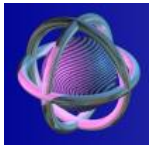
AImage Lab
University of Modena and Reggio Emilia



Biometric System Laboratory
DISI - University of Bologna

Finite State Machines

DOTT. GUIDO BORGHI

A.A. 2019/20

INNOVATION DESIGN

Write on serial monitor

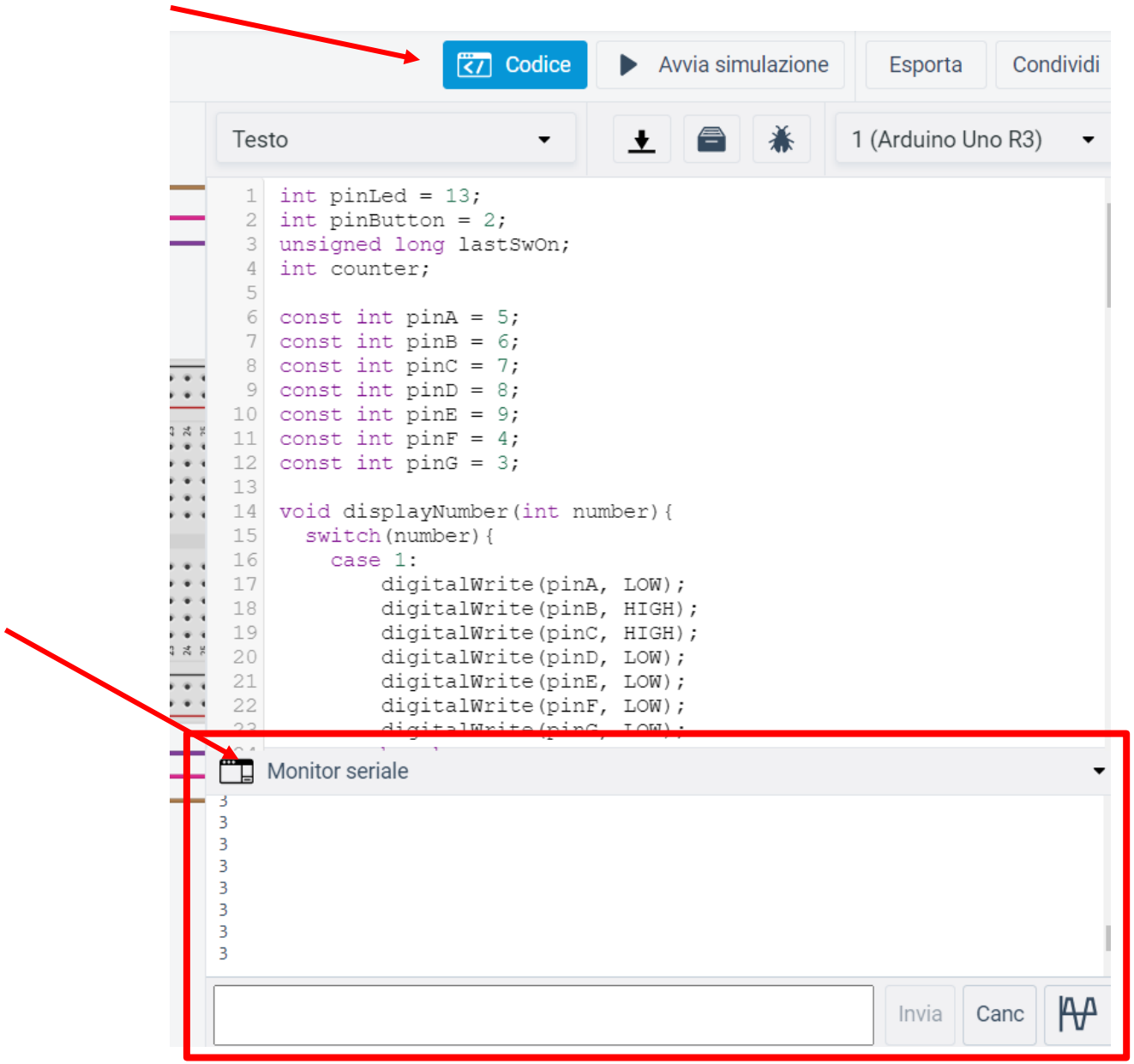
```
void setup()
{
   Serial.begin(9600);
}
```

Sets the data rate in bits per second (baud) for serial data transmission.

