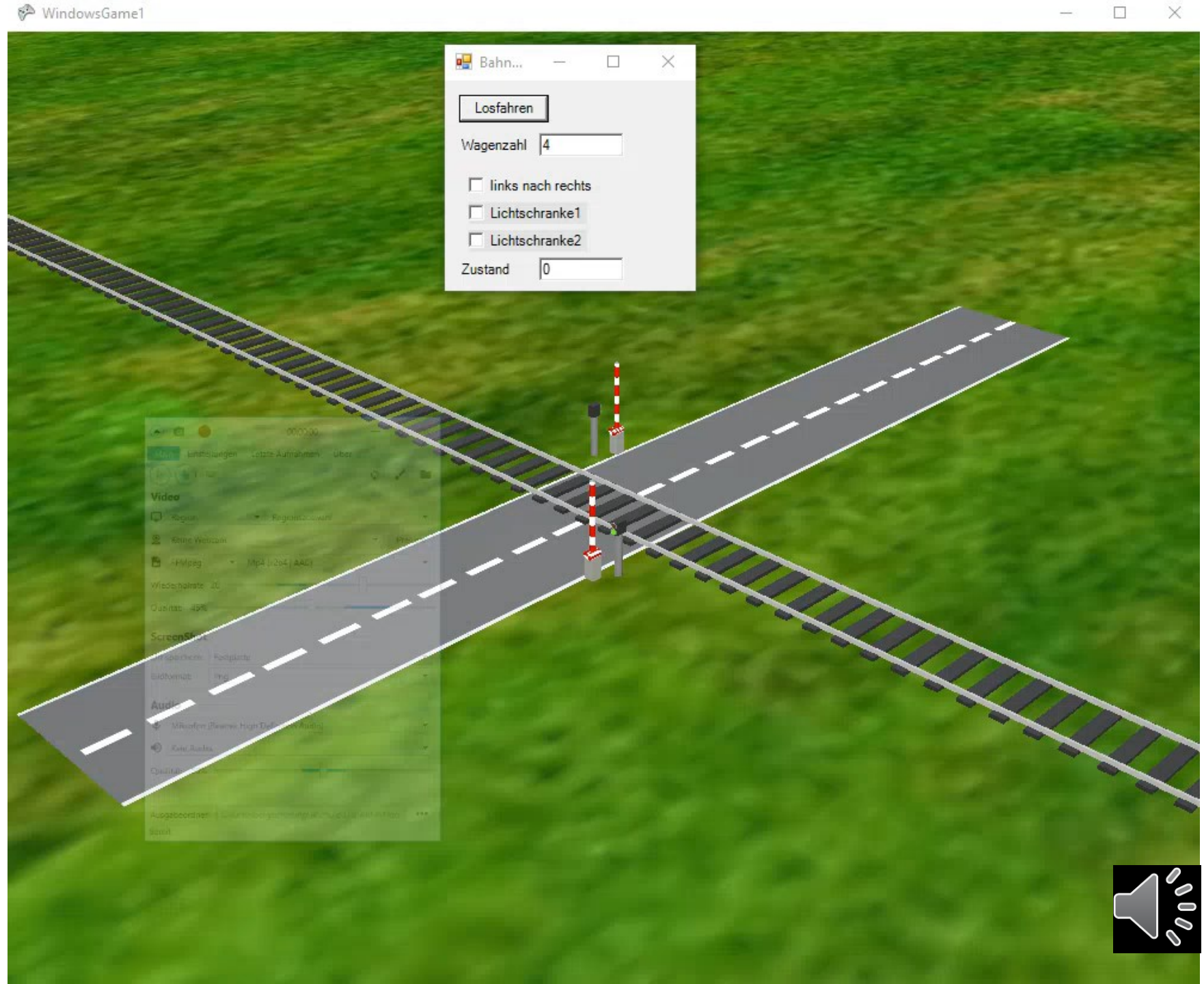
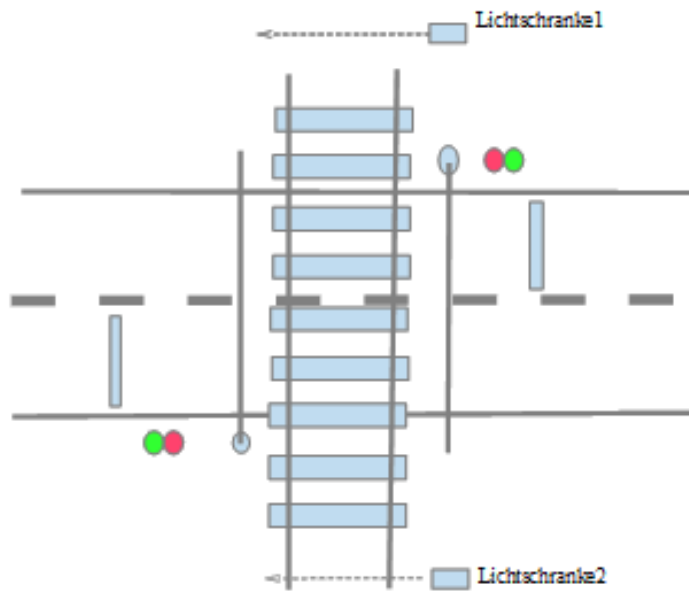


