



01076101

วิศวกรรมคอมพิวเตอร์เบื้องต้น

Introduction to Computer Engineering

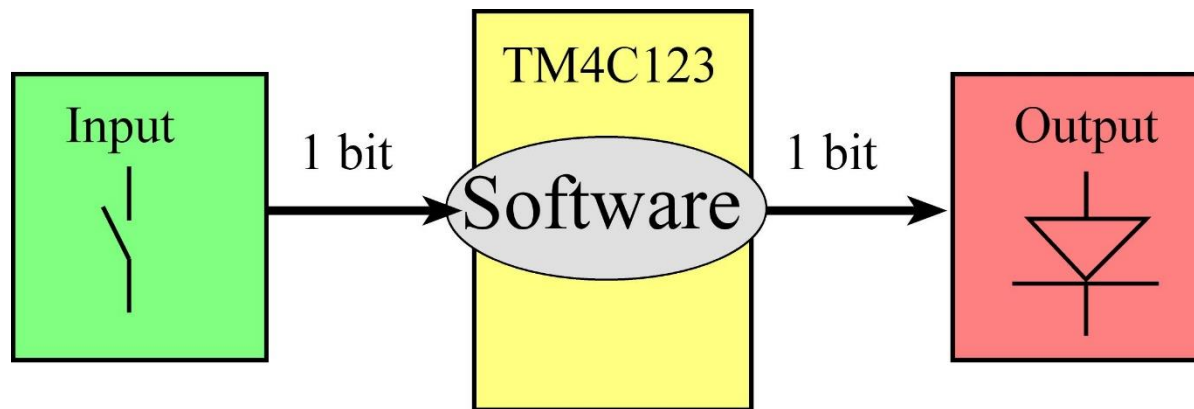
Arduino #5

Finite State Machine



# FSM : Finite State Machine

- งานบางประเภท เราสามารถใช้วิธีการบางอย่างมาใช้เขียนโปรแกรม เพื่อให้งานง่ายขึ้น
- ตัวอย่าง :** กำหนดให้ระบบหนึ่งมี Input 1 บิต และ Output 1 บิต กำหนดให้ระบบนี้อ่านข้อมูลทุกวินาที (หมายถึง 1 วินาทีอ่านข้อมูล 1 ครั้ง) จากนั้นจะนำข้อมูลไปบวกสะสม โดยหากข้อมูลในระบบเป็นเลขคี่ Output จะมีค่าเป็น 1 และหากข้อมูลในระบบเป็นเลขคู่ Output จะมีค่าเป็น 0 โดยแสดงผลออกทาง LED



# FSM : Finite State Machine

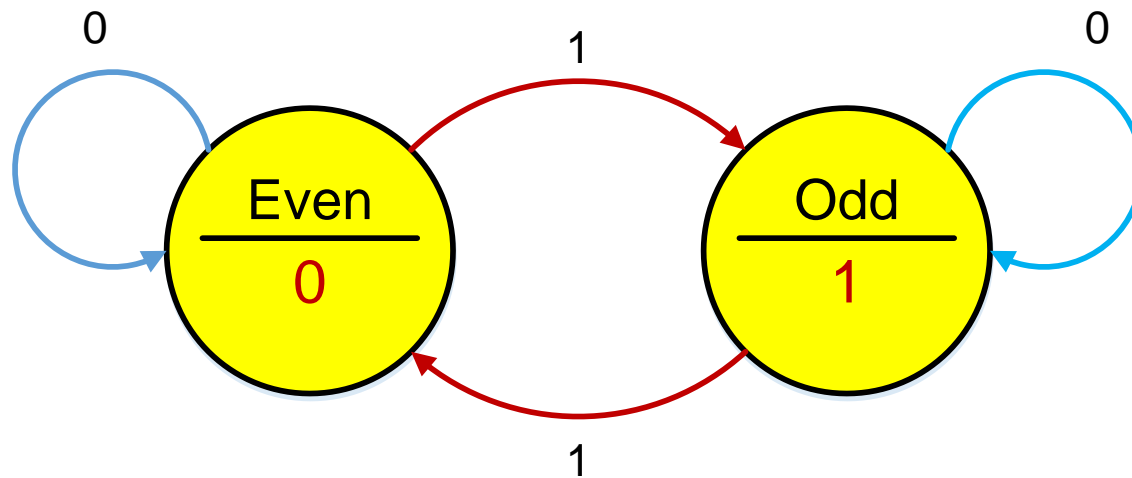


- เป็นวิธีการหนึ่งที่มีการนำไปใช้มากในเรื่องต่างๆ โดยเฉพาะในงานที่สามารถกำหนดเป็น “สถานะ” (state) ต่างๆ ได้
- หลักการพื้นฐานของ FSM คือ การแยกสิ่งที่จะทำ (policies) ออกจากกลไกการทำงาน (mechanisms) ซึ่งจะเป็นผลให้การปรับปรุงหรือปรับเปลี่ยนการทำงานสามารถทำได้อย่างสะดวกมากขึ้น
- องค์ประกอบของ FSM ประกอบด้วย Input, Output, State และ State Transition
- การทำงานของ FSM โดยย่อ คือ ระบบจะเปลี่ยน state ไปตาม Input ที่เข้ามาเพื่อสร้าง output ไปตามที่ต้องการ



