3.2

1. The processor isn’t really designed for this. It has other work to do. As processors have become faster in terms of overwhelming number-crunching brute force, the layers of software and hardware between the processor and the outside world have also grown in number and size. In response to the growing distance between processors and memory, PC-class processors keep many hundreds of instructions in-flight on a single core at once, which has drawbacks when trying to switch rapidly between hard real time tasks.

2. LEDs and push buttons.

3. The pull instruction takes one data item from the transmit FIFO buffer, and places it in the output shift register (OSR). Data moves from the FIFO to the OSR one word (32 bits) at a time.

4. The state machine needs to be told which GPIO or GPIOs to output to. There are four different pin groups which are used by different instructions in different situations. The OSR is able to shift this data out, one or more bits at a time, to further destinations, using an out instruction. The out instruction here takes one bit from the data we just pull-ed from the FIFO, and writes that data to some pins.

5. A method ws2812_default_program_config configures a PIO state machine based on user parameters. The date should be loaded to TX FIFO and then transmitted to the state machine to program a PIO state machine.

6. The function pio_sm_put_blocking( ) writes a word of data to a state machine's TX FIFO, blocking if the FIFO is full. The function is realized by including the header file.

7. Gloss over the details of how the assembly program in the .pio file is translated into a binary program, ready to be loaded into the PIO state machine. pioasm can also be used directly, and has a few features not used by the C++ SDK, such as generating programs suitable for use with the MicroPython PIO library.
Command line:

```
    mkdir pioasm_build
    cd pioasm_build
    cmake $PICO_SDK_PATH/tools/pioasm
make
    And then invoke as:
    ./pioasm
```