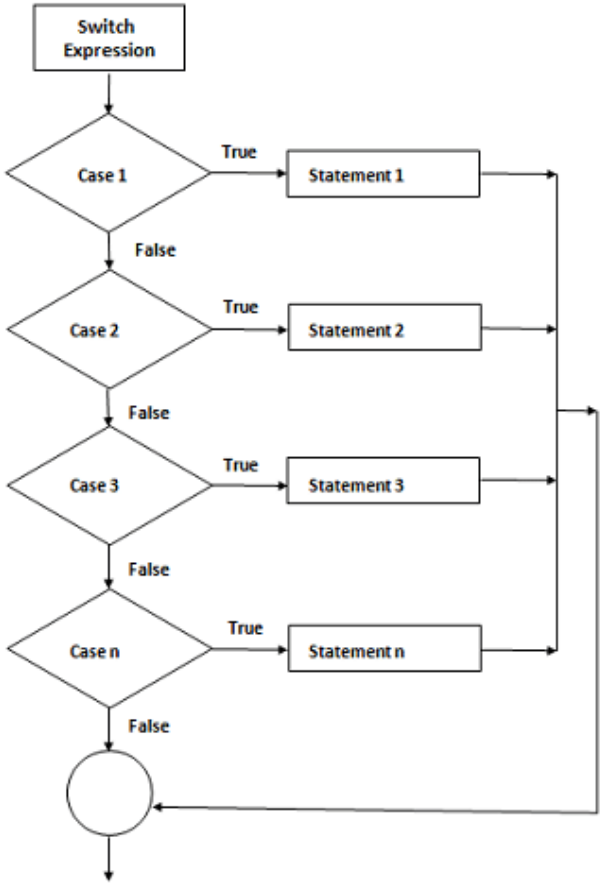
```
void loop()
{
   Serial.println(``Ciao'');
}
```

print();
println();
…

https://www.arduino.cc/reference/en/language/functions/communication/serial/


Codice
Avvia simulazione
Esporta
Condividi
Testo
1 (Arduino Uno R3)
int pinLed = 13;
int pinButton = 2;
unsigned long lastSwOn;
int counter;
const int pinA = 5;
const int pinB = 6;
const int pinC = 7;
const int pinD = 8;
const int pinE = 9;
const int pinF = 4;
const int pinG = 3;
void displayNumber(int number){
switch(number){
case 1:
digitalWrite(pinA, LOW);
digitalWrite(pinB, HIGH);
digitalWrite(pinC, HIGH);
digitalWrite(pinD, LOW);
digitalWrite(pinE, LOW);
digitalWrite(pinF, LOW);
Monitor seriale
3
3
3
3
3
3
3
3
Invia
Canc

Switch statement




```
switch (var) {
  case label1:
    // statements
    break;
  case label2:
    // statements
    break;
  default:
    // statements
    break;
}
```

Switch statement

Like if statements, switch case controls the flow of programs by allowing programmers to specify **different code that should be executed in various conditions**.

In particular, a switch statement compares the value of a variable to the values specified in case statements. **When a case statement is found whose value matches that of the variable, the code in that case statement is run**.

Switch example

```
switch (number) {
  case 1:
    //do something when var equals 1
    break;
  case 2:
    //do something when var equals 2
    break;
  default:
    // if nothing else matches, do the default
    // default is optional
    break;
}
```

Functions

Segmenting code into functions allows a programmer to create **modular pieces of code** that perform a defined task and then return to the area of code from which the function was "called".

