Embedded Software Engineering

# Task 2

* The characteristics of Embedded Systems are:
* Reactive systems
* Real-time systems
* Continuous/discrete/hybrid systems
* Dependable systems
* Distributed systems
* Find examples in at least three application domains  
  (e.g., automotive, transportation, space mission)  
    
  Explain the different characteristics by these examples in detail
  + These includes details of the characteristics like (note: this not a complete list): How are the attributes of dependability addressed (e.g., which safety standard)

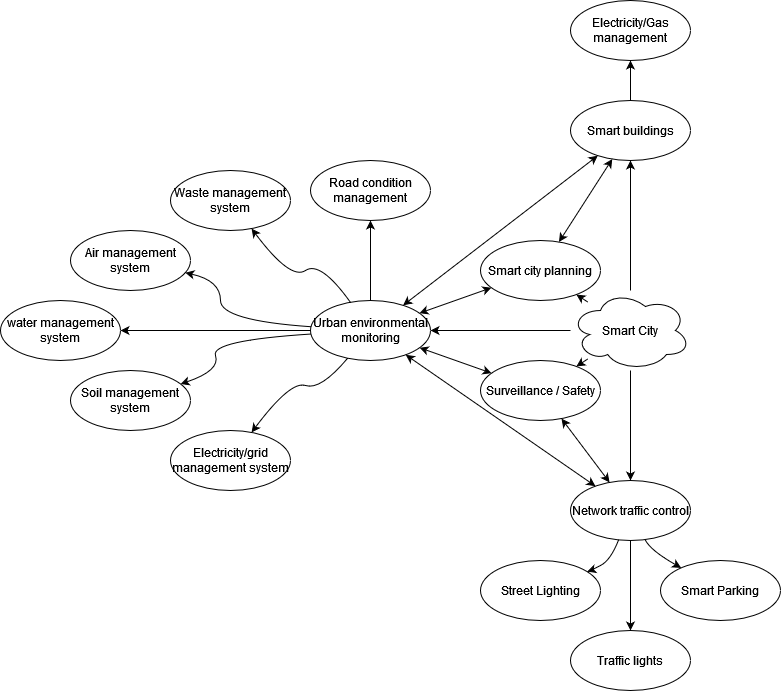
# Task 3 – Smart City

* Mind-map for the Smart City Design
  + Network traffic control, streetlights -
  + Smart parking -
  + Smart transportation / mobility
  + Smart bins -
  + Safety / Security / surveillance -
  + Smart buildings/offices (heating / lighting)
  + Urban Environmental Monitoring (air, noise, water, electricity, waste monitoring, soil humidity)
  + Disability aids
  + Smart city planning
* Define 3 context / use cases, WITH activity diagrams (activity diagrams show message flow from one activity to another)
  + Network Traffic Control
  + Streetlights
  + Smart bins
* Define analysis architecture (context diagram) with block diagrams (at least 10 blocks). Describe the main interaction with the environment and sequence diagrams.

