

## Einheit: Bluetooth-gesteuerter Mini-AutoScooter mit Kollisionssensorik

<b>Inhalt der Einheit:</b>	Eigenen Autoscooter-Roboter aus dem MorphBot3d bauen
<b>Dauer der Einheit:</b>	6. Unterrichtsstunden
<b>Zielgruppe:</b>	Informatikschüler
<b>Methodik:</b>	Das Projekt ist so aufgebaut, dass es in einer vierer Gruppe mithilfe der Anleitung möglich ist, den Roboter aufzubauen
<b>Vorkenntnisse:</b>	Grundlegende Programmierkenntnisse und grundlegende Arduino Kenntnisse

<b>Benötigte Materialien:</b>	<ul style="list-style-type: none"> <li>• Siehe Arbeitsblatt</li> </ul>
-------------------------------	--

<b>Lernziele:</b>	<ul style="list-style-type: none"> <li>• Kompetenzen: Umgang mit dem ESP32, Umgang mit den Schaltkreisen, Umsetzen einer Motorsteuerung, Umgang mit Bluetooth</li> <li>• Befehle: digitalWrite, digitalRead, pinMode, millis, delay, analogRead, analogWrite, Bluetoothbibliothek (Bluepad32)</li> </ul>
-------------------	--

## Musterlösungen zu den Aufgaben

### Aufgabe 1

1a	<ul style="list-style-type: none"> <li>• LED (Leuchtdiode): Die LED wandelt elektrische Energie in Licht um. Sie wird im Roboter als Anzeigeelement für die Leben verwendet. Ein weiteres Einsatzgebiet wäre die Verwendung in Bildschirmen oder Beleuchtungssystemen.</li> <li>• Widerstand: Der Widerstand begrenzt den Stromfluss zur LED, um sie vor Überlastung zu schützen. Widerstände werden auch in vielen anderen elektronischen Schaltungen verwendet, z.B. zur Spannungsteilung oder Strombegrenzung.</li> <li>• DC-Motoren: Die zwei Motoren treiben die Hinterräder des Roboters an. Dadurch kann er sich fortbewegen. DC-Motoren werden auch in Spielzeugen, Modellautos und industriellen Maschinen eingesetzt.</li> <li>• H-Brücke: Die H-Brücke ermöglicht es, die Drehrichtung der Motoren zu ändern und ihre Geschwindigkeit zu steuern. Sie kommt auch in anderen Bereichen zum Einsatz, z. B. in ferngesteuerten Fahrzeugen oder Robotergreifarmen.</li> <li>• Piezo-Speaker: Der Piezo-Speaker erzeugt Töne und akustische Signale. Im Roboter kann er zur Signalisierung bestimmter Zustände oder Warnhinweise genutzt werden. Piezos werden auch in Weckern, Türgong-Systemen oder medizinischen Geräten eingesetzt.</li> <li>• ESP32 (Mikrocontroller): Der ESP32 ist die zentrale Steuereinheit des Roboters. Er verarbeitet die Signale und steuert Motoren, LEDs und den Speaker. Mikrocontroller wie der ESP32 werden in vielen IoT-Geräten, Smart-Home-Systemen oder Sensoranwendungen genutzt.</li> </ul>
1b	<p><b>Warum benötigt die LED einen Widerstand?</b>          Eine LED benötigt einen Vorwiderstand, um den Stromfluss zu begrenzen. Ohne Widerstand würde zu viel Strom fließen, wodurch die LED beschädigt oder zerstört werden könnte.</p> <p><b>Warum wurde ein 100-Ohm-Widerstand gewählt?</b>          Der Widerstandswert wird so berechnet, dass die LED mit einem sicheren Strom betrieben wird. In unserem Fall wurde ein <b>100-Ohm-Widerstand</b> gewählt, da er den Strom in einem Bereich hält, der für die LED geeignet ist (normalerweise 10-20 mA).</p> <p><b>Welche Farbcodierung hat ein 100-Ohm-Widerstand?</b>          Ein 100-Ohm-Widerstand hat die Farbcodierung:</p> <ul style="list-style-type: none"> <li>• Braun</li> <li>• Schwarz</li> <li>• Braun</li> <li>• Gold</li> </ul>

