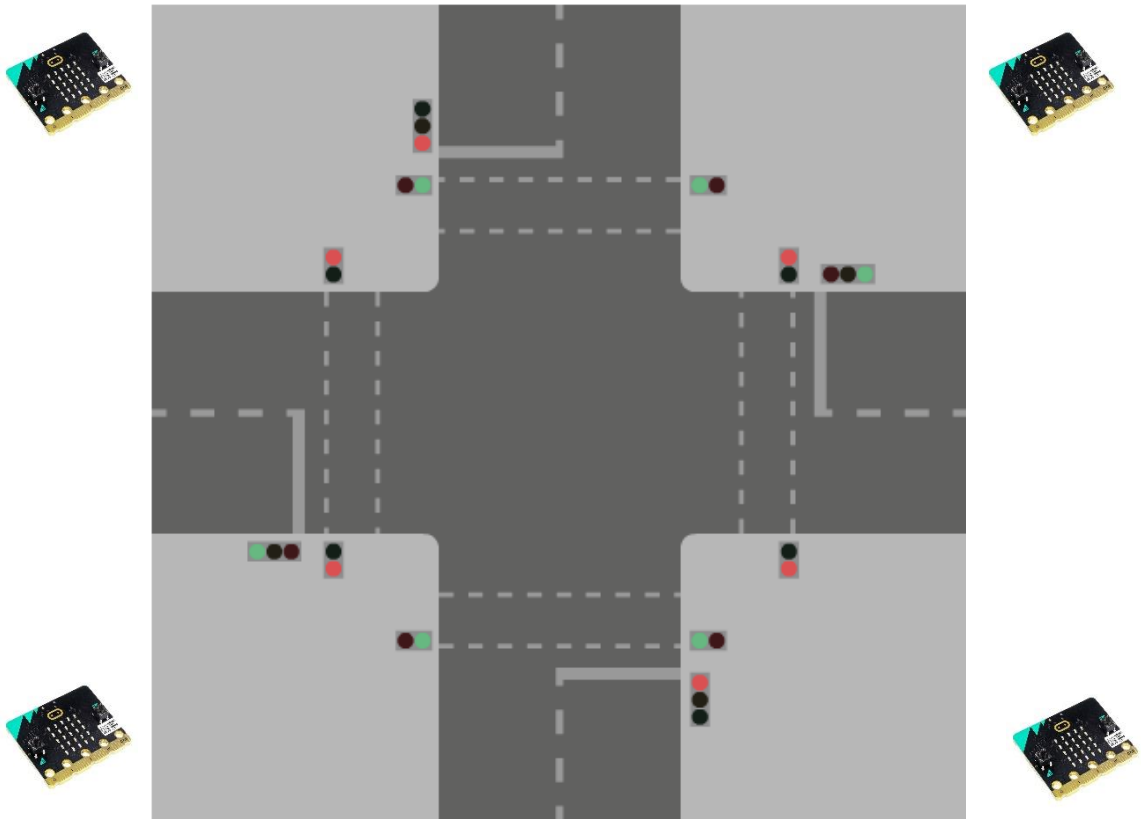
Handreichung PhC

1. Kurzvorstellung

Ampelschaltung für eine Straßenkreuzung. Mit einem Kontroller der Signale an beliebig viele Empfänger sendet uns so eine Ampelschaltung simuliert.

Kontroller

Empfänger



2. Einordnung in die Lehrpläne

2.1 Klassenstufe 8

Lernbereich 1: Algorithmen	**9 Ustd.**
Kennen des Algorithmusbegriffes	
Eigenschaften	→ Kl. 7, LB 3 → MA, Kl. 8, LB 3
Übertragen der Eigenschaften von Algorithmen auf Sachverhalte aus der Erfahrungswelt der Schüler	Soziale Medien, Smarthome, In-App-Käufe, Werbung ⇒ Medienbildung
Beherrschen der algorithmischen Lösung einfacher Problemstellungen in einer didaktisch reduzierten Programmierumgebung	Blocksprachen, visuelle Programmierumgebungen
- Darstellungsformen von Algorithmen	Blockdarstellung, verbale Beschreibung
- Sequenz, Verzweigung und Wiederholung	Zählschleife, kopfgesteuerte oder fußgesteuerte Schleife
Kennen der Bedeutung von Algorithmen im gesellschaftlichen Kontext	

