Internetul Lucrurilor

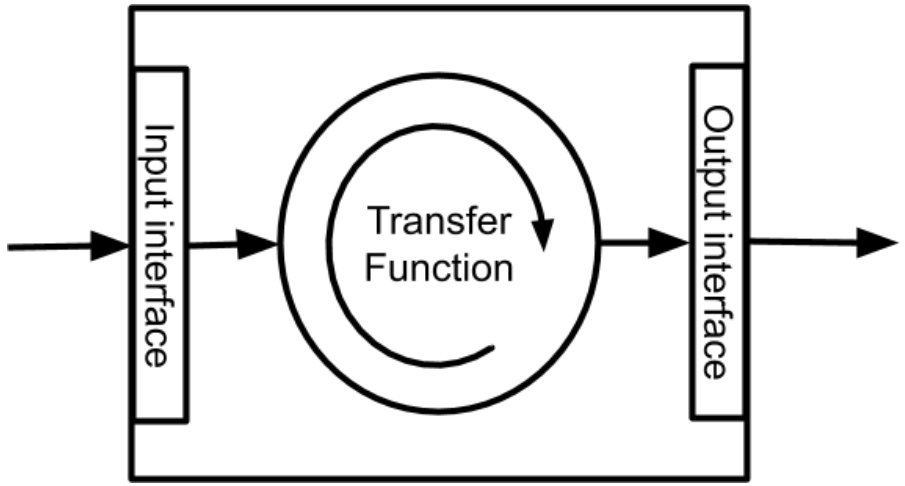
Automate finite

FSM

Automate Finite - FSM

Automat Finit este o abstracție ce descrie o soluție a unei probleme.

Definește comportamentul sistemului sub forma de mecanism care își schimba stările
ca reacție la intrările sistemului si produce ieșiri corespunzătoare

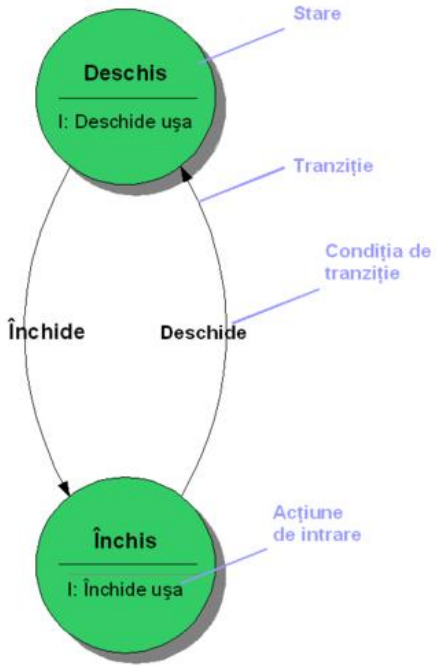


1. Număr finit de stări, *una este definita ca si inițiala*
2. Număr finit de intrări in sistem
3. Număr finit de ieșiri generate.
4. Funcție de transfer pentru tranzițiile dintre stări.
5. Funcție de definire a ieșirilor

