EX1

Consider a virtual reality (VR) boxing duel game , it requires two 2 players for the match to start . The players have to pay using a Magnetic payment card to participate in the game . The system can be implemented using an alternative plan B :

- Each Player presence is detected by its magnetic sensor when the card is scanned , so we have **2 magnetic sensors** .
- The game is off when the players are absent
- The game starts when the 2 players get scanned
- A huge green LED indicate the presence of both players and the beginning of the fight

Dr.Catherine Elias
Eng. Abdalla Mohamed Eng. Mohamed Elshafie Eng. Maysarah El Tamalawy

Embedded Systems CSEN701

Dr. Catherine Elias

Eng. Abdalla Mohamed

Office: C1.211
Mail : abdalla.abdalla@guc.edu.eg

Eng. Mohamed Elshafie

Office: C1.211
Mail : Mohamed.el-shafei@guc.edu.eg

Eng. Maysarah El Tamalawy

Office: C7.201
Mail : Maysarah.mohamed@guc.edu.eg

Outline :

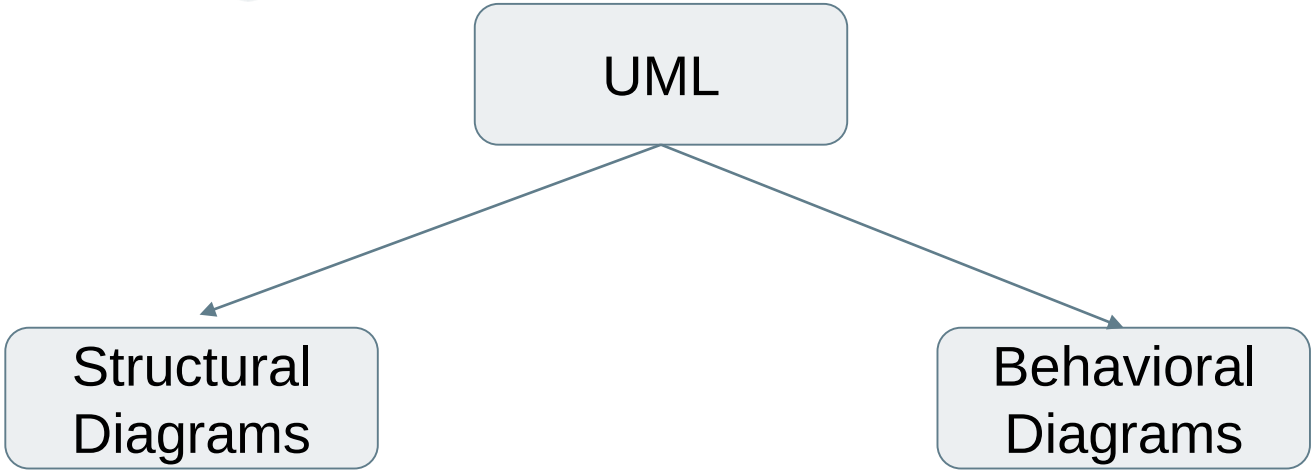
- Introduction to UML.
- FSM.
- Mealy machine vs Moore machine.
- Example.

Outline :

- **Introduction to UML.**
- FSM.
- Mealy machine.
- Moore machine.
- Code Examples

What is UML?

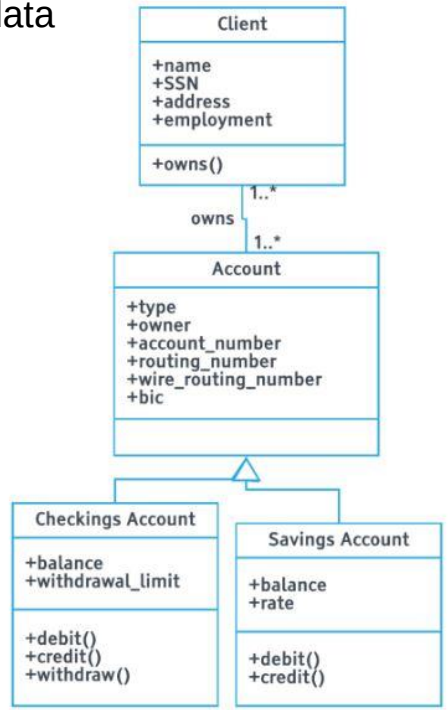
UML (Unified Modeling Language) is a standardized visual language used in software engineering for modeling, designing, and documenting software systems. It provides a common, visual way to represent the structure and behavior of a system. UML is not a programming language; rather, it's a set of diagrams and symbols that allow developers to communicate and design software in a clear and consistent manner.


UML
Structural Diagrams
Behavioral Diagrams

Structural Diagrams

- **Structural Diagrams** in Unified Modeling Language (UML) provide a visual representation of the static aspects of a system. They focus on the elements that make up the system and how they relate to one another. These diagrams emphasize the organization, composition, and relationships between various components, classes, and entities within a system. Structural diagrams help to illustrate the system's architecture and its constituent parts, offering a clear overview of the system's structure.

