Assignment 4 – RTOS
CSCE 4114 – Embedded Systems – Fall 2018
Due Date – Thursday, December 6th, 11:59 PM CT

Description: This assignment is designed to give you experience with implementing a basic RTOS on a microcontroller platform. The program will implement a basic home security system. The security system will have a single sensor (in this case, implemented as a light sensor), a single actuator (a buzzer that plays as an alarm), and a console (implemented using serial over UART insteald of a keypad). These tasks all need to run concurrently. There will be some thought that must go into making sure that these tasks can co-occur without causing an issue with the timing of the other tasks.
Expected Operation: The following will define the behavior of the program you will develop:

1. You will define a Task Control Block that holds four separate tasks to be implemented in a round-robin scheduling.

2. The four tasks will be sensing a light sensor, playing a tone, reading from the serial port, and checking the input from the serial port. Each of these tasks will be implemented as a function with a single parameter.

    - The light sensor will have a parameter that is a structure that holds 1) a threshold, 2) an integer return value from the ADC, and 3)a binary return value if the ADC breaks the threshold. The threshold will be compared to the value from the ADC. If the ADC value is greater, then the binary value will hold true, otherwise it will hold false. The binary value will be used to tell the alarm what to do.
    - The alarm will take have a parameter that is a structure that holds the previous state of the alarm and the threshold value from the light sensor. The state will be composed of three parts: 1) How many microseconds until the pin needs to be changed from high to low or low to high, 2) How many iterations of the current note are left, and 3) What is the current note.
    - The serial read function will take in a parameter that holds three values: 1) A pointer to the read buffer, 2) the current index to read into, and 3) a return value for if the most recent value was a newline character (‘\n’). If there are any bytes available on the serial port, the will be read into the buffer.
    - The check serial input function will only occur when there is a full buffer or a newline character (as noted by the return from task 3). The check serial function is only concerned with two types of input: 1) The value as returned from the main menu selection, and 2) a keycode to disarm the security system. The task will validate the input is an integer between 0 and 9999, and return the input as an integer through its data structure. An invalid input or no input will return a -1. An invalid input will also create a message on the serial prompt saying “Improper value! Try again”.

3. To implement these functions through a TCB, you will need to create four different data structures to hold these objects. The functions themselves will have as their prototype a void pointer that will be cast as a pointer of the type of the data structure to be used.

4. The TCB will be implemented as an array of four function pointers with a single void pointer argument. The TCB will have the light sensor function in position 0, the alarm function in position 1, and the read serial in position 3.

5. On boot up, the program will display a menu through the serial port that looks like this (from setup function):

```
==========Main Menu==========
1) Arm Security System
2) Test Light Sensor
3) Test Alarm
4) Set Light Sensor Threshold
5) Set Keycode
============================
```

6. The program will then loop through the TCB one function at a time. If the security system is not yet armed, the first two functions should be skipped.

7. When the read serial detects a newline character, the main program will determine if it is a valid input. Values 1-4 will perform the following actions:

8. Arming the security system will start a loop that will read the light sensor (Connected to A0), and if the value is above the threshold, will start an alarm that oscillates between A(440 Hz) and D(587.33Hz) every quarter second until the light drops back below the threshold.

9. Arming the system will first print a dialog that says "Security System Armed! Disarm by entering keycode!". When the keycode is entered into the serial console with a newline (e.g. 0000\n), the system stops sampling and playing the alarm.

10. Testing the light sensor should print the ADC value from the light sensor to the serial console, newline separated, for 100 samples (this is to help set the calibration threshold)

11. Testing the alarm should play the alarm (oscillating between A and D every quarter second) for 3 seconds.

12. Navigating to set the sensor threshold should create a new prompt that looks like this:

```
==========
Please provide a value between 0 and 1023 for the new threshold:
```

    This can be a blocking wait on the new threshold value, which will replace the default value of 512.

13. Navigating to set the Keycode should create a new prompt that looks like this:

    ```
    ==========
    Please provide a new 4-digit keycode:
    ```

    This can be a blocking wait on the new keycode, which will replace the default value of 0000.

14. The serial validation task will perform as specified above. The RTOS will check the return value. If it is a -1, then the RTOS will not do anything. A return value will be checked to see if it is a main menu selection, or a disarm keycode, and will perform the logic to select one of the tasks, disarm the system, or prompt with "improper selection/keycode" if the input does not make sense with the current state.

Useful Hints:

- The 'D' tone's half-period is $\approx 851\mu S$. The clock speed of your Arduino is $\approx 48MHz$. This means you maximally have 42k clock cycles per loop through the TCB. Any blocking waiting may cause the system to be unable to perform all tasks in a timely fashion.
- We are not using any global variables between the different tasks. The RTOS is managing message passing by manipulating the variables associated with each of the tasks.
- Unit tests and debug messages are great. "#ifdef DEBUG" can be used with a "#define DEBUG" to print debug messages to the serial, and disabled by commenting out the definition.
- START EARLY! Its a hardware project. Murphy's law will rear its head if you wait until the last day.

Rubric: The project will be graded according to the following rubric:

These documents should be uploaded through Blackboard to the TA before the beginning of the next lab. Documents turned in after this deadline are subject to the course late policy.

Category	Description	Percentage
Implement Behavior	The behavior of the system should be implemented as defined by the expected behavior. Each of the 5 behaviors needed to operate correctly.	50%
State Machine/Functional Diagram	Make a diagram of the state machine(s) that compose the system. The state machine should clearly note the states, transitions, and actions of each state. Shared variables between state machines should be shown, with what SM controls the variable, and what SMs read from the variable.	20%
Coding Comments & Style	Use appropriate coding styles. Comment functions with a description about their behavior, any parameters, any return values, and any shared variables which it manipulates.	15%
Report	A simple, one- or two-page report. The report should have the project name (e.g. Binary Mastermind), a short description of what you did, and the outcomes (e.g. Did it pass all example tests, if not, why not, etc...).	15%

Table 1: Grading Rubric