The typical case for creating a function is when one needs to **perform the same action multiple times in a program**.

```
<type> <function_name> ( <type> <parameters>){
}
```

Function Example

```
int multiplyTwoNumbers(int a, int b){
        result = a * b;
        return result;
}

void loop(){
        …
        res = multiplyTwoNumbers(5, 4);
        Serial.print(res);
}
```

Arduino functions

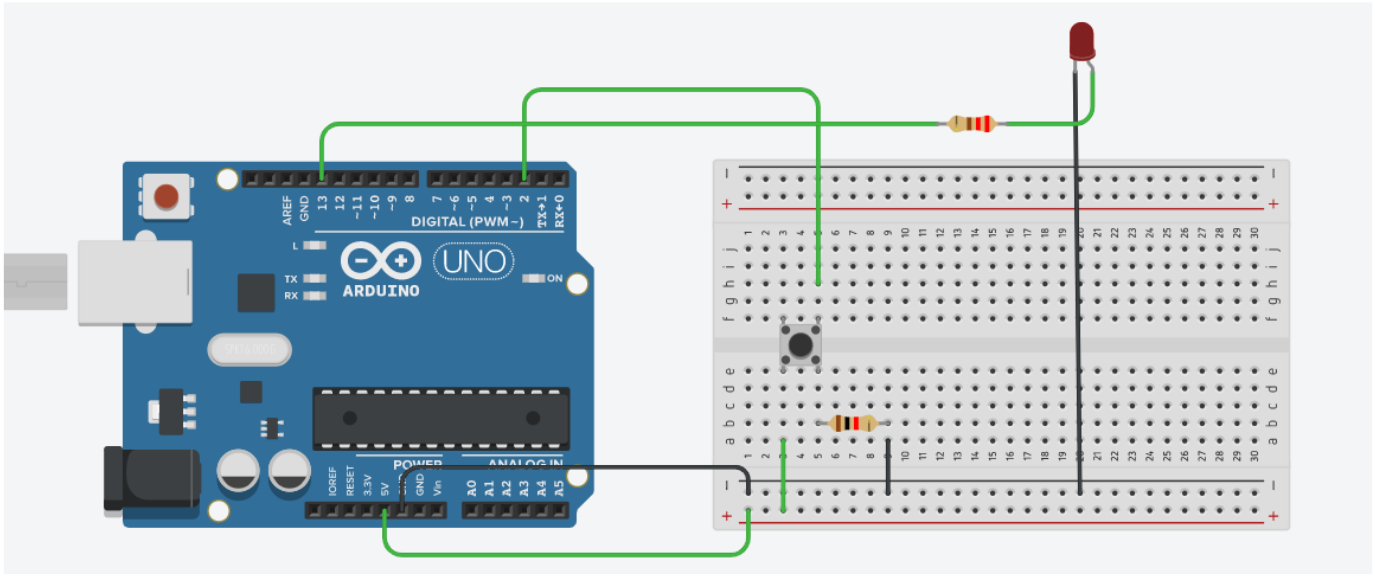
- «`setup`» and «`loop`» are two functions!
- «`void`» if nothing is returned
- A lot of functions have been already developed by programmers, you can find them in the function page: https://www.arduino.cc/reference/en/

- For example, `sqrt()` is the function that calculates the square root of a number.

Arduino programming

Exercise

Turn on/off a LED only when the button is pressed (not released).



Software

```
int buttonState = 0;
int oldButtonState = 0;
int internalState;
int pinLed = 13;
int pinButton = 2;

void setup()
{
  pinMode(pinButton, INPUT);
  pinMode(pinLed, OUTPUT);
  internalState = 0;
  Serial.begin(9600);
}
```

