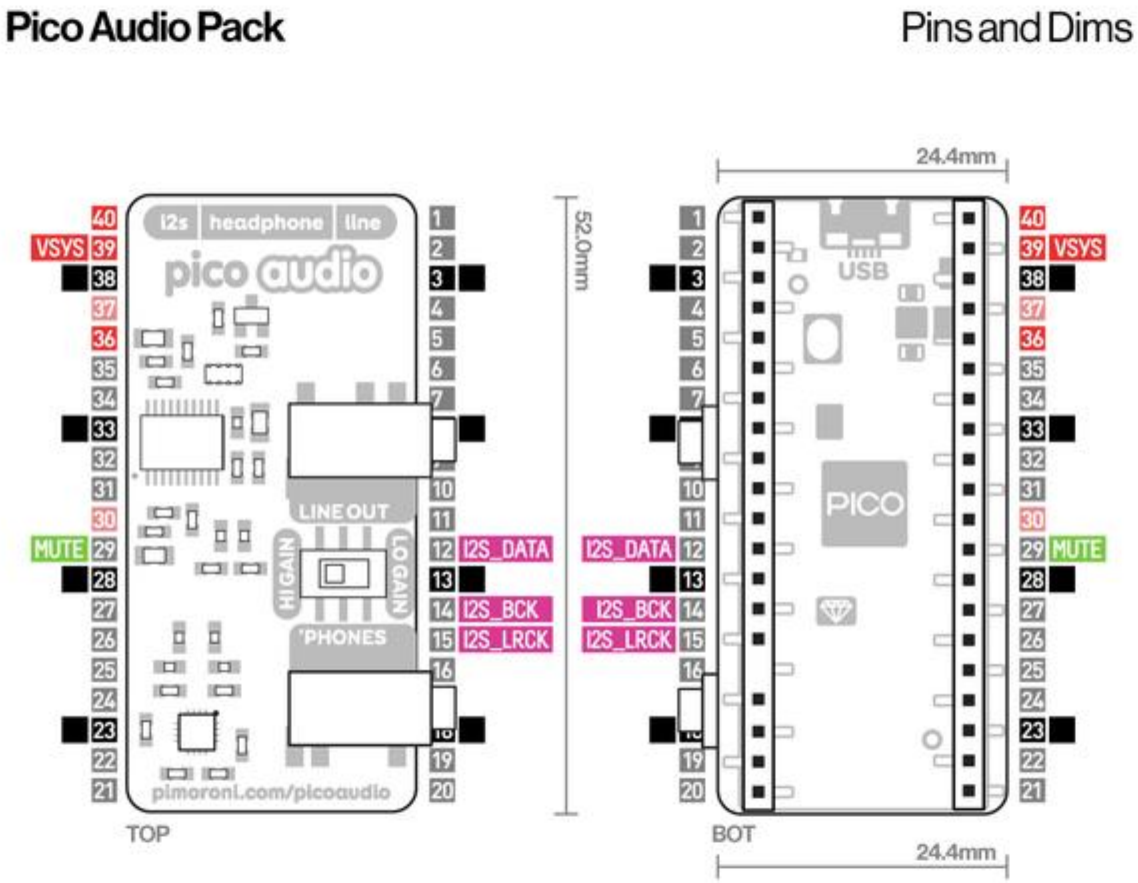
Getting Started with Pico Audio Pack

Pico Audio Pack provides a high quality stereo DAC which uses **I2S_DATA**, **I2S_BCK**, and **I2S_LRCK** to recieve audio data.

We also offer a **MUTE** pin which can be used to silence the output.

You can use Pico Audio Pack with the I2S audio examples provided by Raspberry Pi in their experimental examples (usb_sound_card and sine_wave_i2s). You'll need to **#define PICO_AUDIO_I2S_DATA_PIN** and **PICO_AUDIO_I2S_CLOCK_PIN_BASE** to 9 and 10 respectively to tell the examples which pins are being used for the audio data.



