CME332 Lab 02

Name: Alvi Akbar

Student Number: 11118886

NSID: ala273

Requirements Analysis:

Steps:

Unlock the door:

- Door in CLOSED and locked state
- Enter a valid access code
- Press Lock Key Once

Add New Access Code:

- Door in Open State
- Press the lock key once
- Enter new access code
- Press the Lock key again

Delete Access Code:

- Door in Open State
- Press Lock Key Once
- Enter Existing Access Code
- Enter Lock Key again
- Re-Enter same Access Code
- Press Lock key again

Default Unlock Code:

The default unlock code for this lock is 0101. It supports all feature such as adding and deleting code, however, the max storage size for this design is 4 digits.

We can store upto 16 codes.

Components

DE2 Board components	**Representation**
SW[0] - ON	Door OPEN State
SW[0] - OFF	Door CLOSE State
KEY 1	Lock Key

LED G0	Go to LOCK State
LED R0	Go to OPEN State
LEDG [1-3] Flashes	Successful Unlock/Add/Delete Operation
LEDR [1-3] Flashes	Unsuccessful Unlock/Add/Delete Operation

Tasks Analysis:

Tasks	**Description**
void Task_read_PS2	Read numbers from Keyboard **Trigger:** ● As soon as current state is LOCK state ● If current state is OPEN && KEY1 is pressed **Type:** ● **I/O bound**
void Task_read_KEYS	Reads KEY1 value from Keyboard **Trigger**: ● All States except INIT and VERIFIED **Type:** ● **I/O bound**
void Task_add_access_code	● Adds Access Code if access code is not already present. ● Signals Task_flash_success or Task_flash_fail based on the outcome. **TYPE:** ● **CPU bound**
Task_delete_access_code	Deletes Access Code if code matches existing code. **TYPE:** ● **CPU bound**

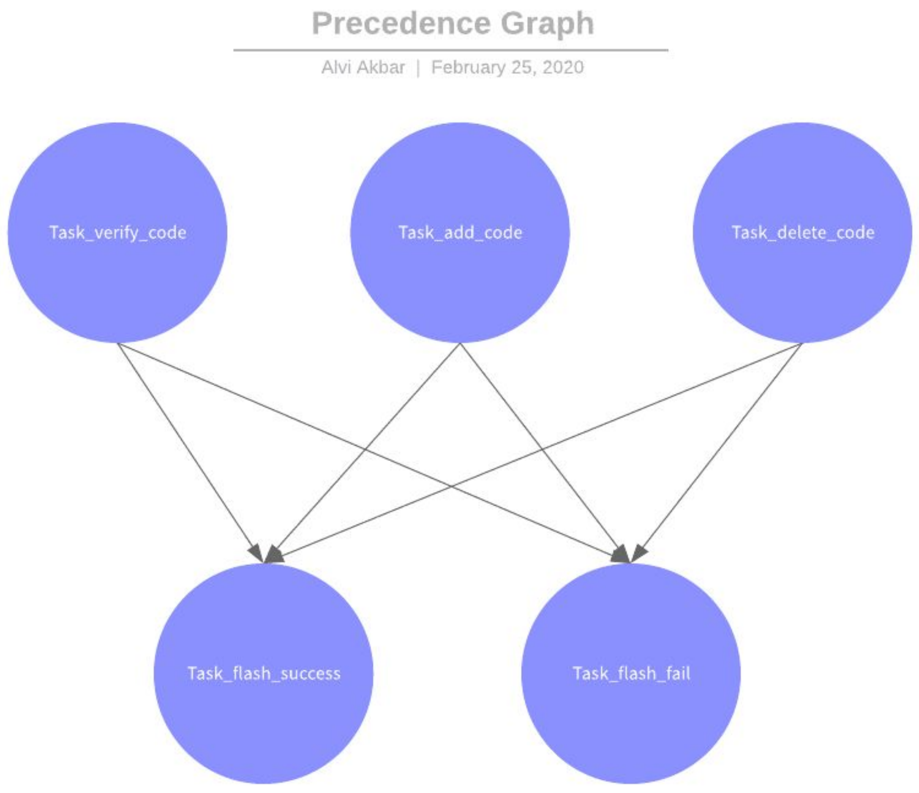
Task_verify_code	Verifies Access Code. **TYPE:** • **CPU bound**
Task_flash_success	LEDG [1-3] Flashes **Type:** • **CPU bound**
Task_flash_fail	LEDR [1-3] Flashes **Type:** • **CPU bound**
Void Task_state_timer	Records current state time **Type:** • **I/O bound**

Since most of the state transitions depend on **KEY 1** Press, therefore, we are considering Task1 to be most important out of all and hence, this is a task with higher priority.

Priorities:
Tasks priorities have been assigned according to **Rate Monotonic Algorithm** i.e Tasks with **shorter period** has **highest priority**. **Task_read_KEYS** has the shortest period as it runs after every **100 ms** and hence it has the highest priority.

Note: Tasks with **higher priorities** will have **less jitters.**
Tasks with **lower priorities** will **have more jitters.**

Task Dependency Diagram:



- Semaphores are used to achieve **Activity synchronization** or Sequence Control as demonstrated in Precedence Graph above.

