**SIT315 M1.T1P QP4 - Reflection Report**

**System Architecture and Logic**

The project implements a whole interrupt-based system integrating Pin Change Interrupts (PCI) and Timer interrupts. The system employs two buttons on pins 8 and 9 (PCINT0 group) for event-based inputs, and Timer1 for time-based periodicity. Each button is individually controlled by its own LED, and a status LED automatically flashes every second through the timer interrupt.

The design decouples time-based from event-based logic. When the button is pressed, the PCI interrupt detects it immediately and sets a flag. The timer interrupt periodically runs every second, setting its own flag. The main loop polls these flags and services them, leaving the interrupt service routines to be fast and minimal.

**Interrupt Configuration and Usage**

For Pin Change Interrupts, I set up the PCICR register to turn on the PCINT0 group by setting bit 0. Then I set bits 0 and 1 of PCMSK0 to turn on pins 8 and 9 individually. The ISR uses digitalRead() to determine which button was pressed.

For Timer1, I employed CTC mode with OCR1A at 15624 for an interval of 1 second. The formula is: (16,000,000 / (1024 × 1)) - 1 = 15624. The prescaler of 1024 divides the 16MHz clock to produce the correct timing. The timer resets automatically when it reaches the compare value.

**Issues Encountered and Solutions**

The most difficult part was learning about register-level programming. At first, I did not realize how PCICR and PCMSK0 interacted, but studying the datasheet and experimenting with varying bit values led me to realize that PCICR allows pin groups to be enabled while PCMSK allows individual pins within these groups.

Another problem was keeping ISRs short. I realized that interrupt service routines should only set flags and perform rapid operations, not complex logic or Serial output. Eliminating all processing from the ISR to the main loop improved the stability and responsiveness of the system.

Understanding the timer calculation formula took time, but testing with different OCR1A values helped me see how it affects timing. This project taught me the importance of interrupt-driven programming for building responsive embedded systems.