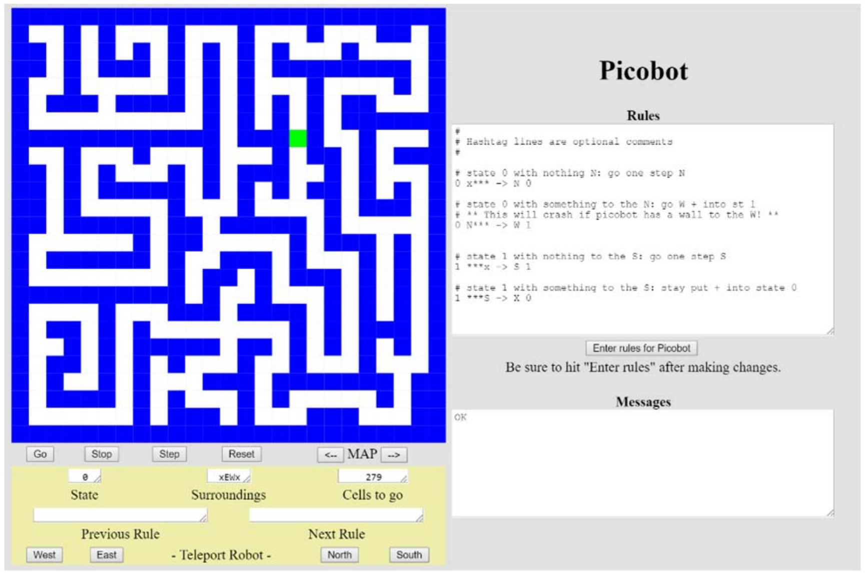
```
While True:
    ...
      if ( ):
           do1()
      elif ( ):
           do2()
      ...
```

```
...

do1()
do2()

...
```

Finite State Machines (FSM)

Picobot



Picobot

Rules

```
#
# Hashtag lines are optional comments
#

# state 0 with nothing N: go one step N
0 x*** -> N 0

# state 0 with something to the N: go W + into st 1
# ** This will crash if picobot has a wall to the W! **
0 N*** -> W 1


# state 1 with nothing to the S: go one step S
1 ***x -> S 1

# state 1 with something to the S: stay put + into state 0
1 ***S -> X 0
```

Enter rules for Picobot

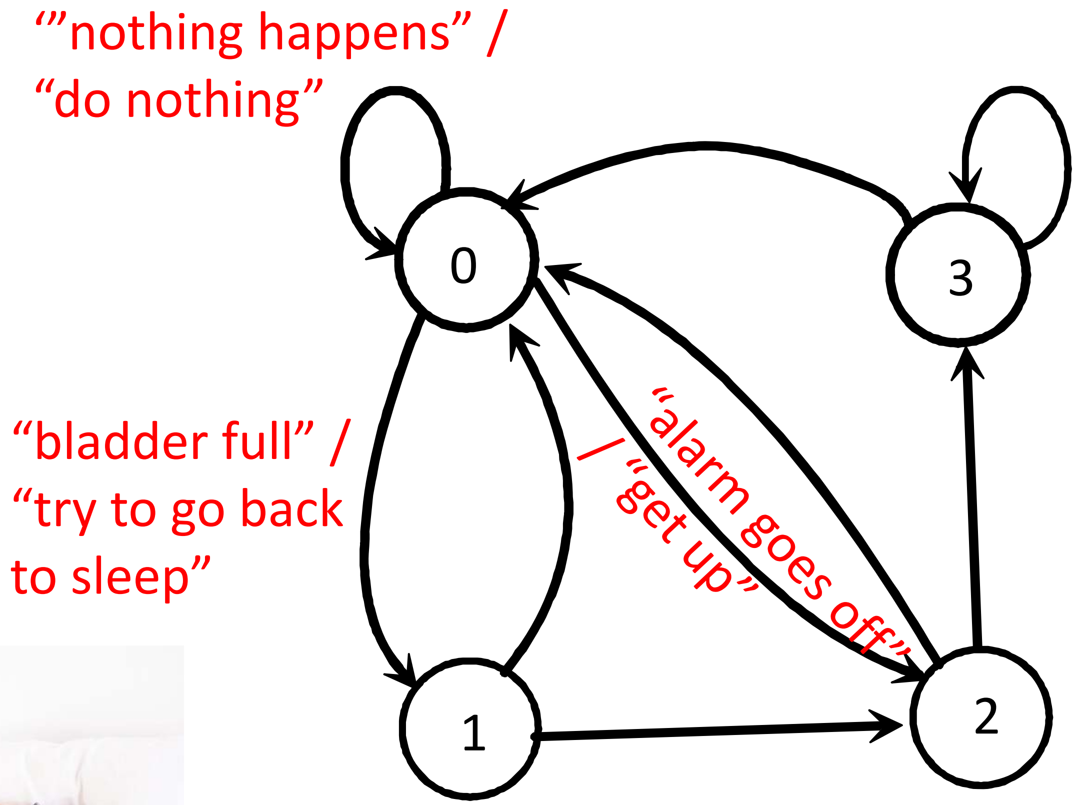
Be sure to hit "Enter rules" after making changes.