- **Rendezvous Synchronization** is used to record the combination of Number key inputs from PS2 Keyboard and time delay sequence during **PROG** or **CODE** State (between Task_read_PS2 and Task_state_timer)

Semaphore Description:

Following semaphores were used to achieve activity/sequence control between tasks.
OS_EVENT *SEM_read_PS2;
OS_EVENT *SEM_read_PS2_done;
OS_EVENT *SEM_read_KEYS;
OS_EVENT *SEM_timer_start;
OS_EVENT *SEM_flash_success;
OS_EVENT *SEM_flash_fail;
OS_EVENT *SEM_add_code;

Following semaphores were used to avoid race conditions.
OS_EVENT *SEM_state_change;
OS_EVENT *SEM_timer_code

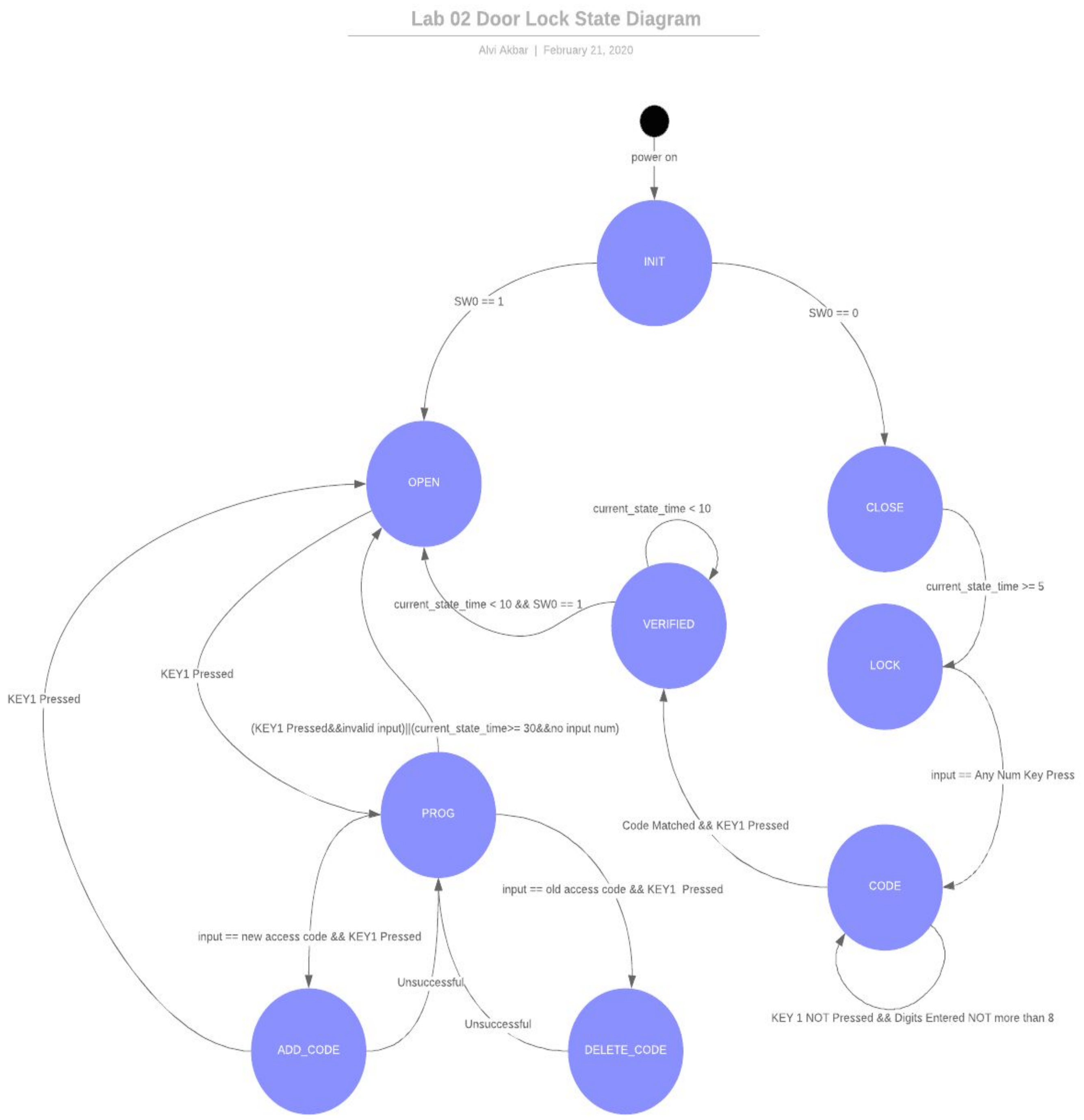
We assigned initial value of SEM_flash_success and SEM_flash_fail to 0 so that it pends and wait first and we signal and use those tasks when required.

Calculations

In order to meet the deadline for all tasks, we have to verify:
Worst Case Execution Time < Deadline == Period

Except the timer detection task which has a period of 1 second, all other periodic tasks repeat itself after 10 ms.

State Diagram:



Lab 02 Door Lock State Diagram

Alvi Akbar | February 21, 2020

power on
INIT
SW0 == 1
SW0 == 0
OPEN
CLOSE
current_state_time < 10
current_state_time >= 5
current_state_time < 10 && SW0 == 1
VERIFIED
LOCK
KEY1 Pressed
KEY1 Pressed
(KEY1 Pressed&&invalid input)||(current_state_time>= 30&&no input num)
input == Any Num Key Press
PROG
Code Matched && KEY1 Pressed
CODE
input == old access code && KEY1 Pressed
input == new access code && KEY1 Pressed
Unsuccessful
Unsuccessful
KEY 1 NOT Pressed && Digits Entered NOT more than 8
ADD_CODE
DELETE_CODE