picoMIDI manual v1.0

Contents: **Page**

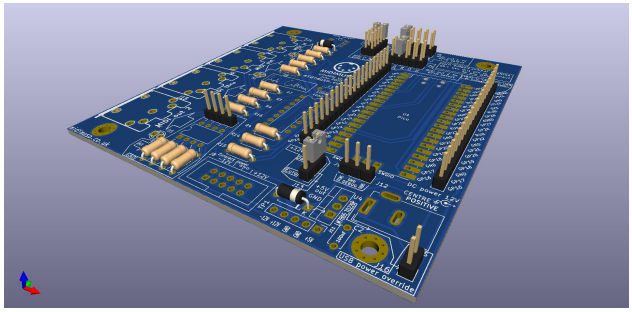
Disclaimer

This Raspberry Pi picoMIDI expansion board is provided as is without any guarantees or warranty. In association with the product, Midimuso makes no warranties of any kind, either express or implied, including but not limited to warranties of merchantability, fitness for a particular purpose, of title, or of noninfringement of third party rights.

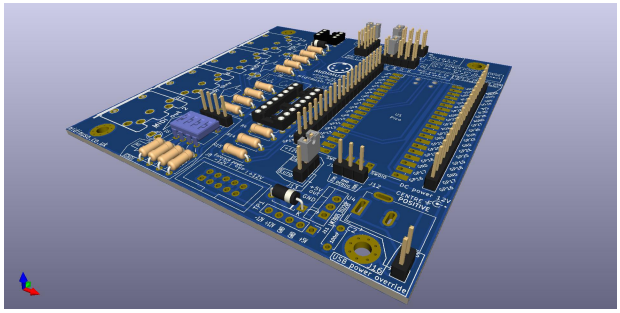
This product can be connected to the user's computer's USB port at the user’s own risk. The software examples given are for guidance only and are not guaranteed to work in the user's particular set-up environment.
The product is intended for connection to auxilliary electronic equipment,but Midimuso cannot be held responsible for any ensuing damage to said connected equipment.
All uses of the product by a user are at the user’s own risk.

