Was hat und was kann ein {CODE}RACER ?

Aktoren & Sensoren

Antriebsmotoren

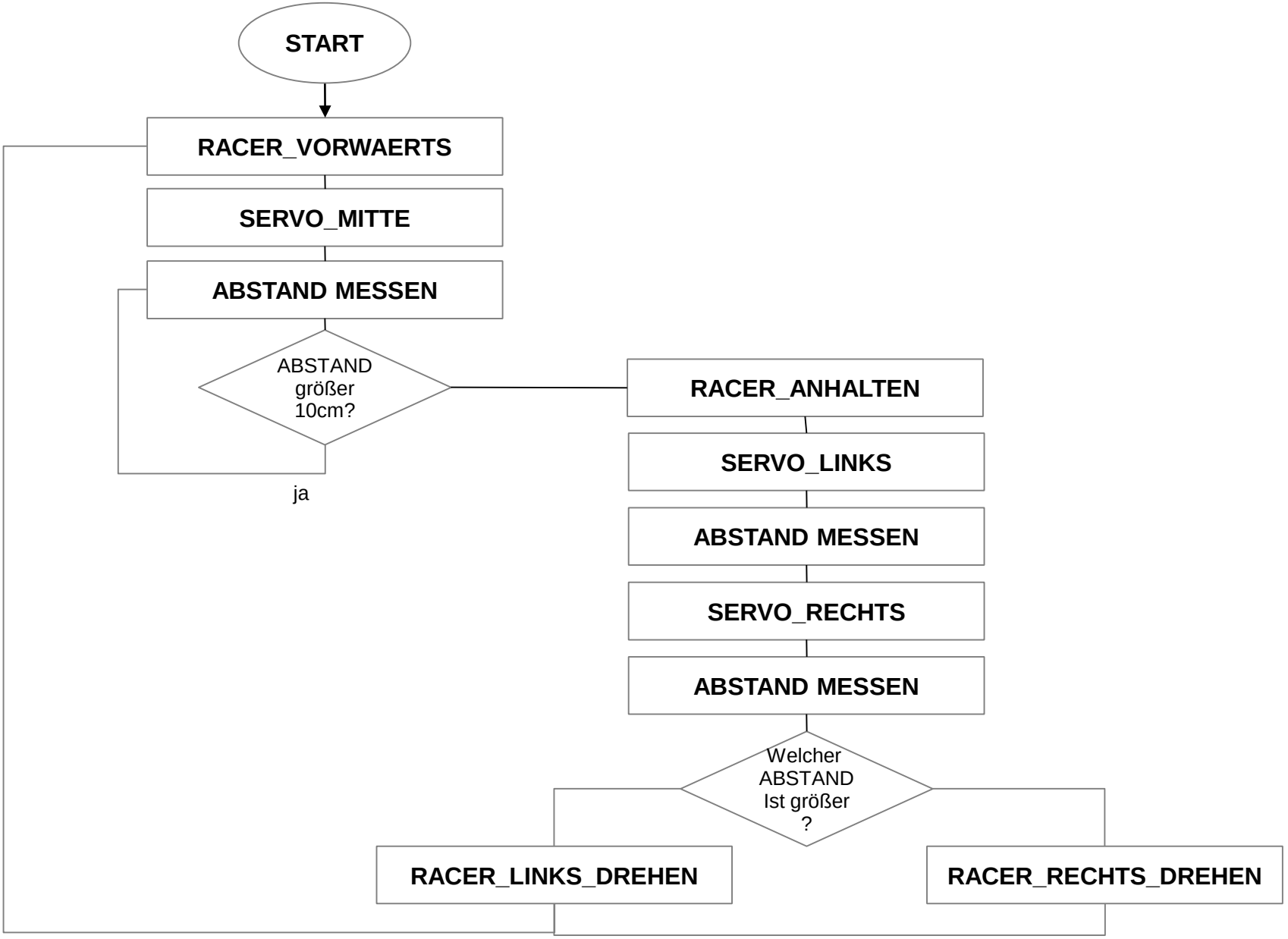
Servo

Ultraschallsensor

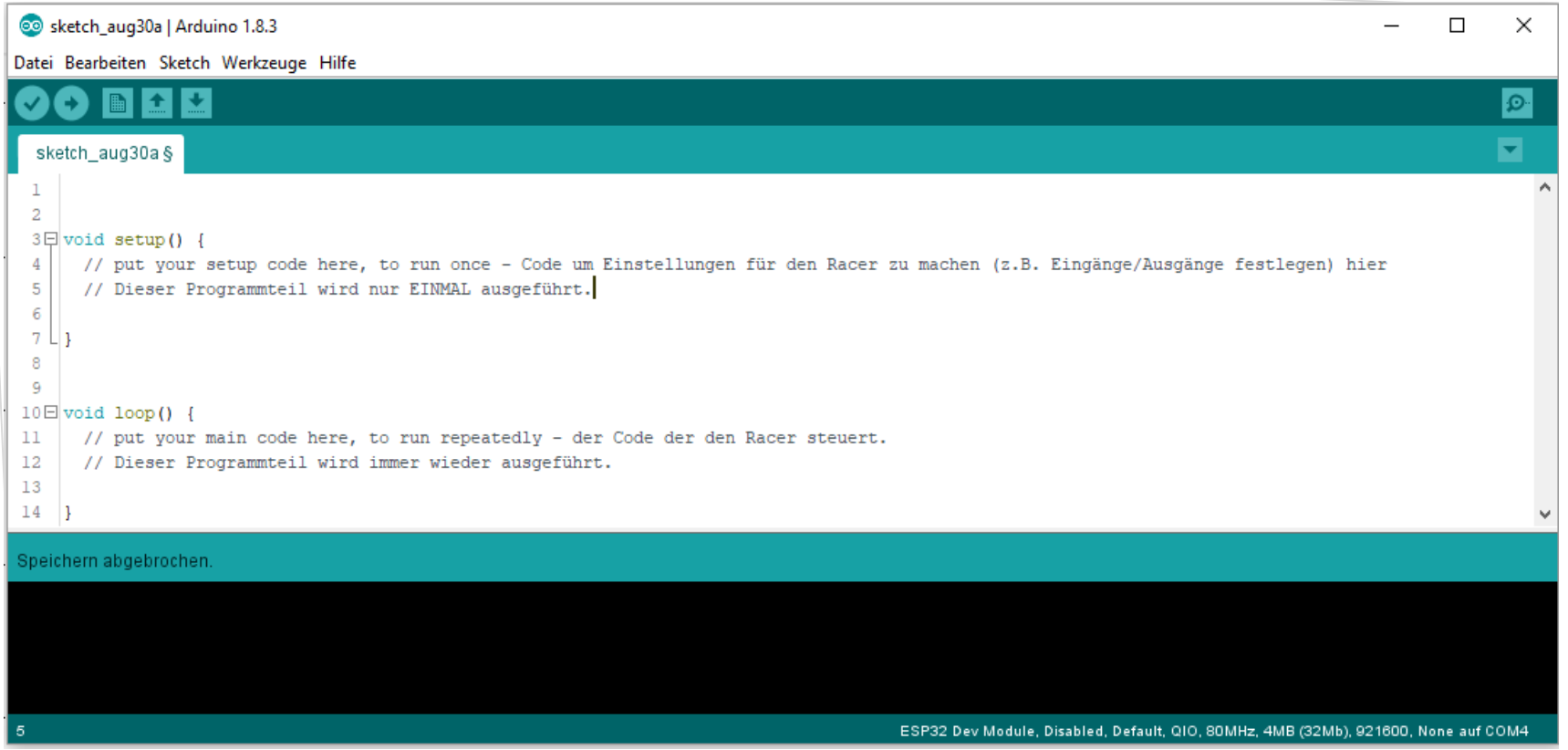
Funktionen

Antriebsmotoren	Servo	Ultraschallsensor
RACER_ANHALTEN - Motor links und rechts aus	**SERVO_MITTE** - Ultraschallsensor nach vorn	**ABSTAND MESSEN** - Abstand vorn (Servo in der Mitte)
RACER_VORWAERTS - Motor links und rechts vorwärts	**SERVO_LINKS** - Ultraschallsensor nach links drehen	
RACER_RUECKWAERTS - Motor links und rechts rückwärts	**SERVO_RECHTS** - Ultraschallsensor nach rechts drehen	
RACER_RECHTS_DREHEN - beide Motoren gleichzeitig: . links kurz vorwärts . rechts kurz rückwärts	**SERVO_SCHWENK** - Ultraschallsensor nach rechts und nach links hin und her drehen	
RACER_LINKS_DREHEN - beide Motoren gleichzeitig: . links kurz rückwärts . rechts kurz vorwärts		

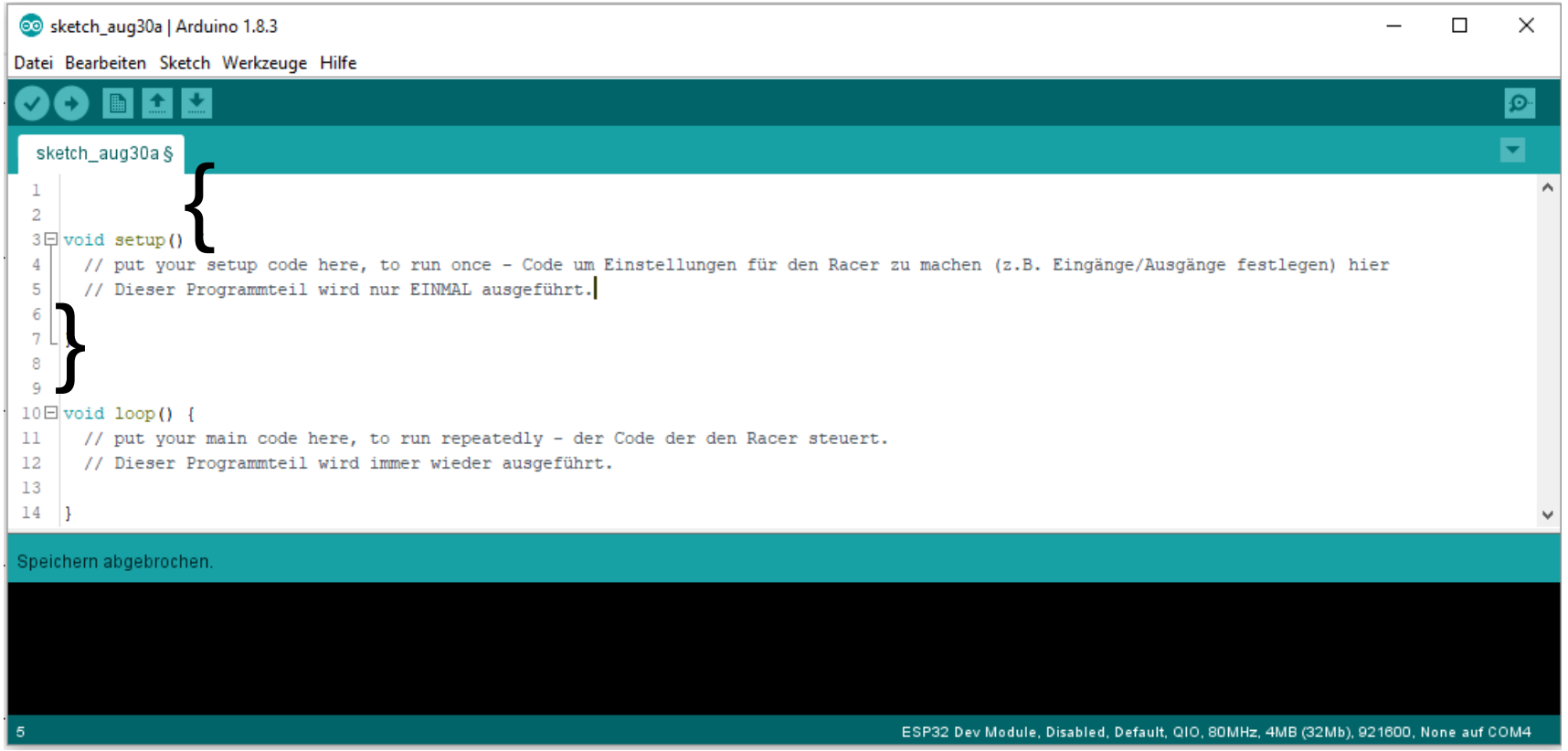
Wie fährt ein {CODE}RACER ohne anzuecken? Das geht auch besser ☺ …



Grundsätzlicher Aufbau von Code in Arduino



Grundsätzlicher Aufbau von Code in Arduino – Setup()

