PHABLABS 4.0

PHOTONICS WORKSHOP

BLINKERLIGHTS

Traffic indicator lights for urban cyclists

DISCLAIMER:

By using this information you agree to be legally bound by these terms, which shall take effect immediately on your first use of the information.

PHABLABS 4.0 consortium and its member organizations give no warranty that the provided information is accurate, up-to-date or complete. You are responsible for independently verifying the information. VUB cannot be held liable for any loss or damage that may arise directly or indirectly from the use of or reliance on the information and/or products provided. PHABLABS 4.0 consortium and its member organizations disclaim all responsibility to the maximum extent possible under applicable laws:

- All express or implied warranties in relation to the information and your use of it are excluded.
- All liability, including for negligence, to you arising directly or indirectly in connection with the information or from your use of it is excluded.

This instruction is published under the Creative Commons licence CC-BY-NC.

PROPERTIES OF THIS WORKSHOP

SUMMARY:

2 LED-strips are attached to a backpack or a high-visibility jacket. The controller can be sewn to the straps. The controller box houses a Feather microcontroller and a battery. The main controller has a small baby buttonbox, that magnetically attaches to a bike's handlebars. The buttons on this little box allow a cyclist to activate the LED-strips as blinking, yellow traffic indicators.

TARGET AUDIENCE:

Young professionals (18+)

SUGGESTED TIME PLANNING: (Total: 6h, this can be performed on one day or in 2 different session of 3h on 2 different days)

Timing in minutes	**activity**
0 - 180	Introduction to Arduino, building the circuit on bread-board & testing the program.
180 - 360	Soldering introduction, building & assembling the controller. Testing final setup.

TOOLS:

Drill press with center drill, 3 mm drill and step drill
Dremel
Soldering Iron

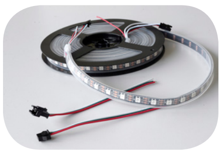
WEBLINK:

All needed files for lasercutting andWemos can be found on:
http://www.phablabs.eu/workshop/photonics-safer-cycling
or via the QR code.

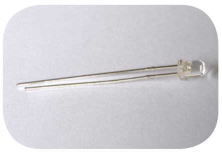
Step 1: Photonics technology

Step 2: Parts list

Photonics Parts:



RGB LED ribbon + 3-pin connectors



Orange LED 3mm
1 pieces

Other Parts:

ABS Black enclosure
1 piece

ABS small enclosure
1 piece

Neodynium magnet 25 mm
1 piece

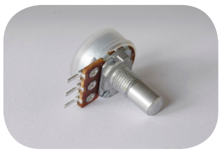
Additional components:

*Electric wire
*Heat shrink sleeves
*Large soft steel washer
*3D printer part to hold magnet - The .stl file can be found here:
http://www.phablabs.eu/workshop/photonics-safer-cycling

Electronic Parts:

Waterproof buttons
2 pieces

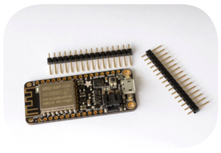
Switch
1 piece



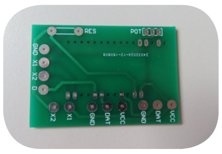
10K Potmeter
1 piece

Resistor 470 Ohm
80 pieces

Resistor 2.2 kOhm
40 pieces



Adafruit Feather HUZZAH
1 piece



PCB
1 piece

3.7V Lipo battery with JST-PH connector
1 piece

4-pin audio socket connector
2 pieces

Coiled 4-pin audio cable
1 piece

Step 3: Parts preparation

The large and small enclosure needs a number of holes. This can be done by a lasercutter, or the participants can do it themselves manually with a drill/dremel.
We recommend cutting these holes ahead of time, as teaching the participants how to use the laser-cutter is not seen as part of this workshop.

The bottom of the small enclosure is a 3D printed part, printed in black PLA.
We recommend printing these ahead of time, and doing one during the workshop as a demo. Alternatively, you could teach the participants how to use a 3D-printer, starting form an existing STL-file. This would add about an hour to the workshop duration, provided you have enough 3D-printers.

The most complex part of the workshop is to get everything to fit nearly inside the enclosure. We highly recommend you build one set yourself before trying to do this as a workshop.

PART 1: BREADBOARDING THE CIRCUIT & INTRODUCTION TO ARDUINO

-You will need to provide a basic 'how to get started with Arduino' instruction.
This introduction should cover the use of buttons/digital inputs, potmeters/analog inputs, the programming interface & structure & the use of outputs.

You will then need to introduce the use of libraries & component-specific code to control the WS2812 LEDs.

-To achieve this, you will build the basic circuit for our indicator light system with the participants & test code step by step.

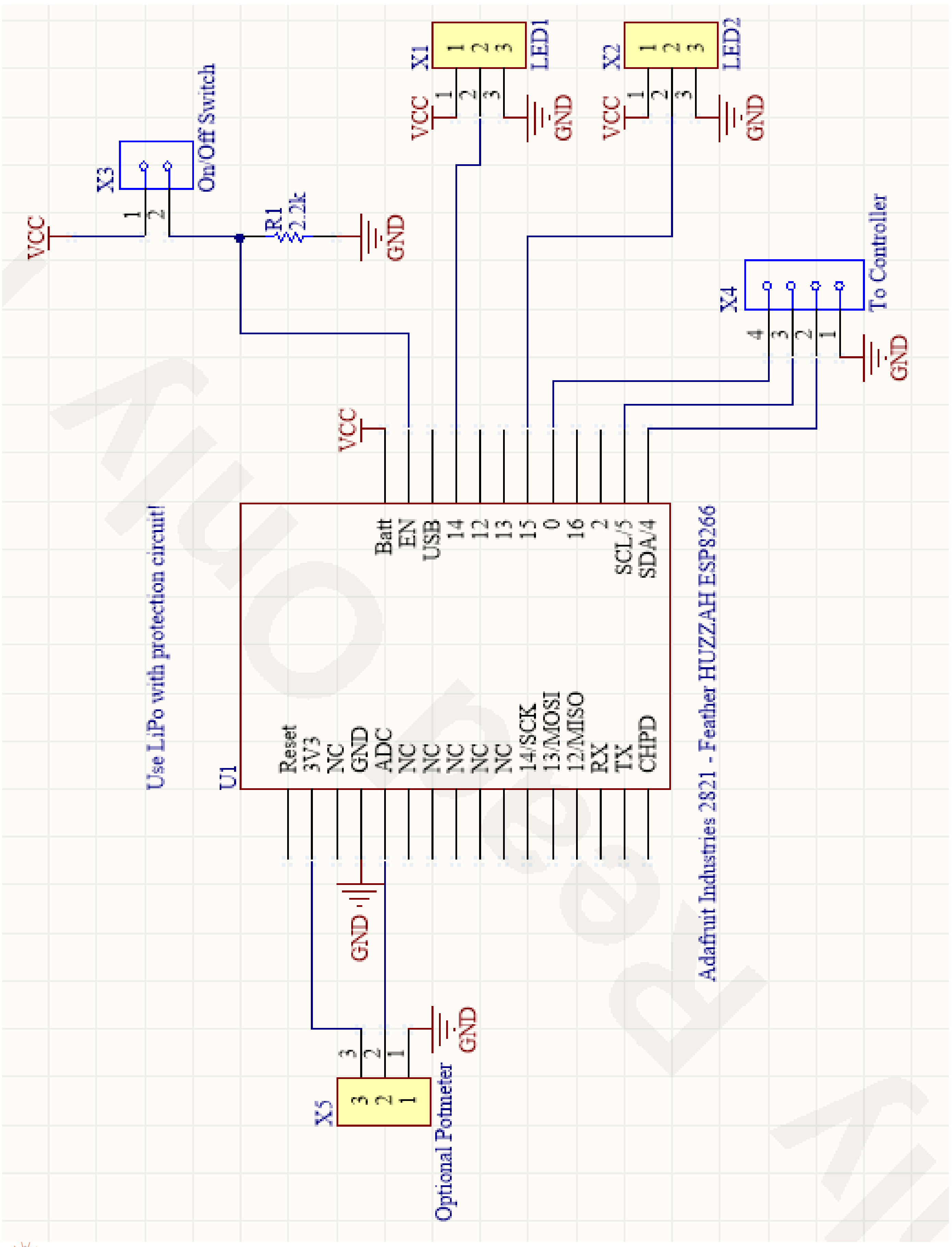
This first part will be mostly step-by-step guided by the instructor in front of the class.