# Zustandsdiagramm

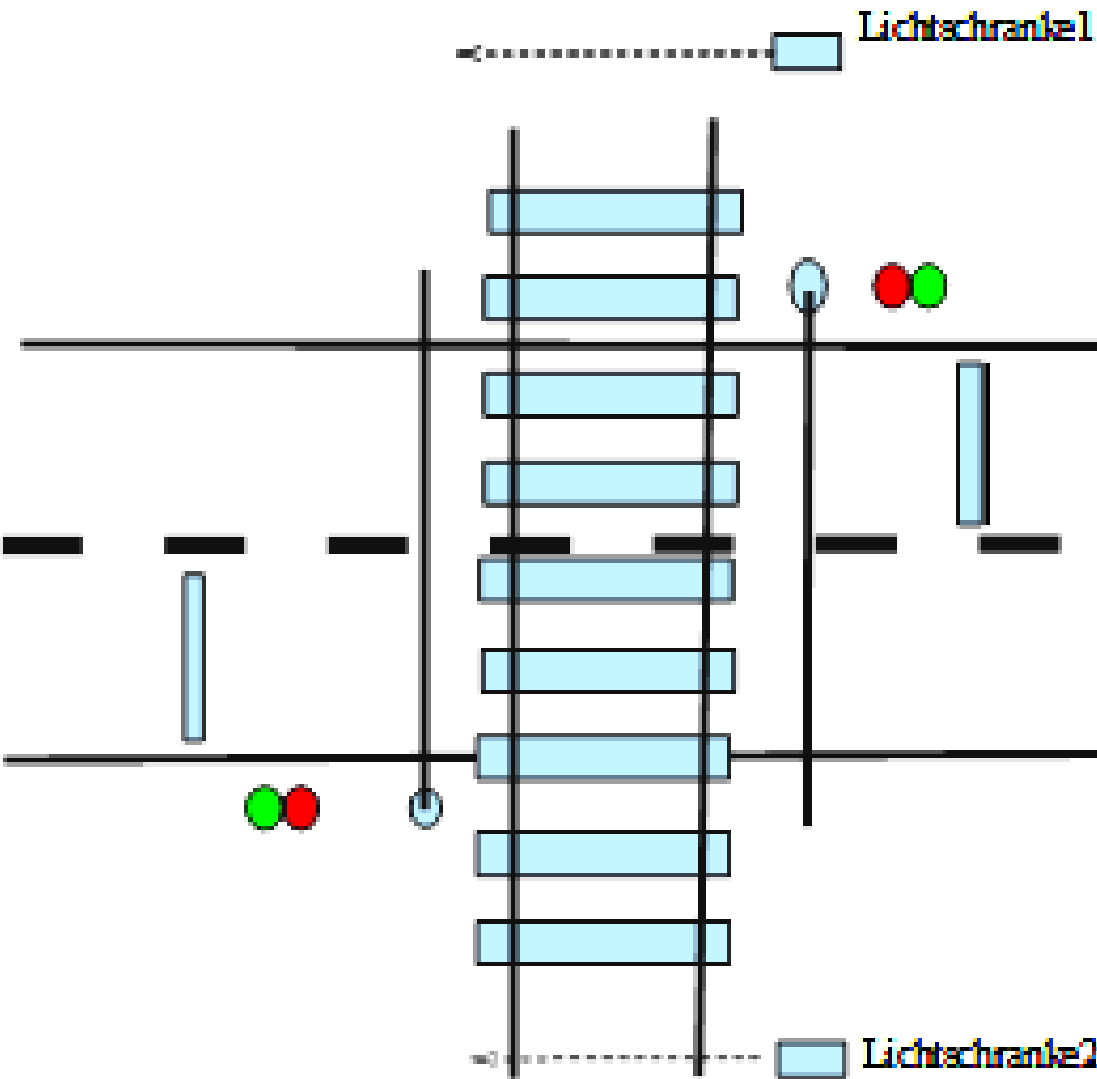
Bahnübergang automatisieren



# Zustandsdiagramm



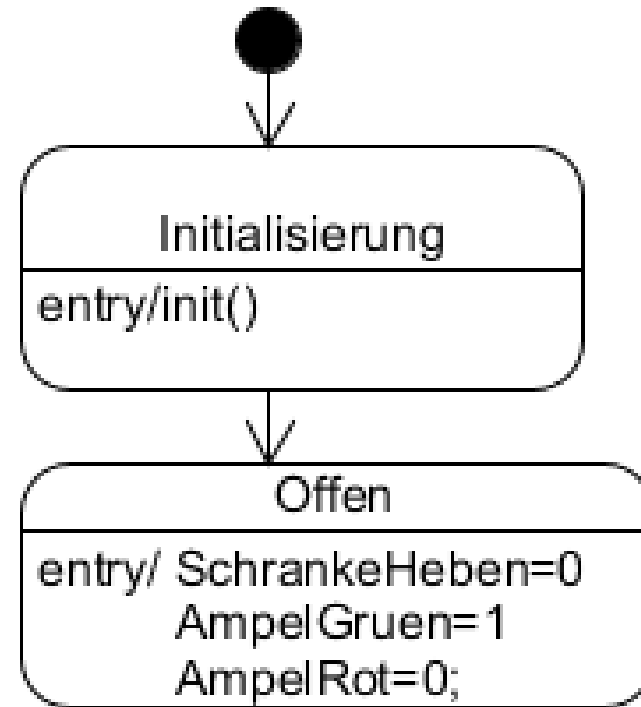
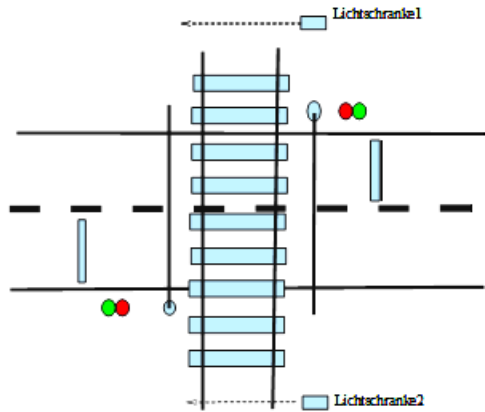
# Zustandsdiagramm



Signal	Art	Anschluss
SchrankeHeben	Ausgang	PC_0
SchrankeSenken	Ausgang	PC_1
AmpelRot	Ausgang	PC_2
AmpelGruen	Ausgang	PC_3
Lichtschränke1	Interrupt Eingang	PA_1
Lichtschränke2	Interrupt Eingang	PA_6
SchrankeOffen	Interrupt Eingang	PA_10
SchrankeZu	Interrupt Eingang	PB_0



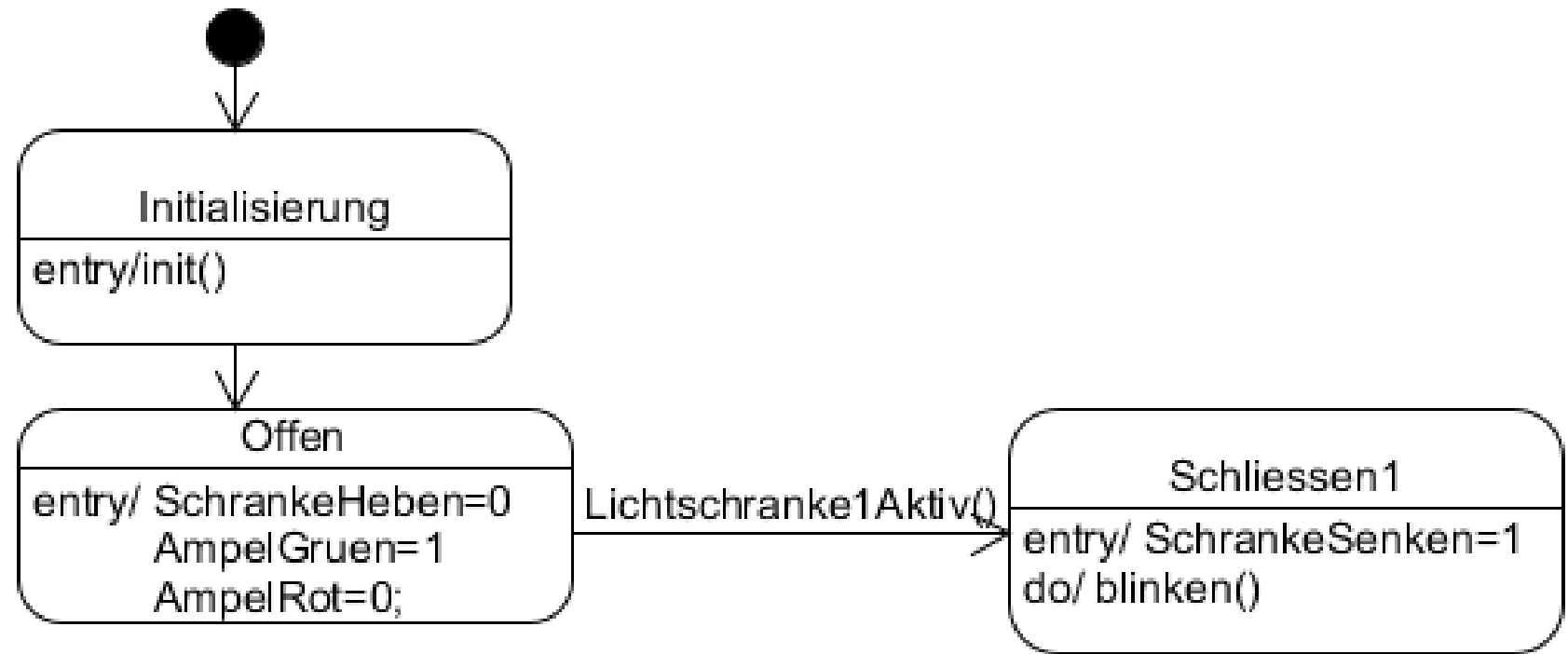
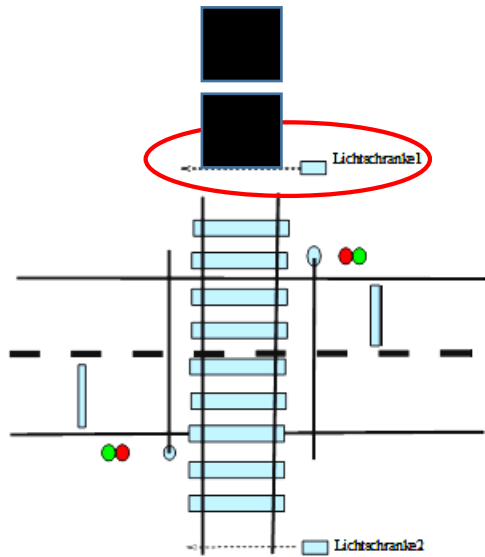
# Zustandsdiagramm