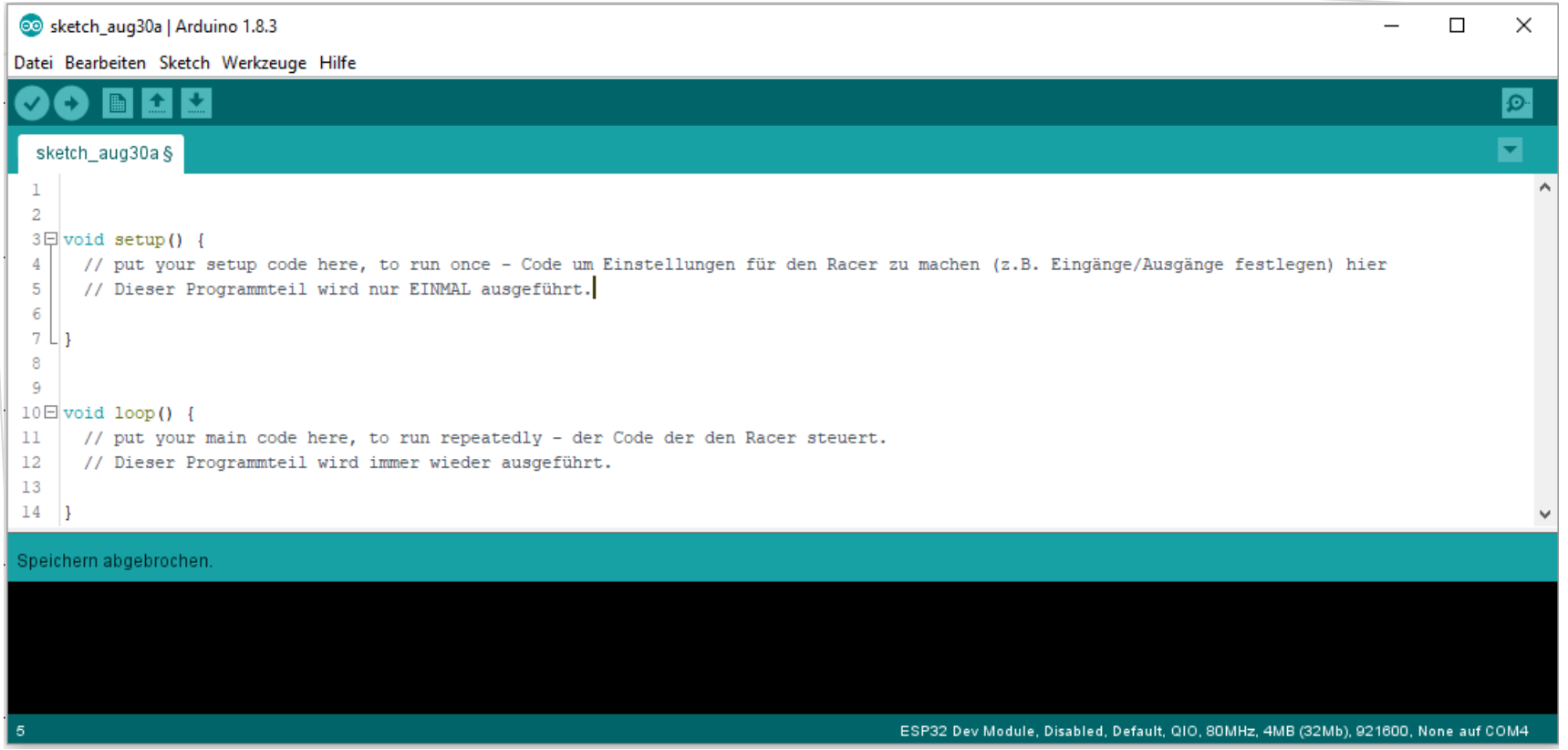


Grundsätzlicher Aufbau von Code in Arduino – Setup()

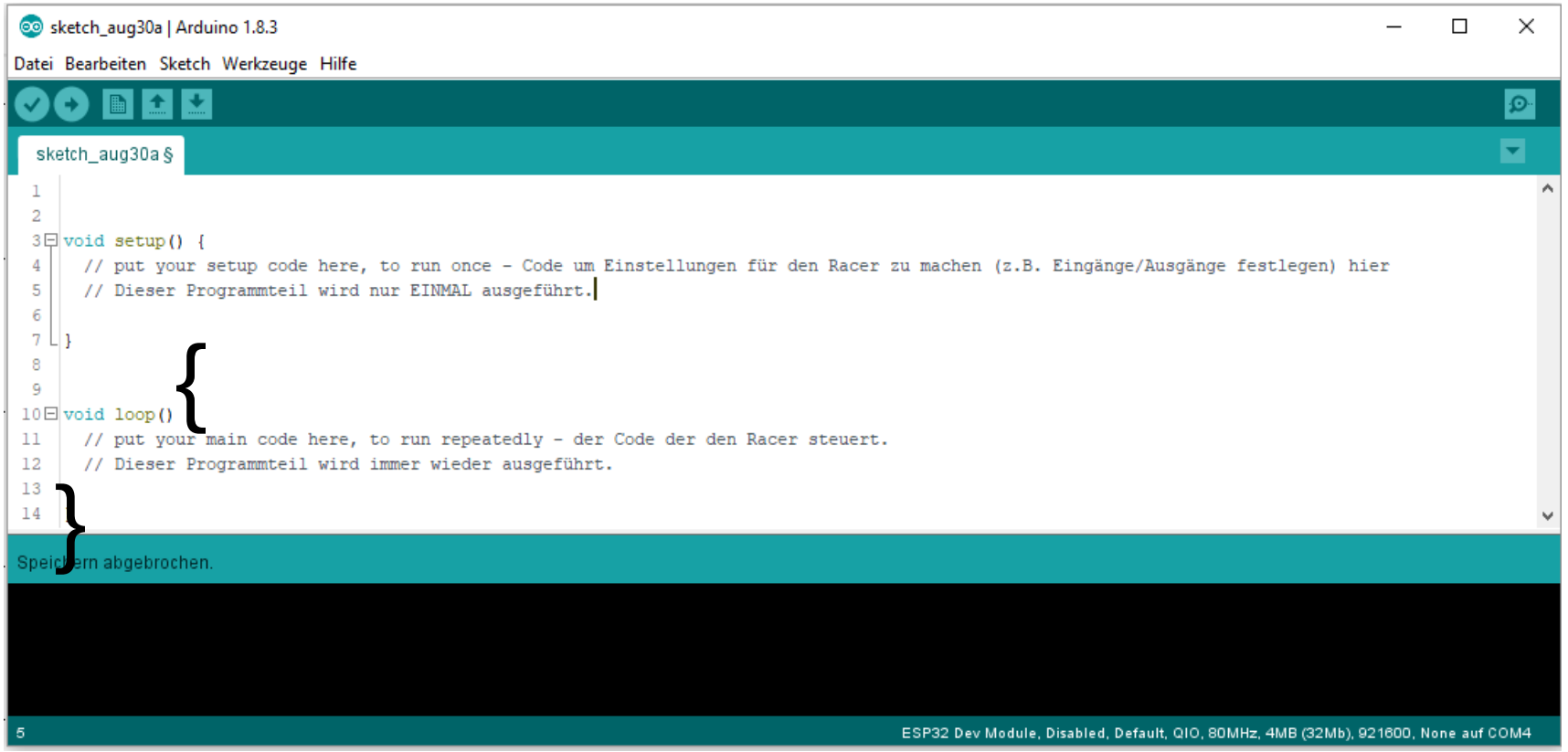
setup() { …. } – Programmteil:

- wird ***einmal*** nach dem Neustart des Microcontrollers ausgeführt !

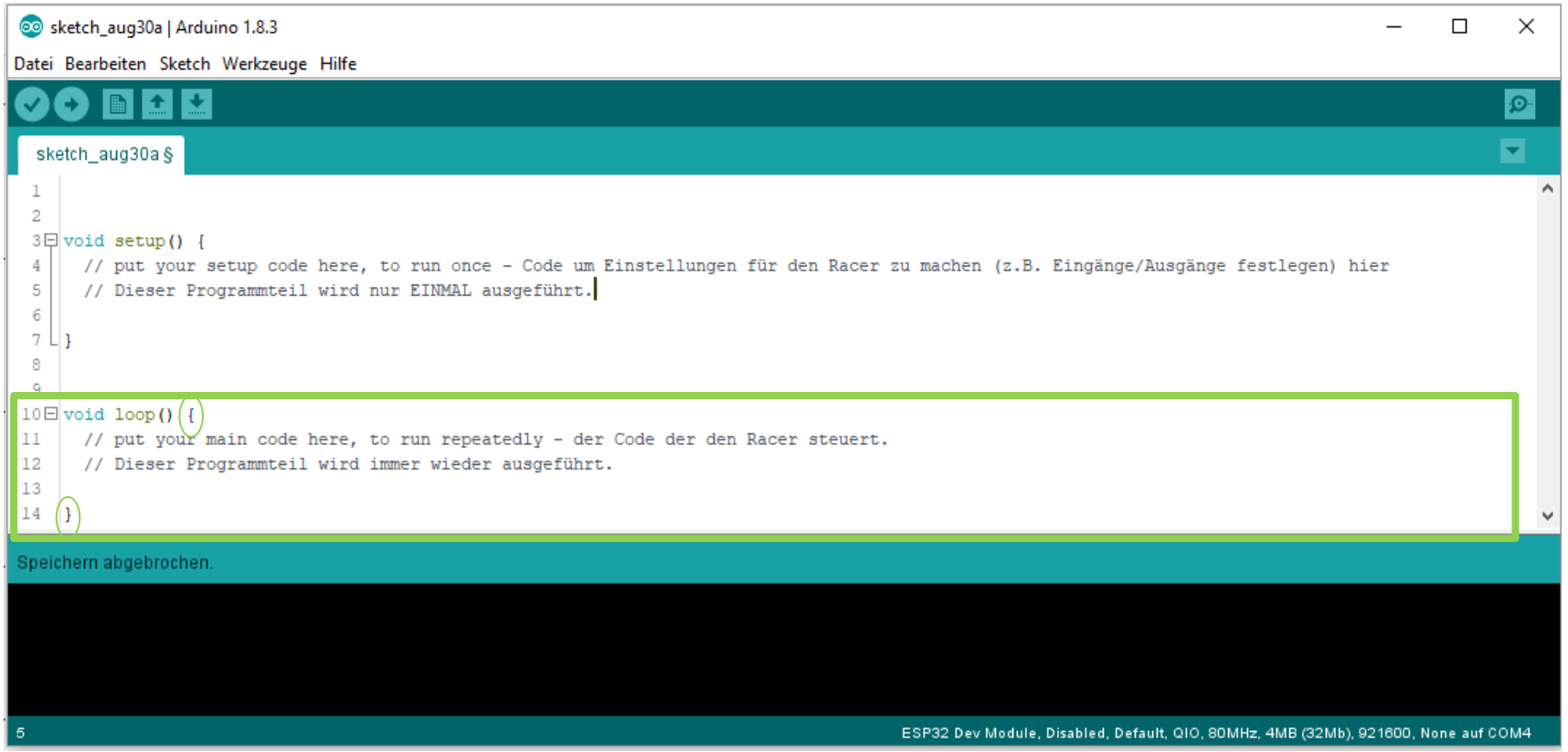
Grundsätzlicher Aufbau von Code in Arduino



Grundsätzlicher Aufbau von Code in Arduino – loop()



Grundsätzlicher Aufbau von Code in Arduino – loop()



loop() { …. } – Programmteil:

- wird ***immer wieder*** ausgeführt !