Messages

```
OK
```

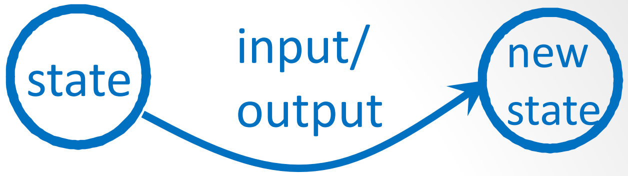
state	surroundings		direction	new state
0	xx**WS**	$\longrightarrow$	N	0

Finite State Machine (FSM)

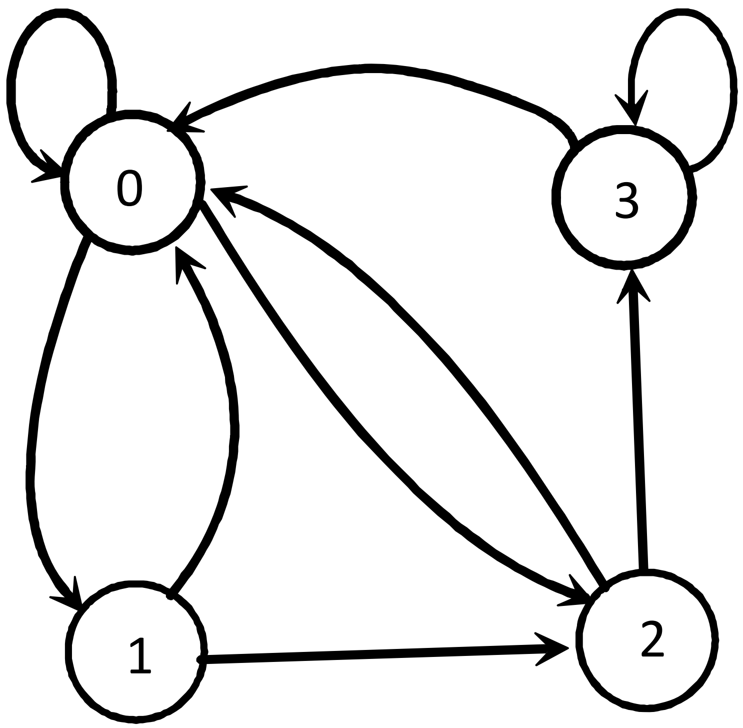

“”nothing happens” / “do nothing”
0
3
“bladder full” / “try to go back to sleep”
“alarm goes off” / “get up”
1
2

State 0: “in bed, asleep”
State 1: “in bed, awake”
State 2: “in the bedroom, out of bed”

