Traffic light system

1. **Functional requirements**

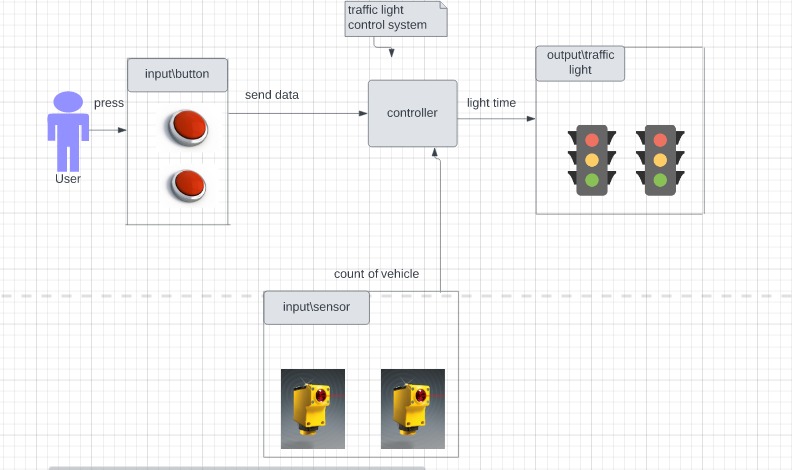
|  |  |
| --- | --- |
| **Requirement ID** | **Requirement Statement** |
| FR001 | 1- automatically switch between red , green and yellow states in traffic light . |
| FR002 | 2- sensor return a Boolean value to the traffic system to identifier if the crowded or not . |
| FR003 | 3- Website shall have a home page that list the types of traffic system . |
| FR004 | 4- if the sensor return that the way is crowded in some way it will add (bonus seconds) to the green time of this way once. |
| FR005 | 5- there is 8 buttons in our traffic system must used to help pedestrian to cross the way without having any accident risk |
| FR006 | 6- whenever every button is pressed it has a list of tasks to do |
| FR007 | 7- cars should stop if traffic is red |
| FR008 | 8- cars can not continue to the right hand turn because it may cause an accident , only it can continue if green and can turn left if arrow is enabled |

