Analog Front-End Design 1

> *If you have suggestions for improvements or would like to share your own designs for a Scoppy analog front end then please head to the Scoppy forum.*

To keep things simple for this first front end design, we'll support a single input voltage range of approximately -6V to +6V.

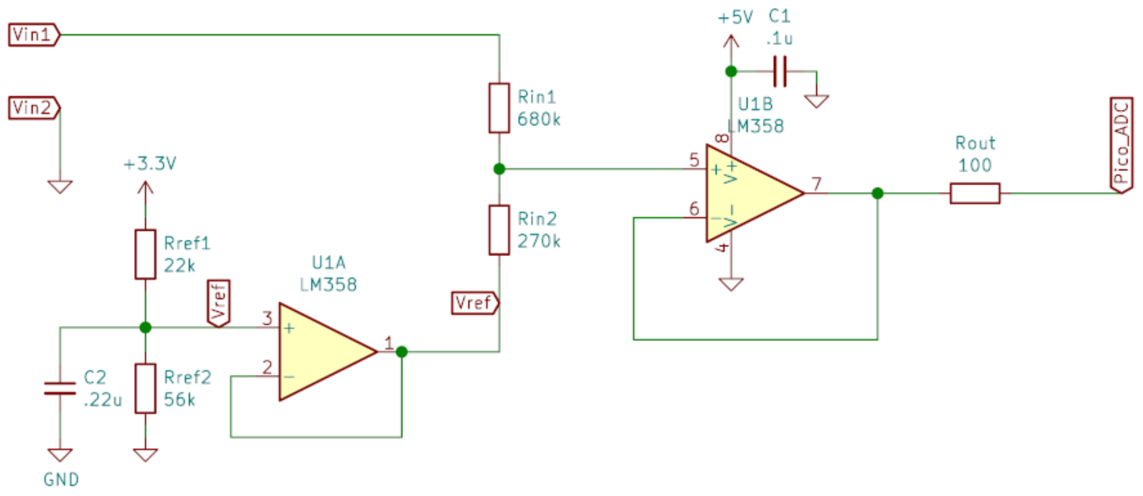
By 'input voltage range' I mean that applying the minimum voltage in the range (eg. -6V) to the input of the frontend will result in a voltage of 0V at the ADC pin of the Pico. Applying the maximum voltage in the range (eg. 6V) will result in a voltage of 3.3V at the ADC pin of the Pico.

Given that the voltage range of the Pico ADC pins is 0V to 3.3V, our frontend will need to attenuate the input signal to a range of 3.3V and shift the voltage level up so that the lowest input voltage (-6V) will become 0V.

An easy way to do this is using a voltage divider in front of the non-inverting input. One end will be our signal to measure (Vin1) and the other will be a reference voltage (Vref). The reference voltage will have the effect of level shifting the output voltage to the level expected by the Pico.

The op-amp used for this design is the LM358. I chose this because it is super cheap, readily available, has two op-amps in the one package and will work with a single supply of 5V.

Schematic



Discussion

First we need to select values for the input resistors Rin1 and Rin2 so that the correct gain is applied to the input signal.

Calculate the required gain of the voltage divider by dividing 3.3 (the Pico's voltage range) by the input voltage range (12V for this design). We end up with a value of 0.275. Select values for Rin1 and Rin2 such that Rin2/(Rin1 + Rin2) is similar to the required gain. Values in the E12 series that would work are 680k and 270k. The gain using these values will be 0.28. Note. High resistor values are used so that the input impedance of our frontend is high.

Now calculate the required value of Vref. The formula for calculating this is: Vref = Vinmin / (1 - 1/gain) Where Vinmin is the minimum value of the input voltage range.

So for our design this is -6V / (1 - (1 / 0.284)) = 2.38V Note that we use 0.284 rather than 0.275 as the gain value as this will be closer to the actual gain of the voltage divider. With a gain of 0.284 the input voltage range will be -5.8V to +5.8V. If you want to be more precise you can of course measure the actual values of the resistors.

Using the above values for the voltage divider and Vref, an input voltage of -5.8V will become 0V at the non-inverting input of the op-amp and an input voltage of 5.8V will be 3.3V. This voltage range is exactly the range of the Pico ADC and so the op-amp can be wired up in a unity gain configuration.

