Raspberry Pico Function Generator- Documentation

This page describes the usage of a function generator made for the raspberry pico and guides through assembly of the according PCB and software configuration. The original project is developed by *rgco* on instructables.com, as seen here: `https://www.instructables.com/Arbitrary-Wave-Generator-With-the-Raspberry-Pi-Pic/`

1 PCB functionality

Technically the shown module is an "Arbitrary Wave Form Generator" (AWG) that uses a series of defined voltage setpoints, occuring at a specific time, to interpolate signals. In contrast to classic function generators this enables to produce totally arbitrary wave forms and not just periodic types such as a sine wave. The PCB is a rather simple design of a 8- Bit- DAC. Unfortunately the Raspberry Pico itself cannot output analog singnals, so this is where the PCB is needed. Main advantage of the Raspberry Pico being used in contrast to other microcontrollers is the possibility to generate high sample rates. Based on the RP2040 chip the Raspberry Pico has two 32- Bit-Cores running at 125 MHz. To create analog signal wave forms, the PCB processes digital inputs coming from the Raspberry Picos GPIO Pins. The original founder of this project (as mentioned above) has developed a specified solution for this process that involves a python script running on the Thonny environment. It is written to communicate with 8 GPIOs which is why the Function Generator is limited to 8 Bit and a very specific DAC design based on a R2R ladder.

A R2R structure requires 7 "R- Resistors" and 9 "2R- Resistors". The PCB however lines up some of the resistors in parallel so there are only resisitors with the same value needed. Besides the resistors and the Raspberry Pico there is a switch on the board, which secures the microcontroller and software to communicate without issues. In detail the switch is hooked up to the reset pin of the Raspberry Pico and has to be pressed every time the signal has to be changed. Also for starting the programm for the first time after powering the Pico and in buggy cases the switch needs to be used.

Signal precision depends mainly on the used resistors. The output frequency is determined by the Raspberry Pico and therefore is very exact. On the other hand the amplitude of the signal gets affected by differences in the resistor values. As a result there is quite a variety in the amplitude when using standard resistors due to scatter. It is highly important to check the resistors for deviation from their nominal setting.

2 PCB assembly

Parts needed for the PCB assembly are shown in the table below: 1:

Type of part	**Specs**	**Quantity**
Resistor	2K, 0.25W, 1%	23
Switch	6 x 6 mm, 12V	1
Pin socket, solderable, female	1 x 20	2
Pin socket, solderable, male	1 x 2	1
Raspberry Pico	–	1

Table 1: PCB parts

Soldering the parts is straight forward, refer to the picture below. As described in the section above, there is no need to position the resistors in a specific way- The right resistor values are built up with the PCB layout automatically. Just make sure to use resistors with a small amount of scatter ($< 1\%$) and with the exact same value for every unit.

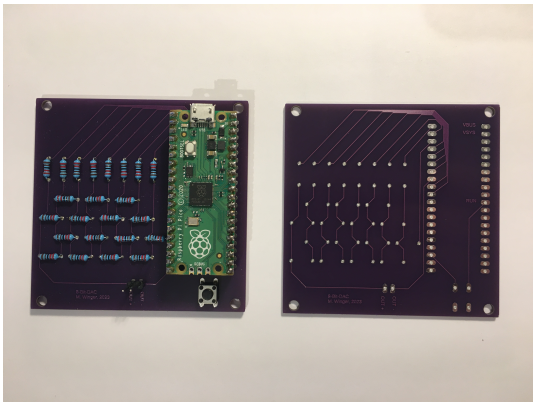

Figure 1: Soldered PCB, frontal view

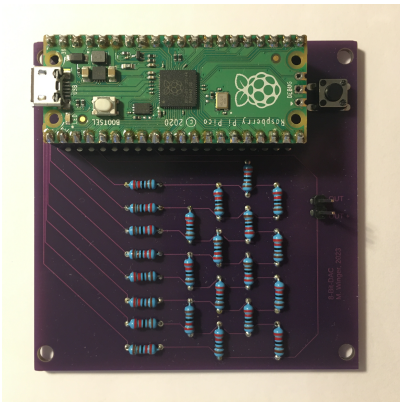

Figure 2: Soldered PCB, close up view

3 Software requirements and usage of the Function Generator

The Function Generator is running on a python script that can be easily controlled via the Python IDE "Thonny". Download file can be found at `https://thonny.org/`. Make sure the Raspberry Pico is running Micro Python firmware. The original script is a demo of various wave forms that are possible with the system. You can download the according file with the link at the very beginning. The same file plus a reduced version to just generate a single wave form is also provided on GitHub under the following repository: `https://github.com/BourneAgainMax/ArbitraryWaveFormGenerator_RaspPico.git`.

In both version there are some functions and definitions made at the beginning of the code. Scrolling down reveals the settings for the wave forms that are going to be output. By changing the right values, things as frequency, amplitude, offset etc. can be set to the desired configuration (compare with 4).

PCB usage- Steps to generate a wave form:

First make sure the Raspberry Pico is plugged in and detected by the Thonny software with the Micro Python firmware uploaded. Open the programm in Thonny as shown in 3.

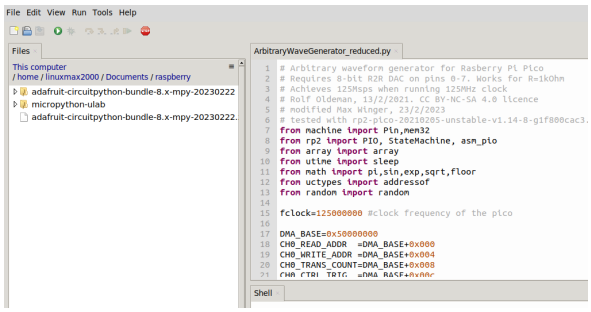

Figure 3: Thonny startup window after opening the Python script

The wave forms can be set at the end of the script. 4 shows the settings of the sine wave which is given as an example in the reduced version of the script.

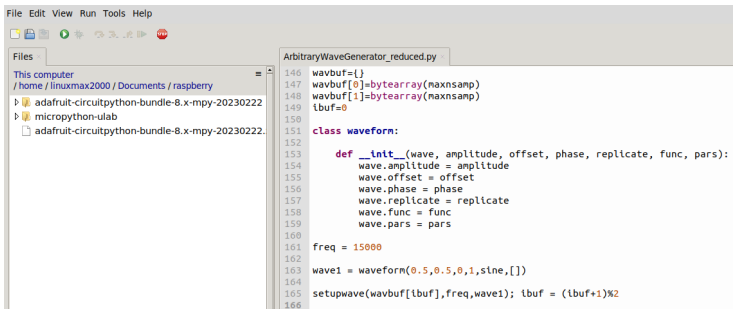

Figure 4: Sine wave example

Connect a device or measuring equipment to the output pins of the PCB and press the reset switch. The python

script is then ready to be started in Thonny (sometime Thonny requires stopping the program at first befor being able to launch it). Unless something is changed the Pico will be producing a wave form. After every modification perform the exact same routine which means resetting the Pico on the PCB, stopping the script in Thonny and then starting with the new changes.

M. Winger, 2023