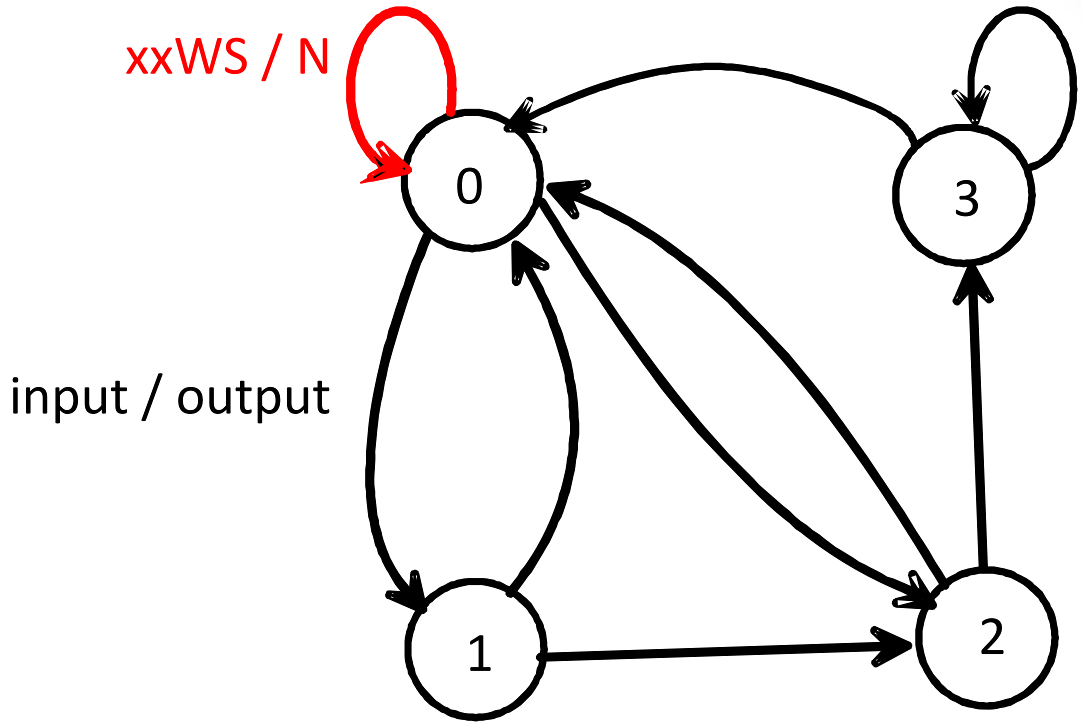
Finite State Machine (FSM)



Mealy FSM



state input $\longrightarrow$ output new state



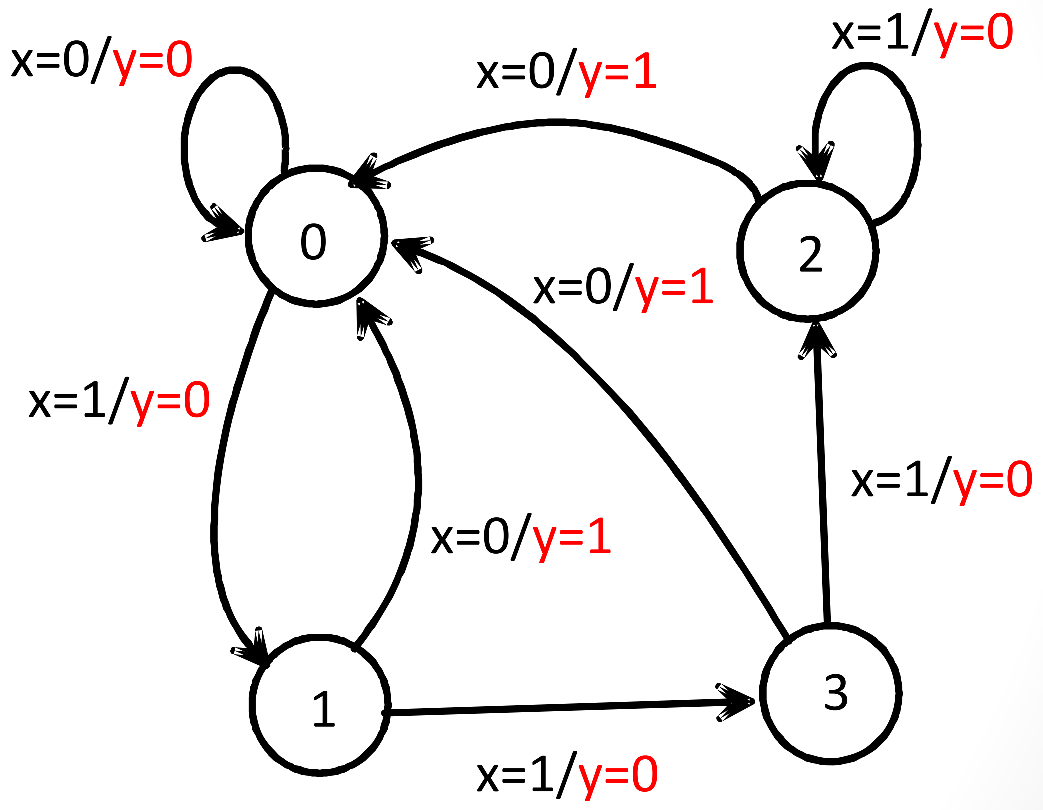
state	surroundings		direction	new state
0	xx**WS**	$\longrightarrow$	N	0