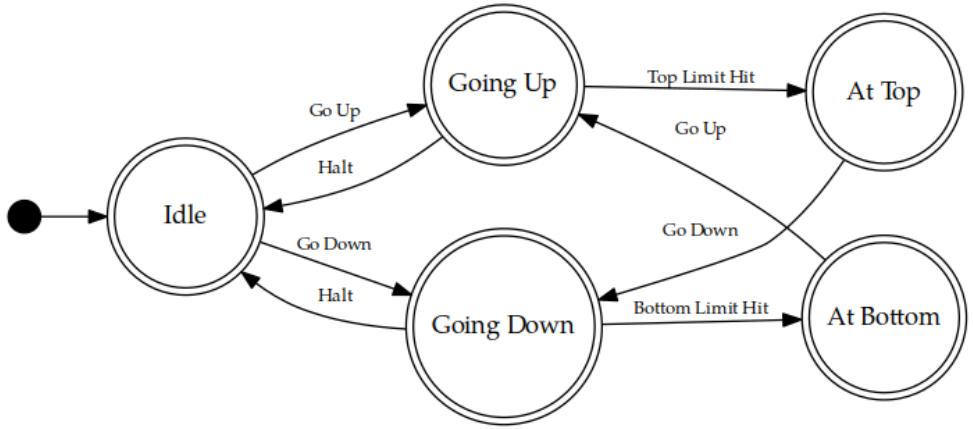
En: Finite State Mashine

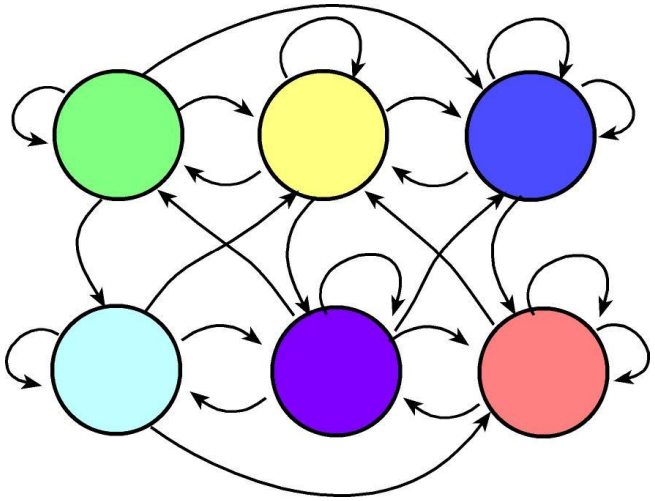
Automate Finite - Diagrame

"mașină cu un număr finit de stări" este un model de comportament compus din stări, tranziții și acțiuni.



- ***O stare*** stochează informații despre trecut, adică reflectă schimbările intrării de la inițializarea sistemului până în momentul de față.
- ***O tranziție*** indică o schimbare de stare și este descrisă de **o condiție** care este nevoie să fie îndeplinită pentru a declanșa tranziția.
- ***O acțiune*** este o descriere a unei activități ce urmează a fi executată la un anumit moment.

