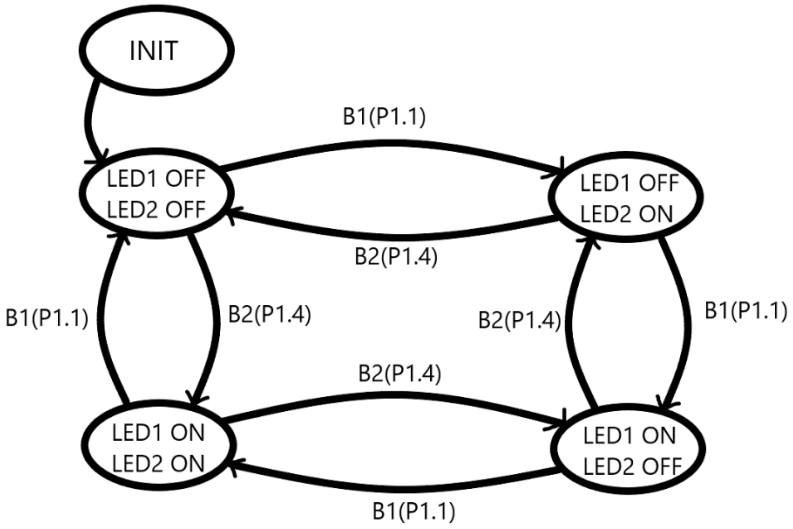
SYSC 3310 Project(UART Communicator)

State Machine

The chosen state machine has 4 states, each with different LED configurations (state 1 means both LEDs are off, state 2 means only LED2 is on, state 3 means only LED1 is on, and state 4 means both LEDs are on). The configuration source code provided configures this state machine as well as a method of traversing through the state machine using the buttons on the embedded processor. More specifically, button 1(PORT1.1) is configured to traverse to the next state in the state machine, while button 2(PORT1.4) is configured to traverse to the previous state in the state machine. Combined with all the required initialization and configuration, we form the state machine diagram above.



Configuration of the Embedded Processor (config.c)

The configuration of the embedded processor was implemented in c using functional programming. After disabling the watchdog timer, the program configures P1.1 and p1.4 as inputs, using pull-up internal resistors. The switch interrupts are then configured at port level, NVIC, and CPU. The LEDs(P1.0 and P2.0, P2.1, P2.2) are then configured as outputs and then initialized. The configuration of these ports is handled by the port_1_config and port_2_config functions. The UART pins are then configured as a secondary function, followed by the configuration of the clock and the baud rate. The UART is configured with 8 data bits, 2 stop bits, no parity bit, with the first bit being the least significant bit. Register EUSCI_A0 is then configured with interrupts for transmitting and receiving and is then configured in the NVIC. All UART configuration is performed in the configure_UART function. The EUSCIA0_IRQHandler function resolves the UART interrupt checking the receive buffer register(RXBUF) for ‘N’ and ’P’ to move to the next state or previous state accordingly. The configure_state function is used to configure the appropriate LEDs based on the current state as well as transmit the current state to the transmit buffer register(TXBUF). The remaining functions handle_next_action, handle_previous_action, and handle_state_action are used to implement the logic of traversing back and forth through the state machine. Overall config.c is responsible for implementing the state machine and configuring the UART port on the embedded processor.

Implementing the Graphical User Interface(UART_Communicator.py)

The GUI was implemented using the Tkinter python library, which contains the required button and label classes for the UI. The pySerial library was used to access the port and transmit ‘N’ and ’P’ to the receive buffer register, as well as to receive the current state from the transmit buffer register. Lastly, the continuous threading library was used to allow for multithreading required for receiving and transmitting simultaneously. The time library was also used to implement any needed delays.