## Programmstart



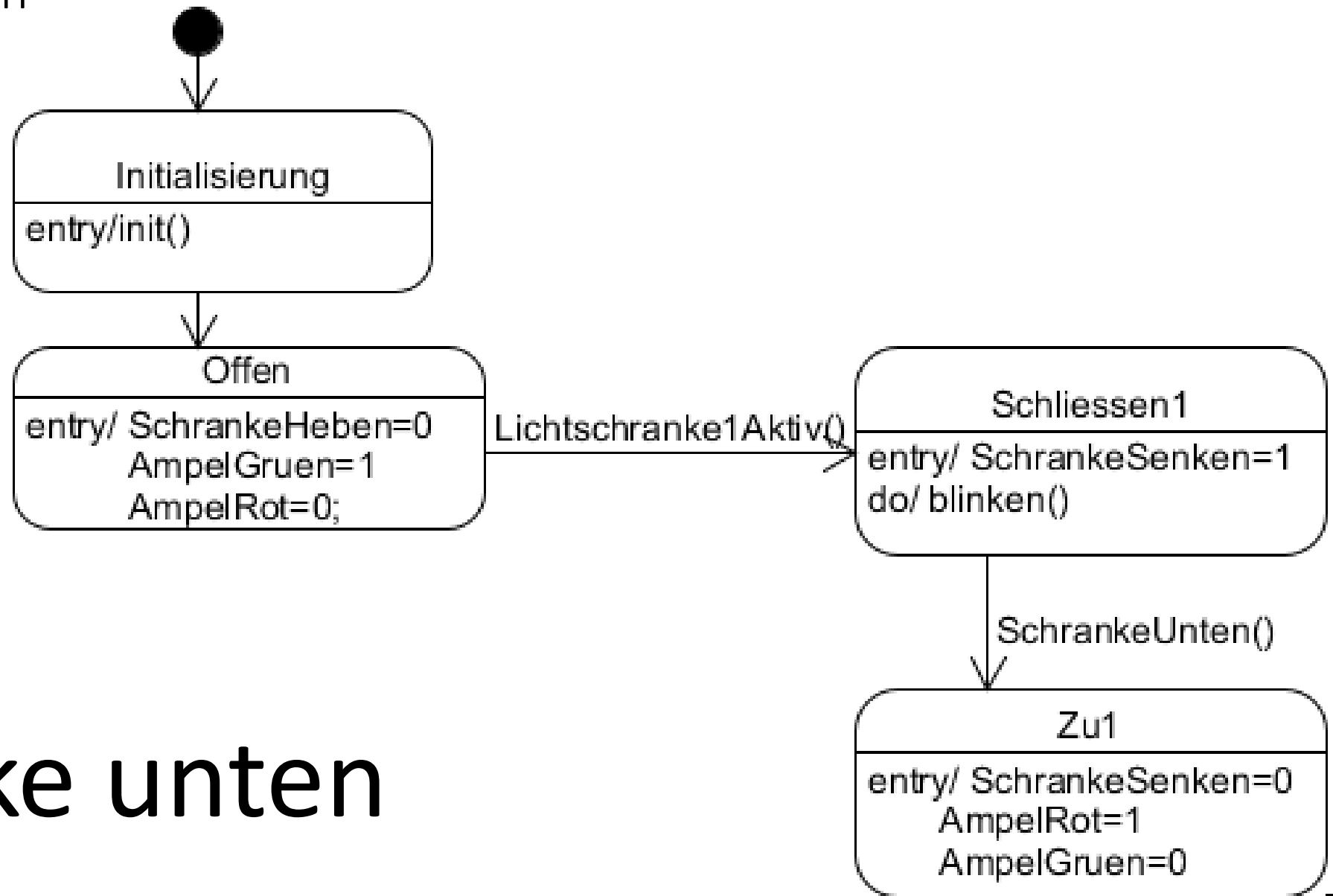
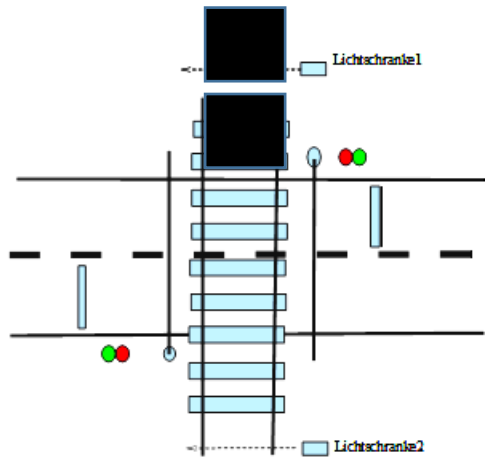
# Zustandsdiagramm



Zug erreicht  
Lichtschranke 1



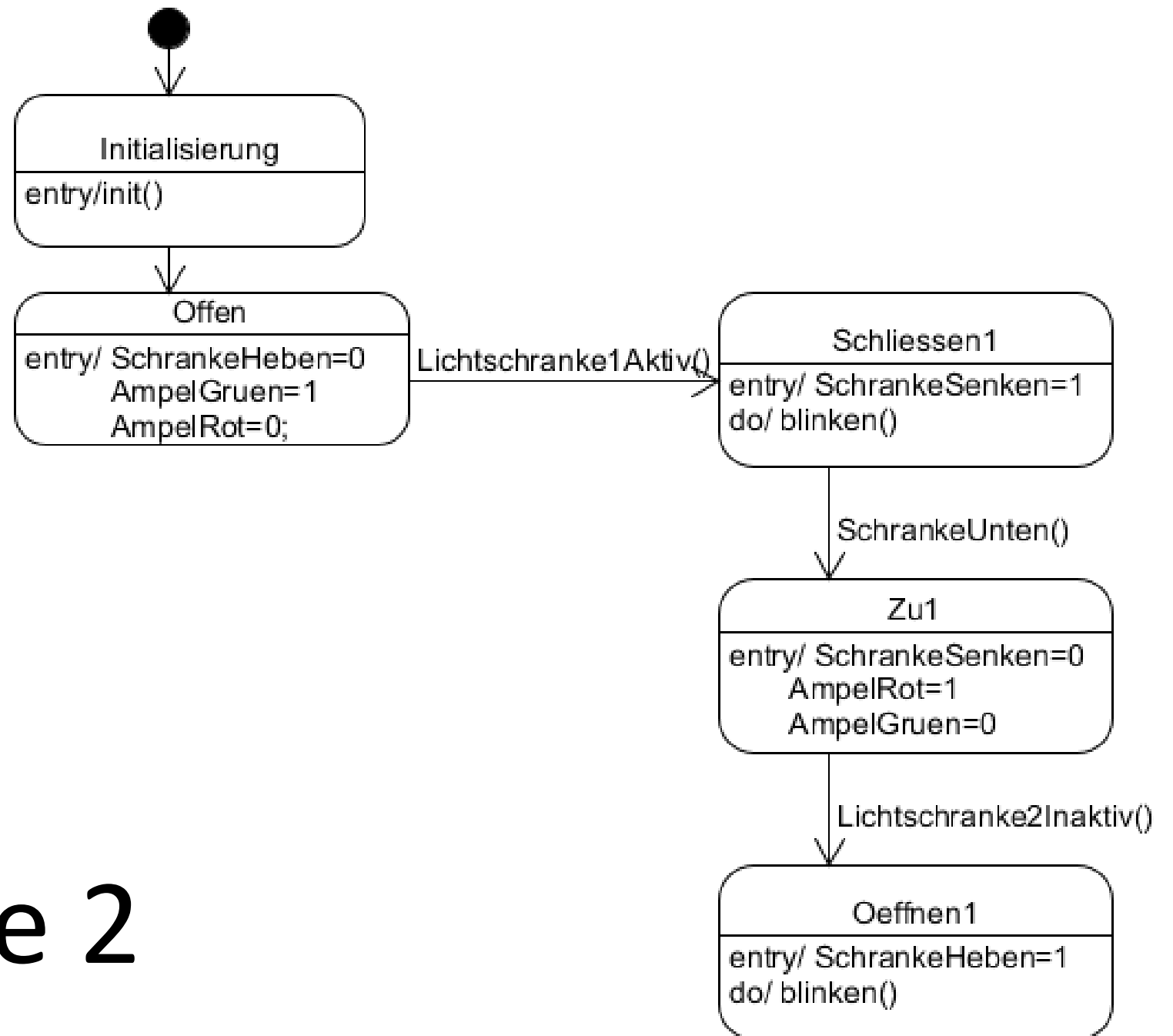
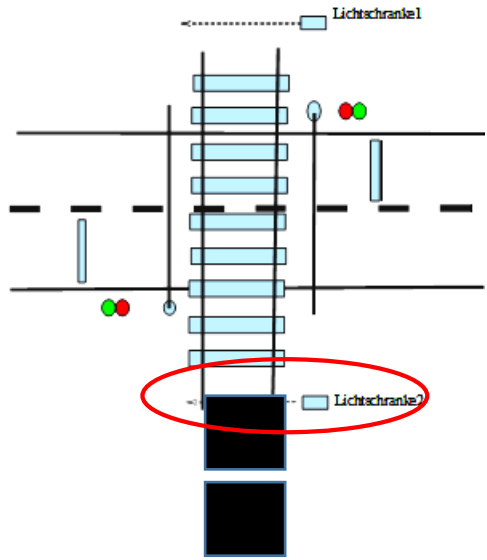
# Zustandsdiagramm



