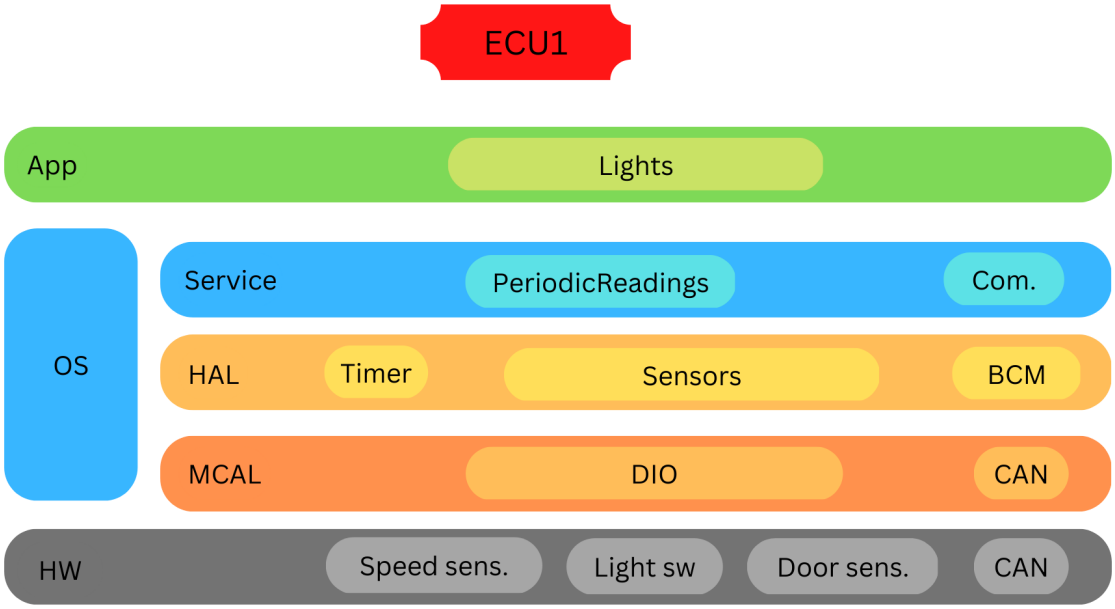
Static design

ECU1:

1) Layered architecture



2) Components and modules

- Timer
- DIO
- CAN

3) APIs for each module as well as a detailed description for the used typedefs

//CAN.h:

//CAN.c:

```
void CanInit();
void CanSend();
void CanReceive();
```

//DIO.h:

```
enum Port{portA, portB, portC, portD};
enum Pin{pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7};
enum value{False, True};
enum state{ouput, input};
```

//DIO.c:

```
void DioInit(Port, Pin, state);
void DioConfig(Port, Pin, state);
void DioSet(Port, Pin, value);
value Dioget(Port, Pin);

//BCM.c:

//Timer.h
//Timer.c
void TimerInit();
void TimerSet();
uint32_t TimerStatus();

//Sens.h
typedef struct {
        Port port;
        Pin pin;
} sens;
def spd sens
def lit sens
def door sens
//Sens.c
void SensInit();
value SensRead();
```

