Cross Traffic Management

ANAEDU UKAMAKA AKUMILI

EMIRKAN SALI

SHIHAB UD DOULA

UNANMA JUSTICE CHISOM

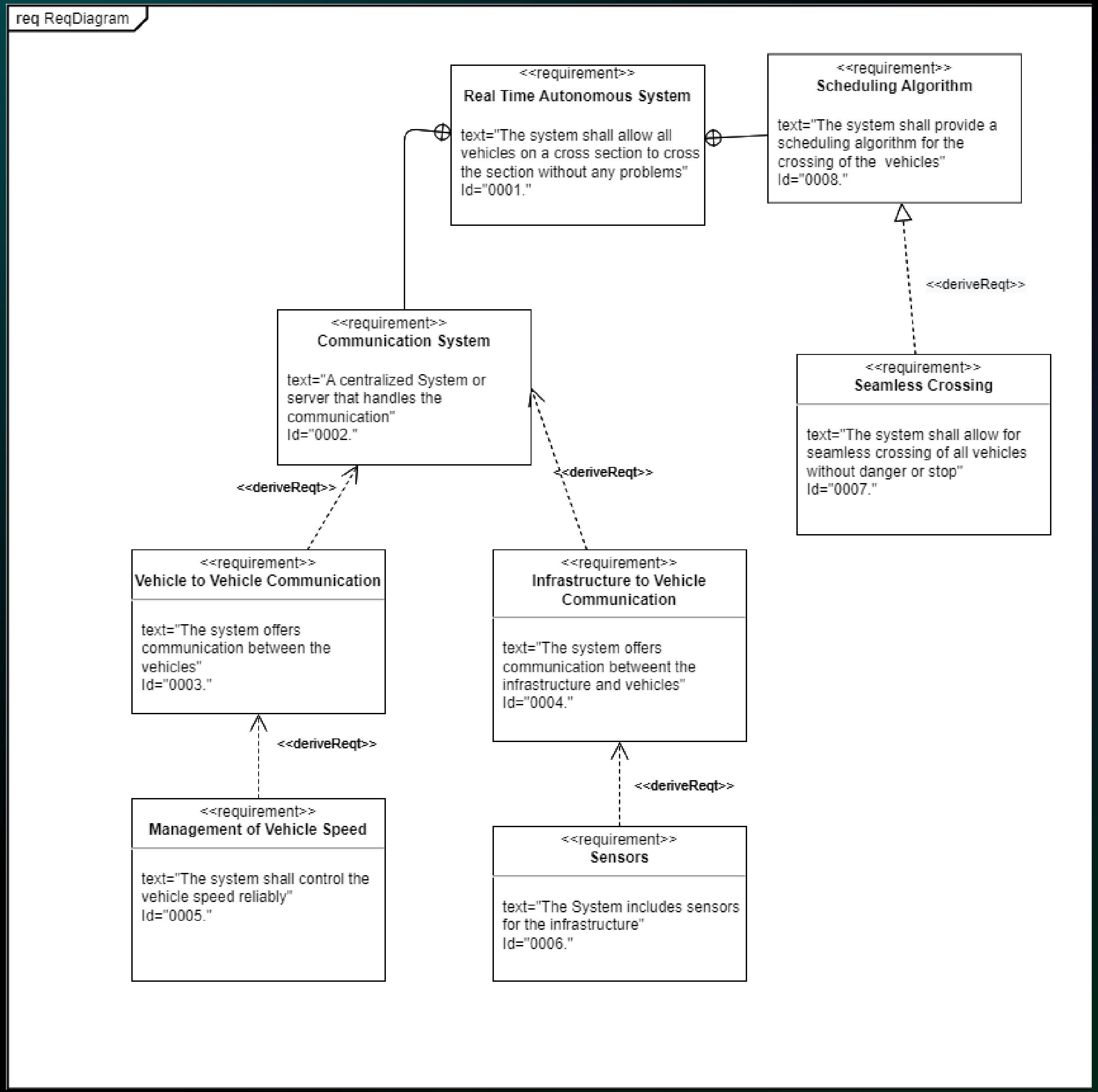
MOTIVATION

- Statement of the problem
- Benefits of autonomous vehicles
- Limitations
- Contributions