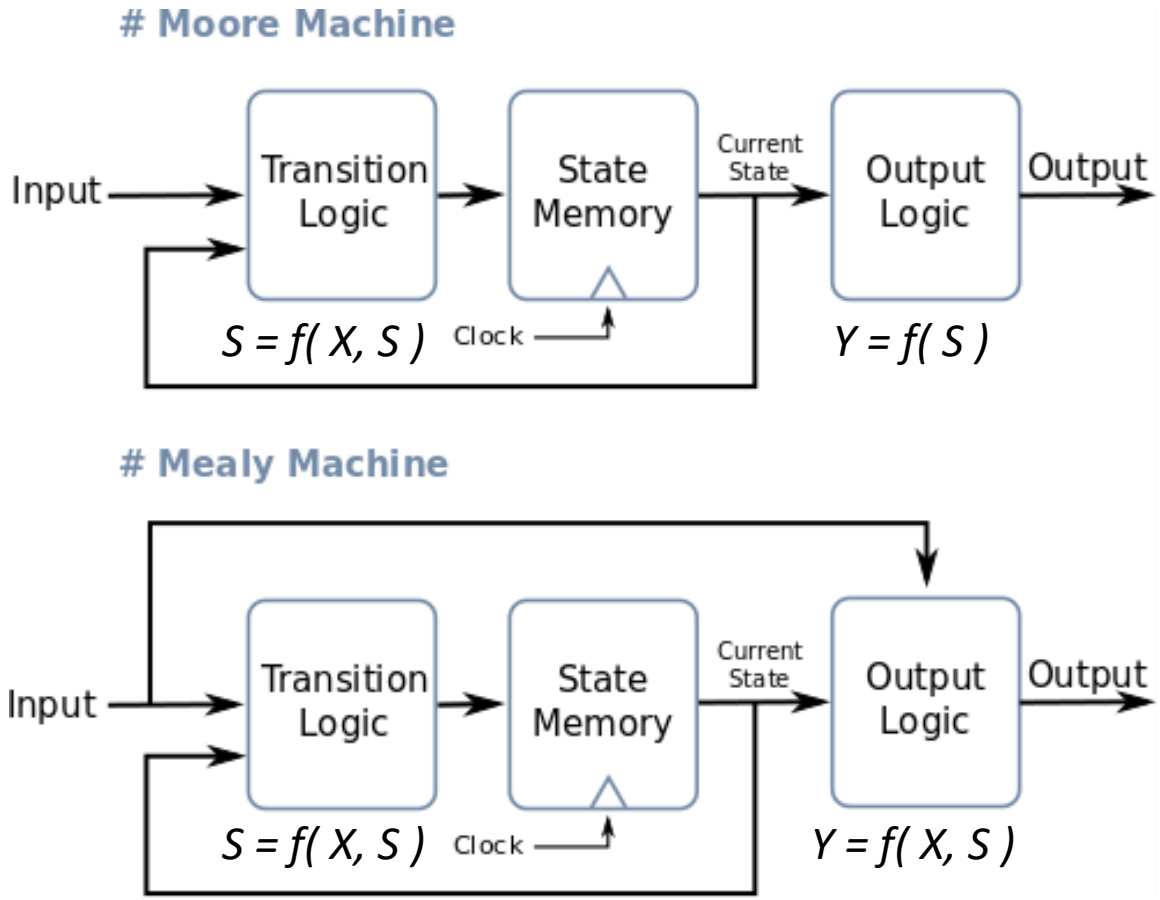


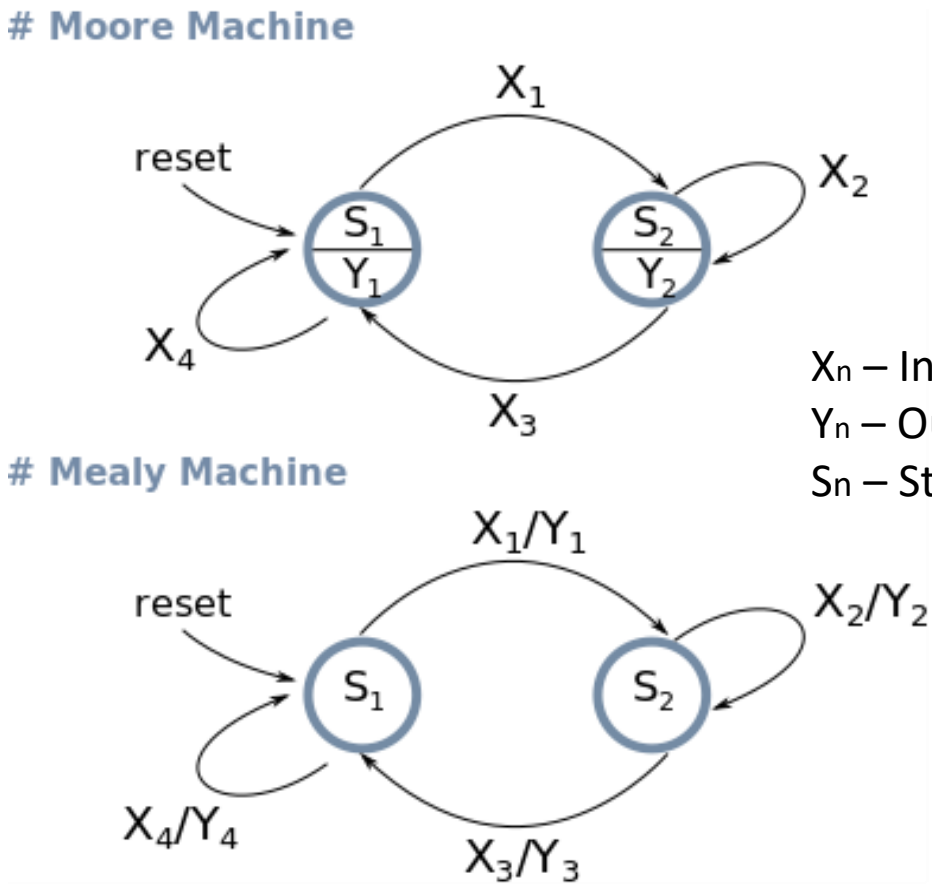


https://ro.wikipedia.org/wiki/Automat_finit

Google Search: ”FSM Diagramm example”

