*
— **Volume level** *(0-3000)*
— **Clash sensitivity** *(lower = more sensitive, higher = less)*
— **“Blade 1” style**
— **“Blade 2” style** *(in case only 1 blade is used, you don’t need this line)*
— **LED configuration** *(use these XP-E2 LED templates to define your LED. If other LED resistors are used, change these values to match: Ohm*1000=<value>)*

```
#ifdef CONFIG_TOP
#include “proffieboard_v2_config.h”
#define NUM_BLADES 2
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

 {“TeensySF”, “tracks/venus.wav”,
  StyleNormalPtr<CYAN, WHITE, 300, 800>(),
  StyleNormalPtr<CYAN, WHITE, 300, 800>(), “cyan”},

 {“SmthJedi”, “tracks/mars.wav”,
  StylePtr<InOutSparkTip<EASYBLADE(BLUE, WHITE), 300, 800> >(),
  StylePtr<InOutSparkTip<EASYBLADE(BLUE, WHITE), 300, 800> >(), “blue”},

 {“SmthGrey”, “tracks/mercury.wav”,
  StyleNormalPtr<RED, WHITE, 300, 800>(),
  StyleNormalPtr<RED, WHITE, 300, 800>(), “red”},

};
BladeConfig blades[] = {

{ 0,
  SimpleBladePtr<CreeXPE2RedTemplate<1000>, CreeXPE2GreenTemplate<0>, CreeXPE2BlueTemplate<240>, NoLED, bladePowerPin1, bladePowerPin2, bladePowerPin3, -1>(),
  SimpleBladePtr<CreeXPE2RedTemplate<1000>, CreeXPE2GreenTemplate<0>, CreeXPE2BlueTemplate<240>, NoLED, bladePowerPin4, bladePowerPin5, bladePowerPin6, -1>(),

CONFIGARRAY(presets) },

};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, “pow”);
Button AuxButton(BUTTON_AUX, auxPin, “aux”);
#endif
```

— Preset 1

— Preset 2, etc....

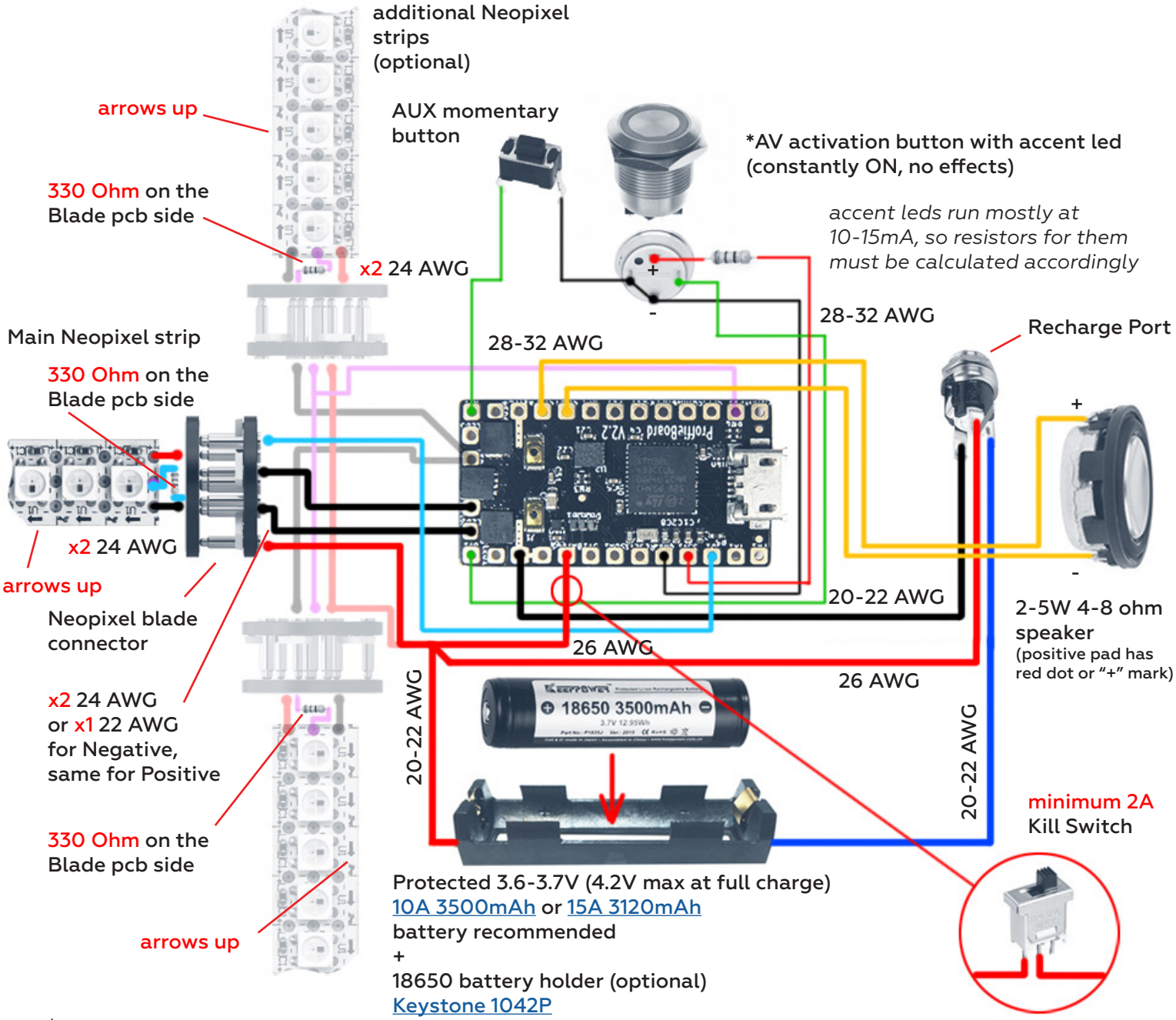
— LED 1 configuration

— LED 2 configuration

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Basic Neopixel wiring diagram



*

In case no additional Neopixel strips are needed, LED channels 4, 5, 6 can be used for 3 controllable (programmable for different effects) accent leds. So AV switch led can be wired to one of these channels. Accent leds also can be wired to Data pads 2-4, please see "Accent LEDs wiring and setup" page.

Recommended power wire gauges (22 AWG) are given for 2-strip blade. For 3-strip blade you gonna need at least 20 AWG wires.

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Basic Neopixel wiring
“config.h” file setup

Use a given or build your wiring diagram on **THIS PAGE**, copy the code to some default *config.h* file and **Save** it under new Name in ...*ProffieOS**config* folder. Follow the instructions on page 29-30 to upload it to the board.

```
“proffieboard_v2_config.h”
NUM_BLADES 2
NUM_BUTTONS 2
VOLUME 1000
CLASH_THRESHOLD_G 1.0
..any blade style..
IgnitionDelay<300, ..any blade style..>
```

— ProffieBoard v2 config setup
— number of “blades” used
— number of buttons used *(1-3)*
— Volume level *(0-3000)*
— Clash sensitivity *(lower = more sensitive, higher = less)*
— “Blade 1” style *(main blade)*
— “Blade 2” style *(CG blades with IgnitionDelay 300)*

```
WS281XBladePtr<..., bladePin,..., PowerPINS<bladePowerPin2, bladePowerPin3> >()
WS281XBladePtr<..., blade2Pin,..., PowerPINS<bladePowerPin4> >()
```

— **strip configuration** *(defines how many pixels it has and to which Data and LED output pad it’s wired)*

```
#ifdef CONFIG_TOP
#include “proffieboard_v2_config.h”
#define NUM_BLADES 2
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

{ “TeensySF”, “tracks/venus.wav”,
  StyleNormalPtr<RED, WHITE, 200, 300>(),
  StylePtr<IgnitionDelay<300,StyleNormalPtr<RED, WHITE, 200, 300>>>() },

{ “SmthJedi”, “tracks/mars.wav”,
  StyleNormalPtr<RED, WHITE, 200, 300>(),
  StylePtr<IgnitionDelay<300,StyleNormalPtr<RED, WHITE, 200, 300>>>()},

 { “SmthGrey”, “tracks/mercury.wav”,
   StyleRainbowPtr<300, 800>(),
   StylePtr<IgnitionDelay<300,StyleNormalPtr<RED, WHITE, 200, 300>>>()},

};
BladeConfig blades[] = {

 { 0,
  WS281XBladePtr<144, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
  WS281XBladePtr<26, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >(),
CONFIGARRAY(presets) },

};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, “pow”);
Button AuxButton(BUTTON_AUX, auxPin, “aux”);
#endif
```

any blade style

— Preset 1

— Preset 2, etc....

strips (blade) 1 configuration

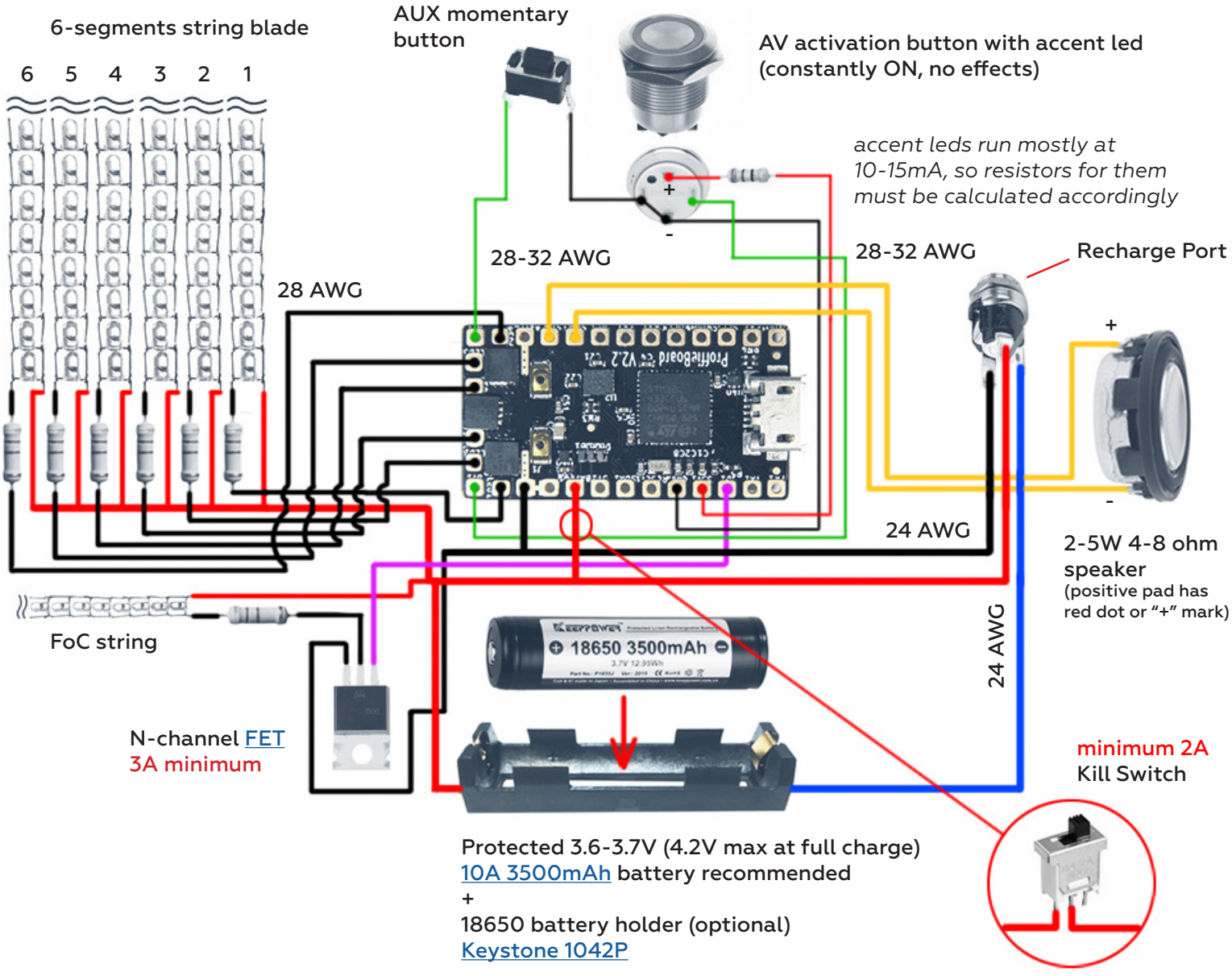
strips (blade) 2 configuration

adjust this number to match your strips leds amount

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Basic Segmented string wiring diagram



Calculate resistors for each led segment of the blade string depending on which leds are used. 5mm leds have max drive current around 25-30mA per led, when 10mm leds can be 100mA and 200mA per led. So pay attention to your led max current and Forward Voltage (Vf) when calculating a segment resistor resistence as well as its wattage. Also choose wire gauges accordingly to meet segments and total blade max current draw level.