Software

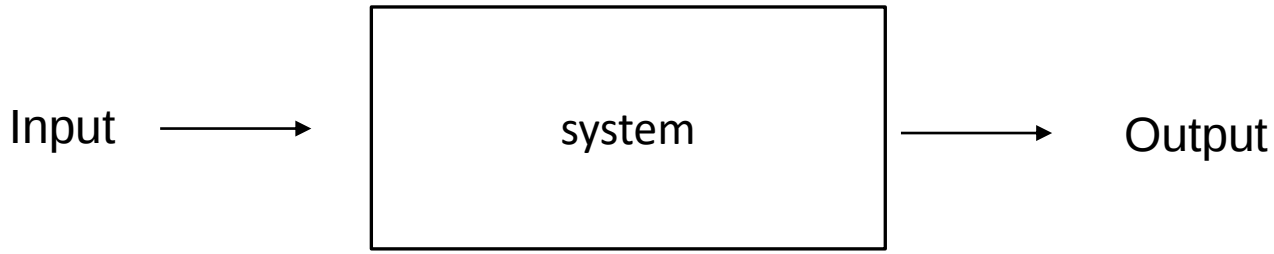
```
void loop()
{
  // read the input
  buttonState = digitalRead(pinButton);
  // update the internal state
  if (buttonState == HIGH && oldButtonState == LOW) {
      if (internalState==0)
        internalState = 1;
      else
         internalState = 0;
  }
```

Software

```
 if (internalState == 1) {
    // turn LED on
    digitalWrite(pinLed, HIGH);
    Serial.println("On");
  } else {
    // turn LED off
    digitalWrite(pinLed, LOW);
    Serial.println("Off");
  }
  delay(10); // Delay a little bit to improve simulation
performance
  oldButtonState=buttonState;
}
```

Combinatorial system

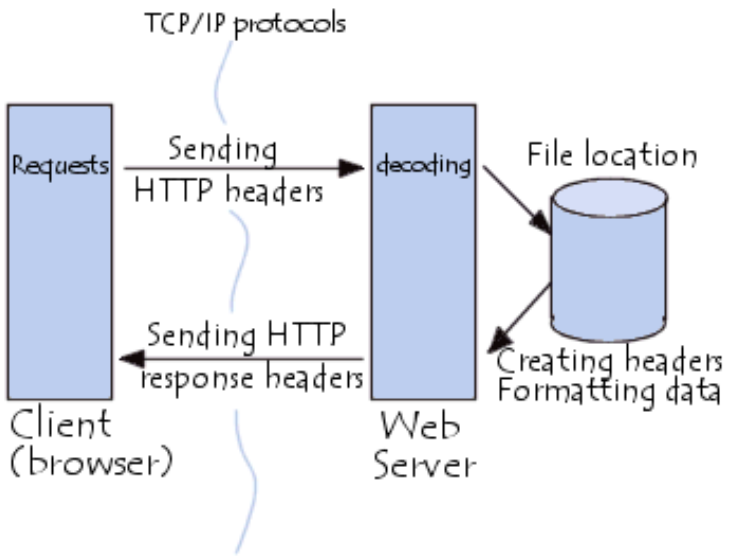
The output of a **combinatorial** system at a given time *t* depends on the input at time *t* only.