Finite State Machine (FSM)

State: s

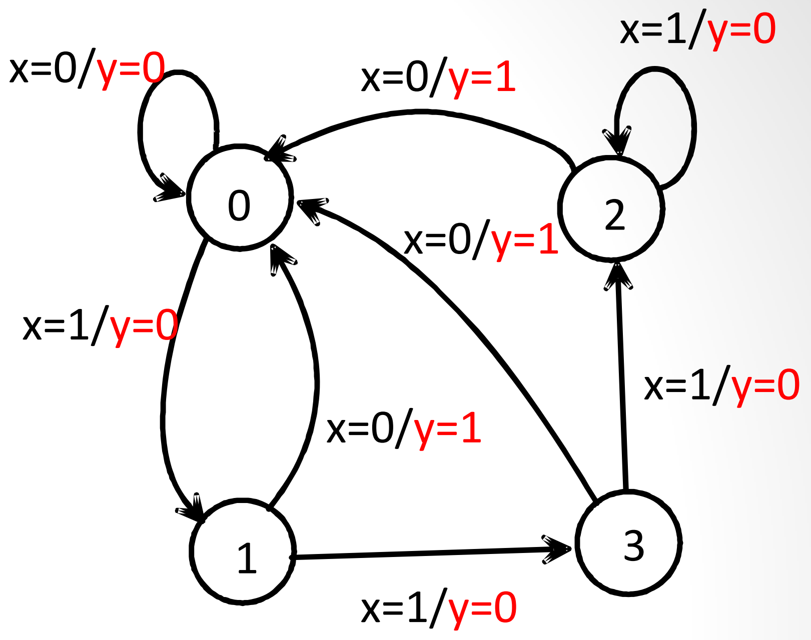
Input: x

Output: y



Mealy FSM

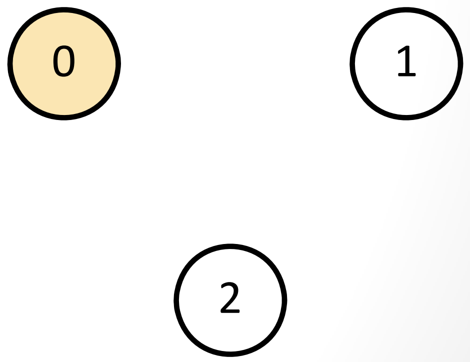
```
if (state == 0):
  if (x == 0):
      y = 0
      next_state = 0
  elif (x == 1):
      y = 0
      next_state = 1

elif (state == 1):
  if (x == 0):
      y = 1
      next_state = 0
  elif (x == 1):
      y = 0
      next_state = 3

…

state = next_state
```



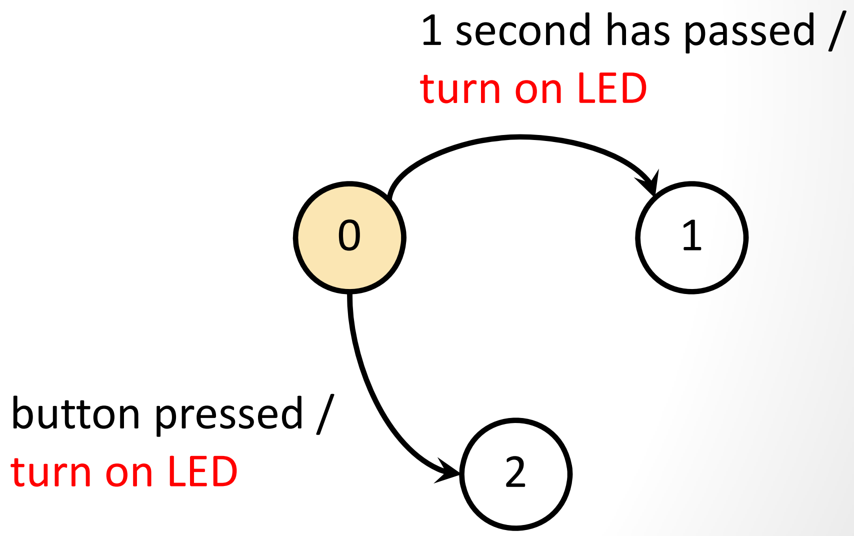
Mealy FSM

Stay in state 0 (with the LED off) for 1 second and then move to state 1 while turning on the LED. However, if at any time during that 1 second wait, the button is pressed, immediately go to state 2 and also turn on the LED.