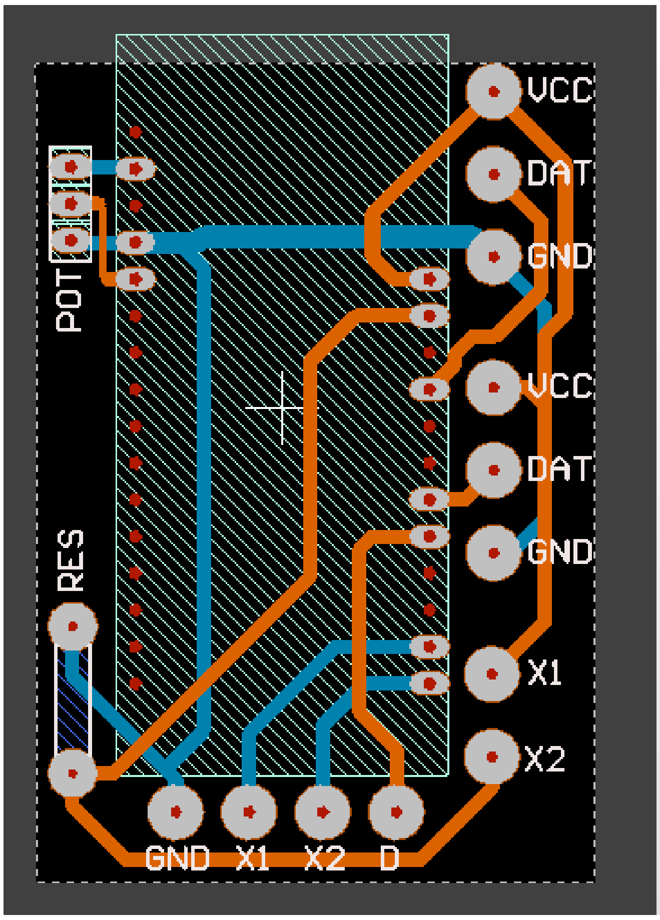
ATTENTION! The potentiometer is optional. With this potentiometer you can control several blinking programmes on the LED strips. But this is not necessary! Without potentiometer only the traffic light indicators will blink.
Depending on the installation of the potentiometer, the correct Arduino programme should be installed.
BikeBlinker01
BikeBlinker01_NoPot

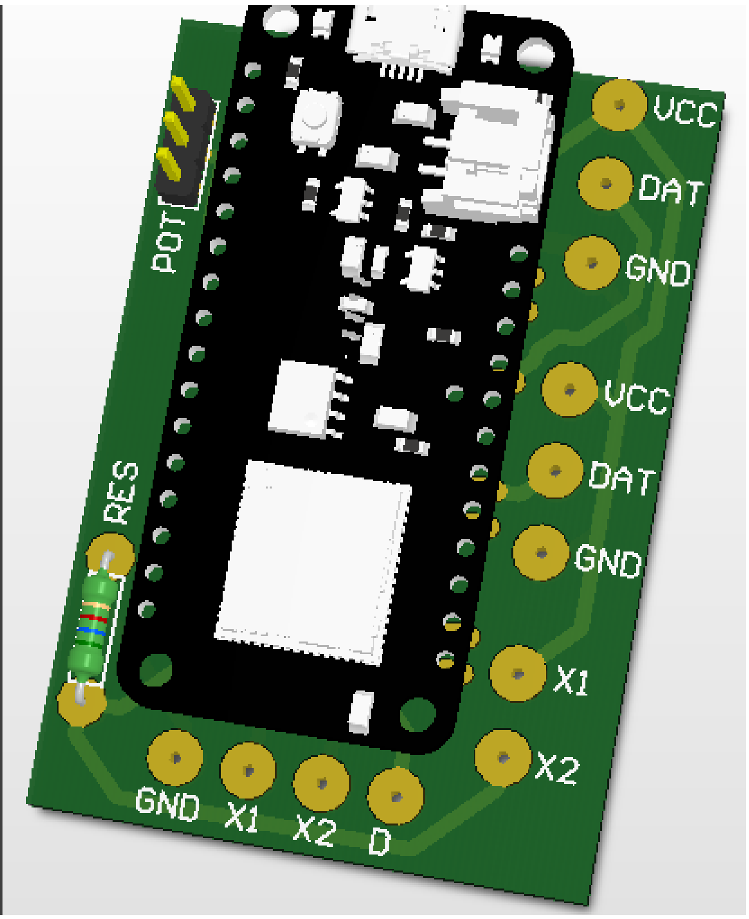
Step 4: Breadboarding the circuit

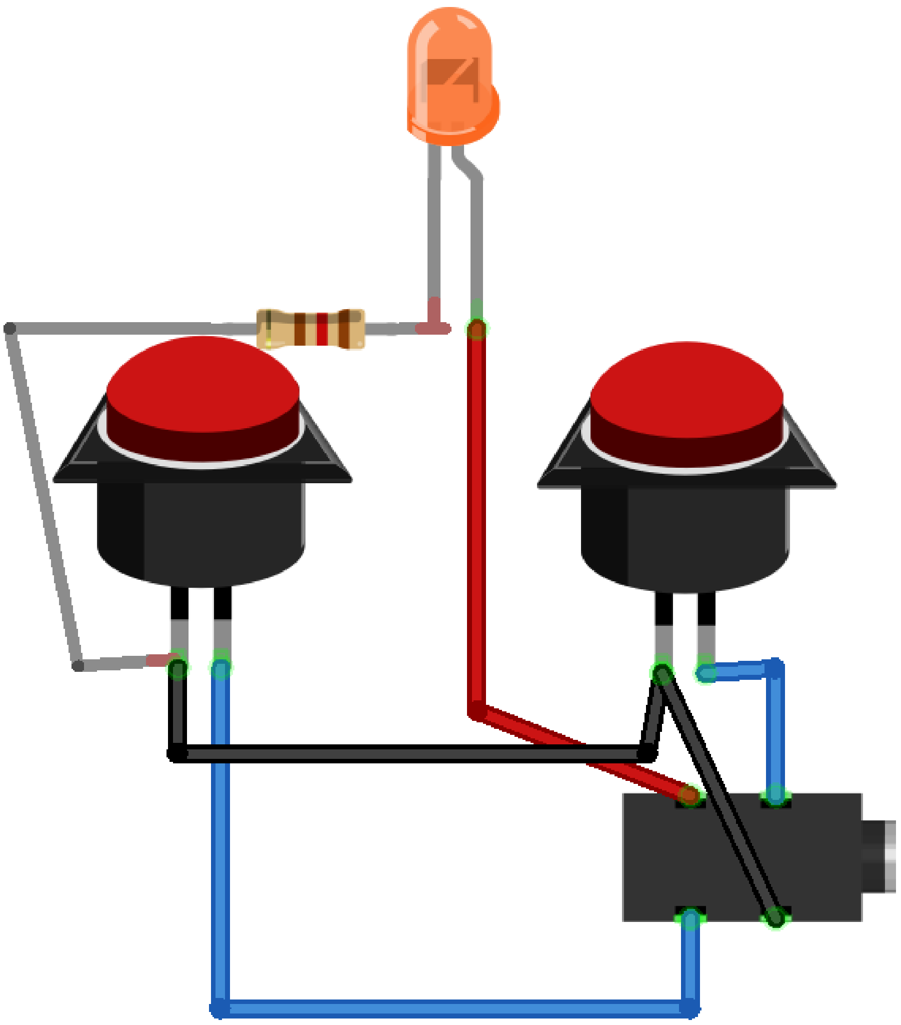
The circuit will be built and tested on a breadboard. This can be done in stages, as each stage allows the instructor to teach part of the circuit & the code.

