This project successfully demonstrates the integration of both pin change interrupts (PCI) and timer-based interrupts to control RGB LEDs based on real-time sensor input and periodic timing logic.

I’ve used the analogue pin A0-A3, located in Port C, to input event driven responses to sensor state changes of 3 PIR sensors and 1 button sensor.

This is achieved through the following lines:

DDRD |= 0b00011100;.//specifies port D and specific pins for input

PCICR = B00000010;//selects Port C to trigger the interrupts

PCMSK1 = B00001111;//selects pin A0,A1,A2,A3 in Port C to trigger the interrupt

These pins are used in ISR (PCINT1\_vect) function which is the vector name for Interrupt Service Routine , Pin Change Interrupt for port C.

The corresponding pins are read when a sensor state is changed through the sensors triggering.

The following line declared the variables as volatile for use by the program and can be updated within the function.

volatile byte r, g, b = 0;

these variables are where the function stores the changes in state to the pins that have been specified in PCMSK1 as follows:

g = PINC & B00000001;// Read the state of pin A0 and store it in 'g'

b = PINC & B00000010; // Read the state of pin A1 and store it in 'b'

r = (PINC & B00001100) ? HIGH : LOW; // Read the state of pin A2, A3 and store it in 'r'

updateLED(); // Update LED with new colour values

these state changes are then read an outputted through Port D digital pins 2,3,4 for the LED to change

Timer1 was configured in CTC mode to cycle through red, green, and blue outputs at a fixed interval.

This is done through the clearing the timer registers TCCR1A = 0 .

Applying the following bitwise notation TCCR1B = 0b0000110 which enables:

* WGM12 which enables CTC-clear timer on compare match- this is bit 3 in the TCCR1B register
* CS12 - bit 2 in the TCCR1B register- used for setting the timer prescaler
* CS10 - bit 0 in the register- used to configure Timer1 prescaler

OCR1A sets the value for Output Compare Register A so the timer will count from 0 to 15624, trigger the ISR and reset to 0

TIMSK1 = 0b00000010; - this bitwise notation changes bits that correspond to specific interrupt sources for timer 1- this is set to Output Compare Match A and enables interrupt when Timer1 reaches value in OCR1A

The timer is triggered when it reached the compare match value every 1 second based on OCR1A = 16000000/1024 - 1; // ≈ 15624.

When this occurs the ISR is triggered which cycles through the LED state as in the following code:

ISR(TIMER1\_COMPA\_vect) {

// Cycle: 0 = Red, 1 = Green, 2 = Blue

ledState = (ledState + 1) % 3;

r = (ledState == 0);

g = (ledState == 1);

b = (ledState == 2);

updateLED();

This uses a variable which is marked as volatile because it is a global variable that is shared between the main program and the Timer 1 ISR:

volatile byte ledState = 0;

Converting everything to bitwise notation made my ability to map the pins very easy. I used the lectures, workshops, internet sources and AI explanations to break the code down to individual bits and follow what was happening without much problem.