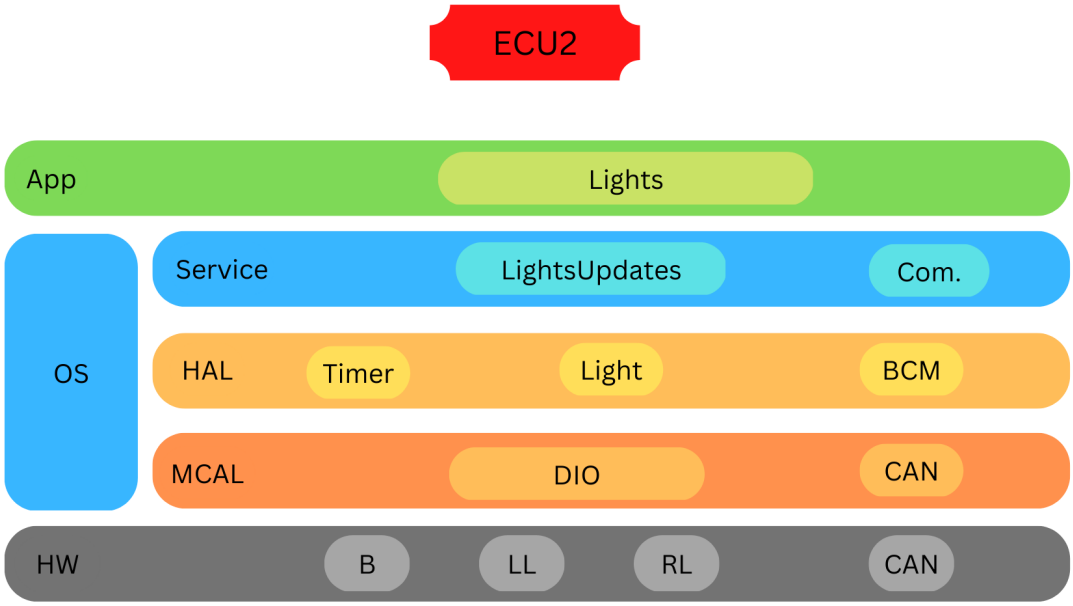
4) Folder structure

- ECU1
    - App
        - LightUpdates.h
        - LightUpdates.c
    - Hal
        - BCM.h
        - BCM.c
        - Timer.h
        - Timer.c
    - Mcal
        - CAN.h
        - CAN.c
        - DIO.h
        - DIO.c
    - Service
        - Sensors.h
        - Sensors.c
        - Com.h

- Com.c

ECU2:

1) Layered architecture



2) Components and modules

- Timer
- DIO
- CAN

3) APIs for each module as well as a detailed description for the used typedefs

//CAN.h:

//CAN.c:

```
void CanInit();
void CanSend();
void CanReceive();
```

//DIO.h:

```
enum Port{portA, portB, portC, portD};
enum Pin{pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7};
enum value{False, True};
enum state{ouput, input};
```

//DIO.c:

```
void DioInit(Port, Pin, state);
```

void DioConfig(Port, Pin, state);
void DioSet(Port, Pin, value);
value Dioget(Port, Pin);

//BCM.c:

//Timer.h
//Timer.c
void TimerInit();
void TimerSet(uint32_t);
void TimerState();

