Info: Embedded systems commonly have time-ordered behavior (more so than in desktop systems). C was not originally intended for time-ordered behavior. Trying to code time-ordered behavior directly with C’s sequential statement computation model results in countless variations of "spaghetti" code. Instead, a disciplined programming approach captures behavior using a state machine computation model.

We will start with a few State Machine Design examples. Prepare a few LED’s and buttons for use with these exercises

Part 1 – SMs Design

Video Demonstration: https://youtu.be/rSnSwGzu7mU

1. Pin 13 and Pin 12 each connect to an LED. Pin 13’s LED is initially on, Pin 12 is initially off. Pressing a button connected to A0 turns off Pin 13's LED and turns on Pin 12's LED, staying that way after button release. Pressing the button again turns off Pin 12’s LED and turns on Pin 13’s LED.

2. A household has a digital combination deadbolt lock system on the doorway. The system has buttons on a keypad. Button 'X' connects to A0, 'Y' to A1, and '#' to A2. Pressing and releasing '#' then pressing 'Y', should unlock the door by setting Pin 13 to 1. Any other sequence fails to unlock. Pressing a button from inside the house (A5) locks the door (Pin 13=0). Be sure to check that only one button is pressed at a time.
Challenge Problems

3. Extend the above door so that it can also be *locked* by entering the earlier code.

4. Extend the above door to require the 4-button sequence #\text{-}X\text{-}Y\text{-}X rather than the earlier 2-button sequence. To avoid excessive states, store the correct button sequence in an array, and use a looping SM.

Part 2 – SynchSM Design

The Intel Galileo has built-in timers with numerous low-level configurable options. Luckily, the open-source community provides us with a Timer library that allows us to use functions provided instead of digging deep into low-level configurations. The Timer1 library has two functions that are essential in order for an Interrupt Service Routine on the Galileo to behave properly. These functions are:

- void Timer1.initialize(M) -- set the timer to tick every M microseconds
- void Timer1.attachInterrupt(function, period) – Calls a function at a specified interval in microseconds

Refer to the Timer1 Documentation for supplemental functions: 
http://playground.arduino.cc/Code/Timer1

The Timer1 library is already included in your Arduino IDE as long as you have installed the Intel Galileo (x86 architectures) as a board into your IDE.
Prepare 3 LED’s for the following examples and exercises. The code below toggles Pins 13 - Pin 11 every 1 second. Copy, Paste and Upload the sketch to the Intel Galileo. All LED’s should now blink on 1 sec and off 1 sec.

```cpp
#include "TimerOne.h"

boolean blinkInterval = false;
const int ledPinOne = 13;
const int ledPinTwo = 12;
const int ledPinThree = 11;

void setup() {
    pinMode(ledPinOne, OUTPUT);
    pinMode(ledPinTwo, OUTPUT);
    pinMode(ledPinThree, OUTPUT);

    Timer1.initialize(10000); // the timer period is 100000 useconds, that is 0.1 sec
    Timer1.attachInterrupt(callback, 1000000); // the callback will be called on each 10th timer interrupt, i.e. every second
}

void callback() {
    if (blinkInterval) {
        digitalWrite(ledPinOne, HIGH);
        digitalWrite(ledPinTwo, HIGH);
        digitalWrite(ledPinThree, HIGH);
    } else {
        digitalWrite(ledPinOne, LOW);
        digitalWrite(ledPinTwo, LOW);
        digitalWrite(ledPinThree, LOW);
    }
    blinkInterval = !blinkInterval;
}

void loop() {
}
```
Exercise

Using the Timer example from above, expand your code to perform the following exercises.

Video Demonstration: https://youtu.be/rSnSwGzu7mU

1. Create a synchSM to blink three LEDs connected to Pin 13, Pin 12, and Pin 11 in sequence, 1 second each.

Challenge Problem

2. Create a simple light game that requires pressing a button on A0 while the middle of three LEDs on Pin 13, Pin 12, and Pin 11 is lit. The LEDs light for 300 ms each in sequence. When the button is pressed, the currently lit LED stays lit. Pressing the button again restarts the game.