CRUMAR®

DRAWBAR CONTROLLER D9U

An open-source Do-it-yourself
project based on Arduino.

MOUNTING INSTRUCTIONS

Welcome to the mounting instruction sheet for the Crumar D9U Drawbar Controller kit. To assemble this kit correctly and set it up to work properly, a certain skill with electronics and computers is required, plus some tools and a little bit of patience and attention.

REQUIRED TOOLS

1. Soldering iron, preferably a temperature-controlled 60W iron with a 1,5 ~ 2,5 mm wide tip;
2. Solder, preferably good quality 0,8 ~ 1 mm diameter;
3. Good quality cutters;
4. Phillips screwdriver;
5. A computer with Arduino IDE installed.

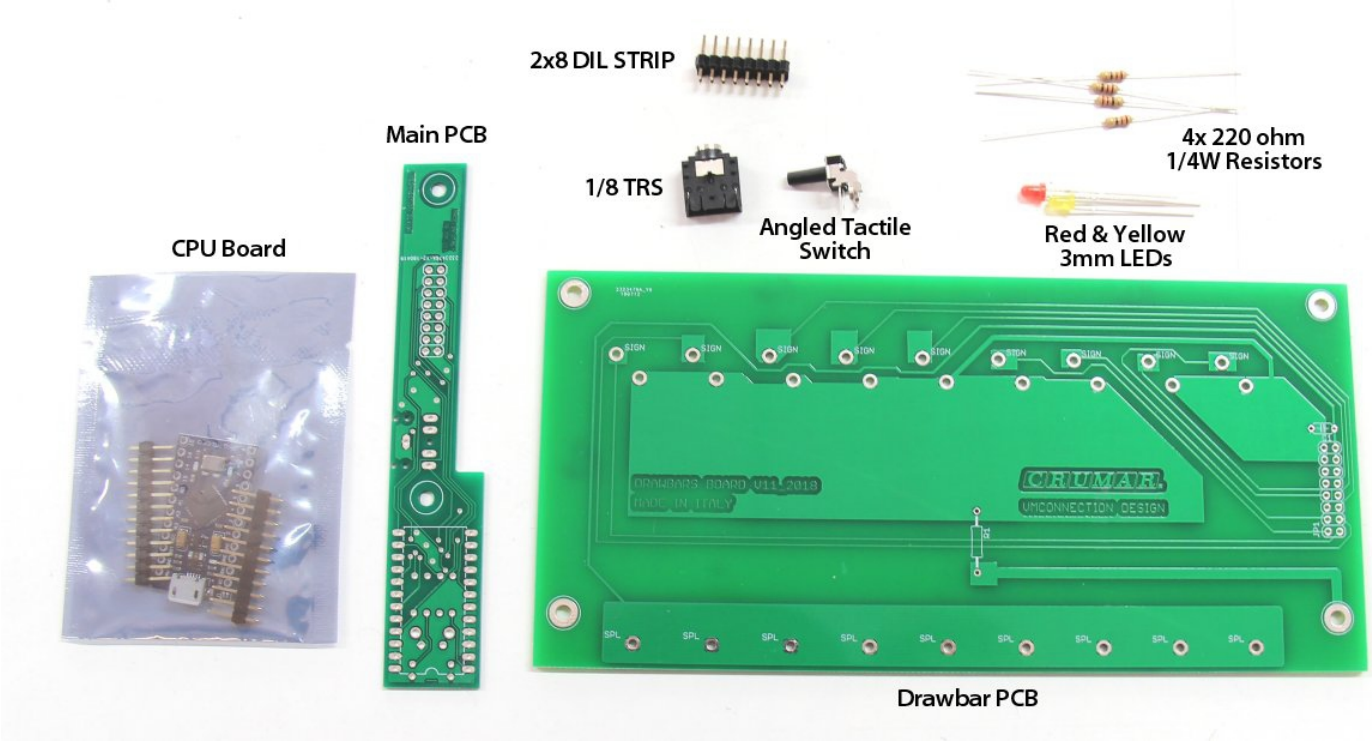
PREPARATION

Prepare a clean and tidy surface, with just the required tools handy and make sure you have discharged your body from electrostatic charge by touching some metal object that makes contact with the floor. Optionally, wear an ESD wristband.

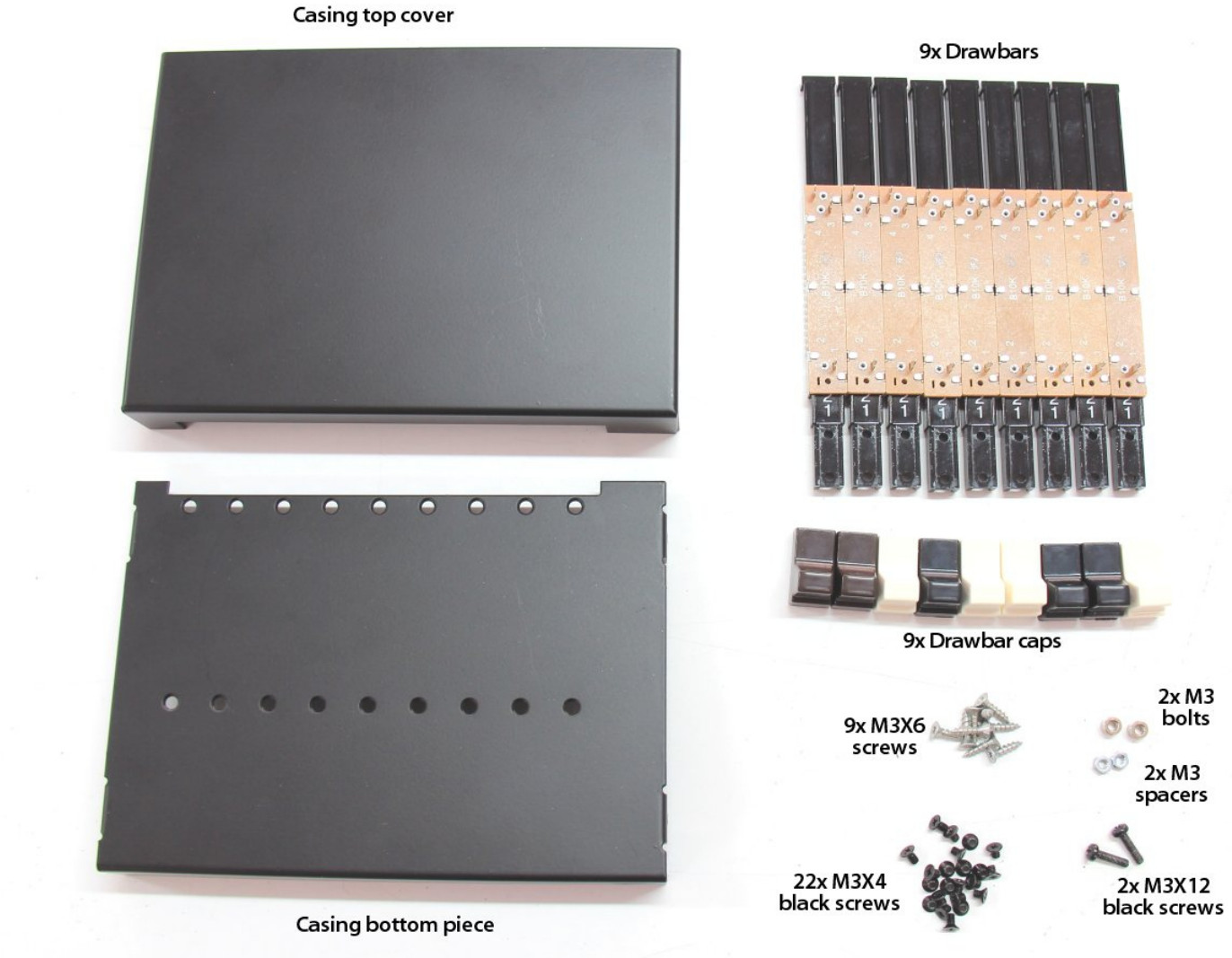
WHAT'S IN THE KIT

- 1x PCB Drawbars v11_2018
- 1x PCB DB_CM_V11
- 1x Arduino Pro Micro (Leonardo clone)
- 1x Metal bottom piece
- 1x Metal top cover
- 9x Drawbars
- 2x Brown Caps
- 3x Black Caps
- 4x White Caps
- 4x 220ohm 1/4W resistors
- 1x Red 3mm LED
- 1x Yellow 3mm LED
- 1x 2x8 DIL Strip
- 1x TRS 3.5mm Jack connector
- 1x Angled Tactile Switch
- 2x M3X12 Black screws
- 2x M3 Spacers
- 2x M3 Bolts
- 22x M3X4 Black screws
- 9x M3X6 screws
- 1x USB – MicroUSB cable
- 4x Adhesive rubber feet

