Using a Raspberry Pi for Wheel Cutting

Adding a Stepper Motor to a Dividing Head

Mark Baird

Cutting wheels is a fairly straightforward process of advancing a wheel blank by a defined angle before cutting another tooth with a cutter such as those supplied by P. P. Thornton. Set-ups to achieve this vary but my approach is to mount a Vertex BS0 dividing head on the bed of a milling machine as shown in **Figure 1.** The dividing head is a 40:1 ratio and is supplied with a number of dividing plates.

Other approaches involve holding the blank in a lathe and using a milling spindle to hold the cutter. The indexing here is usually achieved by having a dividing plate at the back of the headstock for direct indexing.

The approach using a dividing head is well-established, although there are other options commercially available. For the wheels I have needed to cut, such as 30, 90, 96 or 144, it has worked well, although you need to take considerable care when advancing ten 24ths of the dividing plate for a 96-toothed wheel, for example. I have not done it often, but it’s possible to end up with an unfortunate-looking disaster when cutting the final tooth! Whilst versatile, there are also considerable gaps in the divisions possible. Recently, I needed to cut a wheel with 149 teeth which of course wasn’t possible with the available dividing plates. Nor was it possible to use the combinations of other plates to make the required dividing plate which is how I’ve managed before. So how was I to get around this problem?

In order to make effective use of my existing equipment, I decided to look into adapting the dividing head to be advanced by a stepper motor which could be driven by an inexpensive single-board computer. Step forward the Raspberry Pi!

The Raspberry Pi is a credit card sized computer, with a four-core processor, USB ports, HDMI output and an all-important set of programmable input/output pins known as the GPIO (General-Purpose Input/Output) port, **Figure 2**. It was originally aimed at the education sector to introduce children into computing and coding, but priced at around £34 it has a considerable following of hobbyists who use the unit for all sorts of other applications.

The GPIO port offers 27 programmable pins which can be set for input or output operations. Input pins can, for example, be set to monitor switches while output pins can be used to control things like LEDs, relays or motors which, of course, is what we are interested in doing. The GPIO outputs supply a voltage of just 3.3V and are not intended to supply any big current, so it can’t directly drive a powerful stepper motor. To achieve this, the motor needs to be run by a separate driver. There are lots of these available for CNC conversions of machine tools and 3D printers; I chose the widely available TB6600 generic stepper driver. It takes in a low-level signal to control the motor from the Raspberry Pi and drives the motor via a separate power supply. **Figure 3** shows how it is all wired up.

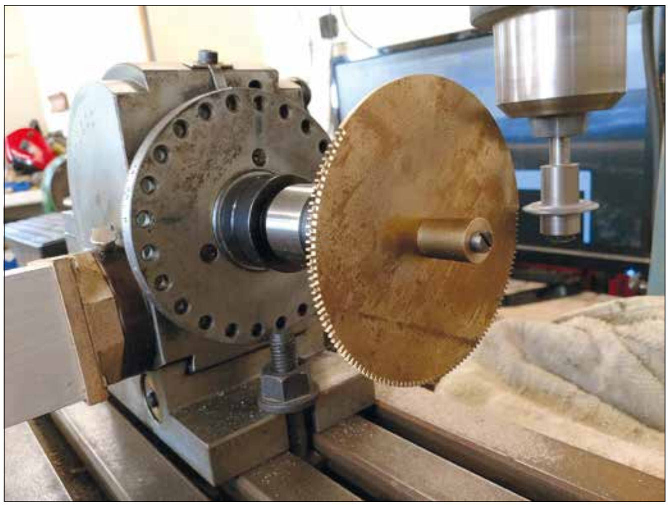


Figure 1.

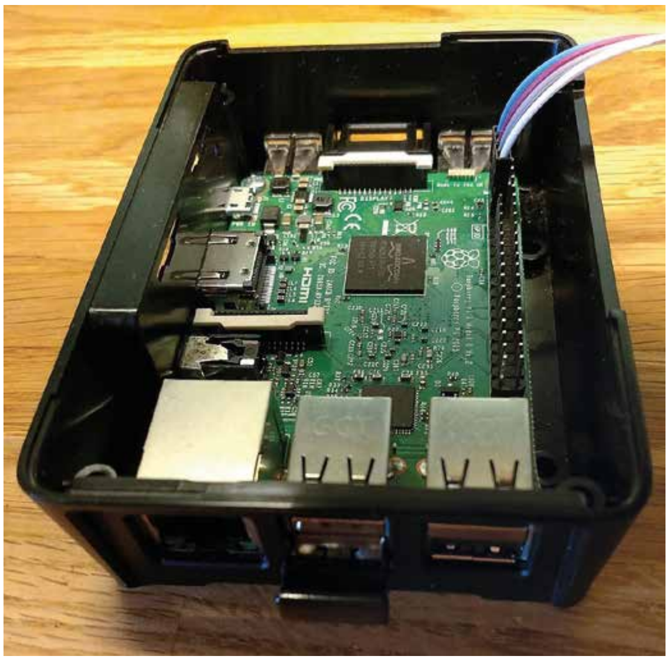


Figure 2.