2.2 Klassenstufe 10

Lernbereich 1: Algorithmen	**11 Ustd.**
Beherrschen der Implementierung der algorithmischen Grundstrukturen	grundlegende, einfache Algorithmen Syntax und Semantik → Kl. 8, LB 1 → LB 2
- Datentypen	
· Zahlen	
· Zeichenketten	
· Wahrheitswerte	
- Variablenzuweisungen	
- verknüpfte Bedingungen	Verkettung durch logische Operatoren
Kennen des Prinzips der Modularisierung	Nutzen von Unterprogrammen und Bibliotheken
Übertragen der Kenntnisse zu Algorithmen auf maschinelle Entscheidungsprozesse	autonomes Fahren, Gesichtserkennung, Wahlcomputer → KL. 9, LB 2 → ETH, Kl. 10, LB 1

3. Lernziele

3.1 Klassenstufe 8

Lernzielkategorien	Lernziele nach Anderson et al (2001)
Kognitive Lernziele	Stufe 1: Erinnern Die SchülerInnen kennen den Algorithmus Begriff und dessen Eigenschaften. Stufe 2: Verstehen Die SchülerInnen beherrschen algorithmische Lösungen einfacher Problemstellungen in einer didaktisch reduzierten Programmierumgebung. Stufe 4: Analysieren Die SchülerInnen übertragen die Eigenschaften von Algorithmen auf Sachverhalte aus ihrer Erfahrungswelt.
Psychomotorische Lernziele	Stufe 2: Verstehen Die SchülerInnen beherrschen das Konstruieren von Schaltnetzen anhand von Schaltplänen.
Affektive Lernziele	Stufe 1: Erinnern Die SchülerInnen kennen die Bedeutung von Algorithmen im gesellschaftlichen Leben Stufe 2: Verstehen Die SchülerInnen beherrschen das Anwenden von problemlösenden Verhalten zum Lösen von einfachen Problemstellungen

3.2 Klassenstufe 10

Lernzielkategorien	Lernziele nach Anderson et al (2001)
Kognitive Lernziele	Stufe 1: Erinnern Die SchülerInnen kennen den Algorithmus Begriff und dessen Eigenschaften. Stufe 2: Verstehen Die SchülerInnen beherrschen algorithmische Lösungen einfacher Problemstellungen mithilfe von Python. Stufe 4: Analysieren Die SchülerInnen übertragen die Eigenschaften von Algorithmen auf Sachverhalte aus ihrer Erfahrungswelt.
Psychomotorische Lernziele	Stufe 2: Verstehen Die SchülerInnen beherrschen das Konstruieren von Schaltnetzen anhand von Schaltplänen.
Affektive Lernziele	Stufe 2: Verstehen Die SchülerInnen beherrschen das Anwenden von problemlösenden Verhalten zum Lösen von einfachen Problemstellungen

4. Voraussetzungen

4.1 Materielle Voraussetzung

4.1.1 Kontroller

- 1x Micro:bit (inklusive Batterien)

4.1.2. Empfänger

- 1x Micro:bit (inklusive Batterien und Verbindungskabel)
- 1x Steckbrett + micro:bit- Halterung
    - 4x Ampeln (2 Autoampeln und 2 Fußgängerampeln [falls vorhanden])
- Mindestens 16x Kabel

Dieses Material reicht aus, um alle Programmteile auszuprobieren. Dies lässt sich aber für die aktive Nutzung noch erweitern. Jede Kreuzung kann auch mit 4 Autoampeln und 8 Fußgängerampeln versehen werden. Hierfür können entweder mehrere Projekte zusammengefügt werden, oder den Schülern mehr Material zur Verfügung gestellt werden.