## System Feature 2 (and so on)

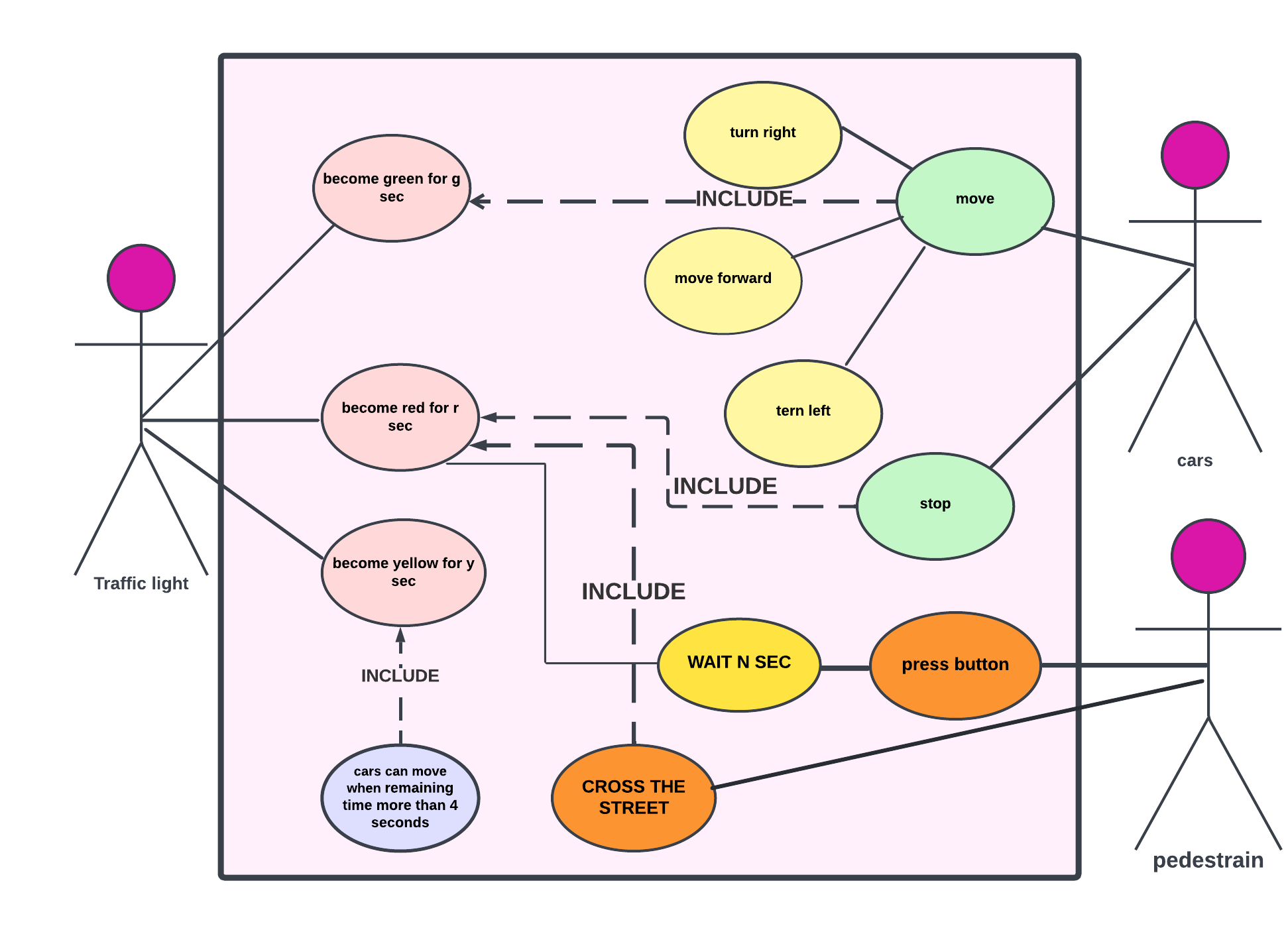
1. **Other Nonfunctional Requirements**

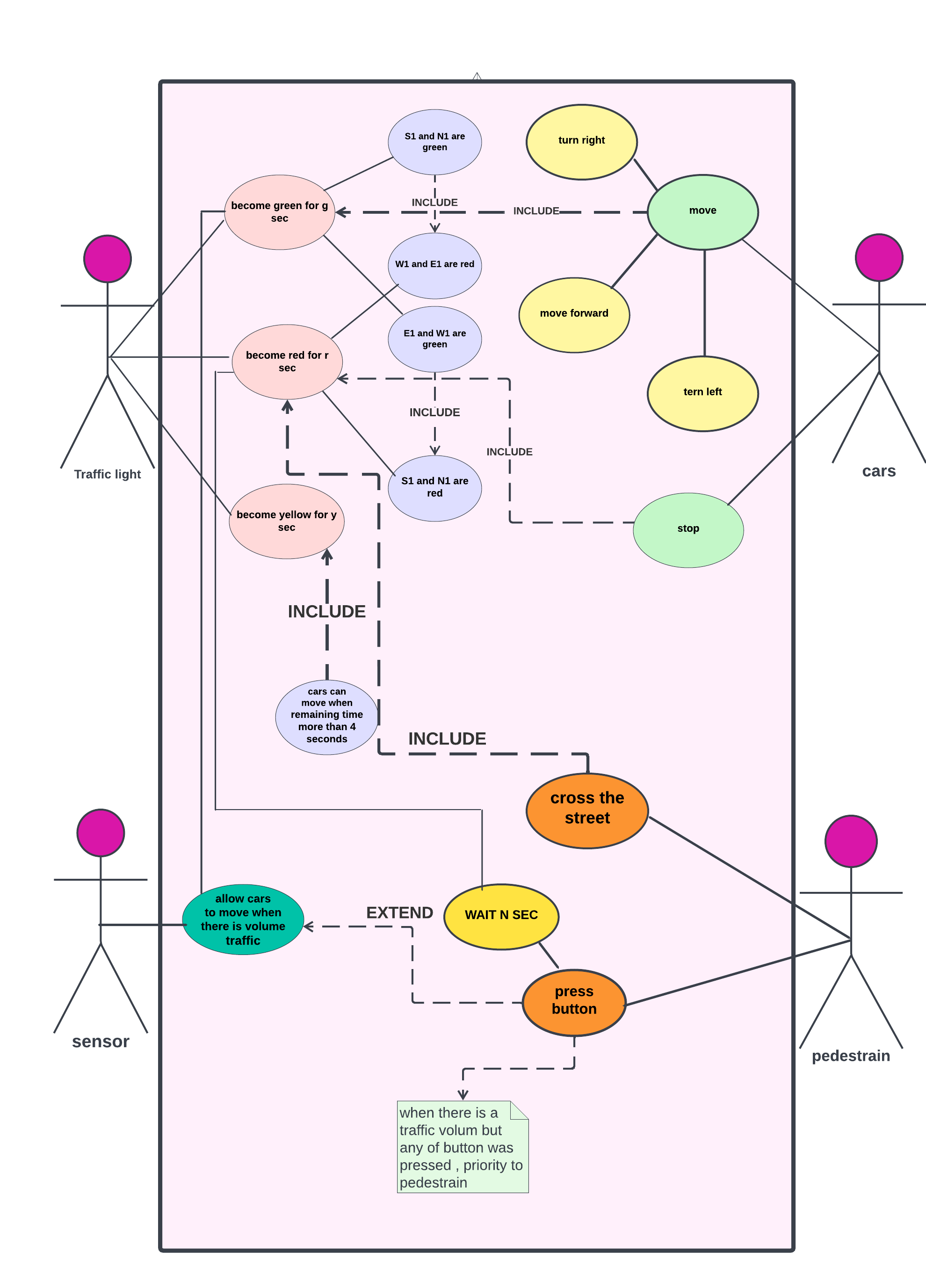
|  |  |
| --- | --- |
| 1-Performance | 1-The system must respond to the button once it pressed. |
| 2-Safety and Security Requirement | 1- cars can not continue while the traffic is red  2- cars can not turn left while arrow is disabled  3- cars can not turn right  4- firebase is read only  5- pedestrian can not cross the while the traffic is green |
| 3-Reliabilty | 1-The program should provide catching of exceptions to avoid system crashes or data validation failures. |
| 4-Usability | 1-The system has to be easy to be used.  2- button is disabled for n seconds after it pressed  3- pedestrian can not press the button while it disabled  4- pedestrian must wait after press the button for w seconds and cross the way |
| 5-Availability | 1-The system should be available 24 hours 7 days a week. |