Grundsätzlicher Aufbau von Code in Arduino – setup() & loop()

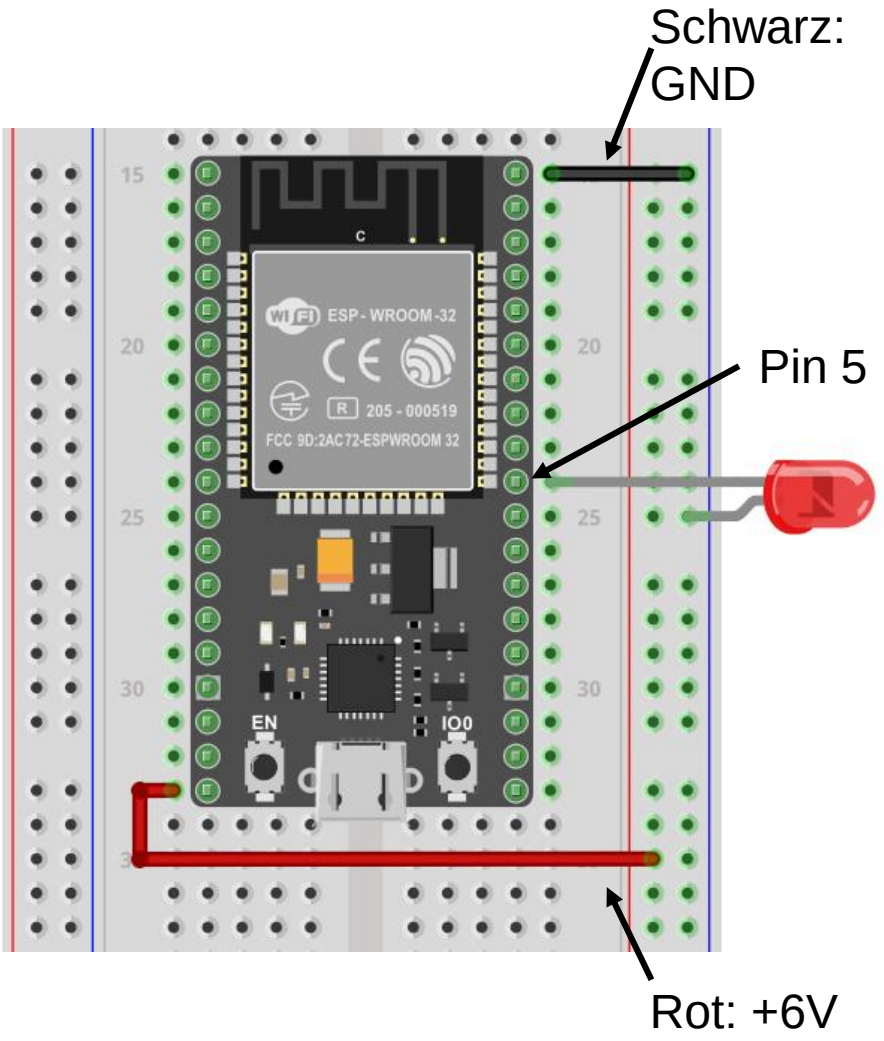
setup() { …. } – Programmteil:

- wird ***einmal*** nach dem Neustart des Microcontrollers ausgeführt !

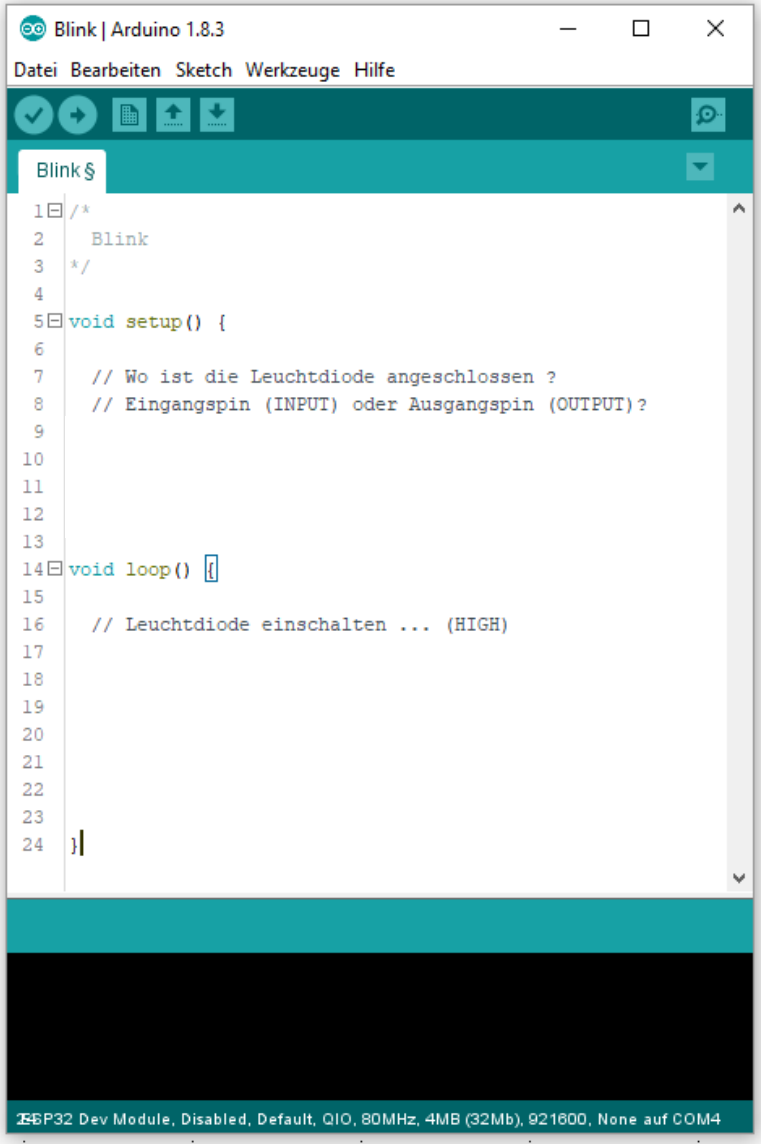
loop() { …. } – Programmteil:

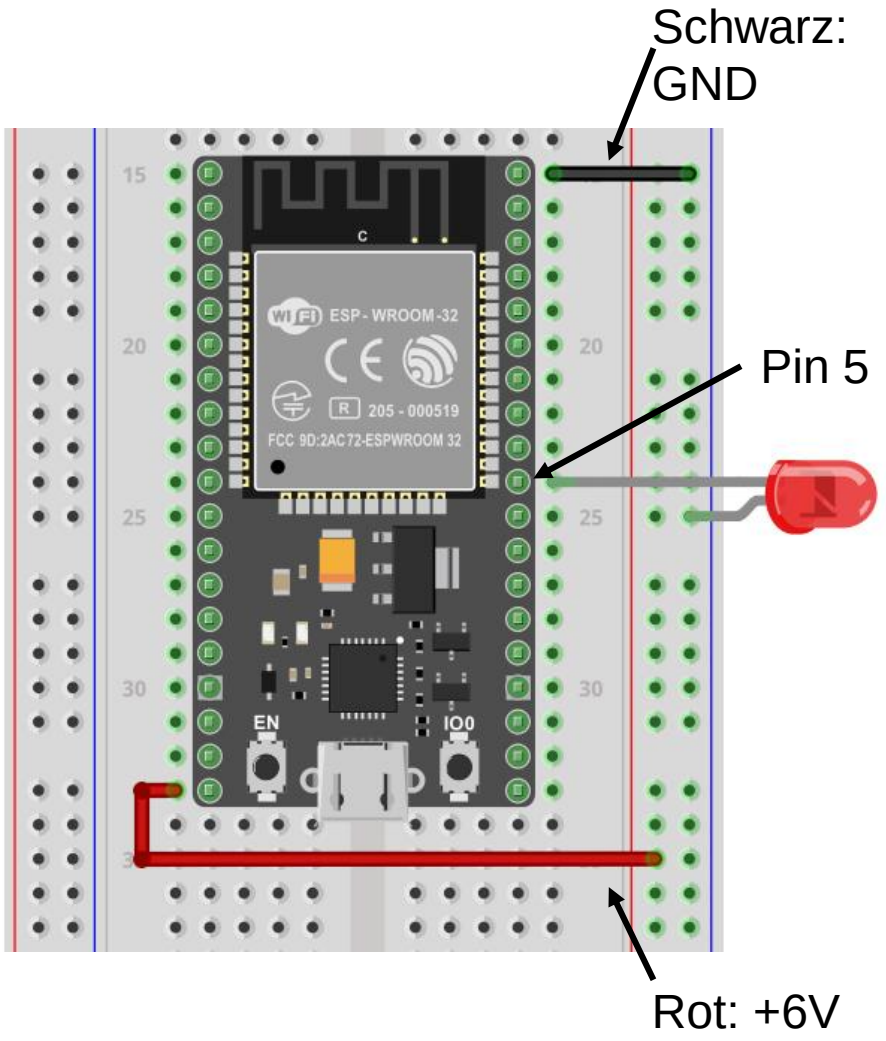
- wird ***immer wieder*** ausgeführt !

Blink Code in Arduino



Blink Code in Arduino





Blink Code in Arduino

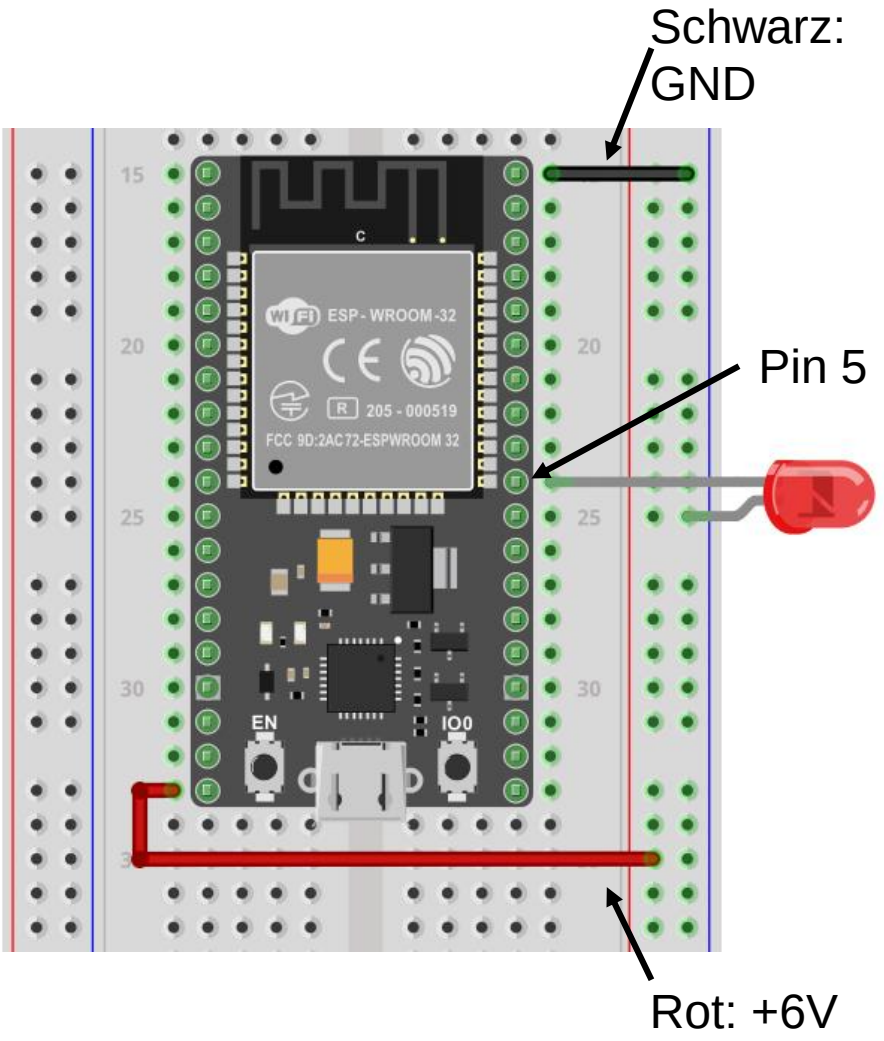
Blink | Arduino 1.8.3

Datei Bearbeiten Sketch Werkzeuge Hilfe

Blink §

```
/*
  Blink
*/

void setup() {

  // Wo ist die Leuchtdiode angeschlossen ?
  // Eingangspin (INPUT) oder Ausgangspin (OUTPUT)?
  pinMode(5, OUTPUT);

}


void loop() {

  // Leuchtdiode einschalten ... (HIGH)









}
```

ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4



Blink Code in Arduino

```
// Wo ist die Leuchtdiode angeschlossen ?
// Eingangspin (INPUT) oder Ausgangspin (OUTPUT)?
pinMode(5, OUTPUT);
```

pinMode(*Pin*, *Art des Pins*);