Automate Finite – Mealy vs Moore





Xn – Inputs
Yn – Outputs
Sn – States

http://calangorobo.blogspot.com/2018/

Automate Finite - Evaluare

Automat Moore

NextState = f(Input, CurrentState)

Output = g(CurrentState)

TaskMooreFSM(){

1. ***Evaluare Ieșiri***, dependente **doar** de ***Starea curenta***
2. ***Reținere*** perioada definita de stare
3. ***Colectare Intrări***
4. ***Evaluare Stare următoare*** dependenta de ***Intrare*** si ***Stare curentă***

}

Automat Mealy

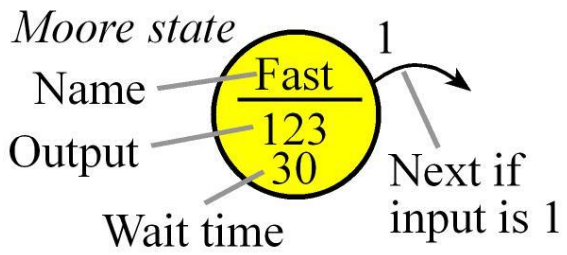
NextState = f(Input, CurrentState)

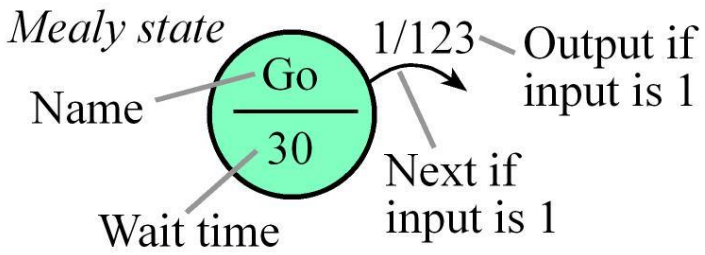
Output = h(Input, CurrentState)

TaskMealyFSM()

1. ***Reținere*** perioada definita de stare
2. ***Colectare Intrări***
3. ***Evaluare Ieșiri***, dependente de ***Intrare*** si ***Starea curentă***
4. ***Evaluare Stare următoare*** dependenta de ***Intrare*** si ***Stare curentă***

