Embedded Software Engineering

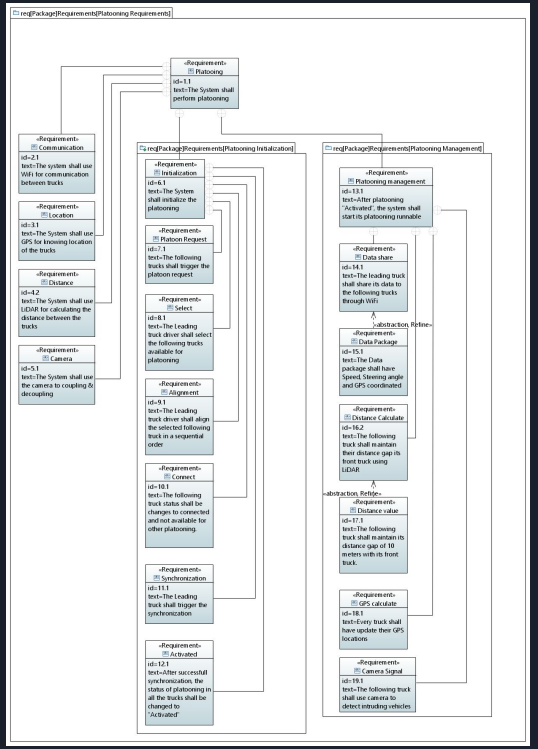
Truck Platooning

Team Outbreak

Namık Mert Tunçbilek - 7213712
Nijat Dashdamirov - 7213892
Mirudhubashini Ramasamy Sridharan - 7213893
Veli Ates - 7213717
Vignesh Arumugam - 7213710

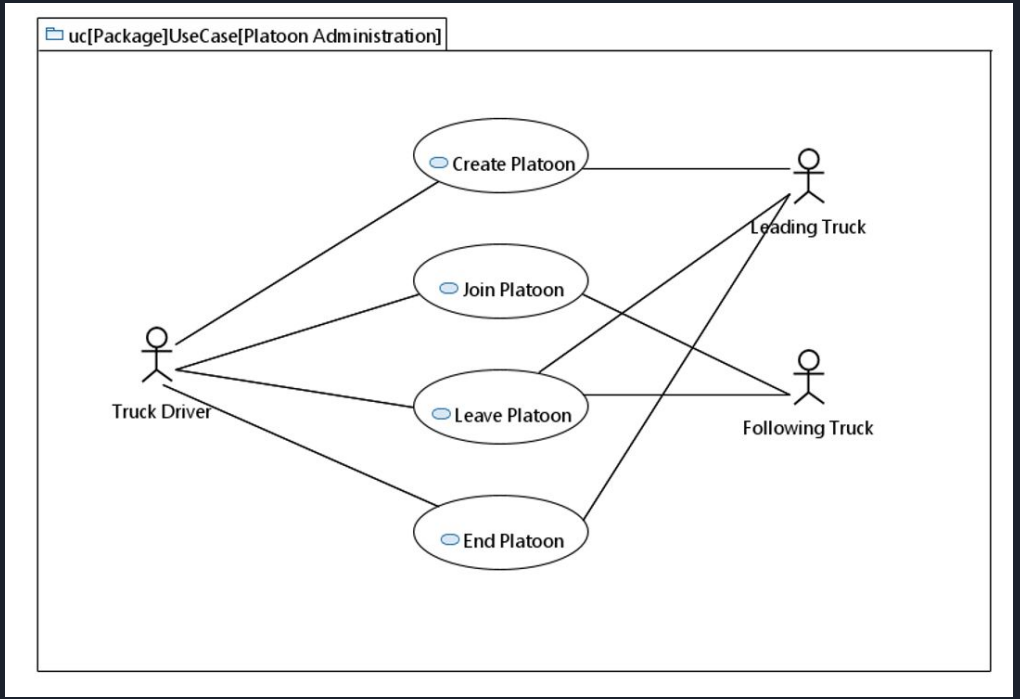
Requirement Diagram

- Requirement Elicitation and analysis where done before starting the requirement diagram.
- Requirements are defined by using blocks.
- Then the composite requirements are deconstructed by containment.
- Refinements are done for required blocks.
- Each requirement block is created with unique ID.
- All the requirements are clearly mentioned in an unambiguous way.
- The total no of requirements in this diagram is 19.
- Requirement IDs are revised for the fixes of inspection review comments.