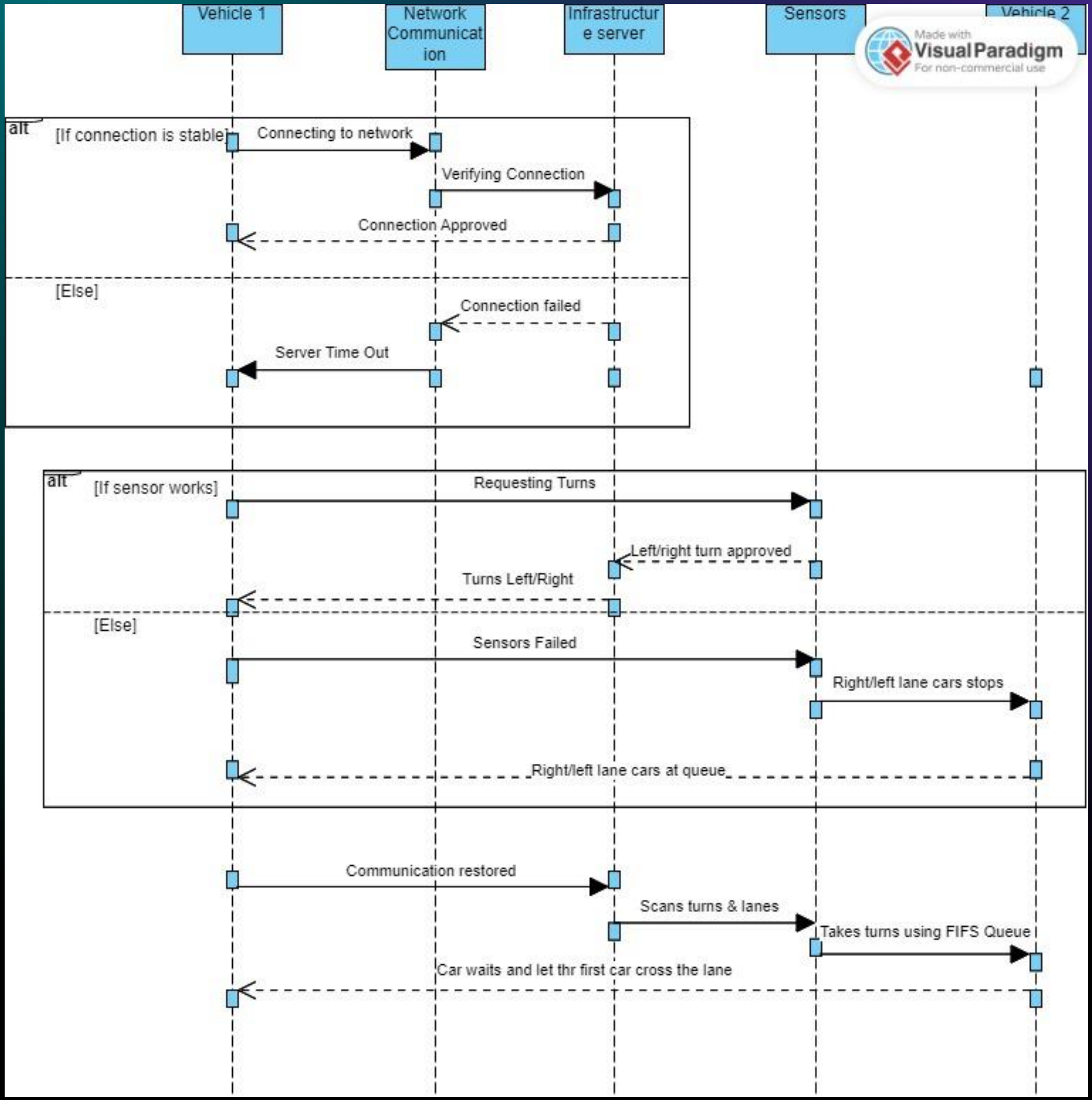
ARCHITECTURAL BEHAVIOR

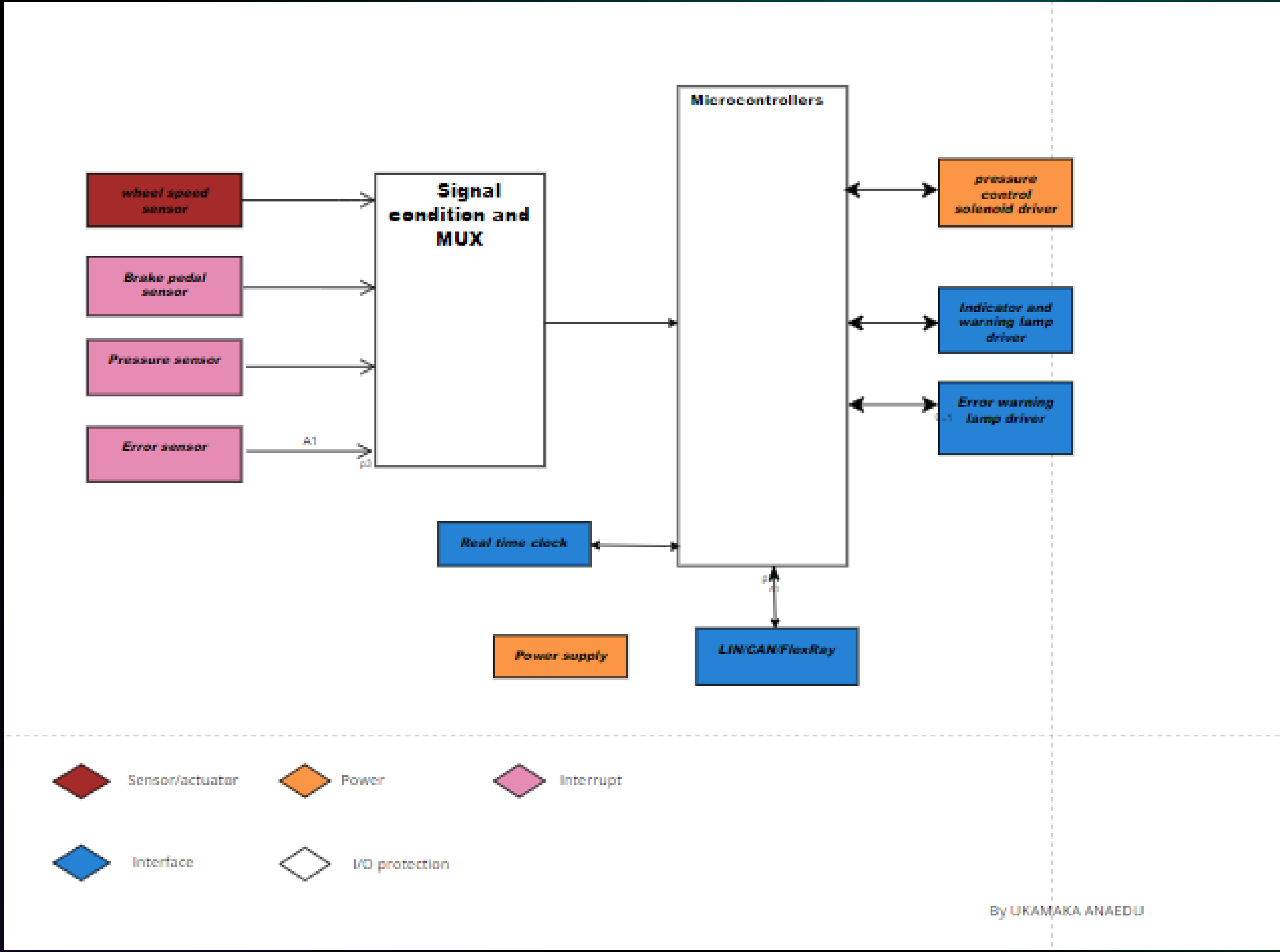
- Seamless crossing
- Automated system
- Communication between cars and system
- Sensors
- infrastructure

