ELEC 327 Lab #3

Liam Brady
Due: Saturday 2/7/26

To accept input, the method `GetNextState()` now takes in an integer, either 1 when the button is not pressed, or 0 when the button is pressed. This comes from the fact that the button connects the pin to ground, and to make sure that the GPIO won't be floating, we make sure the IOMUX enables the pull up resistor, so that when the button is not pressed, the GPIO pin will be pulled high.

From there, whenever the button is pressed, a counter is tick. This allows the driver to then know how long the button is down for. When it's released, the driver can then tell what behavior to implement. Either ignore it if it is less than 5 ms (this implements the debouncing), call it a long press if it is longer than a second, and call it a short press otherwise.

With these presses, as long as a new part of the state defining what mode the clock is in, the state machine can easily move through the different clock modes, as well as easily do different behavior based on the clock mode using a switch statement.

Some assumptions are that there will be no glitches with the button for any holds more than 5ms. This is a good assumption as it makes writing the debouncing code much simplier, but there are lilkely some situations where this assumption could turn into an issue.

To implement the extra credit, to change the LED brightness I added another member to the time state struct, called brightness. This brightness value rannged between 0 and 14, and would also be incremented by a button press while in brightness set mode, and would be mapped to a duty cycle value, leading to the brightness of the LEDS changing.