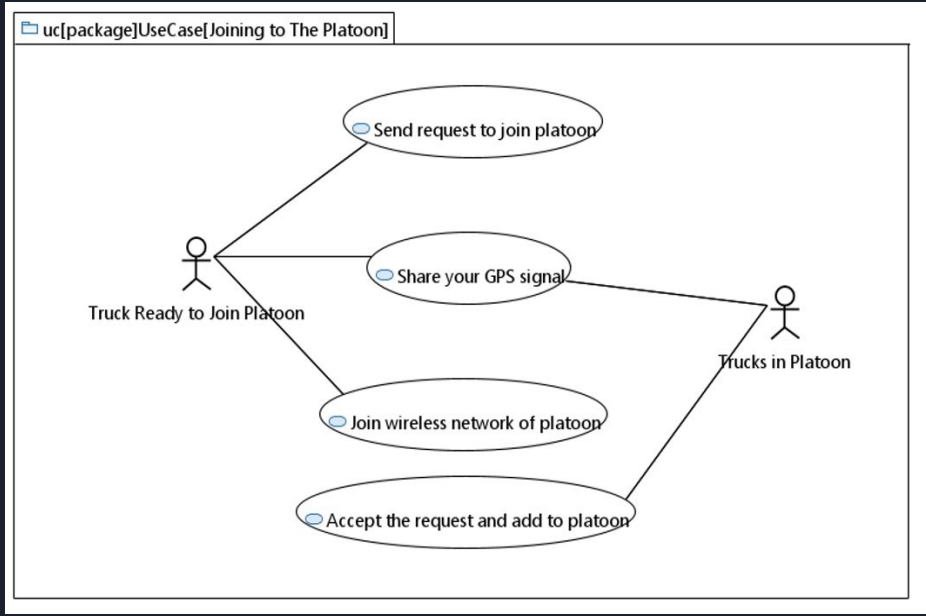


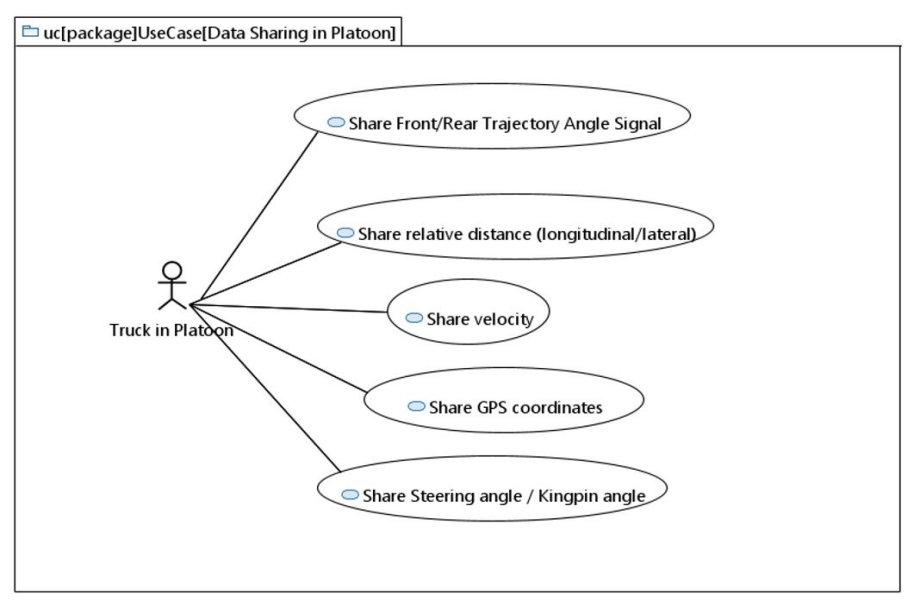
Use Case Diagram

- Use case diagram defines the operations that users want to perform on the system through a function.
- It consists of four main elements: Actors, System, Use Cases and their relations.