# FSM : Finite State Machine

- จากตัวอย่าง สถานะ (state) คือ สถานะที่ Output เป็น 1 และ สถานะที่ Output เป็น 0 (มีทั้งหมด 2 สถานะ)
- สิ่งที่ต้องการคือ เมื่อนับได้เลขคี่ ให้ Output เป็น 1 หากนับได้เลขคู่ ให้ Output เป็น 0
- กลไกการทำงาน คือ การนับและให้ Output
- หาก state ปัจจุบันเป็น เลขคู่ ถ้า Input เป็น 0 จะอยู่ state เดิม แต่ถ้าเป็น 1 จะเปลี่ยน state เป็นเลขคี่ (output จะเปลี่ยนตาม state)



# FSM : Finite State Machine



- 5 ส่วนประกอบที่สำคัญของ FSM
  1. **A finite set of states** คือ จะต้องสามารถระบุจำนวน state ในระบบที่แน่นอนได้ โดยหนึ่งใน state เหล่านั้นจะเป็น Initial State
  2. **A finite set of external inputs** คือ จะต้องสามารถระบุจำนวน Input ที่แน่นอน
  3. **A finite set of external outputs** คือ จะต้องสามารถระบุจำนวน Output ที่แน่นอน
  4. เงื่อนไขที่แน่นอนของการเปลี่ยน state ได้แก่ เงื่อนไข input ของการเปลี่ยน state และจะเปลี่ยนไป state ไต เมื่อมี Input แบบใด
  5. ข้อกำหนดของ output ที่ state นั้นจะส่งออกมา



# FSM : Finite State Machine

- องค์ประกอบของ FSM สามารถแสดงโดย State Transition Graph ตามรูป

- Name เป็นชื่อของ state
- Output เป็นค่าของข้อมูล Output ที่ส่งออก ณ State นั้น

$$\text{Output} = g(\text{CurrentState})$$

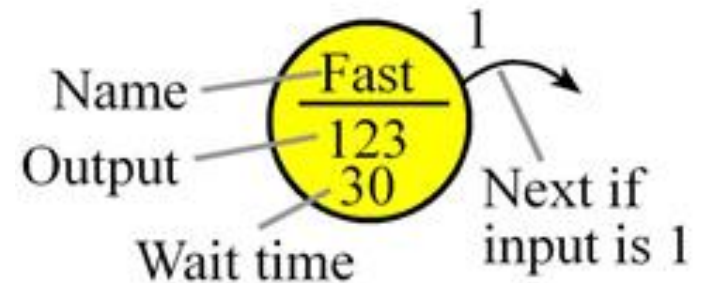
- Wait time เวลาที่ delay ใน State นั้น

- Next State บอกถึง State ถัดไป

ซึ่งจะเปลี่ยน State ตาม Input ที่เข้ามา

(ดังนั้น Next State สามารถมีได้หลายเส้นทาง)

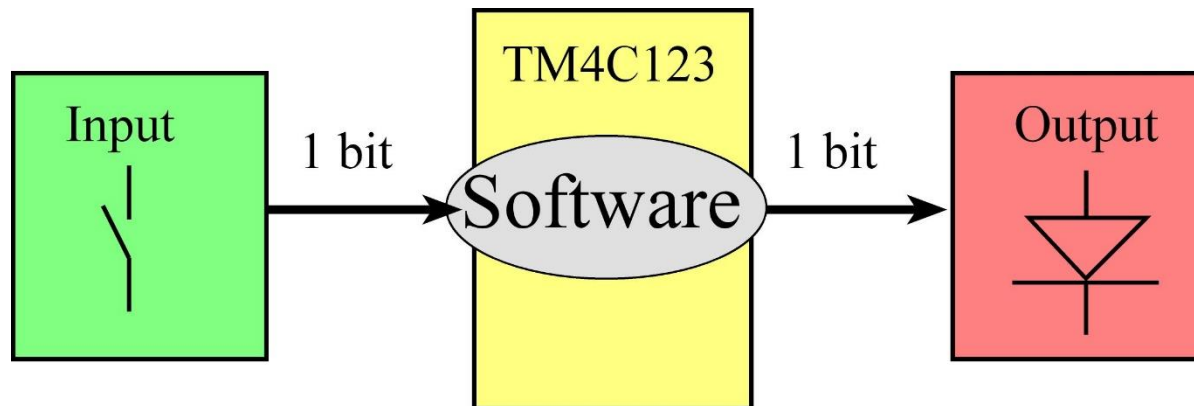
$$\text{NextState} = f(\text{Input}, \text{CurrentState})$$





# FSM : Finite State Machine

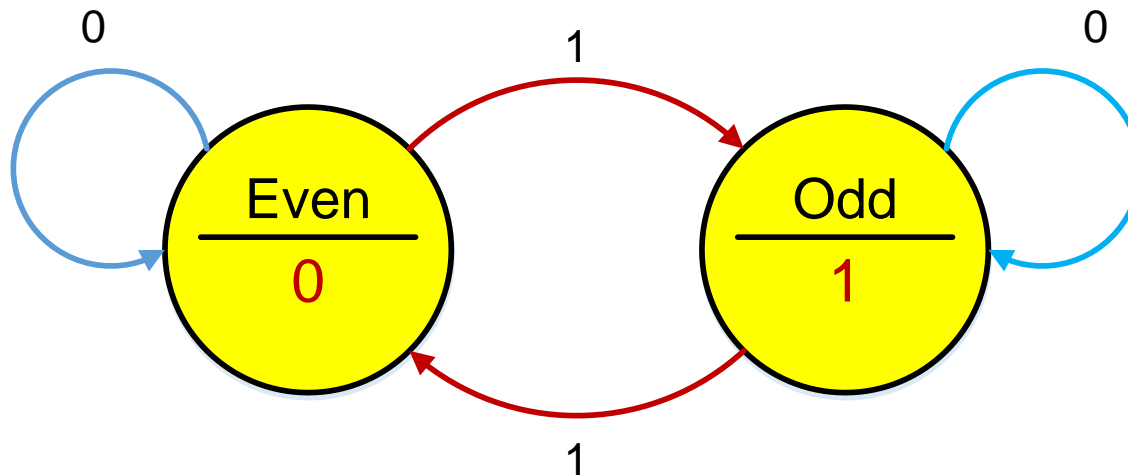
- **ตัวอย่าง :** กำหนดให้ระบบหนึ่งมี Input 1 บิต และ Output 1 บิต โดยระบบนี้จะอ่านข้อมูลทุกวินาที (หมายถึง 1 วินาทีอ่านข้อมูล 1 ครั้ง) จากนั้นจะนำข้อมูลไปบวกสะสม โดยหากข้อมูลในระบบเป็นเลขคี่ Output จะมีค่าเป็น 1 และหากข้อมูลในระบบเป็นเลขคู่ Output จะมีค่าเป็น 0 โดยแสดงผลออกทาง LED