4.1.3. Empfänger für oben genannte großes Projekt

- 4x Micro:bit (inklusive Batterien und 1x Verbindungskabel)
- 4x Steckbrett + micro:bit- Halterung
- 4x Steckbrett ohne micro:bit- Halterung
- 12 Ampeln (4 Autoampeln + 8 Fußgängerampeln falls vorhanden)
- Mindestens 40x Kabel

4.2 Technische Voraussetzungen

- Klassensatz micro:bit
- Zugang zur Programmierumgebung makecode für micro:bit: entweder per Internetzugang https://makecode.microbit.org/ oder als vorinstallierte MakeCode App

4.3 Fachliche Voraussetzung

- Die SchülerInnen besitzen grundlegendes Wissen über Programmierung, sowie einfacher Datentypen
- Die SchülerInnen besitzen grundlegendes Wissen über den Algorithmus-Begriff, sowie dessen Eigenschaften und der Darstellungsformen
- Die SchülerInnen verfügen über die Fähigkeit technische Schaltungen aufgrund von Schaltplänen aufzubauen
- Die SchülerInnen besitzen grundlegendes Wissen über die MakeCode Programmierumgebung bzw. Python

5. Kurzdarstellung

5.1 Allgemeine Informationen

Ampelphasen einer Kreuzung

Phase	Autoampel Straße 1	Autoampel Straße 2	Fußgängerampel 1	Fußgängerampel 2
1	grün	rot	grün	rot
2	gelb	rot	rot	rot
3	rot	rot	rot	rot
4	rot	rot, gelb	rot	rot
5	rot	grün	rot	grün
6	rot	gelb	rot	rot
7	rot	rot	rot	rot
8	rot, gelb	rot	rot	rot

Die per Radiofrequenz gesendeten Zahlen von 1 bis 8 stimmen mit den Ampelphasen in der obigen Tabelle überein.

Anzusteuernde Ampel	**Lampe**	**Pin**
Autoampel 1	grün	0
Autoampel 1	gelb	1
Autoampel 1	rot	2
Fußgängerampel 1	grün	0
Fußgängerampel 1	rot	8
Autoampel 2	grün	9
Autoampel 2	gelb	12
Autoampel 2	rot	13
Fußgängerampel 2	grün	9
Fußgängerampel 2	rot	14

Diese Pinbelegung sollte eingehalten werden. Denn bei anderen kann es zu Problemen führen.

5.2 Kontroller

Der Kontroller steuert mit den ausgesendeten Zahlen per Funkt Frequenz alle micro:bits die in selben Radiogruppe sind. Beim Kontroller sind keine Kabelverbindungen Notwendig, daher ist dieser sehr mobil und kann nicht nur die Ampelschaltung an einer Kreuzung steuern, sondern es wäre möglich mit einem Kontroller verschiedene Ampelschaltungen gleichzeitig zu beeinflussen

5.2.1 Python Code

```
radio.set_group(5)
Modus = 0
radio.send_number(17)

def on_forever():
    if Modus == 1:
        radio.send_number(1)
        basic.pause(5000)
        if Modus == 1:
            radio.send_number(2)
            basic.pause(1500)
            if Modus == 1:
                radio.send_number(3)
                basic.pause(1500)
                if Modus == 1:
                    radio.send_number(4)
                    basic.pause(1500)
                    if Modus == 1:
                        radio.send_number(5)
                        basic.pause(5000)
                        if Modus == 1:
                            radio.send_number(6)
                            basic.pause(1500)
                            if Modus == 1:
                                radio.send_number(7)
```

```
                basic.pause(1500)
                if Modus == 1:
                    radio.send_number(8)
                    basic.pause(1500)
basic.forever(on_forever)

def on_forever2():
    basic.show_number(Modus)
basic.forever(on_forever2)

def on_forever3():
    global Modus
    if Modus <= 1:
        if input.button_is_pressed(Button.A):
            Modus = 2
    elif Modus == 2:
        if input.button_is_pressed(Button.B):
            Modus = 1
basic.forever(on_forever3)

def on_forever4():
    if Modus == 2:
        radio.send_number(10)
        basic.pause(4000)
basic.forever(on_forever4)
```

