# Pure Data with a PICO

First we are going to take a brand new PICO device and put Circuit Python on it. Then we are going to add the MIDI libraries. Then we are going to fire up Thonny and write a program that will send a MIDI note. Finally we are going to open up Pure Data and make a patch to receive the note and play a sound.

## Put Circuit Python onto a PICO

1. Need to download the latest version from Adafruit here: https://circuitpython.org/board/raspberry\_pi\_pico/
2. Now press the SEL button on the device and plug it into the computer. The folder should appear.
3. Drag the UF2 file onto the drive and wait for it to reboot.
4. Show the new folder and the empty lib.

## Add the MIDI libraries

1. We need to put the drivers for the devices into here. These are python files which you can find here: https://circuitpython.org/libraries
2. Mention that we've put them into a local lib directory to use immediately. Copy the contents of that folder into the lib. Now we can import them and use them.

## Create the Python code

1. Mention that we are going to use the Thonny editor for the program.

We are going to open the Cheese Synth program and take a look at how it works.

1. Explain that the CheeseSynth provides drum sounds which are made up of sound samples and generated sounds. It also contains a very simple synthesizer which can generate sine and triangle waves and mix them.
2. Show the contents of the **Cheese Synth** folder. Explain that there are sample files for some of the sounds and sub-patches to break down the application.
3. Open Pure Data (make sure that audio is switched on and the control first)
4. Open the patch **Cheese Synth**
5. Use **Edit->Zoom in** to zoom into the patch.
6. Explain the difference between Edit Mode (Hand cursor) and Run Mode (arrow cursor).
7. Use **CTRL+E** to enter Edit mode.
8. The first thing we need to do is get a way of triggering the drum. Explain the "Bang" object.
9. Add a Bang with **PUT->Bang**
10. Use **Edit->Zoom in** to zoom into the Bang.
11. Explain that objects have properties.
12. **Right click** the **Bang** object and open the **Properties** menu for the Bang.
13. Give the object the label "Kick".
14. Explain that clicking Pure Data objects have an input (at the top) and an output (from the bottom). When the Bang is clicked it will send an output signal.
15. Go into Play mode by pressing **CTRL+E**. The cursor will change to an arrow.
16. Click on the Bang and show that it flashes.

## Make a line

Now we are going to add a line.

1. Explain that we want to make a waveform that will be our drum sound. A waveform is a signal that changes over time. The starting point of our waveform will be a signal that goes from 0 to 1 over the length of the kick drum sound. We are going to use a line object to make this and then do some maths to it to make a sound.
2. Explain that we need to tell the line object what kind of wave we want. We're going to give the line two messages. The first is a single number, which is the start value for the ramp. The second contains two numbers, the end value and the time (in milliseconds) that the ramp will last.
3. Ensure you are in Edit mode (press **CTRL+E** if you need to).
4. Add a Message object (**Put->Message)** and enter: 0,1 200:

A picture containing text

Description automatically generated

1. Explain that some objects are "built in" and some are specified by name. The Line object is specified by name. Add an empty object (**Put->Object)** an empty object and give it the content **line~**

A picture containing text

Description automatically generated

1. Explain that the tilda (twiddle) on the end of the object means that it sends out an analogue signal rather than an event. In the case of our line it will be a wave that takes 200 milliseconds to go from 0 to 1. This would not be a good drum sound. We need a sound wave.
2. First we make the range of our line a bit bigger. We can multiply all the values in the line by 20. Add an empty object (**Put->Object)** an empty object and give it the content **\*~ 20**

Text

Description automatically generated with low confidence

1. Explain that the multiplication is performed on each value of the waveform as it goes through the object. So we now have a line that goes from 0 to 20.
2. Now we can convert it into a wave. For some reason Pure Data only has a cosine function, so we can feed the line through that.
3. Add another object and give it the content **cos~**

A picture containing text

Description automatically generated

1. Explain that the cos function multiplies the value by 360 degrees and then takes the cosine of that value. Now we have a signal that contains 20 waveforms.
2. Explain that we now need to send the sound out via a digital to analogue converter (dac)
3. Add another object and give it the content **dac~**
4. Now we should have all the objects that we need:

Text

Description automatically generated with low confidence

1. Explain that we now have to "wire them up". Connect them together and explain that the audio signal is drawn as thicker.
2. Change to play mode with CTRL+E and click the Bang to play the sound. Note that you should prime the audio output using the test option: **Media->Test Audio and MIDI**
3. This will make a boop. Not a very good kick drum sound.
4. Explain that we need to do some extra processing. Open up the **Kick Graphics** patch and show how it uses the square root to compress the values. Show how you can make graphs. Change the multiplication value to change the sound that you get.

# Auto Drummer

1. Explain that Pure Data provides a "metro" object that can produce regular trigger outputs.
2. Open the **Auto Drummer** patch and explain that this contains two drum sounds triggered by two metronomes.
3. Run it and marvel at the solid beats.