## Schranke unten

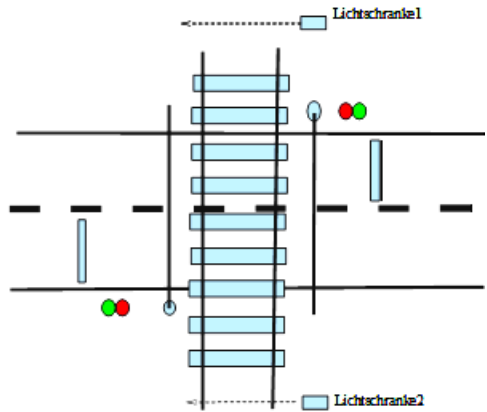


# Zustandsdiagramm

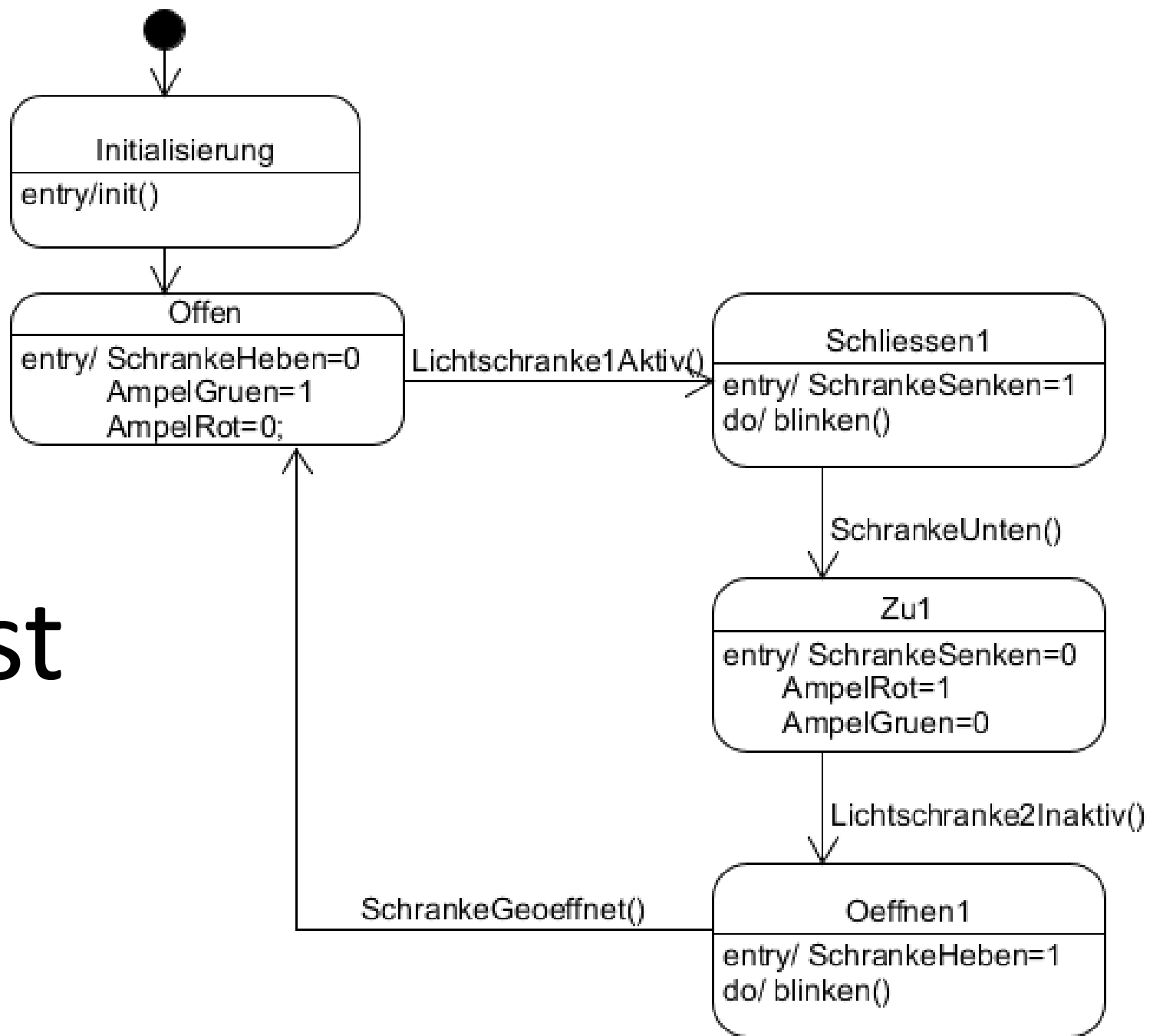


Zug verlässt  
Lichtschranke 2

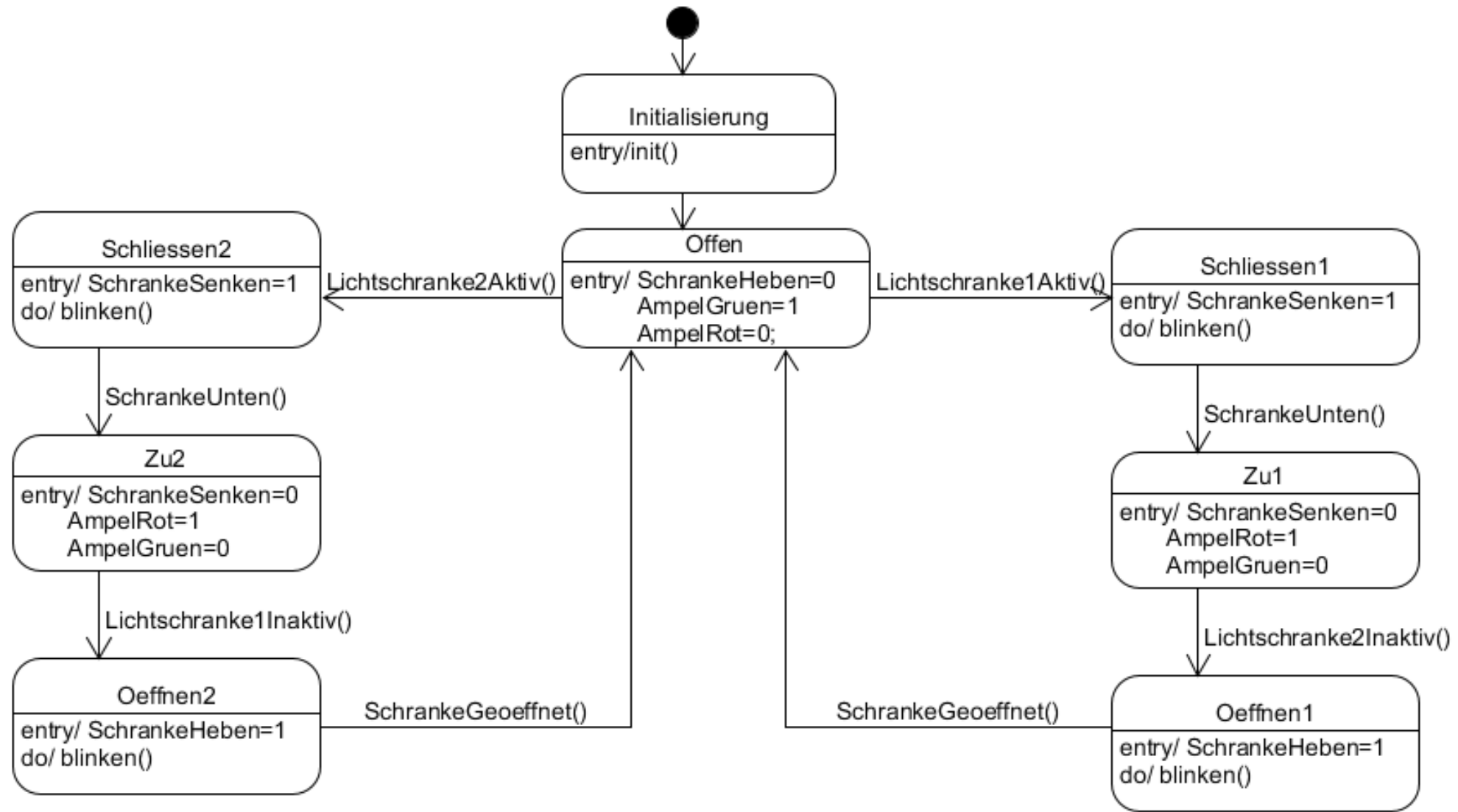
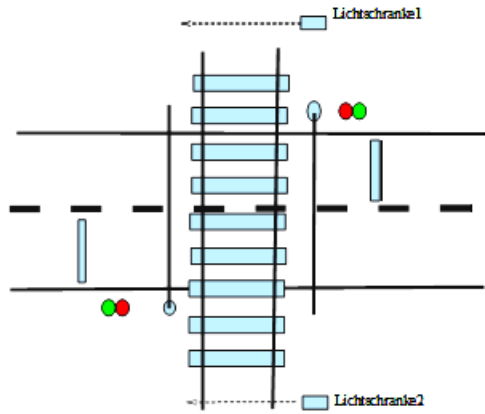
# Zustandsdiagramm



Die Schranke ist  
wieder offen



# Zustandsdiagramma



# Das komplette Programm

# Zustandsdiagramm

//Definition der Zustände

#define Initialisierung 0

#define Offen 1

#define Schliessen1 2

#define Zu1 3

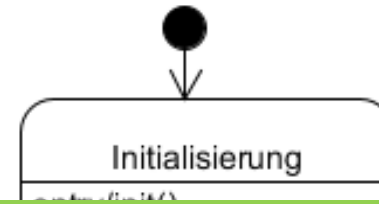
#define Oeffnen1 4

#define Schliessen2 5

#define Zu26

#define Oeffnen2 7

int zustand=Initialisierung



//Deklaration der Ein- und Ausgänge

DigitalOut SchrankeHeben(PC\_0);

DigitalOut SchrankeSenken(PC\_0);

DigitalOut AmpelRot(PC\_2);

DigitalOut AmpelGruen(PC\_3);

InterruptIn Lichtschranke1(PA\_1);