# FSM : Finite State Machine

- สามารถเขียนเป็น state diagram ได้ดังนี้
  - กำหนดให้มี 2 state เนื่องจาก Output จะมี 0 หรือ 1 เท่านั้น
  - หาก state ปัจจุบันเป็น เลขคู่ ถ้า Input เป็น 0 จะอยู่ state เดิม แต่ถ้าเป็น 1 จะเปลี่ยน state เป็นเลขคี่ (output จะเปลี่ยนตาม state)







# C Struct

- ในภาษา C จะมีโครงสร้างข้อมูลแบบหนึ่งเรียกว่า struct หรือ structure

```
struct state {  
    unsigned char out;  
    unsigned int wait;  
} st;
```

```
struct state {  
    unsigned char out;  
    unsigned int wait;  
} ;  
struct state st;
```

- state คือ data type ของ struct (ยังไม่มีตัวตน) st เป็น instance ของ struct มีตัวตนแล้ว สามารถนำไปเก็บข้อมูลได้
- สามารถใช้ `typedef struct state` Stype ได้ ซึ่งเมื่ออ้างถึง Stype จะมีค่าเท่ากับการอ้าง struct state
- การอ้างถึงข้อมูลใน struct จะใช้ st.out, st.wait

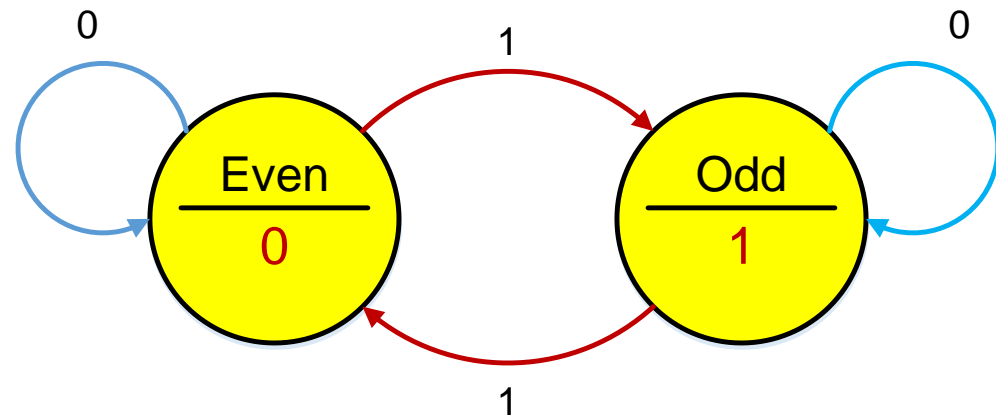


# FSM : Finite State Machine

- โปรแกรมรับ Input จาก Switch ทุก 1 วินาที หากเป็นเลขคู่ให้ LED ดับ หากเป็นเลขคี่ให้ LED ติด

```
#define even 0
#define odd 1
struct state {
    unsigned char out;
    unsigned int wait;
    unsigned char next[2];
}
```

```
typedef struct state SType;
SType FSM[2] = {
    {0,1000,{even,odd}},
    {1,1000,{odd,even}}
};
```



```
unsigned char cState=even;
```

```
while (1) {
    digitalWrite(LED, FSM[cState].out);
    delay(FSM[cState].wait);
    input = digitalRead(PIN);
    cState = FSM [cState].next[input];
}
```

# FSM : Finite State Machine



- สรุป
  - จำนวน State จะแปรตามจำนวน Output
  - เงื่อนไขในการเปลี่ยน state จะแปรตาม Input
  - ในแต่ละ state ต้องไล่เงื่อนไขให้ครบ เช่น ถ้ามี 2 Input จะต้องมี 4 เงื่อนไข ถ้ามี 3 Input ก็จะต้องมี 8 เงื่อนไข