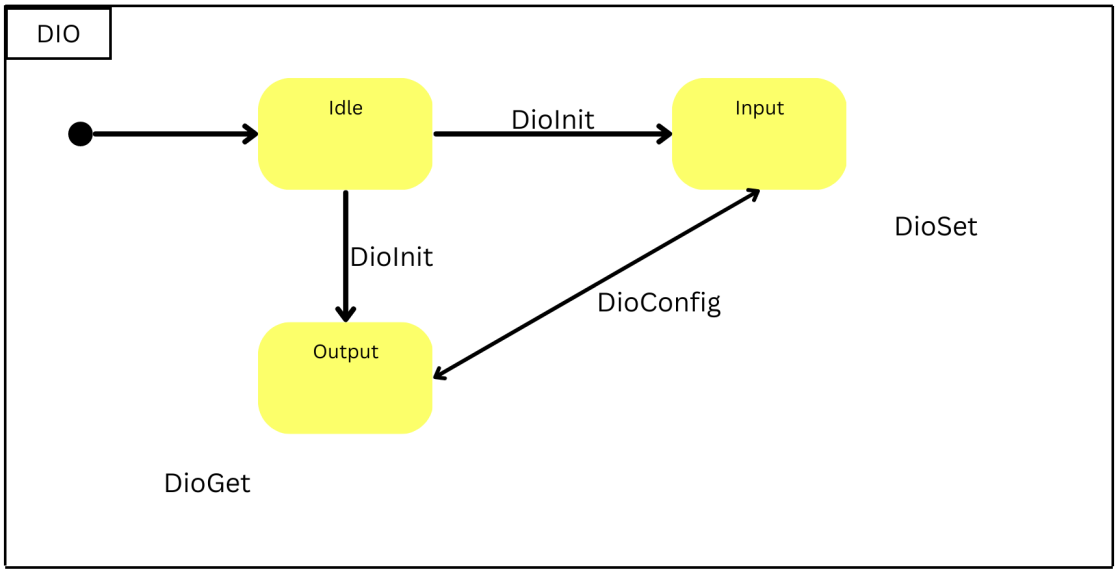
4) Folder structure

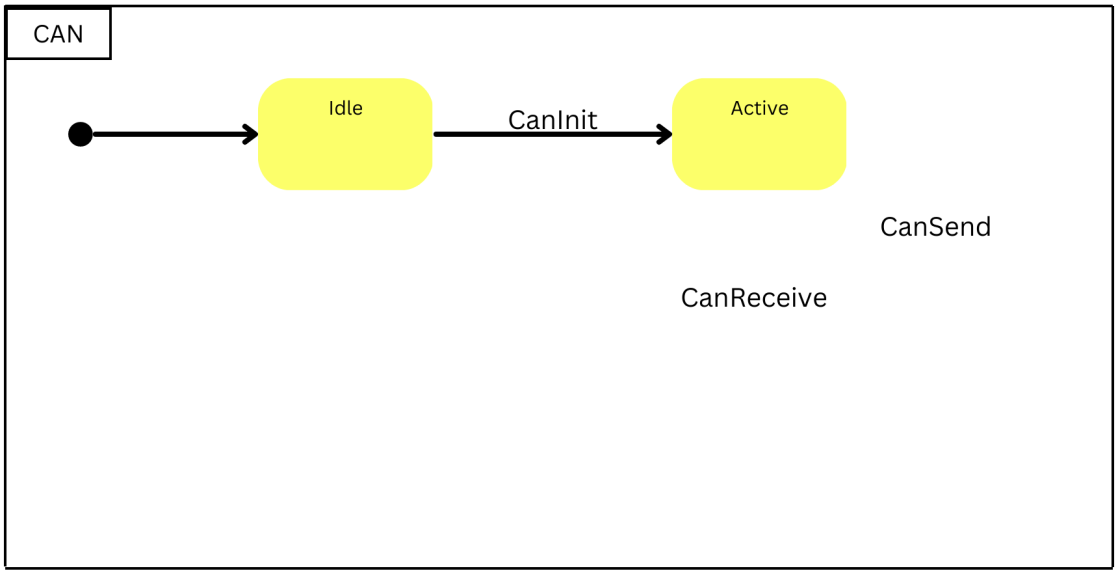
- ECU2
  - App
    - Lights.h
    - Lights.c
  - Service
    - LightsUpdates.h
    - LightsUpdates.c
    - Com.h
    - Com.c
  - Hal
    - Timer.h
    - Timer.c
    - Light.h
    - Light.c
    - BCM.h
    - BCM.c
  - Mcal
    - CAN.h
    - CAN.c
    - DIO.h
    - DIO.c