Requirements

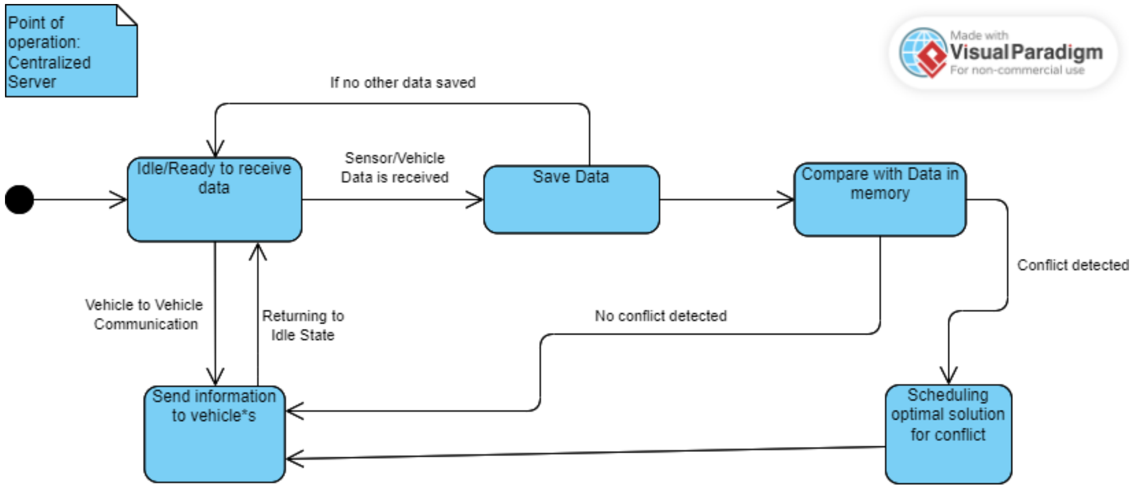


Sequence diagram





BLOCK DIAGRAM

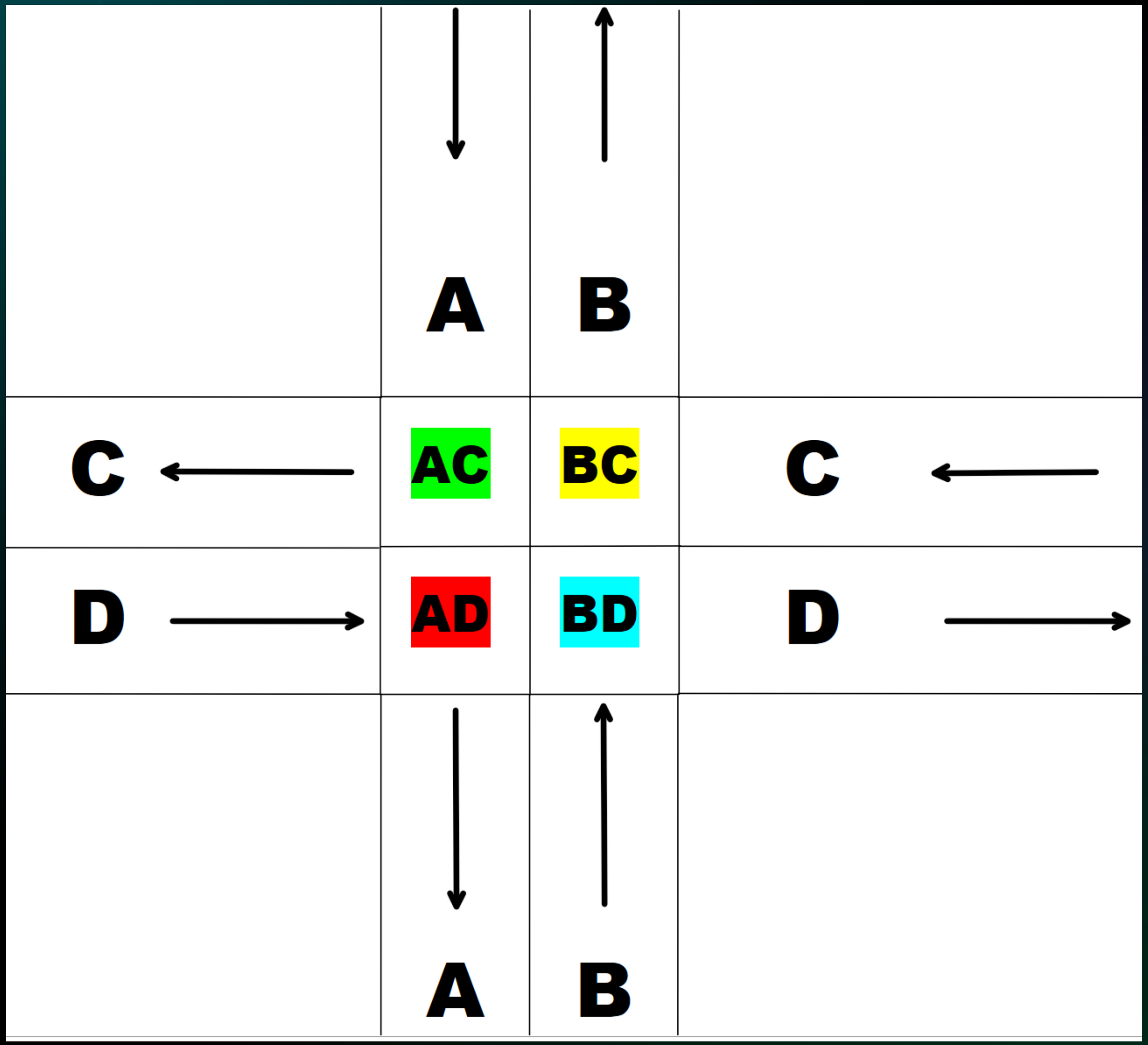


By Emirkan Sali

State diagram

- Concept state diagram
- Main states are ready state and scheduling state
- Compares with internal map