**System architicture**

****

**Use case Diagram**

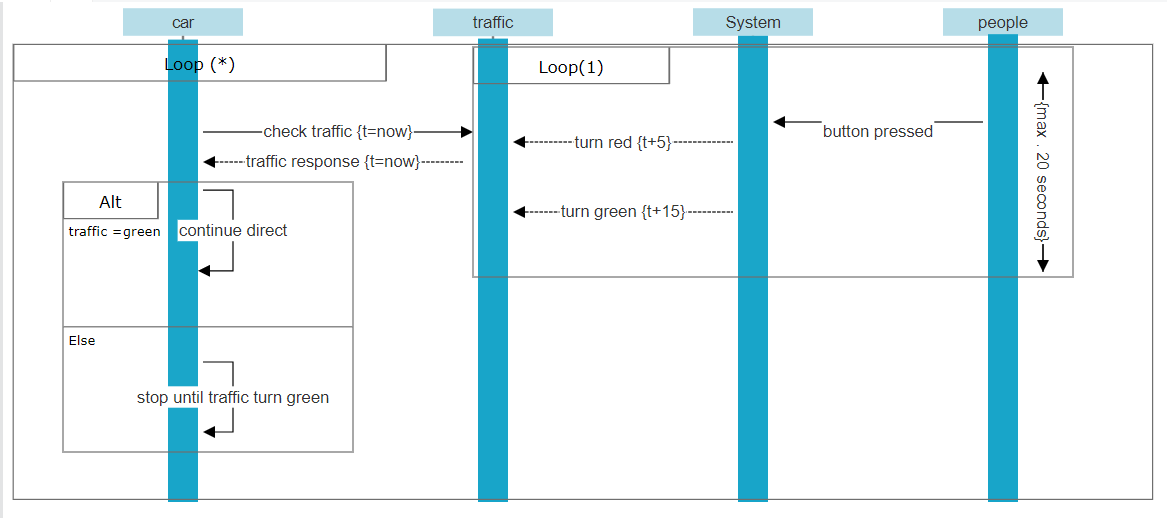




****

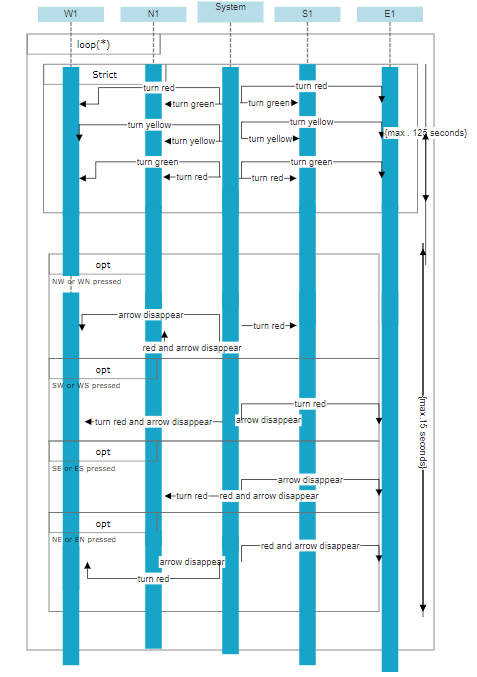
**Sequence Diagram**

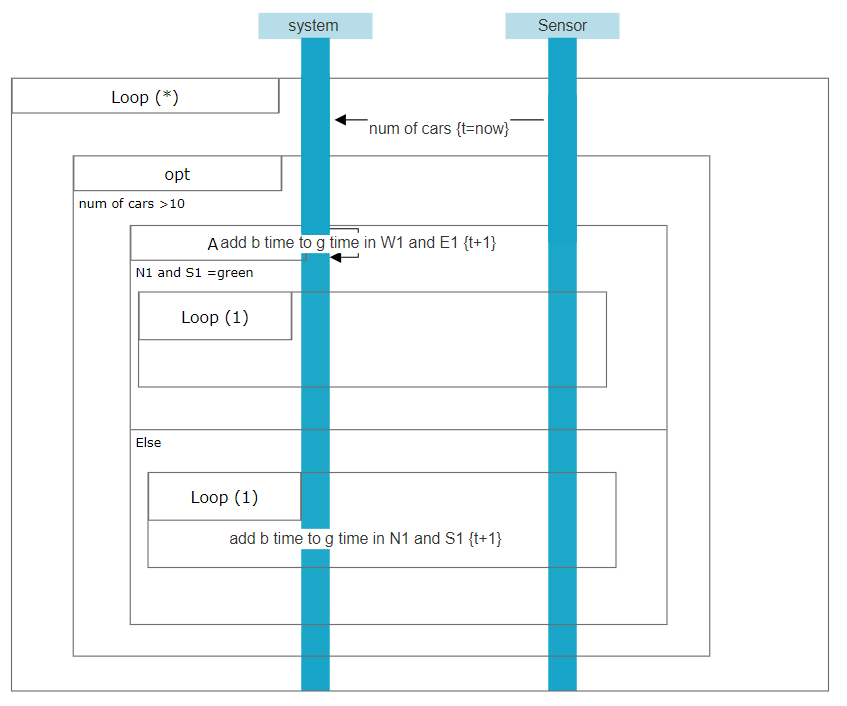
**One way**

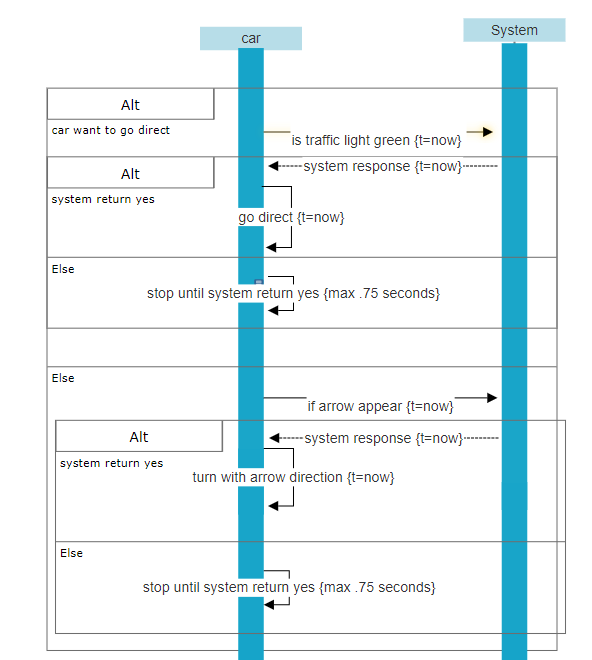
****

**Sequence Diagrams**

**Two Way**

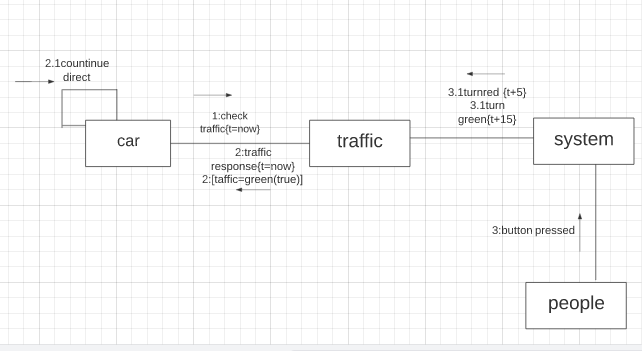