[LED Resistor Calculator]

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Basic Segmented string wiring “config.h” file setup

Use a given or build your wiring diagram on **THIS PAGE**, copy the code to some default *config.h* file and **Save** it under new Name in ...*ProffieOS**config* folder. Follow the instructions on page 29-30 to upload it to the board.

```
“proffieboard_v2_config.h”
NUM_BLADES 1
NUM_BUTTONS 2
VOLUME 1000
CLASH_THRESHOLD_G 1.0
StyleNormalPtr<CYAN, WHITE, 300, 800>()
<Blue3mmLED, BladePin, White3mmLED>
```

— **ProffieBoard v2 config setup**
— **number of “blades” used**
— **number of buttons used** *(1-3)*
— **Volume level** *(0-3000)*
— **Clash sensitivity** *(lower = more sensitive, higher = less)*
— **Blade style**
— **LED string configuration**
(here you mention the color and type of leds used in the main blade string segments and FoC string. BladePin is the FoC signal pin (Blade ID pin))

```
#ifdef CONFIG_TOP
#include “proffieboard_v2_config.h”
#define NUM_BLADES 1
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

{ “TeensySF”, “tracks/venus.wav”,
 StyleNormalPtr<CYAN, WHITE, 300, 800>(), “Ignition” }

};
BladeConfig blades[] = {

 { 0,
  StringBladePtr<Blue3mmLED, BladePin, White3mmLED>(),
CONFIGARRAY(presets) },

};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, “pow”);
Button AuxButton(BUTTON_AUX, auxPin, “aux”);
#endif
```

— **Preset**

— **LED string configuration**

PROFFIEBOARD INSTRUCTIONS

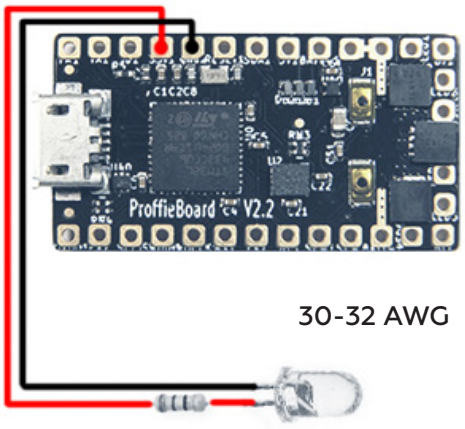
1 WIRING DIAGRAMS

Accent LEDs wiring diagram (optional)

Accent LEDs work with ProffieBoard as additional "blades" when powered by LED outputs 1-6 or Data pads 1-4 as PWM. So they can have any effect that blade can have. If no effects needed, accent led can be powered just by a 3.3V output pad (power-on led indication).

a)

"Power-on" indication accent leds (no effects)

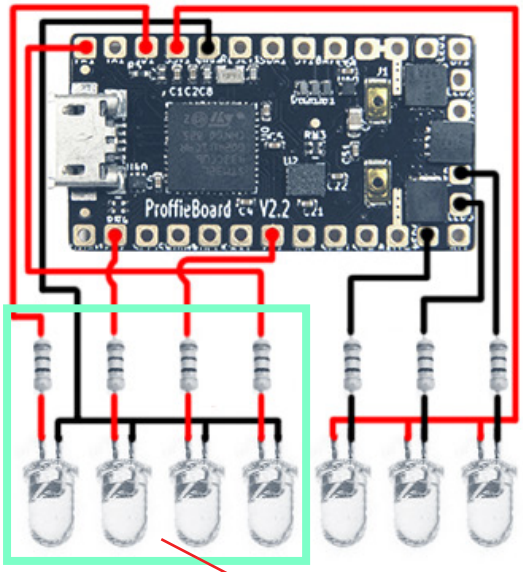


Recommended resistors to use for accent leds at 3.3V power source and 15mA drive:

- 100 Ohm for Red
- 13 Ohm for Green
- 13 Ohm for Blue
- 100 Ohm for Yellow
- 20 Ohm for White

b)

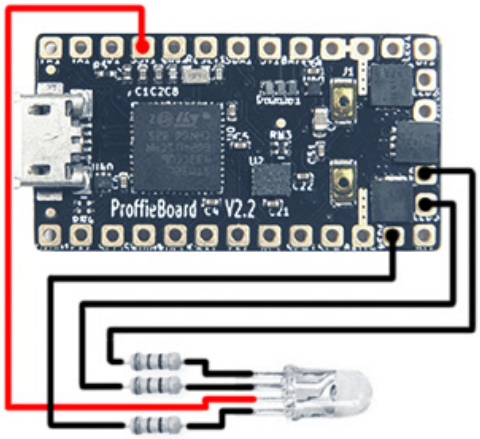
7 accent leds with independent effects



with Neopixel blade setup these 4 outputs don't work for regular accent leds!

c)

RGB accent led



RGB led common-anode

b)

8 "blades": 1 main and 7 accent leds

```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 8
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

 { "TeensySF", "tracks/venus.wav",
   StyleNormalPtr<CYAN, WHITE, 300, 800>(),
   StyleNormalPtr<RED, WHITE, 300, 800>(),
   StyleNormalPtr<RED, WHITE, 300, 800>(),
   StyleNormalPtr<BLUE, WHITE, 300, 800>(),
   StyleNormalPtr<GREEN, WHITE, 300, 800>(),
   StyleNormalPtr<GREEN, WHITE, 300, 800>(),
   StyleNormalPtr<BLUE, WHITE, 300, 800>(),
   StyleNormalPtr<RED, WHITE, 300, 800>(), "cyan"},

};
BladeConfig blades[] = {
 { 0,
   SimpleBladePtr<CreeXPE2RedTemplate<1000>, CreeXPE2GreenTemplate<0>, CreeXPE2BlueTemplate<240>, NoLED, bladePowerPin1,
   bladePowerPin2, bladePowerPin3, -1>(),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin4, -1, -1, -1>(),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin5, -1, -1, -1>(),
   SimpleBladePtr<CH3LED, NoLED, NoLED, NoLED, bladePowerPin6, -1, -1, -1>(),
   SimpleBladePtr<CH2LED, NoLED, NoLED, NoLED, bladePin, -1, -1, -1>(),
   SimpleBladePtr<CH2LED, NoLED, NoLED, NoLED, blade2Pin, -1, -1, -1>(),
   SimpleBladePtr<CH3LED, NoLED, NoLED, NoLED, blade3Pin, -1, -1, -1>(),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, blade4Pin, -1, -1, -1>(),
CONFIGARRAY(presets) },
};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

main blade style (effects)

7 accent leds style (effects)

main blade configuration

CH1LED for Red
CH2LED for Green
CH3LED for Blue

7 accent leds configurations

c)

2 "blades": 1 main and 1 RGB accent led

```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 2
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

 { "TeensySF", "tracks/venus.wav",
   StyleNormalPtr<CYAN, WHITE, 300, 800>(),
   StyleNormalPtr<CYAN, WHITE, 300, 800>(), "cyan"},

};
BladeConfig blades[] = {
 { 0,
   WS281XBladePtr<144, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   SimpleBladePtr<CH1LED, CH2LED, CH3LED, NoLED, bladePowerPin4, bladePowerPin5, bladePowerPin6, -1>(),
CONFIGARRAY(presets) },
};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

main blade style (effects)

accent led style (effects)

main blade configuration

RGB accent led configuration

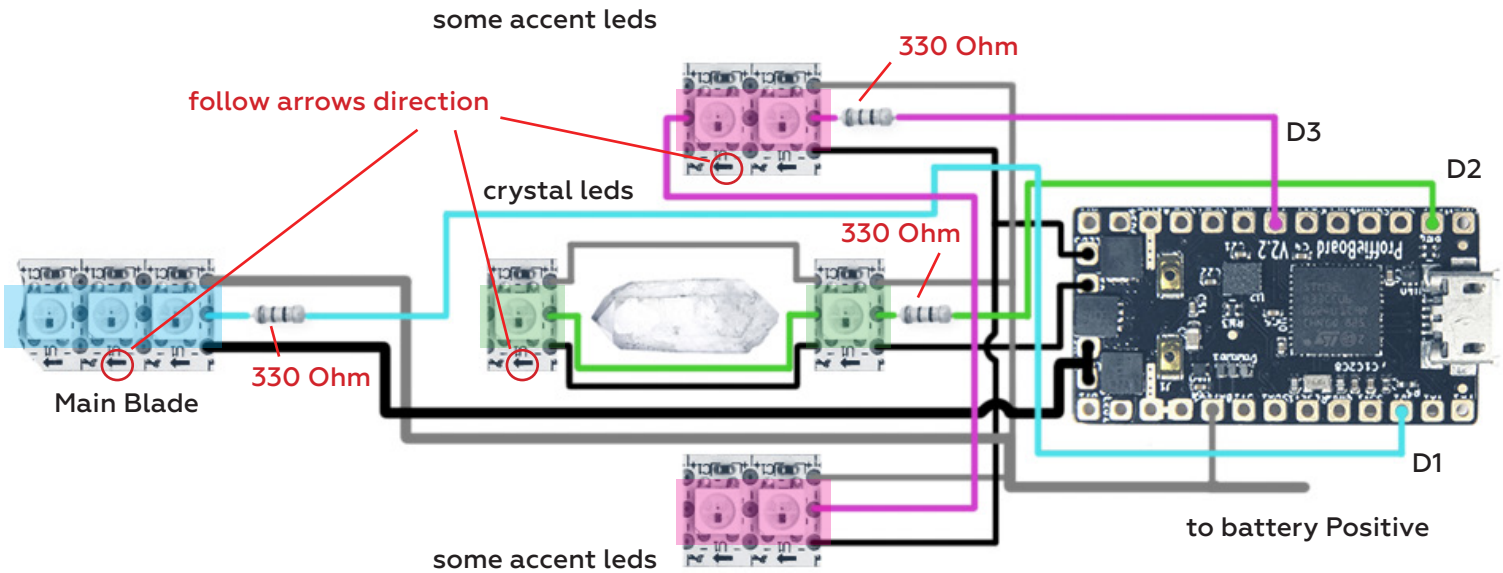
PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Neopixel Accent LEDs wiring diagram (optional)

With Neopixel setup additional neopixel leds or arrays can be used as accent leds. There are 2 ways to wire them: using additional Data pins 2, 3, 4 or "Sub-blades" wiring with just 1 Data output pin. Same way Neopixel connectors with on-board pixels can be wired.

Option 1 – with extra Data pins




```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 3
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

 { "TeensySF", "tracks/venus.wav",
  StyleNormalPtr<CYAN, WHITE, 300, 800>(),
  StyleNormalPtr<RED, WHITE, 300, 800>(),
  StyleNormalPtr<BLUE, WHITE, 300, 800>(),
"cyan" }

};
BladeConfig blades[] = {

 { 0,
  WS281XBladePtr<144, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
  WS281XBladePtr<4, blade3Pin, Color8::GRB, PowerPINS<bladePowerPin5> >(),
  WS281XBladePtr<2, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >(),
CONFIGARRAY(presets) },

};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

3 "blades": 1 main, 1 accent leds array and 1 crystal leds array

main blade style (effects)

accent leds blade style (effects)

crystal leds blade style (effects)

main blade: 144 leds, Data pin 1

accent leds "blade": 4 leds, Data pin 3

crystal leds "blade": 2 leds, Data pin 2

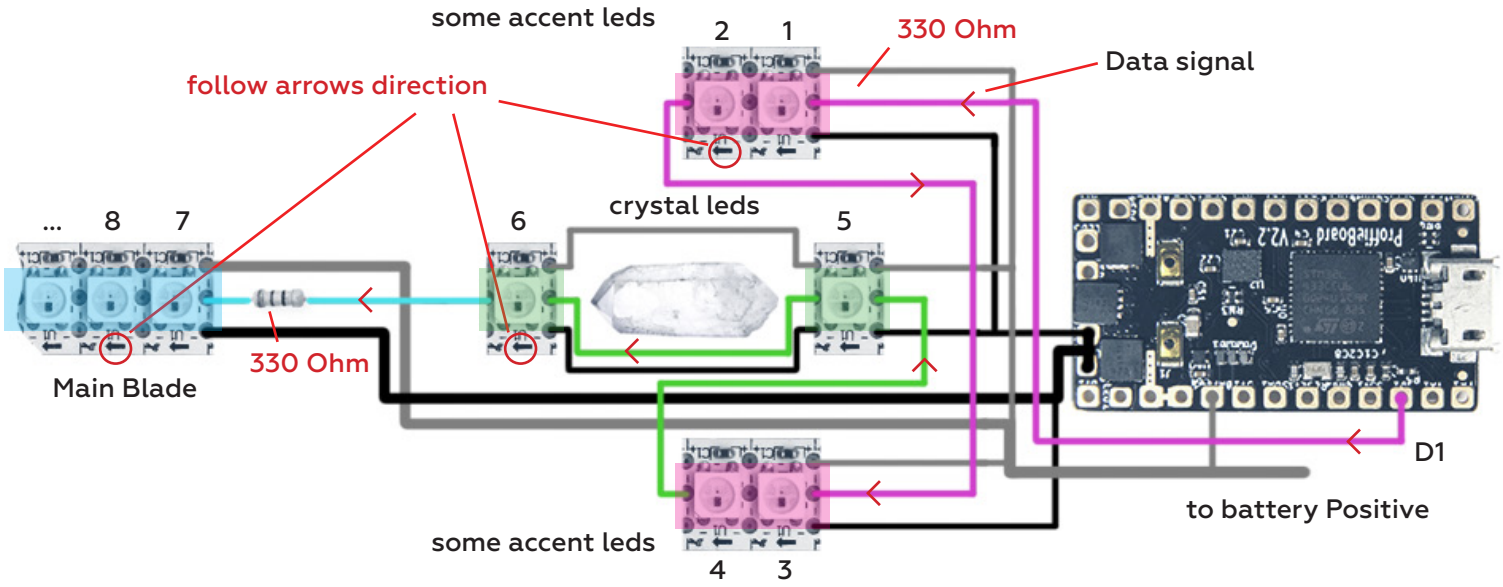
PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Neopixel Accent LEDs wiring diagram (optional)

With this setup a single array of neopixel leds is separated into a couple of sub-blades with their own style configuration and behaviour. This is really usefull, when you want to use only one Data pin. More about "Sub-blades" on ProffieOS wiki page

Option 2 – with "Sub-blades"