Electronic scheme:

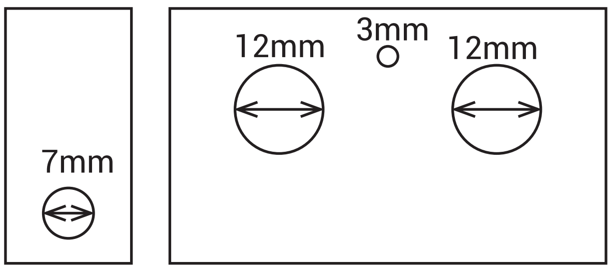



VCC
DAT
GND
VCC
DAT
GND
X1
X2
POT
RES
GND
X1
X2
D

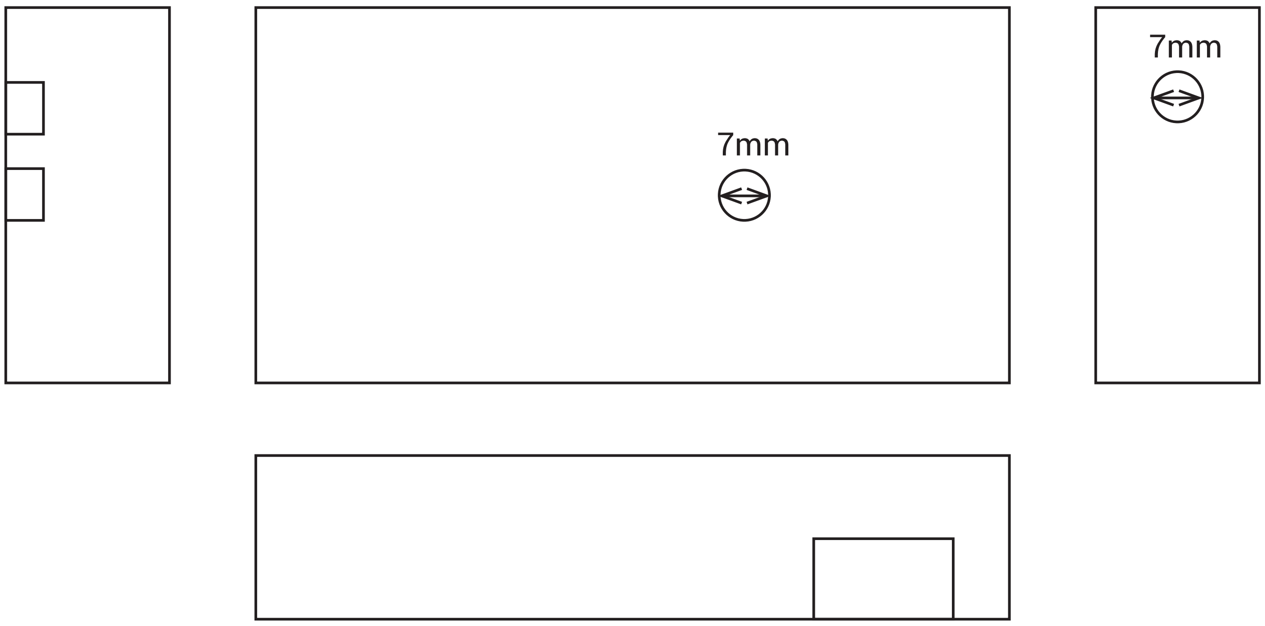

VCC
DAT
GND
VCC
DAT
GND
X1
X2
POT
RES
GND
X1
X2
D



Small enclosure:



Large enclosure:



-The diameter of the holes are indicated. These can be drilled.
-The large enclosue also needs a hole for the switch. This hole is rectangular and can be sawn with a dremel
-The large enclosure also needs a hole for the connector of the Feather Huzzah and another hole for the LED connectors. These can be sawn with a dremel.

You will need to instruct the participants into the correct use of:

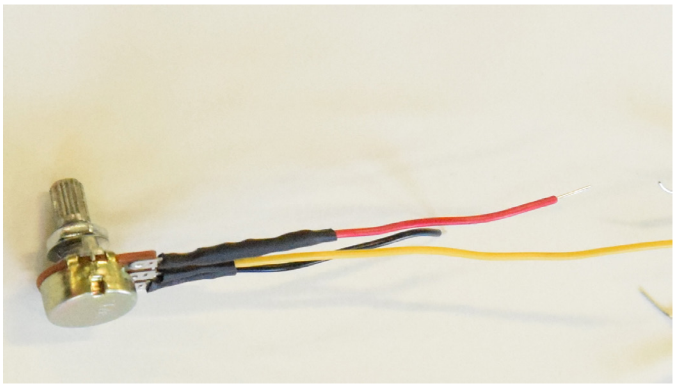
-a soldering iron & the use of a protoboard
-shrink tube, wire strippers and side cutters
-the drill press & the correct drill bits

Participants can then build the circuit & controller box using the provided steps, at their own speed.

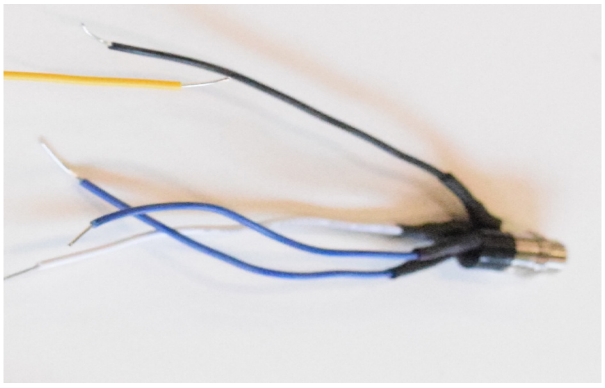
Step 5: Getting started soldering...

This is a good starting point to teach basic soldering skills.

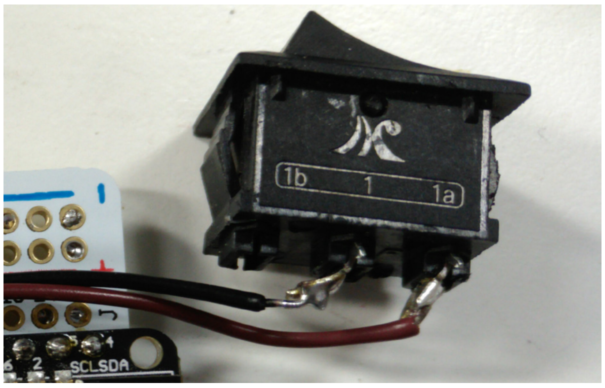
Solder the 8cm-long solid-core wires onto the potmeter. Use red and black wire for the outside connections. Use another colour for the middle (signal) connection. Use shrink tube to insulate the connectors.



To practice your soldering skills you can also wire up the socket connectors. Use black wire to connect to the upper pin (later this will be connected to the GND). Use white wire to solder to the pin underneath. Use blue wires to solder to the outside connections.

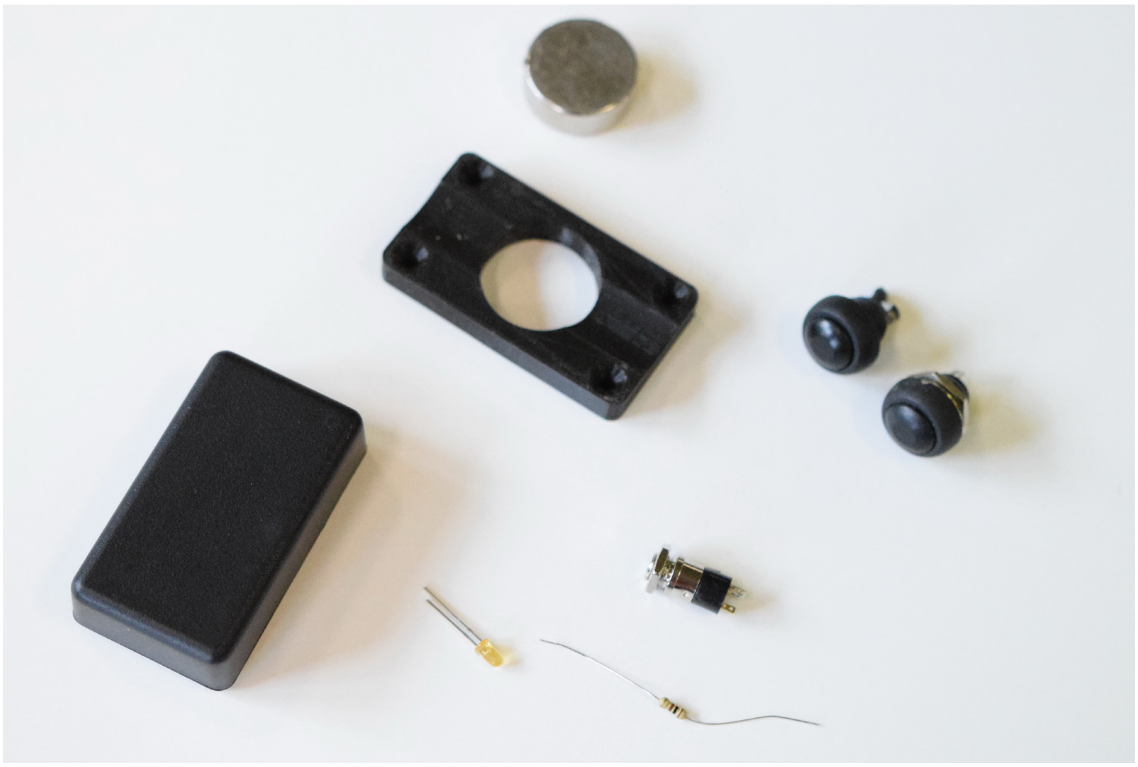


Wire up the switch. Use one black wire and one brown wire.