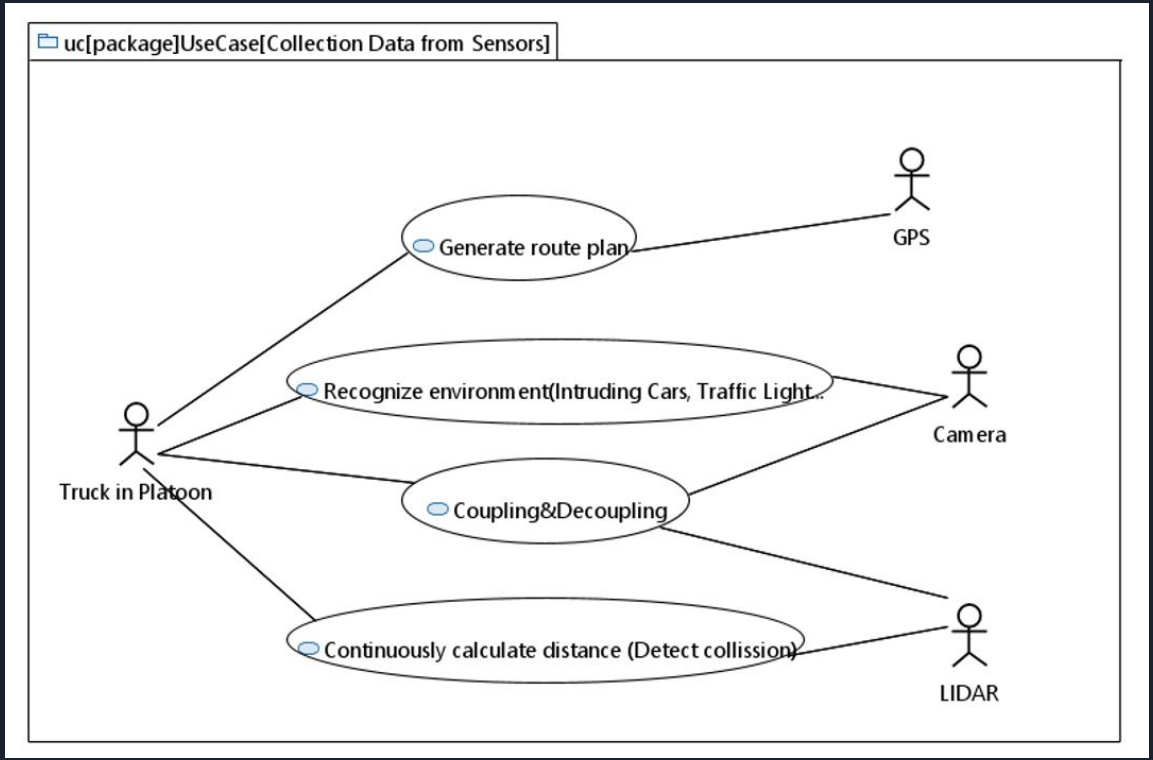


Use Case Diagram



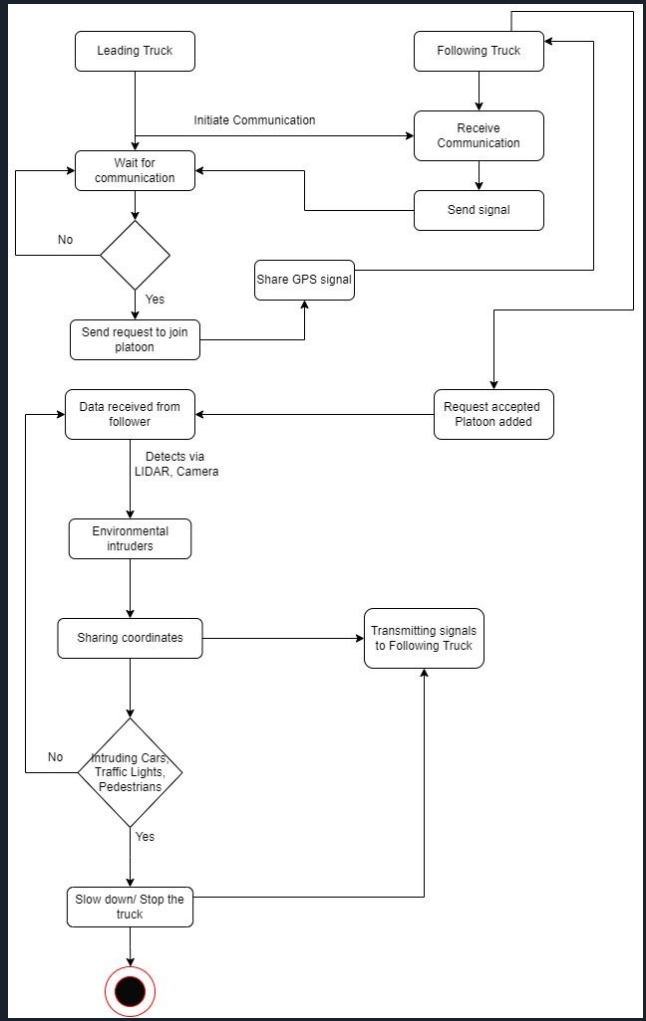


Use Case Diagram



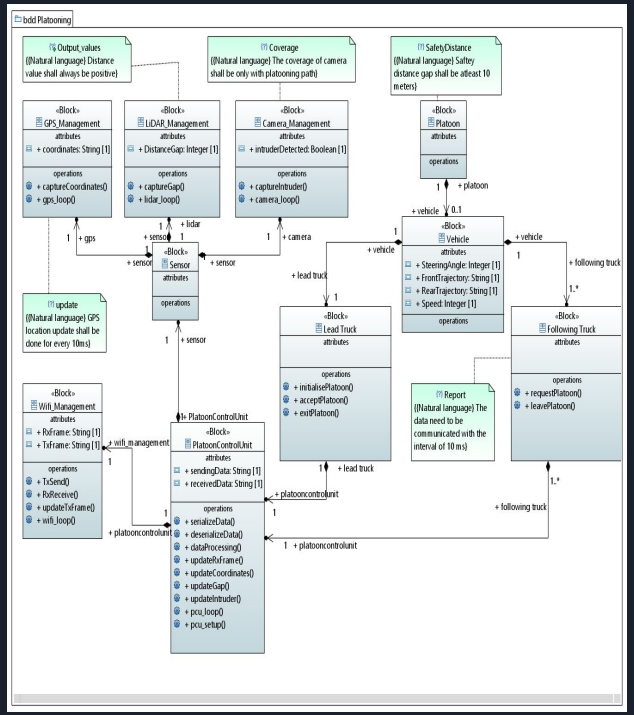
Activity Diagram

- An Activity diagram is a behavioural diagram (it depicts the entire behaviour of the system).
- It shows the control flow from start point to a finish point showing the various decision paths that exists while the activity is being executed.
- Activity diagrams show both computational and organizational forms (i.e., workflows).
- Activity diagrams are developed from a restricted number of shapes, associated with arrows.
- The foremost vital shape types: ellipses speak to actions; diamonds speak to decisions; bars speak to the begin (part) or conclusion (connect) of concurrent activities; a dark circle speaks to the begin (introductory hub) of the workflow; an encompassed dark circle speaks to the conclusion (last node).
- Arrows run from begin towards the conclusion and speaks the order in which the exercises flow.



Block Diagram

- The whole platooning system has been analysed and it is levelled down into modular units.
- Very first structural representation of the platooning system became possible by this block diagram.
- Each block of the block diagram package is considered as the definition of the particular component in that system.
- Block diagram helped to show the main components of the truck platooning system more precisely, i.e., PCU[platooning control unit], GPS, LiDAR, Camera and Wi-Fi
- Relation between the blocks are handled with composite and aggregation connectors.
- Operations and Attributes of the blocks pointed out the behaviour of the component in the system.



Sequence Diagram

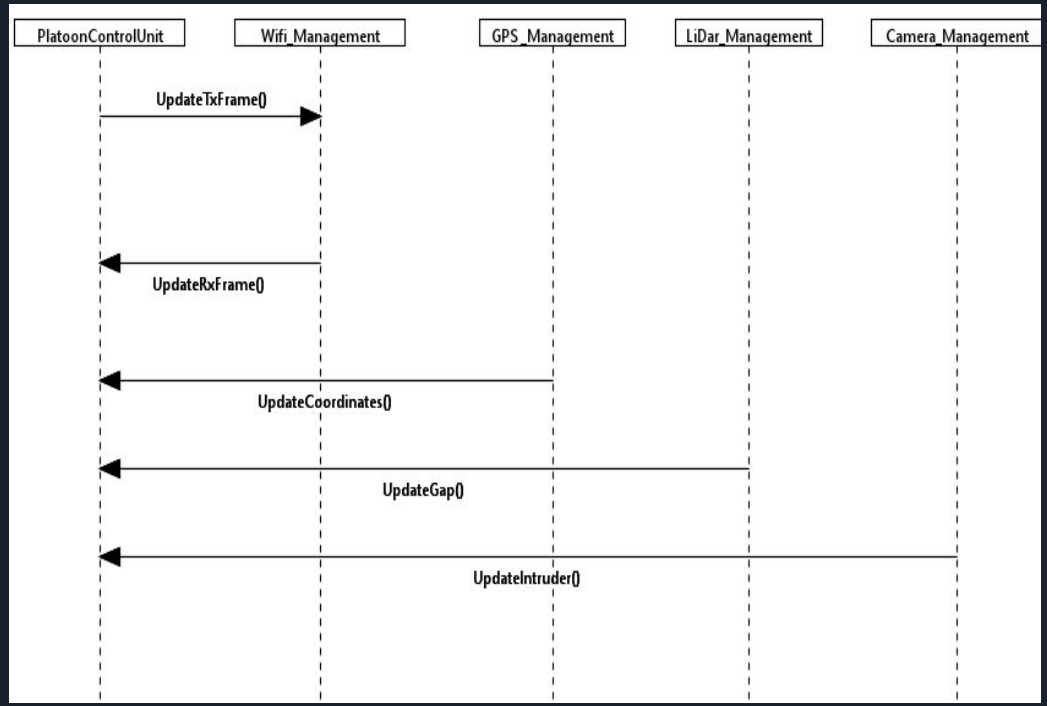
-UpdateTxFrame to update the Wifi credentials