Building the board

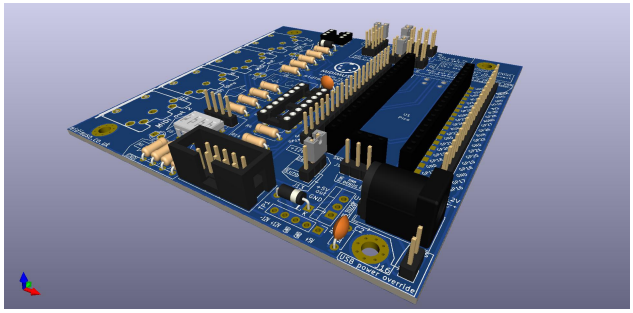
Stage 1



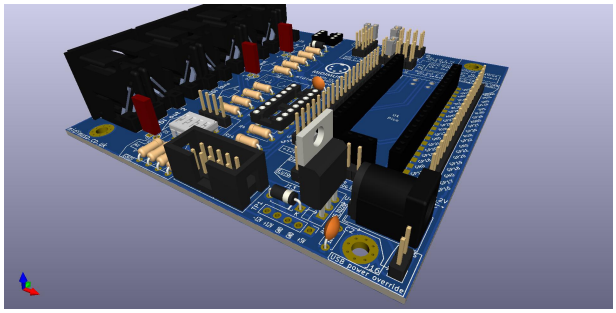
Stage 2



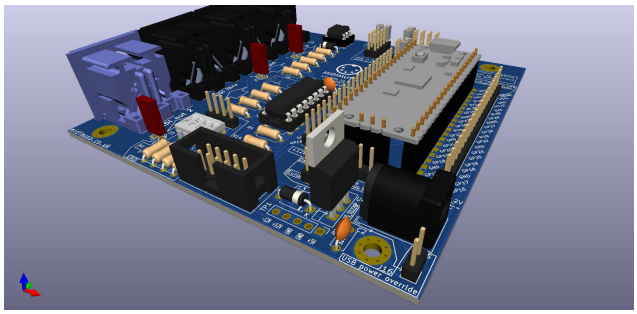
Stage 3



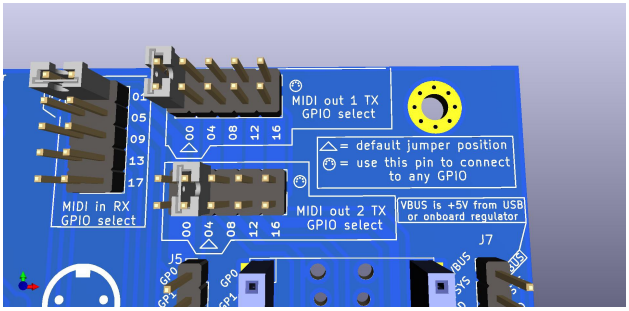
Stage 4



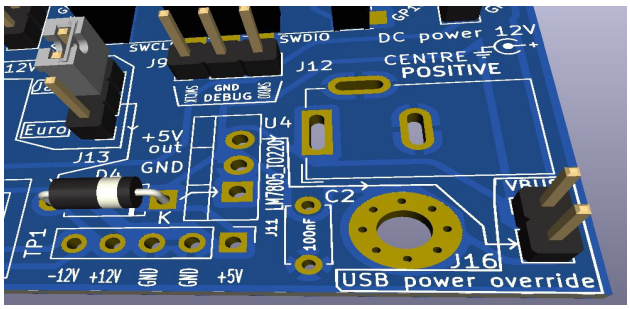
Stage 5



Stage 6



Stage 7



Start with the lowest components first.
stage 1: resistors, male headers, diodes
stage 2: dip switches, 16 and 6 pin DIL sockets
stage 3: female headers (2 x 20 and 1 x 3 pin), 12V power socket, eurorack header, 100 nF capacitors
stage 4: MIDI sockets, LEDs, 7805 regulator
stage 5: CD4050 and 4N 35 ICs, mount the Pi Pico
Stage 6, 7: jumpers on GPIO selectors and 12V source select.
Please Note! No jumper on 2 pin USB power override header (bottom right of PCB)

Initial Tests

Power the Pico board from a USB supply. This will power the whole board for now.
Connect one end of a MIDI cable to the MIDI in socket with the other end connected to a MIDI source e.g. synth or controller keyboard.
Play a few notes quickly – you should see the red LED next to the MIDI socket responding.

Now try using one of the example programs below to make the board generate MIDI depending on which language you intend to use.

Using the Jumper switches – how the UARTS are mapped on the PCB

RX means received by the Pico
TX means transmitted by the Pico
*default jumper position

PICO MIDI in RX select Jumper bank

TYPE	GPIO	UART	Arduino
RX	01	UART0 *	Serial1
RX	05	UART1	Serial2
RX	09	UART1	Serial2
RX	13	UART0	Serial1
RX	17	UART0	Serial1

PICO MIDI out 1 TX select Jumper bank

TYPE	GPIO	UART	Arduino
TX	00	UART0 *	Serial1
TX	04	UART1 **	Serial2
TX	08	UART1	Serial2
TX	12	UART0	Serial1
TX	16	UART0	Serial1

PICO MIDI out 2 TX select Jumper bank

Same mapping as **MIDI out 1 TX select** meaning that MIDI can be output to two sockets in parallel if needed.
default jumper posiiton for **MIDI out 2 is GPIO 04 which is UART1 TX

Getting started with the Arduino IDE
If you don't have it, download the Arduino IDE from
https://www.arduino.cc/en/software

Start the IDE, then **Open the Arduino application** and navigate to **File > Preferences**
In the **settings** tab, you'll see **Additional Board Manager URLs:**
Open this up with its button and, below any existing URLs, add:

https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

Then click **OK** to close the box and **OK** again to close the Preferences dialogue.

Next we need to install the Pico board
Go to:
Tools > Board > Board manager and scroll down until you see
Raspberry Pi Pico / RP2040
Click on that and you'll see an **install** button appear. Click on it.
This will automatically download all the required files, examples and compatible libraries (total size of download is > 300MB)

Wait until it's done and press the **close** button.