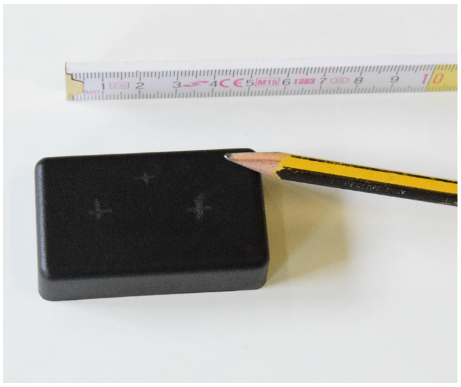
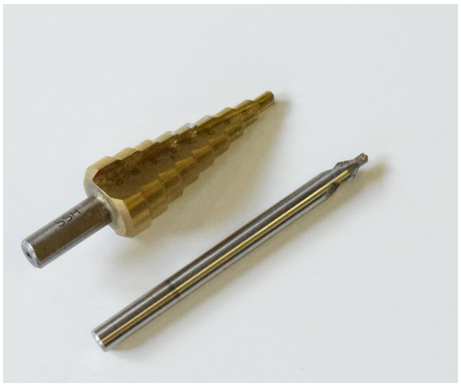
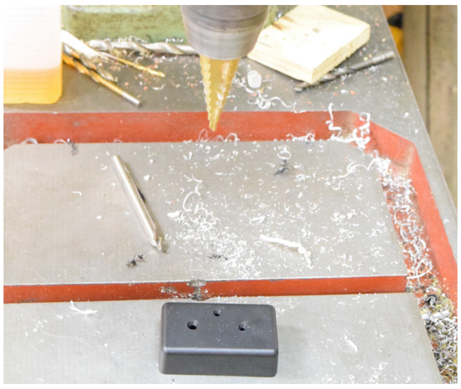


Step 6: Building the small controller box

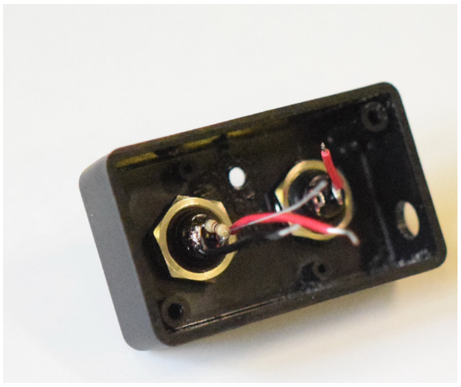
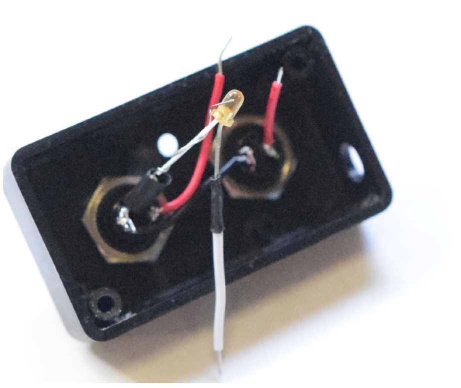
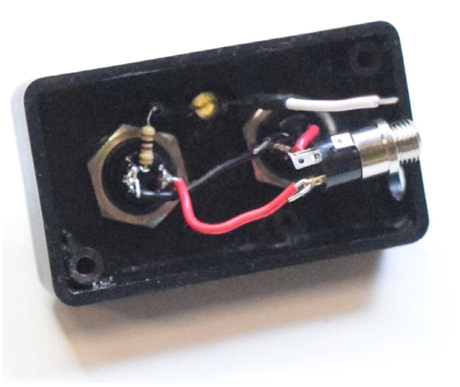
In the next steps you will build the small controller box. You will need the parts shown below:



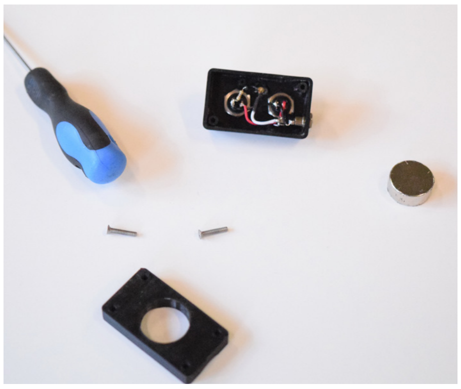
The two water-repellent buttons (hole of 12mm) and the orange LED (hole of 3mm) needs to stick through the top of the small enclosure. Herefore you need to drill 3 holes. Use a pencil and ruler to indicate the holes on the right spot.
The socket connector (hole of 7 mm) should be at the side of the box. Also indicate the hole on the

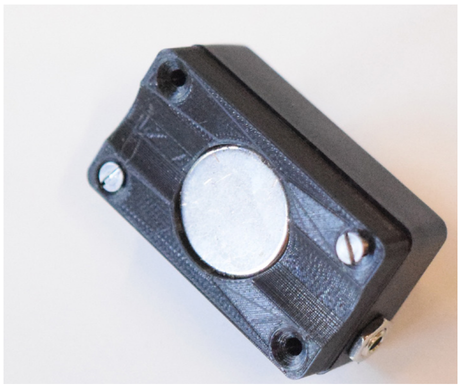




right spot. Use drills of 3mm, 7 mm and 12 mm to make the holes.

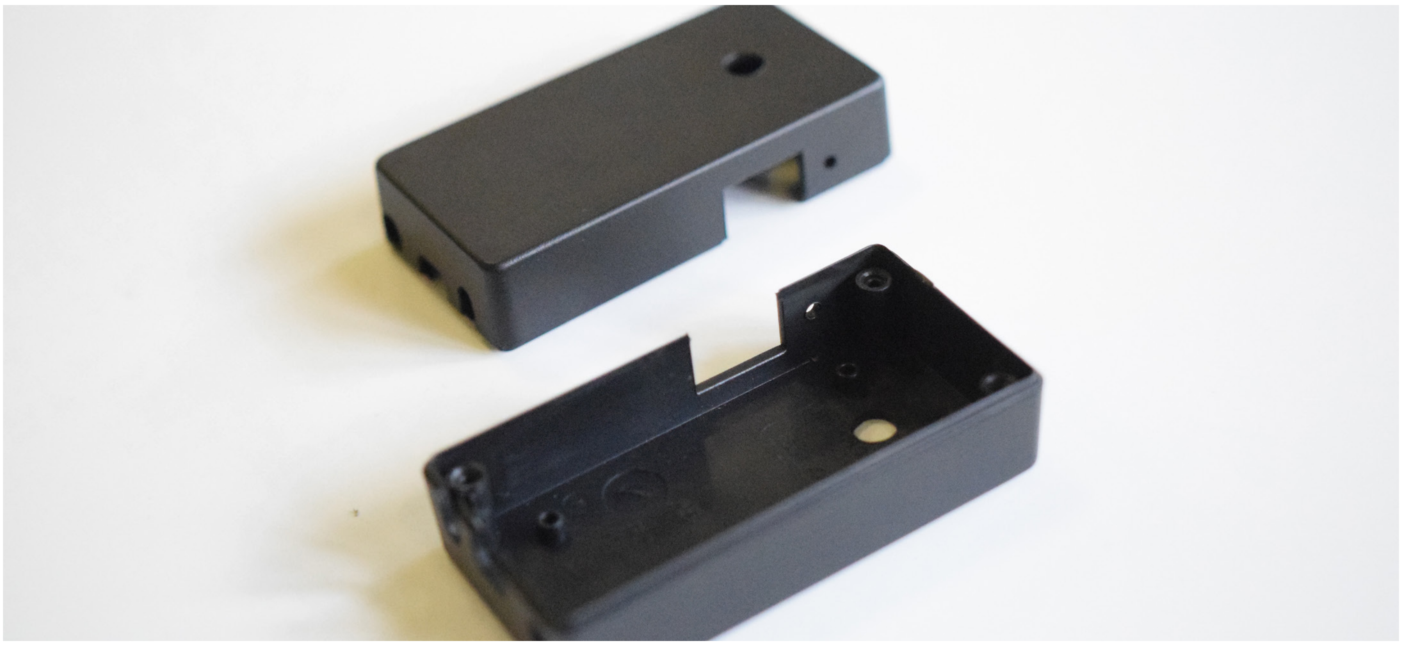
Insert the buttons, the orange LED and the connector. Solder the connections and resistors according to the electrical scheme.
When done soldering, you can close up the small enclosure with the 3D printed part. This part is