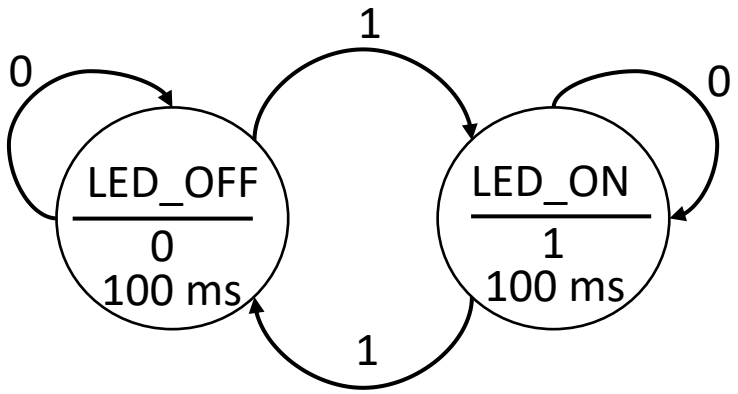
}

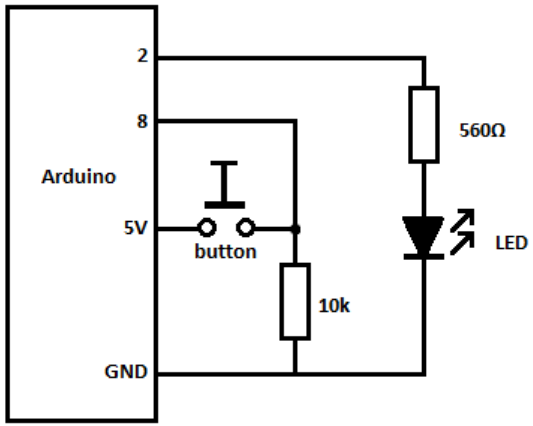




https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.03x+1T2016

Automate Finite – Button/Led : FSM design





Num	Name	Out	Delay	In = 0	In = 1
0	LED_OFF	0	100 ms	LED_OFF	LED_ON
1	LED_ON	1	100 ms	LED_ON	LED_OFF

```
#define LED_OFF_STATE  0
#define LED_ON_STATE   1

struct State {
    unsigned long Out;     // Led State
    unsigned long Time;    // delay in 10ms units
    unsigned long Next[2];// next state for inputs 0,1
};
typedef const struct State STyp;

STyp FSM[2]={
        {0,10,{LED_OFF_STATE, LED_ON_STATE  }},
        {1,10,{LED_ON_STATE,  LED_OFF_STATE }}
};
```

https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.03x+1T2016

Automate Finite – Button/Led : FSM Controller

```
#define LED_PIN 2
#define BUTTON_PIN 8

#define LED_OFF_STATE  0
#define LED_ON_STATE   1

struct State {
    unsigned long Out;    // Led State
    unsigned long Time;   // delay in 10ms units
    unsigned long Next[2];// next state for inputs 0,1
};
typedef const struct State STyp;

STyp FSM[2]={
        {0,10,{LED_OFF_STATE, LED_ON_STATE  }},
        {1,10,{LED_ON_STATE,  LED_OFF_STATE }}
};

int FSM_State = LED_OFF_STATE;

void setup()
{
    // Init Button
    pinMode(BUTTON_PIN, INPUT);
    // Init LED
    pinMode(LED_PIN, OUTPUT);
    // Init Initial State
    FSM_State = LED_OFF_STATE;
}
```

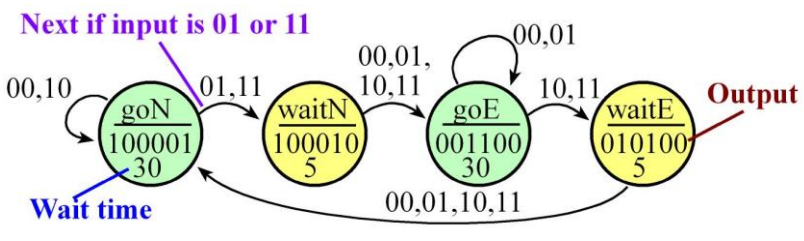
TaskMooreFSM(){

1. ***Evaluare Ieșiri***, dependente **doar** de ***Starea curenta***
2. ***Reținere*** perioada definita de stare
3. ***Colectare Intrări***
4. ***Evaluare Stare următoare*** dependenta de ***Intrare*** si ***Stare curentă***