```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 3
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 146;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

 { "TeensySF", "tracks/venus.wav",
  StyleNormalPtr<CYAN, WHITE, 300, 800>(),
  StyleNormalPtr<RED, WHITE, 300, 800>(),
  StyleNormalPtr<BLUE, WHITE, 300, 800>(),
"cyan" }

};
BladeConfig blades[] = {

{ 0,
  SubBlade (0, 3, WS281XBladePtr<146, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >() ),
  SubBlade (4, 5, NULL),
  SubBlade (6, 145, NULL),
CONFIGARRAY(presets) },

};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

3 "blades": 1 main, 1 accent leds array and 1 crystal leds array

update default 144 to a higher total value if you get *all accent leds + Main blade > 144. Example: update to* **146** *if you have 2 crystal leds + 4 accent leds + 140 Main blade leds =* **146**

accent leds blade style (effects)

crystal leds blade style (effects)

main blade style (effects)

146 leds total used

accent leds sub-blade: 4 leds (1-4); *but from 0 to 3 in the code*

crystal leds sub-blade: 2 leds (5-6); *but from 4 to 5 in the code*

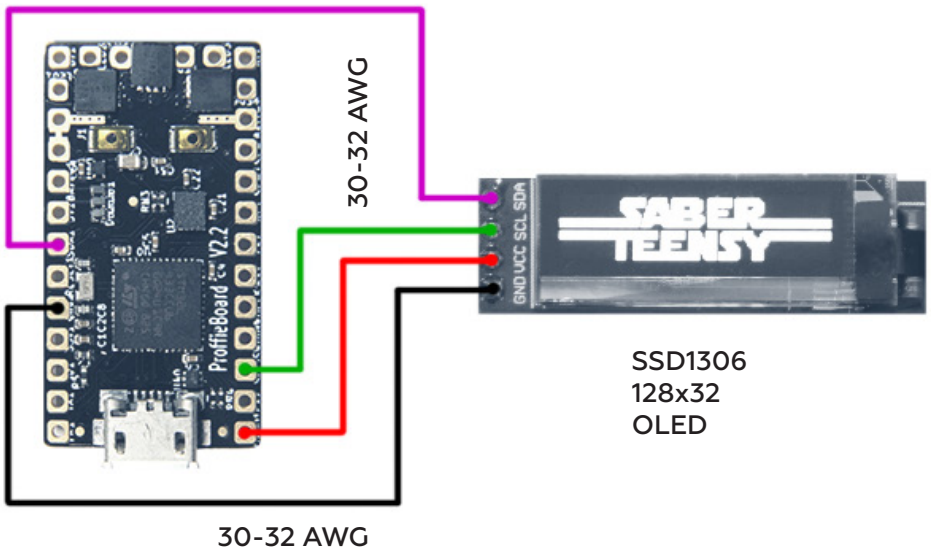
main blade sub-blade: 140 leds (7-146); *but from 6 to 145 in the code*

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

OLED display wiring diagram (optional)

SSD1306 128x32 pixels OLED display allows to show battery level, current preset name, play different animations and even simple games. It can be wired to any blade configuration and requires just one additional line in the code to work. You can get monochrome display in white or blue color.



SSD1306 – with blue or white display color select
SSD1306 – cheaper price
SSD1306 just screen – blue or white select

```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 1
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#define ENABLE_SSD1306
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {
  { "TeensySF", "tracks/venus.wav",
    StyleNormalPtr<CYAN, WHITE, 300, 800>(), "cyan"},
  { "SmthJedi", "tracks/mars.wav",
    StylePtr<InOutSparkTip<EASYBLADE(BLUE, WHITE), 300, 800> >(), "blue"},
  { "SmthGrey", "tracks/mercury.wav",
    StyleFirePtr<RED, YELLOW>(), "fire"},
  { "SmthFuzz", "tracks/uranus.wav",
    StyleNormalPtr<RED, WHITE, 300, 800>(), "red"},
  { "RgueCmdr", "tracks/venus.wav",
    StyleFirePtr<BLUE, CYAN>(), "blue fire"},
```

add this line to enable OLED display

display shows a preset name written in these quotes "..."

2 PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

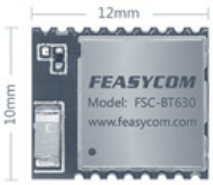
Bluetooth module wiring diagram (optional)

Bluetooth modules **FSC-BT630** and **FSC-BT909** from Feasycom are recommended over other modules on the market because of the best pcb size, quality, functionality and price point.

FSC-BT630 has same functionality as **FSC-BT909** but twice smaller size, lower signal strength and only BLE protocol support (no SPP).

Both modules are recommended for use with ***ForceSync*** mobile app (currently in development) from ShtokCustomWorx.

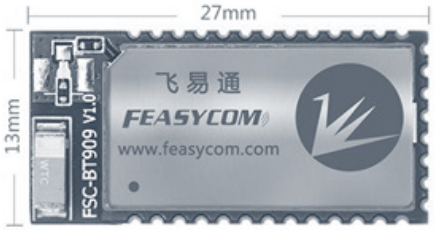
FSC-BT630



Features:

- Nordic nRF52832 chipset
- Bluetooth 5.0/4.2/4.1/4.0 support
- Class 1.5 (signal power up to +4dBm)
- Profiles including GAP, ATT/GATT, SMP, L2CAP
- Built-in ceramic chip antenna, external antenna optional
- Current consumption: 7mA connected, 10mA max
- Connection status LED indication
- PIN code security
- Size: 10x11.9x1.7mm
- Works with: Android - YES; iOS - YES

FSC-BT909

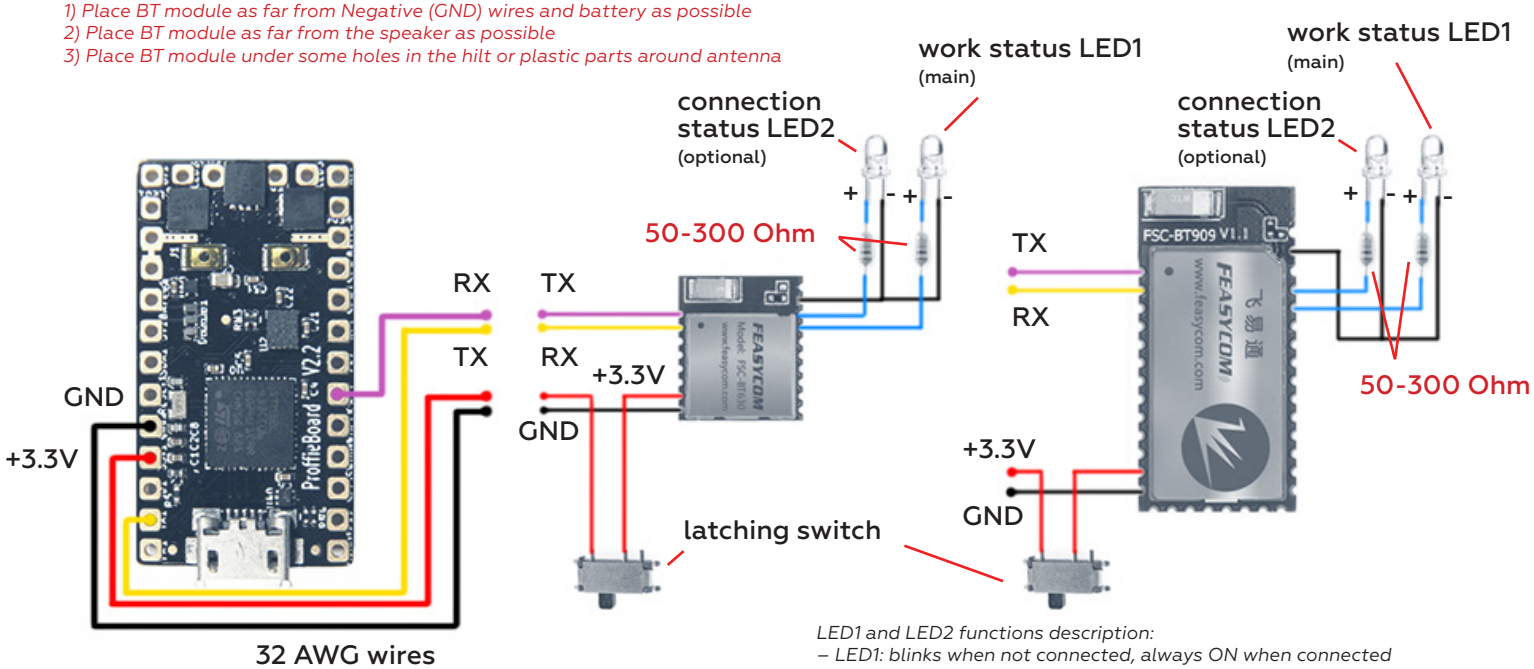


Features:

- CSR8811 chipset
- Bluetooth 4.2/4.1/4.0/3.0/2.1/2.0/1.2/1.1 support
- Class 1 (signal power up to +18dBm)
- Profiles including A2DP, AVRCP, HFP/HSP, SPP, GATT
- Built-in ceramic chip antenna, external antenna optional
- Current consumption: 30mA connected, 50mA max
- Connection status LED indication
- PIN code security
- Size: 13x26.9x2mm
- Works with: Android - YES; iOS - YES

For maximum bluetooth signal efficiency for both modules follow these rules:

1) Place BT module as far from Negative (GND) wires and battery as possible
2) Place BT module as far from the speaker as possible
3) Place BT module under some holes in the hilt or plastic parts around antenna



LED1 and LED2 functions description:
– LED1: blinks when not connected, always ON when connected
– LED2: always OFF when not connected, always ON when connected

PROFFIEBOARD INSTRUCTIONS

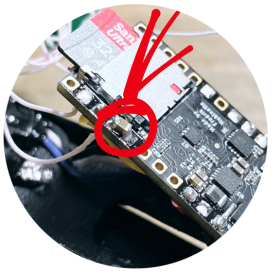
1 WIRING DIAGRAMS

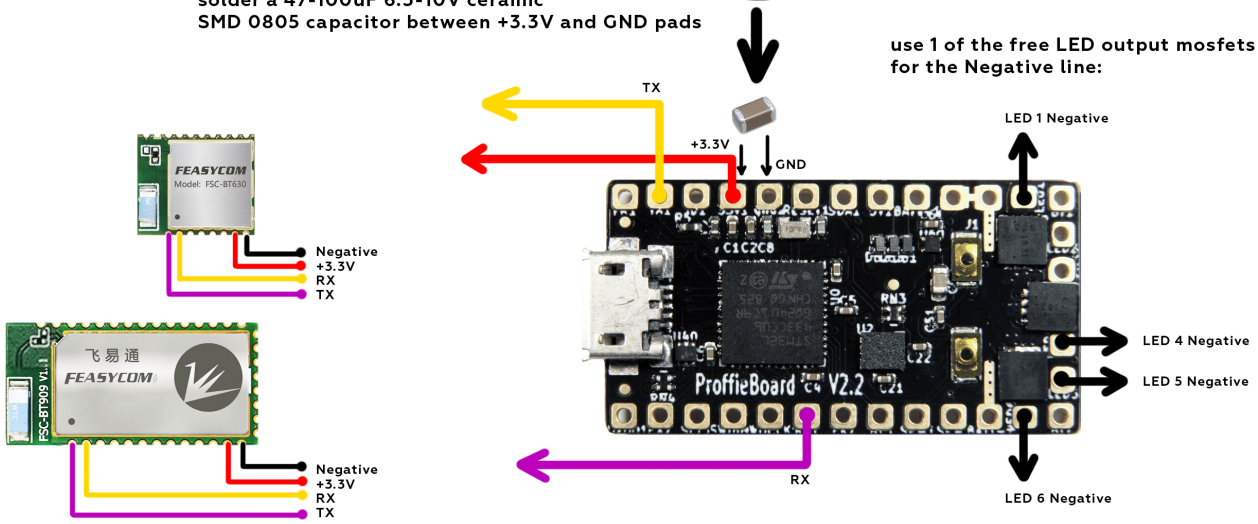
Bluetooth module alternative “no switch” wiring diagram (optional)

Automatic Bluetooth module shut down with Proffieboard v2.2 and v1.5 in standby mode for battery power saving, no switch required.
Only for ProffieOS 3.x and later.