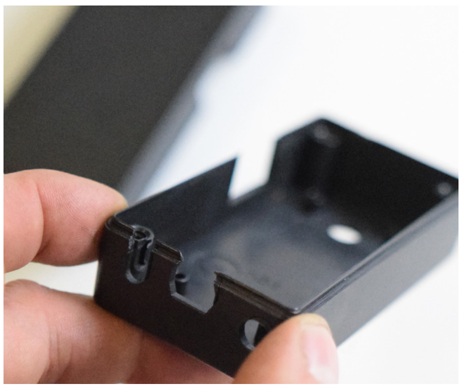
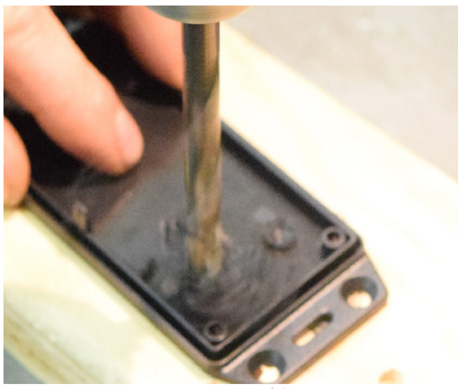


designed to fit a bike handlebars and to contain a magnet. Screw the 3D printed part on top of the small enclosure and push the magnet in the hole.

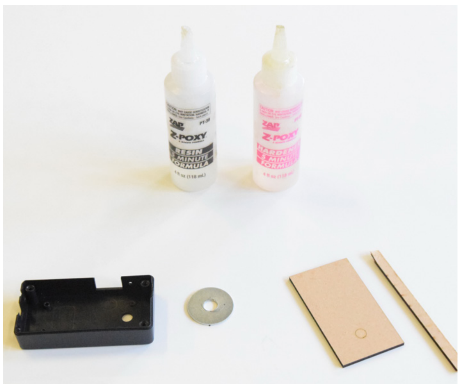
Step 7: Building the large controller box

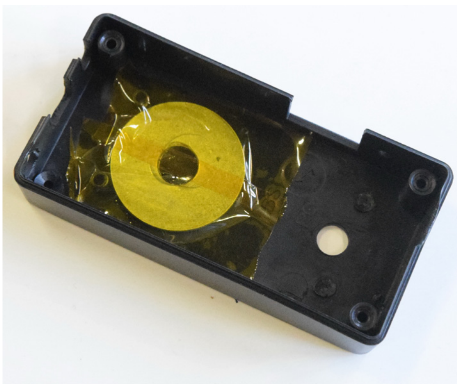


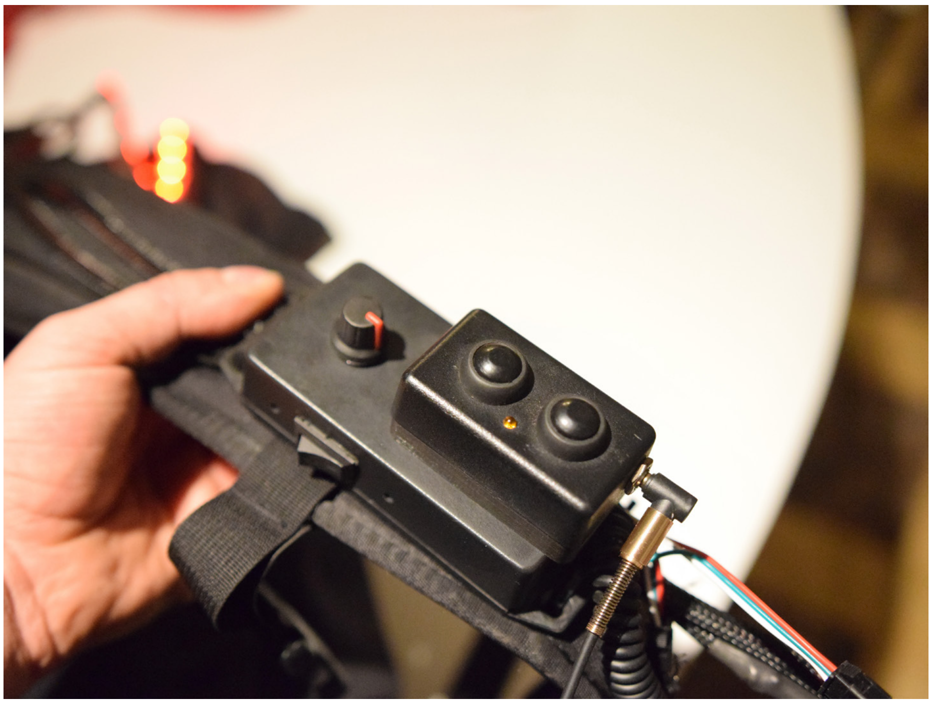
You need to drill 2 holes. One for the potentiometer (hole of 7mm) through the top of the large enclosure and one for the socket connector (hole of 7mm) at the side of the box. Also drill out the upright cilinders of the lid of the box.

Glue the large washer into the large controller box. For this part you will need a prepared enclosure. You also need the large washer, 2-component epoxy glue & a mixing stick. Mix 2 equal parts of the epoxy. Glue the washer in the enclosure. Make sure it is flat, don't use too much glue. Leave the glue to set for 15 minutes, then insulate the washer using Kapton tape. Because of this washer you will able to snap the smaller box on the large box when you are not riding the bike.

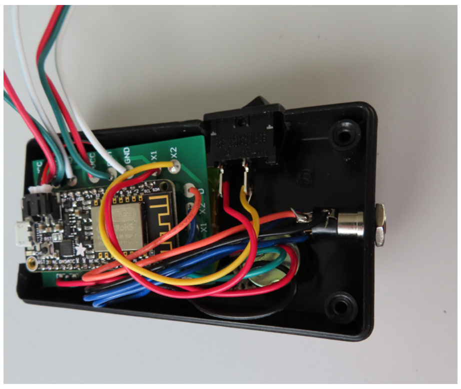






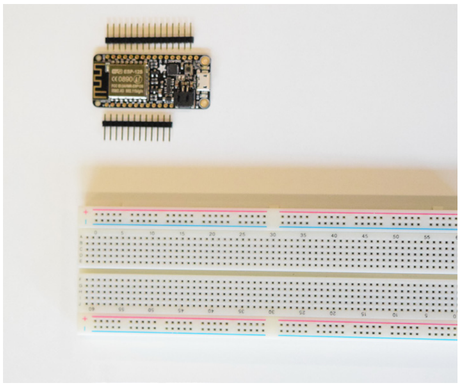
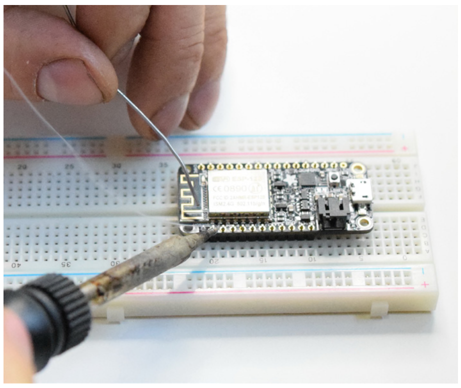
With the dremel you should cut out a square for the switch at the long side of the large enclosure and a hole for the charger of the HUZZAH and the LED connectors at the opposite side of the hole of the socket connector.
All electronic parts fit on the PCB. Mount the electronics board.
You need:
-the adafruit Feather Huzzah board
-the pinholders
-the large enclosure
-thePCB

The board will sit inside the enclosure as shown in the image below.