You can see that three signals are required to drive the motor: one to enable it (ENA) , one to set the direction (DIR) and the final signal is to advance it by 1 step (PUL).

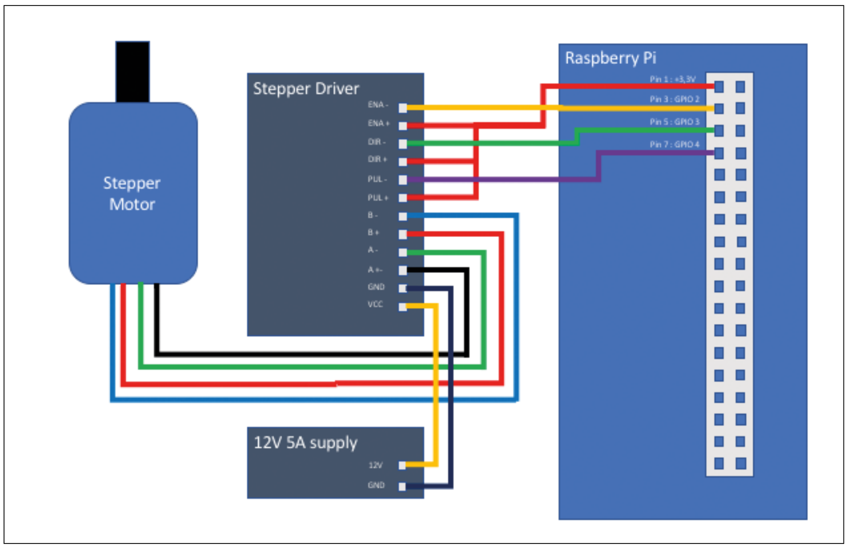


Figure 3.

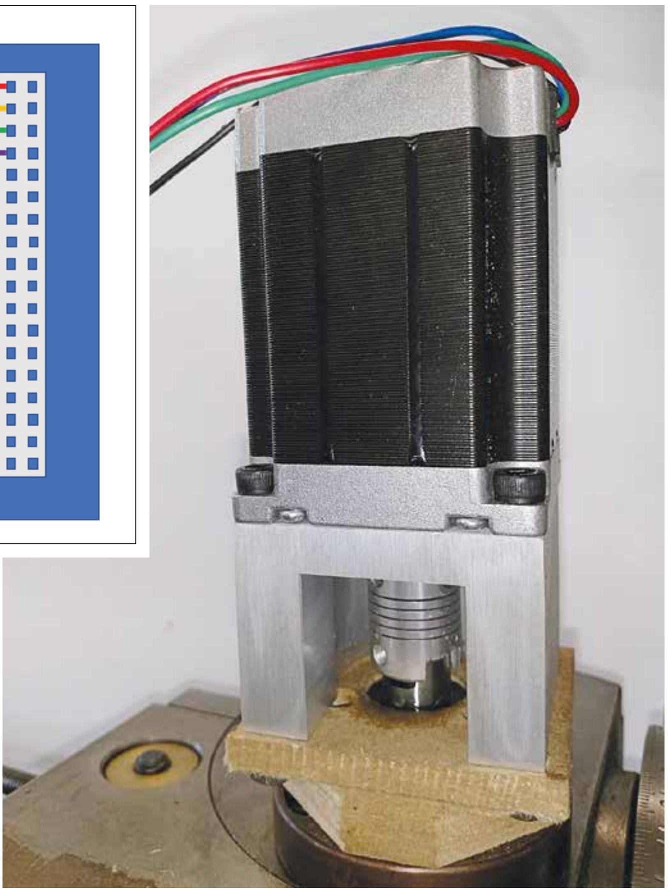


Figure 4.

The TB6600 stepper driver has a set of DIP switches which you need to set up. There is a table on the front of the driver which you can use. For this application, you need to set the switches so that the stepping mode is for 200 pulses per revolution and the output current 2.5A. You can experiment with setting the current lower, but I found that this didn't give the motor enough torque. This resulted in missed steps – not desirable for this application!