Electronic parts

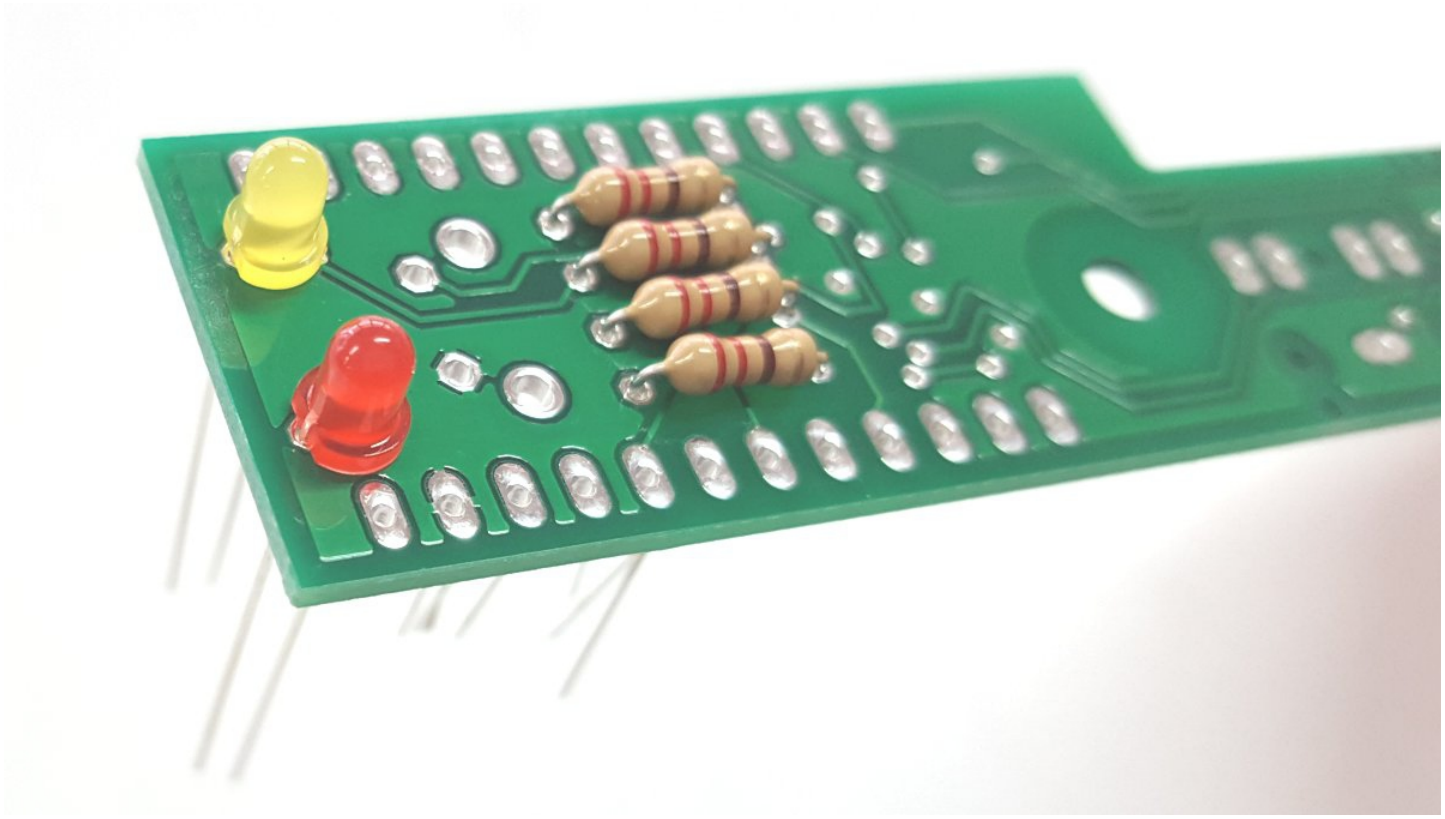


Mechanical parts



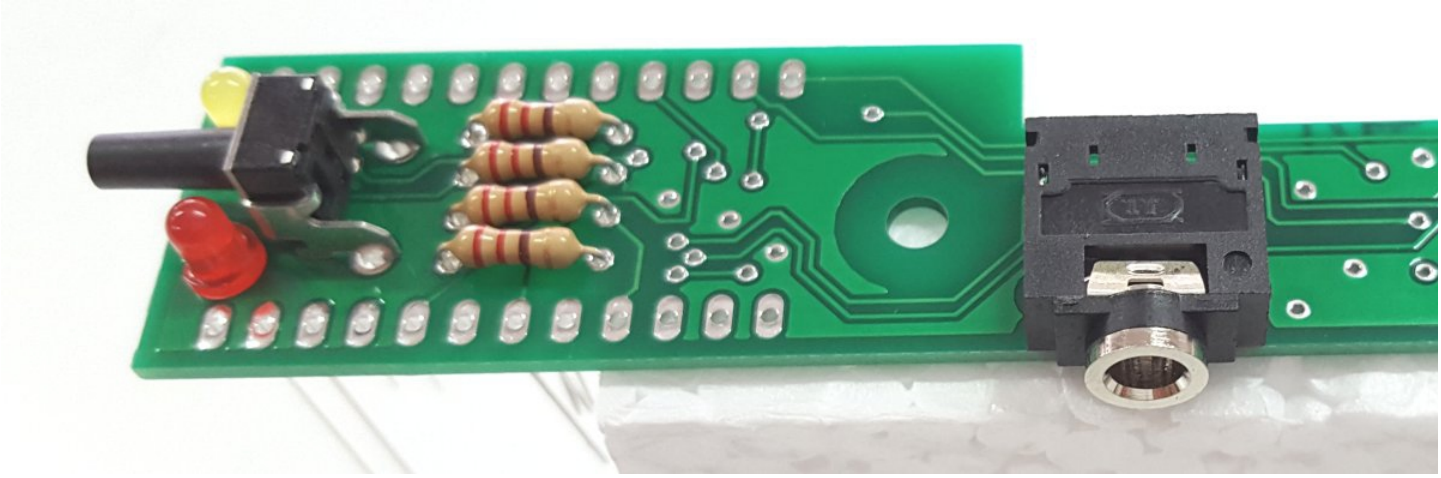
STEP 1: Let's start from the electronics. All passive components, first, starting from the lowest till the tallest. Take the four resistors, bend the terminals 90 degrees and add them to the Main PCB as shown in the pictures below.

Also add the LEDs, pay attention to the polarity and the posisions. The shorter lead towards the edge of the board, yellow on the right, red on the left.