Combinatorial system

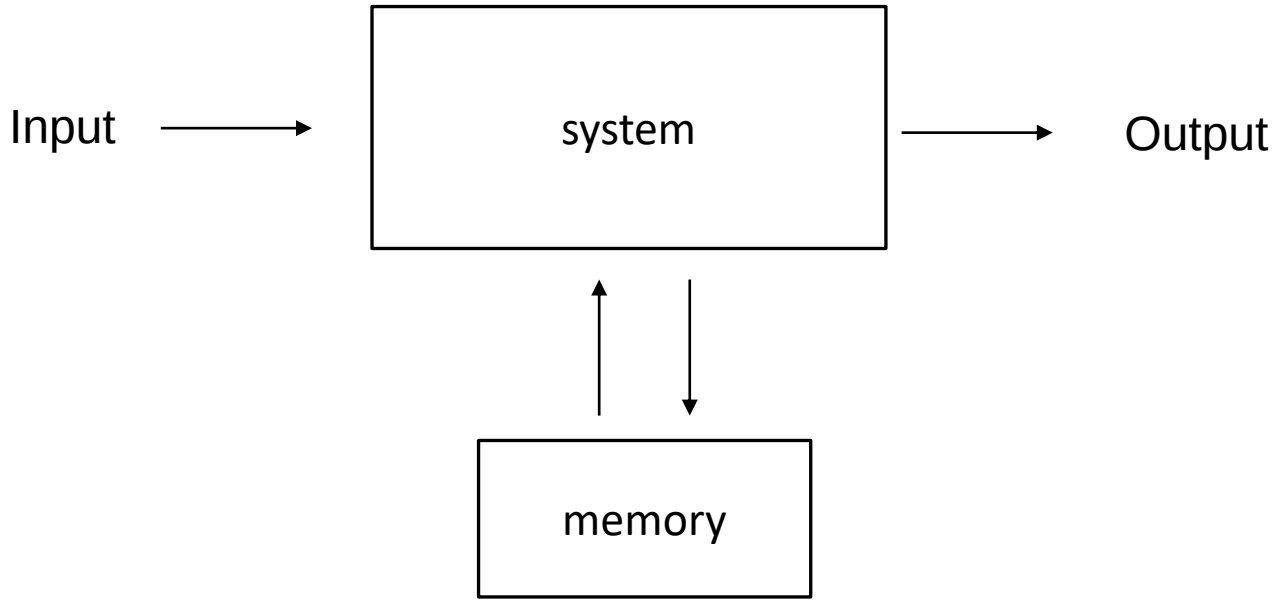
Examples of combinatorial systems:

- Fingerprint lock system (fingerprint → autentication)
- Key-based lock system (key is a combination)
- HTTP is stateless (IP → web page)
  - Cookies are exploited to make HTTP stateful



Sequential systems

The output of a **sequential** system at a given time *t* depends on the input at time *t* as well as the inputs from time 0 to *t*.



Differently from the previous case, sequential systems require a **memory (state)**.

Sequential systems

The state of a sequential system allows to discard all the input data from time 0 to *t*, since it contains a sort of summary of the past.

Given the input at time *t* and the state at time *t*, **it is possible to estimate the state at time *t*+1 as well as the output of the system**.

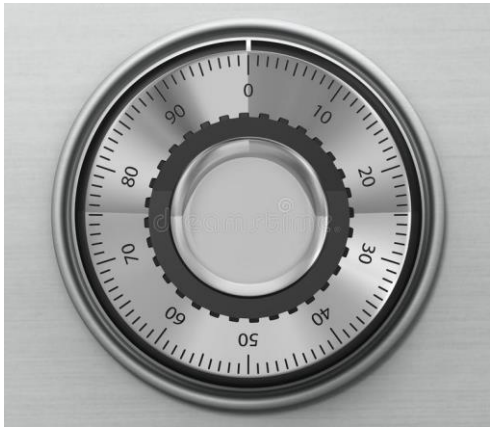
Digital sequential logic circuits are divided into:

- **Synchronous**: the state of the device changes only at discrete times in response to a clock signal;
- **Asynchronous**: the state of the device can change at any time in response to changing inputs;

Combinatorial systems

Examples:

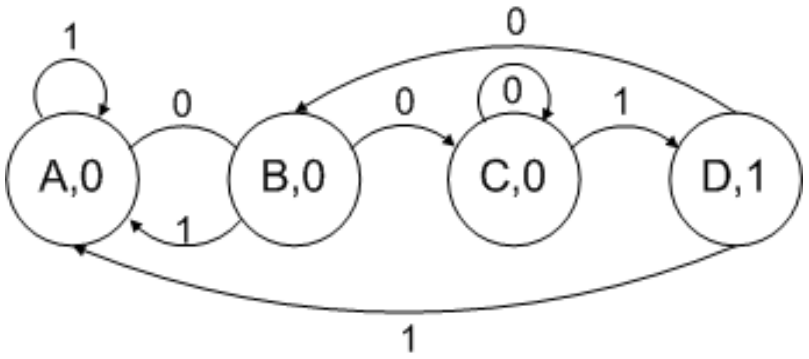
- Coin counter
- Button +/- of the TV volume remote control
- Safe combination