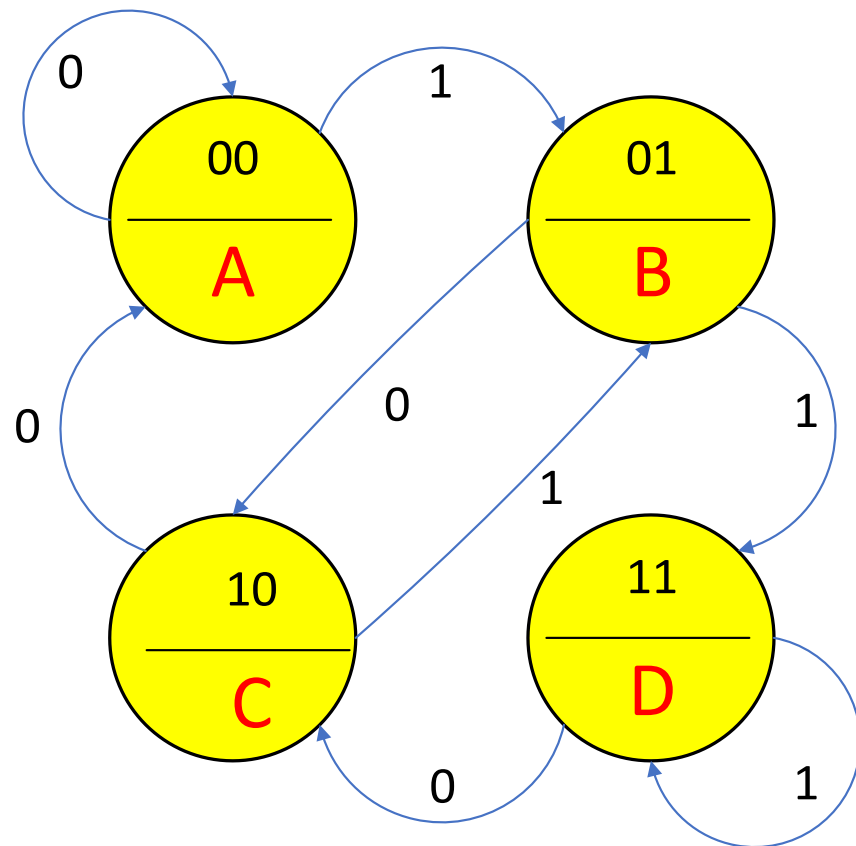
# FSM : Finite State Machine

- **Activity** ให้เขียน FSM Diagram และเขียนโปรแกรมรับ Input จาก Switch จำนวน 1 ตัวทุก 1 วินาที โดยรับ 2 ครั้งติดกัน โดยเริ่มที่ 00
  - หากเป็นเลข 00 ให้แสดง A ที่ Serial Monitor
  - หากเป็นเลข 01 ให้แสดง B ที่ Serial Monitor
  - หากเป็นเลข 10 ให้แสดง C ที่ Serial Monitor
  - หากเป็นเลข 11 ให้แสดง D ที่ Serial Monitor



# FSM : Finite State Mach

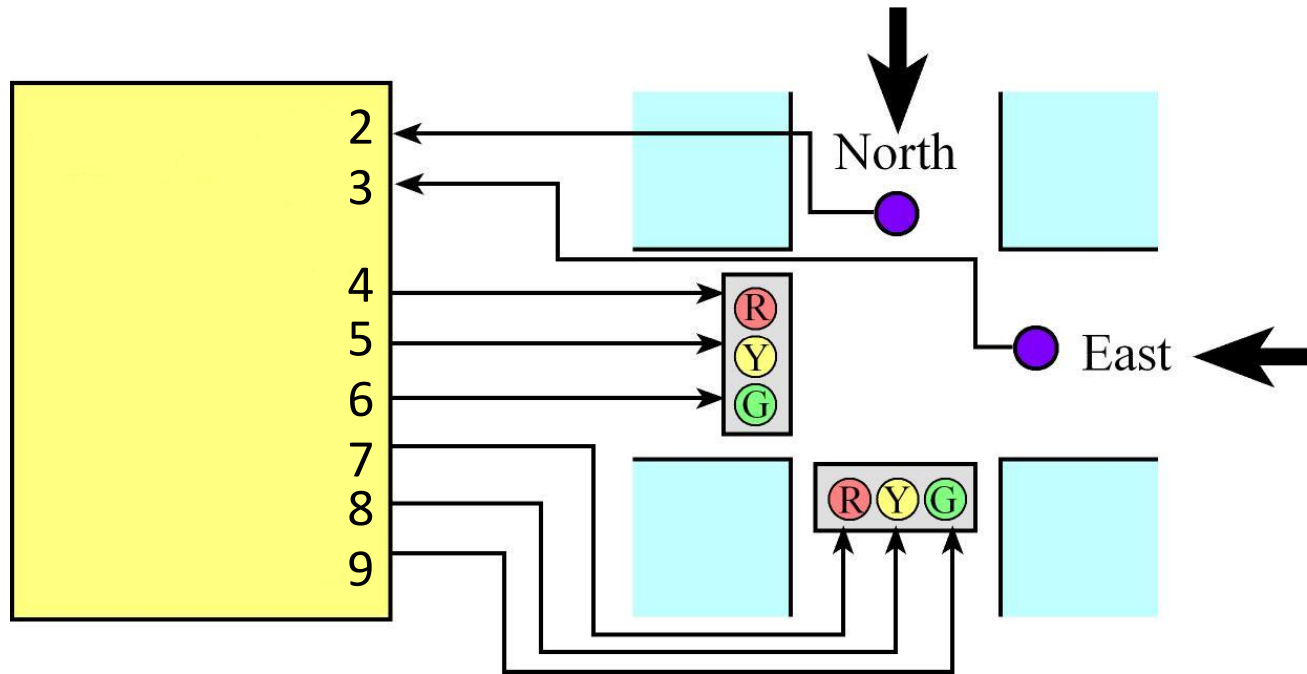
- มี 4 state กำหนดให้ชื่อ state 00, 01, 10, 11
- Output เป็น A, B, C, D