Extra component required:

– 47-100 uF 6.3-10V ceramic SMD 0603 or 0805 capacitor
(can be purchased from digikey, mouser, ebay, aliexpress etc...)






```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 2                                                    ——— add +1 “blade” (the bluetooth module blade style)
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#define ENABLE_SERIAL
#define IDLE_OFF_TIME 60 * 10 * 1000                                    ——— add this line to set automatic OFF time (in “ms”),
                                                                              after which all LED channels (mosfets) will be turned off: “10” is 10 minutes here
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

  { "TeensySF", "tracks/venus.wav",
    StyleNormalPtr<CYAN, WHITE, 300, 800>(),
    StylePtr<Blue>(),                                                   ——— add this bluetooth module blade style
"cyan"},

};
BladeConfig blades[] = {
 { 0,
   WS281XBladePtr<144, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   SimpleBladePtr<CH3LED, NoLED,  NoLED, NoLED, bladePowerPin6, -1, -1, -1>(),   ——— add this bluetooth module “blade” configuration
CONFIGARRAY(presets) },
};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

PROFFIEBOARD INSTRUCTIONS

21 WIRING DIAGRAMS

Bluetooth module setup

Bluetooth Remote Control video Demo link

FSC-BT630 and **FSC-BT909** bluetooth modules are programmed by AT commands using any serial terminal software. But from some vendors (**TCSS**, **Saberbay** Etsy, **KR-sabers**) they come already pre-programmed, so can be wired and installed straight out of the box. Then you just need to connect to one of these two modules and set bluetooth name and pin code via ***FeasyBlue app*** (download on Google Play or App Store) on Android or iOS smartphone/tablet device.

Add *#define ENABLE_SERIAL* line to your Proffieboard config.h file:

```
#define ENABLE_WS2811
#define ENABLE_SD
#define ENABLE_SERIAL

#endif
```

If you buy directly from **Feasycom** manufacturer on alibaba, make sure to ask seller to pre-program ***FSC-BT909*** modules with these AT commands, or you need to program them yourself via ***FeasyBlue app*** (**watch how-to video here**):

AT+PROFILE=3
AT+COD=00050C
AT+TPMODE=1
AT+AUTOCONN=0
AT+PAIR=1
AT+SSP=0
AT+BAUD=115200
AT+BTEN=1

FSC-BT630 modules have no settings to program, they work straight out of the box (we recommend to buy both modules from these vendors: **TCSS**, **Saberbay** Etsy, **KR-sabers**, **JQsabers**, so you could be sure they work properly, because they are already preconfigured and tested to work with our saber sound boards).
Then you just can change ***passcode*** and ***name*** for both modules via ***FeasyBlue app*** using your smartphone (Android app will ask for a Passcode: 20138888).
Both modules support **OTA firmware upgrade** (Over The Air) via bluetooth SPP connection (with Android smartphones only at the moment), so if new features are added to the bluetooth module firmware in the future by Feasycom, modules can be easily updated inside the saber without rewiring. BT630 module firmware can be updated also from iPhone via the ***nRF Connect app***, you will need a new firmware ZIP archive file (contact ShtokCustomWorx to get this file).

– iOS:

ForceSync app on the App Store download link

– Android:

ForceSync app on the Google Play download link

2.1 PROFFIEBOARD INSTRUCTIONS

WIRING DIAGRAMS

Blade ID resistor functions (optional)

– allows Proffieboard to detect and identify the type of the blade that is connected (Tri-Cree LED, led string blade, charging adaptor, Pixel strip blade etc...).

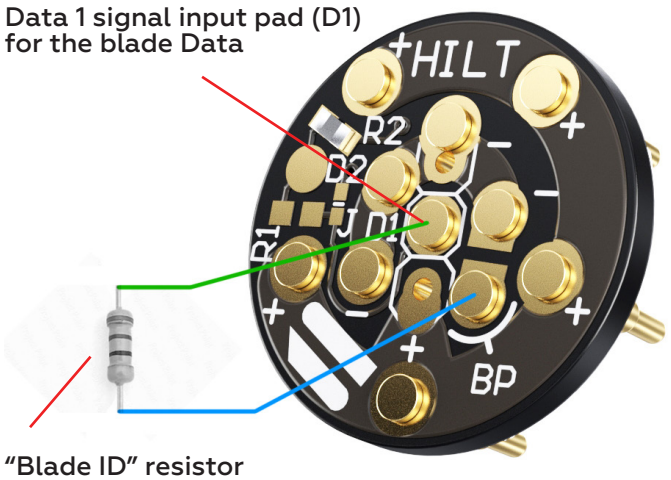
– allows Proffieboard to automatically switch to a specific Preset (blade style, sound font...) when blade is inserted or removed.

"Blade ID" resistor must be soldered between Negative and Data 1 signal pads (only Data pad #1 can read the "Blade ID", please see the Proffieboard pinout diagram) and before the Data resistor that goes to pixel strips Data IN pad (Din). "Blade ID" resistor must be 2-100 kOhm, any wattage (use the smallest you can work with).

A "Blade ID" resistor is located in the saber hilt.

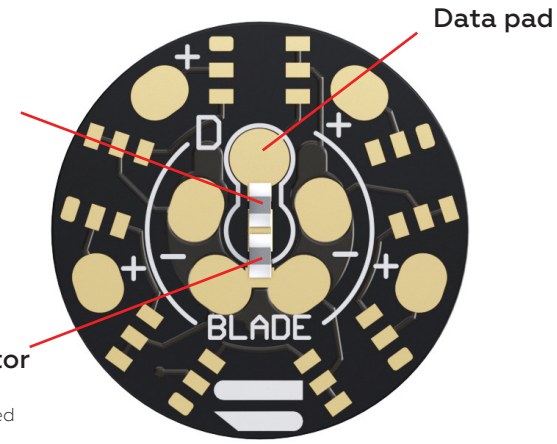
Using a SCW NPXL connector, "Blade ID" resistor must be wired between the "BP" pad and a center pad Data 1 input for the blade, as shown.
"BP" pin is free by default, not connected to anything, but once it makes contact with the Blade side pcb, it gets shorted to Negative line. This way board knows that resistence is changed.

NPXL blade connector from ShtokCustomWorx



B "Blade ID" resistor is located in the blade.

Using a SCW NPXL connector Blade side pcb or any other, solder "Blade ID" resistor between Data pad and any Negative pad.
This ID resistor will tell the board what type of the blade is inserted. Different blade types must have different "Blade ID" resistor values: 10 kOhm, 20 kOhm etc. This way board can automatically adjust leds number in different length Pixelblades, switch between different types of "Blade Configurations": Tri-Cree LED, led string blade, charging adaptor, Pixel strip blade etc.



PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Blade ID resistor functions
“config.h” file setup

A “Blade ID” resistor is located in the saber hilt (Blade Detect feature).

```
#ifdef CONFIG_TOP
#include “proffieboard_v2_config.h”
#define NUM_BLADES 5
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#define SHARED_POWER_PINS
#define ENABLE_POWER_FOR_ID PowerPINS<bladePowerPin2, bladePowerPin3>
#endif

#ifdef CONFIG_PRESETS
Preset chassis[] = {

  { “Yoda”, “tracks/YvsD.wav”,
&style_charging,
StyleNormalPtr<Green, WHITE, 300, 800>(),
&style_charging,
StyleNormalPtr<Green, WHITE, 300, 800>(),
StylePtr<Blinking<Red,Black,3000,800>>(),
“Green”},

};

Preset blade[] = {

  { “Yoda”, “tracks/YvsD.wav”,
StyleNormalPtr<Green, WHITE, 300, 800>(),
StylePtr<Black>(),
&style_charging,
StylePtr<Black>(),
StylePtr<Blinking<Red,Black,3000,800>>(),
“Green”},

};

BladeConfig blades[] = {
{ 150,
   WS281XBladePtr<89, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   WS281XBladePtr<5, blade3Pin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   SubBlade(0, 0, WS281XBladePtr<6, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >() ),
   SubBlade(1, 5, NULL),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin5, -1, -1, -1>(),
CONFIGARRAY(blade) },

{ 23600,
   WS281XBladePtr<89, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   WS281XBladePtr<5, blade3Pin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   SubBlade(0, 0, WS281XBladePtr<6, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >() ),
   SubBlade(1, 5, NULL),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin5, -1, -1, -1>(),
CONFIGARRAY(chassis) },
};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, “pow”);
Button AuxButton(BUTTON_AUX, auxPin, “aux”);
#endif
```

5 “blades”

define in case different blades with different Data pins use same MOSFETS in 1 blade config

define to enable “Blade ID” reading for PowerPins, to which the Main blade is connected (Negative wires)

Presets for “chassis” mode (blade is disconnected)

Presets names for different Blade Configs

Presets for “blade” mode (blade is connected)

resistance value when blade is connected

// Main Blade 89 pixels
// Connector 5 pixels
// Battery Level (PLI) 1 pixel
// Side accent 5 pixels
// Red accent LED

// Main Blade 89 pixels
// Connector 5 pixels
// Battery Level (PLI) 1 pixel
// Side accent 5 pixels
// Red accent LED

(20 kOhm Blade ID resistor on the connector Hilt pcb)
Put a value here that you see in Serial Monitor when Proffieboard is powered by battery and connected via USB cable to PC. Type and send *scanid* command:

Power for ID enabled. Turning on FETs
ID: 448 volts 1.44 resistance= 23666.67

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Blade ID resistor functions
“config.h” file setup

B “Blade ID” resistor is located in the blade.

```
#ifdef CONFIG_TOP
#include “proffieboard_v2_config.h”
#define NUM_BLADES 5
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#define SHARED_POWER_PINS
#define ENABLE_POWER_FOR_ID PowerPINS<bladePowerPin2, bladePowerPin3>
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {

  { “Yoda”, “tracks/YvsD.wav”,
StyleNormalPtr<Green, WHITE, 300, 800>(),
StyleNormalPtr<Green, WHITE, 300, 800>(),
&style_charging,
StyleNormalPtr<Green, WHITE, 300, 800>(),
StylePtr<Blinking<Red,Black,3000,800>>(),
“Green”},

};

BladeConfig blades[] = {
{ 13000,
   WS281XBladePtr<89, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   WS281XBladePtr<5, blade3Pin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   SubBlade(0, 0, WS281XBladePtr<6, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >() ),
   SubBlade(1, 5, NULL),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin5, -1, -1, -1>(),
 CONFIGARRAY(presets) },

{ 23600,
   WS281XBladePtr<136, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   WS281XBladePtr<5, blade3Pin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   SubBlade(0, 0, WS281XBladePtr<6, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >() ),
   SubBlade(1, 5, NULL),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin5, -1, -1, -1>(),
 CONFIGARRAY(presets) },

{ 54000,
   SimpleBladePtr<CreeXPE2RedTemplate<1000>, CreeXPE2GreenTemplate<0>, CreeXPE2BlueTemplate<240>, NoLED>(),
   WS281XBladePtr<5, blade3Pin, Color8::GRB, PowerPINS<bladePowerPin6> >(),
   SubBlade(0, 0, WS281XBladePtr<6, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin4> >() ),
   SubBlade(1, 5, NULL),
   SimpleBladePtr<CH1LED, NoLED, NoLED, NoLED, bladePowerPin5, -1, -1, -1>(),
 CONFIGARRAY(presets) },
};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, “pow”);
Button AuxButton(BUTTON_AUX, auxPin, “aux”);
#endif
```

5 “blades”

define in case different blades with different Data pins use same MOSFETS in 1 blade config

define to enable “Blade ID” reading for PowerPins, to which the Main blade is connected (Negative wires)

1 Presets Block (with name “presets”) for all Blade Configs

(but can be more Presets Blocks with different names, see page 22)