Next, tell the IDE that you are using a Pico by selecting:
Tools > Board > Raspberry Pi / RP2040 Boards > Raspberry Pi Pico

There will be a pause as it loads.

Now test it by loading the blink example:
Whilst pressing the **boot select** button on the Pico, plug it in to the PC's **USB**.
The computer will see it as a mass storage device.

Upload the blink sketch – this will take longer than usual.
(if the IDE complains “no upload port provided” select **Tools > Port > UF2 Board)**

The IDE will know that this is the **first** upload so it will try to also upload the equivalent of the Arduino bootloader to the Pico for future easy uploads.
In the IDE, the Pico now appears as a **COM** port which can be selected with **Tools > Port**
You'll usually see it as **COM3** (Raspberry Pi Pico) but it may use any other port number depending on what's already using ports - the **COM** number doesn't matter so long as you select the **(Raspberry Pi Pico)** option.
From now on, you should be able to upload new sketches without unplugging the Pico and you won't have to press the boot select button.

Windows 7 / General Driver Problems
If you're using **Windows 7** or the Arduino IDE doesn't see the **COM** port after your first upload, the installed **Pico** drivers may have failed, refer to Appendix A for help.
Then select **Tools > Port to select the COM** port**.**
If this still doesn't work, there is more detailed information here:

https://arduino-pico.readthedocs.io/en/latest/install.html

Two Simple MIDI sketches for Arduino

Jumpers:

Make sure:

The MIDI in RX GPIO select jumper is set to its default position (GPIO 01) (noted by a triangle)

MIDI out TX 1 GPIO select jumper is set to its default position (GPIO 00) (noted by a triangle)

UARTS

Pico	**Arduino**	
UART0	Serial1	(has TX and RX)
UART1	Serial2	(has TX and RX)

Serial (no number) is the USB serial port

See: https://arduino-pico.readthedocs.io/en/latest/serial.html for more information

MIDI note player

This sketch uses the picoMIDI board's MIDI out 1 to send MIDI note data.

If the MIDI out 1 jack is connected to a MIDI synth, it will play ascending notes.

```
// ++++++++++++++ ascending notes program starts here ++++++++++++
int delayTime = 64; // time between note events

void setup() {
  // Set MIDI baud rate:
  Serial1.begin(31250);
  // (optional) initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
 }

void loop() {
  // play ascending notes
  for (int note = 12; note < 96; note +=2) {
    //Note on channel 1 (0x90), note value (note), full velocity (127):
    noteOn(0x90, note, 127);
    digitalWrite(LED_BUILTIN, HIGH);   // (optional) turn the LED on (HIGH is the voltage level)
    delay(delayTime);
    //Note on channel 1 (0x90), same note value (note), silent velocity (0):
    noteOn(0x90, note, 0);
    digitalWrite(LED_BUILTIN, LOW);    // (optional) turn the LED off by making the voltage LOW
    delay(delayTime);
  }
}

// function to play a MIDI note.
void noteOn(int MIDIstatus, int pitch, int velocity) {
  Serial1.write(MIDIstatus);
  Serial1.write(pitch);
  Serial1.write(velocity);
}
// ++++++++++++++++++ program ends here ++++++++++++++++
```

MIDI Echo

This Arduino sketch uses the picoMIDI board's MIDI in to receive MIDI data and sends it out on MIDI out 1.
Both TX and RX are on Pico's UART0 which is Arduino's Serial1

Note that Arduino's serial ports are mapped to

Pico	**Arduino**
UART0	Serial1 (has TX and RX)
UART1	Serial2 (has TX and RX)

Serial (no number) is the USB serial port
See:
https://arduino-pico.readthedocs.io/en/latest/serial.html for more information

```
// ++++++++++++++ echo notes begins here +++++++++++++++
int dataByte;

void setup() {
  // Set MIDI baud rate:
  Serial1.begin(31250);  // start serial port at 9600 bps and wait for port to open:
 }

void loop() {
    if (Serial1.available() > 0) {
    // get incoming byte:
    dataByte = Serial1.read();
    Serial1.write(dataByte);
  }
}
// ++++++++++++++ program ends here +++++++++++++++
```