Pico Audio Pack Demo

This demo requires the `pico-extras` repository to compile - https://github.com/raspberrypi/pico-extras.
You should clone it alongside `pico-sdk` and `pimoroni-pico`:

```
git clone https://github.com/raspberrypi/pico-extras
```

And adjust your `cmake` configure command to include the path:

```
cmake .. -DPICO_SDK_POST_LIST_DIRS=/path/to/pico-extras
```

If you're using Visual Studio Code you can add this to `settings.json`:

```
{
    "cmake.configureSettings": {"PICO_SDK_POST_LIST_DIRS": "/path/to/pico-extras"}
}
```

Files

popcorn: add converter
Initial Release
Initial Release
randomly speed up mandelbrot demo :-)
Update hello_sleep.c
Use latest version of upstream pico_sdk_import.cmake (#12)
Initial Release
Initial Release
remove net demo which duplicates tinyusb/net_lwip_webserver
Initial Release
add vga schematic
Initial Release
Use latest version of upstream pico_sdk_import.cmake (#12)

README.md

Note that most of these examples are neither fleshed out nor well documented. They mostly serve the purpose of playing with/testing particular areas of functionality (mostly audio/video related)

Finally, you may wonder why many of these demos set the system clock to 48Mhz. The reason is that until we had physical chips, we were running at a fixed 48Mhz system clock using an FPGA. Most of these examples were written before the RP2040 design was final, so they were all developed with that fixed 48MHz system clock. As a result some of the examples do things in a way that you wouldn't necessarily need to if you had more clock speed available (which you do), but on the plus side, you have that much more time to do even more things!

Full Applications

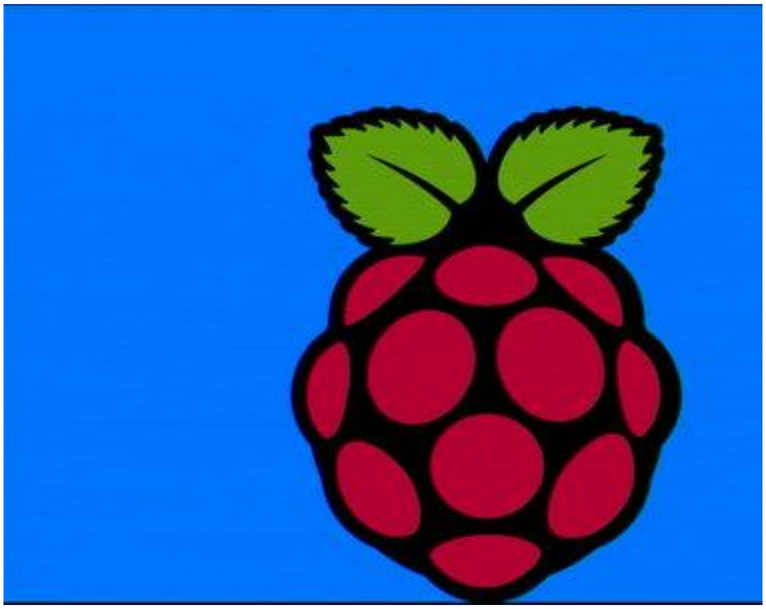
Name	Description
popcorn	This is a movie player for 30fps 320x240 movies with 44100 stereo sound, read in a custom format from SD card... it can even play backwards :-) Sample movie file linked from here.
usb_sound_card	A no frills but functional USB sound card... hooked up via our old (pre TinyUSB) USB device stack. Keeping it around as it works nicely!