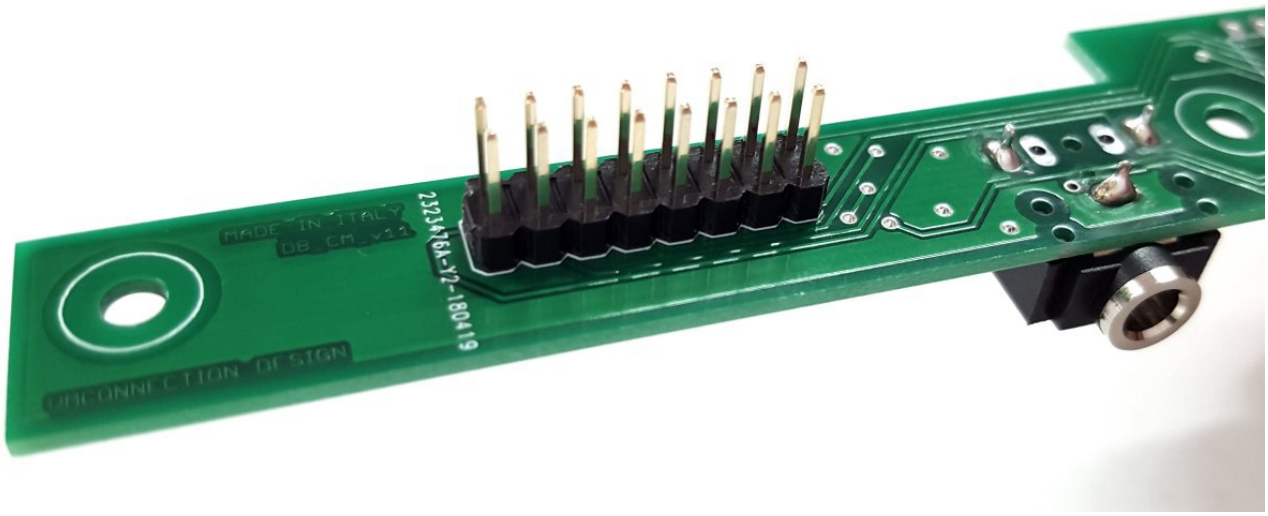


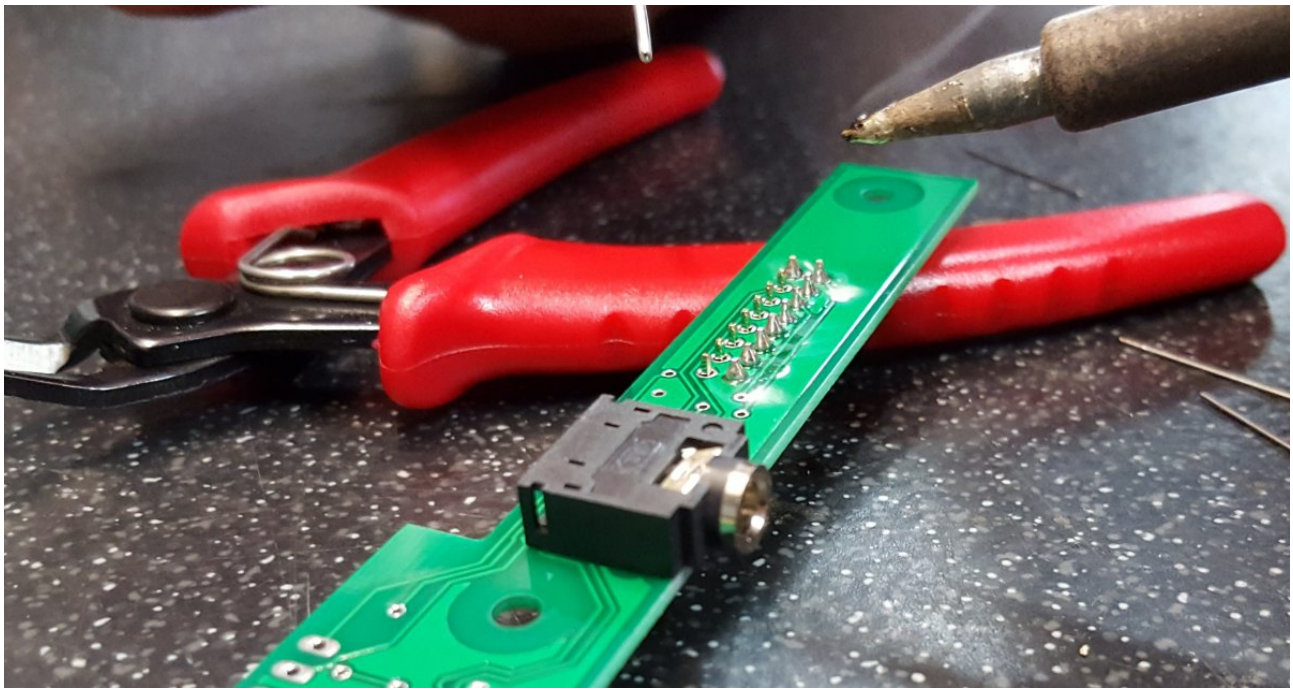
Proceed with the soldering of these first components and cut the exceeding leads.

STEP 2: Add the angled tactile switch and the mini TRS Jack.

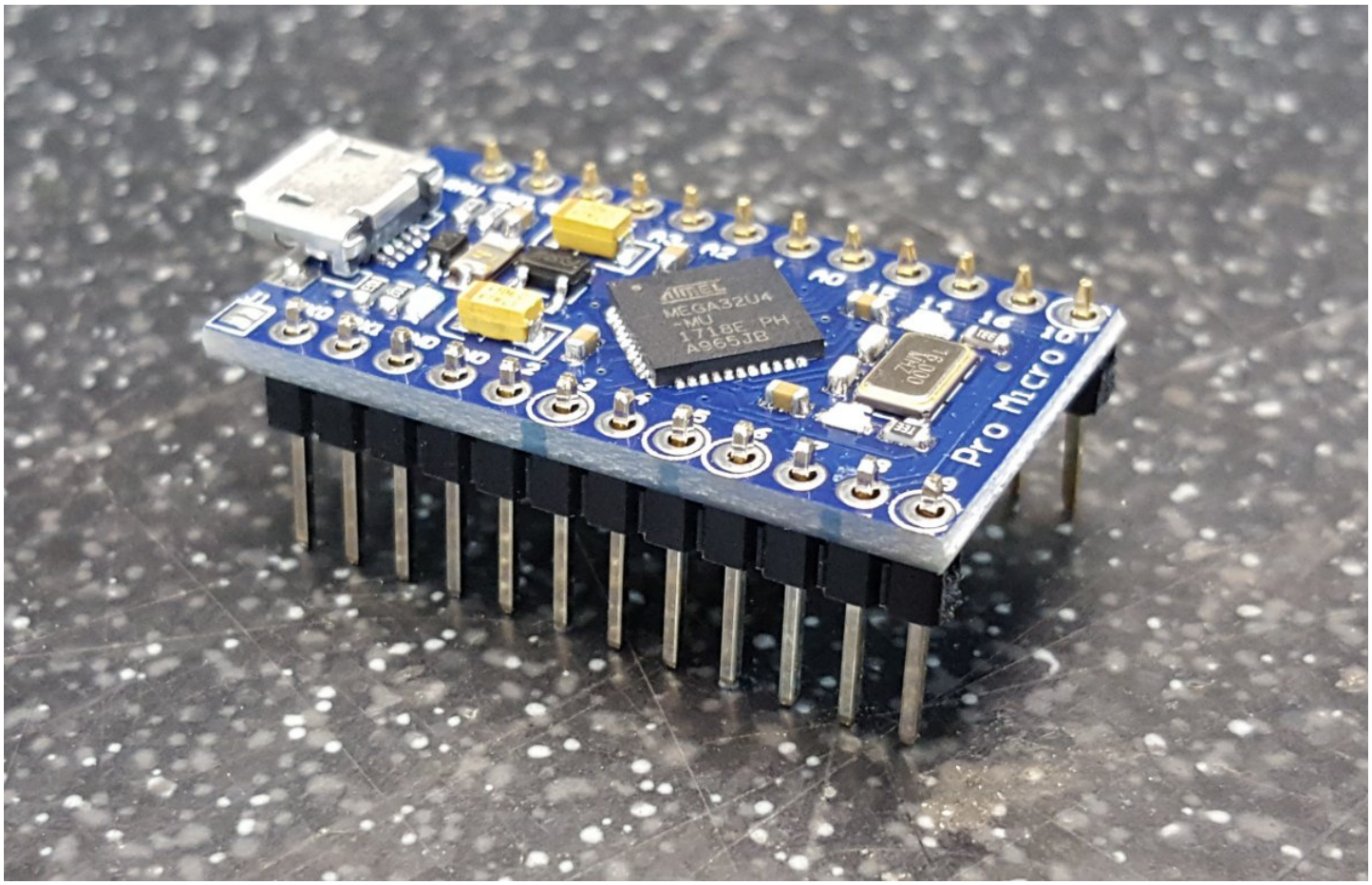


STEP 3: turn the PCB upside down and add the 2x8 DIL strip, as shown in the following picture. Pay attention to have it set straight on the board or you'll have troubles when coupling this board with the drawbar board.