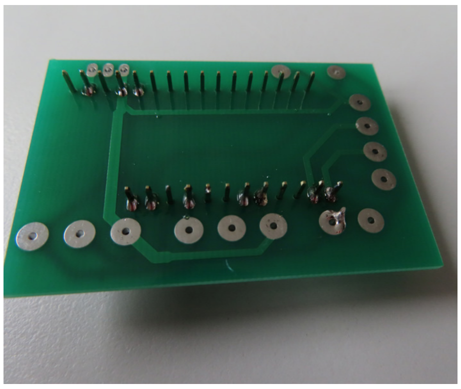
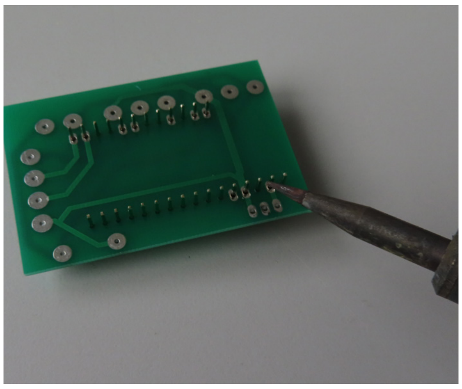
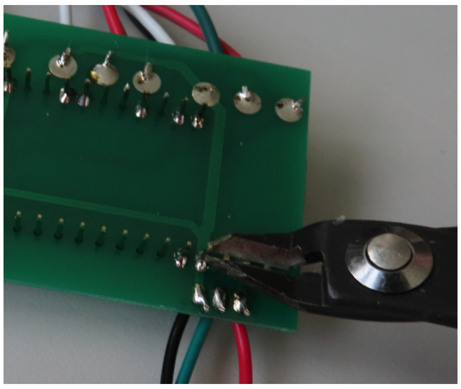


As you can see, the protoboard does not quite fit. Cut out at least one hole for the screw with a knife.

Next solder in the pinheaders in the Feather. So: add pinheaders to all pinout holes. The easy way to do this is to plug everything into a breadboard and solder from the top.

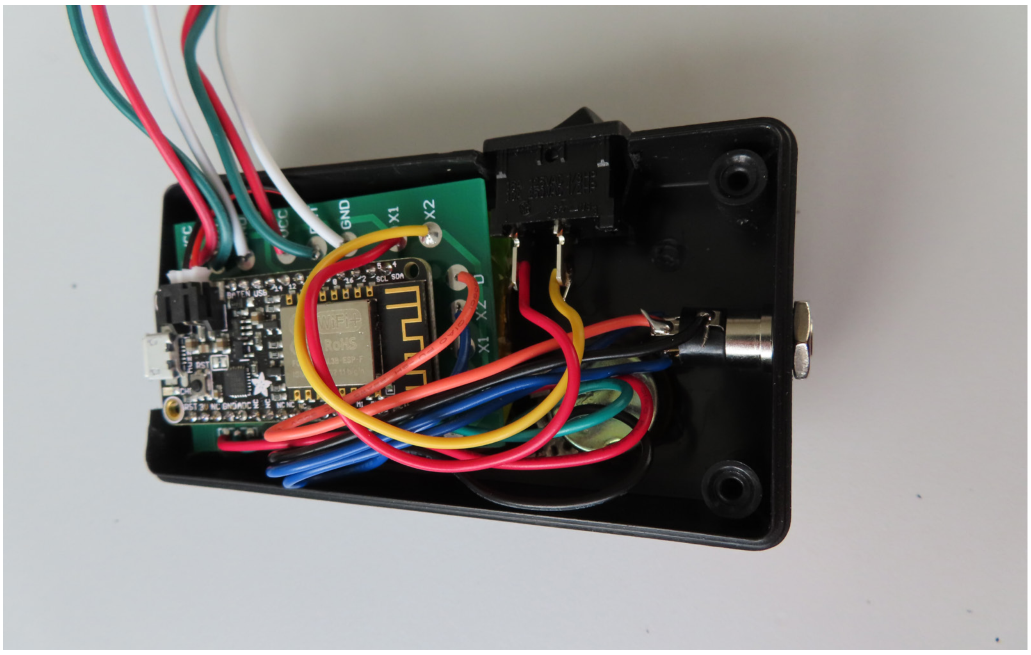



Next, place the Feather on the PCB. There is only one way to solder the Feather correctly on top of the PCB.
Turn the board over and solder in the pinheaders as shown below. Now use the side cutters to cut the pins sticking out of the bottom as flush as possible with the board.

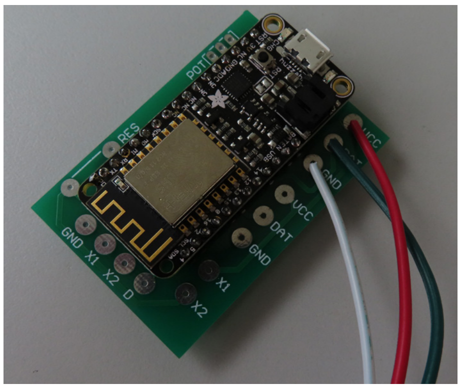
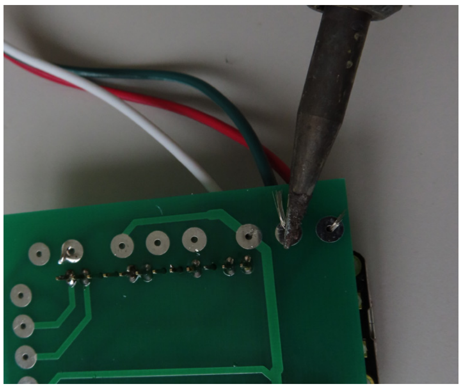
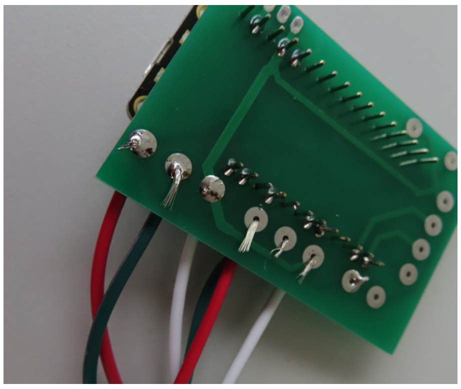




Solder in the wiring & connections.

You will now connect all the parts to the PCB. At the end of this section, your controller should look like the image below.
This is the most tricky step, as you need to make sure your wires are cut to the correct length to make everything fit inside neatly.
Work slowly and do a lot of testfits over the course of the next steps.

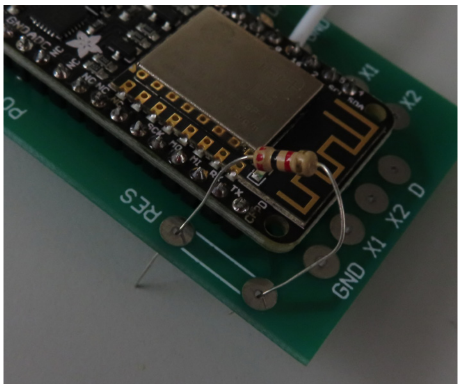
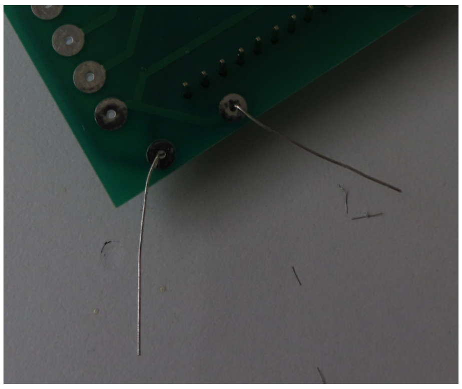


On the right of the PCB, you will find two times: VCC, DAT and GND.
On these holes, you have to solder the LED 3pin-connectors. The red wire (+5V) goes to VCC, the green wire (Dout) goes to DAT and the white wire (GND) goes to GND.

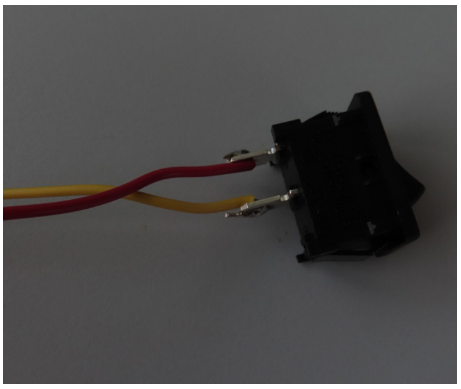
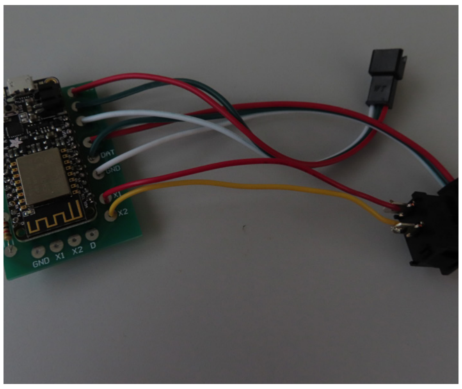
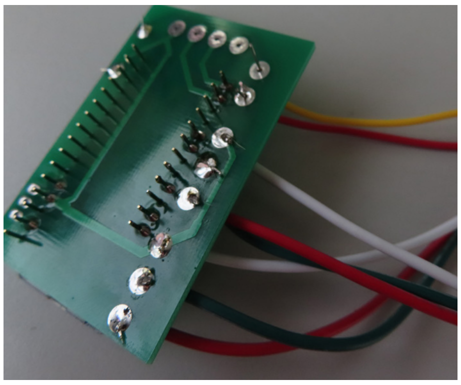




The DAT holes are connected to pin 14 and 15 on the Feather.