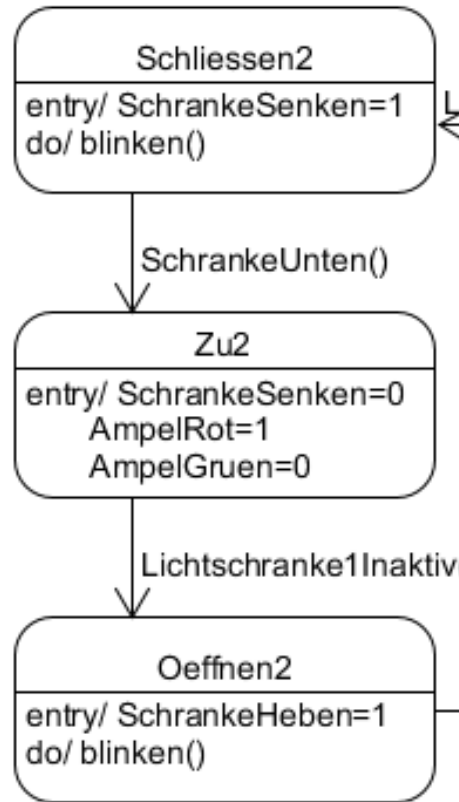
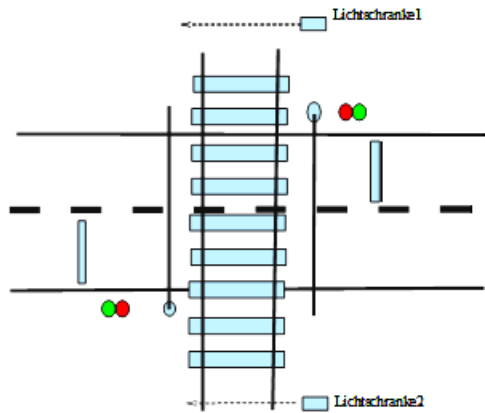
InterruptIn Lichtschranke2(PA\_6);

InterruptIn SchrankeOffen(PA\_10);

InterruptIn SchrankeZu(PB\_0);

# Implementierung

# Zustandsdiagramm



```
void init()
```

```
{
```

```
//Intialisierung der Ein- und Ausgänge
```

```
Lichtschranke1.mode(PullDown);
```

```
Lichtschranke2.mode(PullDown);
```

```
SchrankeOffen.mode(PullDown);
```

```
SchrankeZu.mode(PullDown);
```

```
Lichtschranke1.rise(&Lichtschranke1Aktiv);
```

```
Lichtschranke2.rise(&Lichtschranke2Aktiv);
```

```
Lichtschranke1.fall(&Lichtschranke1Inaktiv);
```

```
Lichtschranke2.fall(&Lichtschranke2Inaktiv);
```

```
SchrankeOffen.rise(&SchrankeGeoeffnet);
```

```
SchrankeZu.rise(&SchrankeUnten);
```

```
SchrankeHeben=0;
```

```
SchrankeSenken=0;
```