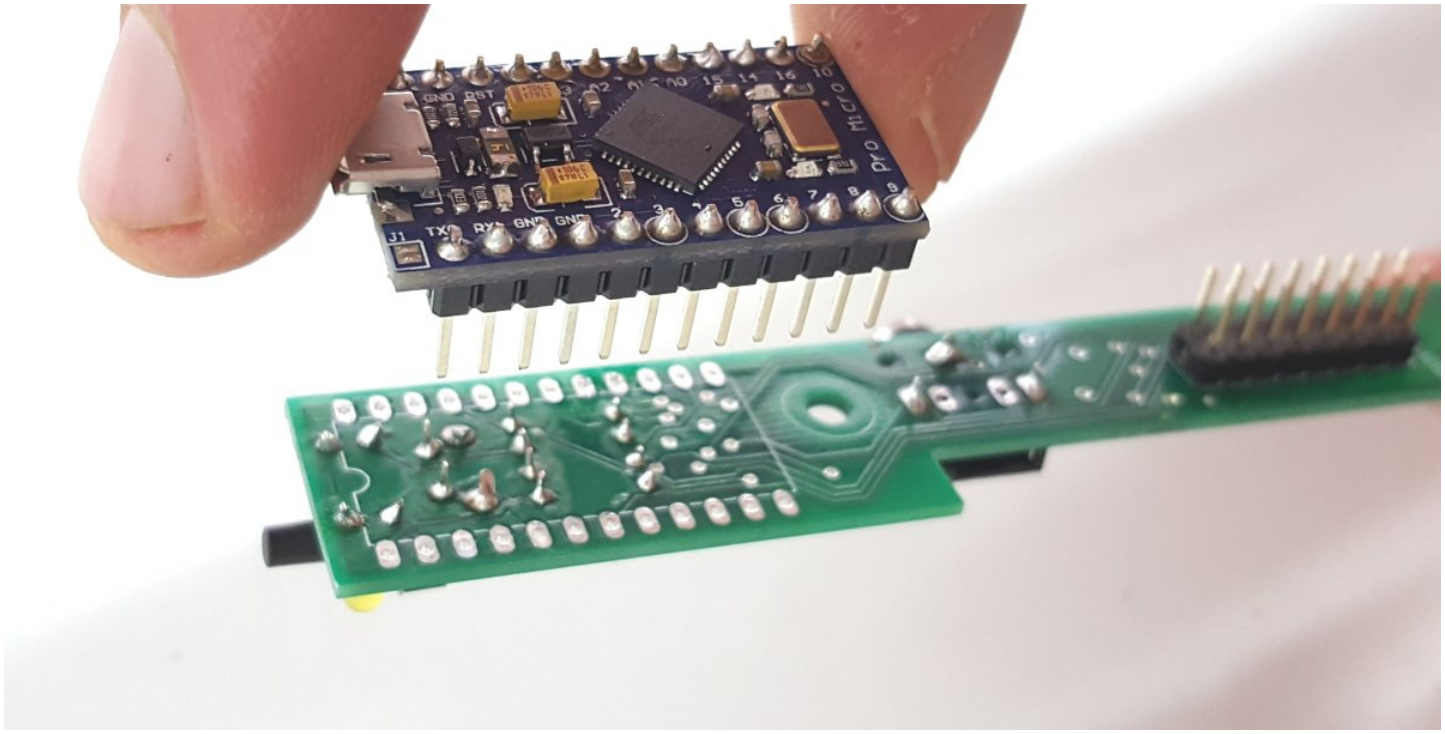




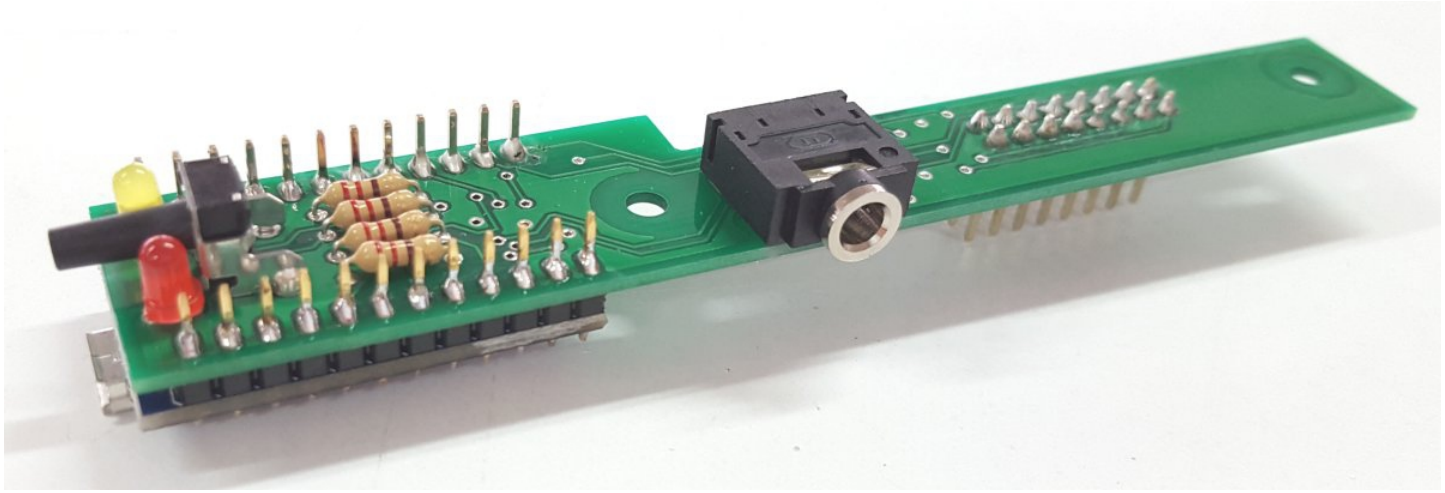
STEP 4: Take the CPU board and solder the SIL terminals to the tiny board. This board contains active SMT electronics and is subject to ESD and overheat, make sure that your soldering iron doesn't exceed 350 °C (~660°F) and don't keep the soldering pin on the terminals for too long. The SIL strip must face the bottom of the board, thus the pins will be soldered on the top side, as shown in the picture below.



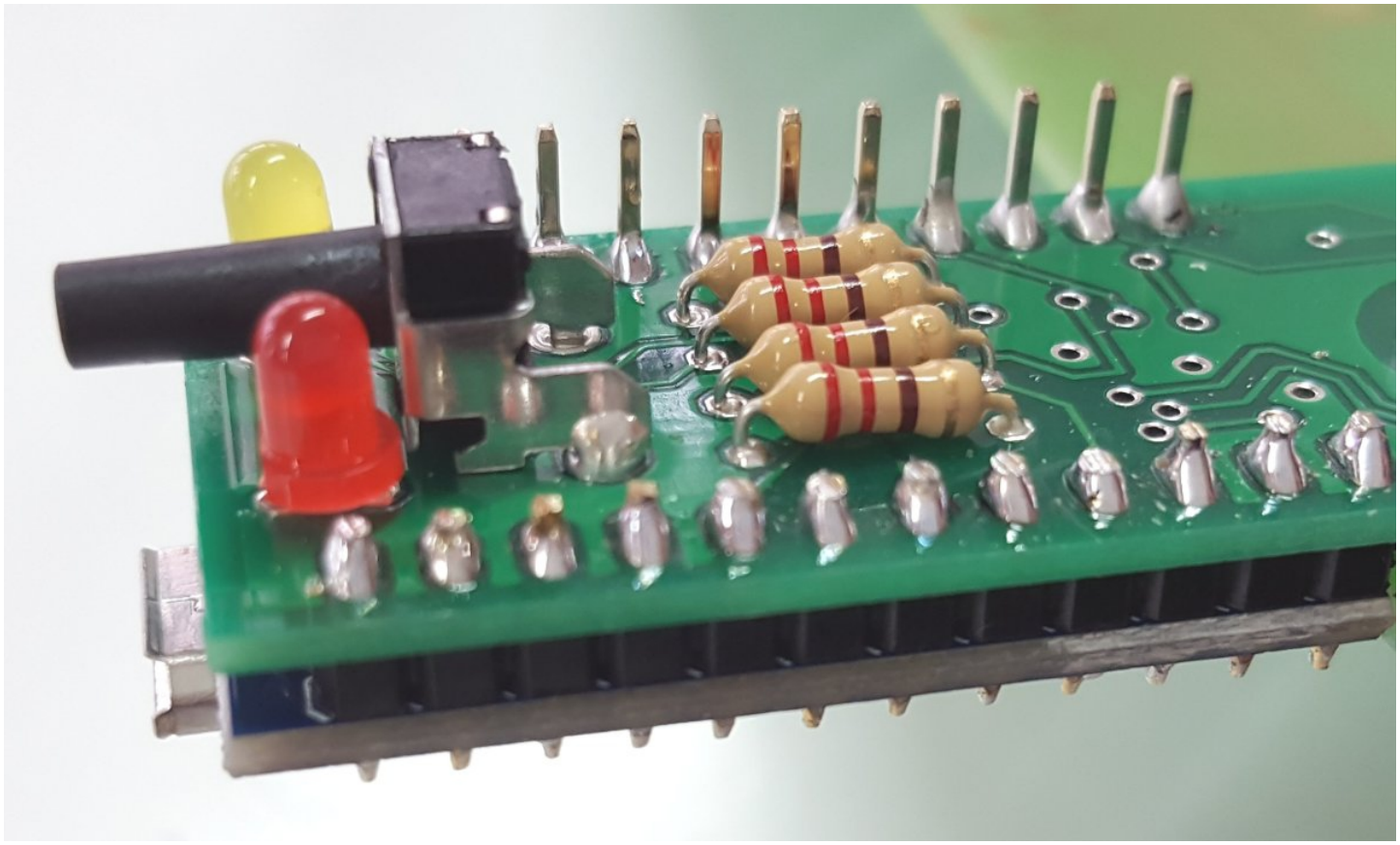
STEP 5: Add the CPU board to the main board, as shown in the following picture. Make sure that the exceeding leads from the resistors and the LEDS have been cut as shortest as possible.



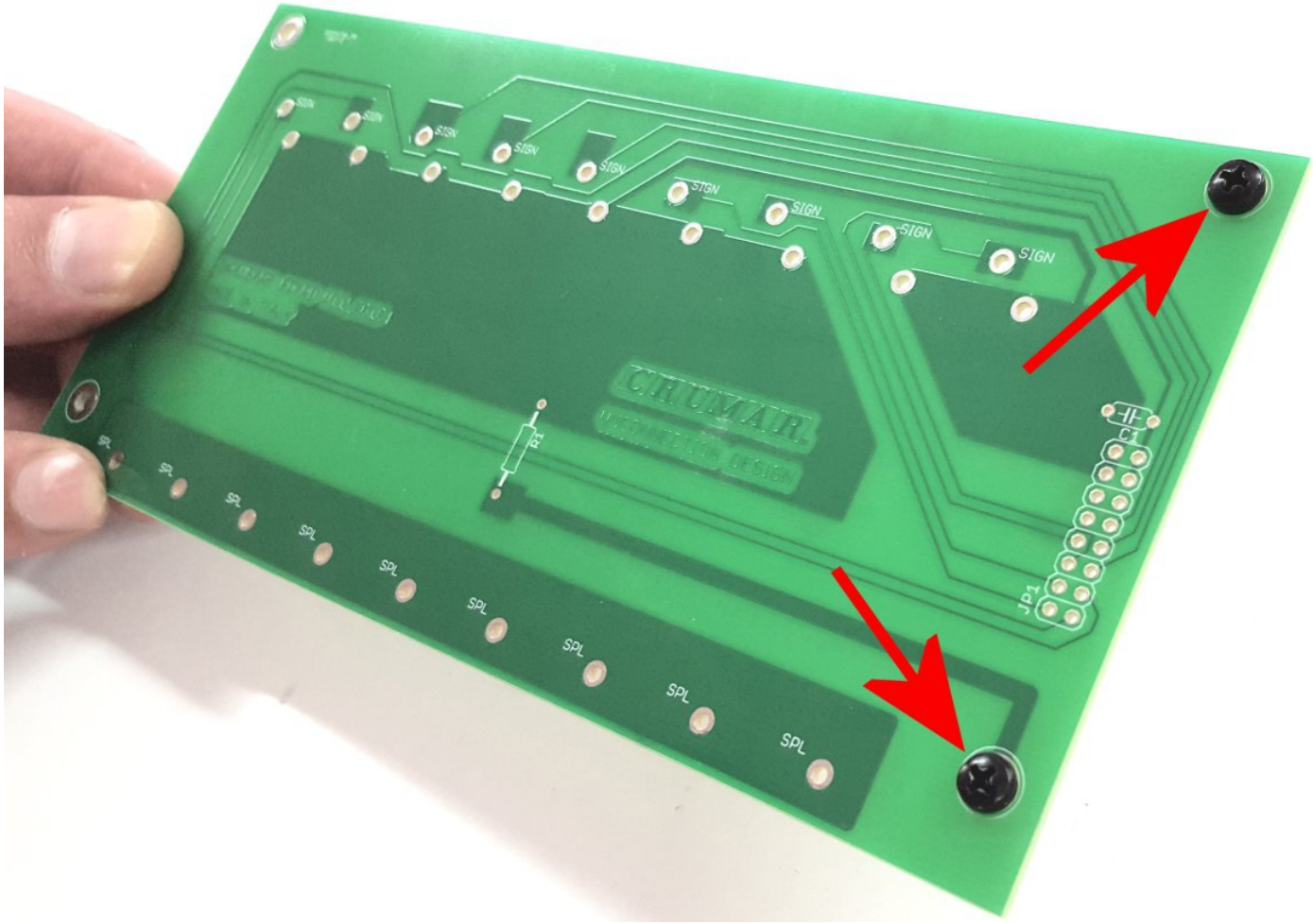
STEP 6: Once soldered the CPU board, the final assembly should look like the picture below.