```
// Main Blade with 89 pixels
// Connector 5 pixels
// Battery Level (PLI) 1 pixel
// Side accent 5 pixels
// Red accent LED

// Main Blade with 136 pixels
// Connector 5 pixels
// Battery Level (PLI) 1 pixel
// Side accent 5 pixels
// Red accent LED

//Main Blade using Tri-Cree RGB LED module
// Connector 5 pixels
// Battery Level (PLI) 1 pixel
// Side accent 5 pixels
// Red accent LED
```

(20 kOhm Blade ID resistor in the Blade)
Put a value here that you see in Serial Monitor when Proffieboard is powered by battery and connected via USB cable to PC.
Type and send *scanid* command:

Power for ID enabled. Turning on FETs
ID: 448 volts 1.44 resistance= 23666.67

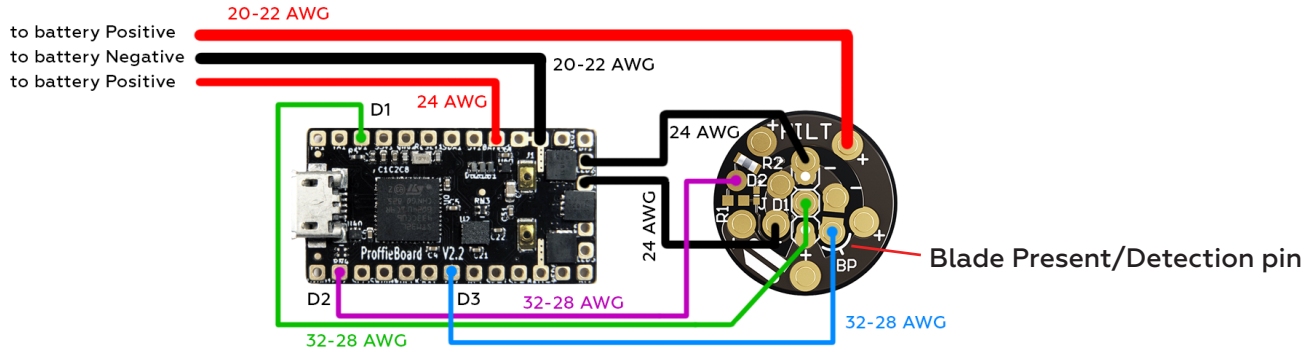
Same way to set other Blade IDs.

PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

Blade Detection wiring and setup

– allows Proffieboard to detect the blade connection and removal in real-time (requires a blade connector with “Blade Detection” pin, like a “NPXL” connector from KR-sabers, SaberBay and ShtokCustomWorx).
– Data 2-4, RX (8), TX (9), Aux2 button 3 (22) pads can be used as a Blade Detection pin.
– you need to have bladein.wav and bladeout.wav sound files in each sound font folder on SD card



Config file setup:

```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 2
#define NUM_BUTTONS 2
#define VOLUME 3000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 3.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#define ENABLE_SERIAL
#define SAVE_STATE
#define SHARED_POWER_PINS
#define IDLE_OFF_TIME 60 * 10 * 1000
#define MOTION_TIMEOUT 60 * 10 * 1000
#define BLADE_DETECT_PIN 17
#endif

#ifdef CONFIG_PROP
#include "../props/saber_shtok_buttons.h"
#endif

//Luke ANH lightsaber

#ifdef CONFIG_PRESETS

Preset no_blade[] = {
  { "Calibrate", "tracks/track04.wav",
StylePtr<InOutHelper<LocalizedClash<Lockup<Blast<OnSpark<AudioFlicker<Blue,Rgb<0,100,100>>,Rgb<255,255,150>,200>,Rgb<255,100,0>>,RandomFlicker<HumpFlicker<White,Red,25>,Yellow>,RandomFlicker<HumpFlicker<White,Orange,50>,Red>,Bump<Scale<BladeAngle<32768,0>,Int<10000>,Int<16384>>>>,Rgb<255,255,100>,100,100>,200,500>>(),
StylePtr<InOutHelper<LocalizedClash<Lockup<Blast<OnSpark<AudioFlicker<Blue,Rgb<0,100,100>>,Rgb<255,255,150>>,Rgb<255,100,0>>,RandomFlicker<HumpFlicker<White,Red,25>,Yellow>,RandomFlicker<HumpFlicker<White,Orange,50>,Red>,-
Bump<Scale<BladeAngle<32768,0>,Int<10000>,Int<16384>>>>,Rgb<255,255,100>,100,100>,200,500,Pulsing<Black,AudioFlicker<Blue,Rgb<0,100,100>>,5000>>>(),
"Calibrate"},

};

Preset blade[] = {
  { "Graflex", "tracks/track05.wav",
StylePtr<InOutHelper<LocalizedClash<Lockup<Blast<OnSpark<AudioFlicker<Blue,Rgb<0,0,100>>,Rgb<255,255,150>,200>,Rgb<255,100,0>>,RandomFlicker<HumpFlicker<White,Red,25>,Yellow>,RandomFlicker<HumpFlicker<White,O-
range,50>,Red>,Bump<Scale<BladeAngle<32768,0>,Int<10000>,Int<16384>>>>,Rgb<255,255,100>,100,100>,300,500>>(),
StylePtr<InOutHelper<LocalizedClash<Lockup<Blast<OnSpark<AudioFlicker<Blue,Rgb<0,0,100>>,Rgb<255,255,150>>,Rgb<255,100,0>>,RandomFlicker<HumpFlicker<White,Red,25>,Yellow>,RandomFlicker<HumpFlicker<White,Orange,50>,Red>,-
Bump<Scale<BladeAngle<32768,0>,Int<10000>,Int<16384>>>>,Rgb<255,255,100>,100,100>,300,500,Pulsing<Black,AudioFlicker<Blue,Rgb<0,0,100>>,5000>>>(),
"Graflex"},

};

BladeConfig blades[] = {
{ 0,
   WS281XBladePtr<138, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   WS281XBladePtr<5, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
CONFIGARRAY(blade) },

{ NO_BLADE,
   WS281XBladePtr<138, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
   WS281XBladePtr<5, blade2Pin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(),
CONFIGARRAY(no_blade) },

};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

define the detect pin on Proffieboard (can be a pin number (17) on the board pinout scheme or a symbolic name like blade3Pin)

Presets for “no blade” mode (blade is disconnected)

Presets names for different Blade Configs

Presets for “blade” mode (blade is connected)

//Main Blade 138 pixels (“blade” #1)
//Connector 5 Pixels (“blade” #2)

//Main Blade (No Blade) (“blade” #1)
//Connector 5 Pixels (“blade” #2)

PROFFIEBOARD INSTRUCTIONS

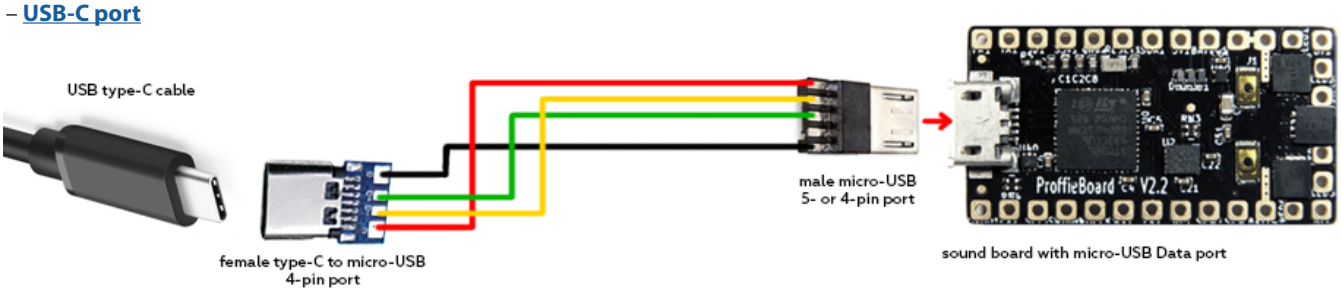
1 WIRING DIAGRAMS

USB-C port and battery charging wiring

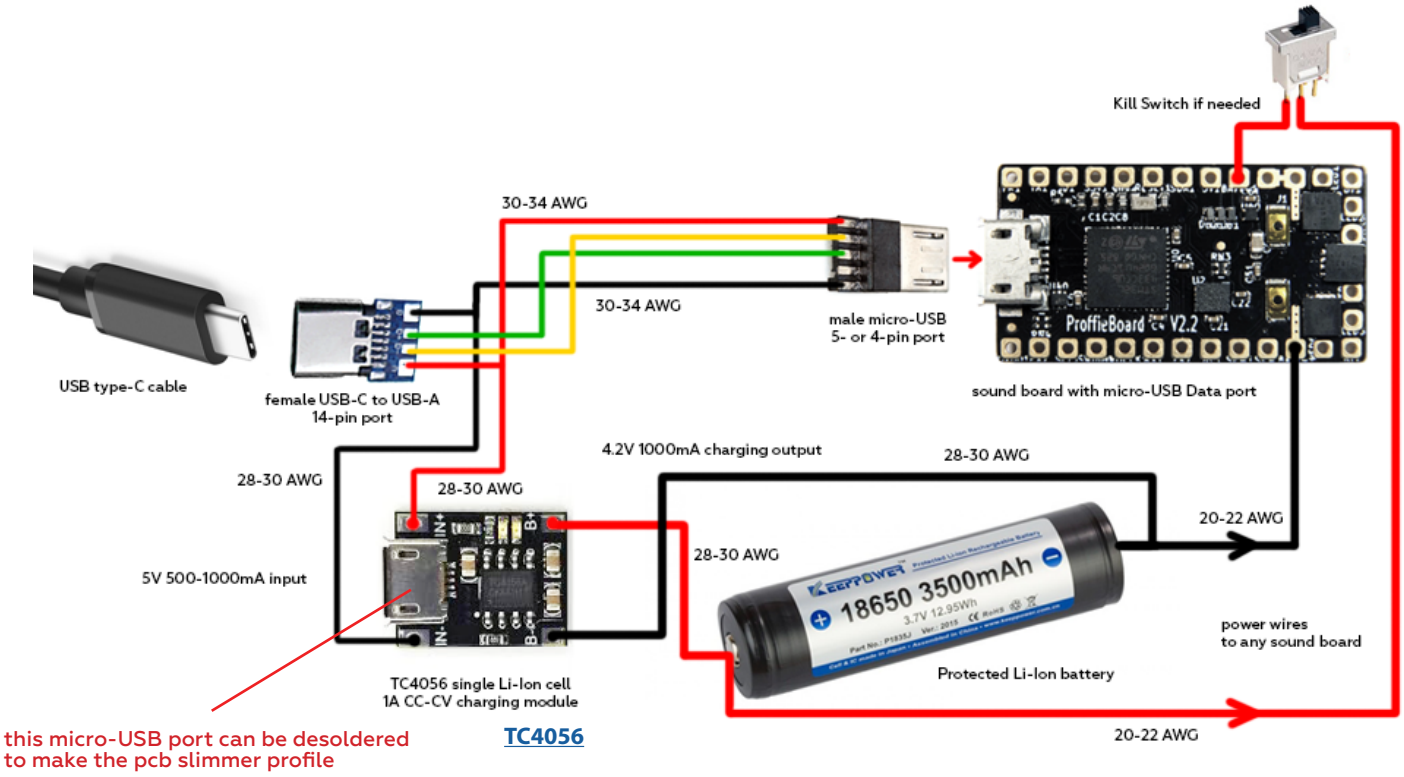
– for connecting Proffieboard equipped saber to PC/Mac via USB-C cable for SD card data access, Proffieboard firmware upgrade and debugging, battery charging.