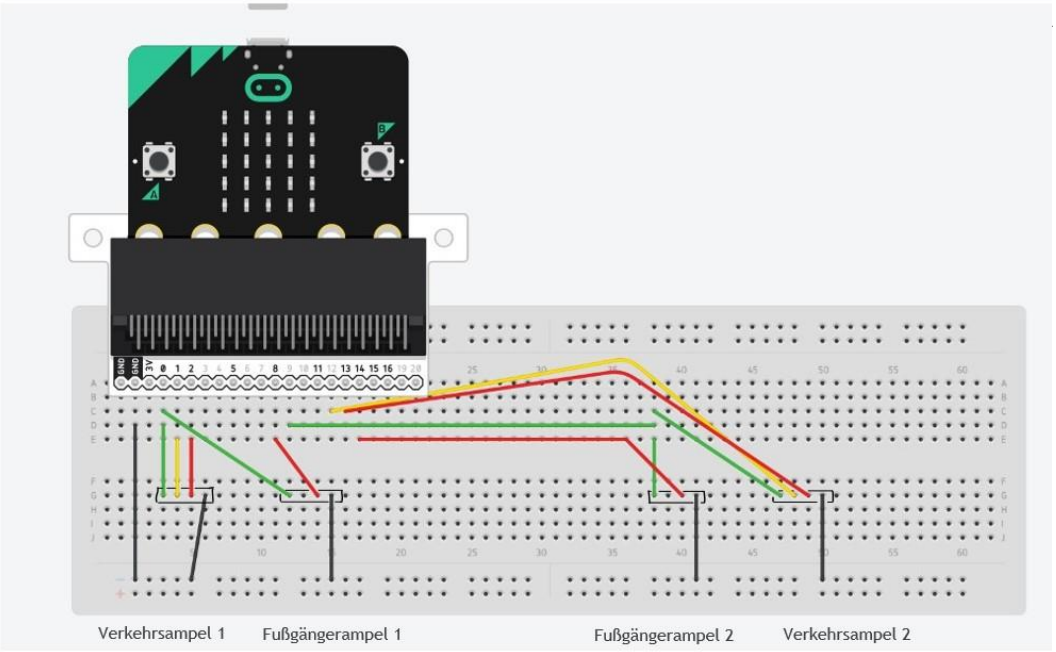
5.2.2 Python Code mit Kommentaren:

```
radio.set_group(5)
# Radiogruppe wird auf 5 gesetzt. (Muss beim Kontroller und allen dazugehörigen Empfängern identisch sein)
Modus = 0
radio.send_number(17)
# sendet die Nummer 17 per Radiofrequenz (Sorgt für das anschalten aller verbundenen Lampen zur Kontrolle)

def on_forever():
# eine unendliche Schleife
    if Modus == 1:
        radio.send_number(1)
        basic.pause(5000)
# wenn der Modus 1 ist, wird die Nummer 1 per Radiofrequenz gesendet und dann 5000ms gewartet
        if Modus == 1:
            radio.send_number(2)
            basic.pause(1500)
# wenn der Modus immer noch 1 ist, wird die Nummer 2 per Radiofrequenz gesendet und dann 1500ms gewartet
            if Modus == 1:
                radio.send_number(3)
                basic.pause(1500)
# wenn der Modus immer noch 1 ist, wird die Nummer 3 per Radiofrequenz gesendet und dann 1500ms gewartet
                if Modus == 1:
                    radio.send_number(4)
                    basic.pause(1500)
# wenn der Modus immer noch 1 ist, wird die Nummer 4 per Radiofrequenz gesendet und dann 1500ms gewartet
                    if Modus == 1:
                        radio.send_number(5)
                        basic.pause(5000)
# wenn der Modus immer noch 1 ist, wird die Nummer 5 per Radiofrequenz gesendet und dann 5000ms gewartet
                        if Modus == 1:
                            radio.send_number(6)
                            basic.pause(1500)
# wenn der Modus immer noch 1 ist, wird die Nummer 6 per Radiofrequenz gesendet und dann 1500ms gewartet
                            if Modus == 1:
                                radio.send_number(7)
                                basic.pause(1500)
```

```
# wenn der Modus immer noch 1 ist, wird die Nummer 7 per Radiofrequenz gesendet und dann 1500ms gewartet
                    if Modus == 1:
                        radio.send_number(8)
                        basic.pause(1500)
# wenn der Modus immer noch 1 ist, wird die Nummer 8 per Radiofrequenz gesendet und dann 1500ms gewartet
basic.forever(on_forever)
# Dies wird so oft wiederholt, wie der Modus 1 ist. Sollte der Modus zwischendurch auf 2 wechseln bricht diese Schleife ab

def on_forever2():
    basic.show_number(Modus)
# Der momentane Modus wird auf dem Kontroller micro:bit angezeigt
basic.forever(on_forever2)

def on_forever3():
    global Modus
    if Modus <= 1:
        if input.button_is_pressed(Button.A):
            Modus = 2
# wenn der Modus <= 1 ist, kann der Knopf A gedrückt werden um den Modus auf 2 zu ändern
    elif Modus == 2:
        if input.button_is_pressed(Button.B):
            Modus = 1
# wenn der Modus = 2 ist, kann der Knopf B gedrückt werden um den Modus auf 1 zu ändern
basic.forever(on_forever3)

def on_forever4():
    if Modus == 2:
        radio.send_number(10)
        basic.pause(4000)
# wenn der Modus = 2 ist, wird die Nummer 10 per Radiofrequenz gesendet und 4000ms gewartet bevor getestet wird, ob sich der Wert für Modus geändert hat, falls nicht, wird der Vorgang wiederholt
basic.forever(on_forever4)
```