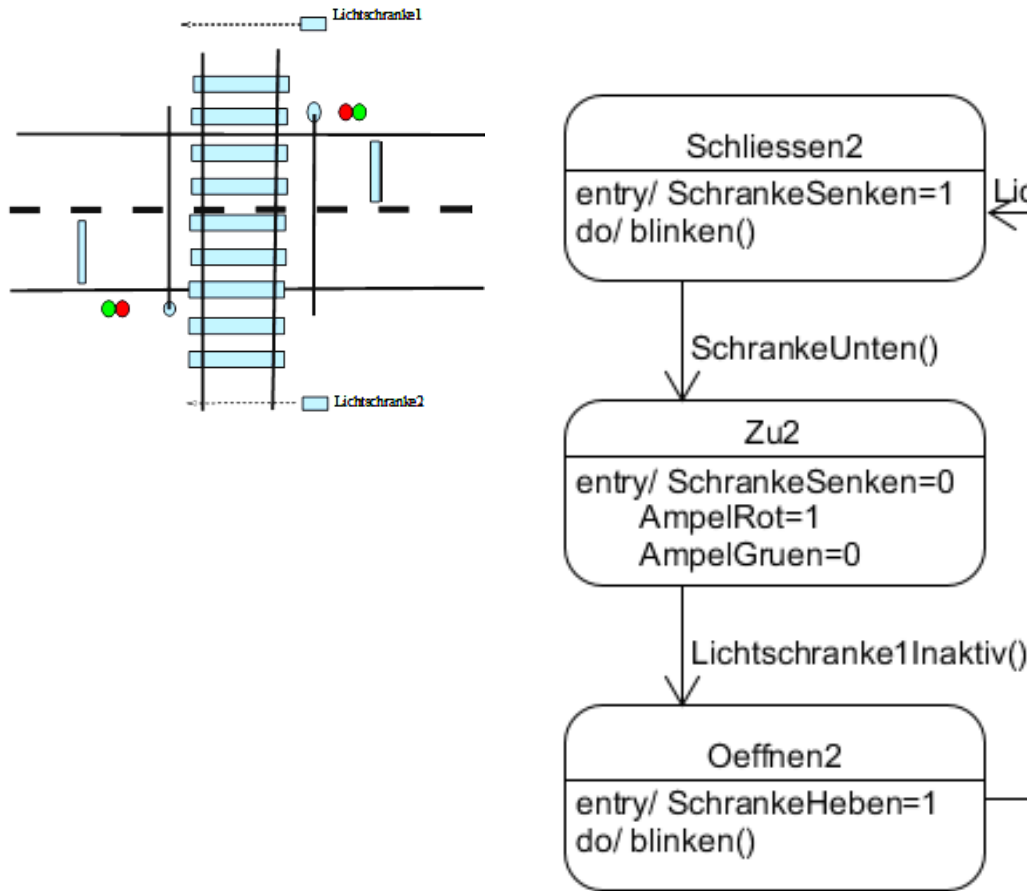
```
AmpelRot=0;
```

```
AmpelGruen=0;
```

```
}
```

# Implementierung

# Zustandsdiagramm

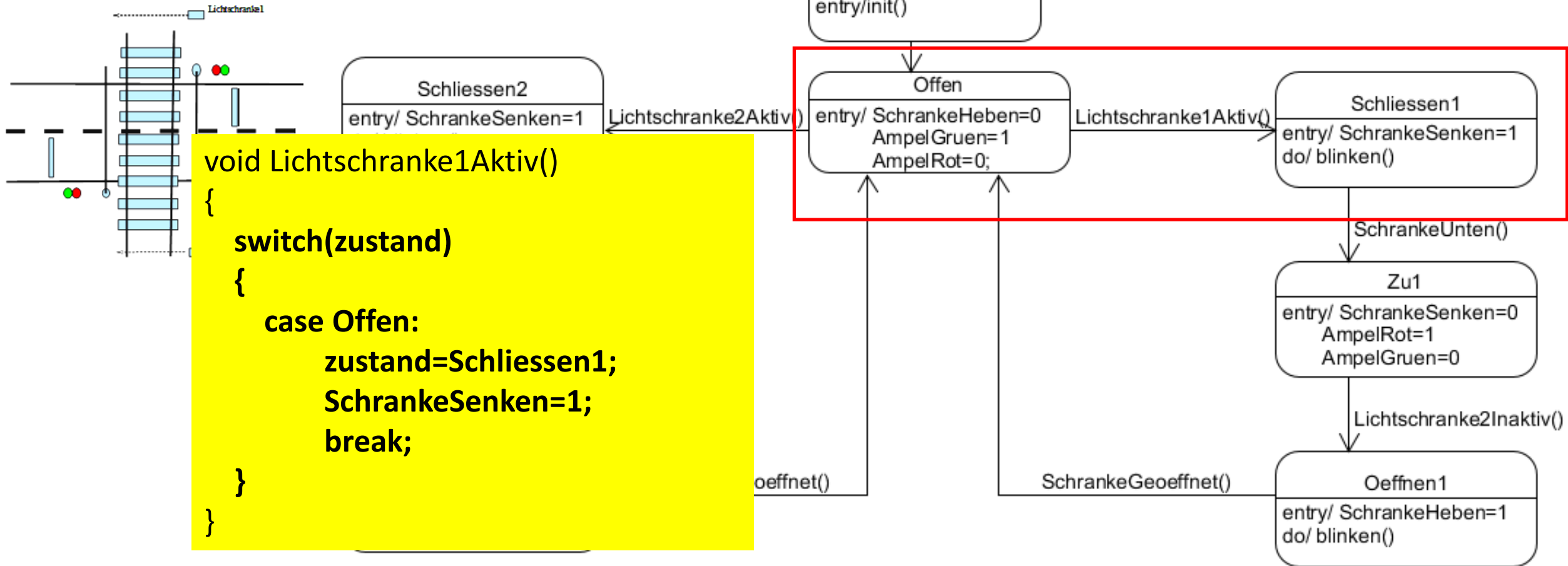


# Implementierung

```
void blinken()
{
    AmpelGruen=!AmpelGruen;
    AmpelRot=!AmpelGruen;
}

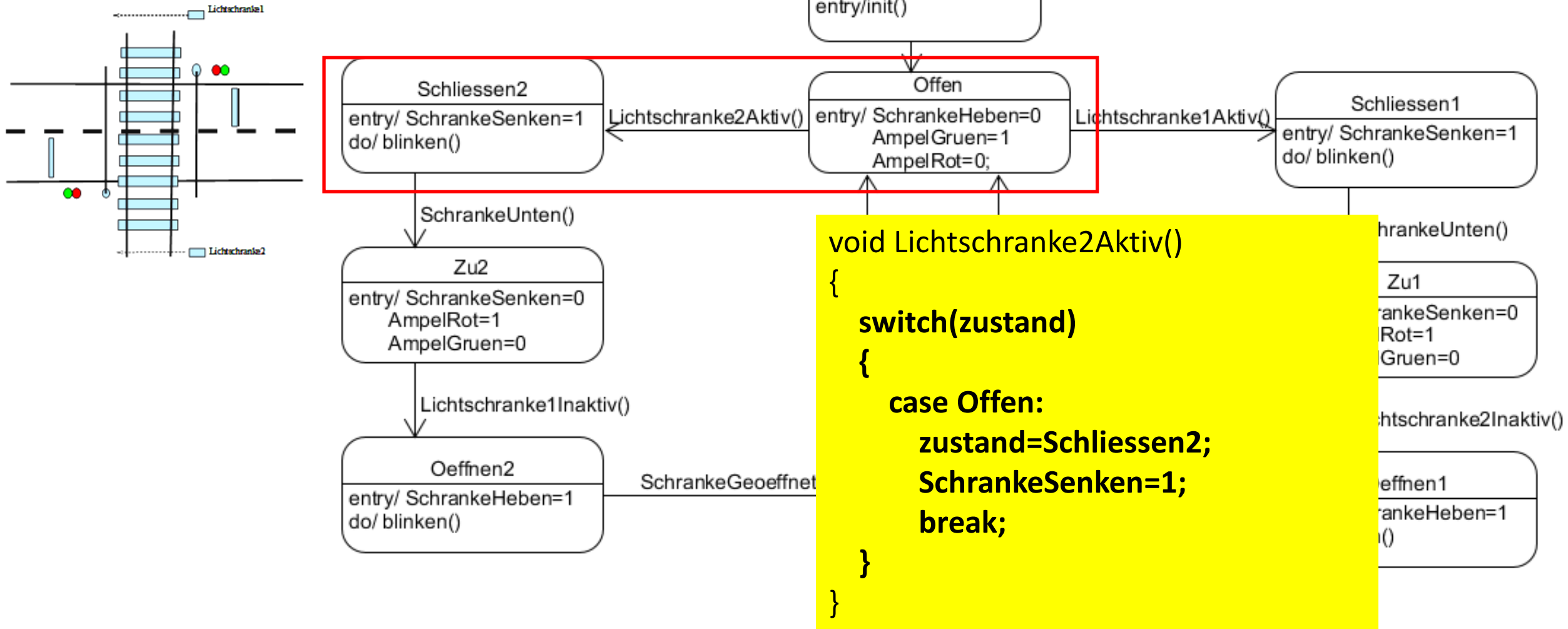
int main()
{
    init();
    //Wechsel in Zustand Offen
    zustand=Offen;
    AmpelGruen=1;
    AmpelRot=0;
    SchrankeHeben=0;
    while (true) {
        switch(zustand)
        {
            case Schliessen1:
            case Schliessen2:
            case Oeffnen1:
            case Oeffnen2: blinken(); break;
        }
        thread_sleep_for(200);
    }
}
```