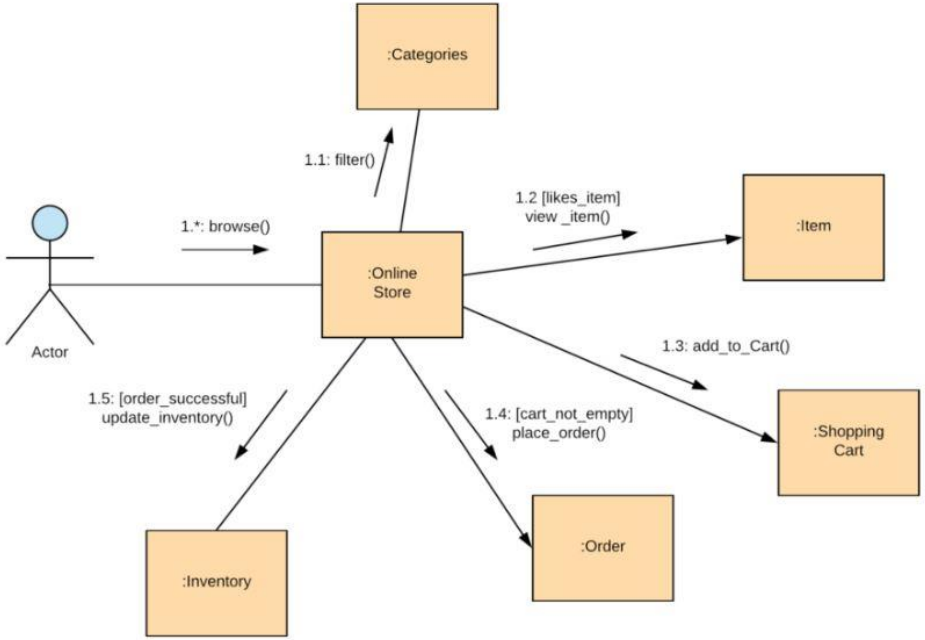
Class diagrams: is example for structural diagrams ,class diagrams contain classes, alongside with their attributes (also referred to as data fields) and their behaviors (also referred to as member functions).



Behavioral Diagrams

- **Behavioral diagrams:** are a type of Unified Modeling Language (UML) diagrams that depict the dynamic behavior of a system. They are used to visualize, specify, construct, and document the dynamic aspects of a system. Behavioral diagrams illustrate how the system behaves and interacts with itself and other entities..

Communication Diagrams: in UML highlight interactions and relationships between objects or components in a system to achieve specific goals. While they primarily focus on behavior by showing message flows, they also offer insights into the system's structure, illustrating involved objects and their collaboration.



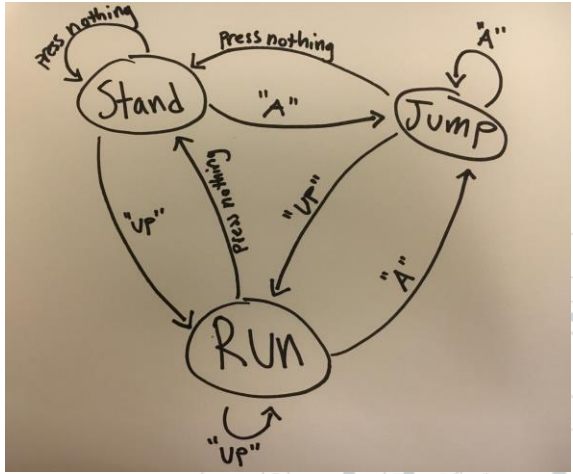
- Before the implementation of any system. The system needs to be designed (modeled) to specify exactly what do we expect from it and how will it behave.
- Embedded systems are no exception.

Outline :

- Introduction to UML.
- **FSM**
- Mealy machine vs Moore machine.
- Examples
- FSM coding