# FSM : Finite State Machine

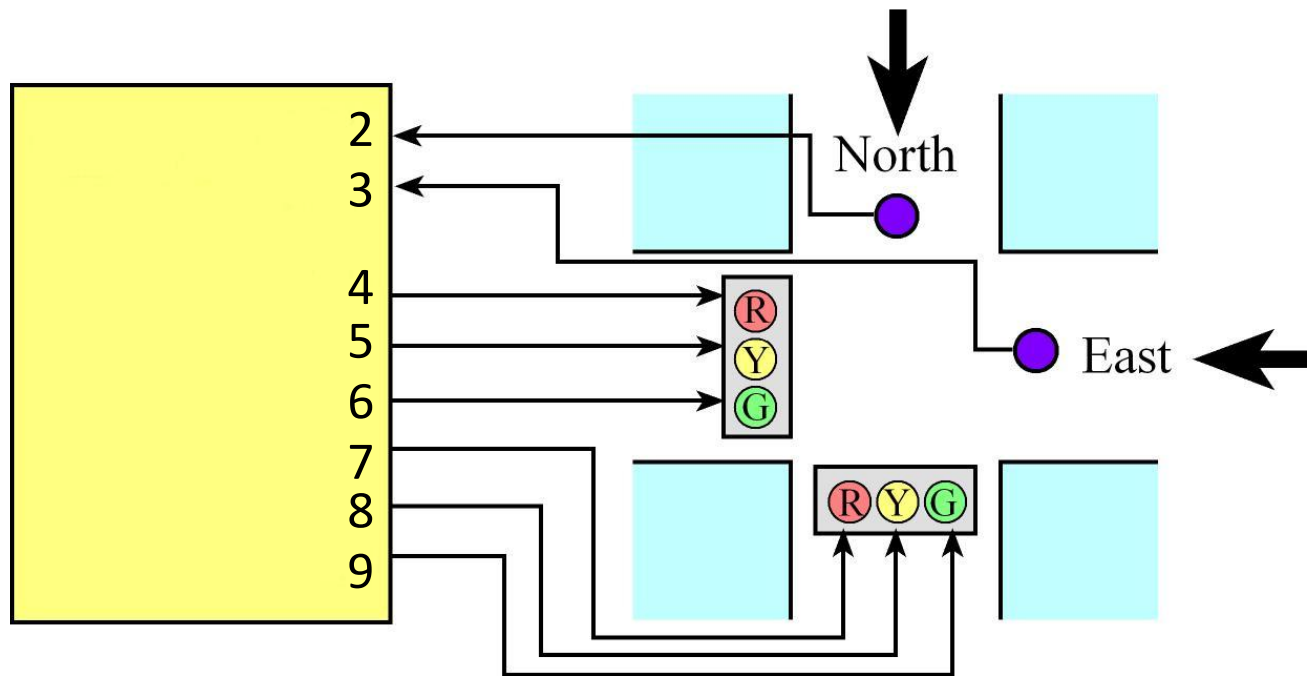
- **Example :** ให้ออกแบบส่วนควบคุมไฟจราจร สำหรับ 4 แยกแห่งหนึ่ง โดยรถวิ่งทางเดียว โดยมีเป้าหมายลดการจราจร และลดการรอไฟแดง ระบบมีเซ็นเซอร์ตรวจจับรถยนต์ที่รอแต่ละด้าน





# FSM : Finite State Machine

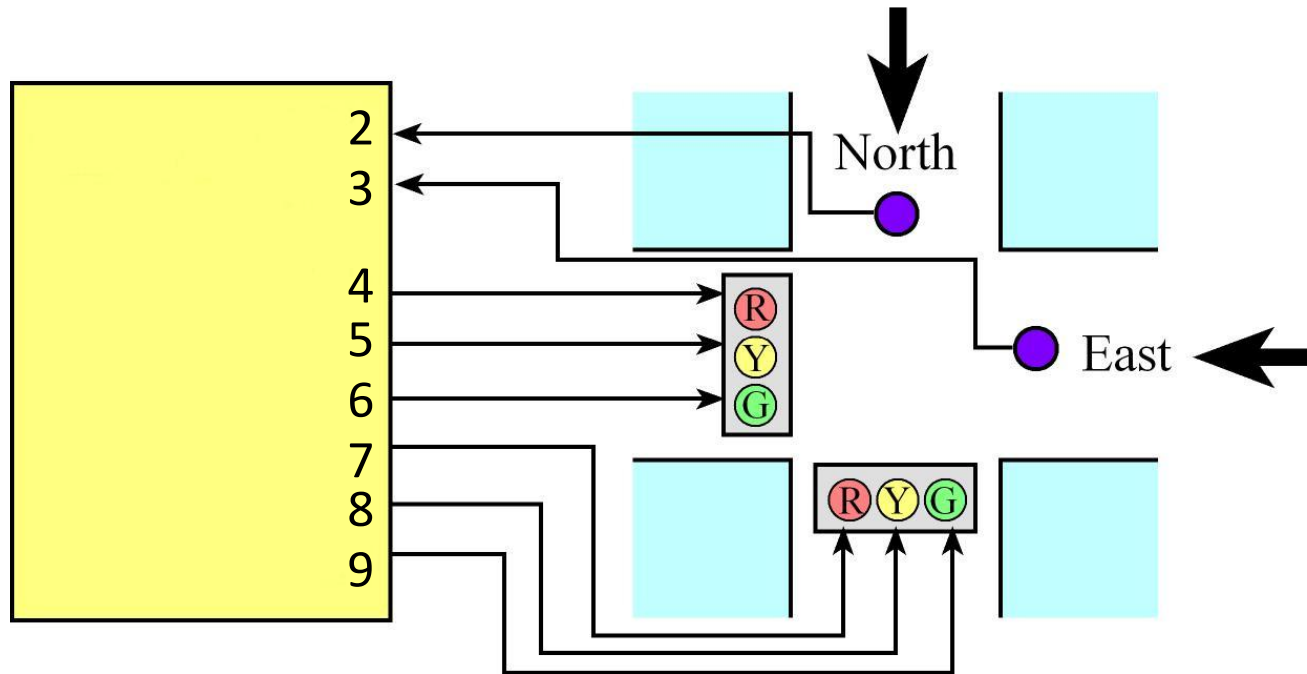
- Input :  $2=0, 3=0$  คือ ไม่มีรถทั้งสองด้าน  
 $2=0, 3=1$  คือ มีรถที่ด้าน East  
 $2=1, 3=0$  คือ มีรถที่ด้าน North  
 $2=1, 3=1$  คือ มีรถทั้งสองด้าน