Complete the FSM …

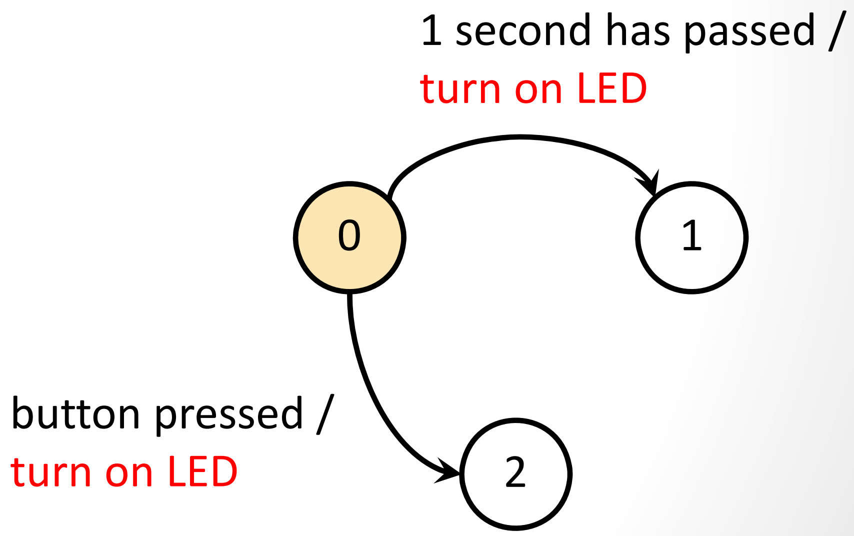


Stay in state 0 (with the LED off) for 1 second and then move to state 1 while turning on the LED. However, if at any time during that 1 second wait, the button is pressed, immediately go to state 2 and also turn on the LED.



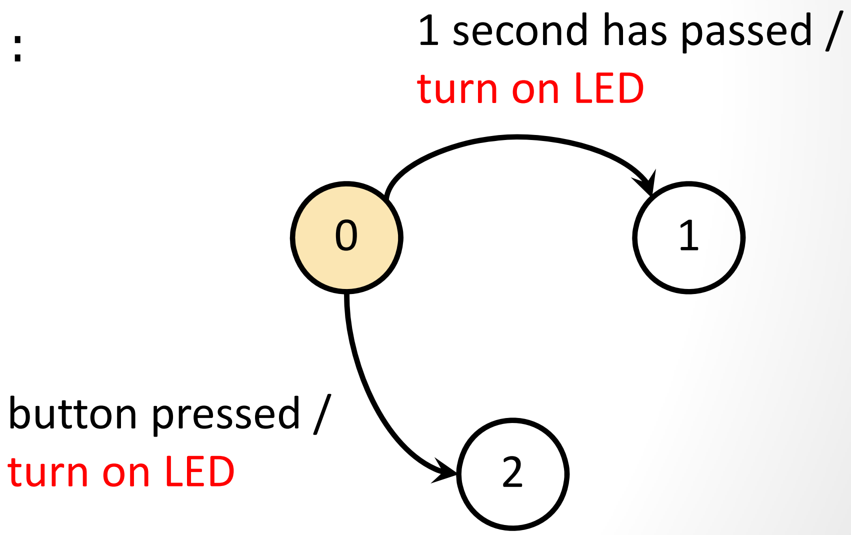
Stay in state 0 (with the LED off) for 1 second and then move to state 1 while turning on the LED. However, if at any time during that 1 second wait, the button is pressed, immediately go to state 2 and also turn on the LED.

```
if (state == 0):
  if (x == 0):
       y = 0
       next_state = 0
  elif (x == 1):
       y = 0
       next_state = 1


…

state = next_state
```



Stay in state 0 (with the LED off) for 1 second and then move to state 1 while turning on the LED. However, if at any time during that 1 second wait, the button is pressed, immediately go to state 2 and also turn on the LED.

```
if (state == 0):
  time.sleep(1)
  if (read_button() == PRESSED):
       turn_on_LED()
       next_state = 2
  else:
       turn_on_LED()
       next_state = 1


…

state = next_state
```