FSM

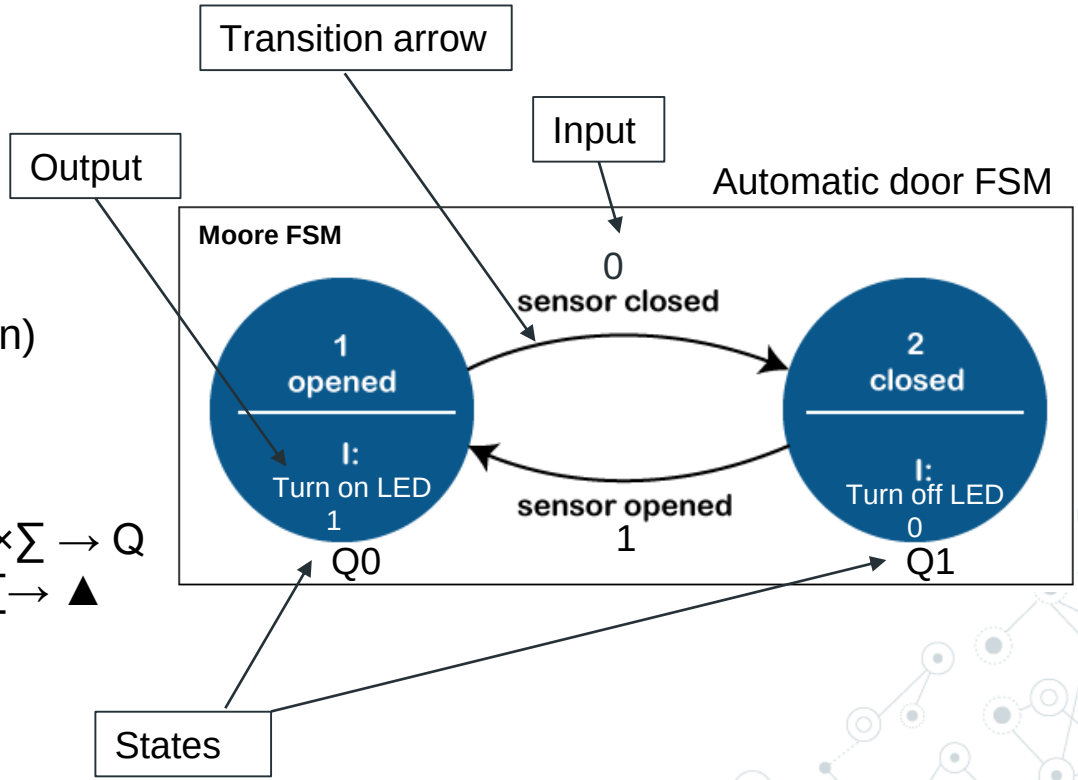
- One of the ways used to model embedded systems is Finite State Machine (or Automata)
- FSM idea is based on the concept that the system is made of states. The system initially starts at a specific.state, and depending on the input given to the system, the system transitions to the next state and produces an output.
- So … the system sequentially transitions from one state to another depending on the input given.



FSM

It has 6 tuples : (Q, q0, ∑, ▲, δ, λ')

- Q is a finite set of states (Q0,Q1. .. ..Qn)
- q0 is the initial state
- ∑ is the input alphabet
- ▲ is the output alphabet
- δ is the transition function that maps Q×∑ → Q
- 'λ' is the output function that maps Q×∑→ ▲

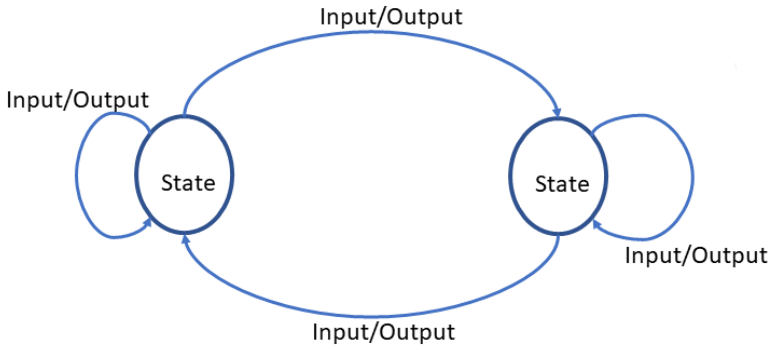


Outline :

- Introduction to UML.
- FSM
- **Mealy machine vs Moore machine.**
- Examples
- FSM coding

Types of FSM

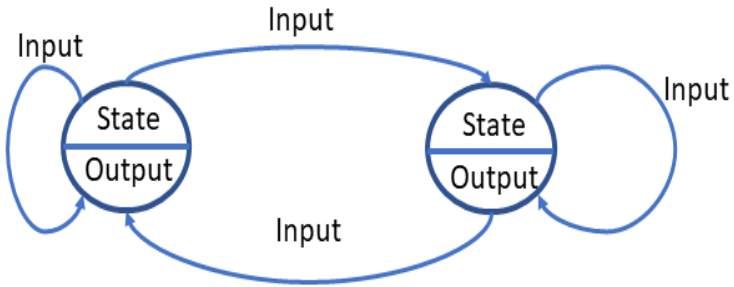
Mealy



Generic Mealy model

The difference is all about the **output** . .