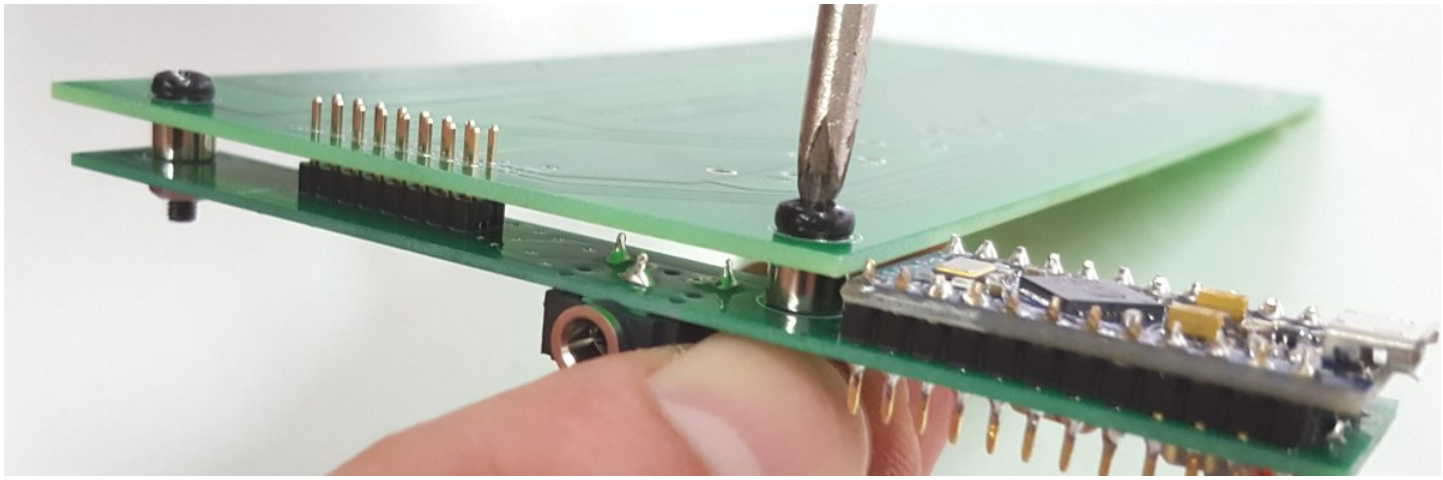
Now take your cutters and cut the exceeding terminal strip leads on the JACK side, as shown below. Optionally, you can also cut the exceeding leads on the other side.



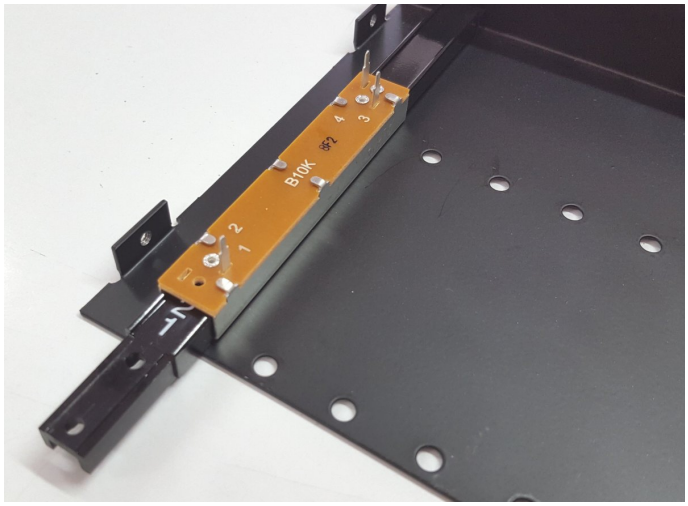
STEP 7: Couple the two boards. The two M3X16 black screws must be set to the right side of the drawbar board as shown here below.

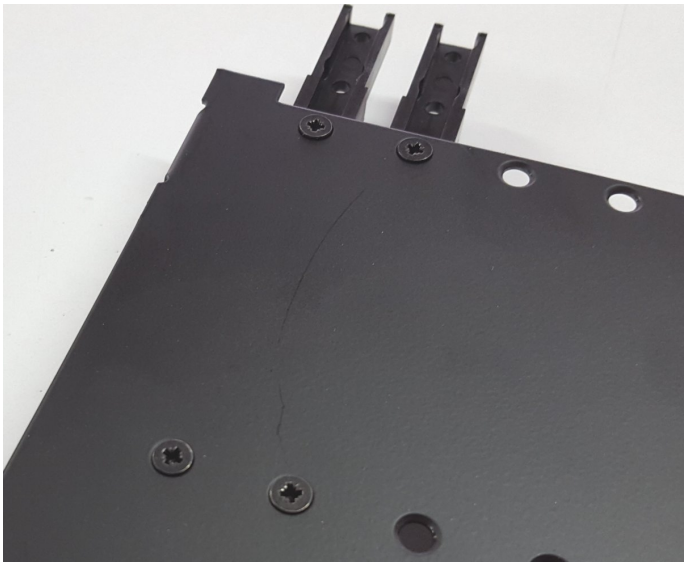


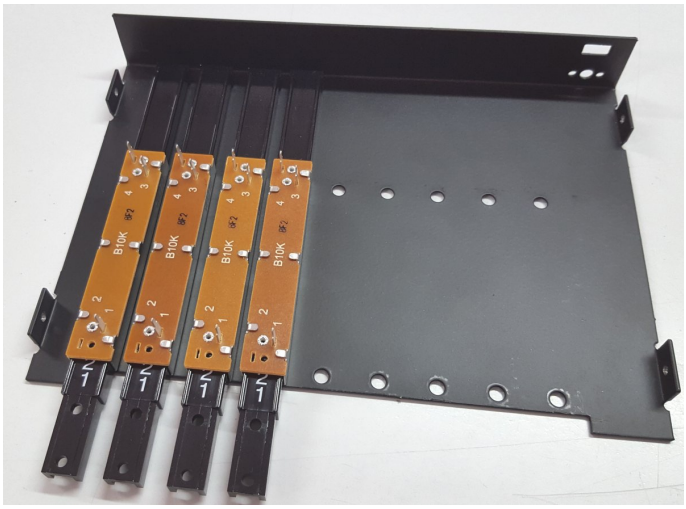
Then, insert the spacers on the other side, add the main board and fix the bolts using your fingers to hold the bolts and a screwdriver to tighten the screws.

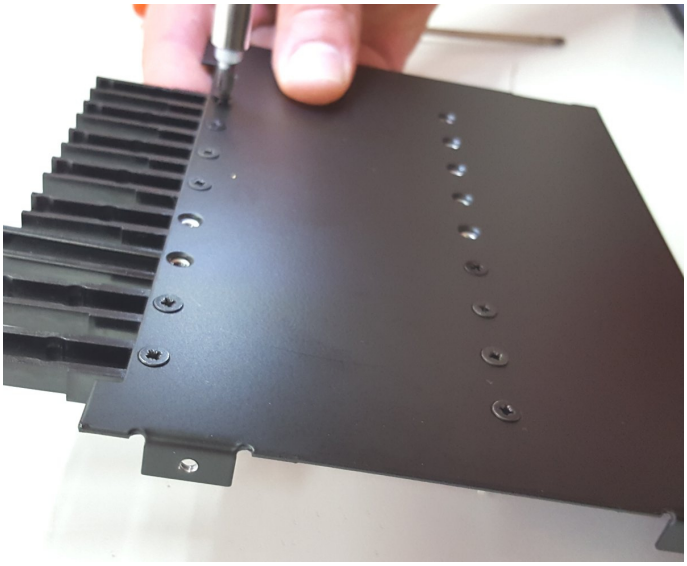


STEP 8: Use 18 out of the 22 M3X4 black screws to fix the drawbars to the metal bottom piece, as shown in the picture sequence below.

