```
/*******************************************************************************
 Module
   WaterService.c

 Revision
   1.0.1

 Description
   This is a file for implementing the water service in our ME 218A project.

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 11/10/19 18:08 LG      began set-up from TemplateFSM.c
*******************************************************************************/
/*----------------------------- Include Files -----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_gpio.h"
#include "inc/hw_types.h"
#include "inc/hw_pwm.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "PWM16Tiva.h"
#include "ADMulti.h"
#include "WaterService.h"
#include "LEDWriteService.h"
#include "GameService.h"

#define IR_BITS_HI (BIT0HI | BIT1HI | BIT2HI)
#define NUM_WATER_LEDS 3
#define IR_THRESHOLD 3500
#define WATER_DURATION 20000
#define BUZZ_DURATION 500

/*----------------------------- Module Defines ----------------------------*/

/*---------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
bool InitWaterService(uint8_t Priority);
bool PostWaterService(ES_Event_t ThisEvent);
bool Check4IRRise(void);
ES_Event_t RunWaterService(ES_Event_t ThisEvent);
void ON_OFF_VIBRATION (uint8_t Value);
void ADC_Convert (void);
void SetRandom (void);
void TurnOffWaterLEDs(void);


/*---------------------------- Module Variables ---------------------------*/
static uint8_t MyPriority;
```

```
static uint32_t triadeStatus[4]; // this is the array that stores the analog
results that we read
static WaterService_t CurrentState = W_Inactivity; // this is the state that
switches in our state machine
static uint32_t IRLastState = 0; // our last state will be used to detect a
rising edge
static uint8_t SetValue = BIT0HI;
static uint8_t FlashCount = 0;

/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitWaterService

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Saves away the priority
 Notes

 Author
     Lisa Gardner
****************************************************************************/
bool InitWaterService(uint8_t Priority)
{
  ES_Event_t ThisEvent;
  MyPriority = Priority;
  CurrentState = W_Inactivity;

  //static uint8_t ADC_NumPins = 4;
  //ADC_MultiInit(ADC_NumPins);

  return true;
}

/****************************************************************************
 Function
     PostWaterService

 Parameters
     EF_Event ThisEvent ,the event to post to the queue

 Returns
     bool false if the Enqueue operation failed, true otherwise

 Description
     Posts an event to this state machine's queue
 Notes

 Author
     J. Edward Carryer, 10/23/11, 19:25
****************************************************************************/
bool PostWaterService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    RunWaterService
```

```
 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here
 Notes

 Author
   Lisa Gardner
****************************************************************************/
ES_Event_t RunWaterService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

  switch(CurrentState) {

    case W_Inactivity: {
      printf("W_Inactivity \r\n");

      if (ThisEvent.EventType == WATER_START) {
        printf("started timer \r\n");
        ES_Timer_InitTimer(WATER_TIMER, WATER_DURATION); //needs to last for 20
seconds

        // Turn on random water LED to start the game
        SetRandom();
        CurrentState = Wait4IR;
      }
    }
    break;

    case Wait4IR:
    {
      printf("Wait4IR\r\n");

      if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
WATER_TIMER)) {

        ES_Event_t Event2Post;
        Event2Post.EventType = WATER_CMD;
        Event2Post.EventParam = 0;
        PostLEDWriteService(Event2Post);

        printf("Water Over \r\n");
        PostGameService(ThisEvent);
        TurnOffWaterLEDs();
        CurrentState = W_Inactivity;
      }

      else if(ThisEvent.EventType == IR_RISE) {
        //printf("IR RISE\r\n");


        if (ThisEvent.EventParam == SetValue) {

          // Get correct value
          // Turn off all water LEDs
          ThisEvent.EventType = WATER_CMD;
          ThisEvent.EventParam = 0;
```

```
          PostLEDWriteService(ThisEvent);
          AddHealth();
          //printf("Add Health \r\n");

          // Tactile feedback: Start buzzing
          ES_Timer_InitTimer(BUZZ_TIMER, BUZZ_DURATION);
          ON_OFF_VIBRATION(BIT3HI);
          //printf("Buzzing \r\n");

          CurrentState = Buzzing;
        }

      // Wrong IR rise for water LED