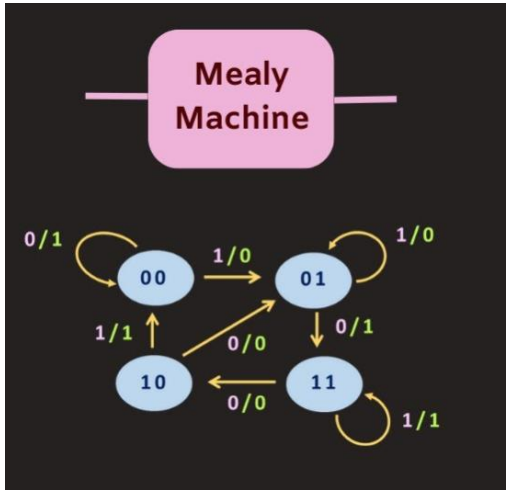
Moore



Generic Moore model

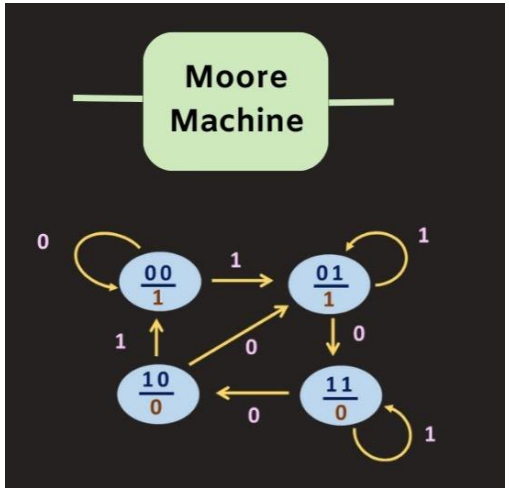
Mealy FSM

If the output of the system at a specific state depends on the input given to the current state , then the FSM is mealy FSM



Moore FSM

If the output of the system at a specific state does not depend on the input given, And it only depends on the current state , then the FSM is Moore FSM .

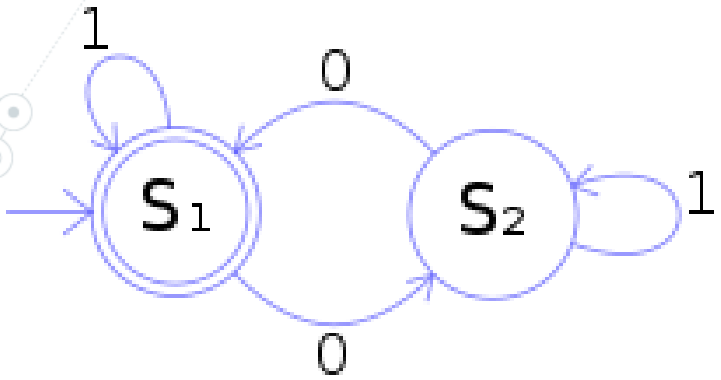


FSM

Using the diagram we obtain :

- State transition table
- State transition function
- Output function

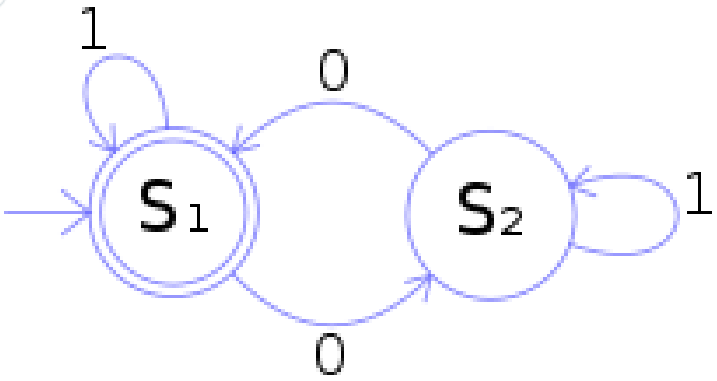
State transition table



State | input	0	1
S1	S2	S1
S2	S1	S1

State transition function

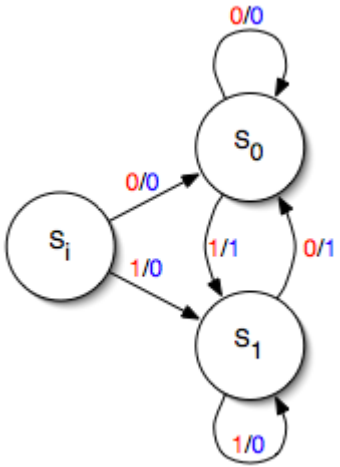
State transition function is a function that takes the current state, and the current input, and returns the next state
T(current state, current input) = next state



With U as input :
S1 = (S1 & U) | (S2 & !U)
S2 = (S1 & !U) | (S2 & U)

Output function

a function that shows when is the certain output value returned
For mealy it depends on the input :

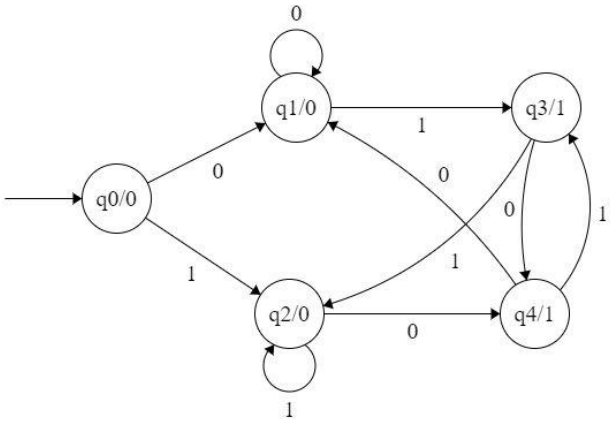


When is your output value = 1 ?

O = (S0 & U) | (S1 & !U)

Output function

a function that shows when is the certain output value returned
For Moore it only depends on the state :



When is your output value = 1 ?