## Smart City Requirements

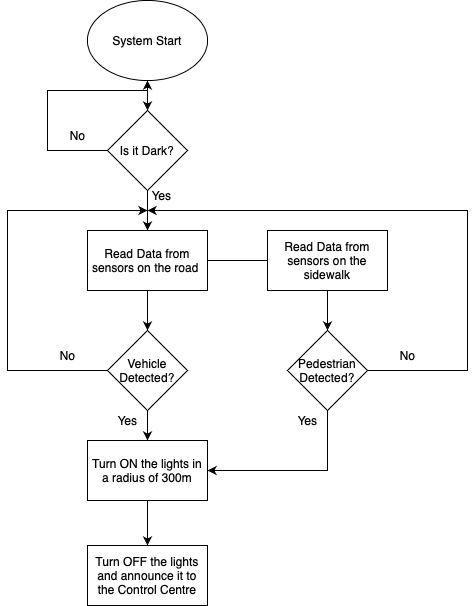
|  |  |
| --- | --- |
| Requirement ID | Description |
| R01 | Vehicles within the transportation network should transmit a location feed to enable smart traffic management |
| R02 | Smart streetlights and traffic lights should enter low-power mode when no vehicles are nearby |
| R03 | Parking should report availability to enable more efficient utilisation and less time required to find a park |
| R04 | Public transport services should make available their data using the Realtime GTFS data specification (Google, 2022) |
| R05 | Rubbish services can increase efficiency by providing smart bins that sense their fill level. This will allow maintenance services to be managed more efficiently and ensure a cleaner city |
| R06 | Surveillance should be provided in areas as required and monitored by AI algorithms to increase safety, security and help manage incidents |
| R07 | Buildings should aim to incorporate smart lighting and heating to increase energy efficiency. This could include various sensors to only enable sectors currently occupied by workers, while switching off areas not occupied |
| R08 | Smart environmental monitoring should be implemented by collating various sources of data into a central database. This could be used to ensure high quality environmental characteristics, such as air, noise, water, soil, and waste monitoring |
| R09 | Urban planning should be mindful of, and incorporate as much as possible, disability aids to ensure everyone can use infrastructure equally |
| R10 | Distances between housing and workplaces should be reduced as much as possible. This can reduce transportation network usage and vehicle pollution, while increasing the free time and work-life balance of the population |