### Aufgabe 2

2	<p><b>Herstellen einer Bluetooth Verbindung mit einem PS4-Controller</b></p> <pre> 1 const int r2Threshold = 10; 2 const int l2Threshold = 10; 3 4 //... other Functions ... 5 6 void processGamepad(ControllerPtr ctl) {</pre>
---	---

```
7  if (ctl->brake() >= r2Threshold) {
8      Serial.println("r2 gedrueckt!");
9  }
10
11  if (ctl->throttle() >= l2Threshold) {
12      Serial.println("r2 gedrueckt!");
13  }
14 }
15
16 //... other Functions ...
```

### Aufgabe 3

#### 3 Frage

##### Aufgabe 3a:

```
1 void moveMotors(bool linksVorwaerts, bool rechtsVorwaerts) {
2
3
4     analogWrite(GSM1, 255);
5     analogWrite(GSM2, 255);
6
7     if (linksVorwaerts && rechtsVorwaerts) {
8         digitalWrite(in1, HIGH);
9         digitalWrite(in2, LOW);
10        digitalWrite(in3, HIGH);
11        digitalWrite(in4, LOW);
12    }
13
14    if (!linksVorwaerts && !rechtsVorwaerts) {
15        digitalWrite(in1, LOW);
16        digitalWrite(in2, HIGH);
17        digitalWrite(in3, LOW);
18        digitalWrite(in4, HIGH);
19    }
20 }

1 void processGamepad(ControllerPtr ctl) {
2
3
4     if (ctl->throttle() >= 10) { //R2
5
6         moveMotors(true,true);
7
8     }
9
10
11    if (ctl->brake() >= 10) { // L2
12
13        moveMotors(false,false);
14
15    }
16 }
```

**Aufgabe 3b:**

```
1 void moveMotors(bool linksVorwaerts, bool rechtsVorwaerts) {
2
3
4     analogWrite(GSM1, 255);
5     analogWrite(GSM2, 255);
6
7
8     if (linksVorwaerts && rechtsVorwaerts) {
9         digitalWrite(in1, HIGH);
10        digitalWrite(in2, LOW);
11        digitalWrite(in3, HIGH);
12        digitalWrite(in4, LOW);
13    }
14
15    if (!linksVorwaerts && !rechtsVorwaerts) {
16        digitalWrite(in1, LOW);
17        digitalWrite(in2, HIGH);
18        digitalWrite(in3, LOW);
19        digitalWrite(in4, HIGH);
20    }
21
22    if (!linksVorwaerts && rechtsVorwaerts) {
23        digitalWrite(in1, LOW);
24        digitalWrite(in2, HIGH);
25        digitalWrite(in3, HIGH);
26        digitalWrite(in4, LOW);
27    }
28
29    if (linksVorwaerts && !rechtsVorwaerts) {
30        digitalWrite(in1, HIGH);
31        digitalWrite(in2, LOW);
32        digitalWrite(in3, LOW);
33        digitalWrite(in4, HIGH);
34    }
35 }

1 void processGamepad(ControllerPtr ctl) {
2
3     if (ctl->throttle() >= 10) { //R2
4
5         if (ctl->axisX() <= 50 && ctl->joystick() >= -50) {
6 //gerade
7             moveMotors(true,true);
8         }
9
10        if (ctl->axisX() > 50) { //rechts
11            moveMotors(true, false);
12        }
13
14        if (ctl->axisX() < -50) { //links
15            moveMotors(false, true);
16        }
17
18    }
```

```
19
20  if (ctl->brake() >= 10) { // L2
21
22      if (ctl->axisX() <= 50 && ctl-> joystick() >= -50) {
23 //gerade
24      moveMotors(false, false);
25      }
26
27      if (ctl->axisX() > 50) { //rechts
28      moveMotors(false, true);
29      }
30
31      if (ctl->axisX() < -50) { //links
32      moveMotors(true, false);
33      }
34  }
35  }
```

HINWEIS: Wenn ihr das Codebeispiel testet, und R2/throttle() nicht wie geplant vorwärts fahren erzeugt, sondern rückwärtsfahren, dann müssen die Kabel des Motors (in1, in2) umgesteckt werden.