Nummer der Pins
→ 5

INPUT oder OUTPUT ? (…)
→ OUTPUT

Blink Code in Arduino

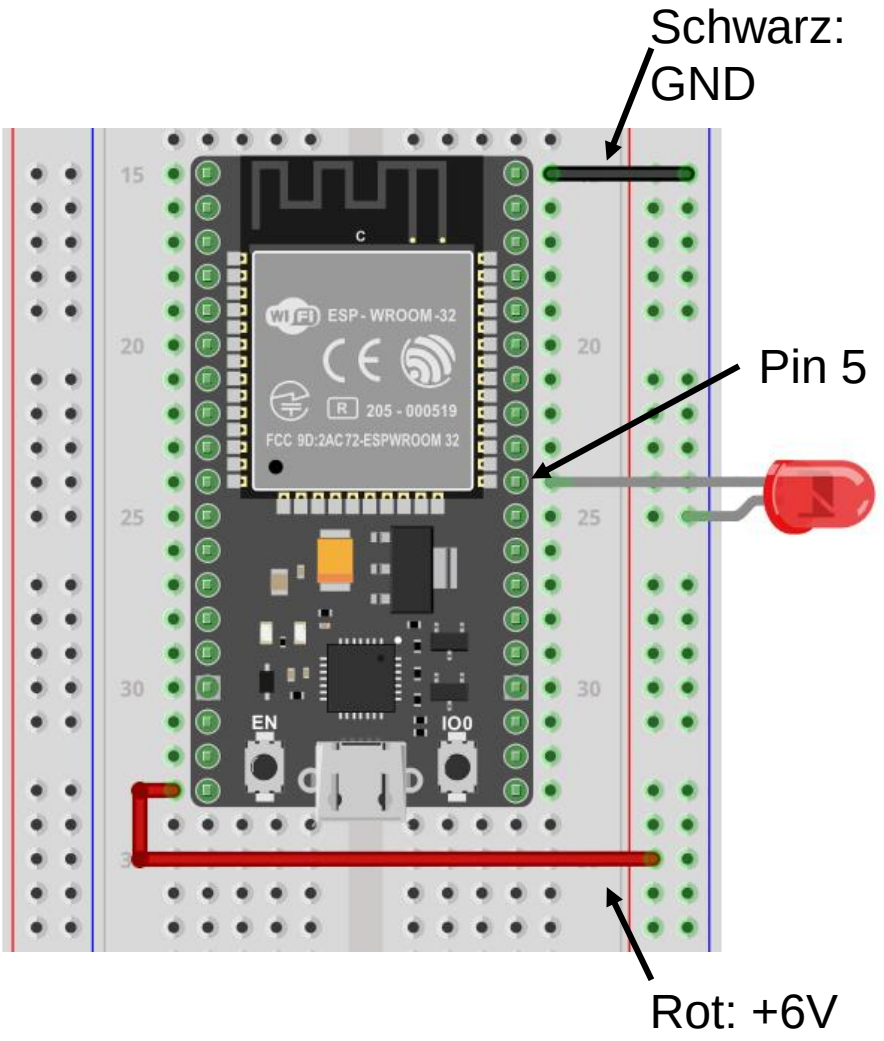
Blink | Arduino 1.8.3

Datei Bearbeiten Sketch Werkzeuge Hilfe

Blink §

```
/*
  Blink
*/

void setup() {

  // Wo ist die Leuchtdiode angeschlossen ?
  // Eingangspin (INPUT) oder Ausgangspin (OUTPUT)?
  pinMode(5, OUTPUT);

}


void loop() {

  // Leuchtdiode einschalten ...
  digitalWrite(5, HIGH);









}
```

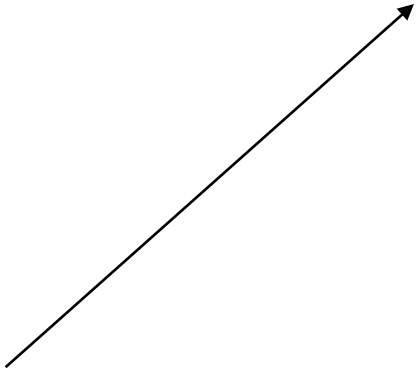
ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4



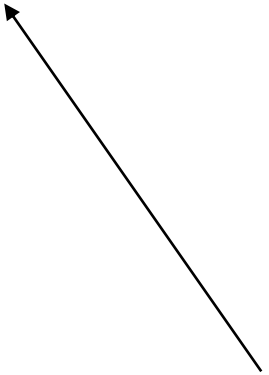
Blink Code in Arduino

```
// Leuchtdiode einschalten ...
digitalWrite(5, HIGH);
```

digitalWrite(*Pin*, *Ein-/Ausschalten*);



Nummer der Pins
→ 5



HIGH (LED an) oder LOW (LED aus) ?
→ HIGH

Blink Code in Arduino

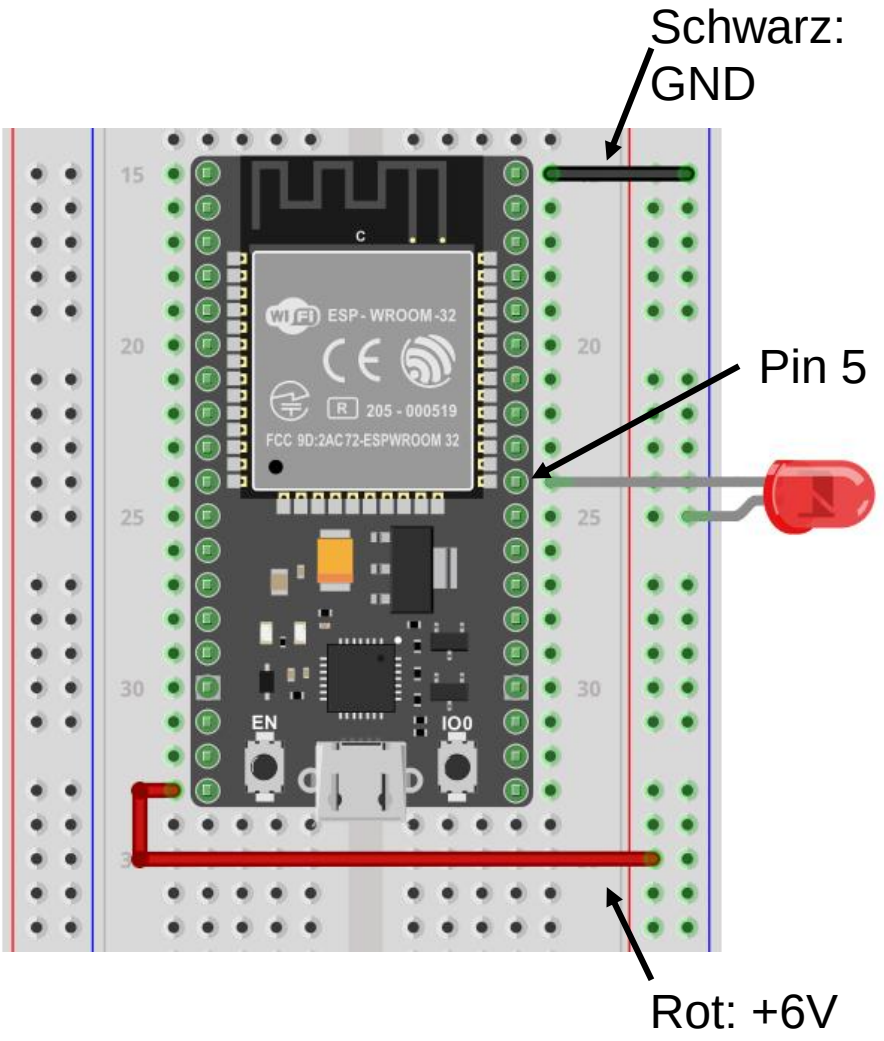
Blink | Arduino 1.8.3

Datei Bearbeiten Sketch Werkzeuge Hilfe

Blink §

```
/*
  Blink
*/

void setup() {

  // Wo ist die Leuchtdiode angeschlossen ?
  // Eingangspin (INPUT) oder Ausgangspin (OUTPUT)?
  pinMode(5, OUTPUT);

}


void loop() {

  // Leuchtdiode einschalten ... (HIGH)
  digitalWrite(5, HIGH);


  // Leuchtdiode ausschalten ...
  digitalWrite(5, LOW);


}
```

ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4



Blink Code in Arduino

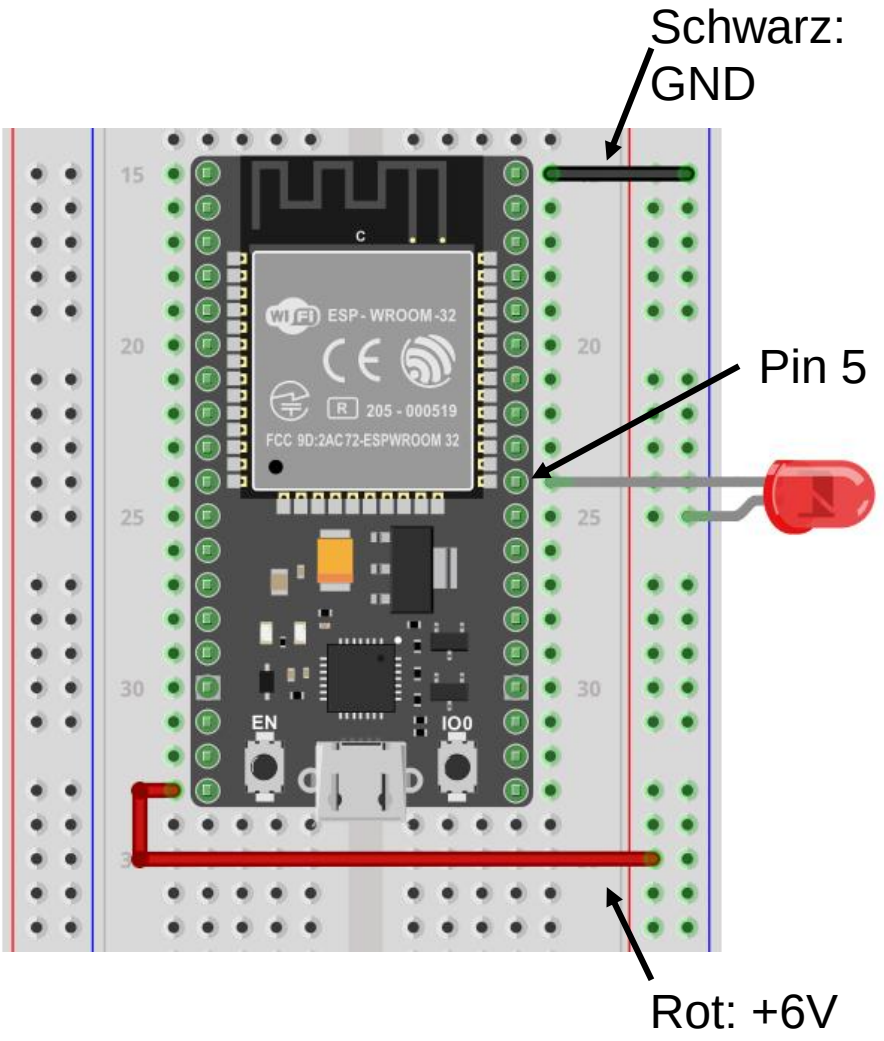
Blink | Arduino 1.8.3

Datei Bearbeiten Sketch Werkzeuge Hilfe

Blink §

```
/*
  Blink
*/

void setup() {

  // Wo ist die Leuchtdiode angeschlossen ?
  // Eingangspin (INPUT) oder Ausgangspin (OUTPUT)?
  pinMode(5, OUTPUT);

}


void loop() {

  // Leuchtdiode einschalten ... (HIGH)
  digitalWrite(5, HIGH);
  delay(1000);

  // Leuchtdiode ausschalten ...
  digitalWrite(5, LOW);
  delay(1000);

}
```

ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4



Blink Code in Arduino

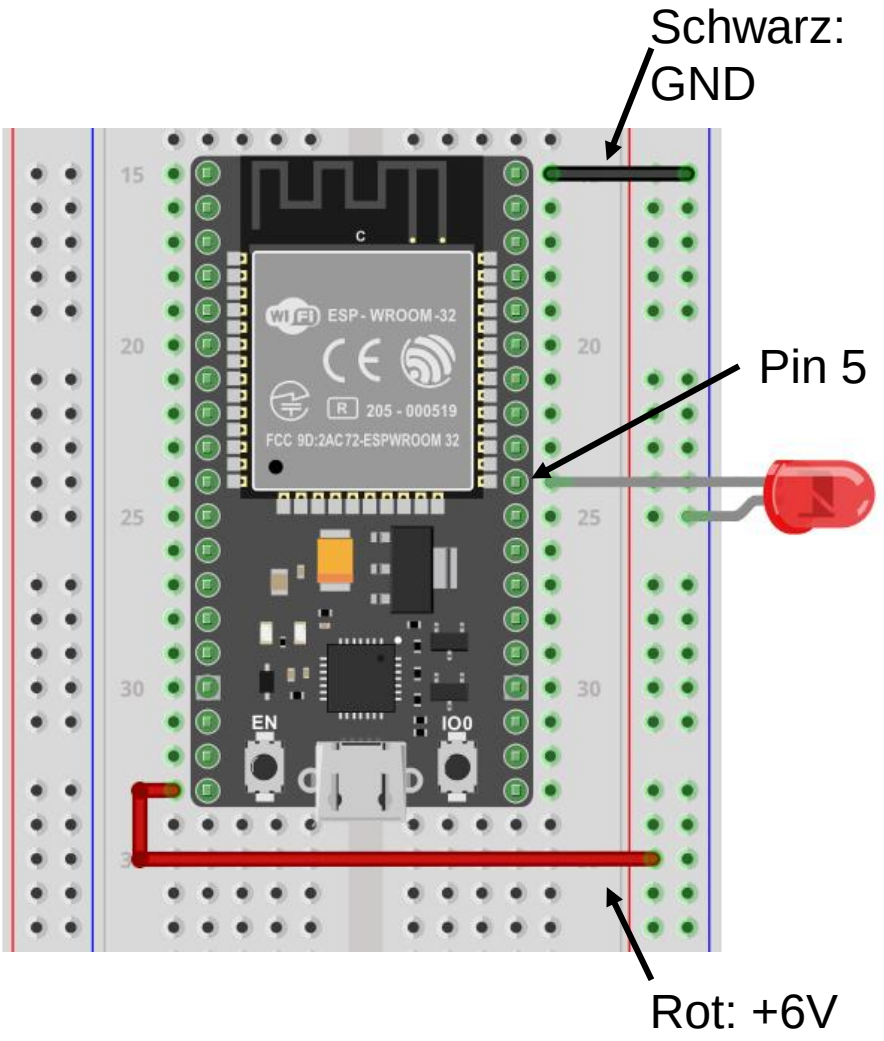
Blink | Arduino 1.8.3

Datei Bearbeiten Sketch Werkzeuge Hilfe

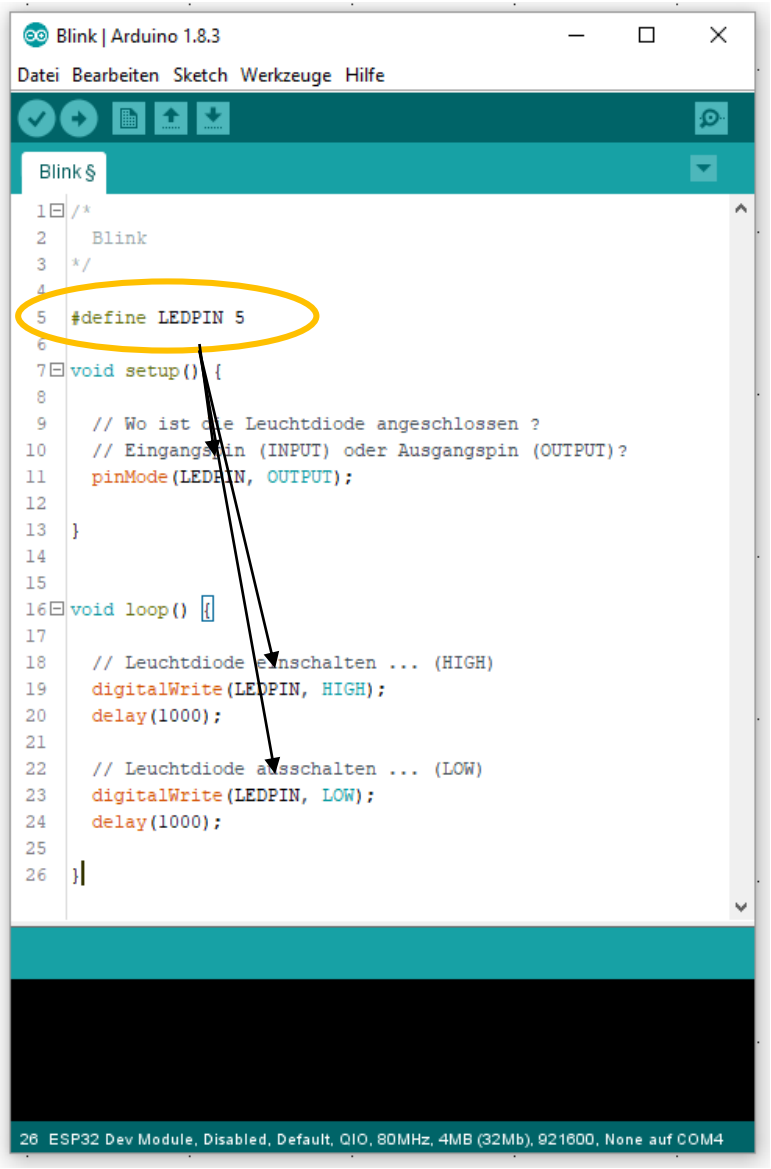
Blink §

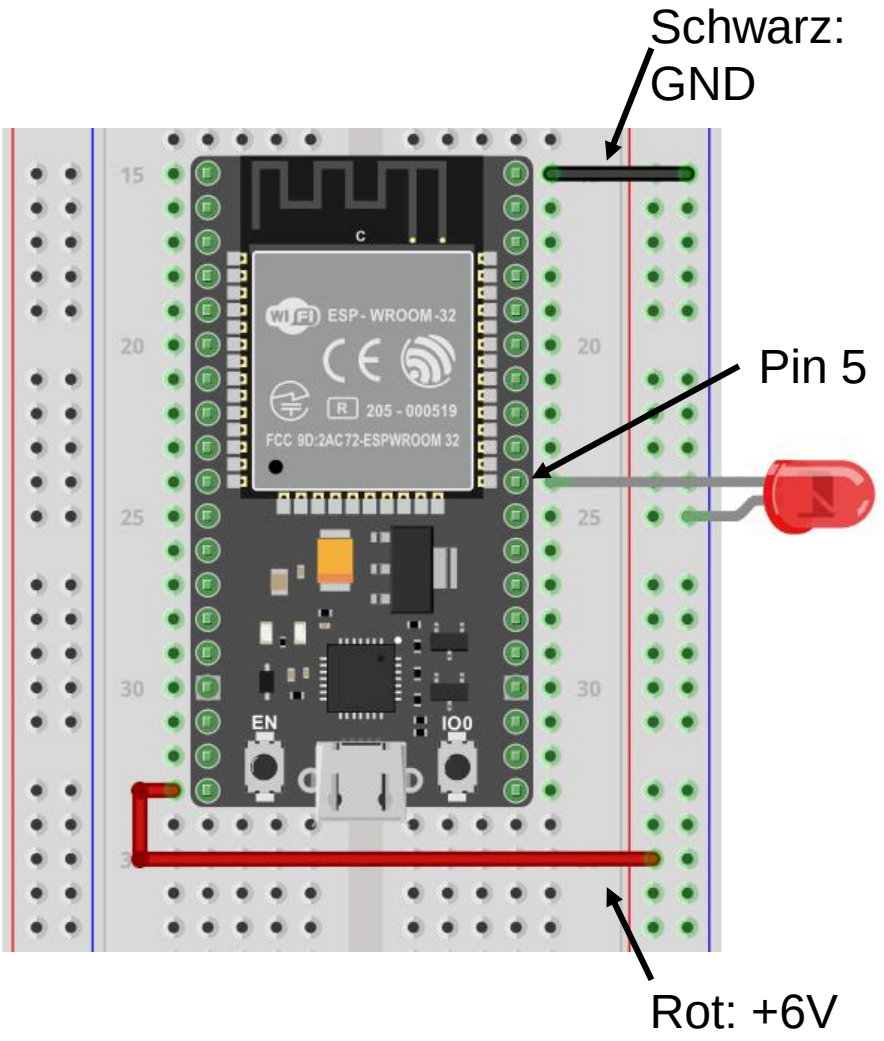
```
/*
  Blink
*/

#define LEDPIN 5

void setup() {

  // Wo ist die Leuchtdiode angeschlossen ?
  // Eingangspin (INPUT) oder Ausgangspin (OUTPUT)?
  pinMode(LEDPIN, OUTPUT);

}


void loop() {

  // Leuchtdiode einschalten ... (HIGH)
  digitalWrite(LEDPIN, HIGH);
  delay(1000);

  // Leuchtdiode ausschalten ... (LOW)
  digitalWrite(LEDPIN, LOW);
  delay(1000);

}
```