****

****

****

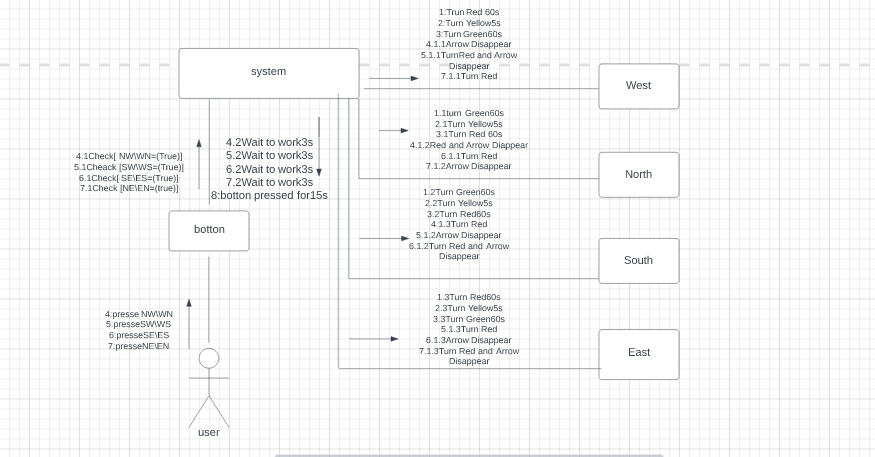
**Colaboration Diagram**

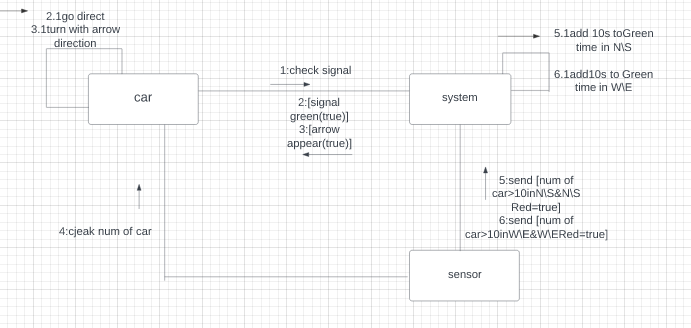
**One way**

****

**Colaboration Diagram**

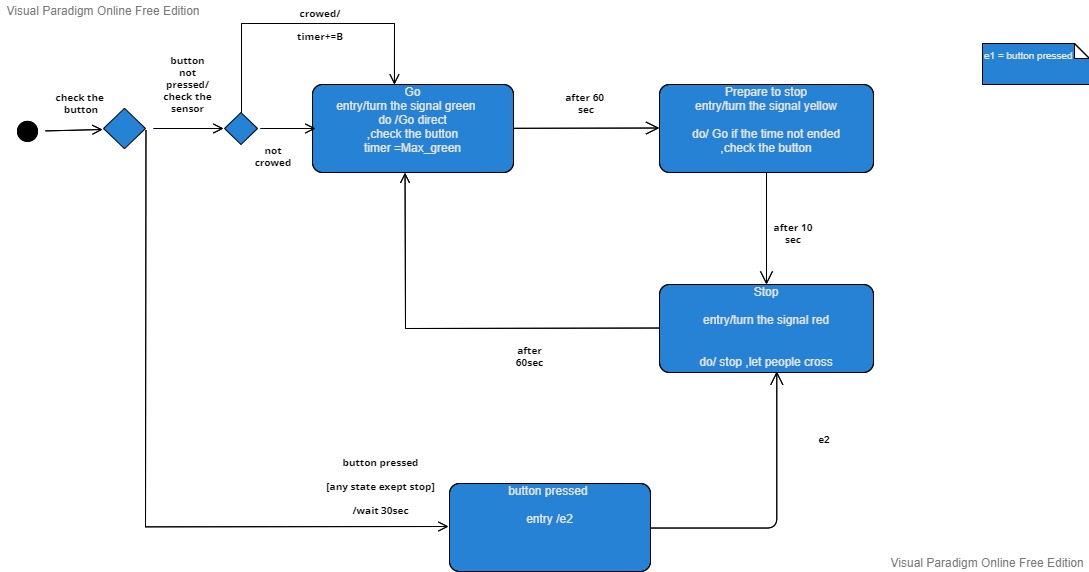
**Two Way**

****

****

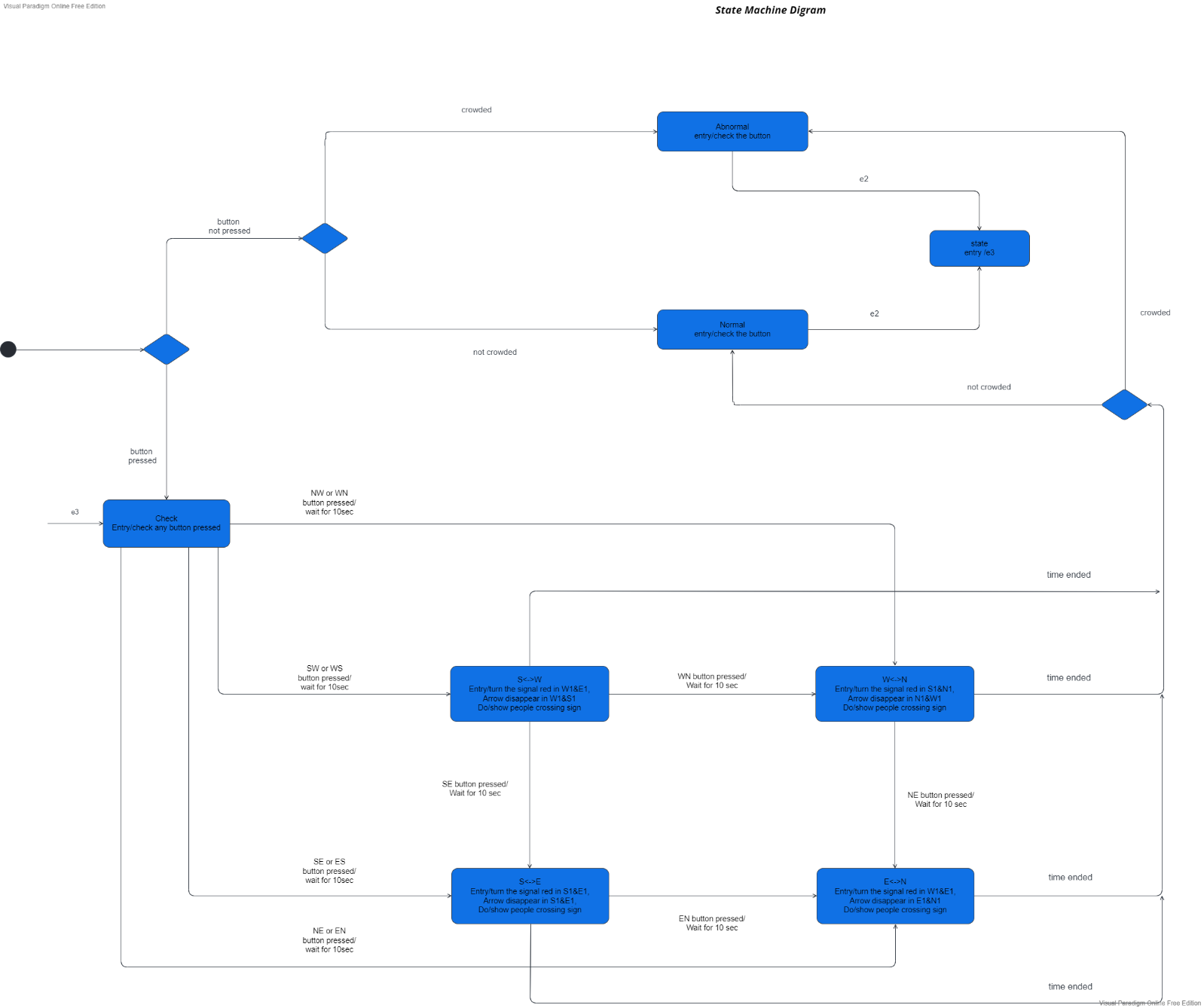