## Aufgabe 4

**4**

### Berührungserkennung

Dieser Code prüft, ob gerade eine Berührung stattfindet, und ob die letzte Berührung länger als 3 Sekunden her ist. Wenn zutrifft, werden leben verringert und auf der Konsole ausgegeben:

Diese beiden Variablen werden zu Beginn angelegt:

```
1
2  const int beruehrungskabel = 5;
   long letzteberuehrung = 0;
```

In der Setup Funktion wird der Pin für die Berührungserkennung als INPUT eingestellt:

```
3  void setup() {
4      pinMode(beruehrungskabel, INPUT);
5  }
```

Dies ist die Implementierung der Aufgabe in loop():

```
6
    bool istfahrzeuggetroffen =
    digitalRead(berührungskabel) == 1 &&
7    millis() - letzteberührung > 3000;
8
9    if (istfahrzeuggetroffen && leben >= 1) {
10        letzteberührung = millis();
        leben -= 1;
        Serial.printf("Leben: %d \n", leben);
    }
```

## Aufgabe 5

### 5 Lebensanzeige mit LEDs

Diese beiden Variablen werden zu Beginn angelegt:

```
1 int leben = 3;
2 long letzteberührung = 0;
```

In der Setup Funktion werden die LED Pins folgendermaßen eingerichtet:

```
void setup() {
1     pinMode(Led1, OUTPUT);
2     pinMode(Led2, OUTPUT);
3     pinMode(Led3, OUTPUT);
4
5     digitalWrite(Led1,
6 HIGH);
7     digitalWrite(Led2,
8 HIGH);
9     digitalWrite(Led3,
10 HIGH);
11
12 //restlicher setup code
}
```

Mit dieser Codeerweiterung wird das Fahren ohne verbleibende Leben verhindert:

```
1 void moveMotors(bool linksVorwaerts, bool rechtsVorwaerts) {
2
```

```
3   if( leben < 1) {
4       analogWrite(GSM1, 0); //Motorengeschwindigkeit = 0 setzen
5       analogWrite(GSM2, 0);
6   } else {
7       analogWrite(GSM1, 255);
8       analogWrite(GSM2, 255);
9   }
10
11 //restlicher Code
12 }
```

Mit dieser Codeerweiterung ist es möglich, durch X- drücken die Leben wieder aufzufüllen:

```
1 void processGamepad(ControllerPtr ctl) {
2
3
4 //== PS4 X button = 0x0001 ==//
5   if (ctl->buttons() == 0x0001 && leben < 1) {
6       //alle 3 leds anmachen (der Reihe nach)
7       digitalWrite(Led1, HIGH);
8       delay(1000);
9       digitalWrite(Led2, HIGH);
10      delay(1000);
11      digitalWrite(Led3, HIGH);
12      delay(1000);
13      leben = 3;
14  }
15
16 //restlicher Code
17 }
```

Mit dieser Codeerweiterung der loop Funktion wird das Piepen bei einem Treffer umgesetzt und die LEDs nach und nach abgeschaltet:

```
1 void loop() {
2
3     bool istfahrzeuggetroffen =
4 (digitalRead(beruhrungskabel) == 1) &&
5 ( (millis() - letzteberuhrung) > 3000);
6     static bool doppelPiepen = false;
7     if (istfahrzeuggetroffen && leben >= 1) {
8         letzteberuhrung = millis();
9         leben -= 1;
10        Serial.printf("Leben: %d \n", leben);
11
12        doppelPiepen = true;
13        digitalWrite(beep, HIGH);
14
15
16        switch (leben) {
17        case 2:
```

```
18         // led3 ausmachen
19         digitalWrite(Led3, LOW);
20         break;
21     case 1:
22         // led2 ausmachen
23         digitalWrite(Led2, LOW);
24         break;
25     case 0:
26         // led1 ausmachen
27         digitalWrite(Led1, LOW);
28         break;
29     }
30
31 }
32
33 if (millis() > letzteberuhrung+100) {
34     digitalWrite(beep, LOW);
35 }
36 if (millis() > letzteberuhrung+3000 && doppelPiepen) {
37     digitalWrite(beep, HIGH);
38     delay(100);
39     digitalWrite(beep, LOW);
40     delay(100);
41     digitalWrite(beep, HIGH);
42     delay(100);
43     digitalWrite(beep, LOW);
44     doppelPiepen = false;
45 }
46
47
48 bool dataUpdated = BP32.update();
49 if (dataUpdated) {
50     processControllers();
51 }
52 delay(150);
53 }
```