Data transfer only:

– **USB-C port**



With battery charging pcb:



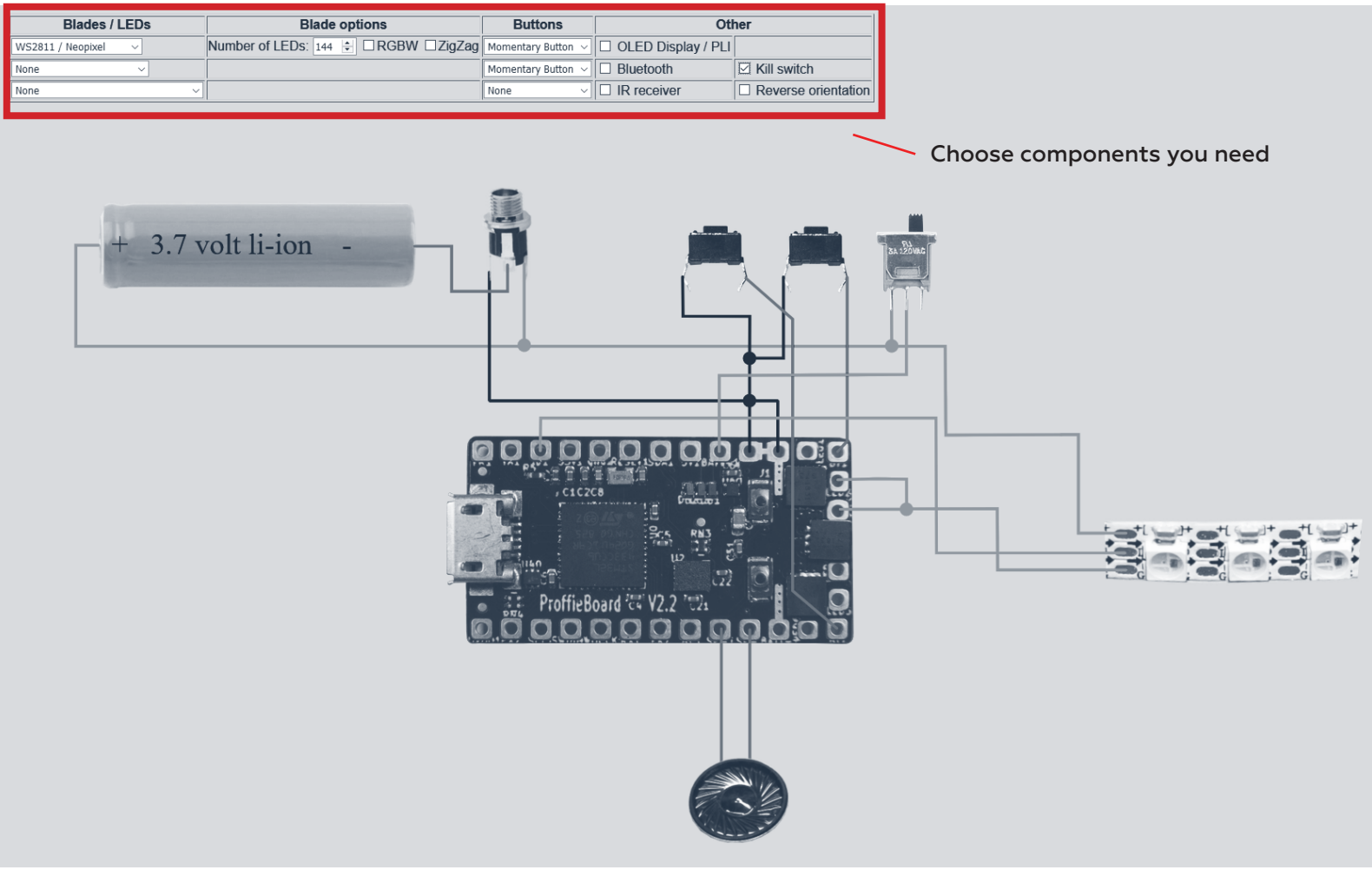
PROFFIEBOARD INSTRUCTIONS

1 WIRING DIAGRAMS

More wiring diagrams

[>>>website ProffieBoard FULL wiring diagrams link<<<]

Scroll the page down till you see the interactive diagram. Above the diagram there are components selection options. Build your saber setup with it and follow diagram to wire your board. Then you need to copy the configuration code below and paste it into your ..._*config.h* file.



PROFFIEBOARD INSTRUCTIONS

HOW TO USE

Buttons behavior

Blade ignition/retraction – assuming you have at least one button, pressing it briefly should turn the saber on or off. If you have an AUX button, pressing it briefly should also turn the saber on and off. If you have no buttons, you can turn the saber on and off by twisting your wrist back and forth. Note that the motion has to be done long enough to count, so a very quick flick of the wrist will not work
Turn On muted – double-click power button
Next preset – while blade is off, click the AUX button
Previous preset – hold AUX button and click the Activation button
Trigger Clash – while blade is on, hit the blade
Trigger Lockup – while blade is on, hold Activation button, then trigger a clash. Lockup releases when you let go of the Activation button
Trigger Drag – like lockup, but point saber mostly down before holding Activation button
Trigger Force – long-click AUX button
Start soundtrack – long-click the Activation button
Trigger Blaster Block – while blade is on, short-click AUX button
Enter Color Changing mode – while blade is on, hold AUX button and quickly press Activation button, it will make a sound, now rotate the hilt to change colors. To exit Color Changing mode hold Activation button until you hear the sound

Different buttons behavior can be made and saved as a file in the "props" folder in ProffieOS. Read more **>>>here<<<**.
As an example by default there are 3 alternative buttons files that have slightly different behavior for different taste, open the file in any text editor to see how to operate the saber:

– **saber_sa22c_buttons.h**
– **saber_shtok_buttons.h** (the latest updated file can be also downloaded here: **saber_shtok_buttons.h**, put this file into the "props" folder replacing the default one with same name)
– **saber_fett263_buttons.h**

Please open the file in any text editor and read how to use buttons there in the comments description.
To use alternative buttons file:

```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 1
#define NUM_BUTTONS 2          —— set the number of buttons 1 or 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#endif

#ifdef CONFIG_PROP
#include "../props/saber_sa22c_buttons.h"
#endif                         —— add these lines between CONFIG_TOP and CONFIG_PRESETS sections
                                  exactly as shown
#ifdef CONFIG_PRESETS
Preset presets[] = {

 { "TeensySF", "tracks/venus.wav",
   StyleNormalPtr<CYAN, WHITE, 300, 800>(),
```

PROFFIEBOARD INSTRUCTIONS

HOW TO USE

Serial Monitor commands:

battery_voltage – get current battery voltage value
get_volume – get current volume value
pow – power On/Off the saber
on – power On the saber
off – power Off the saber
set_volume <0-3000> – set volume value (example: *set_volume 500*)
play – play the default preset track, stop playing track while it's playing
play_track tracks/<track name>.wav – play a specific track from tracks folder (example: *play_track tracks/venus.wav*)
stop_track tracks/<track name>.wav – stop a playing track from tracks folder (example: *stop_track tracks/venus.wav*)
force – play "force" sound effects
drag – play "drag" sound effects
blast – play "blaster" sound effects
lock – play "lockup" sound effects
clash – play "clash" sound effects
reset – reboot the board
n – switch to next preset
p – switch to previous preset
list_presets – show all presets
sdtest – test SD card speed
scanid – print out a Blade ID resistance

New ProffieOS 3.x CONFIG file defines (More >>>here<<<):

#define ENABLE_DEVELOPER_COMMANDS – By default, some commands which are only useful for developers are normally not compiled into the final binary to save memory, if you want them, add this define to enable them
#define DISABLE_DIAGNOSTIC_COMMANDS – To save more memory, you can disable some diagnostic commands like "monitor", "top" and "sdtest" using this define
#define BLADE_ID_CLASS BridgedPullupBladeID<bladeIdentifyPin, BRIDGED_PIN>
#define BLADE_ID_CLASS ExternalPullupBladeID<bladeIdentifyPin, PULLUP_RESISTANCE> – Proffieboards have some challenges when it comes to BladeID, but it's possible to work around them by adding a bridge to another pin, or by adding a pullup resistor. However, when you do so, you have to use the BLADE_ID_CLASS to specify how the OS should calculate the Blade ID. Chose one of these two
#define IDLE_OFF_TIME 60 * 10 * 1000 – ProffieOS has pretty good standby idle time, but if you have accent leds that glow even when the saber is off, that will make your saber run out of batteries pretty fast. This define lets you specify a timeout for such accent leds (in milliseconds), this example would set it to 10 minutes
#define BLADE_DETECT_PIN PIN – This define lets you use a pin to detect when a blade is present or not. The pin will work kind of like a latching button which is pressed when there is a blade in the saber. When there is no blade in the saber,
NO_BLADE (one billion) will be added to the blade ID
#define DISABLE_COLOR_CHANGE – If you want to disable the Color Change feature to save some board memory
#define SAVE_COLOR_CHANGE – Start with the blade color used last
#define SAVE_PRESET – Start at the last selected Preset when you turn the saber on
#define SAVE_VOLUME – Start with the volume used last
#define SAVE_STATE – Is the same as SAVE_COLOR_CHANGE + SAVE_VOLUME + SAVE_PRESET together
#define KEEP_SAVEFILES_WHEN_PROGRAMMING – From OS4.7 onward after uploading all .INI files will be deleted, which may contain some alteration to your presets from colorchangemode, volume etc. to retain this additional information even after re-flashing the firmware
#define ORIENTATION ORIENTATION_USB_TOWARDS_BLADE – If your board is installed in a backwards orientation, you may need to add the following line to make ProffieOS behave properly

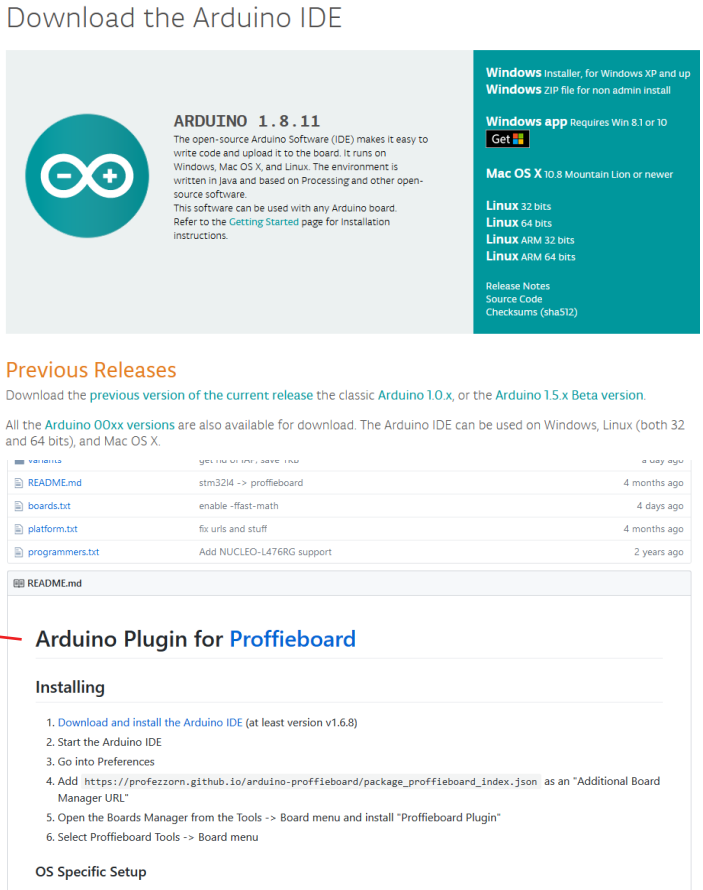
PROFFIEBOARD INSTRUCTIONS

FIRMWARE UPLOAD AND UPDATE

Software installation and setup

To upload firmware to ProffieBoard **Arduino IDE** program is required. Follow these steps to install it to your PC.
You can also watch these tutorial videos: by Desert Sabers, by Megtooth Sith Sabers, by Daniel Newman.

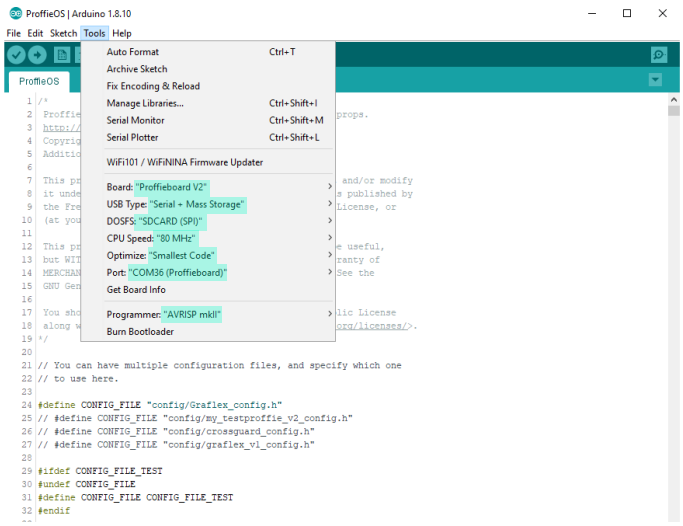
1 Install latest Arduino IDE software (don't use BETA).
Installing as Windows app also is not recommended, because it will be installed in a specific protected folder that won't allow you to install any additional software/plugin in it.
If ProffieBoard won't show up in COM port, use a previous Arduino IDE version.



2 Install the Proffieboard Arduino Plugin and Zadig program.
Follow installation instructions on this page.

3 Select **Proffieboard V2** in *Tools -> Board*
USB Type – **Serial + Mass Storage (or + WebUSB)**
CPU Speed – **80 MHz**
Optimize – **Smallest Code (or Fast/Faster/Fastest)**
DOSFS – **SDCARD (SPI)**
Port – **COM***(the number your PC assigned)* **(Proffieboard)**

Connect Proffieboard via USB cable to PC to be able to select the Port.