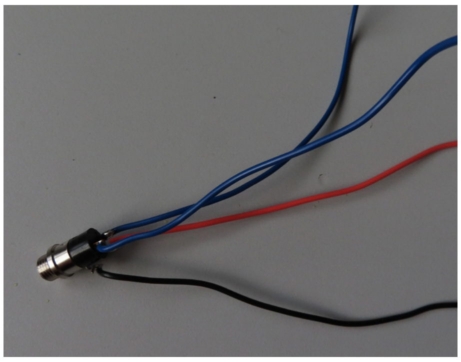
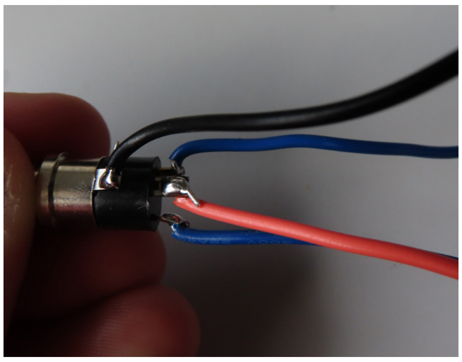
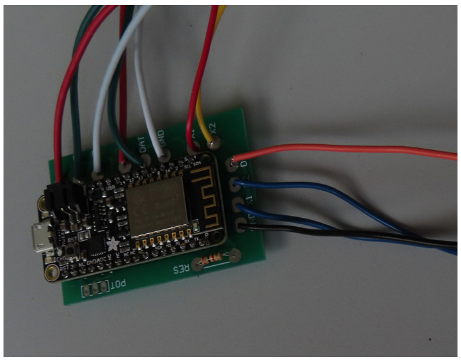
The switch needs a resistor of 2.2kOhm. Solder this resistor on the PCB.The resistor will pull the enable-pin of the microcontroller low when the switch is off. At the on position it will place 3.3V on the Enable pin.
You can also use the X1 pin and a Ground pin for the switch, then you don't need a resistor. But the symbols on the toggle switch will not be correct then.

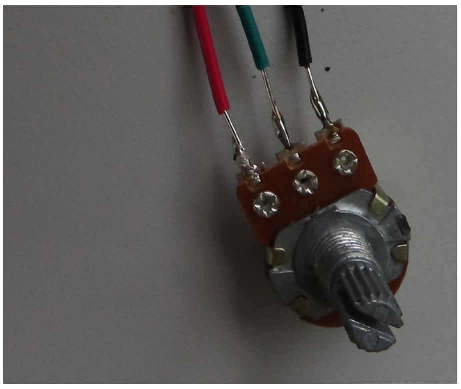
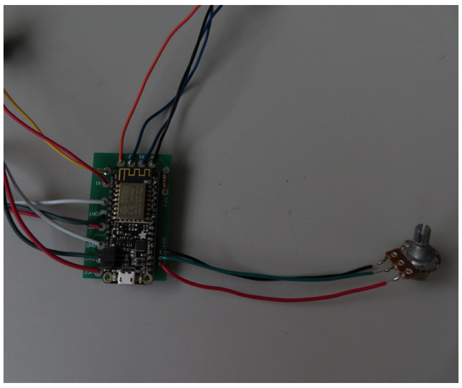
The switch should be soldered to X1 and X2. X1 is connected to both VCC and to +3V. X2 is connected via the resistor to the enable pin.

Next, solder the audio connector to the PCB. Black goes to GND, White goes to D. Blue wires go to X1 and X2, which are connected to pin 4 and pin 5 on the Feather. (Doesn't really matter which one is where, we can switch button assignment in software)

OPTIONAL!
If you have some time left, you can solde the POTmeter to the PCB. You can then programme several programmes to the Feather, and depending how you turn the potmeter, you can lighten the blinkers, or a rainbow effect....

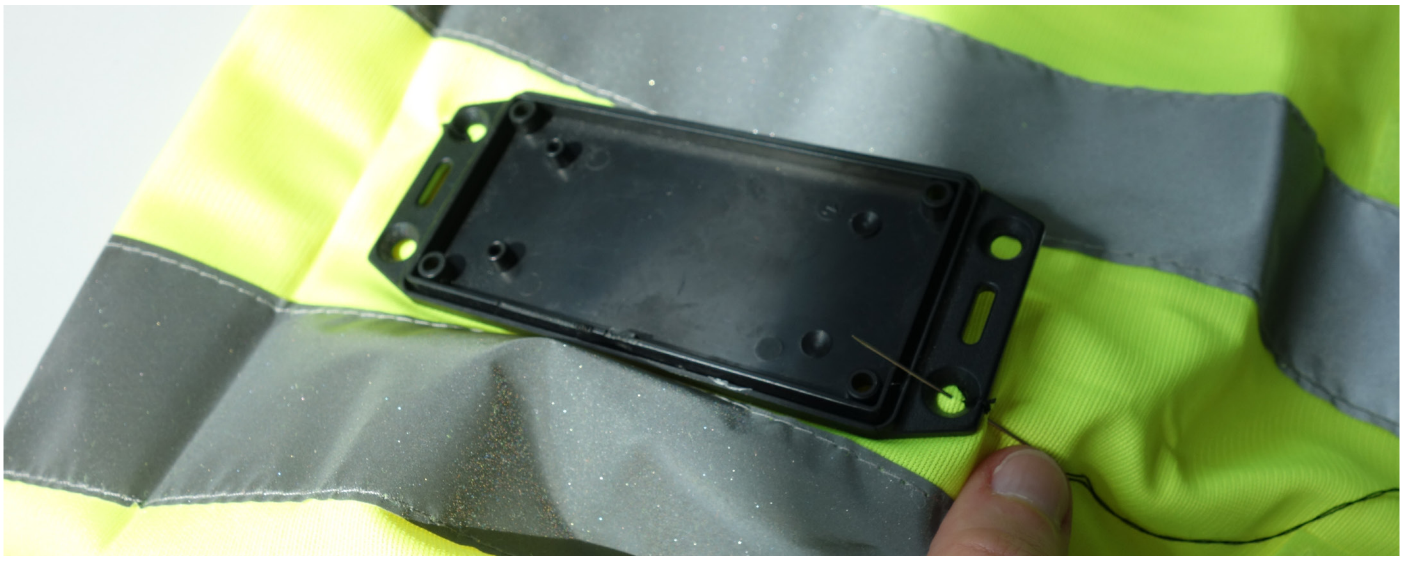



Last but not least: wire in the battery.
The battery has the correct connector to be clicked into the Feather Huzzah.

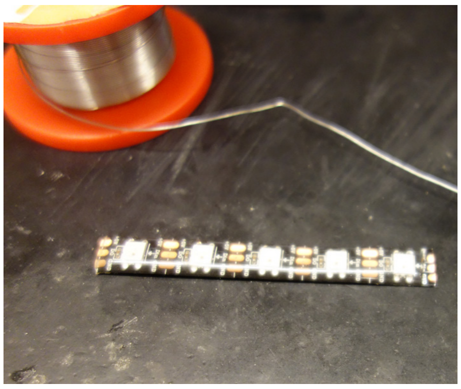
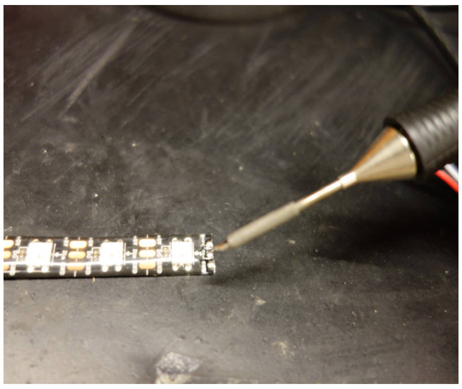
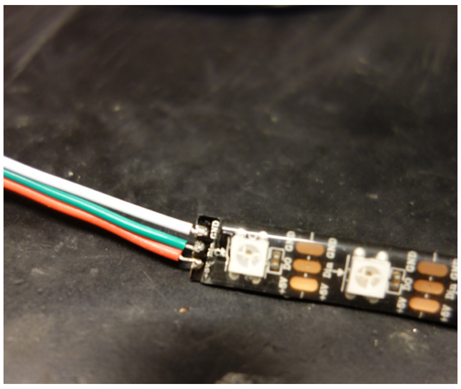
IMPORTANT!
You have to use LiPo's with built-in battery protection. The internal 3.3V regulator of the micro-controller board is not powerful enough to drive all LEDs, therefore the VCC pin of the LED strips is directly connected to the battery.
That means that the built-in shift registers of the individually addressable LEDs will slowly empty the LiPo.
If you do not use it for a week or two, you will have to charge it via the USB port.