      else {

        // Decrease health
        SubtractHealth();

        ThisEvent.EventType = WATER_CMD;
        ThisEvent.EventParam = 0;
        PostLEDWriteService(ThisEvent);

        ES_Timer_InitTimer(WATER_FLASH_TIMER, BUZZ_DURATION);

        //printf("Subtract health \r\n");

        // Go into this state to flash correct LED
        CurrentState = FlashWait;
      }
    }
  }
    break;

    case FlashWait: {
      printf("FlashWait\r\n");

    if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
WATER_FLASH_TIMER)) {

      if (FlashCount == 0) {
        // Turn on correct water LED
        ThisEvent.EventType = WATER_CMD;
        ThisEvent.EventParam = BIT0HI | BIT1HI | BIT2HI;
        PostLEDWriteService(ThisEvent);

        // Increment FlashCount to go into led off
        FlashCount = 1;
        // Start flash timer again
        ES_Timer_InitTimer(WATER_FLASH_TIMER, BUZZ_DURATION);
      }

      else if (FlashCount == 1) {
        // turn off correct water LED
        ThisEvent.EventType = WATER_CMD;
        ThisEvent.EventParam = 0;
        PostLEDWriteService(ThisEvent);

        // Increment FlashCount so that random water LED can be turned on
        FlashCount = 2;
        ES_Timer_InitTimer(WATER_FLASH_TIMER, BUZZ_DURATION);
      }

      else {
        // Set FlashCount back to 0 and light up random water LED
```

```
        FlashCount = 0;
        SetRandom();
        CurrentState = Wait4IR;
      }
    }

    if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
WATER_TIMER)) {
      printf("timeout of entire service from buzzing \r\n");
      TurnOffWaterLEDs();
      PostGameService(ThisEvent);
      CurrentState = W_Inactivity;
    }
    }
    break;

    case Buzzing: {

      if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
WATER_TIMER)) {
        TurnOffWaterLEDs();
        ON_OFF_VIBRATION(BIT3LO);
        PostGameService(ThisEvent);
        CurrentState = W_Inactivity;
      }

      else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
BUZZ_TIMER)) {
        // Turn off vibration motor and light up random water LED
        ON_OFF_VIBRATION(BIT3LO);
        SetRandom();
        CurrentState = Wait4IR;
      }
    }
    break;
}
  return ReturnEvent;
}



bool Check4IRRise(void) {
  ES_Event_t ThisEvent;
  ADC_MultiRead(triadeStatus);
  uint32_t IRCurrentState = 0;

  ADC_Convert();

  for (int i = 1; i < (NUM_WATER_LEDS + 1); i++) {
    IRCurrentState |= triadeStatus[i];
  }
  //printf("curr state: %d\r\n", IRCurrentState);

  if ((IRLastState & IR_BITS_HI) != (IRCurrentState & IR_BITS_HI)) {
    //printf("edge detected\r\n");
    if ((IRCurrentState & BIT0HI) != 0) {
      ThisEvent.EventType = IR_RISE;
      ThisEvent.EventParam = BIT0HI;
      printf("IR 0 HI\r\n");
    }

    else if ((IRCurrentState & BIT1HI) != 0) {
      ThisEvent.EventType = IR_RISE;
      ThisEvent.EventParam = BIT1HI;
```

```
      printf("IR 1 HI\r\n");
    }

    else if ((IRCurrentState & BIT2HI) != 0) {
      ThisEvent.EventType = IR_RISE;
      ThisEvent.EventParam = BIT2HI;
      printf("IR 2 HI\r\n");
    }

    IRLastState = IRCurrentState;
    PostWaterService(ThisEvent);
    return true;
  }

  IRLastState = IRCurrentState;
  return false;
}

void ADC_Convert () {
  uint32_t TempSetValue = BIT0HI;

  for (int i = 1; i < 4; i++) {

    if(triadeStatus[i] >= IR_THRESHOLD) {
      triadeStatus[i] = TempSetValue<<(i-1);
    }
    else {
      triadeStatus[i] = 0;
    }
  }

  return;
}

void ON_OFF_VIBRATION (uint8_t Value) {

  if (Value == BIT3HI) {
    HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA+ALL_BITS)) |= Value;
  } else {
    HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA+ALL_BITS)) &= Value;
  }

  return;
}

void SetRandom(void) {
  ES_Event_t ThisEvent;
  srand(ES_Timer_GetTime());

  uint8_t x = rand() % NUM_WATER_LEDS;

  SetValue = BIT0HI << x;

  ThisEvent.EventType = WATER_CMD;
  ThisEvent.EventParam = SetValue;
  PostLEDWriteService(ThisEvent);
  printf("Random WLED: %u\r\n", SetValue);
}

void TurnOffWaterLEDs(void) {
  ES_Event_t Event2Post;

  Event2Post.EventType = WATER_CMD;
  Event2Post.EventParam = 0;
```

```
  PostLEDWriteService(Event2Post);
}


/*****************************************************************************
 private functions
 *****************************************************************************/

/*------------------------------ Footnotes -------------------------------*/
/*------------------------------ End of file -----------------------------*/
```