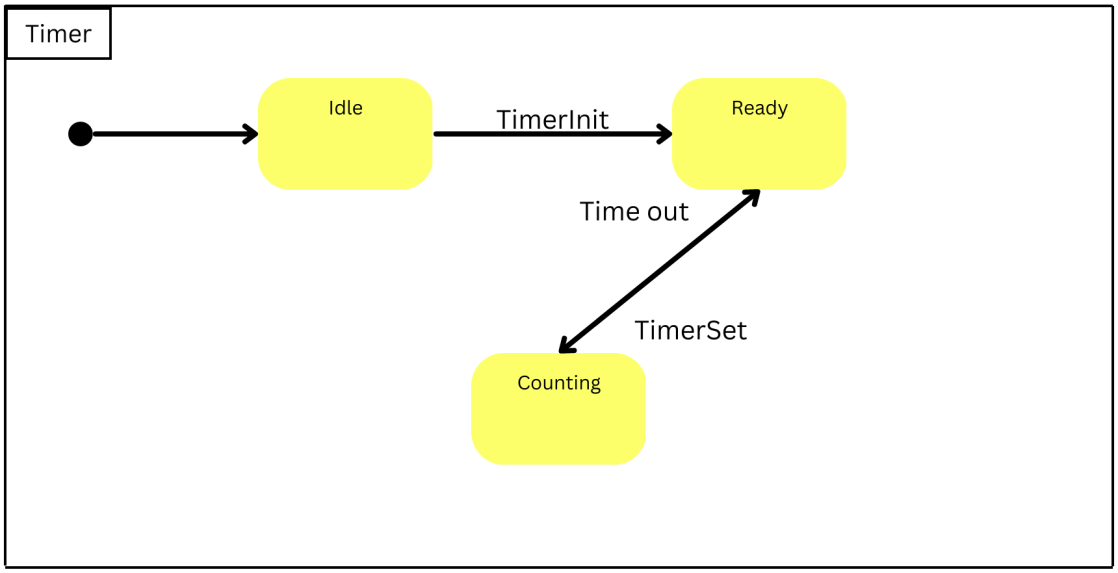
Dynamic design

ECU 1:

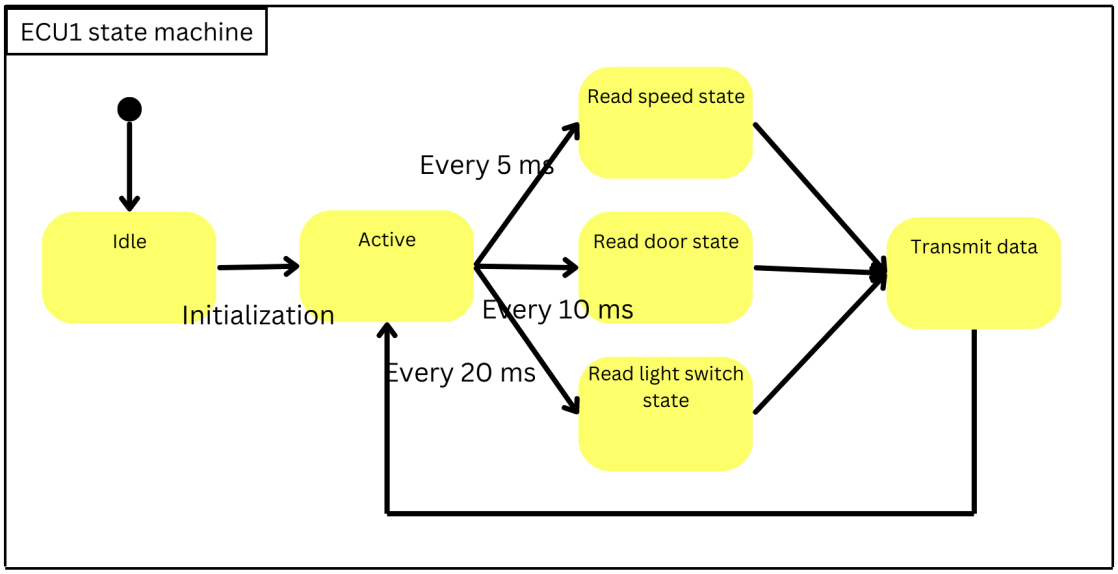
1) Draw a state machine diagram for each ECU component