# Mind Map of a Smart City as a Whole

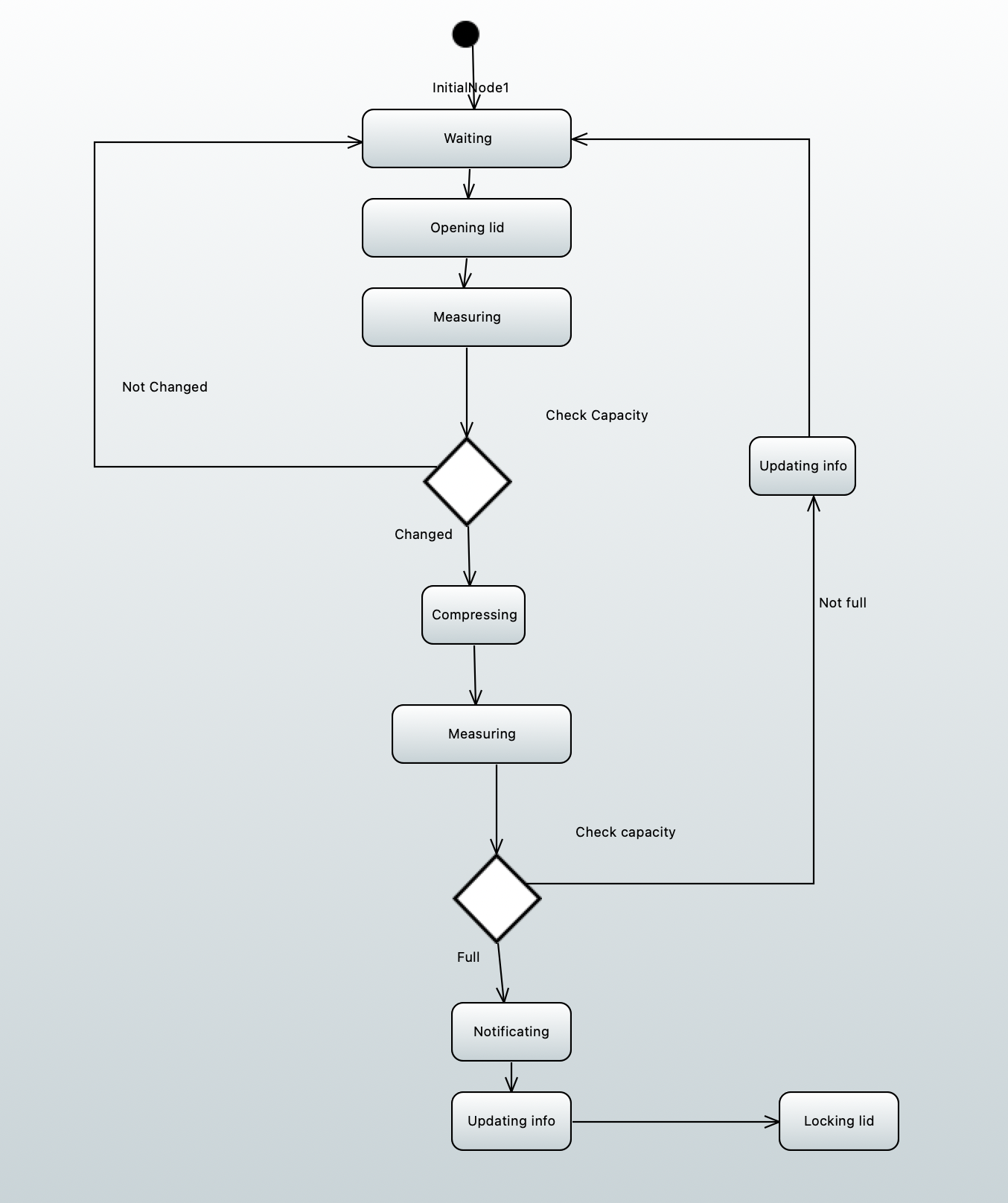


* Leave only the 3 use case diagrams in the mindmap
* Include all stakeholders

# Activity Diagram of a Smart Light Control Sub-System



# Activity Diagram of a Smart Garbage Sub-System



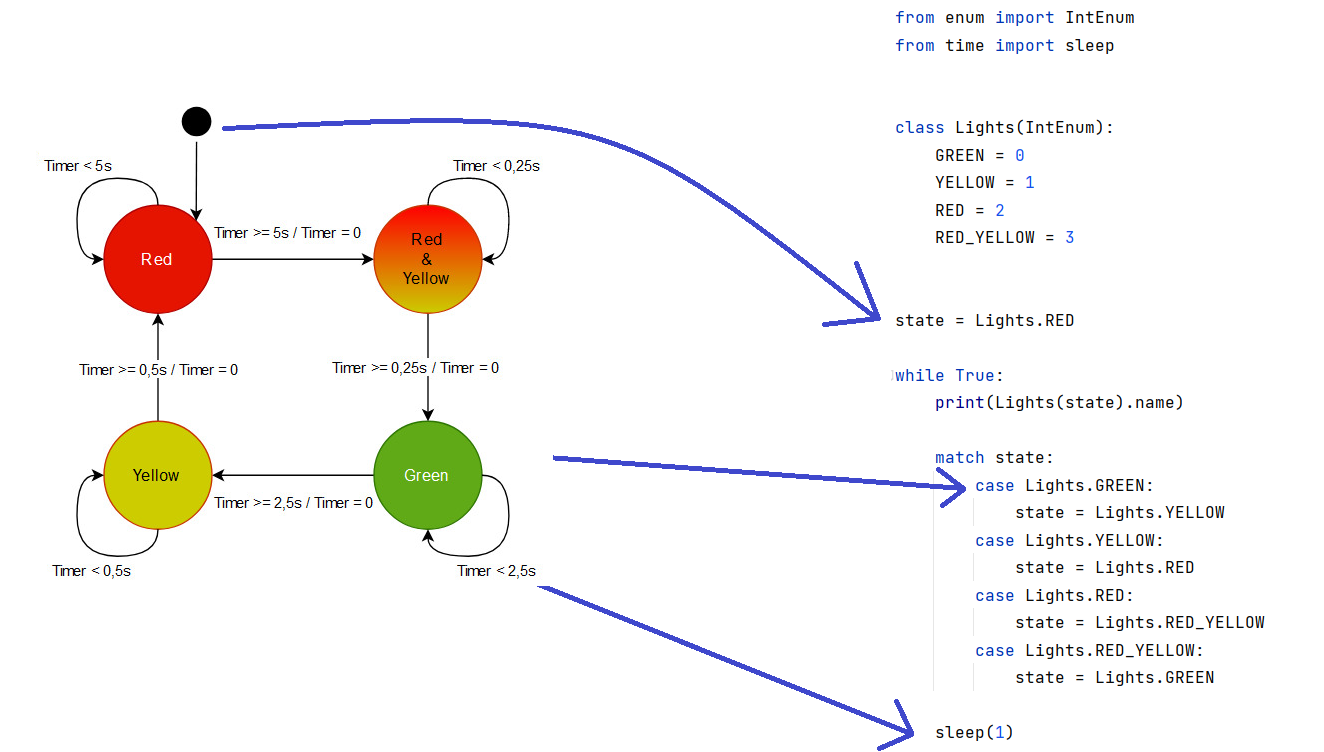
## Networked Traffic Control for Autonomous Cars