-UpdateRxFrame to show the main controller the received output

-UpdateCoordinates is sent from Gps to update longitude and latitude of first truck

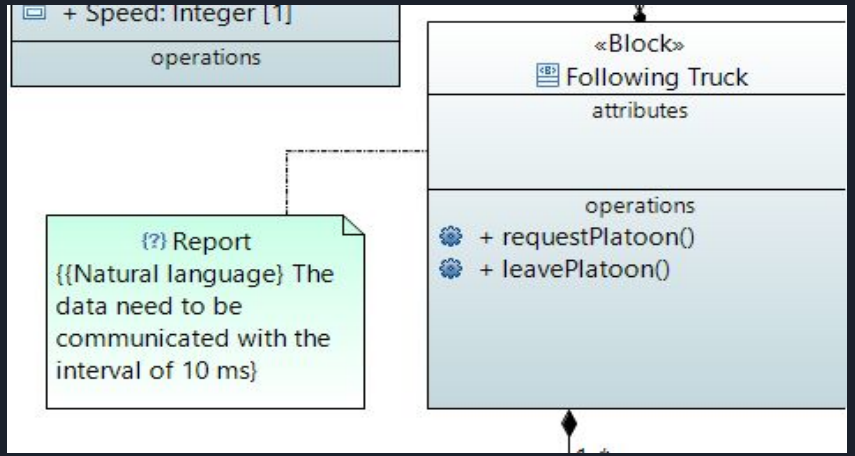
-UpdateGap is getting data from the Radar and updates distance between the trucks

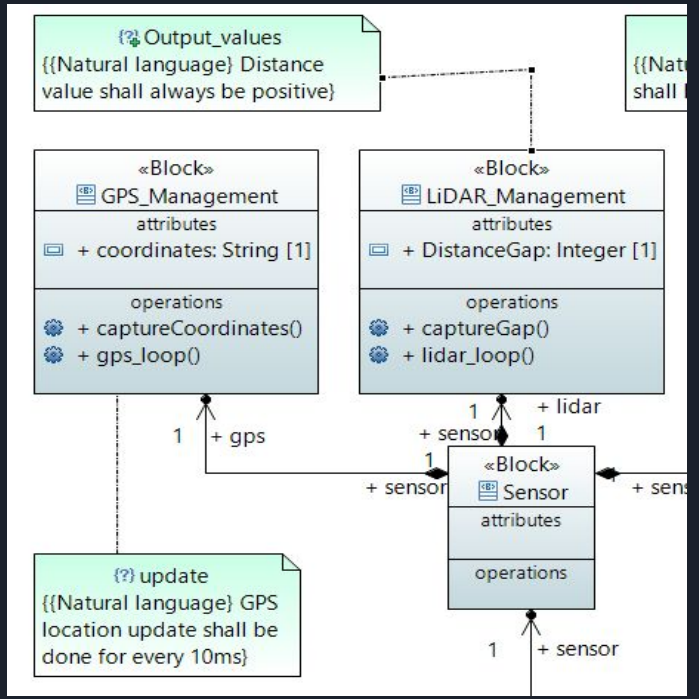
-As there is intruder in the system PCU got informed by UpdateIntruder message



Parametric Constraint Diagram

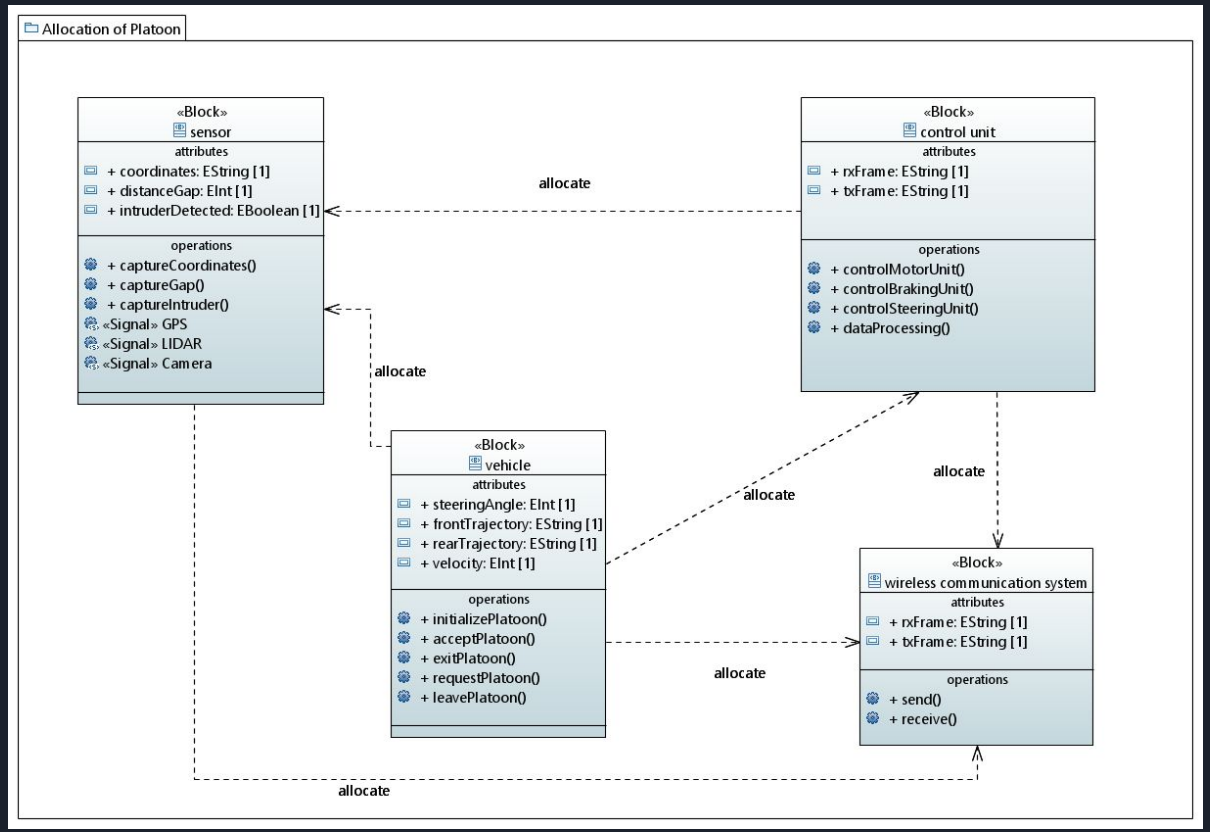
- Parametric constraints are added to the blocks of the block diagram.
- The constraints are added in the Natural language.