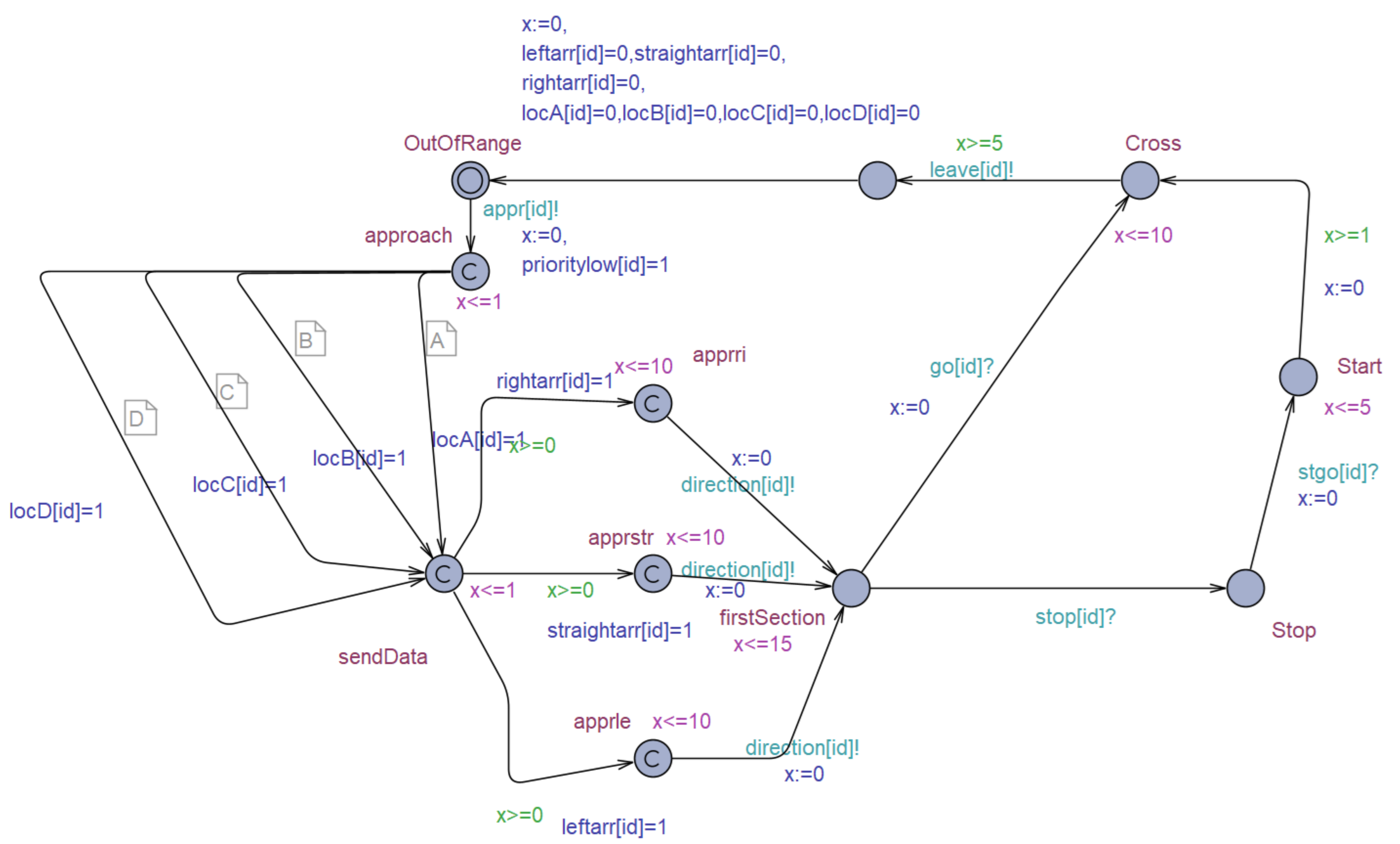
Cross-section concept



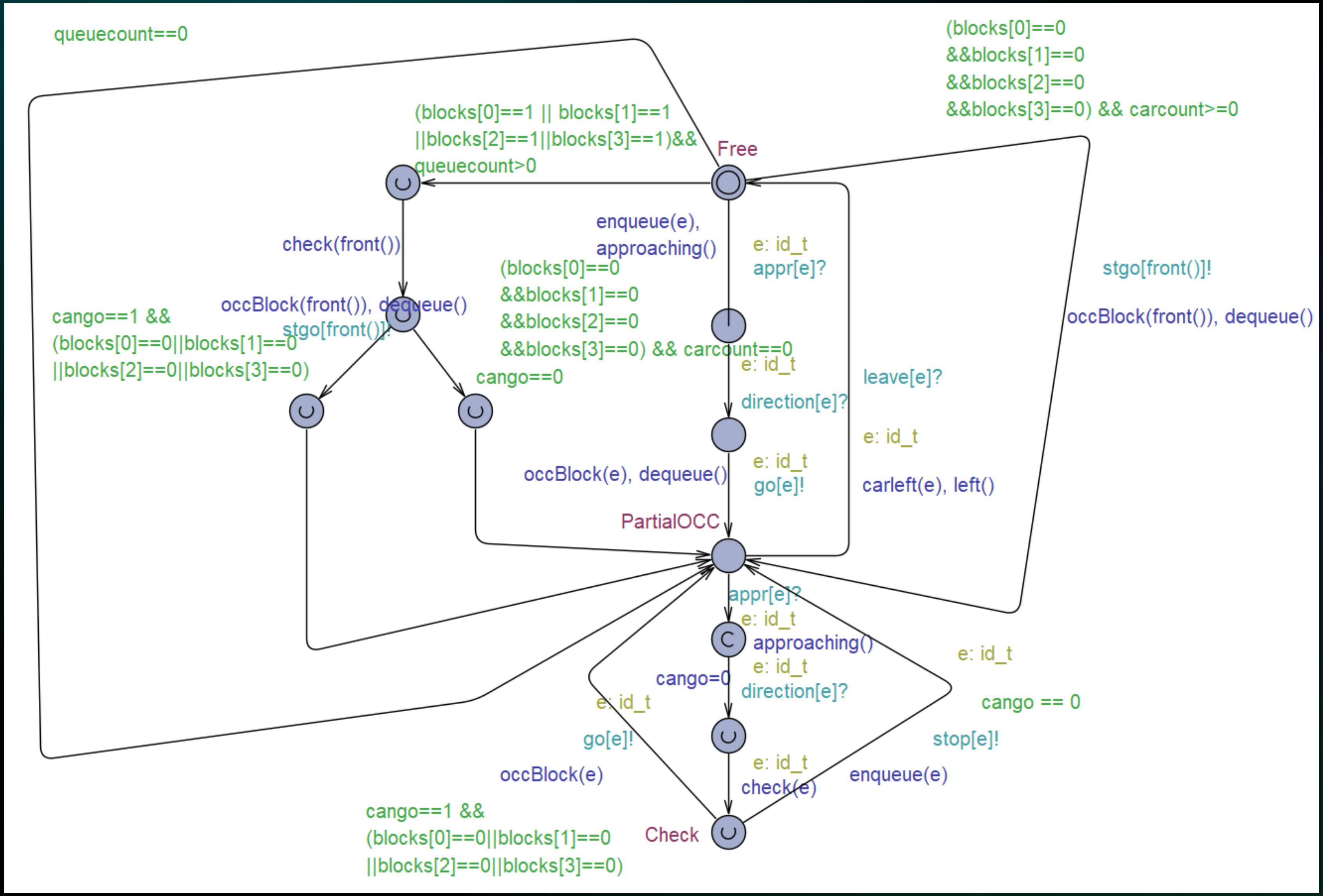
Uppaal Model

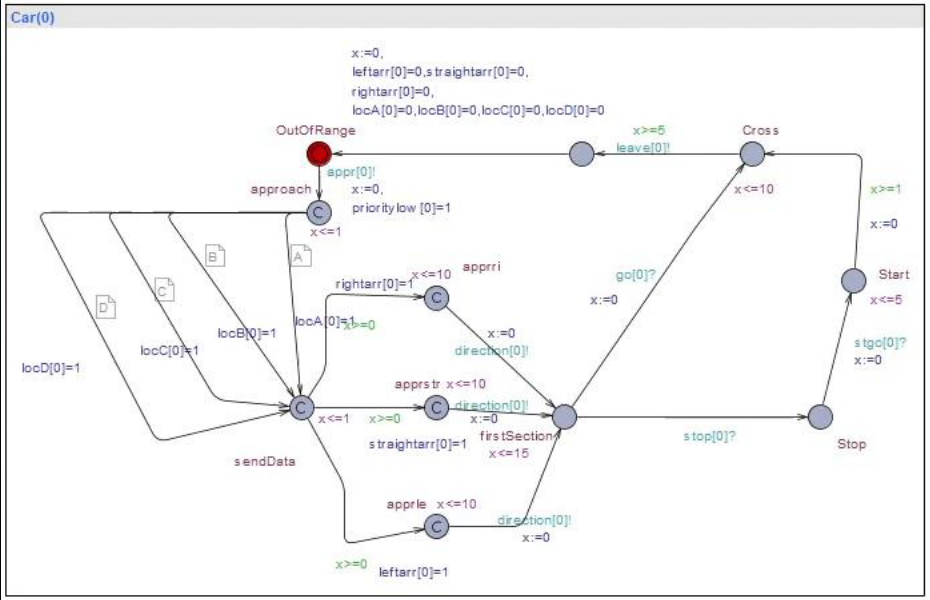
- Separate model for a car and system
- Random allocation of location and action for each car
- Location, action and state of the cross section saved in arrays
- FCFS queue for handling the cars