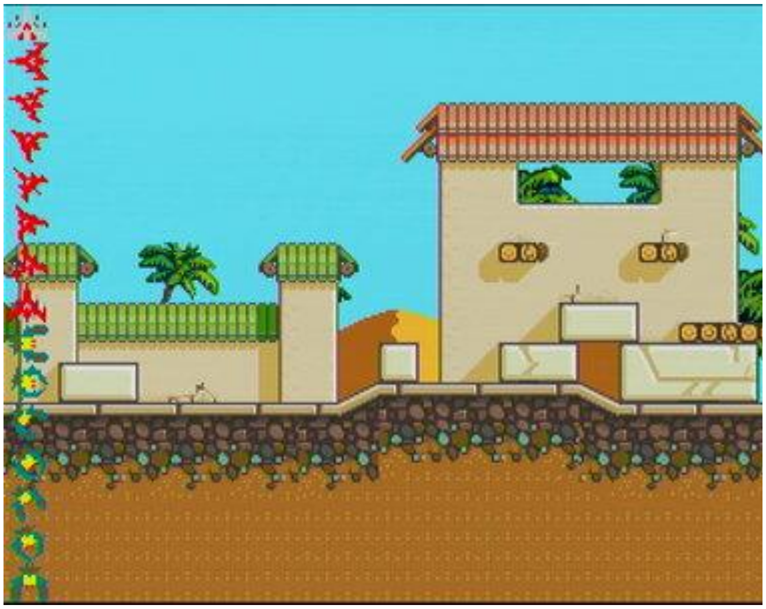
Audio

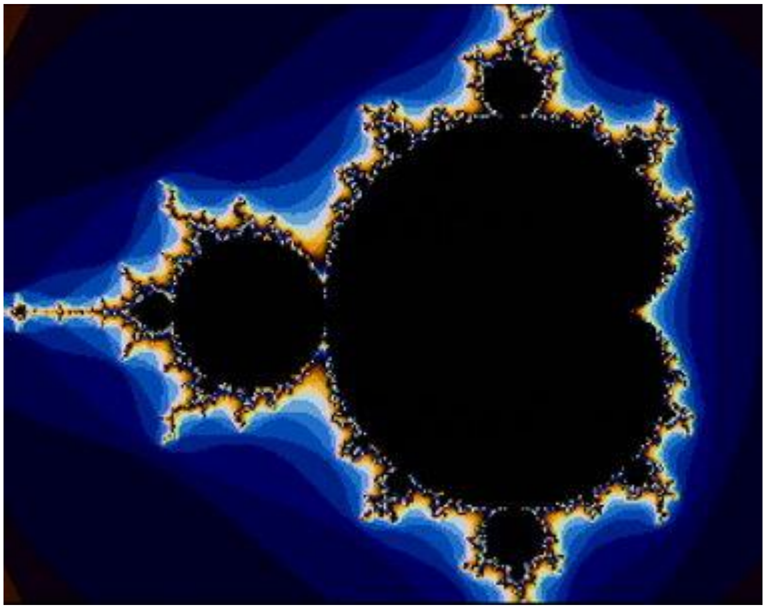
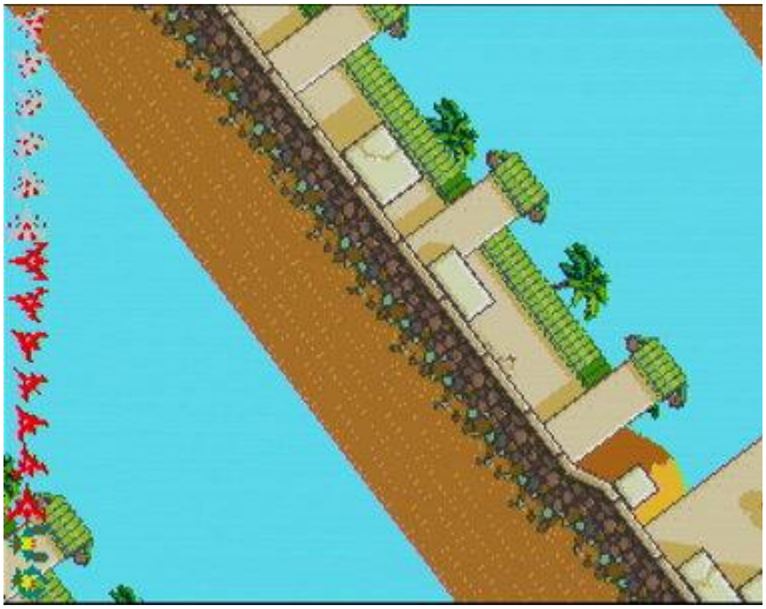
Name	Description
sine_wave_i2s	A simple sine wave audio output using I2S.
sine_wave_pwm	A simple sine wave audio output using PWM.
sine_wave_spdif	A simple sine wave audio output using S/PDIF.