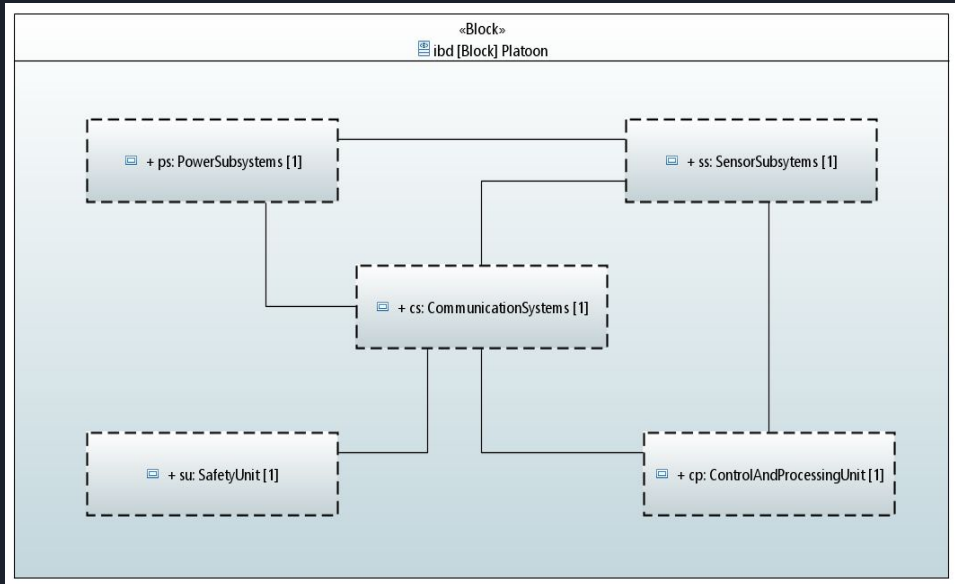


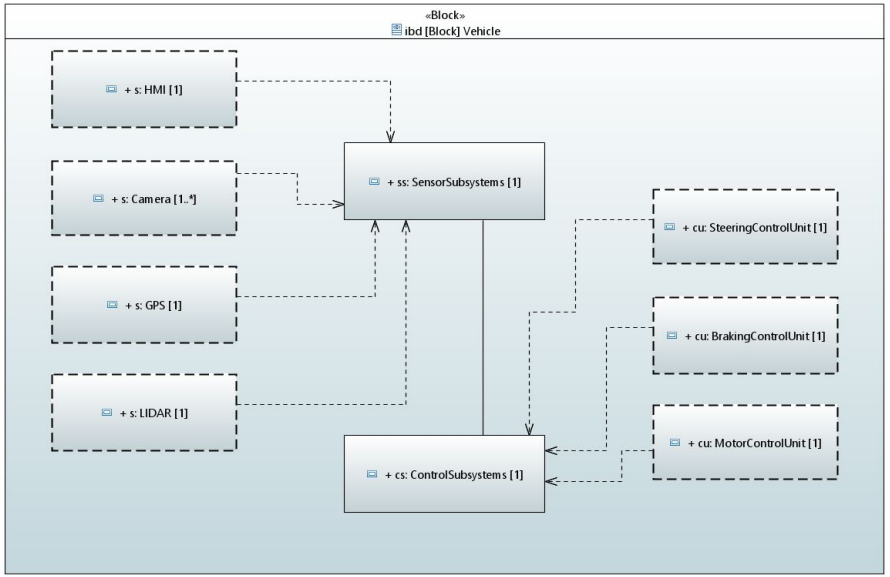
Allocation Diagram

- Allocation diagrams are used to define relationship of various parts of the model.