O = q3 | q4

Outline :

- Introduction to UML.
- FSM.
- Mealy machine vs Moore machine.
- **Examples**
- FSM coding

EX1.A

Consider a virtual reality (VR) boxing duel game , it requires two 2 players for the match to start . The players have to pay using a Magnetic payment card to participate in the game . The system can be implemented using plan A :

- Players presence is detected by a magnetic sensor when the card is scanned
- The game is off when the players are absent
- The game is loading when one player is present
- The game starts when the 2 players exist
- A huge green LED indicate the presence of both players and the beginning of the fight

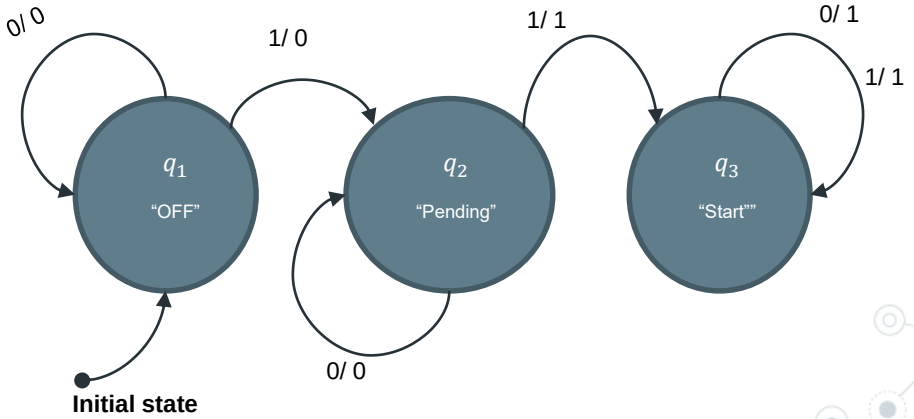
Modelling using Mealy FSA(Mealy)

- Identify the states in our system : we have 3 states of the game Q ={Q0,Q1,Q2)
  - **Q0** --- initial OFF state
  - **Q1** --- Pending State (one player is present)
  - **Q2** --- ON state (2 players are present)

- Identify the inputs to our states : we have only one input **u** which is the sensor reading 1/0 indicating the presence of the players. u = 0/1 .
  - u = 0 indicating no player card is scanned at the moment
  - u = 1 indicating a player card is scanned at the moment
- ∑ = (0,1)
- Identify the outputs of our states : we have only one output **y** which is the LED either On or Off 1/0 indicating the start of the game . y = 0/1 .
  - y = 0 indicates the LED is off
  - y = 1 indicates the LED is on
  - ▲ = (0,1)

Modelling using Mealy FSA

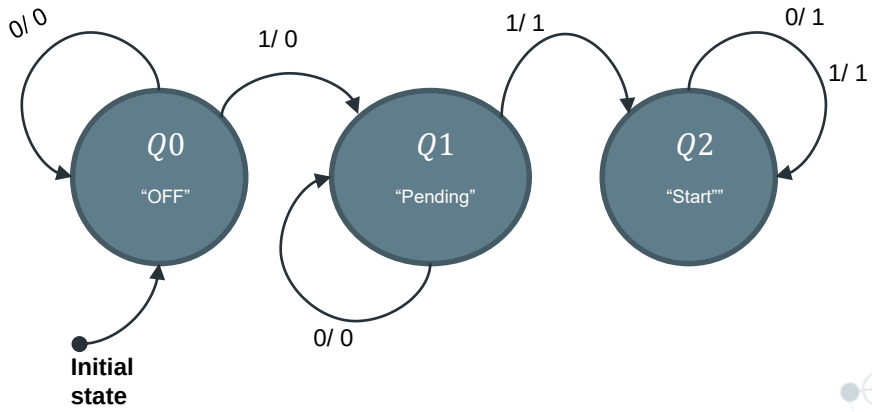
- Indicate the initial state as shown in the figure
- Indicate number of states, inputs and outputs as a mealy FSM .
- Number of states of mealy FSM is independent of the output
- Draw all the states as circles and transitions as arrows .
- All transition arrows are labelled by (input/output) u/y .
- Each states has 2 transition for the different inputs 0/1 .
- U=1 indicates that a new player card is scanned



Modelling using Mealy FSA

State transition Table

States/Inputs	0	1
$Q0$	$Q0$	$Q1$
Q1	$Q1$	$Q2$
Q2	$Q2$	$Q2$



State transition functions with U as input :

- Q0 = Q0 & !U
- Q1 = (Q1 & !U) | (Q0 & U)
- Q2 = (Q2 & !U) | (Q1 & U) | (Q2 & U)

Output function :

y = (Q2 & U) | (Q2 & !U) | (Q1 & U)