DIO
Idle
DioInit
Input
DioSet
DioInit
DioConfig
Output
DioGet

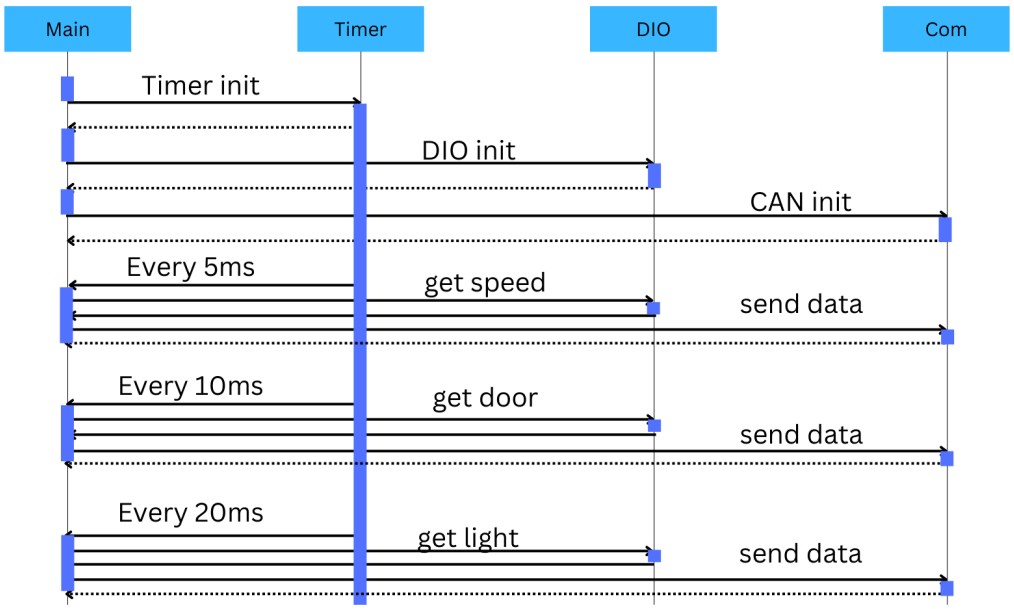

CAN
Idle
CanInit
Active
CanSend
CanReceive



2) Draw a state machine diagram for the ECU operation



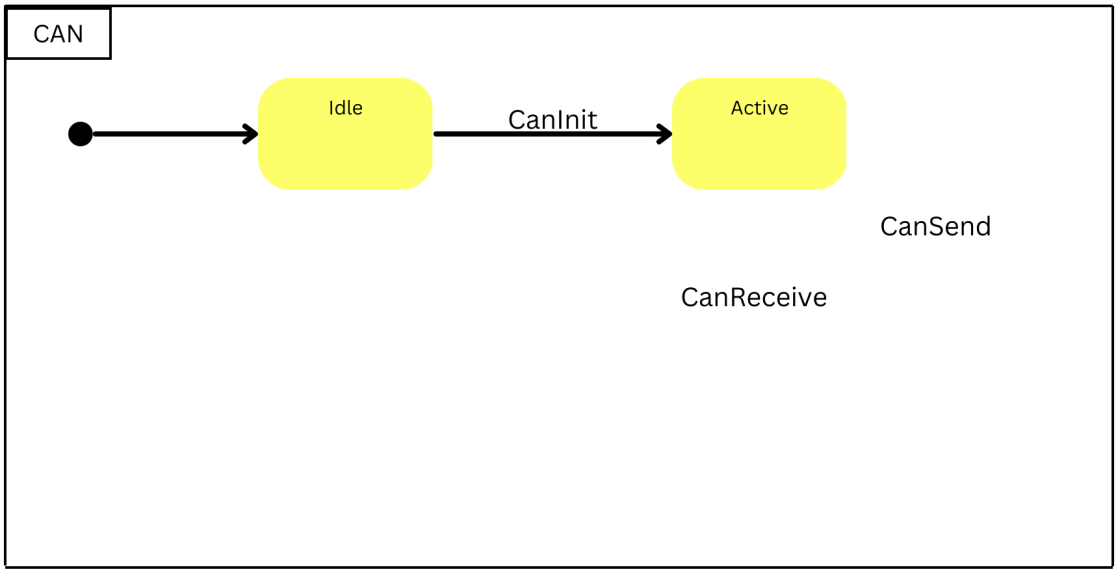
3) Draw the sequence diagram for the ECU