**State-Machine Diagram**

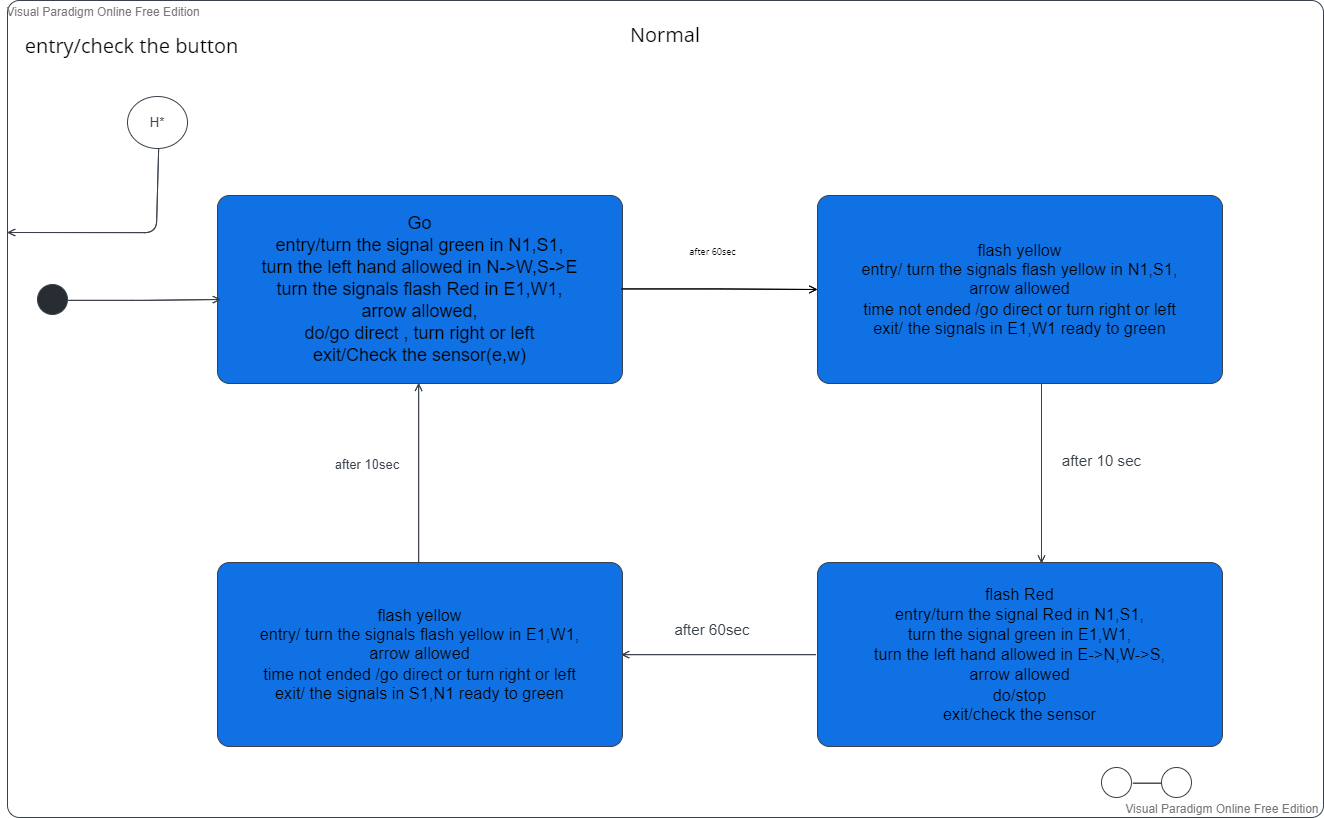
**One way**

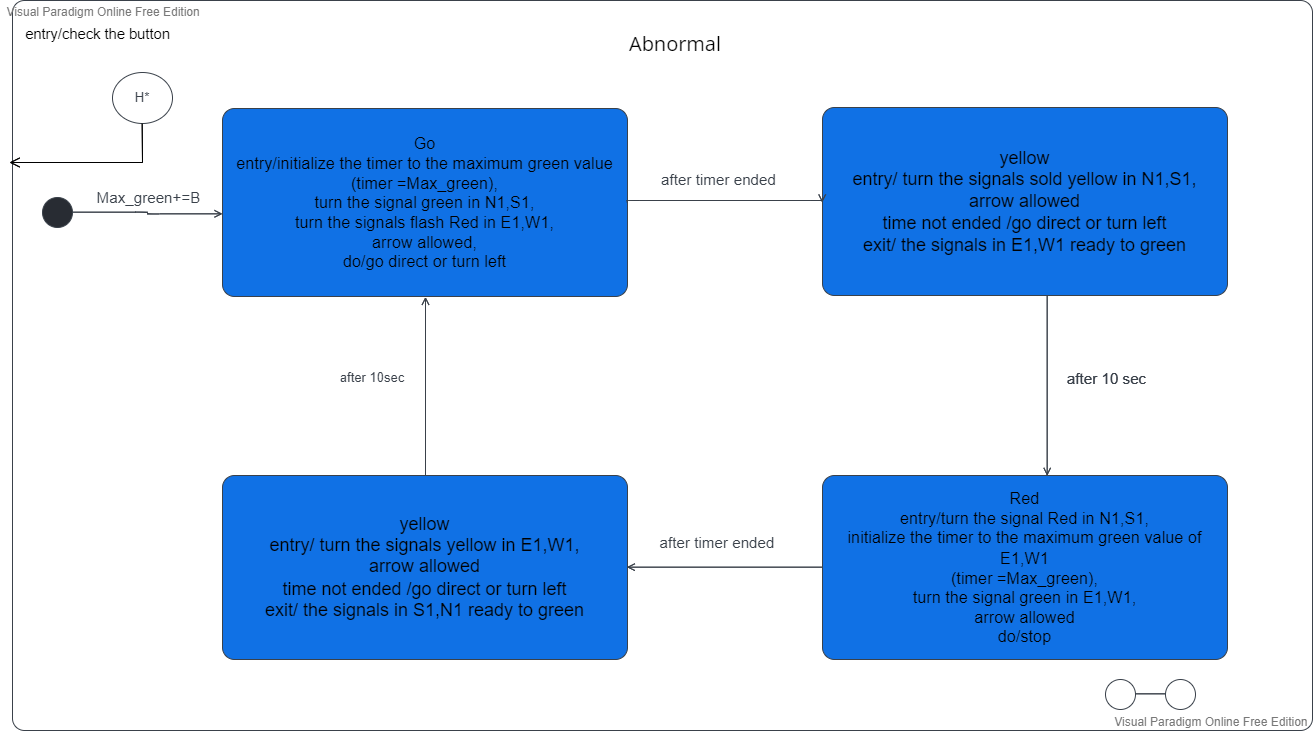


**State-Machine Diagrams**

**Two Way**

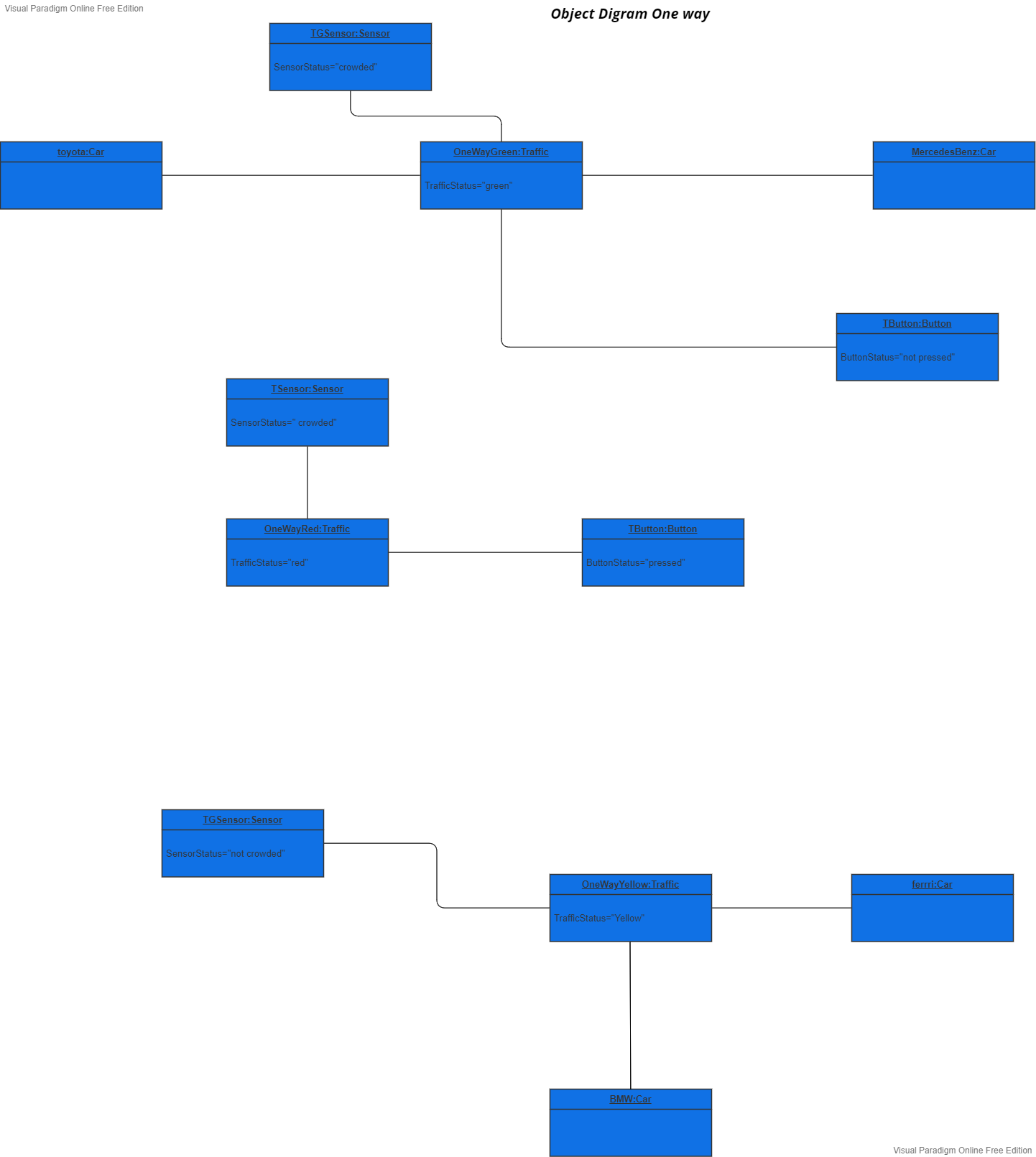
****

****

****

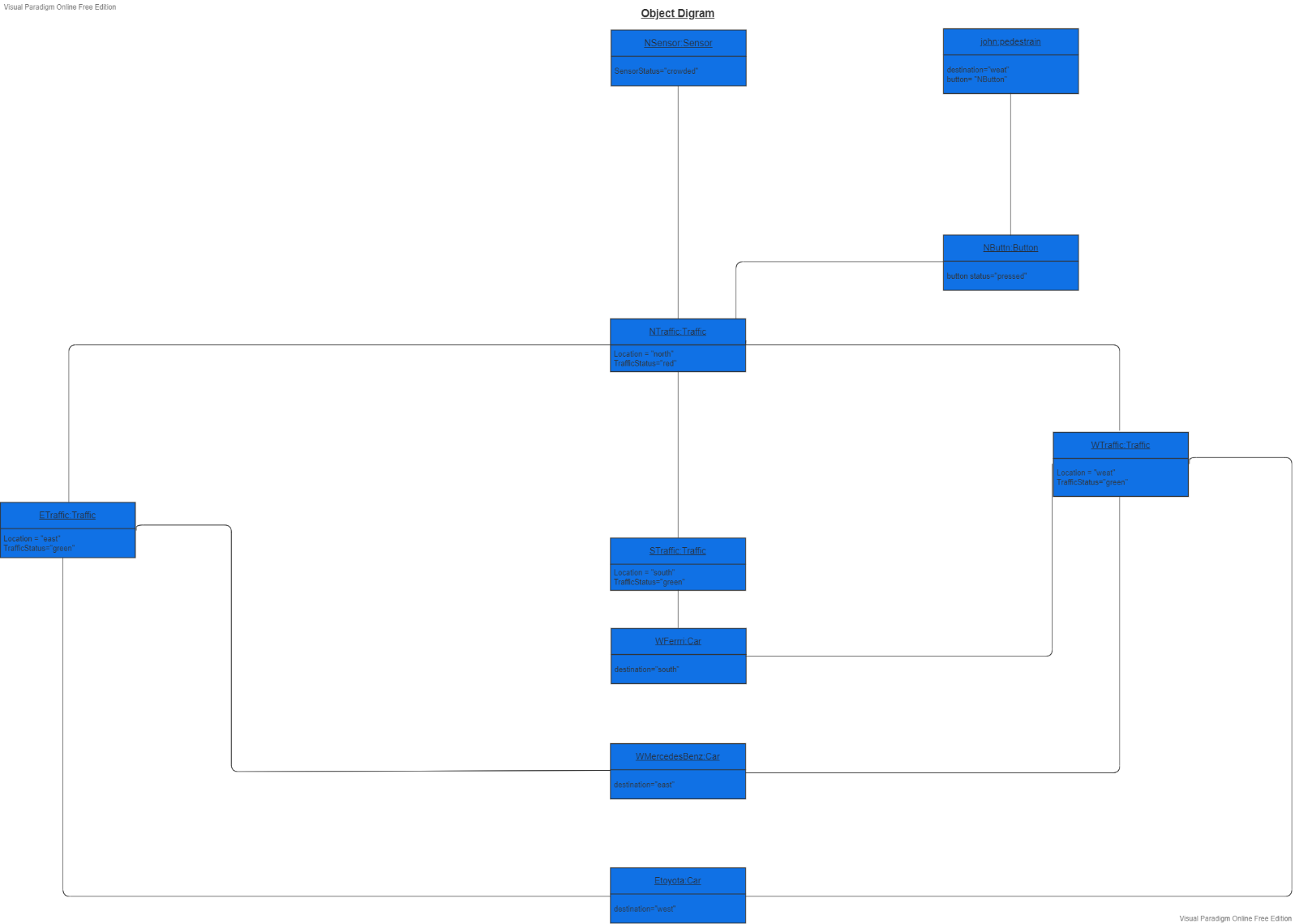
**Object Diagram**

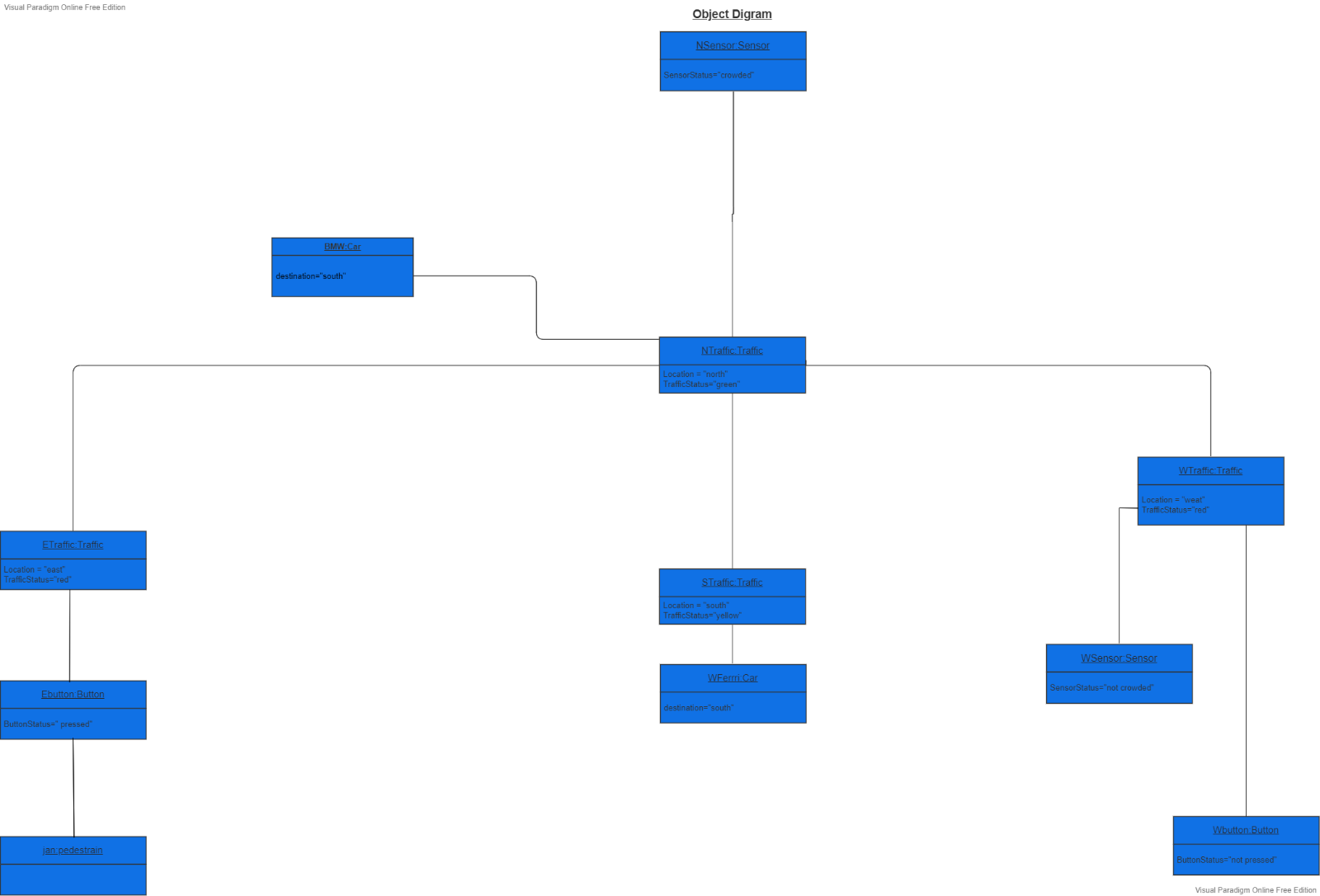