# FSM : Finite State Machine

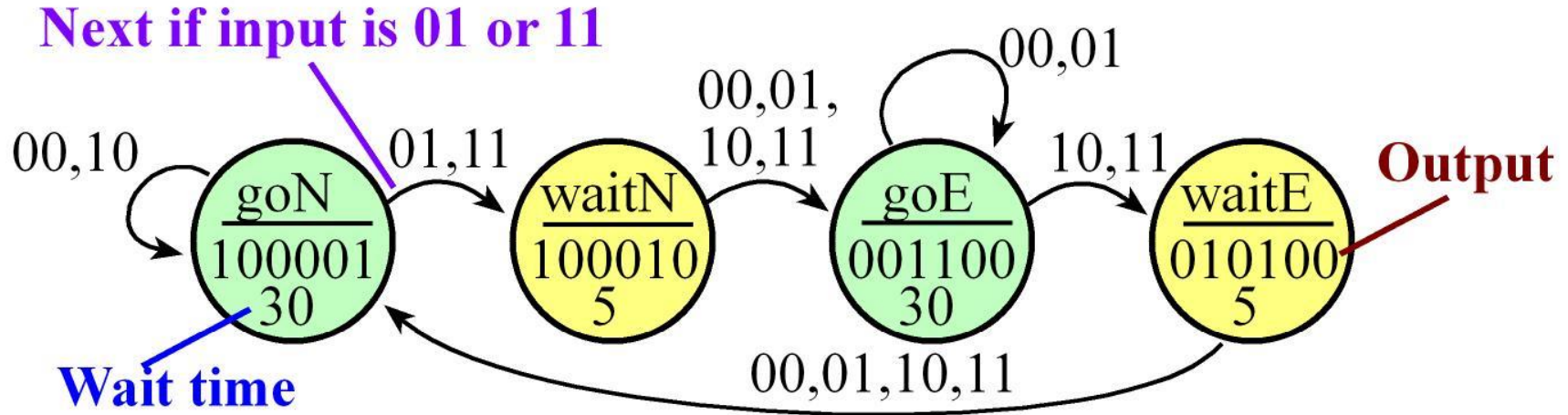
- Output: Pin 4,5,6,7,8,9 (East แดง เหลือง เขียว, North แดง เหลือง เขียว)
- State:   
goN, 100 001 East ไฟแดง, North ไฟเขียว   
waitN, 100 010 East ไฟแดง, North ไฟเหลือง   
goE, 001 100 East ไฟเขียว, North ไฟแดง   
waitE, 010 100 East ไฟเหลือง, North ไฟแดง