Internal Block Diagram

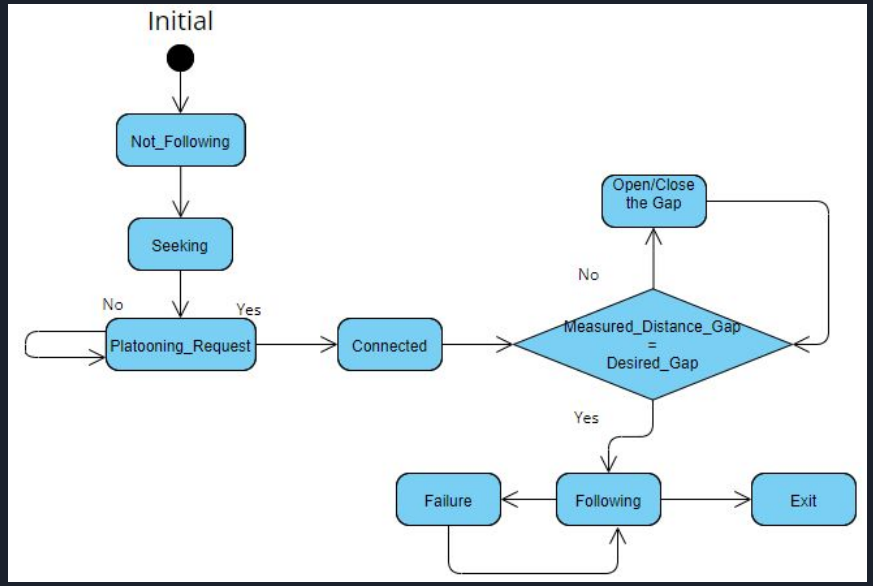




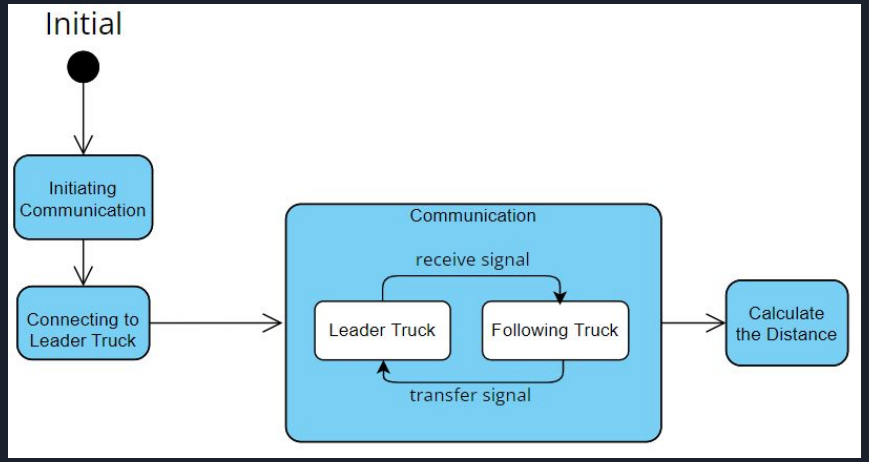
State machine Diagram

State machine diagrams are typically used to describe state-dependent behavior for an object. An object responds differently to the same event depending on what state it is in. State machine diagrams can be shown uniquely depending on the system and components that are going to be chosen based on requirements. In this case, 3 different components (Vehicle, Sensor and Platoon) are chosen to show their behaviors in specific circumstances.

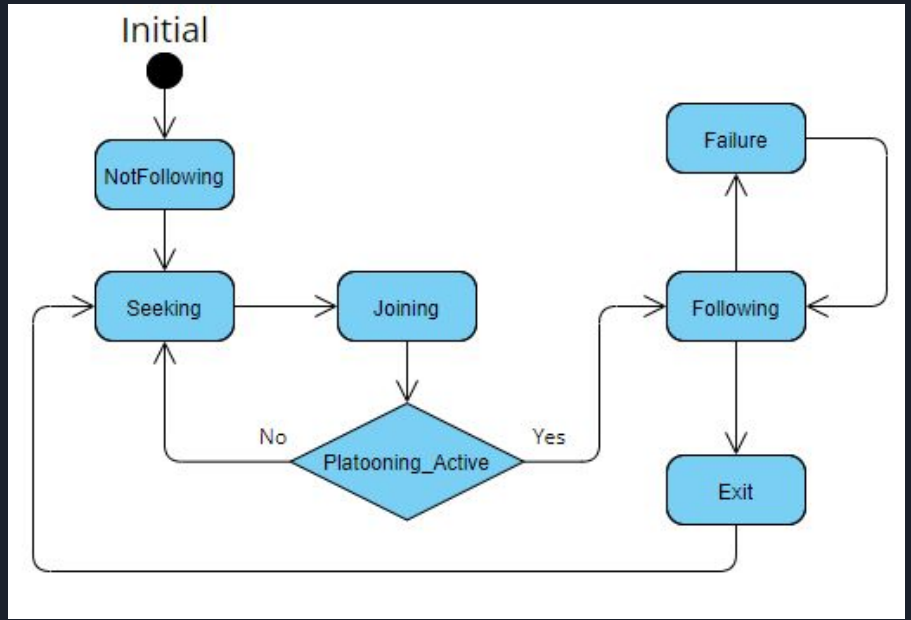
- Vehicle Component: The vehicles in the platoon may vary according to the user's requirements. But every vehicle must have some states in the platoon. In our case, “Vehicle Component” has 9 states. Each state represents the current position of the component. The main purpose of the whole tasks is to make sure if the component works properly.



- Sensor Component: The sensor component is used to calculate the distance between the vehicles. This sensor is a vital in platoon system to make sure if the distance is equal to desired distance. In order to do that, sensor has to receive signals from leader vehicle to calculate the distance.



- Platoon Component: Platoon is a component represents the whole system. This component has 7 states also contains emergency state. When if the missions don't go as expected and the "platoon" encounters an emergency, platoon goes into “Failure” state and tries to figure out the problem to make sure whole system is secured.



Implementation

- The implementation has been done for the component PCU [Platooning control unit].
- The state diagrams are taken as input for the code realization.
- Language used is CPP.
- FreeRTOS is used to handle all the tasks.
- PCU component is tested in the Arduino ESP32 hardware along with the other components with basic implementation.
- Action of the system during the “decrease distance gap” and “Vehicle intruded” are shown in the console screen of the hardware.

Implementation Output

With Intruder

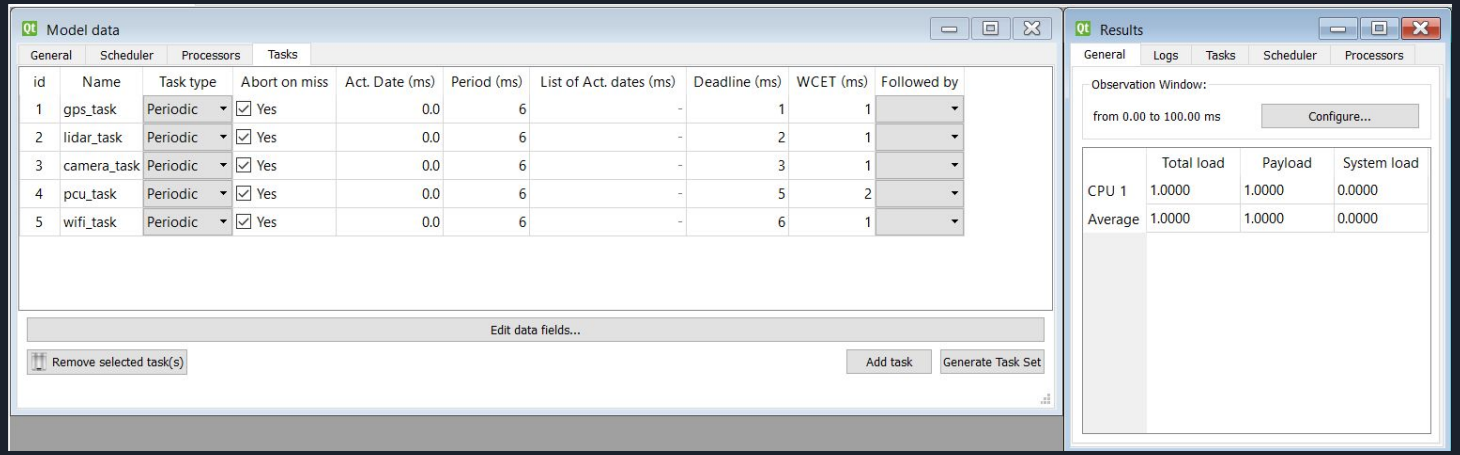
```
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 405
Intruder detected Decreasing Speed
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 405
Intruder detected Decreasing Speed
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 405
Intruder detected Decreasing Speed
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 405
Intruder detected Decreasing Speed
```