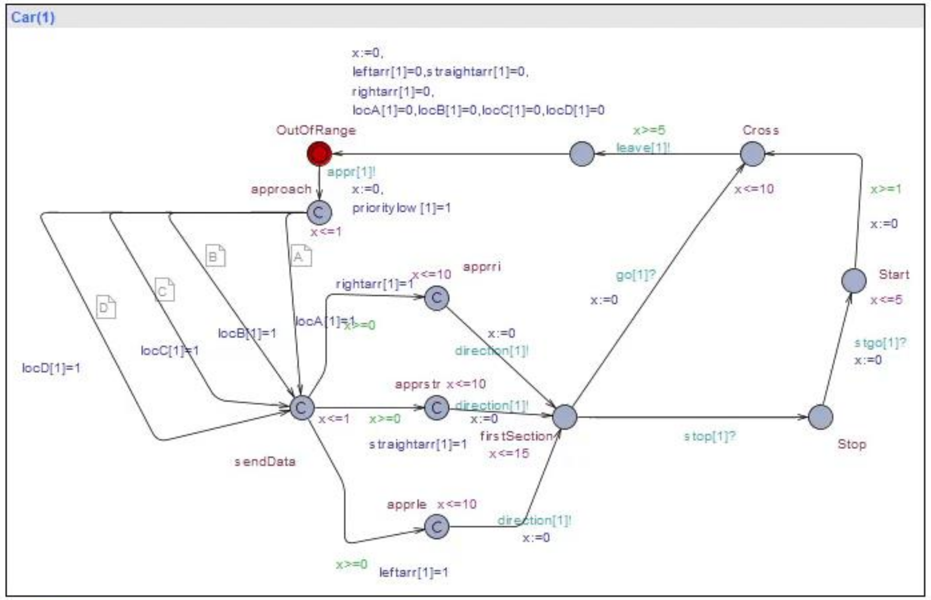
Car Uppal Model

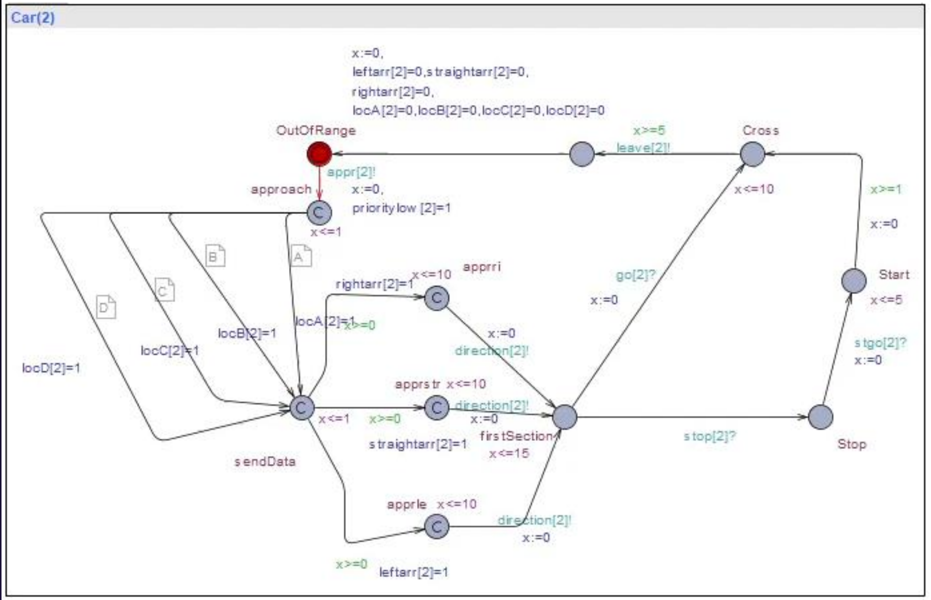


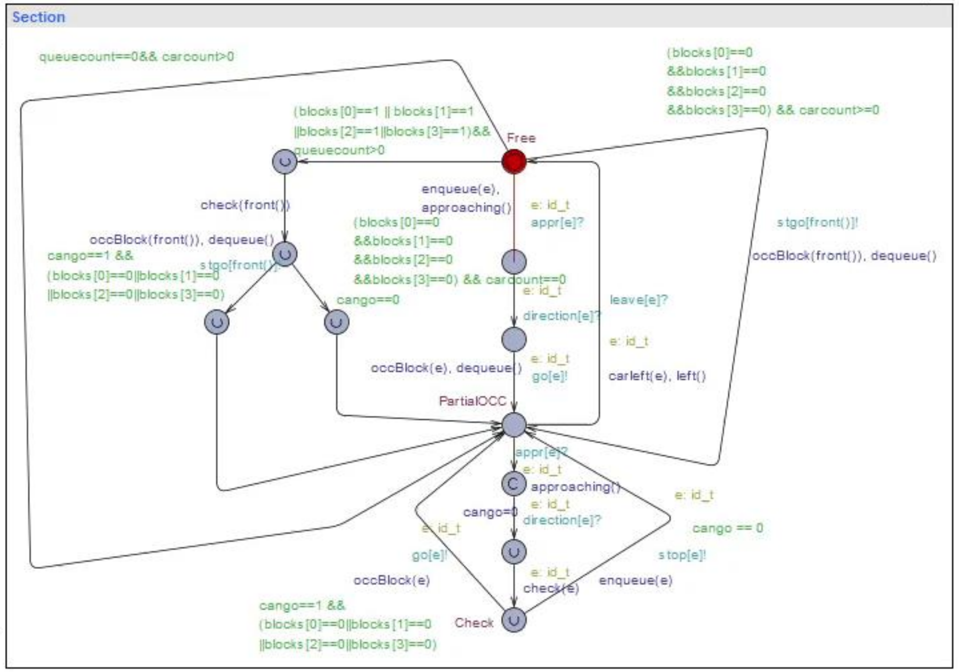
System Model



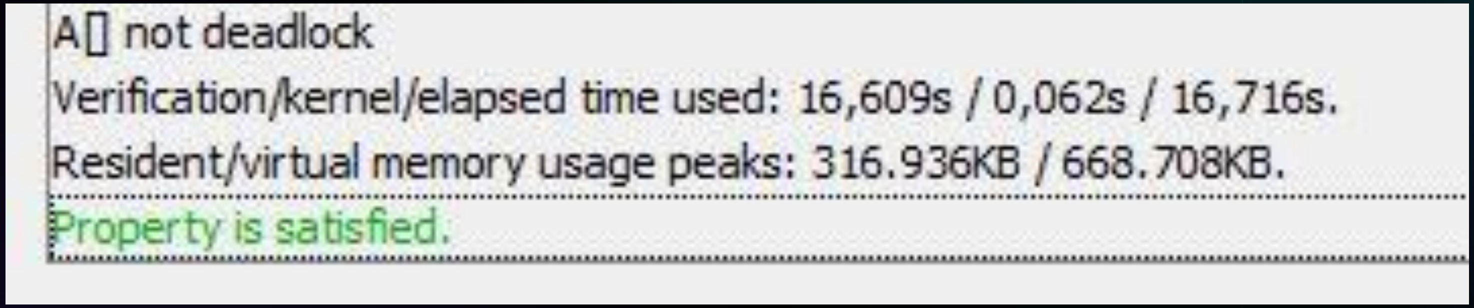

Car(0)
x:=0,
leftarr[0]=0,straightarr[0]=0,
rightarr[0]=0,
locA[0]=0,locB[0]=0,locC[0]=0,locD[0]=0
OutOfRange
x>=5
leave[0]!
Cross
appr[0]!
approach
x:=0,
prioritylow[0]=1
x<=1
x<=10
x>=1
x:=0
B
A
C
D
rightarr[0]=1
x<=10
apprri
go[0]?
x:=0
Start
x<=5
locB[0]=1
locA[0]=1
x>=0
x:=0
direction[0]!
locC[0]=1
locD[0]=1
stgo[0]?
x:=0
apprstr x<=10
direction[0]!
x:=0
x<=1
x>=0
straightarr[0]=1
firstSection
x<=15
stop[0]?
Stop
sendData
apprle x<=10
direction[0]!
x:=0
x>=0
leftarr[0]=1

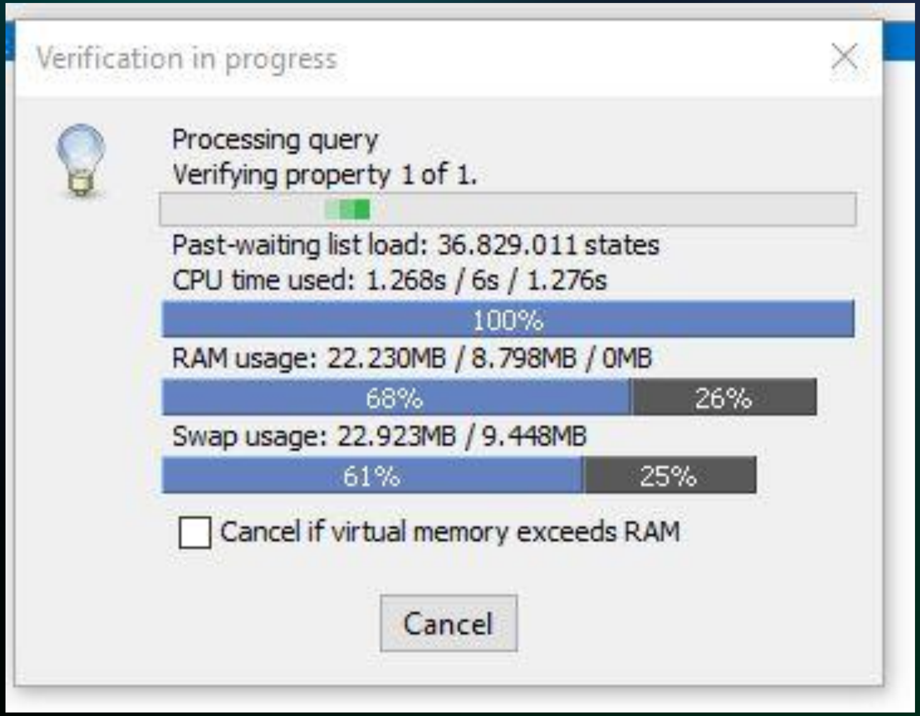

Car(1)
x:=0,
leftarr[1]=0,straightarr[1]=0,
rightarr[1]=0,
locA[1]=0,locB[1]=0,locC[1]=0,locD[1]=0
OutOfRange
x>=5
leave[1]!
Cross
appr[1]!
approach
x:=0,
prioritylow[1]=1
x<=1
x<=10
x>=1
x:=0
B
A
C
D
rightarr[1]=1
x<=10
apprri
go[1]?
x:=0
Start
x<=5
locB[1]=1
locA[1]=1
x>=0
x:=0
direction[1]!
locC[1]=1
locD[1]=1
stgo[1]?
x:=0
apprstr x<=10
direction[1]!
x:=0
x<=1
x>=0
straightarr[1]=1
firstSection
x<=15
stop[1]?
Stop
sendData
apprle x<=10
direction[1]!
x:=0
x>=0
leftarr[1]=1


Car(2)
x:=0,
leftarr[2]=0,straightarr[2]=0,
