Lab 3 Notes

Task 1: Motion-triggered lamp (4 points).

Consider a motion-triggered lamp system with a motion sensor connected to A0. The system defines motion as A0=1 for two consecutive 200ms samples. When motion is detected, the system should illuminate a lamp (by setting B1 to 1), keeping the lamp on for 10 seconds past the last detected motion. The system should also blink a small LED (connected to B0) for 200ms on and off while motion is detected. A simpler capture approach uses three tasks as shown below.

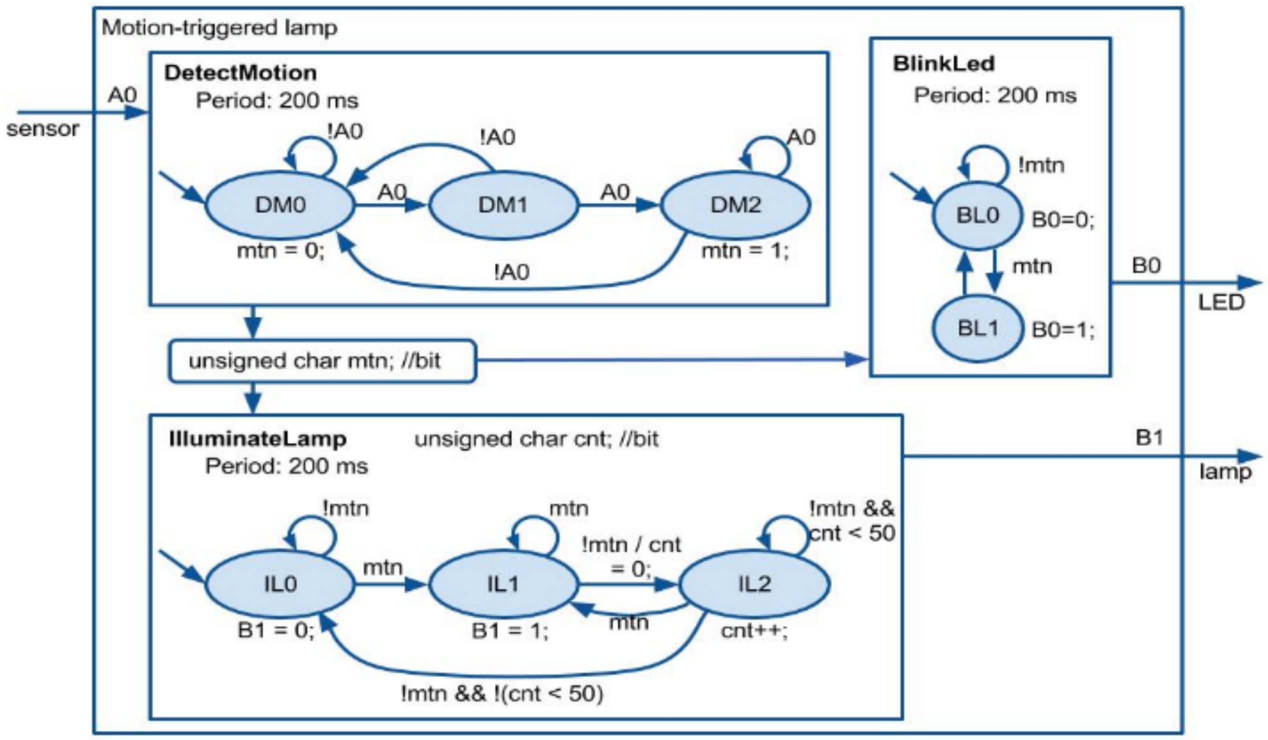


Fig. 1 pseudo-coded tasks

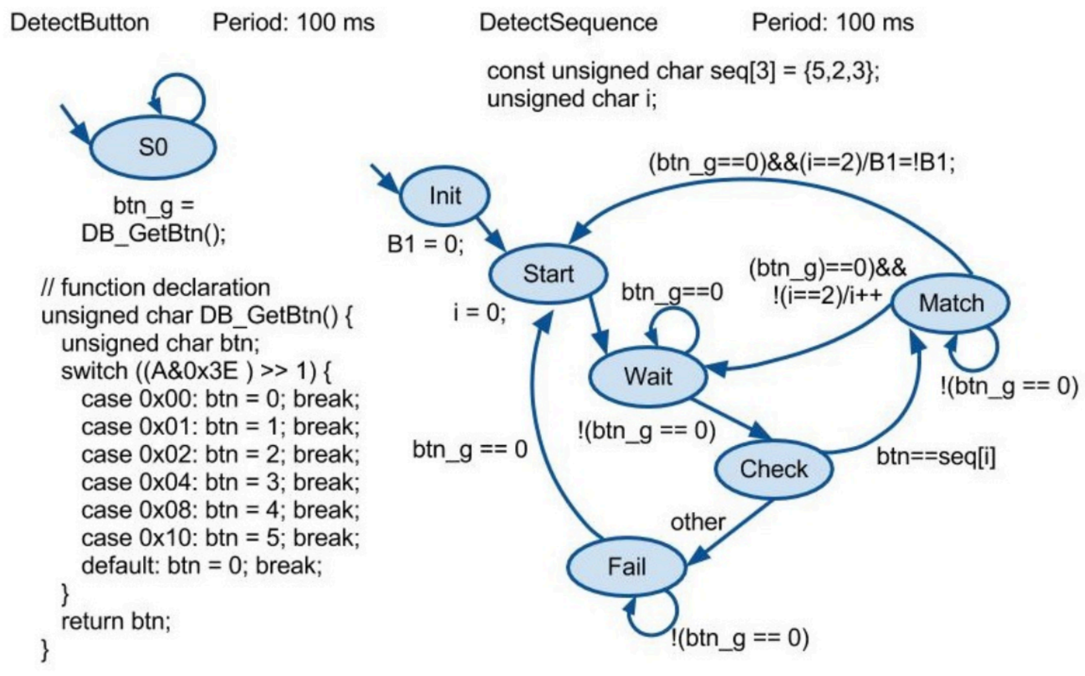
Tasks:

1.1 Implement the above system using RIBS (2 points).

1.2 Extend the above system by: having 3 buttons to connect to A1 A2 A3 to control the duration of keeping the lamp on past the last detected motion(while button is pressed). 5seconds, 10 seconds and 15 seconds for A1 A2 and A3 accordingly (The default duration is 3 seconds, ie., no button is pressed) (2 points)

Task 2: Door-lock system (6 points).

Consider a door-lock system with five input buttons numbered 5, 4, 3, 2, 1, connected to A5, A4, A3, A2, A1. Task DetectButton detects which button is pressed. The task samples with a period of 100 ms and writes the currently pressed button to global variable unsigned char btn_g, writing 0 when no button is pressed or if multiple buttons are pressed, and writing either 5, 4, 3, 2, or 1 when a single button is pressed. The task is captured as a one-state synchSM, and happens to illustrate use of a synchSM that calls a function. Another task DetectSequence samples btn_g every 100 ms and toggles the door lock (B0=1 locks the door) whenever the task observes a particular three-button sequence. The task is captured using a multiple-state synchSM, which happens to have the correct button sequence stored in an array constant. The reference SMs are as below.



```
btn_g =
DB_GetBtn();
```

```
const unsigned char seq[3] = {5,2,3};
unsigned char i;
```

```
// function declaration
unsigned char DB_GetBtn() {
  unsigned char btn;
  switch ((A&0x3E ) >> 1) {
    case 0x00: btn = 0; break;
    case 0x01: btn = 1; break;
    case 0x02: btn = 2; break;
    case 0x04: btn = 3; break;
    case 0x08: btn = 4; break;
    case 0x10: btn = 5; break;
    default: btn = 0; break;
  }
  return btn;
}
```

Fig. 2 pseudo-coded tasks for Door-lock system

Tasks:

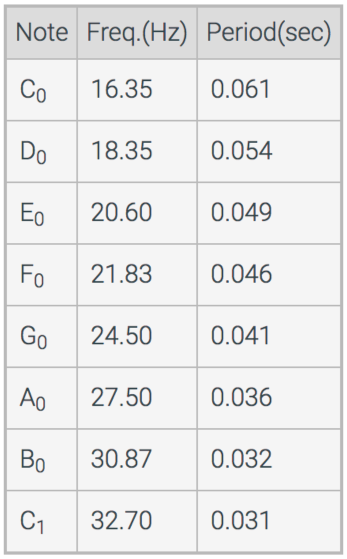
2.1 Implement the above system using RIBS (2 points).

2.2 Extend the above system so that the digital key is **(3 2 1 5 4)**. (2 points)

2.3 Extend the above system so that the LED (connected to B2) will be on for 5 seconds when the lock is open/toggled (2 points)

Task 3: Simplified digital piano (5 points).

A digital piano uses a PWM is to generate a tone on a speaker. A square wave can be created (50% duty cycle H = L) with a period selected for the desired tone frequency. Consider generating music notes. A musical note can be generated using a signal of a particular frequency. Frequencies for some notes are shown in the table below. These are low notes, corresponding perhaps to the leftmost keys on a piano (for reference, "middle C" is C4 which is 261.63 Hz).



Note	Freq.(Hz)	Period(sec)
C_0	16.35	0.061
D_0	18.35	0.054
E_0	20.60	0.049
F_0	21.83	0.046
G_0	24.50	0.041
A_0	27.50	0.036
B_0	30.87	0.032
C_1	32.70	0.031

Figure 1: Musical Notes and Freqs.

The required intervals are half of the notes' periods. The greatest common denominator of the required intervals is 20 ms. Suppose each switch A0, A1, ... A7 corresponds to notes C0, D0, ... C1, with higher inputs having priority over lower inputs. A synchSM can be constructed with a period of 20 ms that spends X ticks in a state that outputs 1 and X ticks in a state that outputs 0. Before those two states, another state can set X to the interval required by the input switch. (Note: RIMS does not presently support periods less than 20 ms). Tones generated by square waves sound rough (sine waves sound smoother).

Tips: Please use 2 SynchSMs, one for reading the Keys, and the other for generating the tones, and use global variable to share data between them. Since RIMS cannot simulate sound output, we assumed X = 20-key. H=L=X/2. Key is A0…A7 value 1-8.

Tasks:

2.1 Design and Implement the above digital piano system in RIBS (4 points)

2.2 Display screenshot of RIMS when A = 32, print H value.(1 point)