Without Intruder

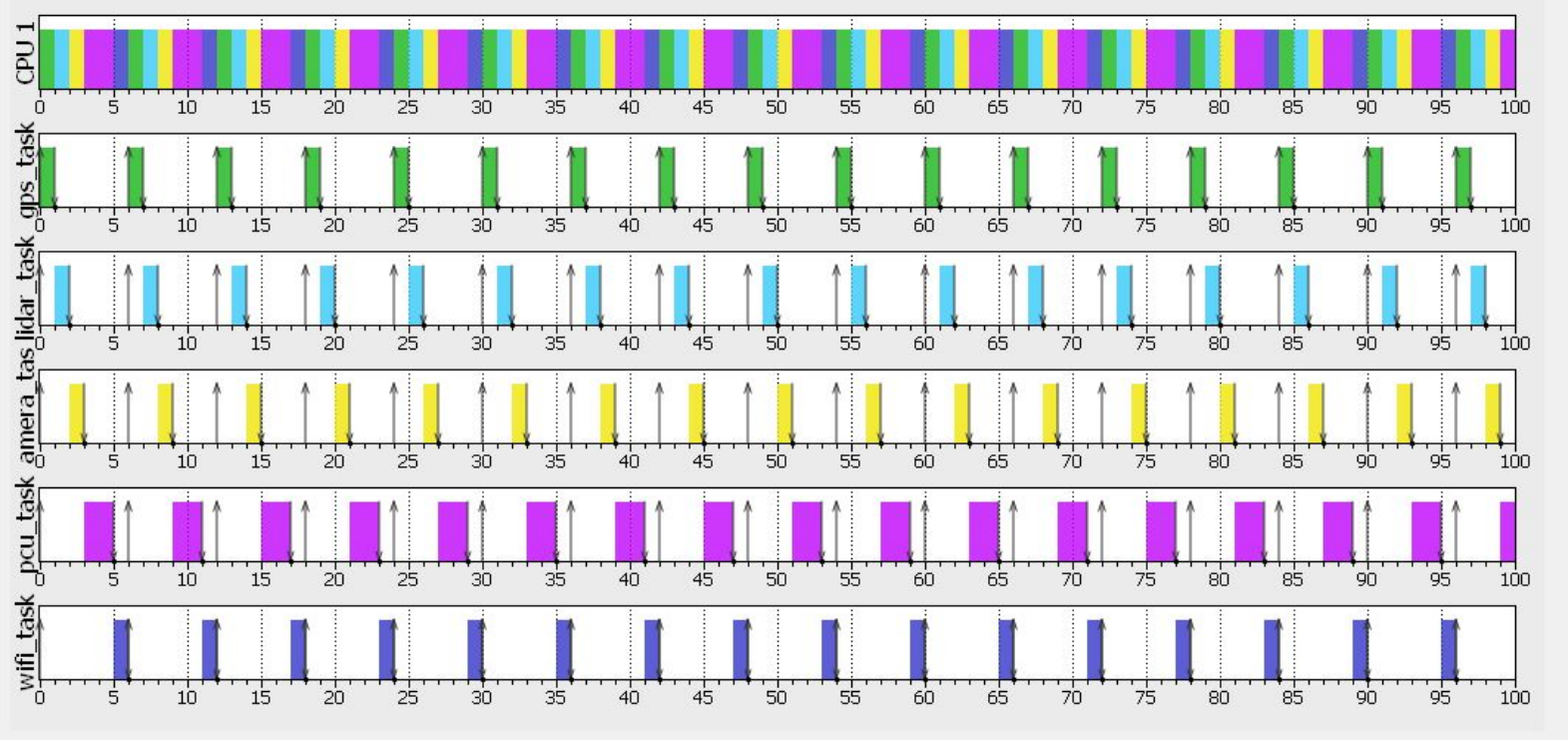
```
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 401
Maintaining Speed
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 402
Maintaining Speed
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 402
Maintaining Speed
Front truck latitude before updating: 200.00
Front truck longitude before updating: 100.00
Front truck latitude after updating: 200.00
Front truck longitude after updating: 100.00
Increasing Speed : Measured_Distance = 402
Maintaining Speed
```