5.3 Empfänger

5.3.1 Für die einfache Verkabelung:



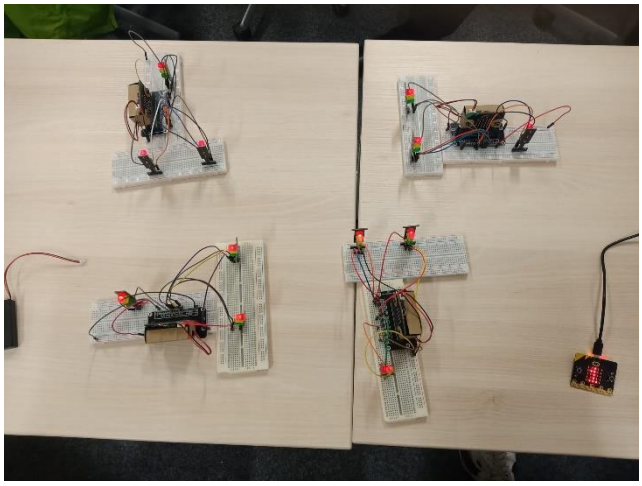
Hierbei werden alle vier verschiedenen Ampeln an einen micro:bit angeschlossen. Dies gibt den SchülerInnen die Möglichkeit ihre Programme zu testen und Fehler frühzeitig zu erkennen.

Obwohl alle vier Ampeln über nur einen micro:bit angesteuert werden können ist dieses nicht notwendig. Es besteht die Möglichkeit weniger Ampeln anzuschließen. Dies liegt dem Nutzer offen und sorgt für eine höhere Flexibilität der Ampelschaltungen.

5.3.2 Für die Nutzung von 4 Empfänger micro:bit´s

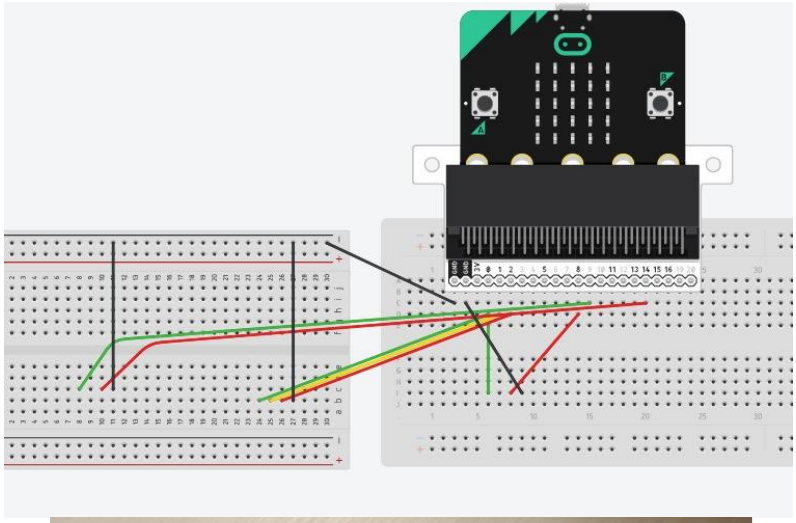
Hier gibt es zwei verschiedene Schaltpläne die jeweils zwei Mal gebaut werden müssen. Die beiden Schaltpläne sind jeweils gespiegelt. Beim späteren Aufbauen werden die identischen Schaltungen in die jeweils entgegengesetzten Ecken der Kreuzung gestellt und die Ampeln so ausgerichtet, das diese passen.