State diagram

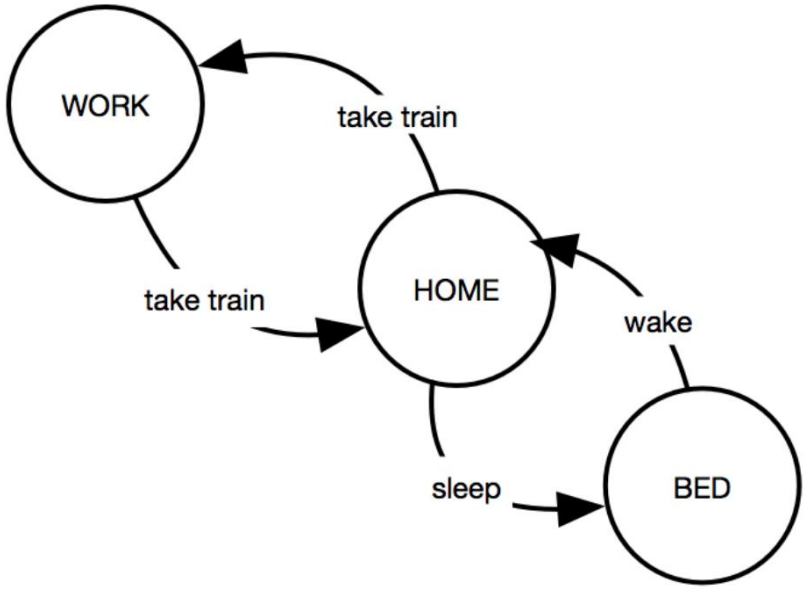
A state diagram is a representation of a sequential process:

- Each node contains a state $q \in Q = \{q_1 .. q_n\}$
- The input can be a symbol from the set $i \in I = \{i_1 .. i_m\}$
- The output can be a symbol from the set o $\in O = \{o_1 .. o_l\}$
- Each arch contains a transition, *i.e.* a specific case of the function



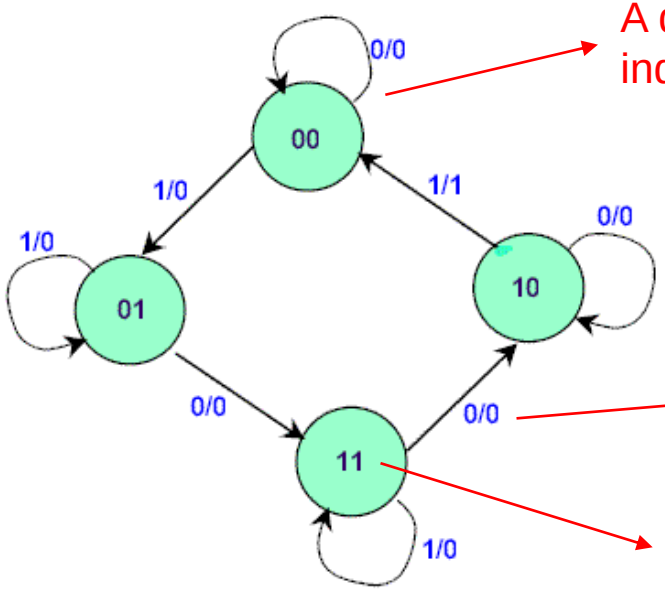
Finite State Machine

Starting from the definition of a *sequential* systems, we can define a Finite State Machine (FSM) as a sequential system that **can be in exactly one of a finite number of states at any given time**.



Mealy machine

A *Mealy* machine is a type of Finite State Machine whose output values are **determined both by its current state and the current inputs**.