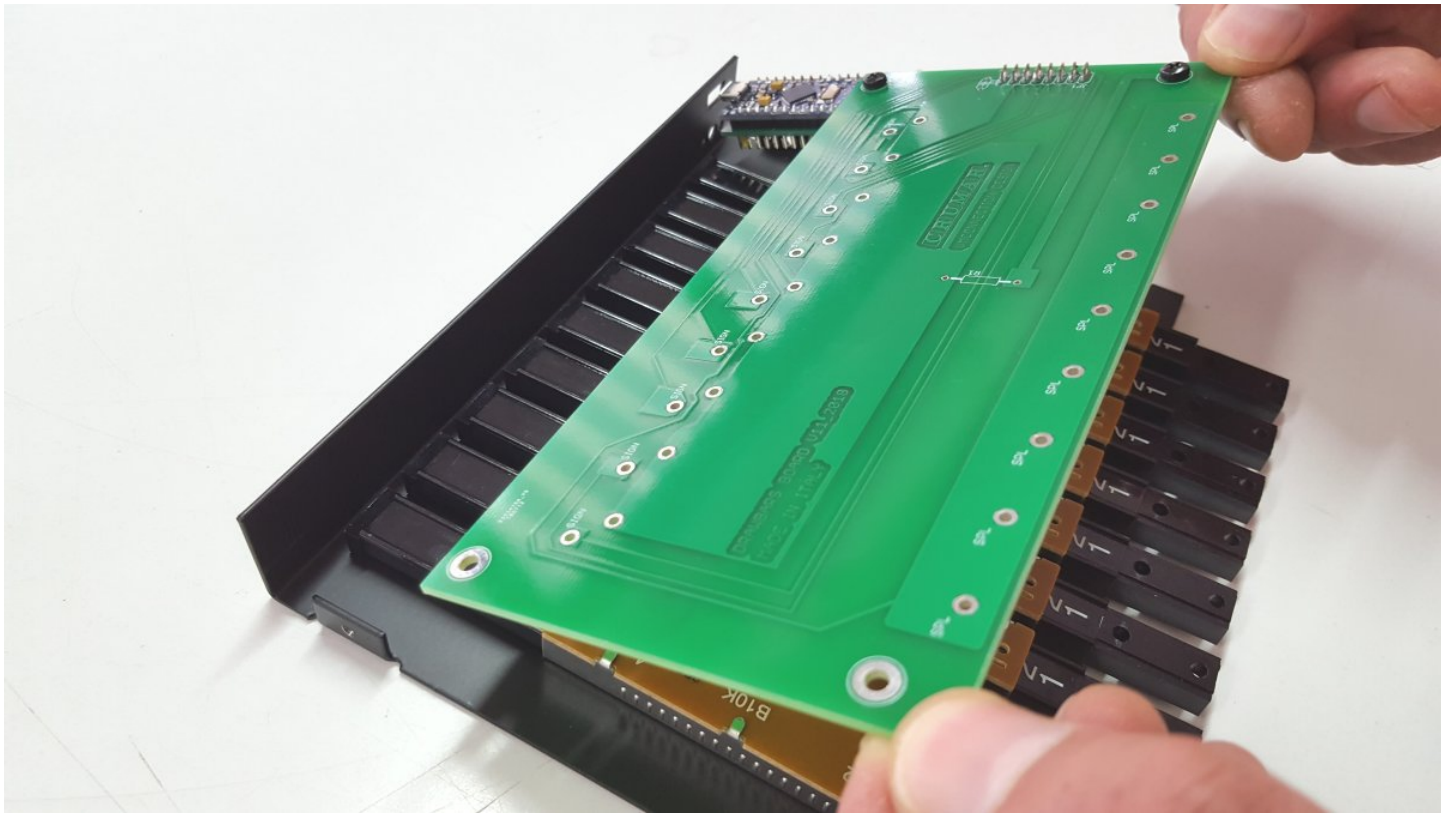








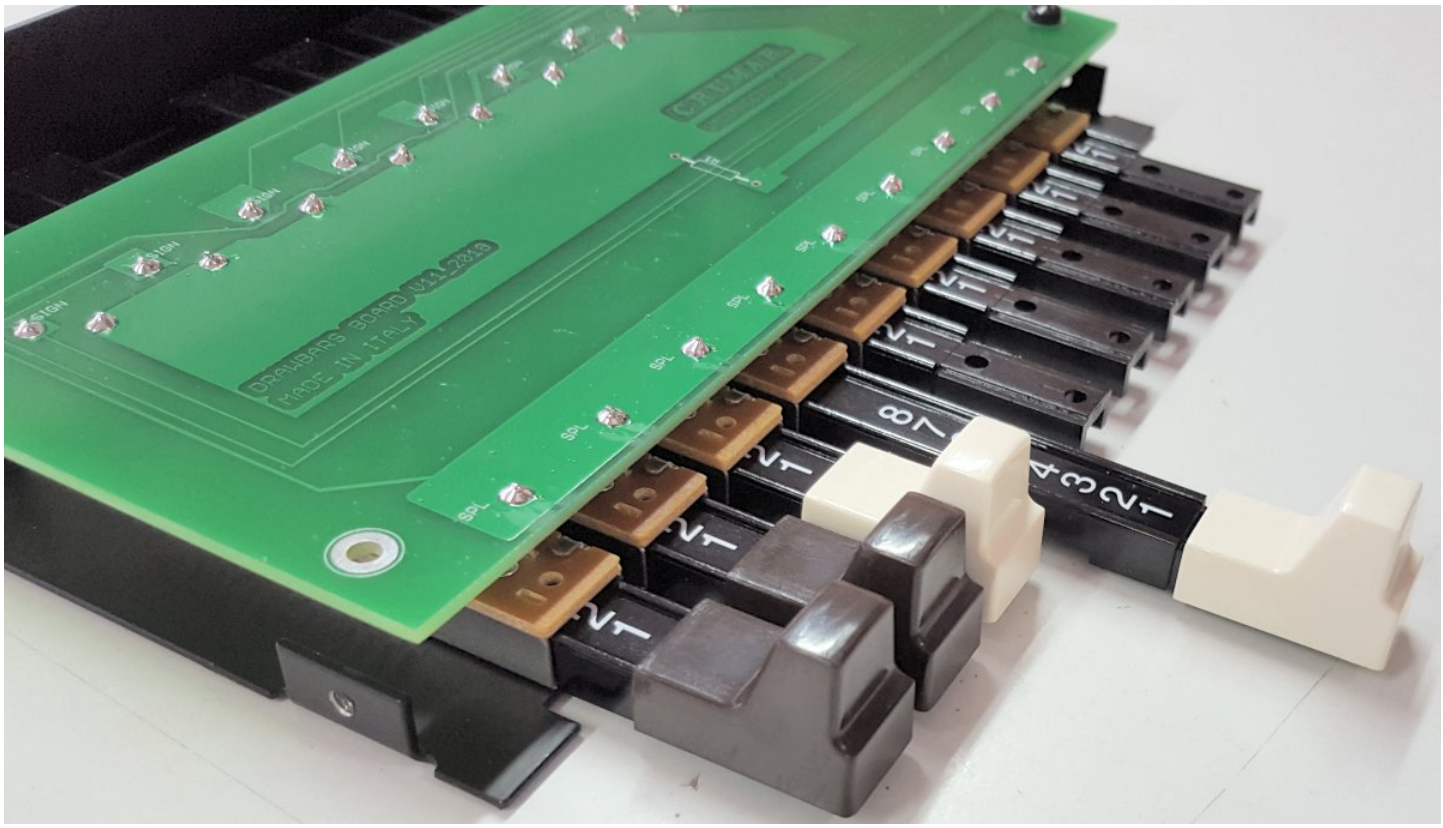
STEP 9: Fix the PCB assembly to the drawbars and solder the DIL terminals. Slide the boards slighly angled so the MicroUSB connector can easily pass through the hole in the metal panel.



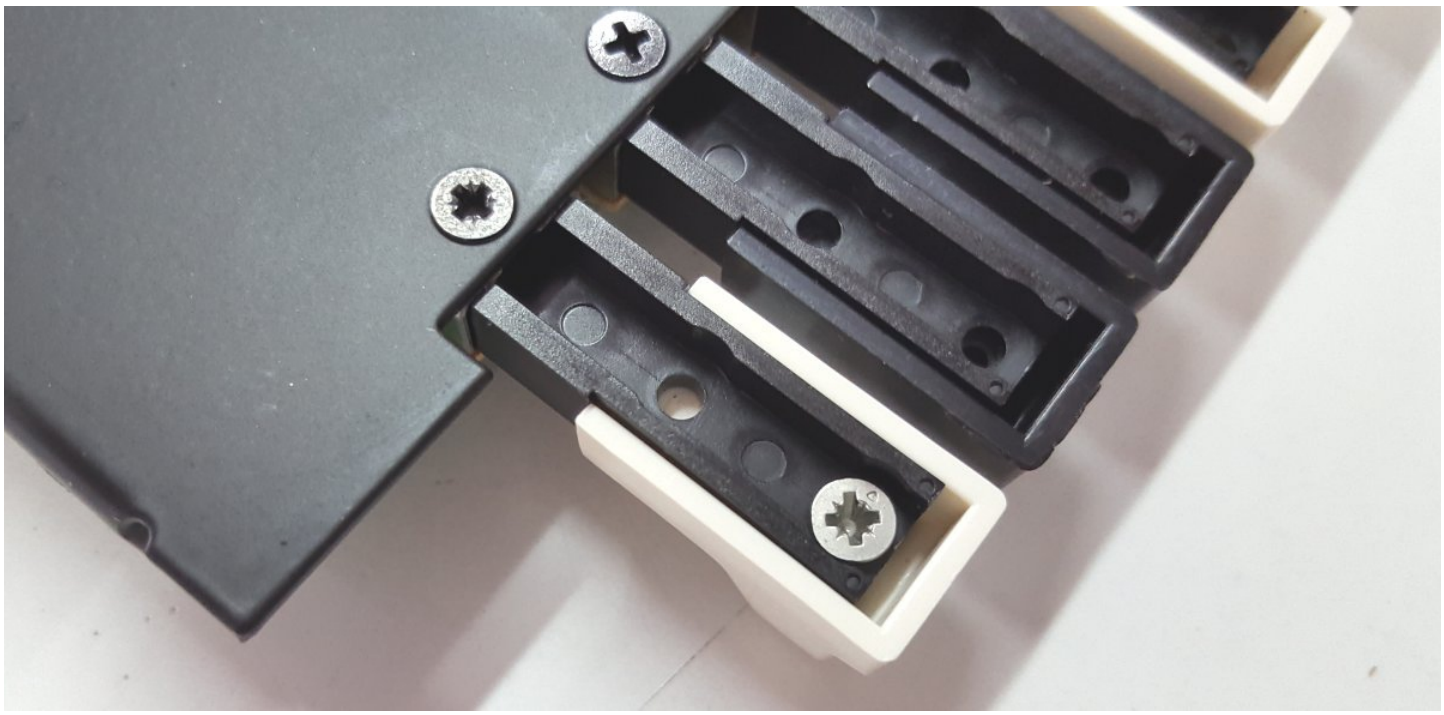
STEP 10: Solder all drawbar terminals and the DIL strip.