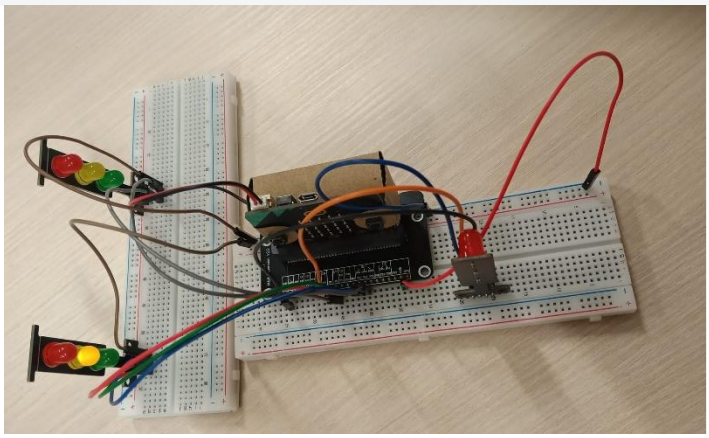
Auf dem folgenden Bild sieht man eine komplett aufgebaute Kreuzung mit dem Kontoller in der rechten unteren Ecke und vier Empfängern mit jeweils 3 angeschlossenen Ampeln.



5.3.2.1 Schaltplan 1

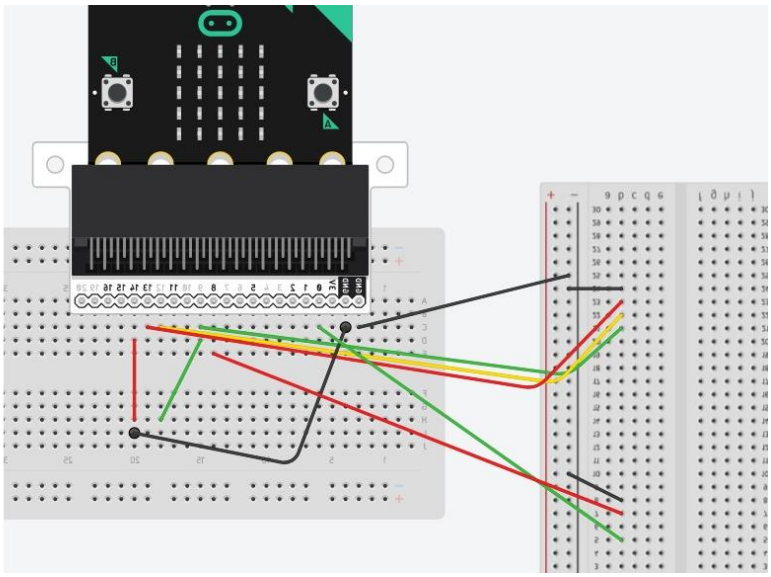
Bei diesem Schaltplan werden die Fußgängerampel 1, Autoampel 2 und die Fußgängerampel 2 angesteuert.





5.3.2.2 Schaltplan 2

Bei diesem Schaltplan werden die Autoampel 1, Fußgängerampel 1 und die Fußgängerampel 2 angesteuert.