rightarr[2]=0,
locA[2]=0,locB[2]=0,locC[2]=0,locD[2]=0
OutOfRange
x>=5
leave[2]!
Cross
appr[2]!
approach
x:=0,
prioritylow[2]=1
x<=1
x<=10
x>=1
x:=0
B
A
C
D
rightarr[2]=1
x<=10
apprri
go[2]?
x:=0
Start
x<=5
locB[2]=1
locA[2]=1
x>=0
x:=0
direction[2]!
locC[2]=1
locD[2]=1
stgo[2]?
x:=0
apprstr x<=10
direction[2]!
x:=0
x<=1
x>=0
straightarr[2]=1
firstSection
x<=15
stop[2]?
Stop
sendData
apprle x<=10
direction[2]!
x:=0
x>=0
leftarr[2]=1


Section
queuecount==0&& carcount>0
(blocks[0]==0
&&blocks[1]==0
&&blocks[2]==0
&&blocks[3]==0) && carcount>=0
(blocks[0]==1 || blocks[1]==1
||blocks[2]==1||blocks[3]==1)&&
queuecount>0
Free
check(front())
enqueue(e),
approaching()
e: id_t
appr[e]?
stgo[front()]!
occBlock(front()), dequeue()
occBlock(front()), dequeue()
cango==1 &&
(blocks[0]==0||blocks[1]==0
||blocks[2]==0||blocks[3]==0)
stgo[front()]!
(blocks[0]==0
&&blocks[1]==0
&&blocks[2]==0
&&blocks[3]==0) && carcount==0
e: id_t
direction[e]?
cango==0
leave[e]?
e: id_t
carleft(e), left()
occBlock(e), dequeue()
e: id_t
go[e]!
PartialOCC
appr[e]?
e: id_t
approaching()
e: id_t
direction[e]?
cango=0
e: id_t
go[e]!
occBlock(e)
e: id_t
check(e)
enqueue(e)
e: id_t
cango == 0
stop[e]!
cango==1 &&
(blocks[0]==0||blocks[1]==0
||blocks[2]==0||blocks[3]==0)
Check