Moving on to the stepper motor, I've chosen a NEMA 23 model with a torque of 1.9Nm and a stepping angle of 1.8 degrees, which equates to 200 steps per revolution. With a 40:1 worm on the dividing head, this means that a wheel will be turned by 8000 steps of the motor.

Let's go back to my troublesome 149-tooth wheel to see how this works out. If a wheel takes 8000 steps to rotate, then we will need 8000 / 149 steps for each tooth. This is a seemingly inconvenient 53.691275167785235! Initially this looks like we've got exactly the same problem with the stepper motor as we had with the dividing plates except, of course, the stepper motor has much finer movements than are achievable from a division plate.

The solution to this fractional number of steps is very similar to how diagonal lines are drawn on computer displays. Computer displays are a bit like stepper motors in that they display entire pixels on a display in the same way that the stepper motor advances only by discrete steps. The technique required here is Bresenham's line drawing algorithm. I won't go in to detail, but will explain the steps we make to achieve 149. These will be made up of a distribution of 53 and 54 steps to make up the 8000 required for the entire wheel. The difference between 53 and 54 steps is equal to only 0.045 of a degree. The distribution of steps sequentially for this wheel will be:

```
[53, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 54, 53,
54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 54, 53, 54, 54, 53, 54, 54,
53, 54, 54, 53, 54, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54,
54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 54, 53, 54, 54,
53, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 54, 53, 54, 54, 53, 54,
54, 53, 54, 54, 53, 54, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 53,
54, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 54, 53, 54,
54, 53, 54, 54, 53, 54, 54, 53, 54, 54, 54, 53, 54, 54, 53, 54, 54, 53,
54, 54, 53, 54, 54, 54]
```

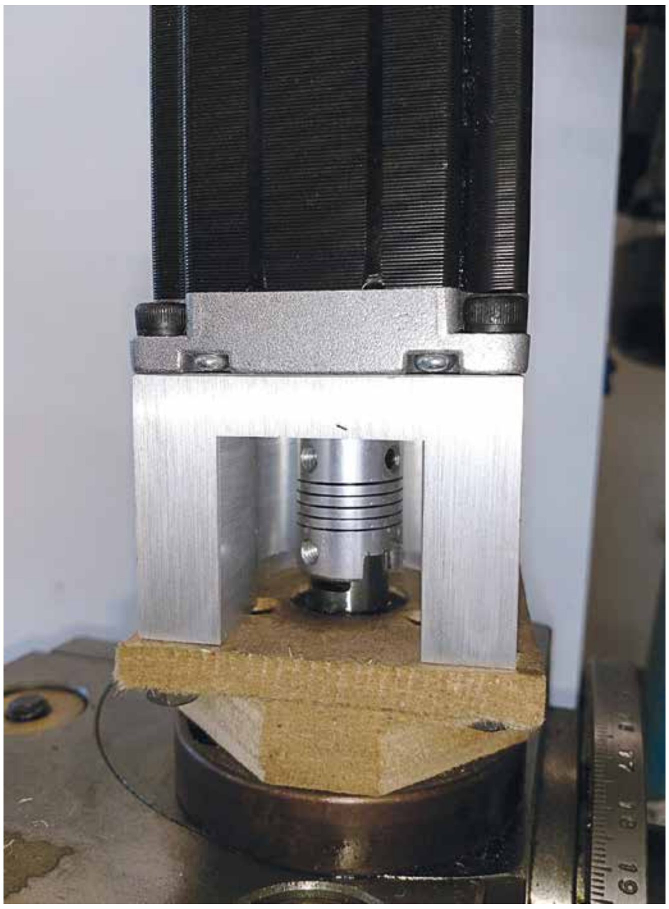


Figure 5.

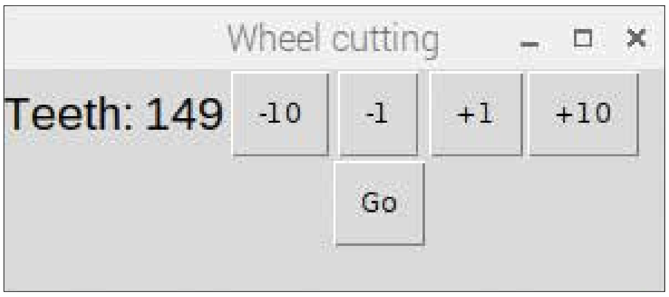
If you can be bothered to go through the sequence, you will find we have 149 numbers which add up to 8000. Using this algorithm, pretty much any number of divisions can be achieved.

Attaching the stepper motor to the dividing head requires a little bit of ingenuity but is certainly not beyond most readers and there are plenty of brackets and couplings available to make this job easy. **Figures 4 and 5** show how I've achieved this with a modified stepper motor coupling to attach the motor to where the turning handle of the dividing head was once mounted.