# Zustandsdiagramm



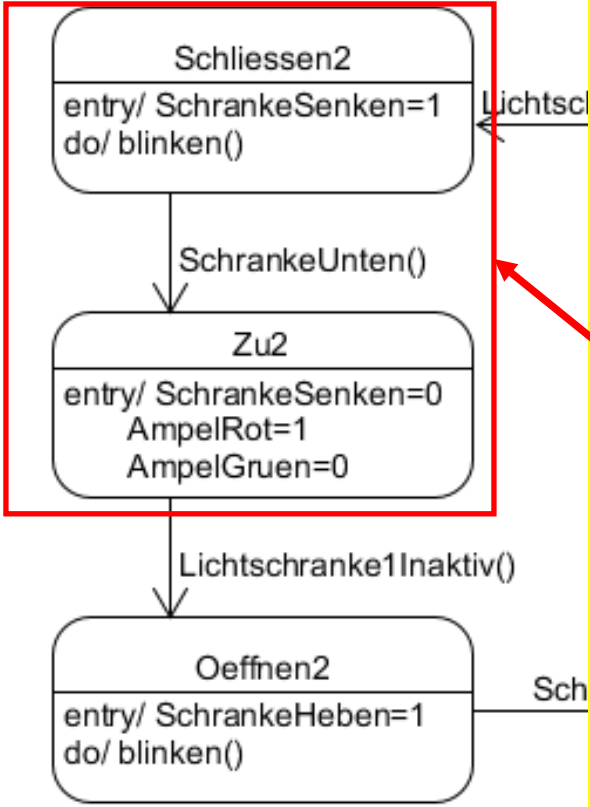
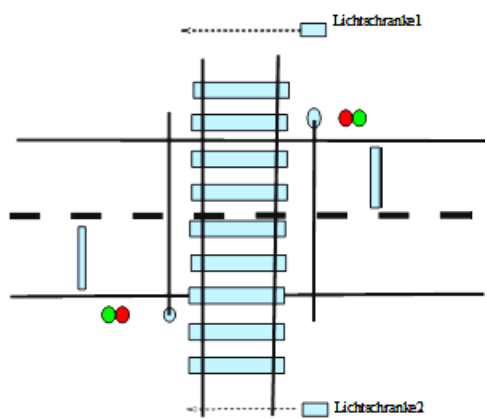
# Implementierung

# Zustandsdiagramm

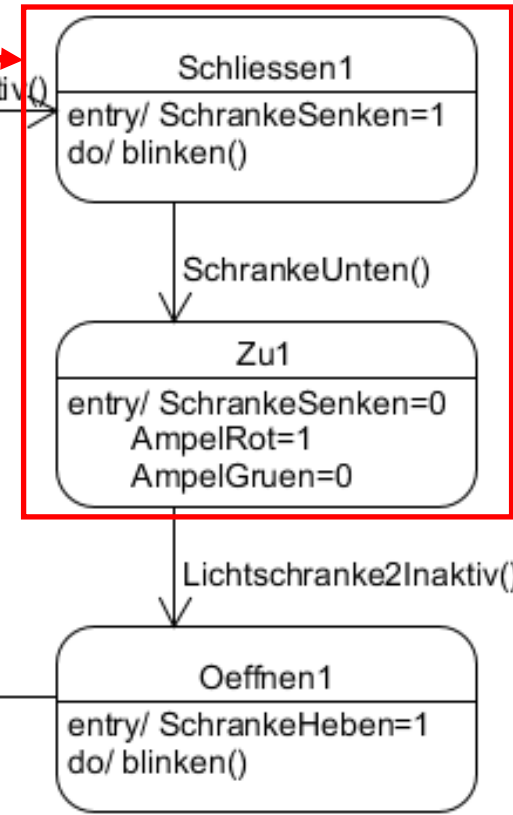


# Implementierung

# Zustandsdiagramm

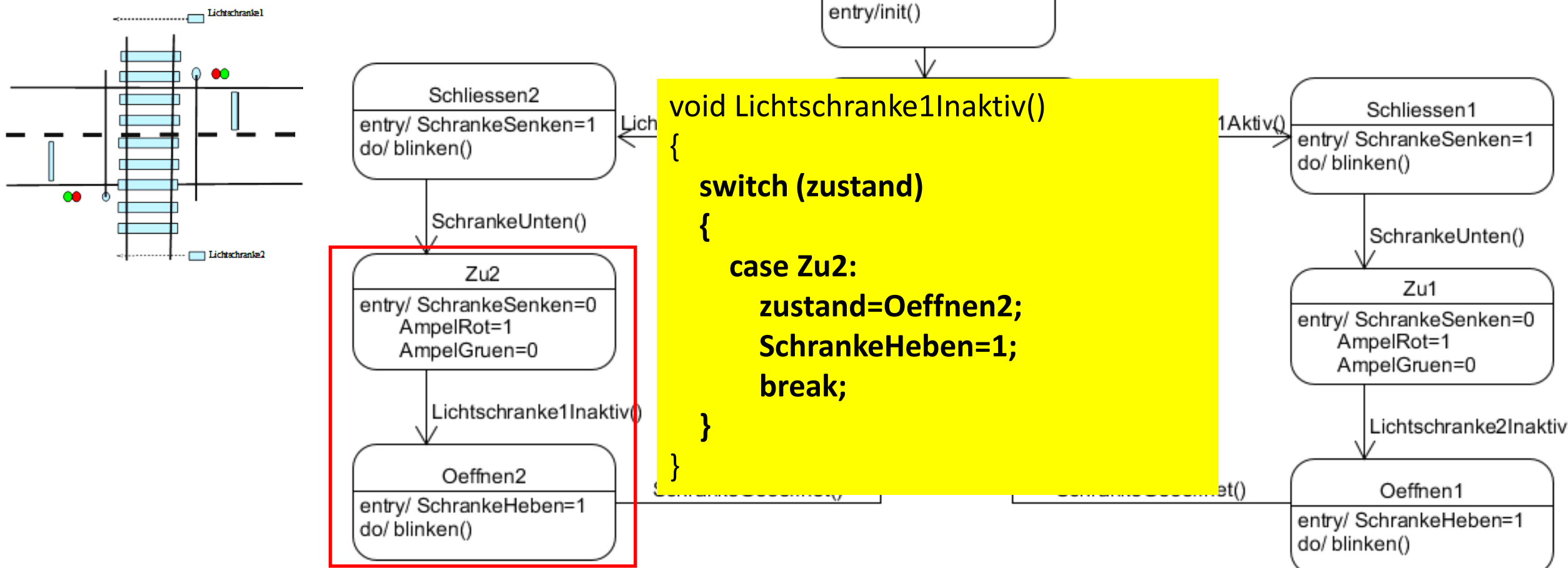


```
void SchrankeUnten()
{
    switch (zustand)
    {
        case Schliessen1:
            zustand=Zu1;
            SchrankeSenken=0;
            AmpelRot=1;
            AmpelGruen=0;
            break;
        case Schliessen2:
            zustand=Zu2;
            SchrankeSenken=0;
            AmpelRot=1;
            AmpelGruen=0;
            break;
    }
}
```