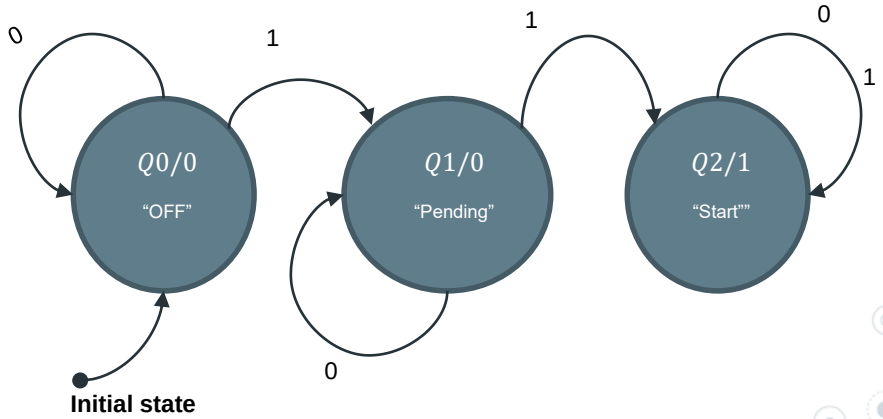
Modelling using Moore FSM

- Identify the states in our system : we have 3 states of the game Q ={Q0/0,Q1/0,Q2/1)
  Q0/0 --- initial OFF state with output 0
  Q1 --- Pending State with output 0 (one player is present)
  Q2 --- ON state with output 1 (2 players are present)

- Identify the inputs to our states : we have only one input **u** which is the sensor reading 1/0 indicating the presence of the players. u = 0/1 .
  u = 0 indicating no player card is scanned at the moment
  u = 1 indicating a player card is scanned at the moment
- ∑ = (0,1)
- Identify the outputs of our states : we have only one output **y** which is the LED either On or Off 1/0 indicating the start of the game . y = 0/1 .
  y = 0 indicates the LED is off
  y = 1 indicates the LED is on
- In Moore FSM the output of each state in integrated with the state itself meaning that the number of states is dependent on the number of outputs . ▲ = (0,1)

Modelling using Moore FSA

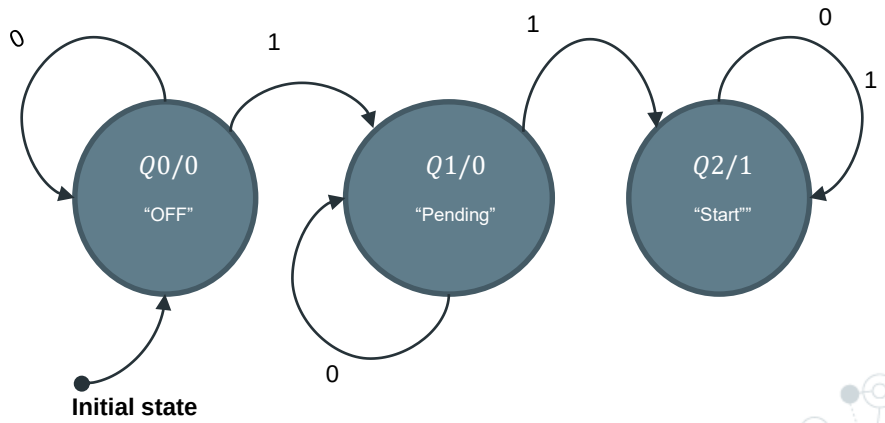
- Indicate the initial state as shown in the figure
- Indicate number of states, inputs and outputs as a mealy FSM .
- Number of states of mealy FSM is independent of the output
- Draw all the states as circles and transitions as arrows .
- All transition arrows are labelled by (input/output) u/y .
- Each states has 2 transition for the different inputs 0/1 .
- U=1 indicates that a new player card is scanned



Modelling using Moore FSA

State transition Table

States/Inputs	0	1
$Q0$	$Q0$	$Q1$
Q1	$Q1$	$Q2$
Q2	$Q2$	$Q2$



State transition functions with U as input :

- Q0/0 = Q0/0 & !U
- Q1/0 = (Q1/0 & !U) | (Q0/0 & U)
- Q2/1 = (Q2/1 & !U) | (Q1/0 & U) | (Q2/0 & U)

Output function :