5.3.3 Python code

```
radio.set_group(5)
def on_received_number(receivedNumber):
    if receivedNumber == 1:
        Str1Grün()
        Str2Rot()
    elif receivedNumber == 2:
        Str1Gelb()
        Str2Rot()
    elif receivedNumber == 3:
        Str1Rot()
        Str2Rot()
    elif receivedNumber == 4:
        Str1Rot()
        Str2RotGelb()
    elif receivedNumber == 5:
        Str1Rot()
        Str2Grün()
    elif receivedNumber == 6:
        Str1Rot()
        Str2Gelb()
    elif receivedNumber == 7:
```

```
        Str1Rot()
        Str2Rot()
    elif receivedNumber == 8:
        Str1RotGelb()
        Str2Rot()
    elif receivedNumber == 10:
        BN()
    elif receivedNumber == 17:
        alleAn()
        basic.pause(2000)
        alleAus()
        basic.pause(2000)
radio.on_received_number(on_received_number)

def alleAn():
    pins.digital_write_pin(DigitalPin.P0, 1)
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P2, 1)
    pins.digital_write_pin(DigitalPin.P8, 1)
    pins.digital_write_pin(DigitalPin.P9, 1)
    pins.digital_write_pin(DigitalPin.P12, 1)
    pins.digital_write_pin(DigitalPin.P13, 1)
    pins.digital_write_pin(DigitalPin.P14, 1)
def Str1RotGelb():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P2, 1)
    pins.digital_write_pin(DigitalPin.P8, 1)
def Str1Rot():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P2, 1)
    pins.digital_write_pin(DigitalPin.P8, 1)
def Str2Rot():
    pins.digital_write_pin(DigitalPin.P9, 0)
    pins.digital_write_pin(DigitalPin.P12, 0)
    pins.digital_write_pin(DigitalPin.P13, 1)
    pins.digital_write_pin(DigitalPin.P14, 1)
```

```
def Str1Grün():
    pins.digital_write_pin(DigitalPin.P0, 1)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P2, 0)
    pins.digital_write_pin(DigitalPin.P8, 0)
def Str2Grün():
    pins.digital_write_pin(DigitalPin.P9, 1)
    pins.digital_write_pin(DigitalPin.P12, 0)
    pins.digital_write_pin(DigitalPin.P13, 0)
    pins.digital_write_pin(DigitalPin.P14, 0)
def BN():
    alleAus()
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P12, 1)
    basic.pause(2000)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P12, 0)
def Str1Gelb():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P2, 0)
    pins.digital_write_pin(DigitalPin.P8, 1)
def alleAus():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P2, 0)
    pins.digital_write_pin(DigitalPin.P8, 0)
    pins.digital_write_pin(DigitalPin.P9, 0)
    pins.digital_write_pin(DigitalPin.P12, 0)
    pins.digital_write_pin(DigitalPin.P13, 0)
    pins.digital_write_pin(DigitalPin.P14, 0)
def Str2Gelb():
    pins.digital_write_pin(DigitalPin.P9, 0)
    pins.digital_write_pin(DigitalPin.P12, 1)
    pins.digital_write_pin(DigitalPin.P13, 0)
    pins.digital_write_pin(DigitalPin.P14, 1)
def Str2RotGelb():
    pins.digital_write_pin(DigitalPin.P9, 0)
```

```
    pins.digital_write_pin(DigitalPin.P12, 1)
    pins.digital_write_pin(DigitalPin.P13, 1)
    pins.digital_write_pin(DigitalPin.P14, 1)
```