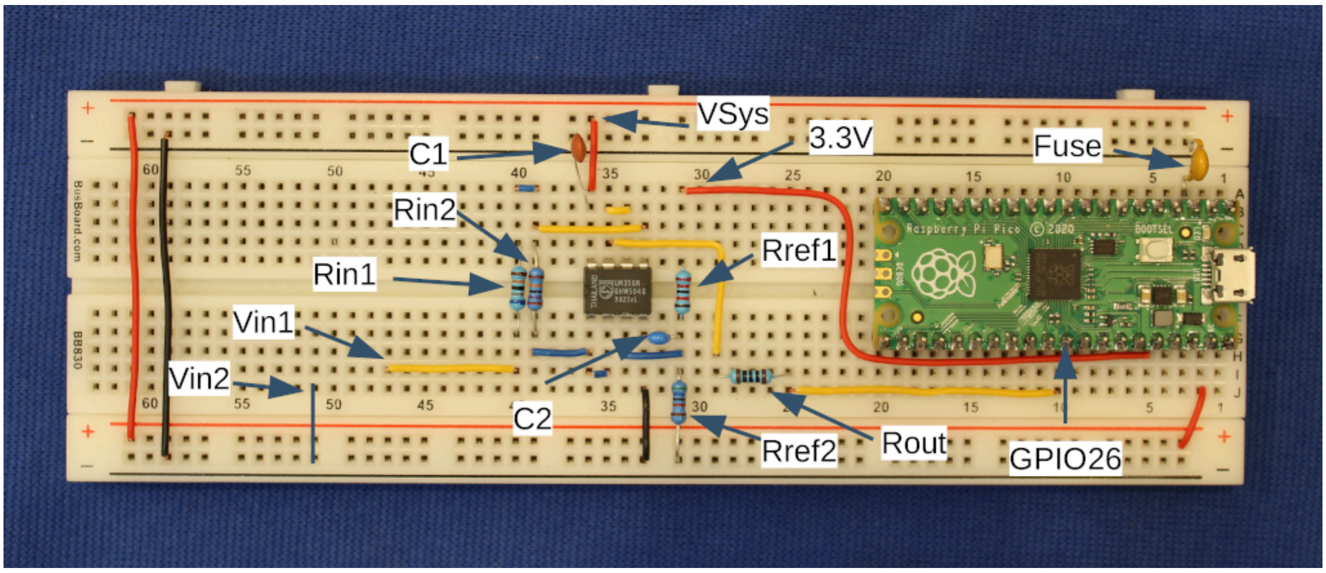
The above procedure for determining the resistor values should work for any voltage range with a magnitude of more than 3.3V. For anything smaller than that we'll need to modify the design so that the incoming signal is amplified.

A current limiting resistor is added to the output of the op-amp to protect the Pico in case the op-amp output goes above 3.3V (in which case current will leak through the ESD diodes of the Pico). This shouldn't happen if we keep our input voltage below 5.8V.

The 2nd of the LM358 op-amps will be used to help generate Vref. A resistor divider from 3.3V to GND is used to create a voltage as close as possible to our desired Vref of 2.38V. Note: If Vref isn't exactly 2.38V it doesn't really matter, it just means that the input voltage range won't be centred around 0. For this design we'll use 22k for Rvref1 and 56k for Rvref2. A capacitor from the Vref input of the op-amp to GND will help reduce noise on the output.

Construction

Assemble the circuit as per the schematic and breadboard image below. The fuse shown in the image is optional - it's just there for klutzes like me who tend to get the wiring wrong and short the supplies.

