```c
/**
* GearMotor.c
* This controls the DC gear motor that opens the sarcophagus door.
*/

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "GearMotor.h"

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"
#include "Mummy.h"

#define MOTOR_OPEN_TIME 2400 // gear motor opening run time [ms]
#define MOTOR_CLOSE_TIME 100 // gear motor closing run time [ms]

static uint8_t MyPriority; // module-level variable for the GearMotor module's priority

/* Prototypes */
bool InitializeGearMotor(uint8_t Priority);
bool PostGearMotor(ES_Event ThisEvent);
ES_Event RunGearMotor(ES_Event CurrentEvent);
static void openDir(void);
static void closeDir(void);
static void brake(void);

/**
* Initializes the GearMotor service.
* @param Priority: priority of the GearMotor service
*/
bool InitializeGearMotor(uint8_t Priority) {
  ES_Event ThisEvent;
  MyPriority = Priority;
  ThisEvent.EventType = ES_INIT;

  if (ES_PostToService(MyPriority, ThisEvent) == true) {
        return true;
  } else {
        return false;
  }
}

/**
* Runs the GearMotor service.
* @param CurrentEvent: the current event that has occurred
*/
ES_Event RunGearMotor(ES_Event CurrentEvent) {
  ES_Event ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

    switch (CurrentEvent.EventType) {
        case DoorOpen: {
            openDir();
            ES_Timer_InitTimer(GEAR_MOTOR_TIMER, MOTOR_OPEN_TIME);
        }
```

```
        break;
        case DoorClose: {
            closeDir();
            ES_Timer_InitTimer(GEAR_MOTOR_TIMER, MOTOR_CLOSE_TIME);
        }
        break;
        case ES_TIMEOUT: {
            if (CurrentEvent.EventParam == GEAR_MOTOR_TIMER){
                brake();
            }
        }
        break;
    }
    return ReturnEvent;
}

/**
* Posts events to the GearMotor service.
* @param ThisEvent: event to post to GearMotor servce
* @return true if successfully posted; false otherwise
*/
bool PostGearMotor(ES_Event ThisEvent) {
    return ES_PostToService(MyPriority, ThisEvent);
}

/**
* Sets the motor to rotate in the opening direction.
*/
static void openDir(void) {
    GMOTOR_IN_1_PORT |= GMOTOR_IN_1_PIN;
    GMOTOR_IN_2_PORT &= ~GMOTOR_IN_2_PIN;
}

/**
* Sets the motor to rotate in the closing direction.
*/
static void closeDir(void) {
    GMOTOR_IN_2_PORT |= GMOTOR_IN_2_PIN;
    GMOTOR_IN_1_PORT &= ~GMOTOR_IN_1_PIN;
}

/**
* Stops the motor.
*/
static void brake(void) {
    GMOTOR_IN_1_PORT |= GMOTOR_IN_1_PIN;
    GMOTOR_IN_2_PORT |= GMOTOR_IN_2_PIN;
}
```