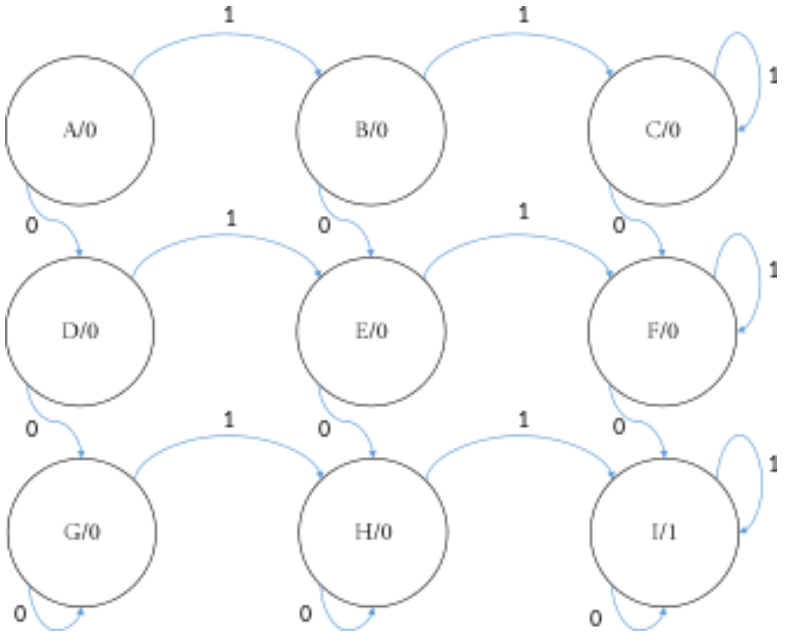
A directed line connecting a circle with itself indicates that no change of state occurs.

Input value | output value

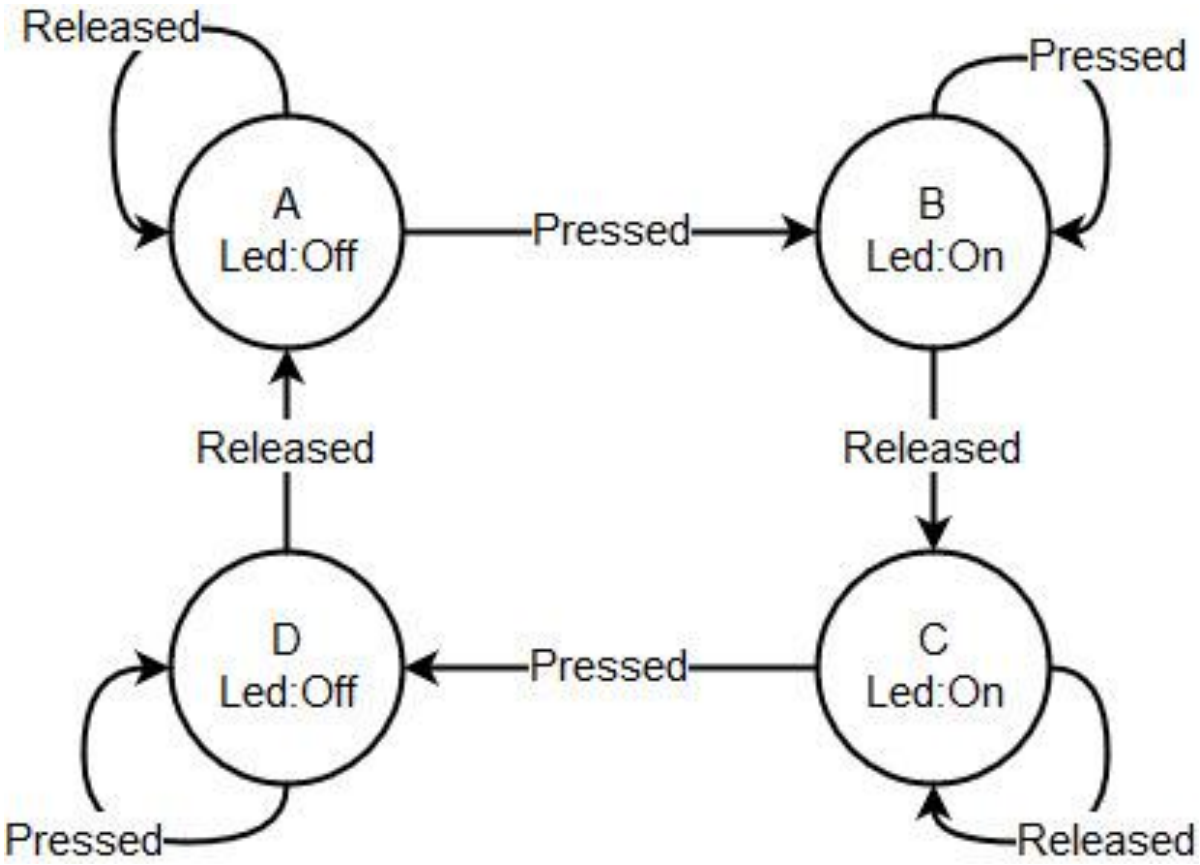
The binary number inside each circle identifies the state the circle represents.