You can solve this problem by adding an extra buck-converter with enable pin. But that would be too complex for beginners in soldering workshops.

It is also important to switch off the power switch only when the lights are off. Since the microcontroller is off, the LEDs will keep one specific color mode (you want to keep R GB 0; 0; 0).



Step 7: Connecting the box and LED strips to the backpack/fluorescent vest

Sew the lid of the large enclosure on top of your backpack or vest.
Decide where the LEDs should come and how long the strips should be. Cut off pieces with the

correct size.
Solder the connectors to the LED strips.
Attach pieces of velcro to the LED strips and on the correct places on your backpack/vest.

Step 8: Programming Arduino

The PCB used in this workshop is not really an Arduino board, but a feather HUZZAH. To use this arduino progamme for this, you should first download additional info to be able to upload information on it.
Feather HUZZAH ESP8266: WiFi built-in battery charging for IoT on-the-go!
You can find 'How to' install this via this link:
https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/using-arduino-ide

If this is all done, you can open the file ('bikeBlinker01') attached to this mail. Connect the Feather Huzzah to your computer. Check via 'tools' if the correct board is selected:
Board: "Adafruit HUZZAH ESP8266"
Select the right port.
Then push the upload button (the arrow pointing to the right)
When it is succefully uploaded, you can disconnect the feather huzzah and use the blinkers.

```
/*
  Feather Huzzah, compiled in Arduino 1.8.1
  Buttons on pins 4 and 5
  Buttons are pull-down to ground


plugsocket - starting from pointy end:
pin4/blue -pin5/green- Red (Led?)- copper (gnd?)


  Pot on A0
  Neopixel leftstrips on 14 and 15
  LED on 0
```

This indicates the pins used on the Feather Huzzah

```
  Version 2019-02_08

//connector on

*/

#include <Adafruit_NeoPixel.h>
```

This indicates the library you should first download

```
#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIN 14
#define PIN2 15

// Parameter 1 = number of pixels in leftstrip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
Adafruit_NeoPixel leftstrip = Adafruit_NeoPixel(27, PIN, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel rightstrip = Adafruit_NeoPixel(27, PIN2, NEO_GRB + NEO_KHZ800);

byte blinkMode = 0;
byte previousBlinkMode = 0;


int button4Toggle = 0;
int button5Toggle = 0;


void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you are not using a Trinket
#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
  delay(100);
  // End of trinket special code
  pinMode(PIN, OUTPUT);
  pinMode(PIN2, OUTPUT);
  pinMode(0, OUTPUT);
  blinkMode = 1;
  indicateBlinkMode();
  pinMode(0, OUTPUT);
  pinMode(4, INPUT_PULLUP); //Pushbutton 1
  pinMode(5, INPUT_PULLUP); //Pushbutton 2


  leftstrip.begin();
  leftstrip.show(); // Initialize all pixels to 'off'
  rightstrip.begin();
  rightstrip.show(); // Initialize all pixels to 'off'
  delay(1); //delays needed to keep ESP2866 stable, not needed for other boards
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  //  colorWipe(leftstrip.Color(255, 0, 0), 50); // Red
  // colorWipe(leftstrip.Color(0, 255, 0), 50); // Green
  //colorWipe(leftstrip.Color(0, 0, 255), 50); // Blue
  delay(1); //delays needed to keep ESP2866 stable, not needed for other boards
  if (analogRead(A0) < 200) {
    //this mode is traffic indicator lights
    blinkMode = 1;
    indicateBlinkMode(); // this function makes the LED blink to indicate the current mode

    colorSet(0);
    digitalWrite(0, LOW);
    //main program option: push buttons for blinkers
    if (digitalRead(5) == LOW ) {
      yellowBlink(leftstrip, 800);
    }

    if (digitalRead(4) == LOW) {
      yellowBlink(rightstrip, 800);
    }
  }

  if (analogRead(A0) > 200 && analogRead(A0) < 412 ) { //This is red lights mode
    blinkMode = 2;
    indicateBlinkMode();

    colorSet(0);
    digitalWrite(0, LOW);
    updateToggles();

    if (button4Toggle == 1) {

      digitalWrite(0, HIGH);
      delay(5);
      digitalWrite(0, LOW);
      colorWipeRed( 150);

      digitalWrite(0, LOW);
      digitalWrite(0, LOW);
    }

    if (button5Toggle == 1) {
      digitalWrite(0, HIGH);
      delay(5);
      digitalWrite(0, LOW);
      softGlowRed(40);

    }
  }

  if (analogRead(A0) > 812) {
    //This is all-white LEDS on right button or 1 chasing LED on button 5
    blinkMode = 4;
    indicateBlinkMode();
    if (digitalRead(4) == LOW) {

      digitalWrite(0, HIGH);
      delay(100);
      digitalWrite(0, LOW);
      colorWipe2( 150);

      digitalWrite(0, LOW);
      digitalWrite(13, LOW);
    }
    if (digitalRead(5) == LOW) {
      softGlow(40);

    }
  }

  if (analogRead(A0) > 412 && analogRead(A0) < 810) {
    blinkMode = 3;
    indicateBlinkMode();
    if (digitalRead(4) == LOW) {
      digitalWrite(13, HIGH);
      delay(100);
      digitalWrite(13, LOW);
      delay(100);
      rainbowCycle(50);
      //  theaterChase(leftstrip.Color(127, 127, 127), 50); // White
      digitalWrite(0, LOW);
    }
  }

}

void indicateBlinkMode() {
  //this function blinks the LED when you turn the potmeter to select a different mode.
  // The LED blinks x times where x is the number of the blinkMode
  if (blinkMode != previousBlinkMode) {
    for (int k = 0; k <= blinkMode; k++) {
      digitalWrite(13, HIGH);
      delay(150);
      digitalWrite(13, LOW);
      delay(150);
    }

  }
  previousBlinkMode = blinkMode; //this makes sure the LED only blinks when whe change mode, not every
loop when the function is called

}

void updateToggles() {   //this function reads the state of the buttons and changes the togglevalues button-
4Toggle and button5Toggle when the buttons are pressed
  //this allows us to turn stuff on and off with short presses of the buttons

  if (digitalRead(4) == LOW) {
    if (button4Toggle == 0) {
      button4Toggle = 1;
    }
    else {
      button4Toggle = 0;
    }
  }
  if (digitalRead(5) == LOW) {
    if (button5Toggle == 0) {
      button5Toggle = 1;
    }
    else {
      button5Toggle = 0;
    }
  }


}
```

Congratulations! You're done! From now on, you can start safely cycle through traffic.

Step 9: End result & conclusions

On the use of this controller:

-To charge the battery, plug the controller into a regular microUSB 5V phone charging cable & toggle the main switch on the controller to the ON position.
We like the charging feature on this board, but it is slow (100mA charge current).
To fully charge the empty battery you will need to leave it plugged in overnight.

PHABLABS 4.0

PHABLABS 4.0 is a European project where **two major trends** are combined into one powerful and ambitious innovation pathway for digitization of European industry:
On the one hand the growing awareness of **photonics** as an important innovation driver and a **key enabling technology** towards a better society, and on the other hand the **exploding network of vibrant Fab Labs** where next-generation **practical skills-based learning** using KETs is core but where photonics is currently lacking.

www.PHABLABS.eu

This workshop was set up by the *Brussels Photonics Team, Vrije Universiteit Brussels* in close collaboration with *Fab Lab Brussel.*

PHOTONICS PUBLIC PRIVATE PARTNERSHIP