Getting started with MicroPython on Thonny

Jumpers:
Make sure:
The MIDI in RX GPIO select jumper is set to its default position (GPIO 01) (noted by a triangle)
MIDI out TX 1 GPIO select jumper is set to its default position (GPIO 00) (noted by a triangle)

Ascending Notes

```
# ++++++++++++++ program starts here ++++++++++++
from machine import Pin,UART
import time
uart = UART(0, baudrate=31250, tx=Pin(0), rx=Pin(1))
uart.init(bits=8, parity=None, stop=1)
led = Pin("LED", Pin.OUT)
note = 24

while True:
    midimessage = bytearray([0x90, note, 64])
    uart.write(midimessage)
    led.toggle()
    midimessage = bytearray([0x90, note, 0])
    uart.write(midimessage)
    time.sleep(0.125)
    note+=2
    if note > 84:
        note=24
# ++++++++++++++ program ends here ++++++++++++
```

MIDI echo

```
# ++++++++++++++ program starts here ++++++++++++
from machine import Pin,UART
import time
uart1 = UART(0, baudrate=31250, tx=Pin(0), rx=Pin(1))
uart2 = UART(1, baudrate=31250, tx=Pin(4), rx=Pin(5))
uart1.init(bits=8, parity=None, stop=1)
led = Pin("LED", Pin.OUT)


while True:
    if uart1.any() > 0:
        data = uart1.read(1)
        uart1.write(data)
        uart2.write(data)
        led.toggle()
# ++++++++++++++ program ends here ++++++++++++
```

Getting started with C++ in Visual Studio Code

Jumpers:

Make sure:

The MIDI in RX GPIO select jumper is set to its default position (GPIO 01) (noted by a triangle)

MIDI out TX 1 GPIO select jumper is set to its default position (GPIO 00) (noted by a triangle)

The easiest way to get started is to modify hello_serial.c which is included in Raspberry's examples folder **pico-examples\hello_world\serial**
it can also be found on github here:
https://github.com/raspberrypi/pico-examples/blob/master/hello_world/serial/hello_serial.c

// Program to send ascending notes to MIDI out

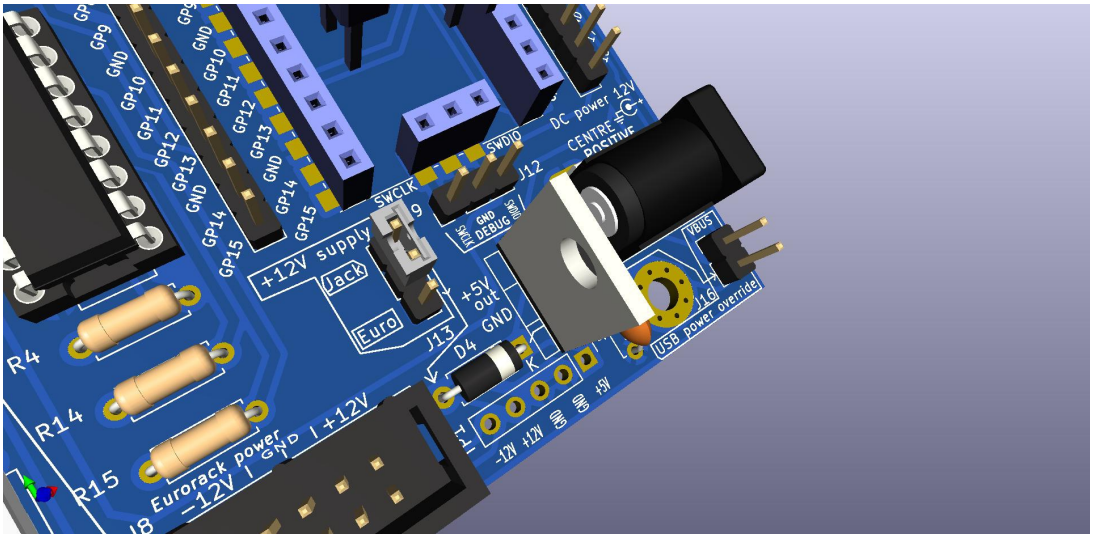
```
#include <stdio.h>
#include "pico/stdlib.h"

int main() {
    char midi_status = 0x90;    // note on
    char midi_pitch = 20;       // low note
    char midi_vel = 127;        // velocity full on
    stdio_init_all();
    uart_init(uart0, 31250);    // MIDI baud rate is 31,250
// Set the GPIO pins using UART 0 for MIDI
    gpio_set_function(0, GPIO_FUNC_UART); // TX
    gpio_set_function(1, GPIO_FUNC_UART); // RX
    while (true) {
        uart_putc(uart0, midi_status);
        uart_putc(uart0, midi_pitch);
        uart_putc(uart0, midi_vel);
        sleep_ms(125);
        midi_pitch++;
        if(midi_pitch > 80) midi_pitch = 20;
    }
    return 0;
}
```

```
// Program to echo bytes from MIDI in to MIDI out
#include <stdio.h>
#include "pico/stdlib.h"

int main() {
    char midi_byte;
    stdio_init_all();
    uart_init(uart0, 31250);    // MIDI baud rate is 31,250
// Set the GPIO pins using UART 0 for MIDI
    gpio_set_function(0, GPIO_FUNC_UART); // TX
    gpio_set_function(1, GPIO_FUNC_UART); // RX
    while (true) {
        // MIDI echo (blocking)
        midi_byte = uart_getc(uart0);
        uart_putc(uart0, midi_byte);
    }
    return 0;
}
```