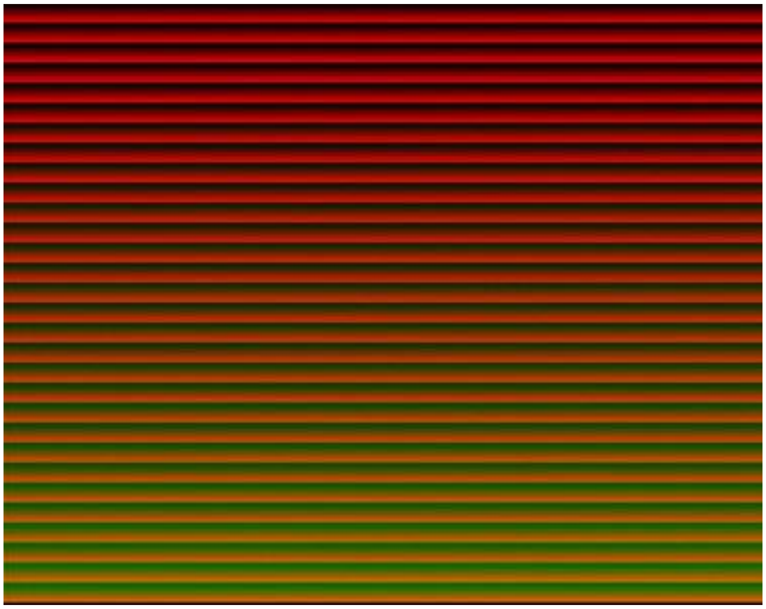
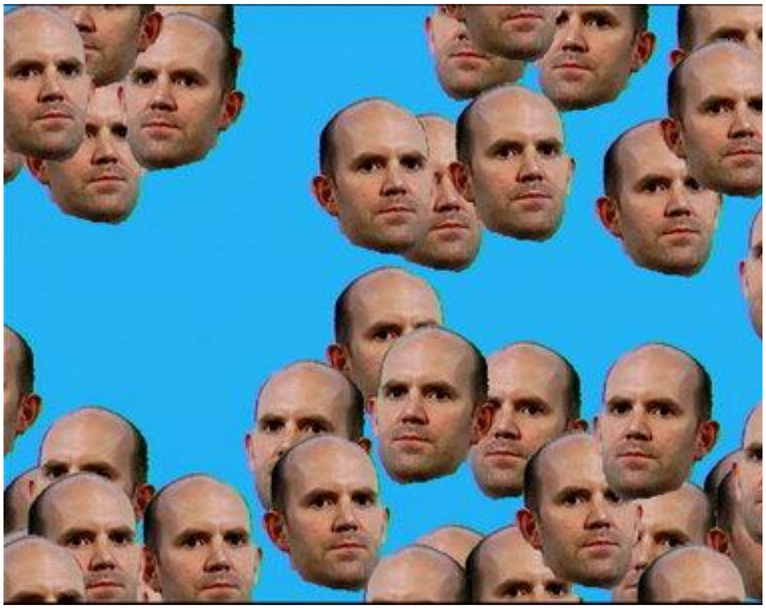
Scanout Video