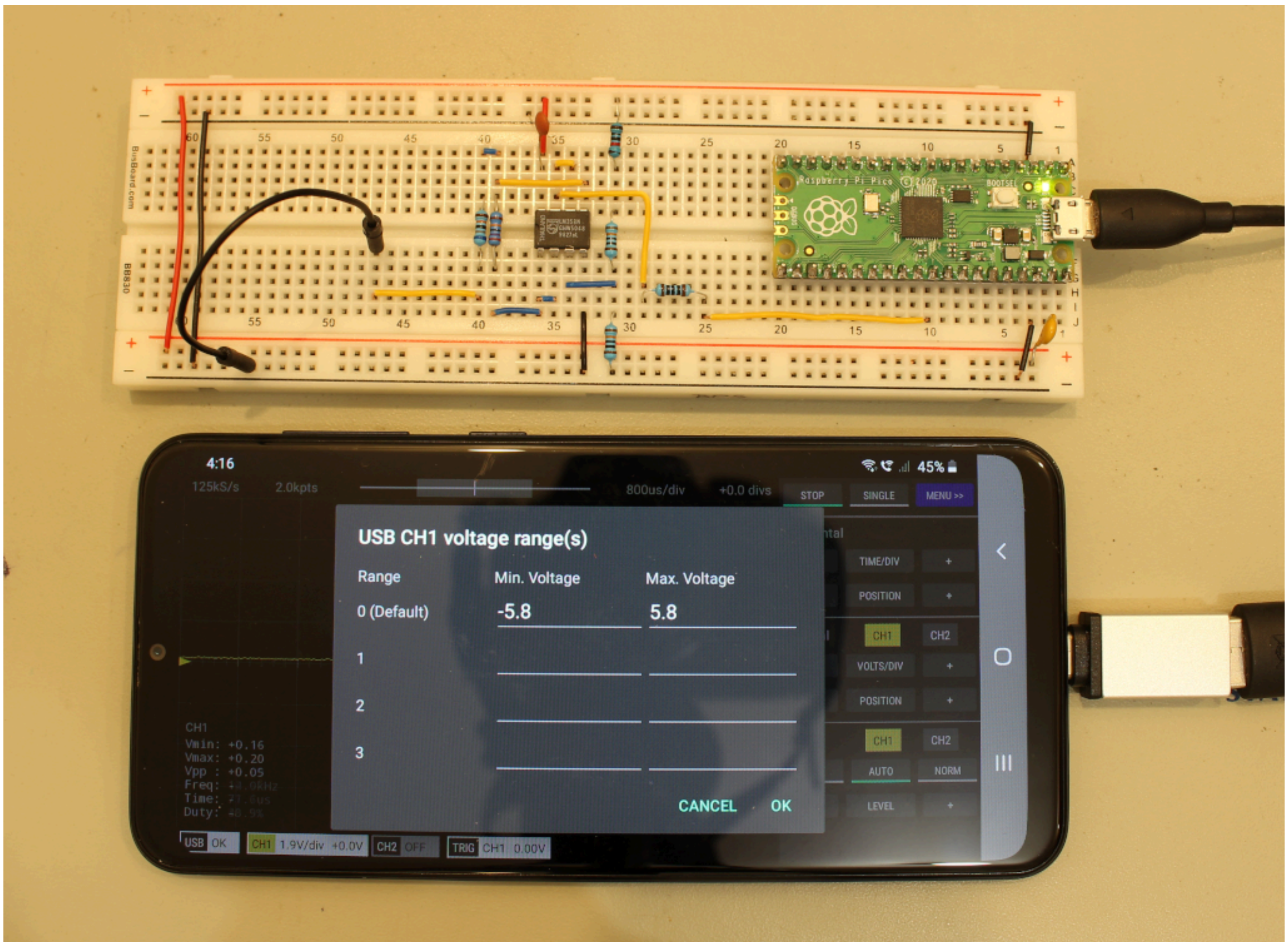


> *Caution. When the LM358 non-inverting input is floating the output might be a higher voltage than the maximum recommended voltage of the ADC pins (VIODD+300mV according to the Pico datasheet). To prevent possible damage to the Pico you might want to only connect the ADC pin to the front end once your signal source is connected to Vin. Having said that, I've applied voltages of over 4V to the ADC pins without any apparent damage to the Pico (with a 100R current limiting resistor in series). Future designs will add over-voltage protection so that you won't have to worry about this.*

Now we're ready to power on the Pico. Plug your OTG (on-the-go) cable/adapter into your Android device. Connect the other end to the Pico using a standard USB cable (USB-A to USB Micro). The LED on the Pico should light up and then start blinking about once per second.

Configure the voltage range in the Scoppy app

Open the Scoppy app and Android should ask your permission to connect to the Pico (you might have to tap the Run button).
Now connect the two Vin inputs of the frontend to a voltage source somewhere between -5.8V and 5.8V. You should see a horizontal trace on the screen. You'll notice that the trace is at the wrong voltage. E.g. An input voltage of 0V will be displayed at 1.65V instead of 0V. This is because the Scoppy app thinks the input voltage range is 0V to 3.3V (the default) rather than our -5.8V to +5.8V. To fix this, tap the Ch 1 badge at the bottom of the screen. A menu should be displayed. Tap Settings and then Voltage Ranges. Change the first range so that the min is -5.8V and max is 5.8V. Tap OK and Scoppy should now show the trace at approximately the correct voltage level.



For more accurate calibration of the voltage ranges see here.

> *The author makes no warranty, representation or guarantees regarding the suitability of this design for any particular purpose. Nor does the author assume any liability arising out its use and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages.*

See Also

Documentation Index
Scoppy on GitHub
Using the App
Scoppy Forum & Support
FHDM Store

Scoppy - Oscilloscope and Logic Analyzer

Scoppy - Oscilloscope and Logic Analyzer

fhdm-dev

Scoppy is an oscilloscope and logic analyzer powered by your Android phone/tablet and Raspberry Pi Pico or Pico W.