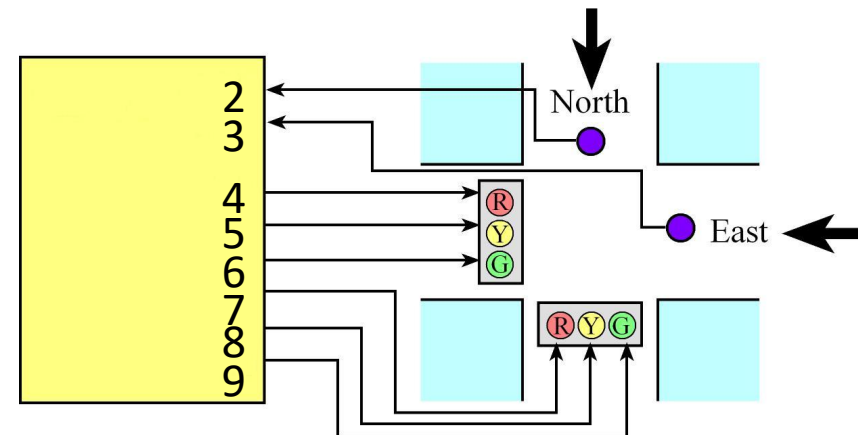




# FSM : Finite State Machine

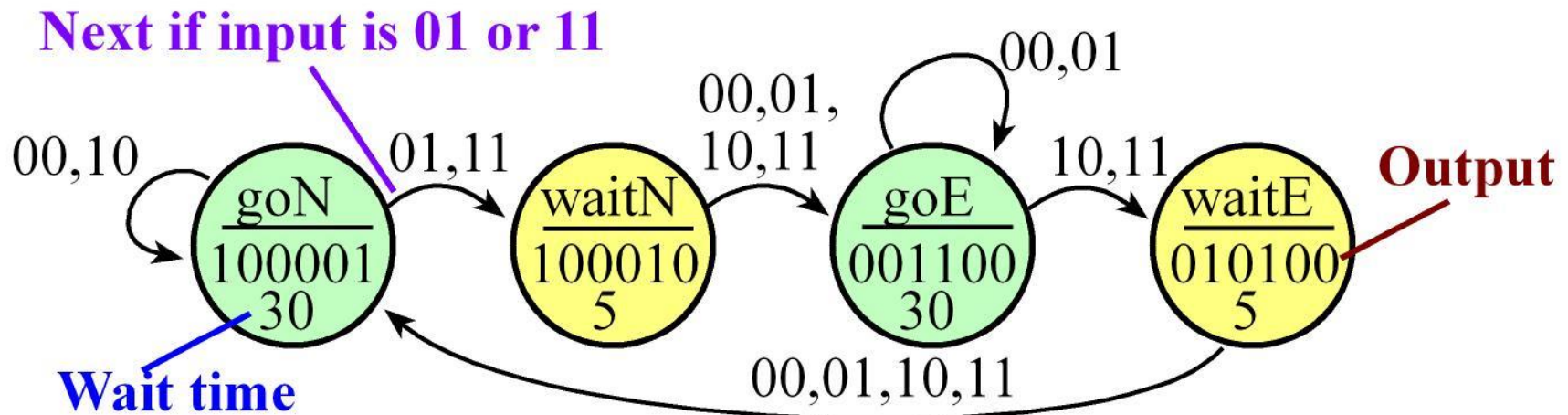


- 2=0, 3=0 คือ ไม่มีรถทั้งสองด้าน
- 2=0, 3=1 คือ มีรถที่ด้าน East
- 2=1, 3=0 คือ มีรถที่ด้าน North
- 2=1, 3=1 คือ มีรถทั้งสองด้าน





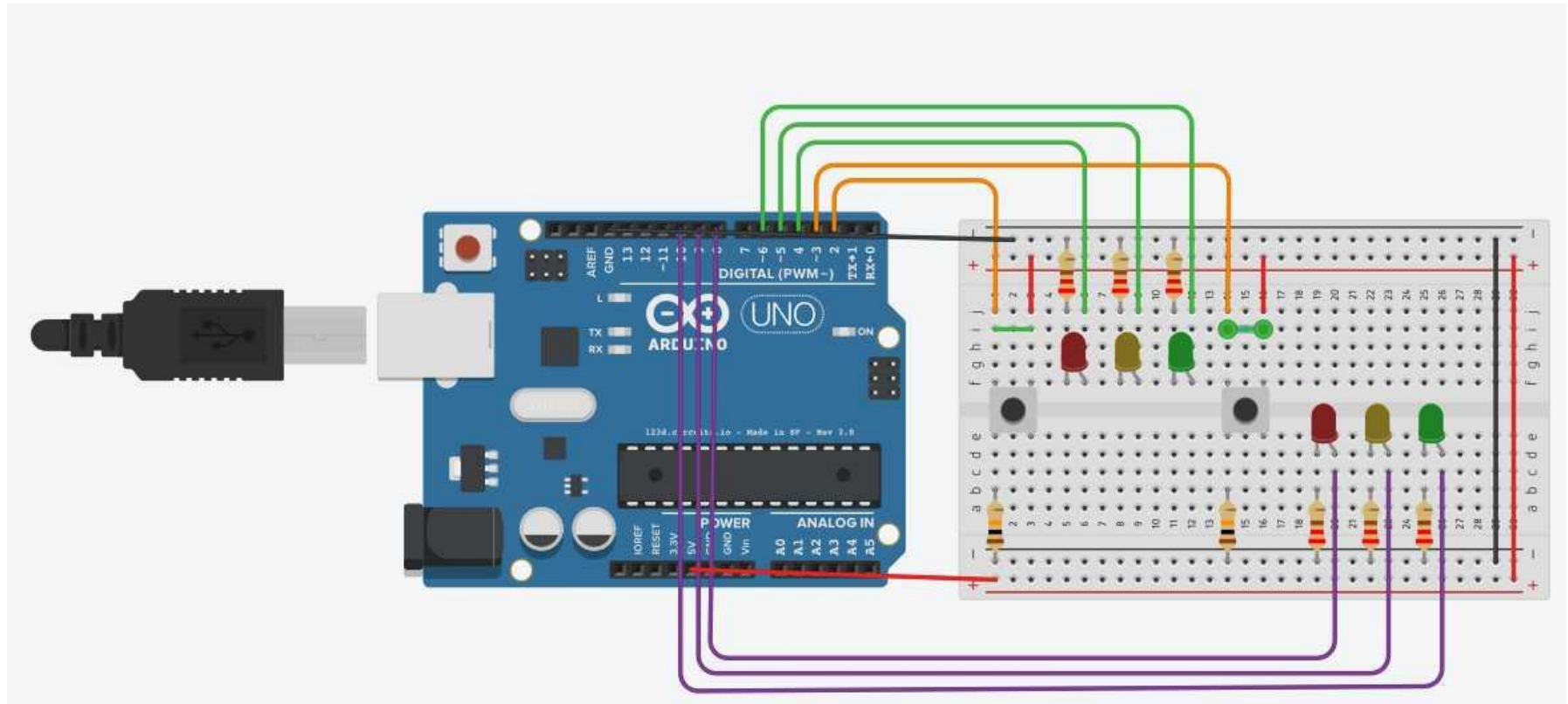
# FSM : Finite State Machine



State Transition Table

Num	Name	Lights	Time	In=0	In=1	In=2	In=3
0	goN	100001	30	goN	waitN	goN	waitN
1	waitN	100010	5	goE	goE	goE	goE
2	goE	001100	30	goE	goE	waitE	waitE
3	waitE	010100	5	goN	goN	goN	goN