26 ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4

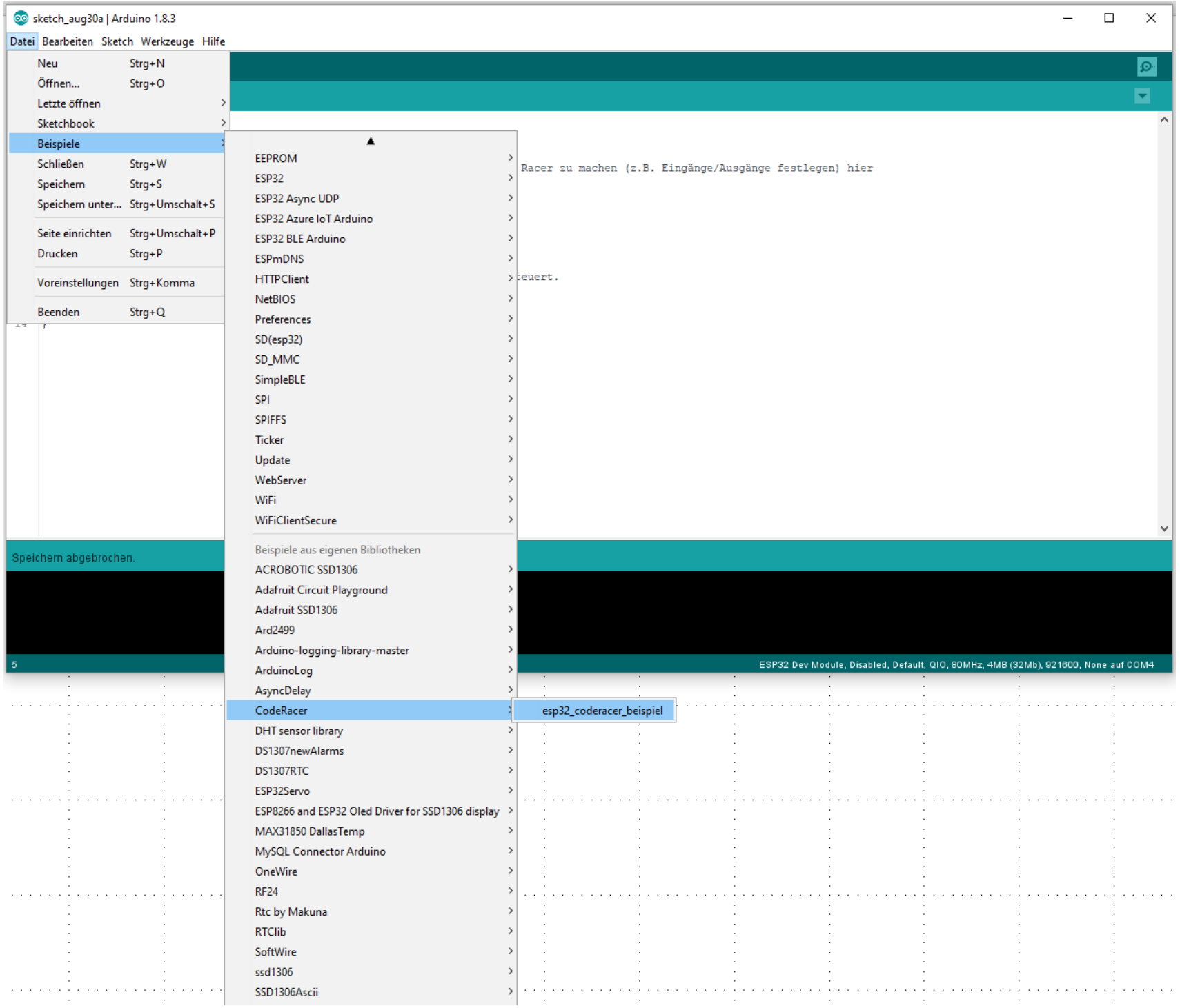


Blink Code in Arduino





CodeRacer in Arduino



esp32_coderacer_beispiel | Arduino 1.8.3

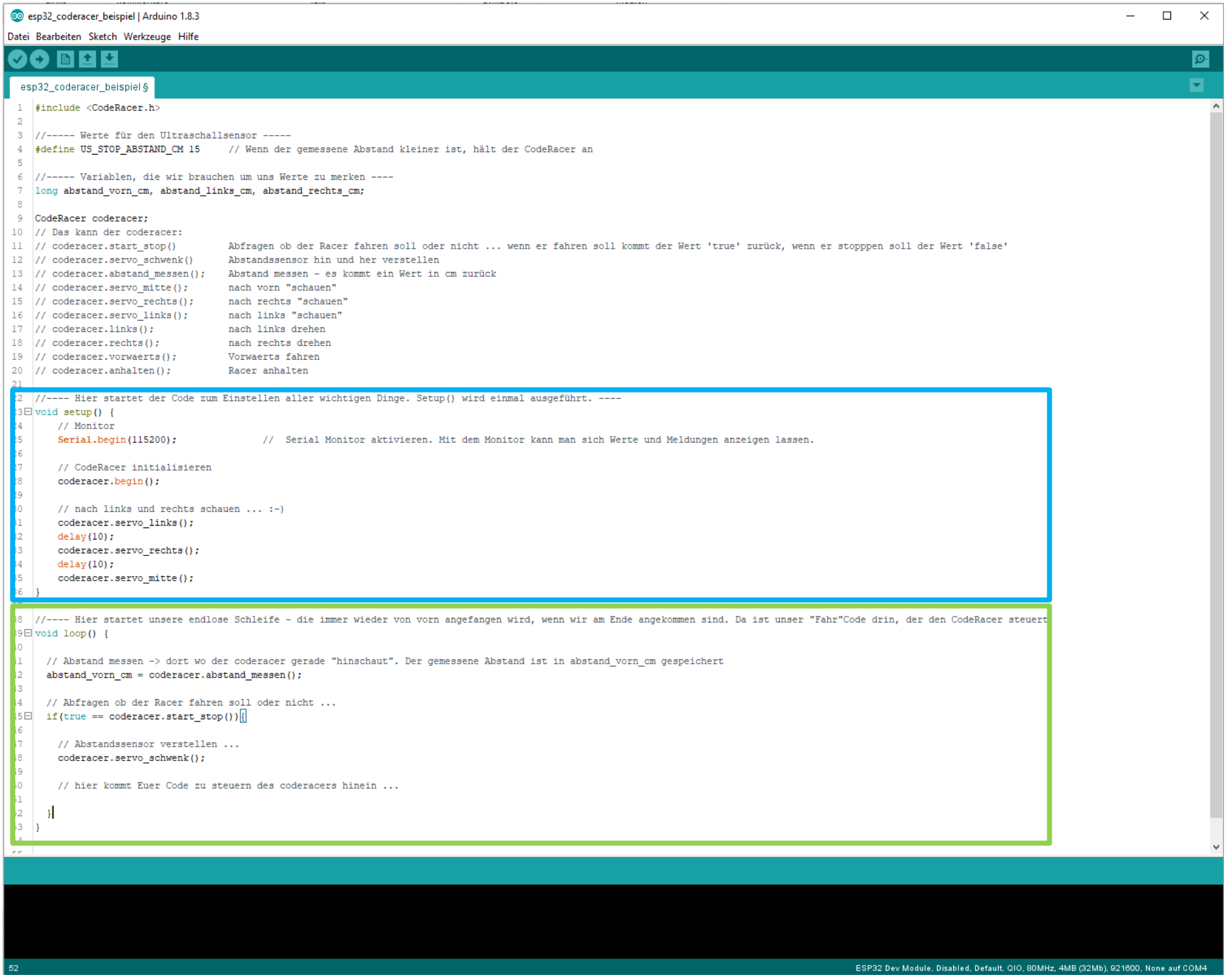
Datei Bearbeiten Sketch Werkzeuge Hilfe

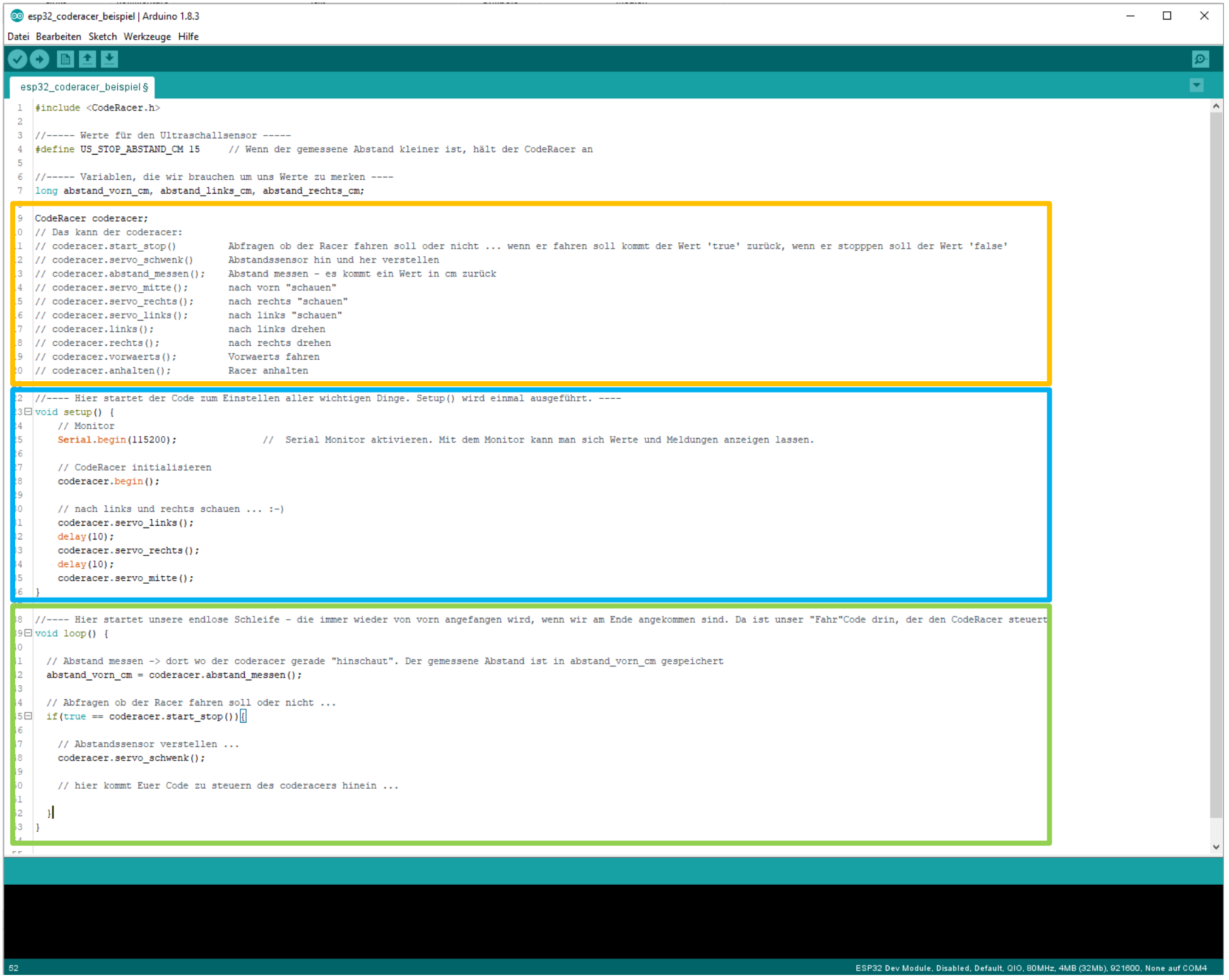
esp32_coderacer_beispiel §

```
#include <CodeRacer.h>

//----- Werte für den Ultraschallsensor -----
#define US_STOP_ABSTAND_CM 15     // Wenn der gemessene Abstand kleiner ist, hält der CodeRacer an

//----- Variablen, die wir brauchen um uns Werte zu merken ----
long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;

CodeRacer coderacer;
// Das kann der coderacer:
// coderacer.start_stop()         Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
// coderacer.servo_schwenk()      Abstandssensor hin und her verstellen
// coderacer.abstand_messen();    Abstand messen - es kommt ein Wert in cm zurück
// coderacer.servo_mitte();       nach vorn "schauen"
// coderacer.servo_rechts();      nach rechts "schauen"
// coderacer.servo_links();       nach links "schauen"
// coderacer.links();             nach links drehen
// coderacer.rechts();            nach rechts drehen
// coderacer.vorwaerts();         Vorwaerts fahren
// coderacer.anhalten();          Racer anhalten

//---- Hier startet der Code zum Einstellen aller wichtigen Dinge. Setup() wird einmal ausgeführt. ----
void setup() {
    // Monitor
    Serial.begin(115200);              //  Serial Monitor aktivieren. Mit dem Monitor kann man sich Werte und Meldungen anzeigen lassen.

    // CodeRacer initialisieren
    coderacer.begin();

    // nach links und rechts schauen ... :-)
    coderacer.servo_links();
    delay(10);
    coderacer.servo_rechts();
    delay(10);
    coderacer.servo_mitte();
}

//---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
void loop() {

  // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
  abstand_vorn_cm = coderacer.abstand_messen();

  // Abfragen ob der Racer fahren soll oder nicht ...
  if(true == coderacer.start_stop()){

    // Abstandssensor verstellen ...
    coderacer.servo_schwenk();

    // hier kommt Euer Code zu steuern des coderacers hinein ...

  }
}
```

52

ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4


esp32_coderacer_beispiel | Arduino 1.8.3
Datei Bearbeiten Sketch Werkzeuge Hilfe
esp32_coderacer_beispiel §
#include <CodeRacer.h>
//----- Werte für den Ultraschallsensor -----
#define US_STOP_ABSTAND_CM 15 // Wenn der gemessene Abstand kleiner ist, hält der CodeRacer an
//----- Variablen, die wir brauchen um uns Werte zu merken ----
long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
CodeRacer coderacer;
// Das kann der coderacer:
// coderacer.start_stop() Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
// coderacer.servo_schwenk() Abstandssensor hin und her verstellen
// coderacer.abstand_messen(); Abstand messen - es kommt ein Wert in cm zurück
// coderacer.servo_mitte(); nach vorn "schauen"
// coderacer.servo_rechts(); nach rechts "schauen"
// coderacer.servo_links(); nach links "schauen"
// coderacer.links(); nach links drehen
// coderacer.rechts(); nach rechts drehen
// coderacer.vorwaerts(); Vorwaerts fahren
// coderacer.anhalten(); Racer anhalten
//---- Hier startet der Code zum Einstellen aller wichtigen Dinge. Setup() wird einmal ausgeführt. ----
void setup() {
// Monitor
Serial.begin(115200); // Serial Monitor aktivieren. Mit dem Monitor kann man sich Werte und Meldungen anzeigen lassen.
// CodeRacer initialisieren
coderacer.begin();
// nach links und rechts schauen ... :-)
coderacer.servo_links();
delay(10);
coderacer.servo_rechts();
delay(10);
coderacer.servo_mitte();
}
//---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
void loop() {
// Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
abstand_vorn_cm = coderacer.abstand_messen();
// Abfragen ob der Racer fahren soll oder nicht ...
if(true == coderacer.start_stop()){
// Abstandssensor verstellen ...
coderacer.servo_schwenk();
// hier kommt Euer Code zu steuern des coderacers hinein ...
}
}
52
ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4


esp32_coderacer_beispiel | Arduino 1.8.3
Datei Bearbeiten Sketch Werkzeuge Hilfe
esp32_coderacer_beispiel §
#include <CodeRacer.h>
//----- Werte für den Ultraschallsensor -----
#define US_STOP_ABSTAND_CM 15 // Wenn der gemessene Abstand kleiner ist, hält der CodeRacer an
//----- Variablen, die wir brauchen um uns Werte zu merken ----
long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
CodeRacer coderacer;
// Das kann der coderacer:
// coderacer.start_stop() Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
// coderacer.servo_schwenk() Abstandssensor hin und her verstellen
// coderacer.abstand_messen(); Abstand messen - es kommt ein Wert in cm zurück
// coderacer.servo_mitte(); nach vorn "schauen"
// coderacer.servo_rechts(); nach rechts "schauen"
// coderacer.servo_links(); nach links "schauen"
// coderacer.links(); nach links drehen
// coderacer.rechts(); nach rechts drehen
// coderacer.vorwaerts(); Vorwaerts fahren
// coderacer.anhalten(); Racer anhalten
//---- Hier startet der Code zum Einstellen aller wichtigen Dinge. Setup() wird einmal ausgeführt. ----
void setup() {
// Monitor
Serial.begin(115200); // Serial Monitor aktivieren. Mit dem Monitor kann man sich Werte und Meldungen anzeigen lassen.
// CodeRacer initialisieren
coderacer.begin();
// nach links und rechts schauen ... :-)
coderacer.servo_links();
delay(10);
coderacer.servo_rechts();
delay(10);
coderacer.servo_mitte();
}
//---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
void loop() {
// Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
abstand_vorn_cm = coderacer.abstand_messen();
// Abfragen ob der Racer fahren soll oder nicht ...
if(true == coderacer.start_stop()){
// Abstandssensor verstellen ...
coderacer.servo_schwenk();
// hier kommt Euer Code zu steuern des coderacers hinein ...
}
}
52
ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4

CodeRacer in Arduino

```
9  CodeRacer coderacer;
10 // Das kann der coderacer:
11 // coderacer.start_stop()          Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
12 // coderacer.servo_schwenk()      Abstandssensor hin und her verstellen
13 // coderacer.abstand_messen();    Abstand messen - es kommt ein Wert in cm zurück
14 // coderacer.servo_mitte();       nach vorn "schauen"
15 // coderacer.servo_rechts();      nach rechts "schauen"
16 // coderacer.servo_links();       nach links "schauen"
17 // coderacer.links();             nach links drehen
18 // coderacer.rechts();            nach rechts drehen
19 // coderacer.vorwaerts();         Vorwaerts fahren
20 // coderacer.anhalten();          Racer anhalten
21
```

CodeRacer in Arduino

```
9  CodeRacer coderacer;
10 // Das kann der coderacer:
11 // coderacer.start_stop()          Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
12 // coderacer.servo_schwenk()       Abstandssensor hin und her verstellen
13 // coderacer.abstand_messen();     Abstand messen - es kommt ein Wert in cm zurück
14 // coderacer.servo_mitte();        nach vorn "schauen"
15 // coderacer.servo_rechts();       nach rechts "schauen"
16 // coderacer.servo_links();        nach links "schauen"
17 // coderacer.links();              nach links drehen
18 // coderacer.rechts();             nach rechts drehen
19 // coderacer.vorwaerts();          Vorwaerts fahren
20 // coderacer.anhalten();           Racer anhalten
21
```

Was hat und was kann ein {CODE}RACER ?