Alternative power sources

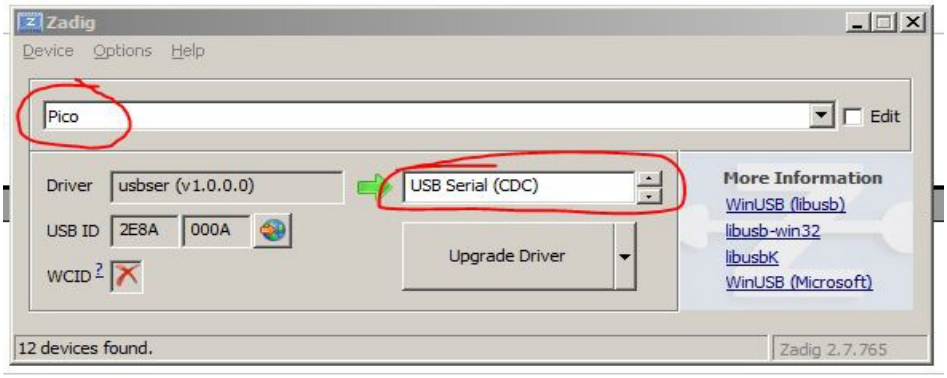


The board can be powered by 12 volts either from the **centre-positive** DC barrel jack or from the Eurorack connector. Use a jumper on the 3 pin header (see above) to select which.

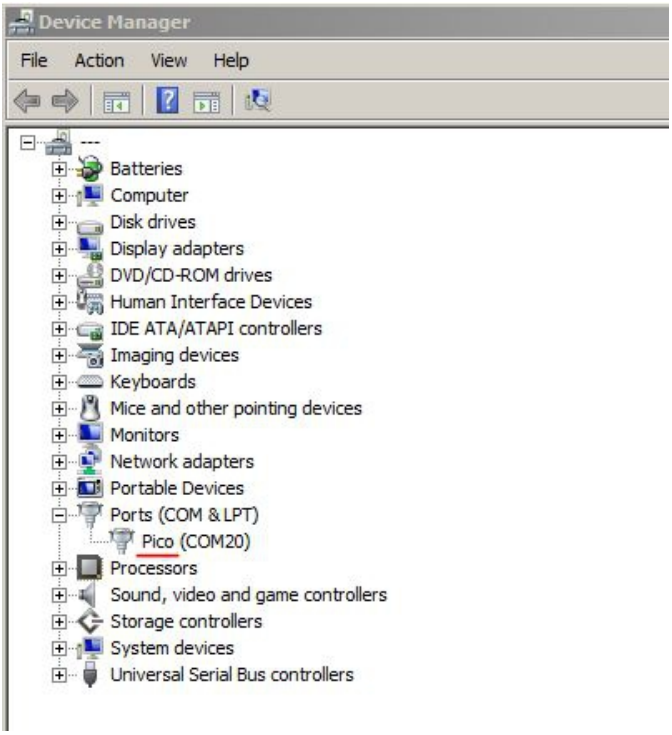
CAUTION: if the board is powered from a 12 volt supply **AND** a jumper is placed on the 2 pin USB power override header (see above) **AND** the Pico is plugged into a USB supply or computer, the 5 volt board-generated and 5 volt USB voltages will fight which may harm a host computer.
If you absolutely need to have the board powered whilst still connected to USB, we would strongly suggest you use an inexpensive, powered USB hub to isolate your computer from any possible, expensive harm.