y = (Q2/1 & U) | (Q2/1 & !U) == Q2/1

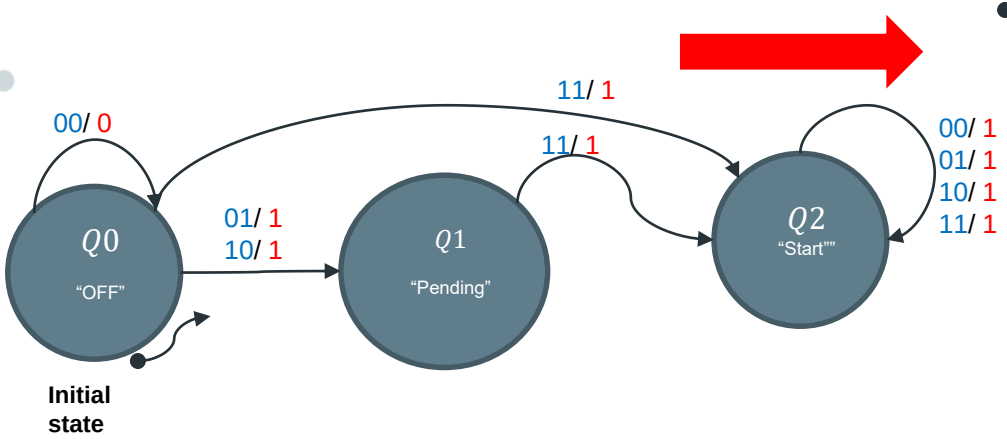
EX1.B

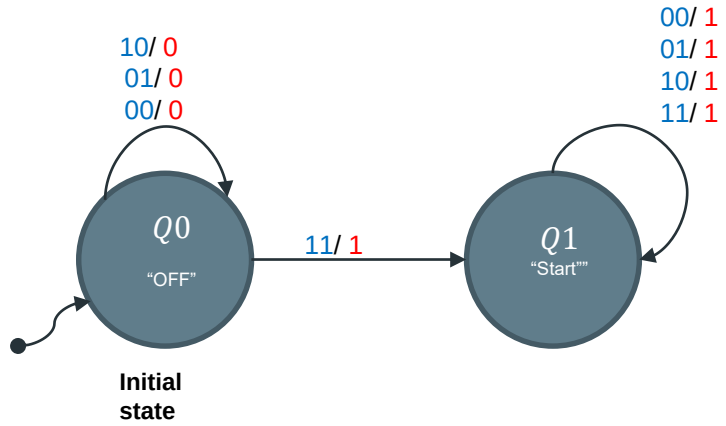
Consider a virtual reality (VR) boxing duel game , it requires two 2 players for the match to start . The players have to pay using a Magnetic payment card to participate in the game . The system can be implemented using an alternative plan B :

- Each Player presence is detected by its magnetic sensor when the card is scanned , so we have **2 magnetic sensors** .
- The game is off when the players are absent
- The game starts when the 2 players get scanned
- A huge green LED indicate the presence of both players and the beginning of the fight

Mealy In Plan B :

- We can implement the FSM using only two states , as the process of scanning is not sequential so the pending state can be skipped (redundant state) .
- The input will represent the 2 sensor so u = u1u0 the set of inputs would be ∑ = (00, 01, 10, 11) .

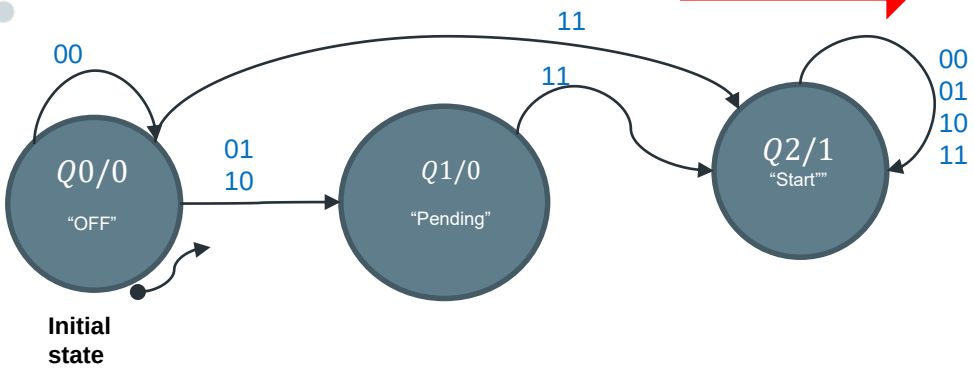


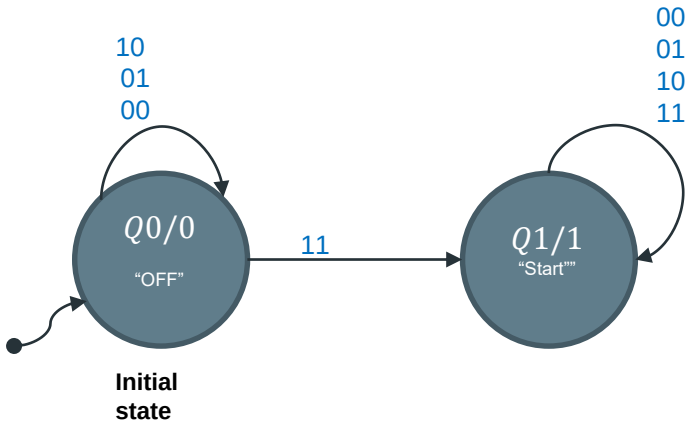


Mealy Machine	Moore Machine
It is a finite state machine whose current output is determined by its current state and the existing external input and output symbol is linked to the transition state and input of a finite state machine.	It is a finite state machine whose current output is solely determined by its current state, and each state of a finite state machine has an output symbol.
It needs fewer states for the same functions' implementation than the Moore machine.	It needs a more significant number of states for the same functions' implementation than the Mealy machine.
It isn't straightforward to design a Mealy machine.	It is easy to design Moore's machine.
Mealy machines react to changes more quickly. They all seem to respond in the same way.	Decoding the output necessitates more logic, resulting in longer circuit delays. They usually react after one clock cycle.
It requires less hardware for designing purposes.	It requires more hardware for designing purposes.
A counter can't be referred to as a mealy machine.	A counter can be referred to as a Moore machine.
The output function's value becomes a function of transitions and changes when the input logic is done in the present state.	The output function's value becomes the function of its current state along with the changes at the edges of the clock whenever a change occurs in the state.