**One way**

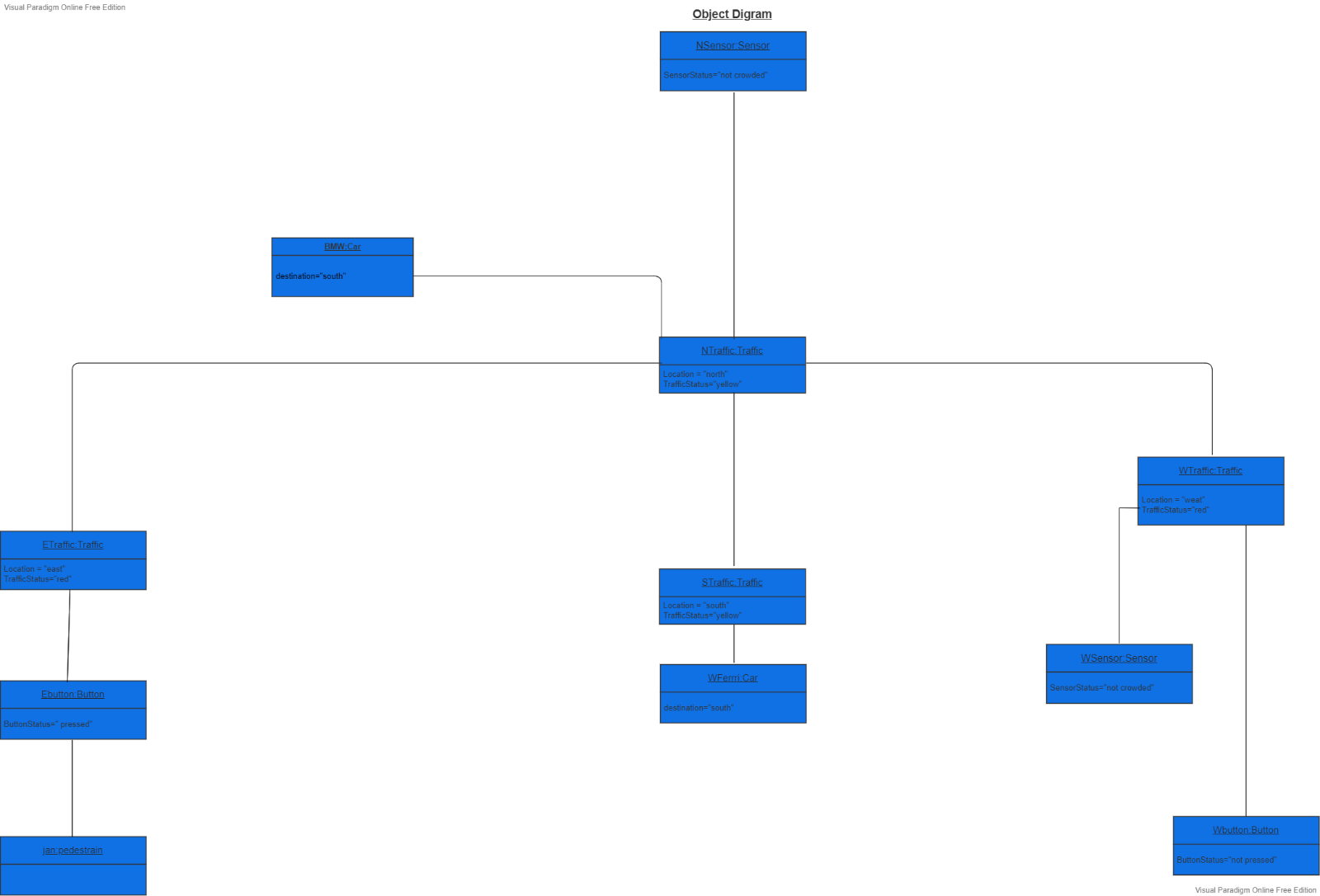
****

**Object Diagrams**

**Two Way**

****

****

****

**2 Design Patterns Applied**

**Delegation design pattern**

**Used on :**

button and traffic light state

**Context :**

designing method in class and realize that another class has a method which provides the required service