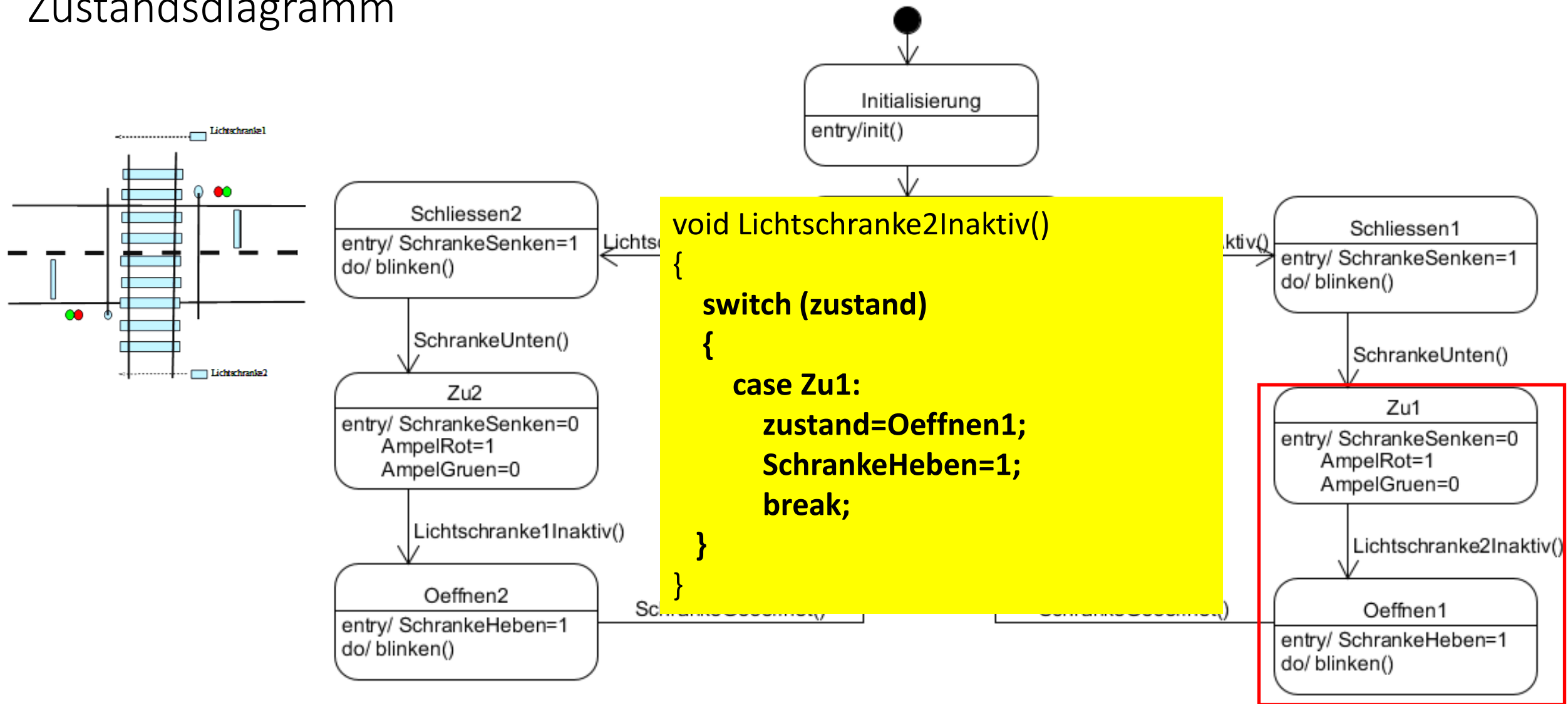
# Implementierung

# Zustandsdiagramm



# Implementierung

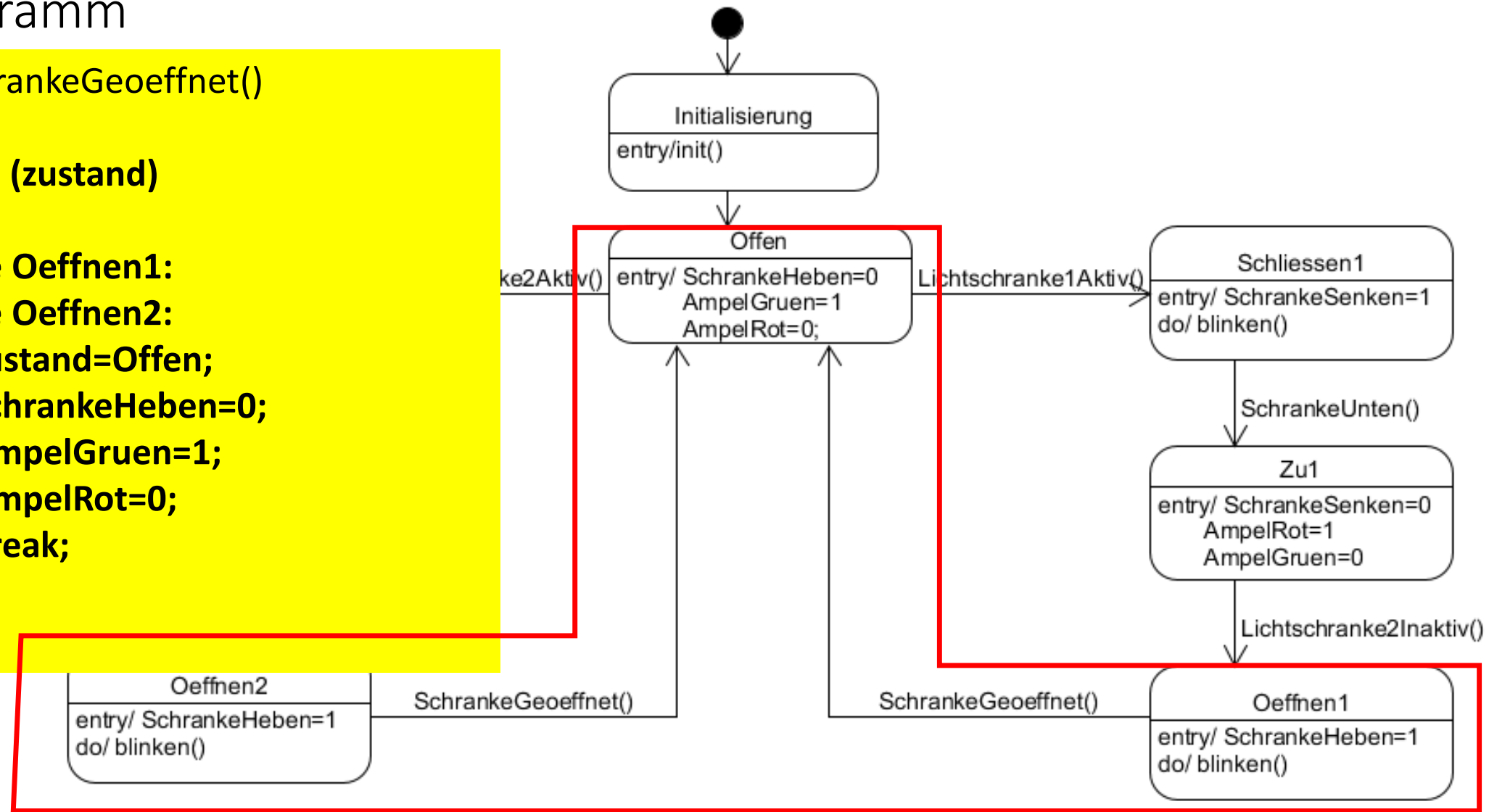
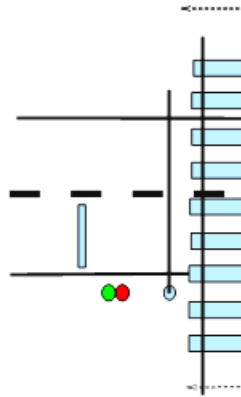
# Zustandsdiagramm



# Implementierung

# Zustandsdiagramm

```
void SchrankeGeoeffnet()
{
    switch (zustand)
    {
        case Oeffnen1:
        case Oeffnen2:
            zustand=Offen;
            SchrankeHeben=0;
            AmpelGruen=1;
            AmpelRot=0;
            break;
    }
}
```



# Implementierung