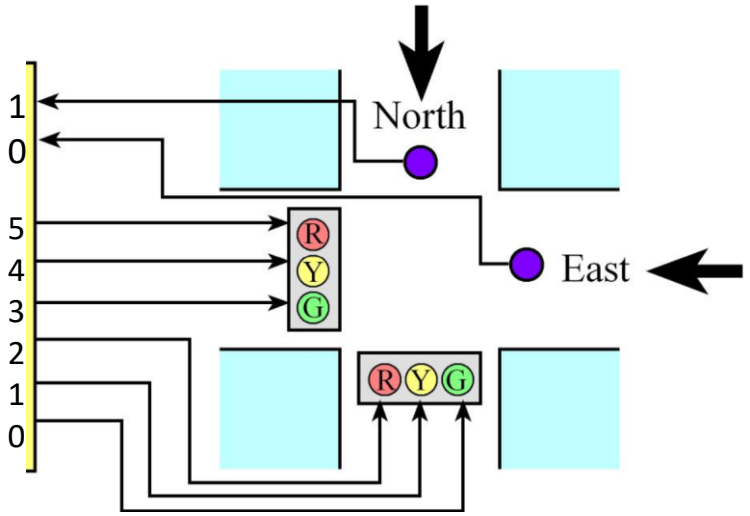
}

```
// The loop function is called in an endless loop
void loop()
{
    // 1. Output Based on current state
    int output = FSM[FSM_State].Out;
    digitalWrite(LED_PIN, output);

    // 2. wait for time relevant to state
    delay(FSM[FSM_State].Time * 10);

    // 3. Get Input
    int input = digitalRead(BUTTON_PIN);

    // 4. Change state based on input and current state
    FSM_State = FSM[FSM_State].Next[input];

}
```

Automate Finite – Semafor : FSM design



Num	Name	Out	Delay	In = 0	In = 1	In = 2	In = 3
0	goN	100 001	30 s	goN	waitN	goN	waitN
1	waitN	100 010	5 s	goE	goE	goE	goE
2	goE	001 100	30 s	goE	goE	waitE	waitE
3	waitE	010 100	5 s	goN	goN	goN	goN




```
struct State {
  unsigned long Out;  // 6-bit pattern to output
  unsigned long Time; // delay in 10ms units
  unsigned long Next[4];}; // next state for inputs 0,1,2,3

typedef const struct State STyp;

    #define goN   0 // 0b00
    #define waitN 1 // 0b01
    #define goE   2 // 0b10
    #define waitE 3 // 0b11

    STyp FSM[4]={
     {0b100001, 3000,{goN,  waitN, goN,   waitN }},
     {0b100010, 500, {goE,  goE,   goE,   goE   }},
     {0b001100, 3000,{goE,  goE,   waitE, waitE }},
     {0b010100, 500, {goN,  goN,   goN,   goN   }}};
```

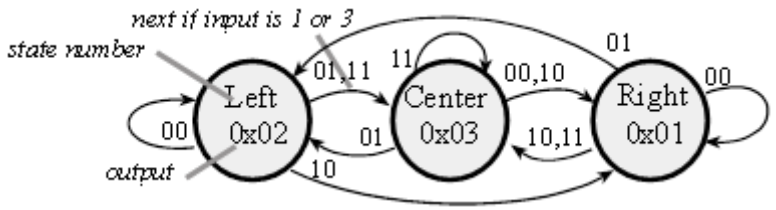
https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.03x+1T2016