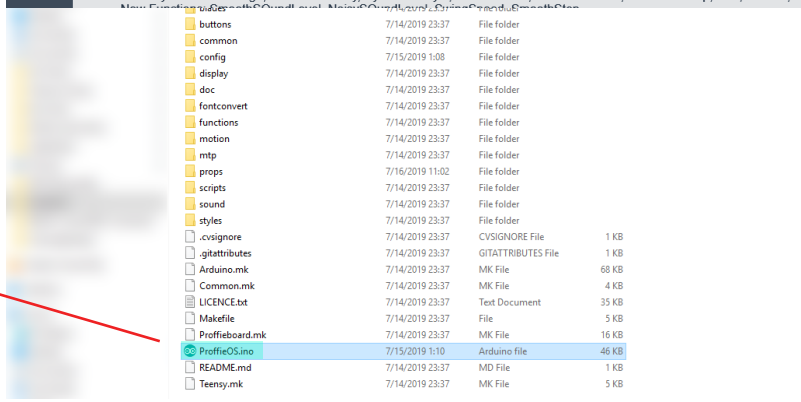
PROFFIEBOARD INSTRUCTIONS

FIRMWARE UPLOAD AND UPDATE

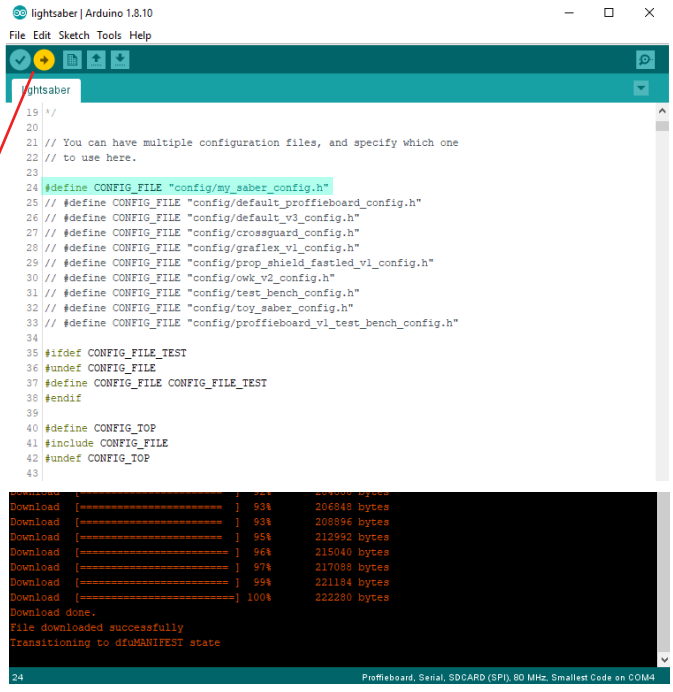
Uploading firmware

1 Download the Proffieboard firmware and SD card content. Unzip *ProffieOS-v2.2.zip* to your **Documents** directory or to **Desktop**, but not to Arduino program folder or anywhere in **Programs** directory! You will see a *ProffieOS-v2.2* folder and *ProffieOS* folder inside it: ...*ProffieOS-v2.2**ProffieOS*. Don't rename or move any of these folders and files inside them to any other location outside the *ProffieOS* folder! Unzip *ProffieOS_SD_Card.zip* to the directory where you keep *ProffieOS-v2.2* folder. Copy all files from *ProffieOS_SD_Card* folder to your SD card.

2 Unhide file extensions in File Explorer settings to see ***.h*** ending of config files. **Don't add ".h" to the config file name!** Go to **config** folder and create you own *config.h* file (see page 31 for how-to). Double-click the *ProffieOS.ino* file.



3 Add the name of your *config.h* file as shown and **Save** this *ProffieOS.ino* file. Make sure the other config files are commented out, there should be only one *CONFIG_FILE* without by "//". You can have multiple config files in **ProffieOS\config** folder and just define the one you need in *ProffieOS.ino* file and upload it again to Proffieboard.



4 Connect Proffieboard to your PC by a data transfer micro-USB to-USB cable. Press **arrow button**, it will compile and upload firmware to the board. Wait for red text progress bars to stop at 100%, ProffieBoard will play boot sound if speaker is connected. Properly eject the SD card if "Serial + Mass Storage" is used. Now you can unplug the USB cable. Done!
If it gives an error instead, this means your *config.h* file has issues, #define CONFIG_FILE name has mistakes, *config.h* file is out of **config** folder, your PC user name is non-latin...

PROFFIEBOARD INSTRUCTIONS

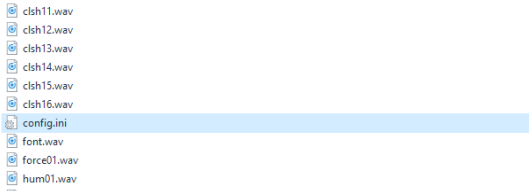
4 CHANGING PARAMETERS

config.h file structure, editing

All sound files (sound fonts, music tracks) are stored on the micro SD card. Add required sound fonts folders (Plecter, NEC and Smoothswing fonts are supported, no need to change WAV files names, just copy and paste) to SD card root directory as it's done in the default *ProffieOS_SD_Card* content folder and music tracks to the *tracks* folder.

Make sure to name all music tracks and sound fonts folders with latin characters and without using any special characters (like ?,.|\}{[/- etc.).

Make sure you have a *config.ini and smoothsw.ini* files in each sound font folder, if there is none - copy one from some default Proffieboard sound font and paste into newly added sound font folder.



All blade effects, LED configuration, volume level, clash sensitivity etc. are changed in the *config.h* file located in **ProffieOS\config** folder. To do that open any ..._*config.h* file in any Text Editor like Notepad, delete all text, make your wiring diagram config code here, copy and paste it into your ..._*config.h* file, **Save** it under new name. Follow the instructions on page 29-30 to upload it to the board.

More info >>>here<<<.

```
#ifdef CONFIG_TOP
#include "proffieboard_v2_config.h"
#define NUM_BLADES 1
#define NUM_BUTTONS 2
#define VOLUME 1000
const unsigned int maxLedsPerStrip = 144;
#define CLASH_THRESHOLD_G 1.0
#define ENABLE_AUDIO
#define ENABLE_MOTION
#define ENABLE_WS2811
#define ENABLE_SD
#define SAVE_STATE
#endif

#ifdef CONFIG_PRESETS
Preset presets[] = {
 { "TeensySF", "tracks/venus.wav",
  StyleNormalPtr<CYAN, WHITE, 300, 800>(), "cyan"},
 { "SmthJedi", "tracks/mars.wav",
  StylePtr<InOutSparkTip<EASYBLADE(BLUE, WHITE), 300, 800> >(), "blue"},
 { "SmthGrey", "tracks/mercury.wav",
  StyleFirePtr<RED, YELLOW>(), "fire"},
 { "SmthFuzz", "tracks/uranus.wav",
  StyleNormalPtr<RED, WHITE, 300, 800>(), "red"},
 { "RgueCmdr", "tracks/venus.wav",
  StyleFirePtr<BLUE, CYAN>(), "blue fire"},
 { "TthCrstl", "tracks/mars.wav",
  StylePtr<InOutHelper<EASYBLADE(OnSpark<GREEN>, WHITE), 300, 800> >(), "green"},
 { "TeensySF", "tracks/mercury.wav",
  StyleNormalPtr<WHITE, RED, 300, 800, RED>(), "white"},

};
BladeConfig blades[] = {
 { 0, WS281XBladePtr<144, bladePin, Color8::GRB, PowerPINS<bladePowerPin2, bladePowerPin3> >(), CONFIGARRAY(presets) },
};
#endif

#ifdef CONFIG_BUTTONS
Button PowerButton(BUTTON_POWER, powerButtonPin, "pow");
Button AuxButton(BUTTON_AUX, auxPin, "aux");
#endif
```

Proffieboard v2 config

number of "blades" used

number of buttons used *(1-3)*

volume level *(0-3000)*

Clash sensitivity *(lower = more sensitive, higher = less sensitive, try with 0.5 step)*

Makes the board save last selected Volume, Preset and blade Color

sound font folder name

track name

Blade style code

Preset 1

Preset name

blade configuration

PROFFIEBOARD INSTRUCTIONS

4 CHANGING PARAMETERS

Blade Styles

Proffieboard uses Blade Styles for the main saber blade and any other accent leds to define all light effects (color changing, flashes, flickering, delays, ignition/retraction timing etc...).
Use Blade Style Editor to create and adjust Blade Styles. **Megtooth Sith Sabers** did a great **video tutorial** where he shows and explains how to use Blade Style Editor. Also you can grab some pre-made Blade Styles or share yours here on TRA forums or from Fett263 Library.

A Blade Style example of simple **flickering Green** blade with **Spark on start**, **Clash**, **Blaster**, **Lockup** and **Drag**, **Ignition/Retraction** effects:

```
StylePtr<InOutHelper<SimpleClash<Lockup<Blast<OnSpark<AudioFlicker<Rgb<0,255,0>,Rgb<50,100,0>>,Rgb<255,255,0>,150>,Rgb<255,50,0>>,AudioFlicker<Rgb<100,255,0>,Rgb<255,0,150>>>,Rgb<255,100,150>,40>,200,300,Black>>
```

– this is how the Blade Style code looks pasted in the *config.h* file Preset (it sits inside a StylePtr<...> container)

```
InOutHelper<SimpleClash<Lockup<Blast<OnSpark<AudioFlicker<Rgb<0,255,0>,Rgb<50,100,0>>,Rgb<255,255,0>,150>,Rgb<255,50,0>>,AudioFlicker<Rgb<100,255,0>,Rgb<255,0,150>>>,Rgb<255,100,150>,40>,200,300,Black>
```

– this is how the Blade Style code looks when editing it inside a Blade Style Editor
Each Blade Style is made of a variety of **Effects**, each added effect goes instead of a *base color* in the previous effect:

InOutHelper<*base color*,200,300,Black> – *base color* can be defined by words (WHITE, RED, GREEN, PURPLE etc..) or by *Rgb<0-255,0-255,0-255>* values for more custom shades; *200* is extension length in milliseconds; *300* is retraction length in milliseconds; *Black* is color when retracted (also can be any other color)
SimpleClash<*base color*,*clash color*,40> – clash effect; *40* is clash duration in milliseconds
Lockup<*base color*,*lockup color*> – lockup effect
Blast<*base color*,*blast color*> – blaster effect
OnSpark<*base color*,*spark color*,*150*> – spark on ignition effect; *150* is spark duration in milliseconds
AudioFlicker<*"A" color*,*"B" color*> – flickering effect (blade flickers to the actual saber hum sound); the more difference between "A" and "B" colors - the more abrupt is flickering
Rgb<255,50,0> – actual color in RGB format (*0* is no light; *255* is the maximum brightness value for Red, Green or Blue channel)

PROFFIEBOARD INSTRUCTIONS

SD CARD RECOMMENDATIONS

Recommended micro-SD cards

Here is a list of tested micro-SD cards with Proffieboard. Any card with speed over 900 kb/s is recommended, the higher the speed – the better. Memory size of 4-16Gb is more than enough. Cards were tested with a default firmware compiled with "Smallest Code" under Optimize, "default_proffieboard_config.h" file and default ProffieOS SD card sound files (7 folders).
To test your micro-SD card speed simply hook up Proffieboard to PC, open **Arduino IDE**, go to **Tools** and open **Serial Monitor**, make sure you have New Line and 115200 baud rate selected on the bottom of Serial Monitor window, type and send ***sdtest*** command, wait for the test result.