**How to implement:**

the delegating method (change state ) in delegator class ( button) calls methods in the delegate class ( traffic light state )to perform the required task .

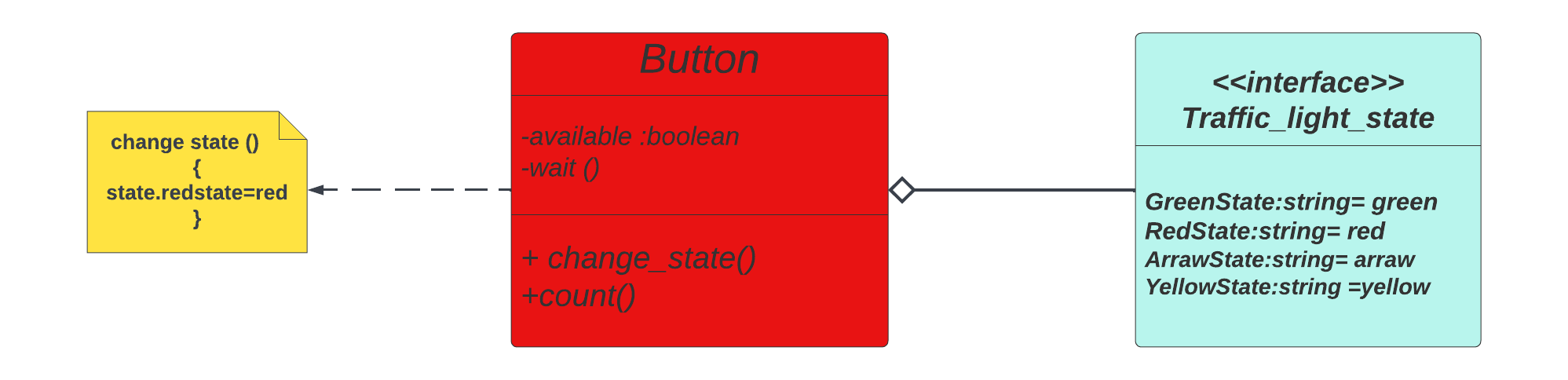
an association must exist between the delegator ( button ) and delegate class (traffic light state )