5.3.4 Python code mit Kommentaren

```
radio.set_group(5)
# die Radiogruppe wird auf 5 gesetzt. (Muss beim Kontroller und allen dazugehörigen Empfängern identisch sein)
def on_received_number(receivedNumber):
    if receivedNumber == 1:
        Str1Grün()
        Str2Rot()
# Wird die Nummer 1 empfangen, werden die beiden Funktionen Str1Grün() und Str2Rot() ausgeführt
    elif receivedNumber == 2:
        Str1Gelb()
        Str2Rot()
# Wenn die Nummer 2 empfangen wird, werden die beiden Funktionen Str1Gelb() und Str2Rot() ausgeführt
    elif receivedNumber == 3:
        Str1Rot()
        Str2Rot()
# Wenn die Nummer 3 empfangen wird, werden die beiden Funktionen Str1Rot() und Str2Rot() ausgeführt
    elif receivedNumber == 4:
        Str1Rot()
        Str2RotGelb()
# Wenn die Nummer 4 empfangen wird, werden die beiden Funktionen Str1Rot() und Str2RotGelb() ausgeführt
    elif receivedNumber == 5:
        Str1Rot()
        Str2Grün()
# Wenn die Nummer 5 empfangen wird, werden die beiden Funktionen Str1Rot() und Str2Grün() ausgeführt
    elif receivedNumber == 6:
        Str1Rot()
        Str2Gelb()
# Wenn die Nummer 6 empfangen wird, werden die beiden Funktionen Str1Rot() und Str2Gelb() ausgeführt
    elif receivedNumber == 7:
        Str1Rot()
        Str2Rot()
# Wenn die Nummer 7 empfangen wird, werden die beiden Funktionen Str1Rot() und Str2Rot() ausgeführt
    elif receivedNumber == 8:
```

```
        Str1RotGelb()
        Str2Rot()
# Wenn die Nummer 8 empfangen wird, werden die beiden Funktionen Str1RotGelb() und Str2Rot() ausgeführt
    elif receivedNumber == 10:
        BN()
#Wenn die Nummer 10 empfangen wird, wird die Funktion BN() ausgeführt
    elif receivedNumber == 17:
        alleAn()
        basic.pause(2000)
        alleAus()
        basic.pause(2000)
#Wird die Nummer 17 empfangen, werden die Funktion alleAn(), alleAus() mit einer Pause von 2000ms ausgeführt
radio.on_received_number(on_received_number)

def alleAn():
    pins.digital_write_pin(DigitalPin.P0, 1)
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P2, 1)
    pins.digital_write_pin(DigitalPin.P8, 1)
    pins.digital_write_pin(DigitalPin.P9, 1)
    pins.digital_write_pin(DigitalPin.P12, 1)
    pins.digital_write_pin(DigitalPin.P13, 1)
    pins.digital_write_pin(DigitalPin.P14, 1)
def Str1RotGelb():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P2, 1)
    pins.digital_write_pin(DigitalPin.P8, 1)
def Str1Rot():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P2, 1)
    pins.digital_write_pin(DigitalPin.P8, 1)
def Str2Rot():
    pins.digital_write_pin(DigitalPin.P9, 0)
    pins.digital_write_pin(DigitalPin.P12, 0)
    pins.digital_write_pin(DigitalPin.P13, 1)
    pins.digital_write_pin(DigitalPin.P14, 1)
```

```
def Str1Grün():
    pins.digital_write_pin(DigitalPin.P0, 1)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P2, 0)
    pins.digital_write_pin(DigitalPin.P8, 0)
def Str2Grün():
    pins.digital_write_pin(DigitalPin.P9, 1)
    pins.digital_write_pin(DigitalPin.P12, 0)
    pins.digital_write_pin(DigitalPin.P13, 0)
    pins.digital_write_pin(DigitalPin.P14, 0)
def BN():
    alleAus()
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P12, 1)
    basic.pause(2000)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P12, 0)
def Str1Gelb():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 1)
    pins.digital_write_pin(DigitalPin.P2, 0)
    pins.digital_write_pin(DigitalPin.P8, 1)
def alleAus():
    pins.digital_write_pin(DigitalPin.P0, 0)
    pins.digital_write_pin(DigitalPin.P1, 0)
    pins.digital_write_pin(DigitalPin.P2, 0)
    pins.digital_write_pin(DigitalPin.P8, 0)
    pins.digital_write_pin(DigitalPin.P9, 0)
    pins.digital_write_pin(DigitalPin.P12, 0)
    pins.digital_write_pin(DigitalPin.P13, 0)
    pins.digital_write_pin(DigitalPin.P14, 0)
def Str2Gelb():
    pins.digital_write_pin(DigitalPin.P9, 0)
    pins.digital_write_pin(DigitalPin.P12, 1)
    pins.digital_write_pin(DigitalPin.P13, 0)
    pins.digital_write_pin(DigitalPin.P14, 1)
def Str2RotGelb():
    pins.digital_write_pin(DigitalPin.P9, 0)
```

```
    pins.digital_write_pin(DigitalPin.P12, 1)
    pins.digital_write_pin(DigitalPin.P13, 1)
    pins.digital_write_pin(DigitalPin.P14, 1)
#Hier werden die Funktionen definiert, indem die jeweiligen Pins auf 1 (Pin ist aktiv) oder 0 (Pin ist nicht aktiv) gesetzt werden
```