# FSM : Finite State Machine





# FSM : Finite State Machine

```
#define LED_W_R 4
#define LED_W_Y 5
#define LED_W_G 6
#define WEST_BUTTON_PIN 2

#define LED_S_R 8
#define LED_S_Y 9
#define LED_S_G 10
#define SOUTH_BUTTON_PIN 3

#define goW 0
#define waitW 1
#define goS 2
#define waitS 3

struct State {
    unsigned long ST_Out;    // 6-bit pattern to street output
    unsigned long Time;      // delay in ms units
    unsigned long Next[4];}; // next state for inputs 0,1,2,3

typedef const struct State SType;

SType FSM[4]={
    {B00001100,2000,{goW,goW,waitW,waitW}},
    {B00010100,300,{goS,goS,goS,goS}},
    {B00100001,2000,{goS,waitS,goS,waitS}},
    {B00100010,300,{goW,goW,goW,goW}}
};
```



# FSM : Finite State Machine

```
unsigned long S=0;  // index to the current state
```

```
void setup() {  
    pinMode(LED_W_R, OUTPUT);  
    pinMode(LED_W_Y, OUTPUT);  
    pinMode(LED_W_G, OUTPUT);  
    pinMode(WEST_BUTTON_PIN, INPUT);  
  
    pinMode(LED_S_R, OUTPUT);  
    pinMode(LED_S_Y, OUTPUT);  
    pinMode(LED_S_G, OUTPUT);  
    pinMode(SOUTH_BUTTON_PIN, INPUT);  
}
```

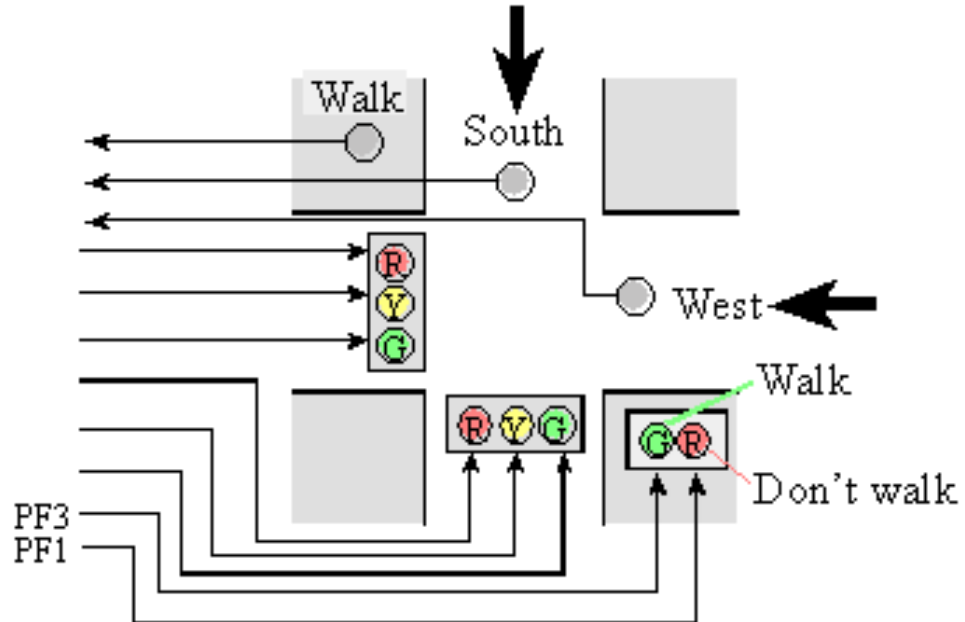
```
int input,input1, input2;
```

```
void loop() {  
    digitalWrite(LED_W_R, FSM[S].ST_Out & B000000001);  
    digitalWrite(LED_W_Y, FSM[S].ST_Out & B000000010);  
    digitalWrite(LED_W_G, FSM[S].ST_Out & B000000100);  
  
    digitalWrite(LED_S_R, FSM[S].ST_Out & B000001000);  
    digitalWrite(LED_S_Y, FSM[S].ST_Out & B000010000);  
    digitalWrite(LED_S_G, FSM[S].ST_Out & B000100000);  
  
    delay(FSM[S].Time);  
  
    input1 = digitalRead(WEST_BUTTON_PIN);  
    input2 = digitalRead(SOUTH_BUTTON_PIN);  
  
    input = input2*2+input1;  
    S = FSM[S].Next[input];  
}
```

# Assignment #5



- ให้สร้างระบบจำลองไฟจราจร 4 แยก แบบ one-way (S->N)(W->E) โดยกำหนดให้มี 3 Sensor ได้แก่ 2 car sensor และ 1 sensor สำหรับคนข้าม
- ข้อกำหนด คือ 1) ต้องไม่มีไฟเหลืองหรือเขียวพร้อมกัน 2 ทาง 2) ถ้ารถวิ่งในทิศทางหนึ่ง ในอีกทิศทางต้องเป็นไฟแดง 3) หากคนข้ามไฟทำงาน ไฟรถต้องแดงทั้ง 2 ทิศทาง 4) ต้องมีการกระพริบเตือนคนข้ามว่าใกล้จะไฟแดงแล้ว โดยกระพริบติดดับ 3 ครั้ง 5) ถ้าไม่มีการกดจะค้างสถานะเดิม 6) ถ้าทุกทางมีรถ/คนหมด จะวนเขียวไปเรื่อยๆ 7) เวลาในแต่ละ state ให้กำหนดได้เอง
- ให้ใช้ delay ในโปรแกรม  
ได้จุดเดียวที่ delay ของ state  
ห้ามใช้ delay หรือ millis()  
กระพริบไฟคนข้าม



# Assignment #5



- การส่งงาน (5 คะแนน)
  1. ให้ Demo กับ Staff
  2. เอกสารให้ส่งใน MS Teams รายงาน 1 ฉบับ ประกอบด้วย state transition graph, state transition table, รูป, source code และคำอธิบายโดยย่อ



***For your attention***