Aktoren & Sensoren

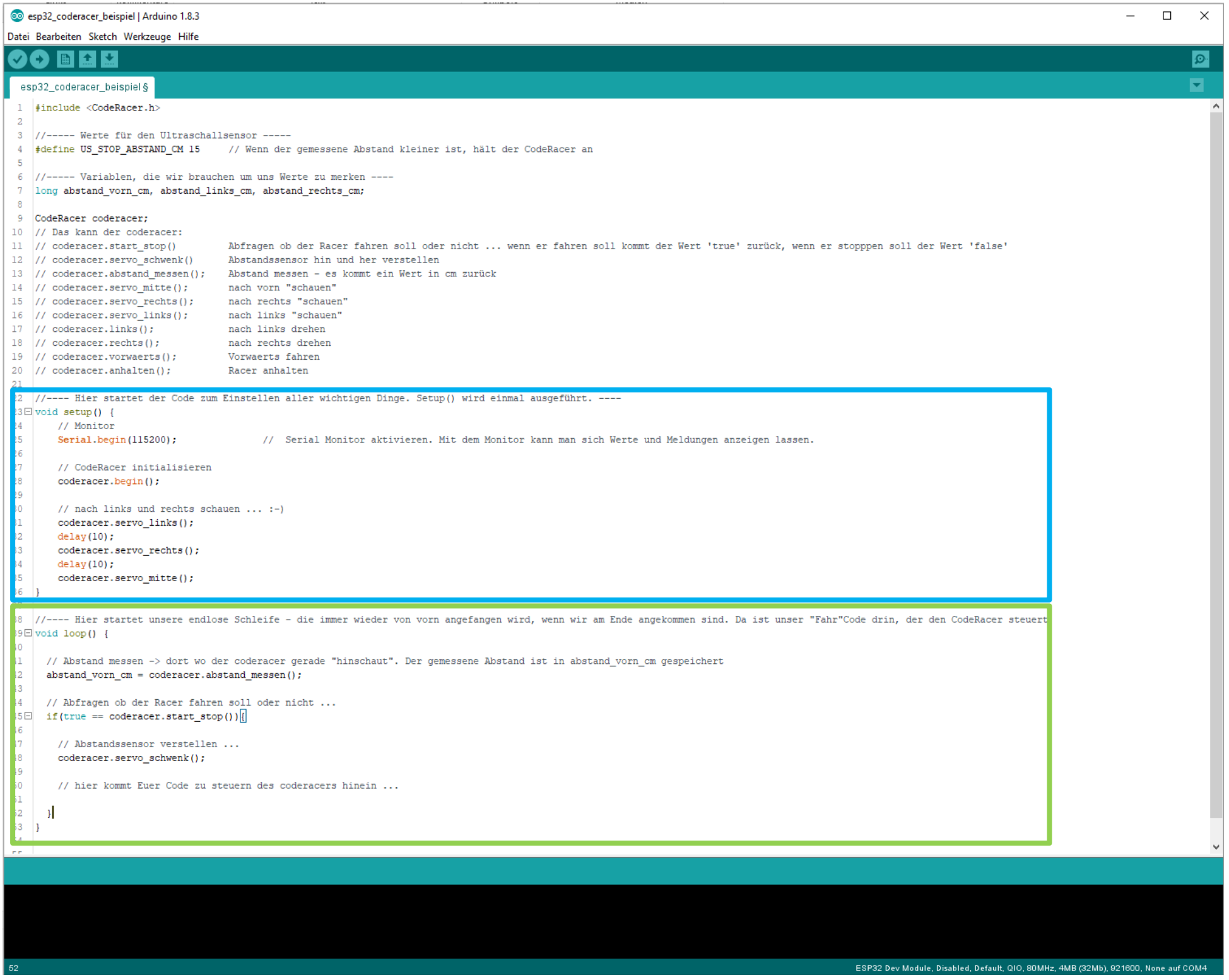
Antriebsmotoren

Servo

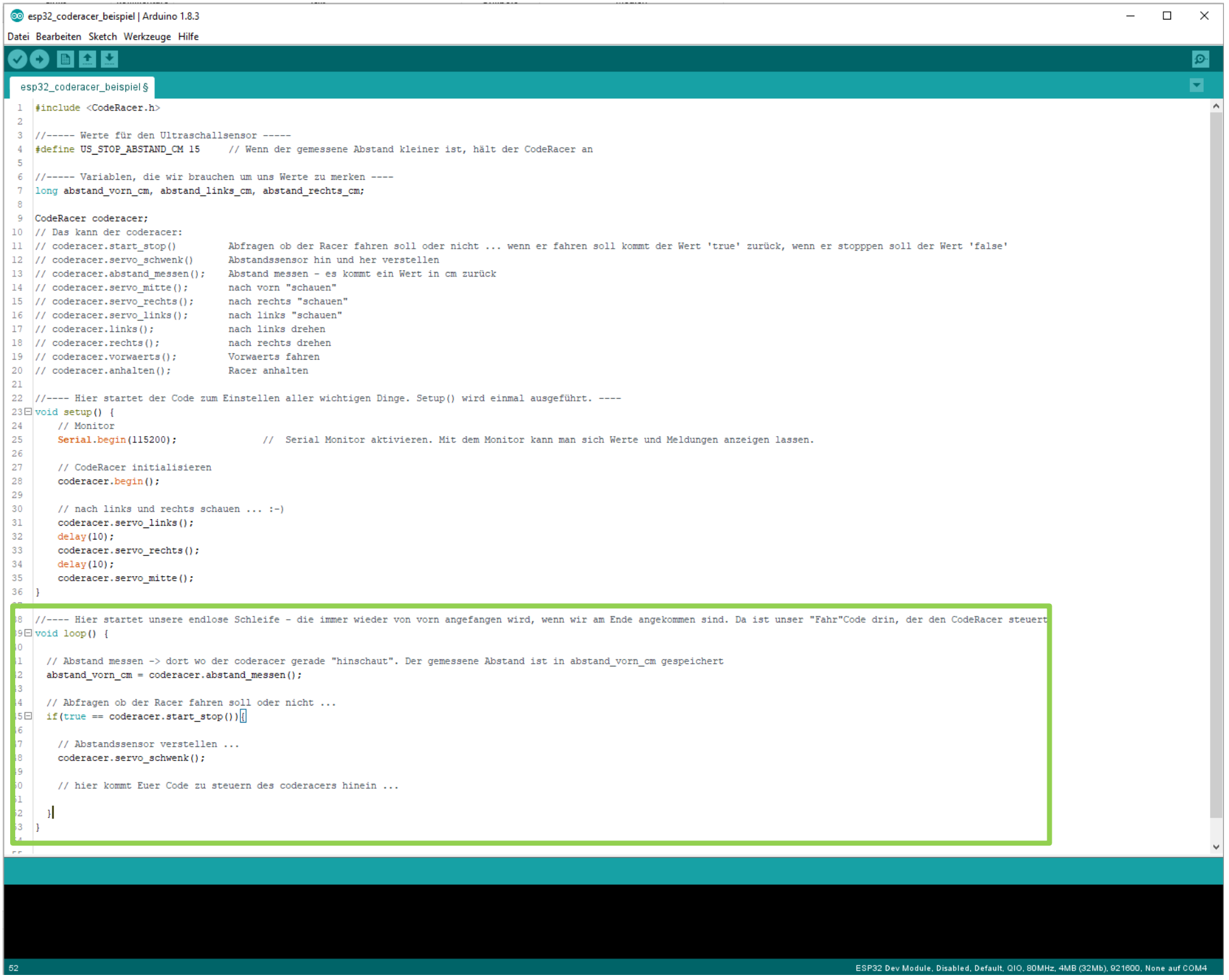
Ultraschallsensor

Funktionen

Antriebsmotoren	Servo	Ultraschallsensor
RACER_ANHALTEN - Motor links und rechts aus	**SERVO_MITTE** - Ultraschallsensor nach vorn	**ABSTAND MESSEN** - Abstand vorn (Servo in der Mitte)
RACER_VORWAERTS - Motor links und rechts vorwärts	**SERVO_LINKS** - Ultraschallsensor nach links drehen	
RACER_RUECKWAERTS - Motor links und rechts rückwärts	**SERVO_RECHTS** - Ultraschallsensor nach rechts drehen	
RACER_RECHTS_DREHEN - beide Motoren gleichzeitig: . links kurz vorwärts . rechts kurz rückwärts	**SERVO_SCHWENK** - Ultraschallsensor nach rechts und nach links hin und her drehen	
RACER_LINKS_DREHEN - beide Motoren gleichzeitig: . links kurz rückwärts . rechts kurz vorwärts		


esp32_coderacer_beispiel | Arduino 1.8.3
Datei Bearbeiten Sketch Werkzeuge Hilfe
esp32_coderacer_beispiel §
#include <CodeRacer.h>
//----- Werte für den Ultraschallsensor -----
#define US_STOP_ABSTAND_CM 15 // Wenn der gemessene Abstand kleiner ist, hält der CodeRacer an
//----- Variablen, die wir brauchen um uns Werte zu merken ----
long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
CodeRacer coderacer;
// Das kann der coderacer:
// coderacer.start_stop() Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
// coderacer.servo_schwenk() Abstandssensor hin und her verstellen
// coderacer.abstand_messen(); Abstand messen - es kommt ein Wert in cm zurück
// coderacer.servo_mitte(); nach vorn "schauen"
// coderacer.servo_rechts(); nach rechts "schauen"
// coderacer.servo_links(); nach links "schauen"
// coderacer.links(); nach links drehen
// coderacer.rechts(); nach rechts drehen
// coderacer.vorwaerts(); Vorwaerts fahren
// coderacer.anhalten(); Racer anhalten
//---- Hier startet der Code zum Einstellen aller wichtigen Dinge. Setup() wird einmal ausgeführt. ----
void setup() {
// Monitor
Serial.begin(115200); // Serial Monitor aktivieren. Mit dem Monitor kann man sich Werte und Meldungen anzeigen lassen.
// CodeRacer initialisieren
coderacer.begin();
// nach links und rechts schauen ... :-)
coderacer.servo_links();
delay(10);
coderacer.servo_rechts();
delay(10);
coderacer.servo_mitte();
}
//---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
void loop() {
// Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
abstand_vorn_cm = coderacer.abstand_messen();
// Abfragen ob der Racer fahren soll oder nicht ...
if(true == coderacer.start_stop()){
// Abstandssensor verstellen ...
coderacer.servo_schwenk();
// hier kommt Euer Code zu steuern des coderacers hinein ...
}
}
52
ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4

```
22  //---- Hier startet der Code zum Einstellen aller wichtigen Dinge. Setup() wird einmal ausgeführt. ----
23  void setup() {
24      // Monitor
25      Serial.begin(115200);              //  Serial Monitor aktivieren. Mit dem Monitor kann man sich Werte und Meldungen anzeigen lassen.
26
27      // CodeRacer initialisieren
28      coderacer.begin();
29
30      // nach links und rechts schauen ... :-)
31      coderacer.servo_links();
32      delay(10);
33      coderacer.servo_rechts();
34      delay(10);
35      coderacer.servo_mitte();
36  }
```