BEST	– Kingston CANVAS Select Plus 16-32GB microSDHC UHS-I/U1 A1 Class 10 *1300.00+ kb/s = 15.00+ simultaneous audio streams*	
	– Kingston 16GB microSDHC UHS-I/U1 Class 10 *1280.90 kb/s = 14.52 simultaneous audio streams*	
	– SanDisk Ultra 16GB microSDHC UHS-I/U1 A1 Class 10 *1112.4 kb/s = 12.61 simultaneous audio streams*	
GOOD	– SanDisk 8GB microSDHC Class 4 (Genuine) *1085.06 kb/s = 12.30 simultaneous audio streams*	
	– SanDisk 16GB microSDHC Class 4 *1069.57 kb/s = 12.13 simultaneous audio streams*	
	– SanDisk Ultra 16GB microSDHC UHS-I/U1 Class 10 *1039.09 kb/s = 11.78 simultaneous audio streams*	
BAD	– Smartbuy 4GB microSDHC Class 4 *754.37 kb/s = 8.55 simultaneous audio streams*	
	– Kingston 8GB microSDHC Class 4 *752.09 kb/s = 8.22 simultaneous audio streams*	
	– SanDisk 4GB microSDHC Class 4 (Fake) *677 kb/s = 7.69 simultaneous audio streams*	

WIRE GAUGE GUIDE

Which wire gauge is recommended to use for Positive and Negative power leads for maximum blade brightness efficiency

AWG gauge	Conductor Diameter Inches	Conductor Diameter mm	Conductor cross section in mm^2	Ohms per 1000 ft.	Ohms per km	Maximum amps for chassis wiring	
14	0.0641	1.62814	2.08	2.525	8.282	32	*Chart from PowerStream.com*
15	0.0571	1.45034	1.65	3.184	10.44352	28	
16	0.0508	1.29032	1.31	4.016	13.17248	22	
17	0.0453	1.15062	1.04	5.064	16.60992	19	
18	0.0403	1.02362	0.823	6.385	20.9428	16	Neopixel strips Battery Recharge Port Kill Switch
19	0.0359	0.91186	0.653	8.051	26.40728	14	
20	0.032	0.8128	0.519	10.15	33.292	11	
21	0.0285	0.7239	0.412	12.8	41.984	9	
22	0.0253	0.64516	0.327	16.14	52.9392	7	
23	0.0226	0.57404	0.259	20.36	66.7808	4.7	
24	0.0201	0.51054	0.205	25.67	84.1976	3.5	Tri-Cree LED Battery Recharge Port Kill Switch
25	0.0179	0.45466	0.162	32.37	106.1736	2.7	
26	0.0159	0.40386	0.128	40.81	133.8568	2.2	
27	0.0142	0.36068	0.102	51.47	168.8216	1.7	
28	0.0126	0.32004	0.080	64.9	212.872	1.4	
29	0.0113	0.28702	0.0647	81.83	268.4024	1.2	
30	0.01	0.254	0.0507	103.2	338.496	0.86	Everything else
31	0.0089	0.22606	0.0401	130.1	426.728	0.7	
32	0.008	0.2032	0.0324	164.1	538.248	0.53	

Neopixel strips build (3-17 amperes load)			Tri-Cree LED build (1-4 amperes load)
2-strip	3-strip	4-strip	28-24 AWG recommended for battery wiring, choose regarding particular build *30 AWG possible for single 3W Cree LED wiring (one wire per die)*
22 AWG single or 24 AWG dual in parallel	20 AWG single or 23 AWG dual in parallel	18 AWG single or 22 AWG dual in parallel	

For all other components except Neopixel blade strips, high power Tri-Cree LEDs, battery and recharge port/Kill Switch – a 30-32 AWG wire can be used because they are low current circuits (5-500mA) (accent leds, activation and AUX switches, speaker, bluetooth module, RICE port etc.).

RECHARGE PORTS AND KILL SWITCHES

	3 Amps	5 Amps	6 Amps	7 Amps	8 Amps	11 Amps
 2.1mm Switchcraft 721A Recharge port	OK	OK	OK	OK	OK	96%
 1.3mm Recharge port CUI PJ-013D Martin Beyer 1.3mm Recharge port	OK	OK	OK	98%	97%	95.5%
 Martin Beyer Kill Switch	OK	OK	OK	OK	OK	98%
 1.3mm Recharge port CUI PJ-075DH	OK	OK	OK	97%	95%	94%
 regular cheap 1.3mm Recharge port	75%	melted	melted	melted	melted	melted
 3A Kill Switch CK TS01CQE	OK	OK	OK	OK	OK	98%
 Mini 6pin SMD Slide Switch MSS22D18	OK	OK	70%	melted	melted	melted

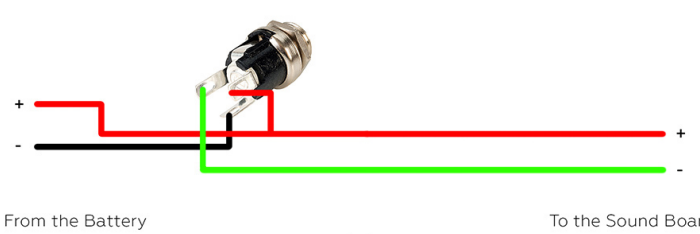
OK — safe to use

1-96% — efficiency (less than 95% not recommended!)

SHTOK CUSTOM WORX

How to wire Recharge Ports

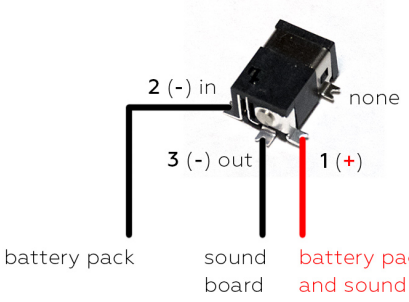
High power Recharge Port

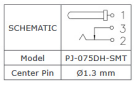


From the Battery

To the Sound Board

CUI PJ-075DH-SMT
High power 1.3mm recharge port
wiring diagram





] tested at 6 Amps

2 and 3 are the so-called "Kill Switch"

NEOPIXEL STRIPS CURRENT DRAW

Neopixel WS2812B/SK6812 strips tested approximate current consumption chart

Tested at 3.7V, 143 leds per strip, at max brightness

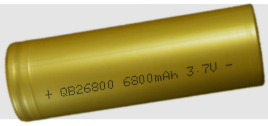
№ of strips	current	1 color without flicker / with flicker	2 colors mixed without flicker / with flicker	3 colors mixed for white without flicker / with flicker
1	Total	2 / 1.9 A	3.6 / 3.3 A	5.2 / 4.9 A
	Per LED	14 / 12.9 mA	12.6 / 11.5 mA	12.1 / 11.4 mA
2	Total	3.7 / 3.5 A	6.9 / 6.4 A	9.9 / 9.3 A
	Per LED	13 / 12.2 mA	12 / 11.1 mA	11.5 / 10.8 mA
3	Total	5.4 / 4.5 A	10.1 / 9.5 A	14.4 / 13.5 A
	Per LED	12.6 / 11.6 mA	11.8 / 11.1 mA	11.2 / 10.5 mA
4	Total	7.1 / 6.7 A	13 / 12.4 A	17.7 / 16.6 A
	Per LED	12.4 / 11.8 mA	11.4 / 10.8 mA	10.3 / 9.7 mA
5	Total	8.8 / 8.4 A	15.7 / 15 A	20.6 / 19.5 A
	Per LED	12.3 / 11.7 mA	11 / 10.5 mA	9.6 / 9.1 mA

RECOMMENDED BATTERIES CHART

SHTOK CUSTOM WORX

Best batteries for sabers

2020

SIZE	BRAND/MODEL	
18350	Keeppower 1200mAh 8A Protected 	Keeppower 1200mAh 10A Unprotected (requires external protection pcb) 
14500	Keeppower P1450C2 1000mAh 4A Protected 	Efest IMR14500 V2 650mAh 9A Unprotected (requires external protection pcb) 
14650	Efest IMR14650 950mAh 5A Unprotected (requires external protection pcb) 	Keeppower 1100mAh 2-3A Protected 
16650	Keeppower 2500mAh 5A Protected 	Sanyo UR16650ZTA 2500mAh 5A Unprotected (requires external protection pcb) 
18500	Keeppower P1850J2 2000mAh 4A Protected 	Keeppower IMR18500 1100mAh 10A Unprotected (requires external protection pcb) 
18650	Keeppower 3500mAh 10A Protected 	Keeppower 3120mAh 15A Protected 
21700	Acebeam 5100mAh 20A Protected 	Keeppower 5000mAh 10A Protected 
26650	Keeppower 6000mAh 10-15A Protected 	Keeppower 5500mAh 10A Protected 
26800	QueenBattery 6800mAh 30A Unprotected (requires external protection pcb) 	

These batteries work for both Tri-Cree and Neopixel setups.

For Tri-Cree LED setup choose the one with highest capacity value (**mAh**), for Neopixel setup choose the one with highest max drain value (**A**).

mAh – *milliamperes per hour: the battery energy capacity rating, means how long the battery will run at a single charge* – ***run time***

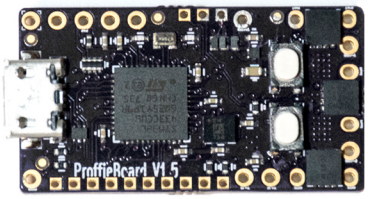
A – *Amperes: the battery energy max drain rating, means how much Amperes this battery can provide continuousl* – ***blade brightness performance under high current load***

For Tri-Cree LED setup batteries with 2-3A drain rating are OK (can be higher, but lower are not recommended). For Neopixel setup batteries with 10-15A drain rating are recommended (can be higher, but lower are not recommended). For battery sizes 14500, 14650, 16650, 18350, 18500 it's hard or impossible to find a good capacity with high drain rating, so 5-8A examples from the chart above are best options.

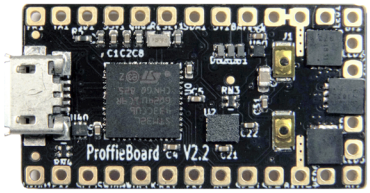
PROFFIEBOARD COMPARISON CHART

DIFFERENCES CHART
Proffieboard v1.5 vs v2.2

Sound board



Proffieboard v1.5



Proffieboard v2.2

Price	$50-65	$50-65
size (without SD and micro USB port)	33.2x17.8x5.7mm	33.2x17.8x4.0mm (+3.2mm more sticking out micro SD card)
pads		larger, easier to solder to
breadboard compatibility		✓
5V booster for audio amplifier	up to 1.2 Amps	up to 4 Amps
neopixel Data signal outputs	5	4
"Deep Sleep" feature support possibility		✓
"GND" and "BATT-" pads	separate	bridged (can be separated if needed)
battery reverse polarity protection		✓
potential support for analog out		✓
built-in 470 ohm resistor on Data #1 signal output		✓
BOOT and RESET buttons		smaller

5 TROUBLESHOOTING

Quick troubleshooting tips

How to solve most common issues

"Font directory not found. SD card not found" spoken error...
– Check if sound fonts folders names on SD card exactly match the font names in each Preset in your config.h file. Reformat SD card in FAT32 and try again.

Proffieboard is not recognized by computer (nothing under Port selection in Arduino IDE)...
– Make sure a charged 3.7V battery is connected to the board, micro-USB cable is a data transfer cable, all plugins and drivers are installed – check again pages 29-30. Try a different cable and USB port on your computer.

Proffieboard is recognized by computer always only as "STM32 BOOTLOADER" (nothing under Port selection in Arduino IDE)...
– If **Zadig** driver is installed properly but Proffieboard is still recognized by PC always as "STM32 BOOTLOADER" instead of "Proffieboard" and only after pressing RESET button while holding BOOT button – open Arduino IDE, make sure you use latest ProffieOS and Proffieboard plugin version, without selecting the Port under Tools tab click the **Verify** code button and after it's finished click the **Upload** button. Firmware must now update on Proffieboard and it will be recognized correctly next time you plug it into USB port.

Sketch (code) compile error in Arduino IDE...
– Check your *#define CONFIG_FILE "config/..._config.h"* line in opened ProffieOS.ino file if it's written correctly with *config/* in it. Check if the *..._config.h* file you defined in the ProffieOS.ino sketch file is same name as in the **ProffieOS-"firmware version"/ProffieOS/config** folder and is located in this folder.

Sketch (code) compile error in Arduino IDE: *... sketch\config/proffiev2config.h:42:1: error: cannot convert 'StyleFactory*' to 'const char*' in initialization ...*
– Missing blade style in one of the Presets: NUM_BLADES set to 2 or more, but Presets have less blade styles.

Sketch (code) compile error in Arduino IDE: *... sketch\config/proffiev2config.h:42:1: error: too many initializers for 'Preset' ...*
– Check if *#define NUM_BLADES* value matches the number of your blade styles in each Preset. Some of the Presets have more blade styles than needed.

Sound doesn't play...
– Remove SD card and insert again, check speaker wiring. Make sure all sound files on SD card are correctly named. Re-format SD card in FAT32, load sound files and try again, try another SD card.

Board says "LOW POWER"...
– Charge the battery.

Serial Monitor shows info sent by the board but your commands don't work...
– In the bottom right corner of Serial Monitor window make sure the Line Ending drop down is set to New Line.

Sound is weird and distorted...
– Check your SD card speed (see page 33). Check speaker wiring, try a new good quality speaker. Charge the battery to full 4.1-4.2V, make sure you are using a recommended battery (see page 37).

For more help please check these links:

ProffieOS/ProffieBoard/TeensySaber wiki on GitHub
Ask your question on The Rebel Armory forums
Ask your question on FX-sabers forums
Ask your questions, share your builds in a facebook group