C code:

The main structure is made up of 3 functions and 2 interrupt functions (in addition to the usual main method and initialization of gpio functions). The 3 functions include forward_state, previous_state, and update_leds. Both the former functions call update_leds at the end of their method. The led_state is kept in a variable that is initialised to zero in the main function at the beginning of the program. For the sake of this demo the LEDs are configured to be pins 1.0 for the Red LED and 2.0 and 2.1 for the RGB LED just so that there is a colour differentiation to make it more obvious what state we are in.

The main function includes all the configurations in addition to init_gpio where we set inputs, outputs, their directions, enable them as pull up resistors, etc. Update LEDs will set or clear the respective LEDs according to the led_state variable that is updated upon a button push interrupt or uart interrupt through an update on RXBUF. Depending on if the user wants to jump states or increment/decrement, forward_state, previous_state may be called and in either case update_leds will definitely be called. Forward_state increments the led_state variable by 1 or resets it if it is at 3 back to 0. Previous_state does the opposite (decrements and sets it back to 3 if it's at 0).

Python code:

Utilizes the serial class by setting the appropriate port and baudrate. A list of acceptable inputs in initialized to refer to when the user inputs a state. I use threading to run two continuous loops so that the program will constantly be reading or write whenever there's an input. Thus, if a button is pushed the read loop will output it because the button interrupt function in C calls update_leds which updates TXBUF which will notify the read thread. The user can input a direct state to jump to or increment to forward/decrement to previous with key words next/previous or n/p. This input triggers a uart interrupt and value is held in rxbuf. The interrupt will then check what the user wants to do. The state is updated and outputted in terminal whenever there is a change.