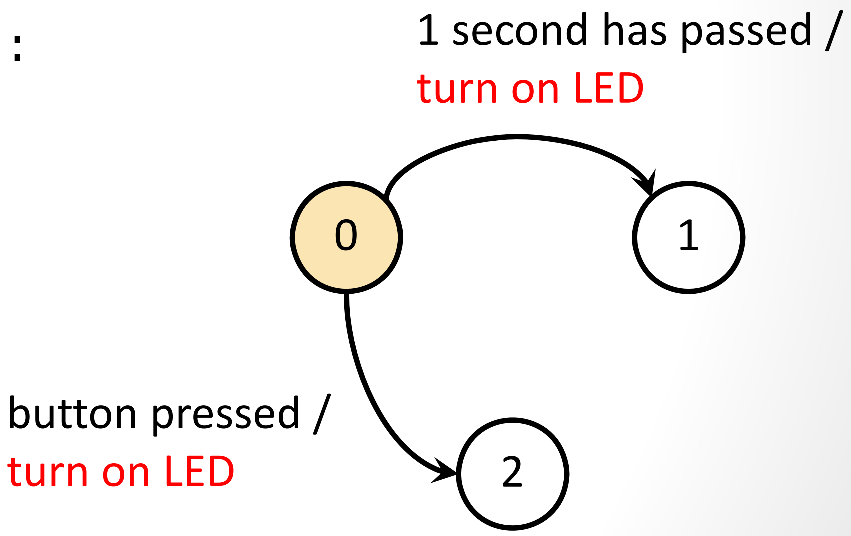
Does this implement the desired functionality?

A. Yes B. No

Stay in state 0 (with the LED off) for 1 second and then move to state 1 while turning on the LED. However, if at any time during that 1 second wait, the button is pressed, immediately go to state 2 and also turn on the LED.

This code is blocking!

```
if (state == 0):
  time.sleep(1)
  if (read_button() == PRESSED):
       turn_on_LED()
       next_state = 2
  else:
       turn_on_LED()
       next_state = 1


…

state = next_state
```

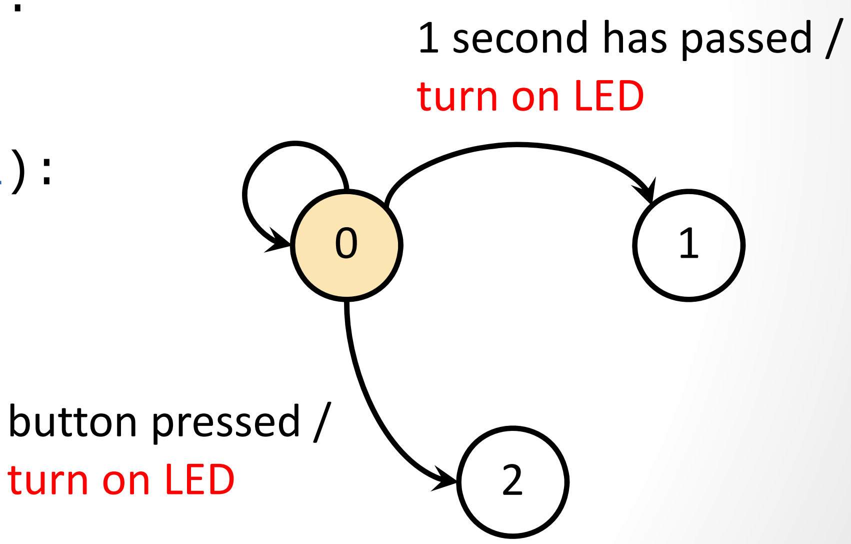


How can we implement the desired functionality?

Stay in state 0 (with the LED off) for 1 second and then move to state 1 while turning on the LED. However, if at any time during that 1 second wait, the button is pressed, immediately go to state 2 and also turn on the LED.

Go through FSM as fast as possible …

```
if (state == 0):
  if (read_button() == PRESSED):
       turn_on_LED()
       next_state = 2
  elif (now-last_transition > 1):
       turn_on_LED()
       next_state = 1
  else:
      next_state = 0

…

state = next_state
```


1 second has passed /
turn on LED
0
1
2
button pressed /
turn on LED