STEP 11: Add the plastic drawbar caps. First set them in position using the following color sequence:

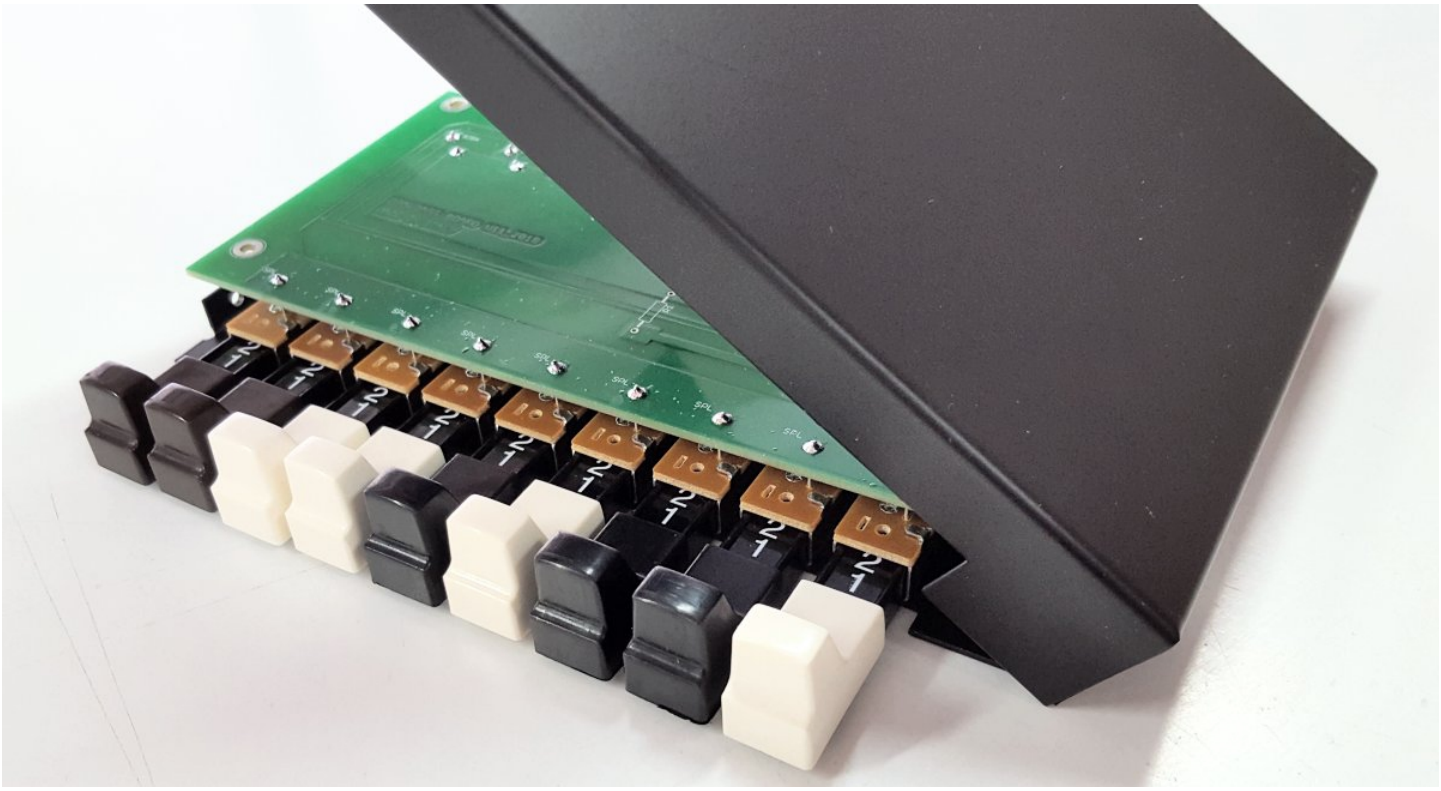
BROWN – BROWN – WHITE – WHITE – BLACK – WHITE – BLACK – BLACK- WHITE



Once done, turn the whole assembly upside down and fix them using the 9 M3X6 screws.



STEP 12: Put the metal top cover. In order to have the jack connector easily pass through the hole in the metal panel, start from an angle from the connector's side.



Use the remaining 4 M3X4 screws to close the cover.

PROGRAMMING THE CPU WITH ARDUINO

You need a computer with Arduino IDE installed. Get the IDE from the URL https://www.arduino.cc/en/Main/Software and install it on your computer following all the instructions given by the Arduino documentation.

If you haven't downloaded it yet, go to www.Crumar.it and download the Arduino sketch for the D9U from the Support section.

Below the complete sketch listing with comments.

```
////////////////////////////////////////////////////////////////////
// Crumar Drawbar Controller D9U
// by Guido Scognamiglio
// Runs on Atmel ATmega32U4 Arduino Leonardo (with MIDI USB Library)
// Reads 9 analog inputs from internal ADCs
// Sends MIDI CC numbers 12-20 or 21-29 according to selected mode
// Last update: July 2018
//


////////////////////////////////////////////////////////////////////
// This is where you can define your CC numbers for the Bank 0 or 1
int CCMap[2][9] =
{
  { 12, 13, 14, 15, 16, 17, 18, 19, 20 }, // Upper drawbars
  { 21, 22, 23, 24, 25, 26, 27, 28, 29 }  // Lower drawbars
};


////////////////////////////////////////////////////////////////////
// You should not modify anything else below this line
// unless you know what you're doing.
////////////////////////////////////////////////////////////////////

// Define I/O pins
#define LED_RED     15
#define LED_GREEN   16
#define BUTTON      5

// Define global modes
#define DEBOUNCE_TIME 150
#define DEADBAND    8

// Include libraries
#include <EEPROM.h>
#include <MIDIUSB.h>
#include <MIDI.h>

MIDI_CREATE_DEFAULT_INSTANCE();


// Init global variables
int mode = 1; // Should be either 0 or 1
int prev_val[9] = { -1, -1, -1, -1, -1, -1, -1, -1, -1 };
int debounce_timer = DEBOUNCE_TIME;


// ADC reference map
int ADCmap[9] = { A0, A1, A2, A3, A6, A7, A8, A9, A10 };
int ADCcnt = 0;
```

```
// Called then the pushbutton is depressed
void set_mode()
{
    digitalWrite(LED_RED, mode ? LOW : HIGH);
    digitalWrite(LED_GREEN, mode ? HIGH : LOW);
    EEPROM.write(0x01, mode);
}



// Called to generate the MIDI CC message
void SendMidiCC(int channel, int num, int value)
{
  midiEventPacket_t CC = {0x0B, 0xB0 | channel, num, value};
  MidiUSB.sendMIDI(CC);
  MidiUSB.flush();

  // Midi lib wants channels 1~16
  MIDI.sendControlChange(num, value, channel+1);
}



// Called to check whether a drawbar has been moved
void DoDrawbar(int d, int value)
{
  // Get difference from current and previous value
  int diff = abs(value - prev_val[d]);

  // Exit this function if the new value is not within the deadband
  if (diff <= DEADBAND) return;

  // Store new value
  prev_val[d] = value;

  // Get the 7 bit value
  int val7bit = value >> 3;

  // Send Midi
  SendMidiCC(mode > 0 ? 1 : 0, CCMap[mode][d], val7bit);
}



// The setup routine runs once when you press reset:
void setup()
{
  // Initialize serial MIDI
  MIDI.begin(MIDI_CHANNEL_OMNI);

  // Set up digital I/Os
  pinMode(BUTTON, INPUT_PULLUP); // Button
  pinMode(LED_RED, OUTPUT);      // Led 1
  pinMode(LED_GREEN, OUTPUT);    // Led 2

  // Recall mode from memory and set
  // Make sure mode is either 0 or 1
  mode = EEPROM.read(0x01) > 0 ? 1 : 0;
  set_mode();
}
```

```
// The loop routine runs over and over again forever:
void loop()
{
  // Read analog inputs (do the round robin)
  DoDrawbar(ADCcnt, analogRead(ADCmap[ADCcnt]));
  if (++ADCcnt > 8) ADCcnt = 0;

  // Read Button
  if (digitalRead(BUTTON) == LOW)
  {
    if (debounce_timer > 0) --debounce_timer;
  } else {
    debounce_timer = DEBOUNCE_TIME;
  }

  if (debounce_timer == 2)
  {
    mode = !mode; // Reverse
    set_mode(); // and Set!
  }
}
```

