```c
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000UL  // Annahme von 16 MHz Takt


void ADC_Init() {
	ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);  // ADC aktivieren, Prescaler auf
128 setzen
	ADMUX = (1 << REFS0);             // Referenzspannung auf AVCC setzen

}

uint16_t ADC_Read(uint8_t channel) {
	ADMUX = (ADMUX & 0xF0) | (channel & 0x0F);            // ADC-Kanal auswählen
	ADCSRA |= (1 << ADSC);                                // Starte ADC-Konvertierung

	while (ADCSRA & (1 << ADSC))                          // Warte, bis Konvertierung abgeschlossen ist
;
	return ADC;                                           // Gib den ADC-Wert zurück

}

void PWM_Init() {
	TCCR0A = (1 << COM0A1) | (1 << WGM01) | (1 << WGM00); // Fast PWM-Modus konfigurieren
	TCCR0B = (1 << CS01);                                 // Prescaler auf 8 setzen

	DDRD |= (1 << PD6);                                   // PD6 (Pin 6) als PWM-Ausgang setzen


}

void SetPWM(uint8_t dutyCycle) {
	OCR0A = dutyCycle;                                    // PWM Duty Cycle setzen


}

int main(void) {
	ADC_Init();   // ADC initialisieren

	PWM_Init();   // PWM initialisieren

	DDRH |= (1 << PH6) | (1 << PH5);  // PH6 und PH5 als Ausgänge für LEDs setzen

	while (1) {
		uint16_t adcValue = ADC_Read(0);         // ADC-Wert vom Potentiometer lesen
		uint8_t dutyCycle = adcValue >> 2;       // Duty Cycle berechnen (Rechtsschiebeoperation um 2 Bits


		// LED an PH6 steuern (statt PB7)
		if (dutyCycle >= 64) {
			PORTH |= (1 << PH6);// Setze das Bit für den Pin PH6

			} else {
			PORTH &= ~(1 << PH6);
				// Lösche das Bit für den Pin PH6
		}
```

```
if (dutyCycle >= 40 && dutyCycle <= 50) {// LED an PH5 ein-/ausschalten, wenn Duty Cycle zwischen 40% und 50%
liegt

PORTH |= (1 << PH5);                                    // Setze das Bit für den Pin PH5
                    } else {
                    PORTH &= ~(1 << PH5);               // Lösche das Bit für den Pin PH5

            }

            SetPWM(dutyCycle);                          // PWM Duty Cycle setzen

            _delay_ms(100);                             // Kurze Pause für Stabilität

      }

      return 0;
}
```