Verification





C++ implementation

- Implementation in c++ using class for car entity
- Class asigns priority, location and action randomly
- Loop simulates behaviour of a cross section
- Cars can spawn and approach based on random events
- Car timing is updated in variable
- Car checking occurs to ensure safe crossing

Code output

```
Car spawned with ID: 1
Car spawned with ID: 2
Car spawned with ID: 3
3Car ID: 1
Priority: 2
Location: C
Action: Straight

Car ID: 2
Priority: 3
Location: A
Action: Right Turn

Car ID: 3
Priority: 2
Location: D
Action: Right Turn

Car ID: 1
priority: 1

New Status: approaching
State Change. car approaches
Enqueueing car with ID: 1

New Status: crossing
```

```
CurrentLoc = 'C'
timing of car '1' = 3

CurrentLoc = 'D'
timing of car '3' = 3

CurrentLoc = 'A'
timing of car '2' = 2
this car can not start yet

timing advanced


timing advanced


CurrentLoc = 'C'
timing of car '1' = 4

Dequeuing

New Status: Out of Range
Blocks array =0
0
1
0
Car '1' has left the cross section
```

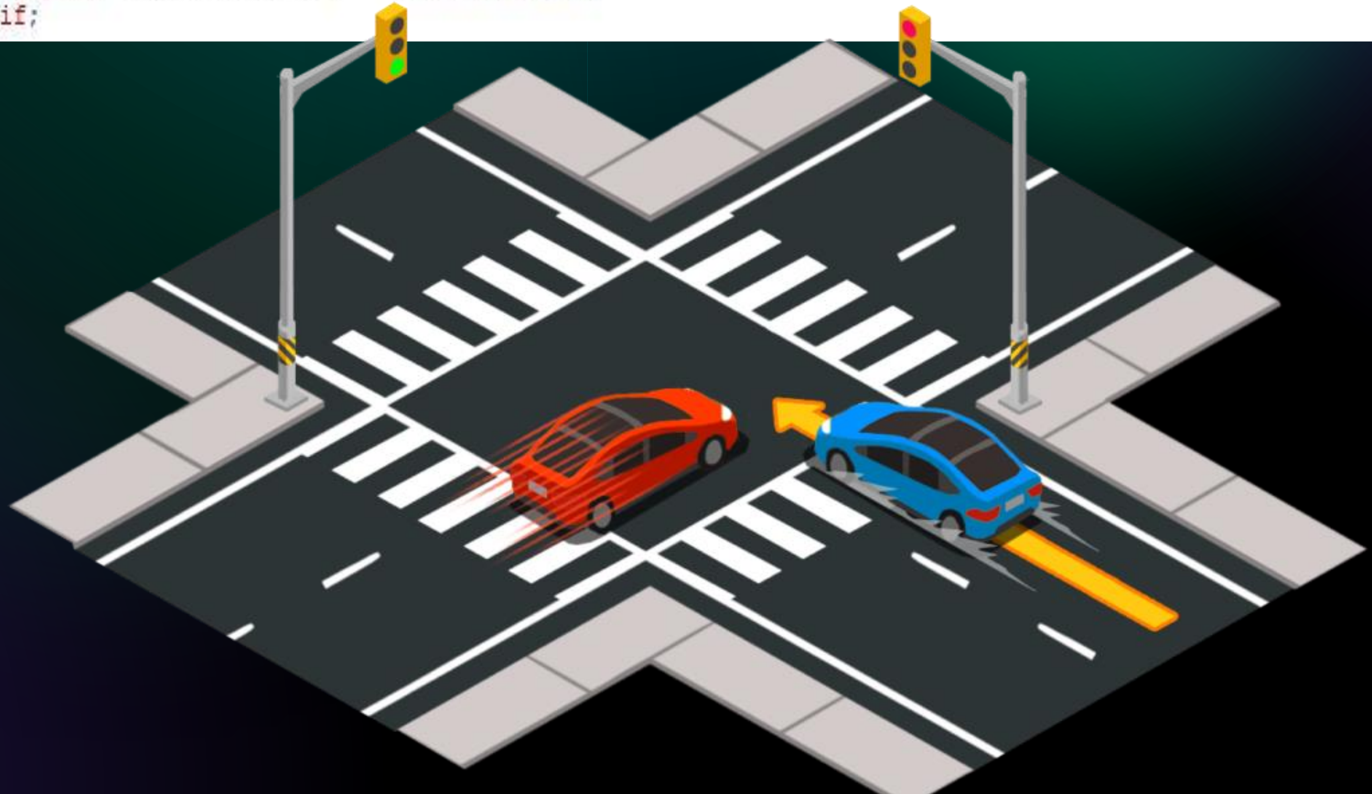
Hardware Implementation(VHDL)

- Why VHDL
- Entity Declaration

```
architecture behavior of Traffic is
      -- here, we declare the type and signal
type t_state is (NW,NE,SE,SW);
signal state: t_state;
signal FirstBit: std_logic_vector (1 downto 0) := Direction(3) & Direction(2);
signal SecondBit: std_logic_vector (1 downto 0) := Direction(1) & Direction(0);
```

Check for direction

```
Process (Direction) is
        begin

        if (Direction /= "0000" and Direction /= "1111")   --no action is expected if the directions are either Nort-North or East-East
        then
                case State is
                        when NW =>
                              if (secondBit /= "01" and slot = '1') -- Check if the second bit of the direction is different from "01" and there is a car waiting
                              then State <= SW; -- Go to the next state
                              else State <= NW;
                              -- Otherwise remain to the state
                              end if;
                              if (secondBit = "01") --If the second bit of the direction is the same as the second bit of the state.
                              then report "Exit succesful"; -- Exit the state
                              end if;
```