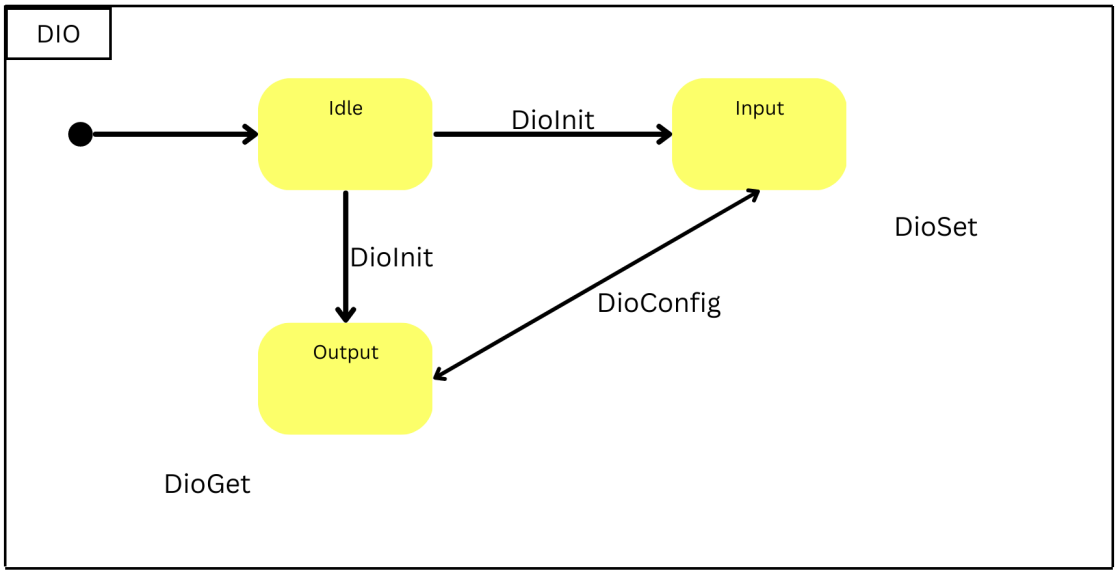


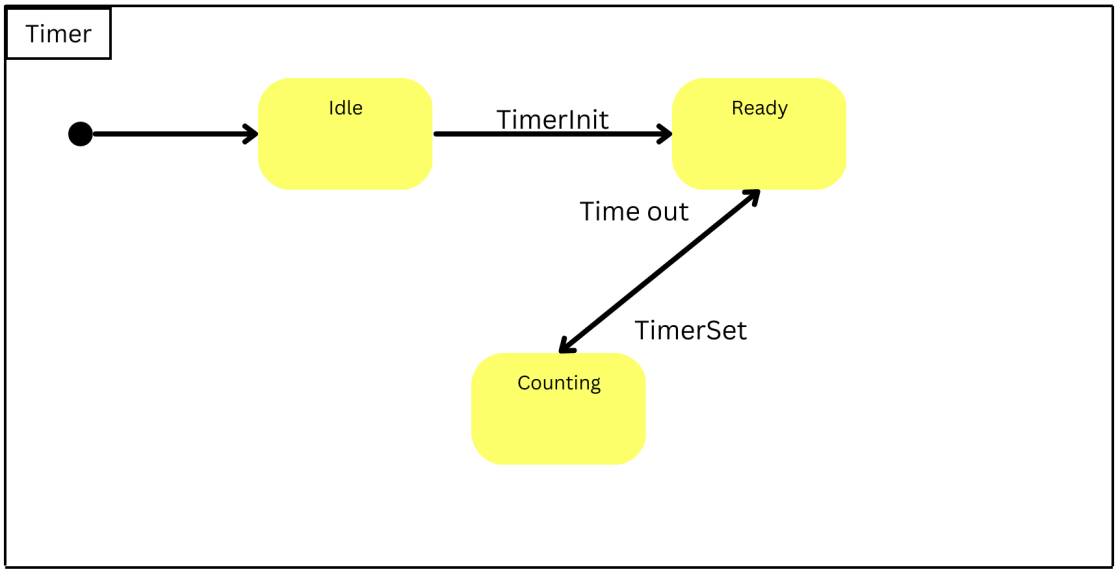
4) Calculate CPU load for the ECU
Hyper period= 20 ms.
CPU load= (4*speed task + 2*door task+ 1*Light switch task)/20

ECU 2:

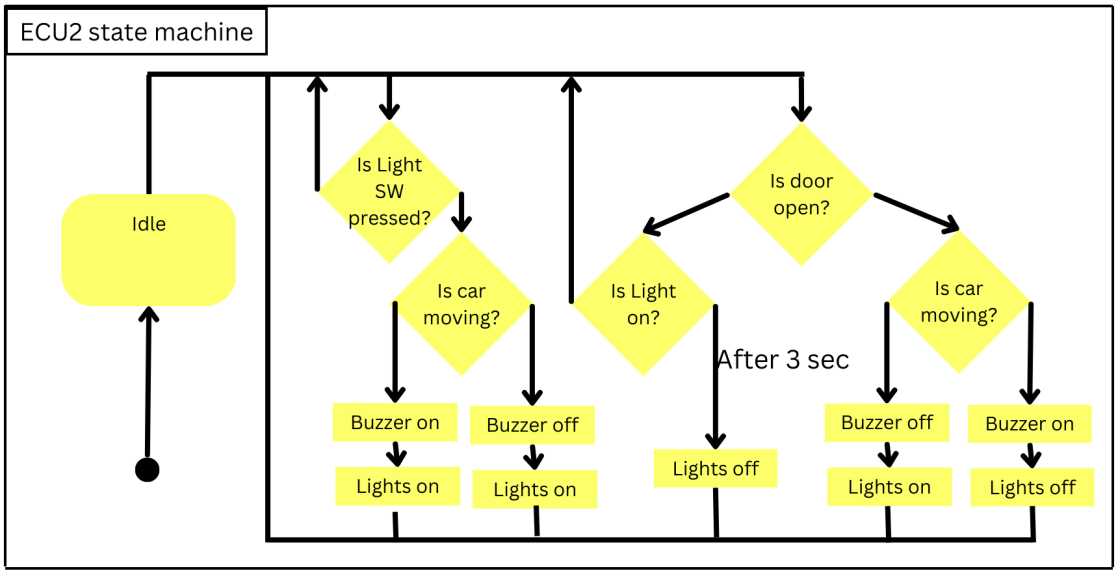
1) Draw a state machine diagram for each ECU component


CAN
Idle
CanInit
Active
CanSend
CanReceive

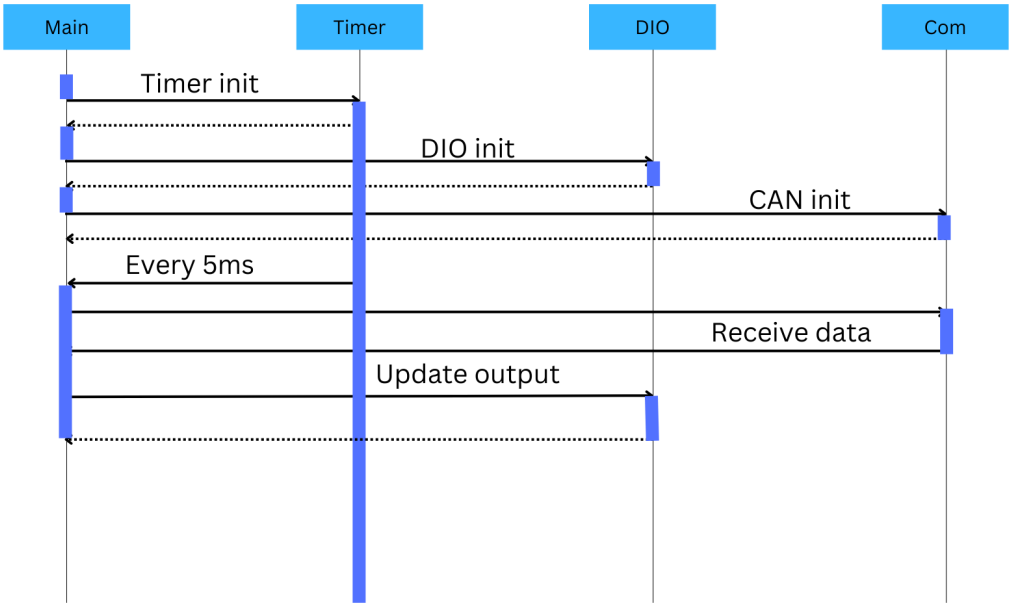

DIO
Idle
DioInit
Input
DioSet
DioInit
DioConfig
Output
DioGet



2) Draw a state machine diagram for the ECU operation



3) Draw the sequence diagram for the ECU



4) Calculate CPU load for the ECU
Hyper period = 5ms.
CPU load= Can receive task+ update output task / 5