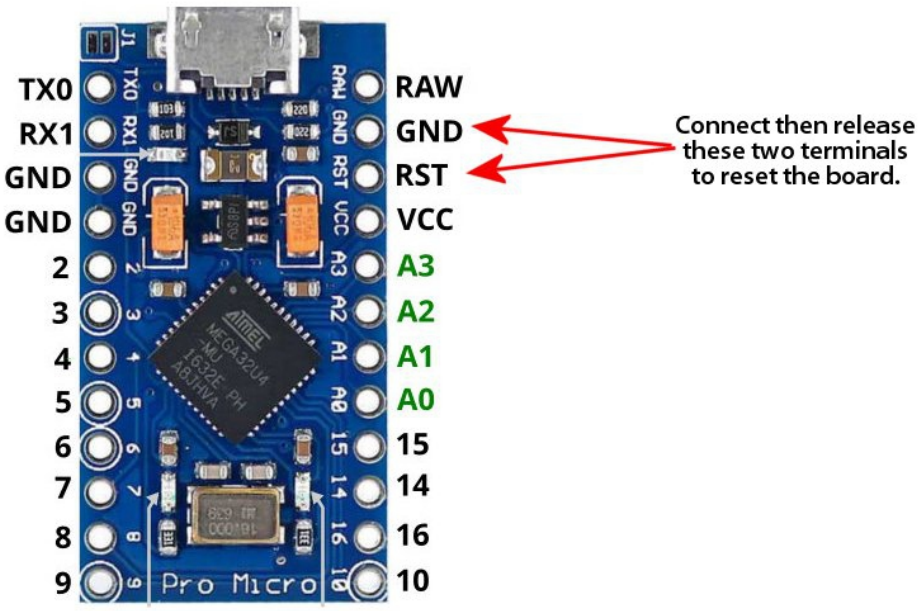
PLEASE NOTE: before compiling the sketch, make sure you have installed the required libraries (see next chapter).

Connect the D9U to your comptuer using the provided USB cable, start the IDE, load the sketch, then:

1. From the TOOLS menu, select BOARD -> Arduino Leonardo;
2. From the TOOLS menu, make sure the selected PORT points to Leonardo;
3. Click the icon with the arrow pointing right, this will compile the sketch and upload it to the board.

PLEASE NOTE: once the sketch is properly uploaded to the board, the USB port will not be seen by the IDE because it changes its function to USB-MIDI. To check that it is actually seen as a MIDI device, if you're using Windows (preferred), download and install the free application MIDI-OX and check that the ARDUINO MIDI device is listed among your MIDI ports, select it as an input port to MIDI-OX and check that it is correctly sending the expected CC messages.

In case you need to reprogram the board, one second before clicking the "LOAD" icon in the Arduino IDE, you have to "reset" the board by making a contact between the terminals labeled RST and GND using a small screwdriver.



THE ARDUINO SKETCH EXPLAINED

The Atmel Atmega32U4 processor used on the Arduino Leonardo (or Pro Micro) board has 9 built-in ADC inputs (Analog-to-digital converters), just what we need to read 9 linear 10K potentiometers, that in our case are the drawbars. All this sketch has to do is to read the drawbars and generate MIDI CC messages. Optionally, it can read a push-button and lit 2 LEDs that are used to swtich between two banks of pre-defined CC numbers, that we can use to control either the Upper or the Lower manual of a Clonewheel organ.

The MIDI stream is output both from the USB-MIDI port, using the 32u4 built-in USB controller, and from the UART port clocked at 31250 Baud as the MIDI specs require.

So this sketch uses the following libraries (that should be installed separately using the IDE library functions, in case they aren't pre-installed):

- MIDI Library by Fourty Seven Effects - https://github.com/FortySevenEffects/arduino_midi_library Used to generate MIDI messages to be sent to the UART PORT
- MIDIUSB by Gary Grewal - https://www.arduino.cc/en/Reference/MIDIUSB Used to generate MIDI messages to be sent via USB
- EEPROM (built-in) Used to store and recall the selected CC bank into the internal EEPROM

The loop() function cycles through the 9 ADCs by reading their current values and storing them into an array; the same function also "listens" whether the push button has been depressed (contact to ground, value LOW). The pin to which the button is connected has an internal pull-up resistor.

The DoDrawbar() function compares the value of a given ADC read by loop() with the previous value of the same ADC, if the difference is greater than the value set by DEADBAND, it is considered as a potentiometer value change and generates the MIDI event to be sent. The DEADBAND value is set to 8, because the ADCs have a 10 bit resolution (2^10 = 1024 values), but we need to scale it down to 7 bits (2^7 = 128 value), so we can discard 7 values each 8. In other words, if previous value was 1000 and current is 1004, nothing changes, but if the new value is 1009, then this is interpreted as a value change. This mainly prevents unwanted messages to be sent when drawbars aren't actually moved by the user.

The SendMidiCC() function generates the actual MIDI CC message, while set_mode() is called each time the button is depressed, switches the LED, sets the value of the "mode" variable and stores it into the internal EEPROM.

The CC numbers to be used are stored into the two-dimensional array CCMap[2][9], which, as the numbers in the brackets suggest, contains 2 banks of 9 values each.

USING THE OPTIONAL SERIAL MIDI OUTPUT

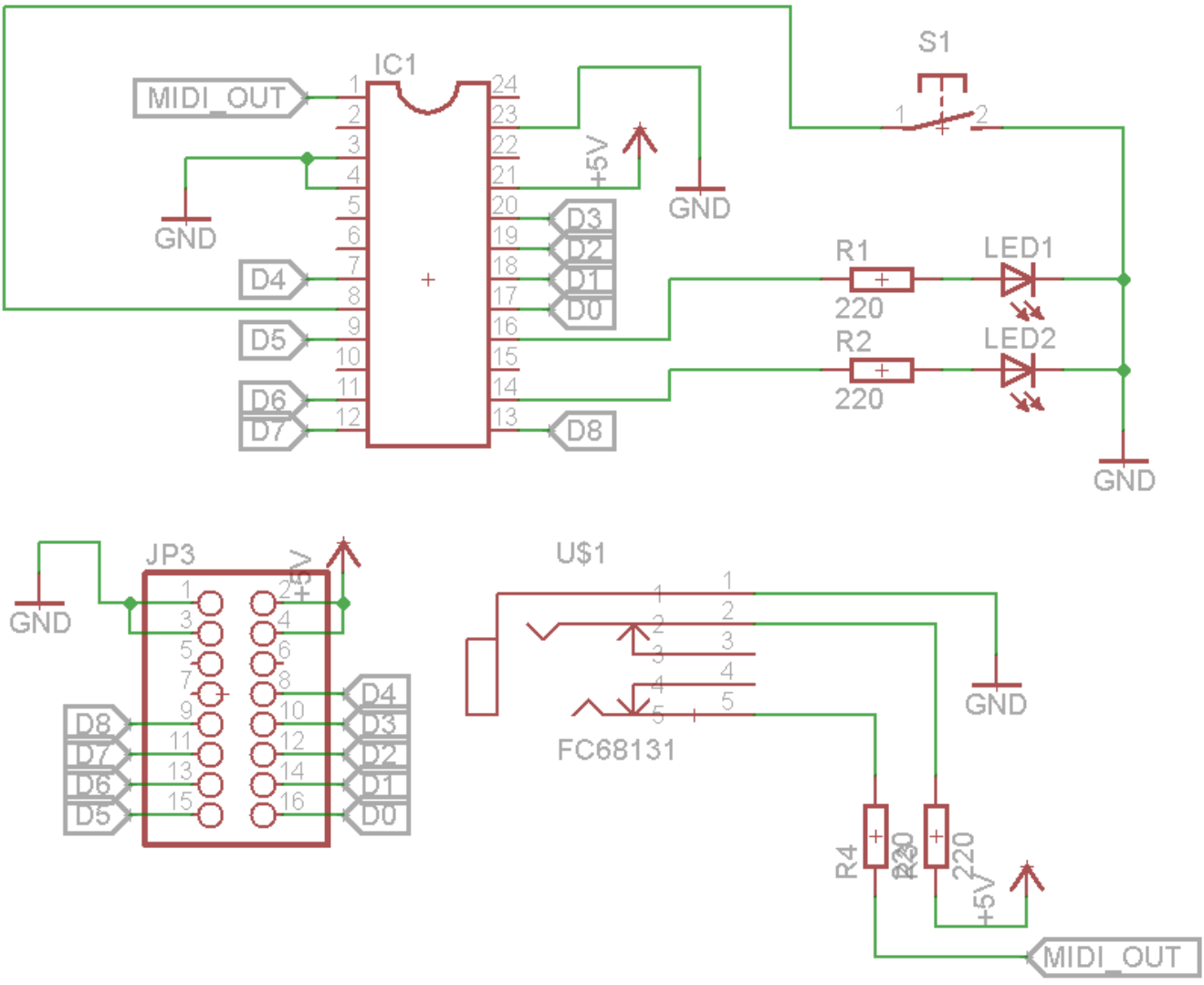
The D9U can also be used with traditional MIDI devices that use the common DIN5 connector, but needs a special "MiniJack to DIN5" adapter. This kind of MIDI connection has become a standard in the recent years, but there are two types. More informations can be found on the MIDI.org website at the following URL:

https://www.midi.org/articles-old/updated-how-to-make-your-own-3-5mm-mini-stereo-trs-to-midi-5-pin-din-cables

The D9U uses TYPE B.

Use a 5V cellphone USB Charger/PSU to power the D9U and use your adapter for the MIDI connection.

SCHEMATIC DIAGRAM



Crumar D9U is sold on www.MyRigShop.com by V.M.Connection
an enterprise based in Veneto, Italy.
For more informations please visit www.Crumar.it