Moore machine

In the theory of computation, a Moore machine is a finite-state machine whose output values are determined **only by its current state**.



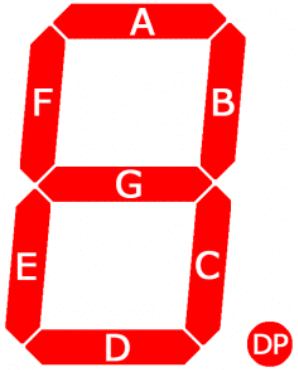
Example: pushbutton

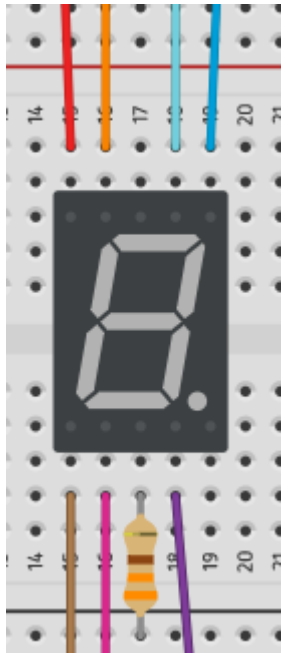


Exercises

1. Implement the «pushbutton» Finite State Machine.
   a) After 4 seconds of time delay turn off the led (use the `millis()` function)

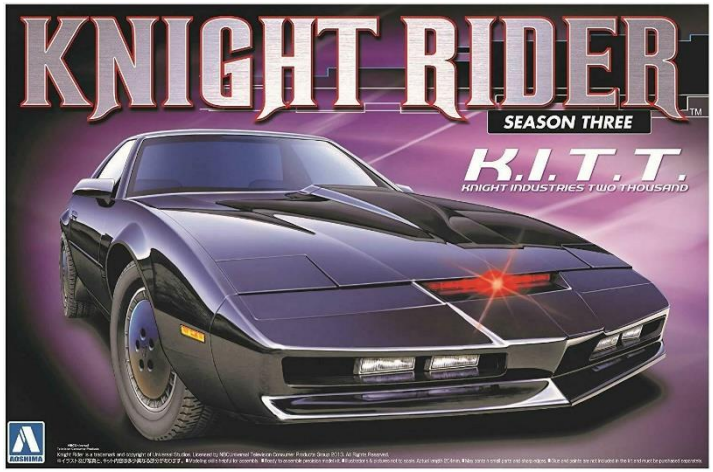
2. Implement a simple FSM that counts the number of times in which the button is pushed.
   a) Define a new function «`displayNumber()`»
   b) Use the `switch()` statement
   c) Display the counter on a 7 segments display





Exercises

3. Implement the red light of the



4. Turn on a LED. Its light intensity is inversely proportional to the light in the environment (use a light sensors!).

https://www.youtube.com/watch?v=oNyXYPhnUIs&ab_channel=NBCClassics