Diagram

Description automatically generated

## State Machine for a Single Traffic Light

*Specify trivial specification for state machine, show how it can be mapped in a structured way into code (c/java). – One line (lane?) trivial traffic light.*



## Lesson Notes:

Use case

* Overall use case
* Sub use case from overall use case

Intersection control (not a physical traffic light)

* Can handle multiple vehicles
* Co-ordination via messages
* Can be extended or related to smart lighting
* One intersection is connected to the next one
* Vehicles and pedestrians (via sensors?) communicate with the system

Event based system

* Send events to the intersection
  + E.g. I like to cross from South to North
  + Use this as input for turning on the lights
* Can be street sensors as well as vehicle messaging
* Does not necessarily need to include pedestrians

Keep it simple

* Just include more information e.g. receiving more sensor input and message data to make the system smart
* Integrate smart lighting with intersection control
* Consider pedestrians, only turn on the lights if…

Code simulation

* As written on the slides
* Part -> mapped to an embedded domain (e.g. TinkerCad or some other MCU)
* Should show something visual e.g. lights / traffic lights
* 2 or 3 intersections

Diagrams

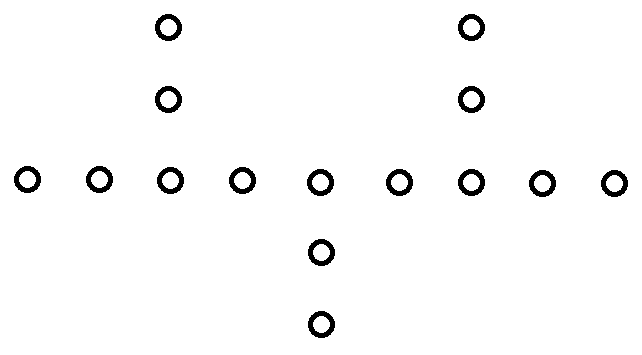
* Map the diagrams together to show how they are interconnected
* State machine
  + Delete sending block
  + Add more text to the lines
* Internal block diagram
  + Just for a part of the system
  + Show the flow (material / information / control)
  + Add the missing ports

To Do:

* Task 4
  + Finalize diagrams:
    - Allocation Diagram (Furkan Ali)
    - Internal block diagram (Ian)
    - State machine diagram (Elbek)
  + Implement **one block** into prototype hardware (tinkercad)
    - Check Simon's code
    - Implementation should exist of state machines
* Task 5
  + Research pyCPA
    - <https://pycpa.readthedocs.io/en/latest/>
    - What is wcet?
  + Is the implemented component(?) schedulable?
  + Inspection of models/code/documentation by each team member
  + Testing with use of Junit/Cunit framework

## Task 4 – TinkerCad Options

Smart **streetlights** with 3 intersections. A vehicle can pass from left to right, with the lights becoming bright as the vehicle passes. As it passes the intersection, all surrounding lights turn on.