The completed set-up needs an HDMI monitor, keyboard and mouse connected to the Raspberry Pi and, of course, some software to send the stepping signals to the stepper motor driver. To make life easier for anyone who wants to make a similar unit, I've written software for this using the easy to understand Python language and published the details on the code sharing site github.com. If you go to the web address: https://github.com/mbcoder/wheel-cutter you will find all the information needed to set up the Raspberry Pi. I have also included a shopping list of possible places to acquire the parts.

I have also written a short wiki page to explain how to connect it up safely and test things before it is properly powered up: https://github.com/mbcoder/wheel-cutter/wiki/Connecting-it-up

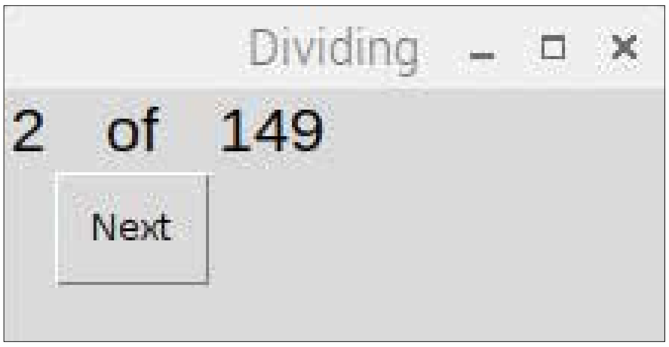
The Python application has a basic user interface which when started up allows the user to select the number of teeth needed for their wheel, **Figure 6**.


Wheel cutting

Teeth: 149

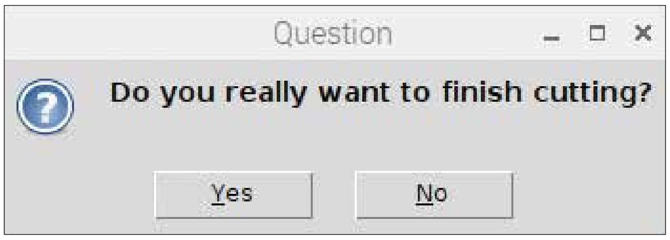
Figure 6.

Once the number of teeth is selected, press GO and the software will work out the steps required for each tooth. A window is then presented which allows for the wheel to be advanced for each tooth to be cut, **Figure 7**.


Dividing

2 of 149

Figure 7.


Question

Do you really want to finish cutting?

Figure 8.

Once you are finished you can close the window to select another wheel cutting division. It does, however, ask you if you have really finished, **Figure 8**.

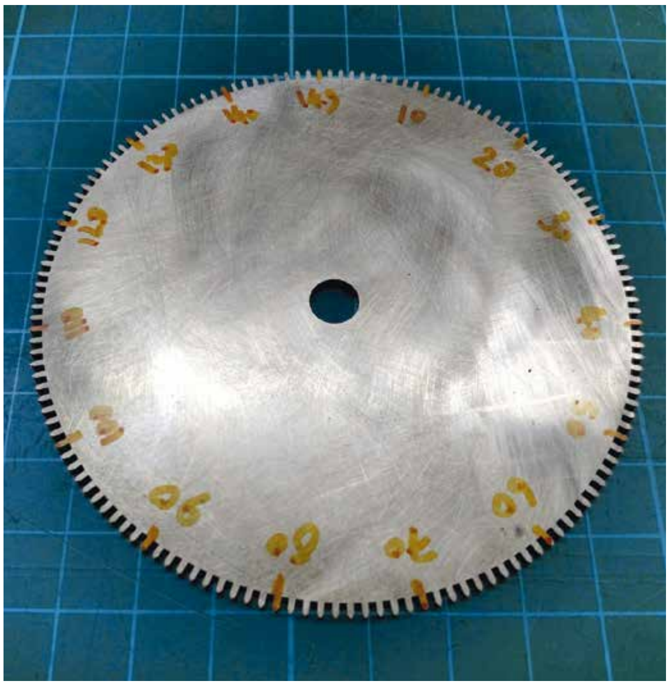


Figure 9.

Finally, to prove it works, **Figure 9** shows my 149 tooth wheel ready for crossing out and mounting.

Before you go ahead and spend time cutting anything like this, however, I would suggest that you set the application to cut four teeth and simply observe that the dividing head returns to the initial position afterwards. This give you confidence that everything is working correctly and that the stepper isn't missing steps.

If you find any issues with the software or indeed have ideas for improving it, you can submit questions or requests on the github.com webpage. I will try to assist.