Moore In Plan B :

- We can implement the FSM using only two states , as the process of scanning is not sequential so the pending state can be skipped (redundant state) .
- The input will represent the 2 sensor so u = u1u0 the set of inputs would be ∑ = (00, 01, 10, 11) .





Outline :

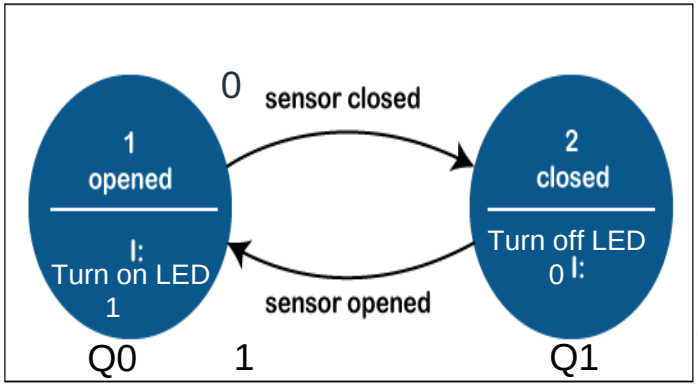
- Introduction to UML.
- FSM.
- Mealy machine vs Moore machine.
- Examples
- **FSM coding**

FSM in C code

```
5  enum state  {off, on } ;  // the list of Qs (states)
6  enum input  {sensor_close,sensor_open} ;
7  enum output {low,high} ;
8  int main( void) {
9  enum state current_state = off ;
10 enum input current_input = low ;
11 enum output current_output = low  ;
12   // initial state , input , output
13   while(1){
14       switch (current_state){
15       case 0 :
16              switch(current_input) {
17                case 0  :  current_output = low; // do an output if mealy then change the state
18                           current_state = off ;    break ;   // if moore just change the state
19                case 1 :   current_output = high ; // if mealy set the output first
20                           current_state  = on ; break ;
21                      }  break ;
22       case 1 :
23      switch(current_input) {
24                           case 0  :  current_output = low; //(Mealy) do an output then change the state
25                                      current_state = off ; break ; // if moore just change the state
26                           case 1 :   current_output = high ; // if mealy set the output first
27                                      current_state  = on ; break ;
28                           }  break ;
29       } break ;
30   }
```

The code will be a list of switch on the states followed by a switch on the inputs .

Recall the first diagram



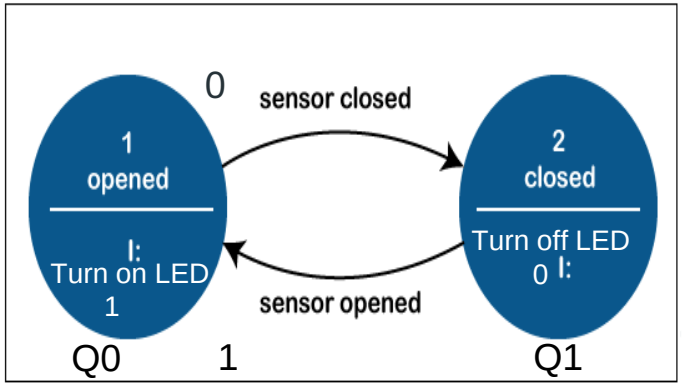
FSM in embedded C code on AVR

```
// build FSM code on AVR
int sensor =  3  ;
int LED = 5 ;
enum state  {off, on } ;  // the list of Qs (states)
enum input  {sensor_close,sensor_open} ;
enum output {low,high} ;
int main( void) {
    enum state current_state = off ;
    int LED_value = low  ;
    int sensor_status = low ;  // initial value of the sensor ( input)
    // initial state , input , output
    while(1){

        switch (current_state){
        int sensor_status = digitalRead(sensor)  ; //Continuously reading the sensor value on pin 3
        case off :
            switch(sensor_status) {
            case low  :  digitalWrite(LED,low); // do
            current_state = off ;    break ;   // if
            case high :  digitalWrite(LED,high) ; //
            current_state  = on ; break ;
            }  break ;
            case on :
                switch(sensor_status) {
                case low  : digitalWrite(LED,low); //
                current_state = off ; break ; // if m
                case high :  digitalWrite(LED,high) ;
                current_state  = on ; break ;
                }  break ;
        } break ;
    }}
```

Get the input using digitalRead() to read the sensor value and use digitalWrite() to induce the output (LED)

Recall the first diagram



THANK YOU