Scheduling

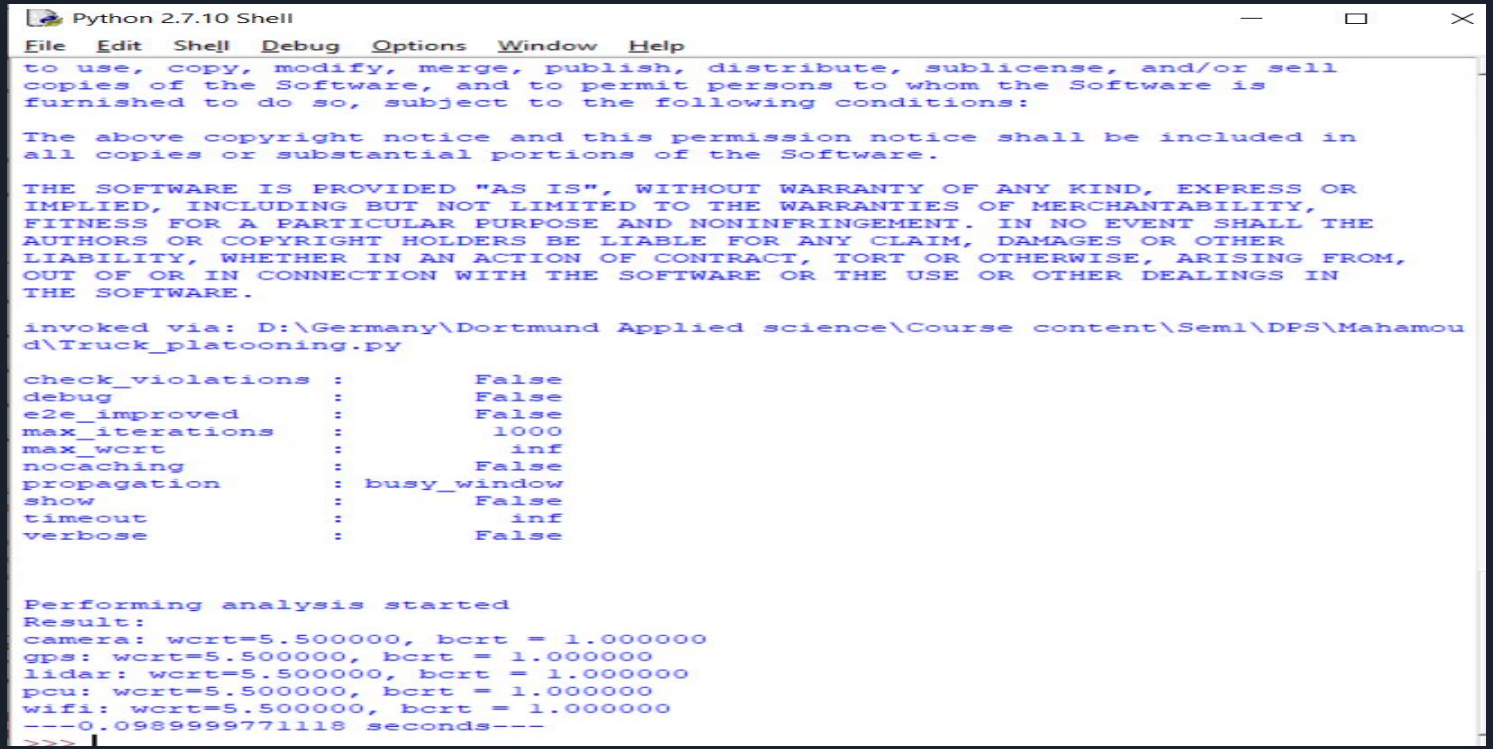
- The Scheduling of the task is calculated with both Simso and pyCPA.
- Single core is used for the scheduling.
- EDF scheduling is followed for this system.
- Reason is, the task like GPS, LiDAR, and Camera needs to be completed before PCU task.
- And these three tasks are operation wise requires less computation time.
- As per the EDF working principle, the tasks are scheduled as like below table,



Scheduling Output



Scheduling pyCPA Output



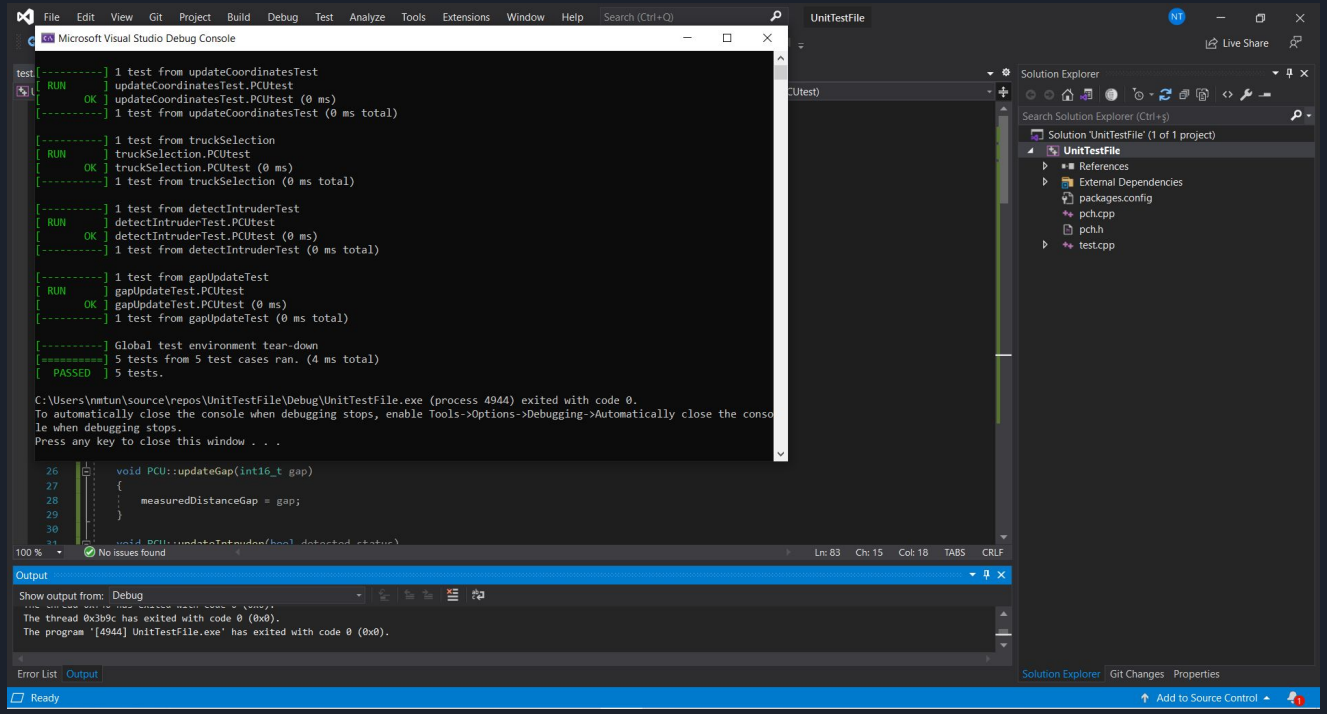
Inspection

The Inspection has been done for all the model diagrams and code.

All the documents using google sheet

Unit testing

- The main purpose of unit testing is showing that the components of system works as expected.
- According to testing experience and common errors, we can vary test cases.



Component Testing

There are 5 components in the system. PCU, Wifi, Lidar, Camera and Gps. PCU is main object so I tested other using PCU object functions

-Test Camera if it informs the PCU about intruder

-Test GPS weather it updates coordinates correctly

-Test Lidar to send distance data to PCU and check if PCU is responding in correct way

-Test Wifi via comparing output from it with input from PCU

Thank you :)