esp32_coderacer_beispiel | Arduino 1.8.3
Datei Bearbeiten Sketch Werkzeuge Hilfe
esp32_coderacer_beispiel §
#include <CodeRacer.h>
//----- Werte für den Ultraschallsensor -----
#define US_STOP_ABSTAND_CM 15 // Wenn der gemessene Abstand kleiner ist, hält der CodeRacer an
//----- Variablen, die wir brauchen um uns Werte zu merken ----
long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
CodeRacer coderacer;
// Das kann der coderacer:
// coderacer.start_stop() Abfragen ob der Racer fahren soll oder nicht ... wenn er fahren soll kommt der Wert 'true' zurück, wenn er stopppen soll der Wert 'false'
// coderacer.servo_schwenk() Abstandssensor hin und her verstellen
// coderacer.abstand_messen(); Abstand messen - es kommt ein Wert in cm zurück
// coderacer.servo_mitte(); nach vorn "schauen"
// coderacer.servo_rechts(); nach rechts "schauen"
// coderacer.servo_links(); nach links "schauen"
// coderacer.links(); nach links drehen
// coderacer.rechts(); nach rechts drehen
// coderacer.vorwaerts(); Vorwaerts fahren
// coderacer.anhalten(); Racer anhalten
//---- Hier startet der Code zum Einstellen aller wichtigen Dinge. Setup() wird einmal ausgeführt. ----
void setup() {
// Monitor
Serial.begin(115200); // Serial Monitor aktivieren. Mit dem Monitor kann man sich Werte und Meldungen anzeigen lassen.
// CodeRacer initialisieren
coderacer.begin();
// nach links und rechts schauen ... :-)
coderacer.servo_links();
delay(10);
coderacer.servo_rechts();
delay(10);
coderacer.servo_mitte();
}
//---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
void loop() {
// Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
abstand_vorn_cm = coderacer.abstand_messen();
// Abfragen ob der Racer fahren soll oder nicht ...
if(true == coderacer.start_stop()){
// Abstandssensor verstellen ...
coderacer.servo_schwenk();
// hier kommt Euer Code zu steuern des coderacers hinein ...
}
}
52
ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None auf COM4

```
6  //----- Variablen, die wir brauchen um uns Werte zu merken ----
7  long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
```

```
38 //---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
39 void loop() {
40
41   // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
42   abstand_vorn_cm = coderacer.abstand_messen();
43
44   // Abfragen ob der Racer fahren soll oder nicht ...
45   if(true == coderacer.start_stop()){
46
47     // Abstandssensor verstellen ...
48     coderacer.servo_schwenk();
49
50     // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52   }
53 }
```

```
//----- Variablen, die wir brauchen um uns Werte zu merken ----
long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
```

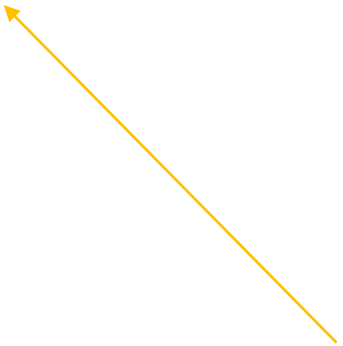
```
//---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
void loop() {

  // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
  abstand_vorn_cm = coderacer.abstand_messen();

  // Abfragen ob der Racer fahren soll oder nicht ...
  if(true == coderacer.start_stop()){

    // Abstandssensor verstellen ...
    coderacer.servo_schwenk();

    // hier kommt Euer Code zu steuern des coderacers hinein ...

  }
}
```

```
6  //----- Variablen, die wir brauchen um uns Werte zu merken ----
7  long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
8
```

„Gedächtnis“ zum merken vom Abstand

```
38 //---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
39 void loop() {
40
41   // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
42   abstand_vorn_cm = coderacer.abstand_messen();
43
44   // Abfragen ob der Racer fahren soll oder nicht ...
45   if(true == coderacer.start_stop()){
46
47     // Abstandssensor verstellen ...
48     coderacer.servo_schwenk();
49
50     // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52   }
53 }
54
```

```
41  // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
42  abstand_vorn_cm = coderacer.abstand_messen();
```



Abstand in cm messen
→ das „kann“ der CodeRacer

```
41  // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
42  abstand_vorn_cm = coderacer.abstand_messen();
```

Unser Gedächtnis (Speicher)
für den Abstand in cm

Abstand in cm messen
→ das „kann“ der CodeRacer

```
41  // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
42  abstand_vorn_cm = coderacer.abstand_messen();
```

= speichert den gemessenen Abstand in unserem „Gedächtnis“

Abstand in cm messen
→ das „kann“ der CodeRacer

Unser Gedächtnis (Speicher) für den Abstand in cm

```
6  //----- Variablen, die wir brauchen um uns Werte zu merken ----
7  long abstand_vorn_cm, abstand_links_cm, abstand_rechts_cm;
```

```
38 //---- Hier startet unsere endlose Schleife - die immer wieder von vorn angefangen wird, wenn wir am Ende angekommen sind. Da ist unser "Fahr"Code drin, der den CodeRacer steuert
39 void loop() {
40
41   // Abstand messen -> dort wo der coderacer gerade "hinschaut". Der gemessene Abstand ist in abstand_vorn_cm gespeichert
42   abstand_vorn_cm = coderacer.abstand_messen();
43
44   // Abfragen ob der Racer fahren soll oder nicht ...
45   if(true == coderacer.start_stop()){
46
47     // Abstandssensor verstellen ...
48     coderacer.servo_schwenk();
49
50     // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52   }
53 }
```

```
44  // Abfragen ob der Racer fahren soll oder nicht ...
45  if(true == coderacer.start_stop()){
46
47    // Abstandssensor verstellen ...
48    coderacer.servo_schwenk();
49
50    // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52  }
```

```
44  // Abfragen ob der Racer fahren soll oder nicht ...
45  if(true == coderacer.start_stop()){
46
47    // Abstandssensor verstellen ...
48    coderacer.servo_schwenk();
49
50    // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52  }
```

if(*Bedingung erfüllt***){***dann tue das hier alles …***;}**

Bedingungen:

==	- ist gleich	z.B:	5 == 5	← richtig oder falsch? (true oder false?)
			2 == 3	← true oder false ?
<	- kleiner als	z.B:	0 < 2	← ?
			4 < 2	← ?
			6 < 6	← ?
>	- größer als	z.B:	8 > 1	← ?
			3 > 10	← ?

```
44  // Abfragen ob der Racer fahren soll oder nicht ...
45  if(true == coderacer.start_stop()){
46
47    // Abstandssensor verstellen ...
48    coderacer.servo_schwenk();
49
50    // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52  }
```

if(*Bedingung erfüllt***){***dann tue das hier alles …***;}**

Bedingungen:

- == - ist gleich
- < - kleiner als
- > - größer als

Beispiel:

```
abstand_vorn_cm = 12;
if(abstand_vorn_cm < 10){  ???  ;}
```

```
44  // Abfragen ob der Racer fahren soll oder nicht ...
45  if(true == coderacer.start_stop()){
46
47    // Abstandssensor verstellen ...
48    coderacer.servo_schwenk();
49
50    // hier kommt Euer Code zu steuern des coderacers hinein ...
51
52  }
```

```
if(abstand_vorn < 10){
    coderacer.anhalten();
    coderacer.servo_links();
    abstand_links_cm = coderacer.abstand_messen()
}
```

Anhang – Bastelmaterialien ☺

ANTRIEBSMOTOREN

SERVO

ULTRASCHALL-SENSOR

RACER_VORWAERTS

SERVO_MITTE

ABSTAND MESSEN

RACER_ANHALTEN

ABSTAND MESSEN

SERVO_LINKS

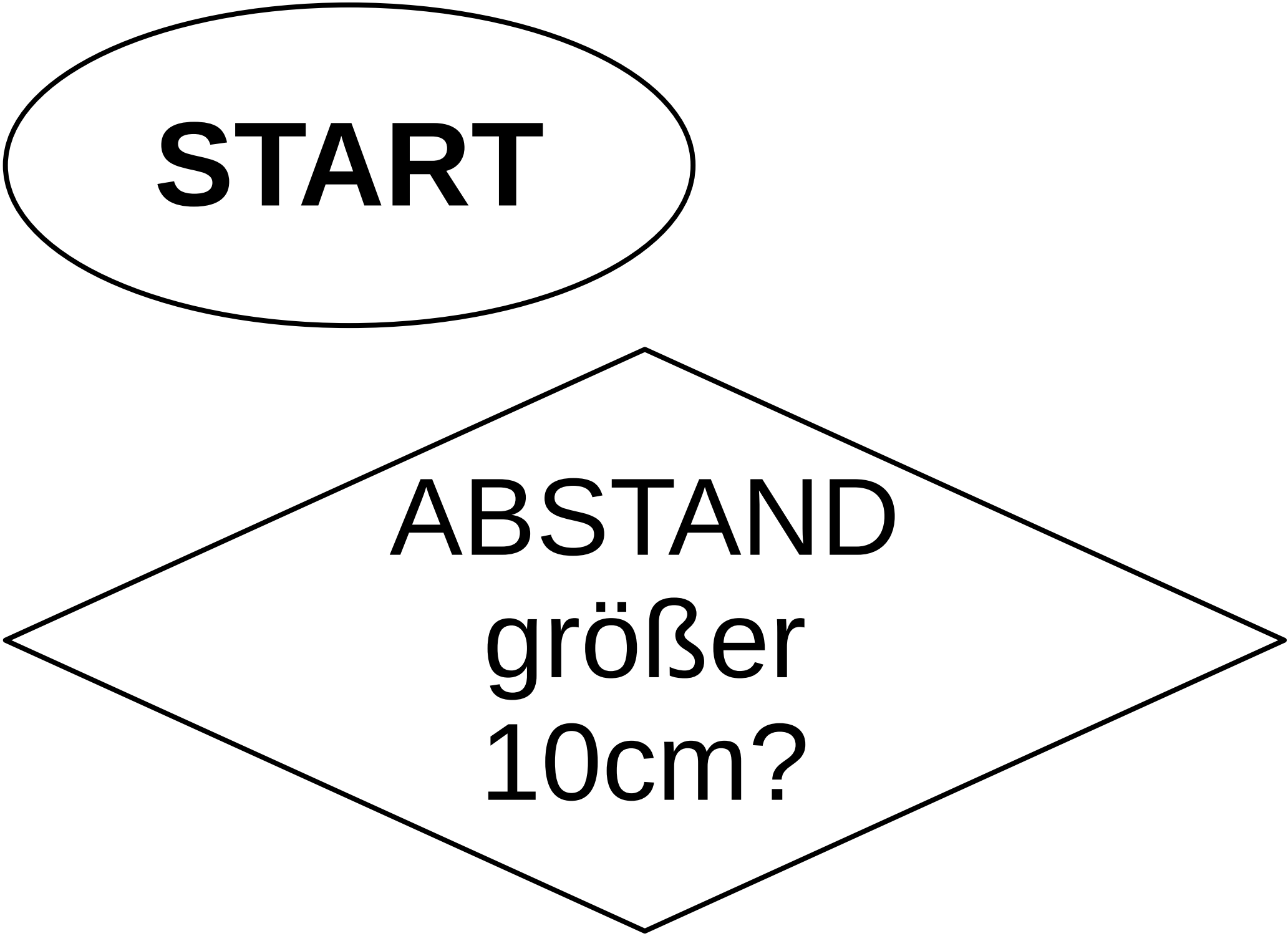
ABSTAND MESSEN

SERVO_RECHTS

ABSTAND MESSEN

RACER_RECHTS_DREHEN

RACER_LINKS_DREHEN


START
ABSTAND
größer
10cm?


welcher
ABSTAND
ist größer?

RACER ANHALTEN

- Motor links und rechts aus

RACER VORWAERTS

- Motor links und rechts vorwärts

RACER LINKS DREHEN

- Motoren gleichzeitig:
  . links kurz rückwärts
  . rechts kurz vorwärts

SERVO MITTE

- Servo in die Mitte drehen
- {CODE}RACER „schaut“ nach vorn

SERVO RECHTS

- Servo nach rechts drehen
- {CODE}RACER „schaut“ nach rechts

SERVO LINKS

- Servo nach links drehen
- {CODE}RACER „schaut“ nach links

SERVO SCHWENK

- Servo hin und herdrehen
- {CODE}RACER „schaut“ umher

ABSTAND MESSEN

- Abstand mit Ultraschall-sensor messen, in cm

RACER RUECKWAERTS

- Motor links und rechts rückwärts

RACER_RECHTS_DREHEN

- Motoren gleichzeitig:
  . links kurz vorwärts
  . rechts kurz rückwärts