Smart **traffic control**. Two lanes, 3-4 vehicles. Only 1 vehicle can be in an intersection at once. The vehicles can drive around randomly, following a track or do figure eights etc.

Shape, square

Description automatically generated

Task 5 – Scheduling

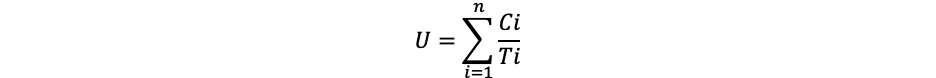
We have a one core processor microcontroller for our demonstration which is an Arduino UNO.

We will break down to the level of scheduling the tasks that are included in this component of the whole Smart City Design.

For calculating the **wcet/computation time (Ci)**, we have done many tries on the hardware and software and have come up with the results as shown in the table below. As for the **arrival time(Ai)**, it is worth mentioning that because our hardware is 1-core processor, the tasks will come one after the other and not at the same time  
   
We can also calculate WCRT with the help of “PYCPA” based on the documentation found: <https://pycpa.readthedocs.io/en/latest/spp_example.html>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Clear model** | **Add street/intersection lighting to model** | **Add vehicles to model** | **Update LEDs** | **Update vehicle positions** |
| Ai | 0ms | 0.5ms | 1ms | 1.5ms | 4ms |
| Ci | 0.39ms | 0.21ms | 0.06ms | 2.06ms | 0.36ms |
| Di | 5.5ms | 5.5ms | 5.5ms | 5.5ms | 5.5ms |
| Ti | 5.5ms | 5.5ms | 5.5ms | 5.5ms | 5.5ms |

Show that the implemented component with its set of tasks is scheduable (EDF or RMS or both).   
  
To give an answer to this we must do some calculations based on the values of the table above.



This formula calculates the utilization of the processor.

U = 0.39/5.5 + 0.21/5.5 + 0.06/5.5 + 2.06/5.5 + 0.36/5.5 = 0.56   
  
For EDF scheduling, Schedulability test is:

*U ≤ 1*

So, we can say that the component is **schedulable in EDF**.

For RMS scheduling, for the Schedulability test we need one more condition to be passed:

*U ≤ B(n)*

Where B(n) is the Utilization Bound and is calculated:

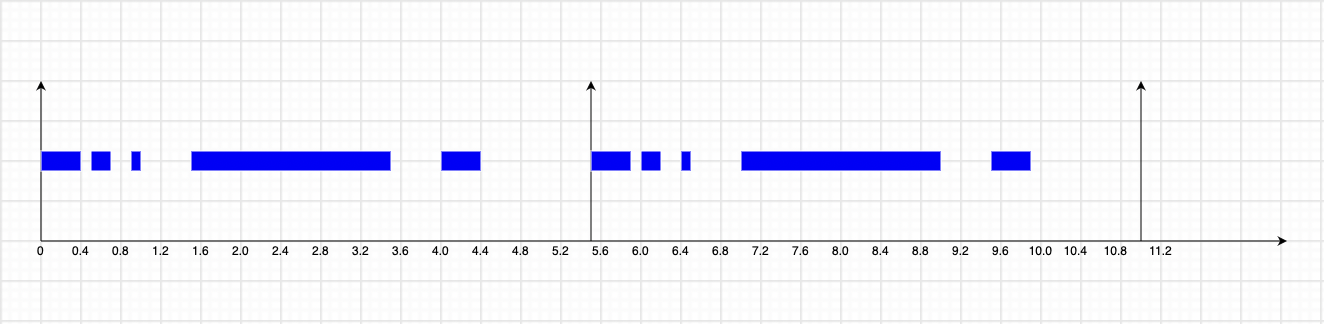
*B(n) = n\*(21/n-1)*

The variable n is the number of tasks.

B(n) = 5\*(21/5 - 1) = 0.743

As we can see the component is also **schedulable in RMS**.

Graphical representation of the component



Constraint Diagram  
