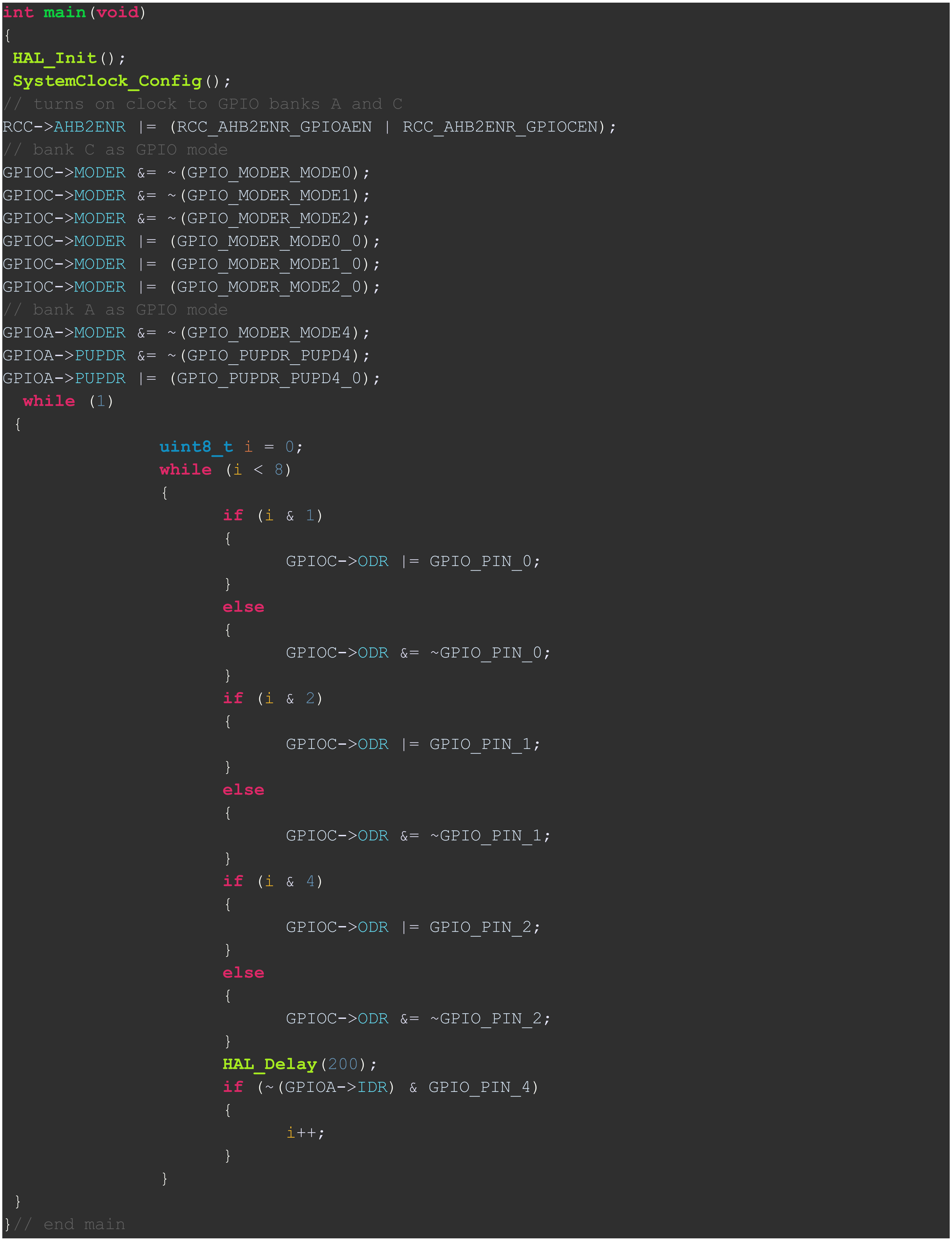

```
int main(void)
{
 HAL_Init();
 SystemClock_Config();
// turns on clock to GPIO banks A and C
RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN | RCC_AHB2ENR_GPIOCEN);
// bank C as GPIO mode
GPIOC->MODER &= ~(GPIO_MODER_MODE0);
GPIOC->MODER &= ~(GPIO_MODER_MODE1);
GPIOC->MODER &= ~(GPIO_MODER_MODE2);
GPIOC->MODER |= (GPIO_MODER_MODE0_0);
GPIOC->MODER |= (GPIO_MODER_MODE1_0);
GPIOC->MODER |= (GPIO_MODER_MODE2_0);
// bank A as GPIO mode
GPIOA->MODER &= ~(GPIO_MODER_MODE4);
GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD4);
GPIOA->PUPDR |= (GPIO_PUPDR_PUPD4_0);
  while (1)
 {
               uint8_t i = 0;
               while (i < 8)
               {
                     if (i & 1)
                     {
                           GPIOC->ODR |= GPIO_PIN_0;
                     }
                     else
                     {
                           GPIOC->ODR &= ~GPIO_PIN_0;
                     }
                     if (i & 2)
                     {
                           GPIOC->ODR |= GPIO_PIN_1;
                     }
                     else
                     {
                           GPIOC->ODR &= ~GPIO_PIN_1;
                     }
                     if (i & 4)
                     {
                           GPIOC->ODR |= GPIO_PIN_2;
                     }
                     else
                     {
                           GPIOC->ODR &= ~GPIO_PIN_2;
                     }
                     HAL_Delay(200);
                     if (~(GPIOA->IDR) & GPIO_PIN_4)
                     {
                           i++;
                     }
               }
 }
}// end main
```

Execution Timing Benchmarks

Timing Function	uint8_t	int32_t	float	double
Subroutine Call	324 ns	310 ns	486 ns	592 ns
test_var = num + 1	350 ns	323 ns	434 ns	1.90 us
test_var = num * 3	373 ns	347 ns	486 ns	2.22 us
test_var = num / 3	385 ns	373 ns	586 ns	3.05 us
test_var = sqrt(num)	NA	21.5 us	21.5 us	20.5 us
test_var = sin(num)	NA	38.4 us	38.2 us	38.2 us

For the simple operations (nop, add, multiply, divide) the majority of the time is due to the function call overhead. For more complex operations that make use of floating point numbers, execution time can vary greatly, and the function call overhead is minor.

Video Demonstration