**Anti pattern :**

1. Overuse generalization and inherit the method that’s to be reused
2. Duplication of chunks code

**Forces :**

Minimizing developing cost by reusing methods



**Behavioural ( observer /subject ) design pattern**

**Used on :**

sensor and traffic

**Context :**

When partitioning a system into individual classes you want the coupling between then to be loose so you have the flexibility to vary them independently

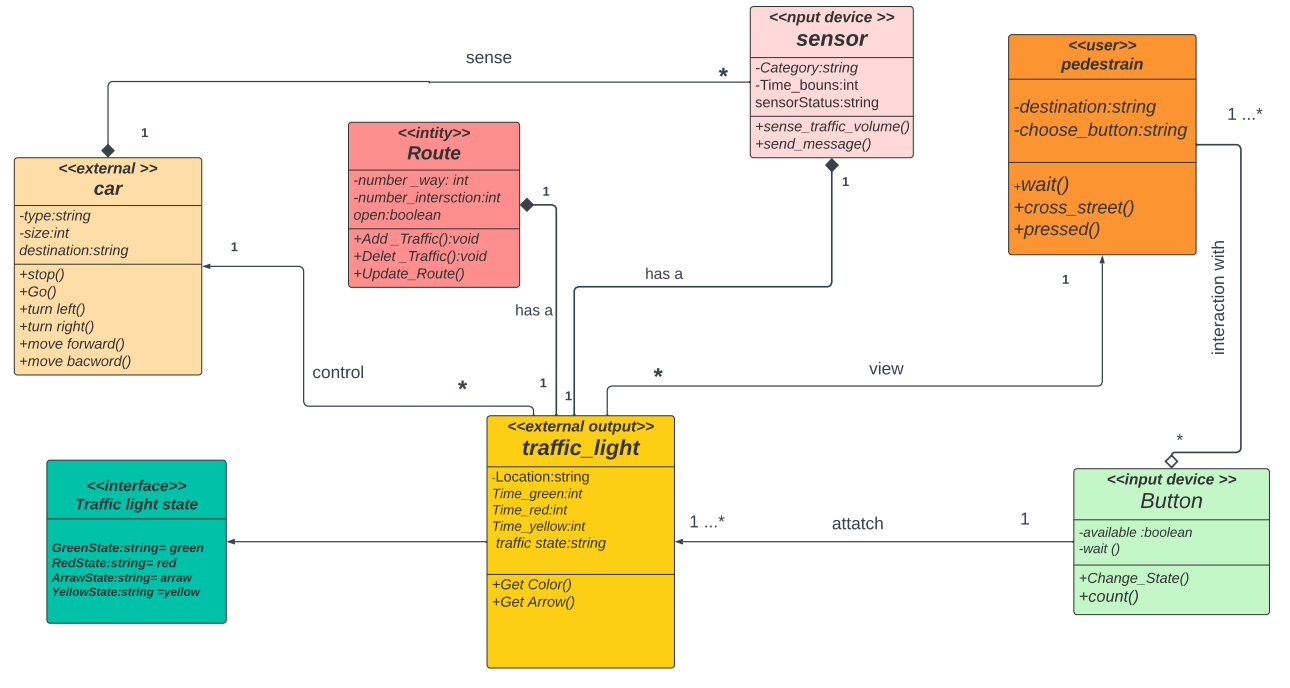
**How to implement :**

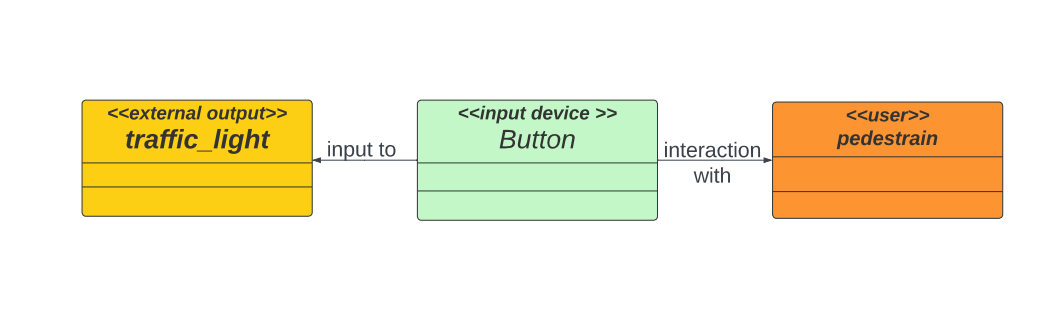
One object has the role of the subject (sensor) and one or more other objects the role of observers(traffic light) . The observers register themselves with the subject, if the state of the subject changes( sensor sense the traffic volume the observers traffic light) are notified by detailed information and can update themselves (change their state).

**Forces :**

sensor and traffic have to kept in step with one another without too tightly coupled

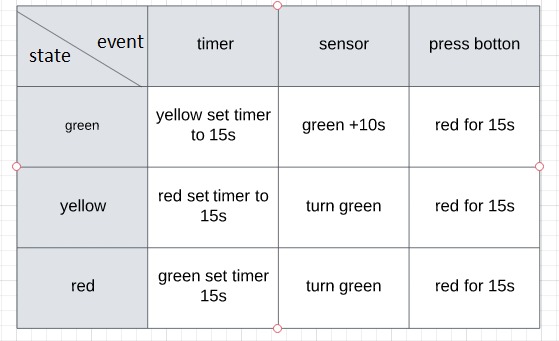
**Detailed Class diagram**

****

****

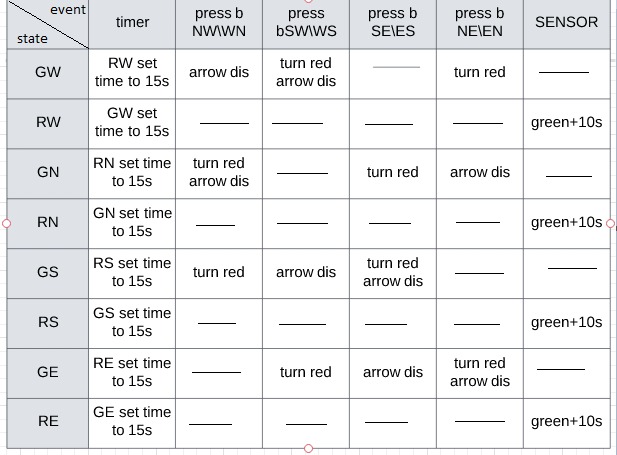
**State Transition Table**

**One Way**

****

**State Transition Table**

**Two Way**

****

**Activity diagram Diagram**

**One Way**

**https://cloud.smartdraw.com/editor.aspx?credID=-37874682&depoId=33707998&flags=128**

**Activity diagram Diagram**

**Two Way**

**https://cloud.smartdraw.com/share.aspx/?pubDocShare=AAD84C3166C90C824BF82F64582632EB3EC**