In short, the 2 pin USB power override header (bottom right of PCB) should only be jumpered when the USB is disconnected from the Pico.

Appendix A
Windows 7 / General Driver Problems When Using Pico as a USB COM port.
If the Device manager doesn't see the Pico as a COM port, the installed Pico drivers may have failed.
We used Zadig https://zadig.akeo.ie/ to install the Pico as a USB serial (CDC) device.



Download and run **Zadig**
Select **Pico** from the device list
Select **USB serial (CDC)** from the driver list
Press the **Install** (Upgrade) driver button
This may take some time. If you watch device manager, you will see the Pico possibly moving from one device branch / type to another as its driver is installed / changed
The **Pico** then appears as a **COM** port in Device Manager.



Appendix B

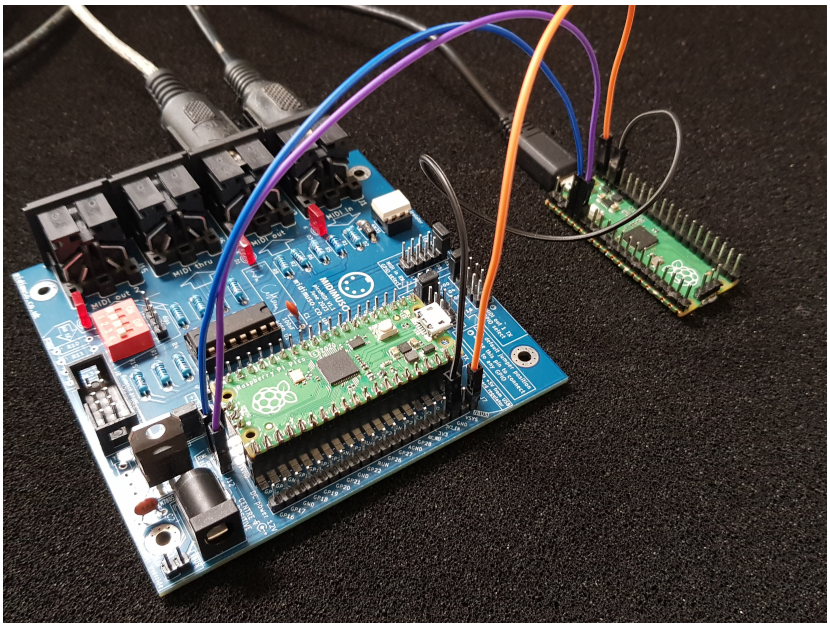
Picoprobe wiring

This the same as in the “Getting started with Raspberry Pi Pico” manual by the Raspberry Pi foundation EXCEPT the picoprobe powers the picoMidi board with VBUS not VSYS. VSYS is the USB-supplied voltage which allows the CD4050 buffer to have the full 5 volts. The wiring difference is:

	Picocprobe pin	**picoMIDI pin**
from	39	39
to	40	40

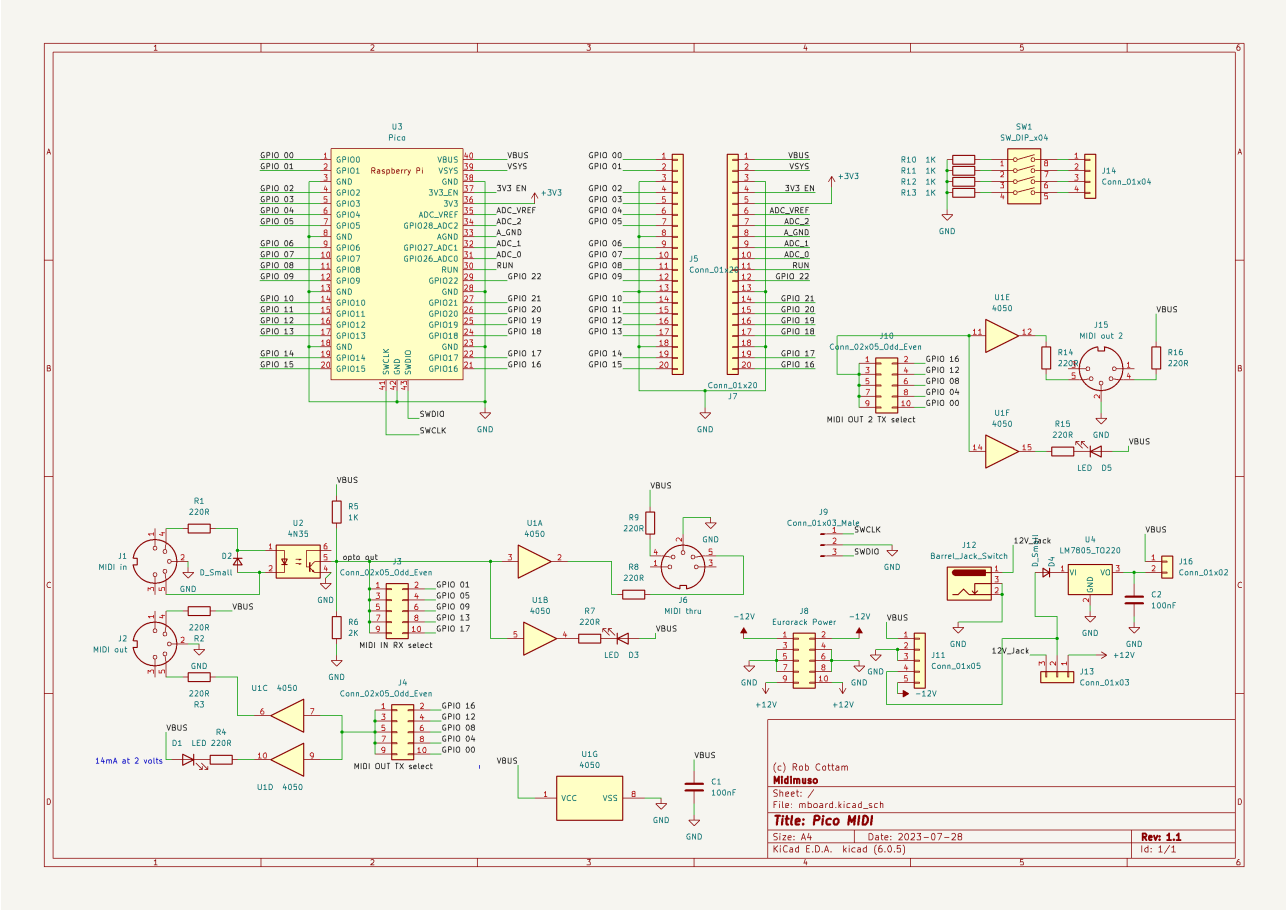
The other connections are as in the manual and they are:

Picocprobe pin	**picoMIDI pin**
GND	GND
4(GP2)	SWCLK
5(GP3)	SWDIO



In the manual, the target board's UART0 is also wired up for serial comms. It can be very useful for debugging. If you want to do this, you'll only have one MIDI out on UART1. You'll need to set the jumpers so that MIDI in and MIDI out 1 are attached to UART1. MIDI in could use GP 05 or 09.
MIDI out 1 could use 04 or 08
Your code needs to initialise the pins you're using so they are committed as UART pins.

Schematic



Continuity