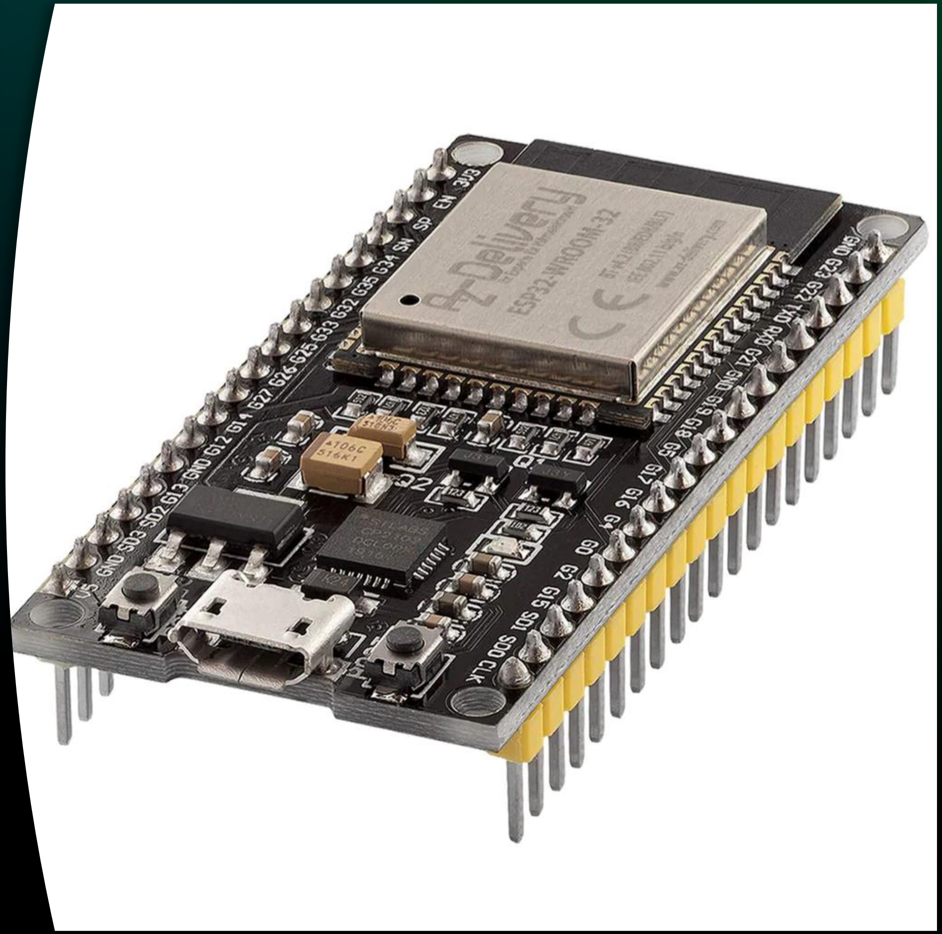
FreeRTOS

- Why ?
  - Open source
  - Task Management
  - Real time Responsiveness
  - Resource Management
- What it includes?
  - Semaphores
  - Mutexes
  - Queues

Implementations

FreeRTOS API

- **xTaskCreatePinnedToCore**
- **xSemaphoreCreateMutex**
- **xSemaphoreTake** & **xSemaphoreGive**
- **vTaskDelay**



Implementation

1

```
#include <Arduino.h>
#include <vector>
#include <algorithm>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
```

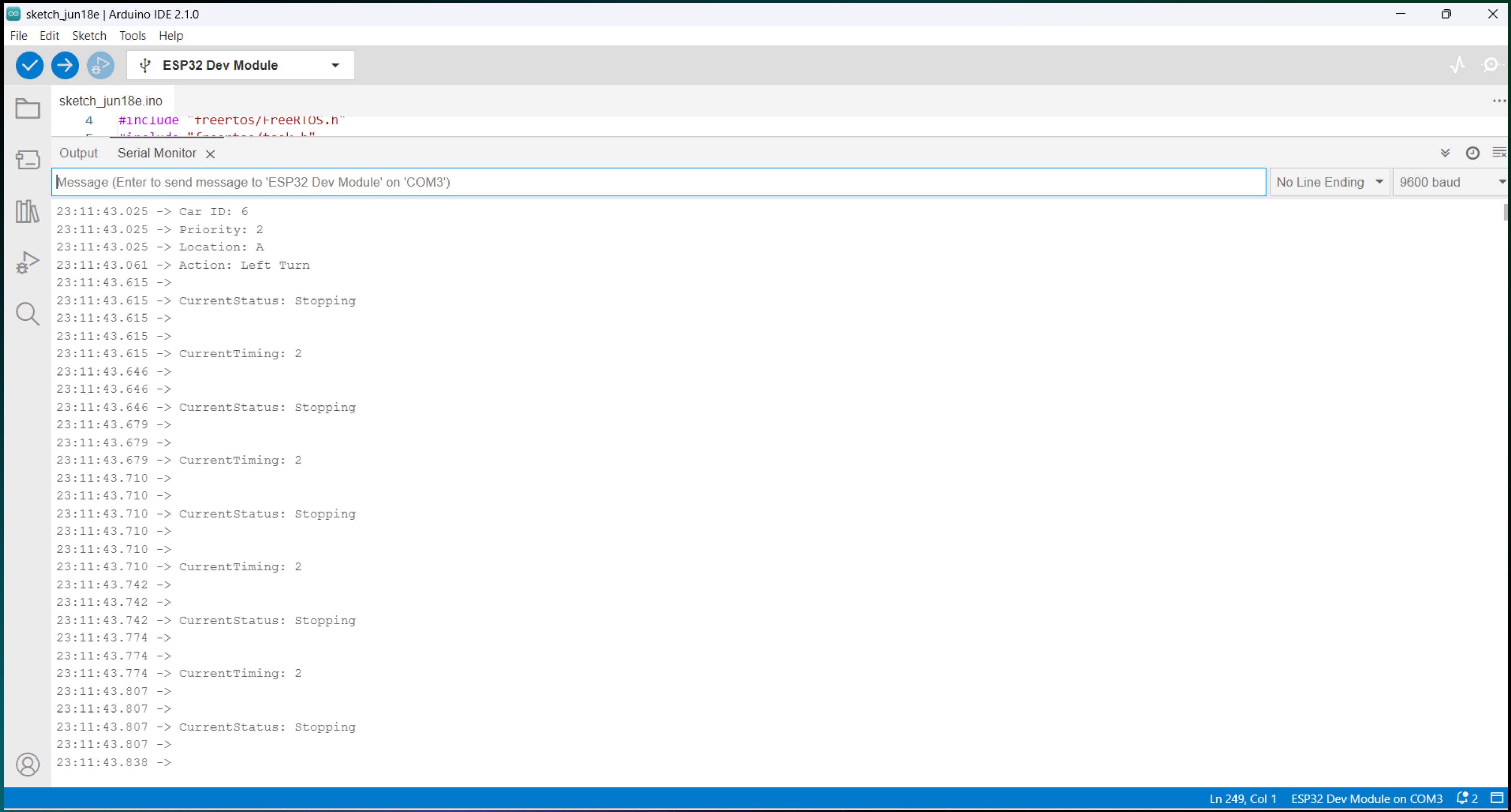
2

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

  // Initialize random seed
  randomSeed(analogRead(0));

  // Create 6 cars
  for (int i = 1; i <= N; i++)
  {
    cars.push_back(Car(i));
  }

  xTaskCreatePinnedToCore(task1, "Task1", 2048, NULL, 1, NULL, 0);
  xTaskCreatePinnedToCore(task2, "Task2", 2048, NULL, 1, NULL, 1);
}

void loop()
{
  // Empty loop as tasks handle the execution
}
```

3

```
      car.printCarInfo();
      vTaskDelay(pdMS_TO_TICKS(1000));
    }
  }
}
```

Output in Serial Monitor



Conclusion

- ✓Successful implementation
- ✓Integration of multiple technologies
- ✓Identified limitations
- ✓Potential of the designed system