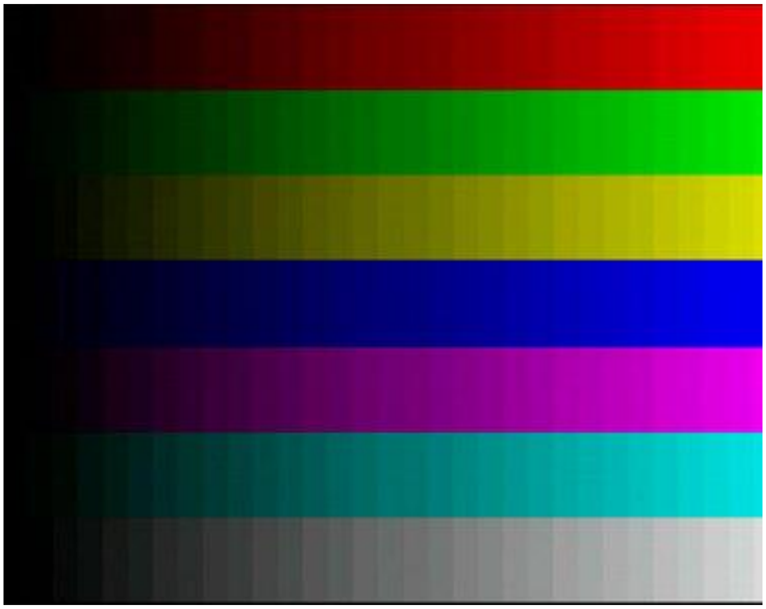
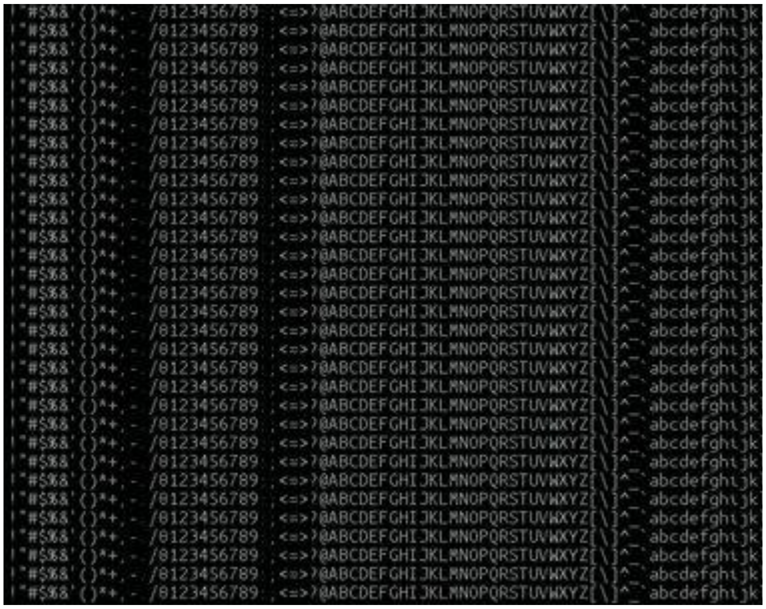
In *scanout* video, every pixel is driven by the PIO every frame, and a framebuffer is not (necessarily) used (which is useful when you only have 264K of RAM).

Name	Screenshot	Description
demo1		So named because it was the first demo program written that used video.. it is a bit dated now and hails from a time where there was *much* less RAM on the RP2040

Name	Screenshot	Description
flash_stream		Streams video data out of flash fast enough to drive 640x480x60fps bitmap display
hscroll_dma_tiles		A horizontal scrolling test app which uses a second video plane (PIO) to overlay sprites

Name	Screenshot	Description
mandelbrot		A mandelbrot generator using a 320x240x16 framebuffer
mario_tiles		Confusingly named as a reference to Super Mario Kart's tiled psuedo-3D rendering. This is similar to hscroll_dma_tiles exc ept the whole tiled scrolling area is now rotated and zoomed.

Name	Screenshot	Description
scanvideo_minimal		A very basic video output generator which generates a test pattern
render		A very dated rendering library used by demo1 - avoid!
sprite		A small sprite library used by sprite_demo
sprite_demo		Some bouncing Eben heads

Name	Screenshot	Description
test_pattern		Display color bars
textmode		Shows off chained DMA to generate scanlines out of glyph fragments via DMA/PIO

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
```

The above are intended to be used with the VGA demo board as described in Hardware Design with RP2040 however it is possible to wire your own VGA output according to the following schematic: ![](scanvideo/Raspberry Pi Pico to VGA Connection Schematic.png)

Sleep

Examples of using low power mode; these use `pico_sleep` from pico_extras which is not yet stable.

Name	Description
hello_dormant	Demonstrates dormant mode

Name	Description
hello_sleep	Demonstrates low power sleep and waking from RTC

Stdio

Examples of adding additional stdio support. Note the interface for stdio drivers is not yet considered stable even though it is in the Pico SDK

Name	Description
stdio_pio	Demonstrates adding a custom STDIO driver using a PIO UART