Automate Finite – Semafor: FSM Controller

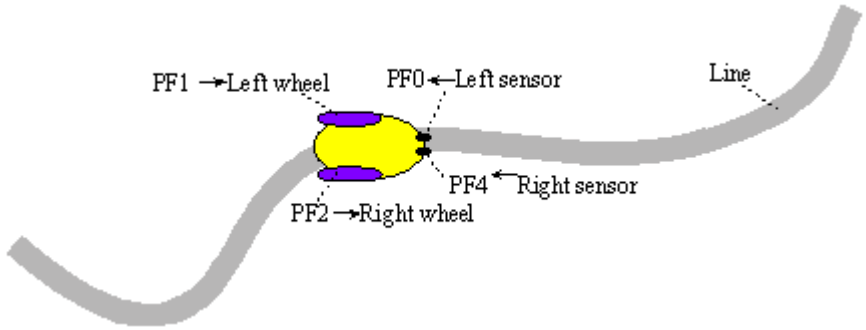
```
#define NORTH_PIN 1
#define EAST_PIN  2
#define EAST_RED_PIN     3
#define EAST_YELLOW_PIN  4
#define EAST_GREEN_PIN   5
#define NORTH_RED_PIN    6
#define NORTH_YELLOW_PIN 7
#define NORTH_GREEN_PIN  8
#define goN   0 // 0b00
#define waitN 1 // 0b01
#define goE   2 // 0b10
#define waitE 3 // 0b11

struct State {
    unsigned long Out;     // 6-bit pattern to output
    unsigned long Time;    // delay in 10ms units
    unsigned long Next[4]; // next state for inputs 0,1,2,3
};
typedef const struct State STyp;

STyp FSM[4]={
 {0b100001, 3000,{goN,  waitN, goN,   waitN }},
 {0b100010, 500, {goE,  goE,   goE,   goE   }},
 {0b001100, 3000,{goE,  goE,   waitE, waitE }},
 {0b010100, 500, {goN,  goN,   goN,   goN   }}
};
int FSM_State = goN;

void setup() {
    // Init Button
    pinMode(NORTH_PIN, INPUT);
    pinMode(EAST_PIN, INPUT);
    // Init LED
    pinMode(EAST_RED_PIN, OUTPUT);
    pinMode(EAST_YELLOW_PIN, OUTPUT);
    pinMode(EAST_GREEN_PIN, OUTPUT);
    pinMode(NORTH_RED_PIN, OUTPUT);
    pinMode(NORTH_YELLOW_PIN, OUTPUT);
    pinMode(NORTH_GREEN_PIN, OUTPUT);
    // Init Initial State
    FSM_State = goN;
}
```

```
int GetInput(void) {
    int northButton = digitalRead(NORTH_PIN);
    int eastButton = digitalRead(EAST_PIN);
    if (northButton && eastButton)
        return 0b11;
    else if (northButton)
        return 0b10;
    else if (eastButton)
        return 0b01;
    else
        return 0b00;
}

void SetOutput(int out) {
    int ledState;
    ledState = (out & (1 << 5)) ? HIGH : LOW;
    digitalWrite(EAST_RED_PIN, ledState);
    ledState = (out & (1 << 4)) ? HIGH : LOW;
    digitalWrite(EAST_YELLOW_PIN, ledState);
    ledState = (out & (1 << 3)) ? HIGH : LOW;
    digitalWrite(EAST_GREEN_PIN, ledState);
    ledState = (out & (1 << 2)) ? HIGH : LOW;
    digitalWrite(NORTH_RED_PIN, ledState);
    ledState = (out & (1 << 1)) ? HIGH : LOW;
    digitalWrite(NORTH_YELLOW_PIN, ledState);
    ledState = (out & (1 << 0)) ? HIGH : LOW;
    digitalWrite(NORTH_GREEN_PIN, ledState);
}

// The loop function is called in an endless loop
void loop() {
    // 1. Output Based on current state
    int output = FSM[FSM_State].Out;
    SetOutput(output);
    // 2. wait for time relevant to state
    delay(FSM[FSM_State].Time * 10);
    // 3. Get Input
    int input = GetInput();
    // 4. Change state based on input and current state
    FSM_State = FSM[FSM_State].Next[input];
}
```

Automate Finite – Line Follow : FSM design



Num	Name	Out	Delay	In = 0	In = 1	In = 2	In = 3
0	Center	0x03	1 ms	Right	Left	Right	Center
1	Left	0x02	1 ms	Left	Center	Right	Center
2	Right	0x01	1 ms	Right	Left	Center	Center




```
struct State {
  unsigned long out;        // 2-bit output
  unsigned long delay;      // time to delay in 10ms
  unsigned long next[4];};  // Next if 2-bit input is 0-3
typedef const struct State STyp;
```

```
#define Center 0
#define Left   1
#define Right  2
STyp FSM[3]={
  {0x03, 1, { Right, Left,   Right,  Center }},  // Center
  {0x02, 1, { Left,  Center, Right,  Center }},  // Left
  {0x01, 1, { Right, Left,   Center, Center }}   // Right
};
```

https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.03x+1T2016