Inhaltsverzeichnis

Was macht die App?

Die App RemoCar (Remote Car) ermöglicht das Fernsteuern eines Raspberry Pi's (Micro-Computer), welches an einem Modellauto angeschlossen ist und diese Motoren somit steuern kann. Eine genauere Erklärung folgt in Kapitel 3.

Bedienung

Um die App zu installieren benötigen sie Android Studio und bestenfalls ein Android Handy.

Öffnen sie anschließend das Projekt „Androidcar" in Android Studio und wählen sie oben ihr angeschlossenes Handy aus. Drücken sie auf „Run App" und die App installiert sich auf ihrem Handy.

Beim Öffnen der App erscheint dieses Fenster, mit dem sie das Auto steuern können. Stellen sie außerdem sicher, dass ihr Handy als auch der Raspberry Pi mit dem Internet verbunden ist. Sobald der Raspberry Pi mit Strom versorgt ist startet das Python-Programm automatisch.

1. Über die vier Pfeile (vorwärts, rückwärts, rechts, links) können sie die Richtung steuern, in die das Auto fahren soll. Beim Steuern muss außerdem der Button gedrückt und gehalten werden und sobald man wieder loslässt bleibt das Auto stehen.
2. Mit dem Button „Foto" können sie mit der Kamera, die am Raspberry Pi angeschlossen ist ein Foto machen und dieses Foto wird anschließend auf dem Fenster oben angezeigt.
3. Wenn sie den Button „Kreis drehen" betätigen, dreht sich das Auto Rechts im Kreis.
4. Mit dem Button „Autonom" fährt das Auto eine zufällige Strecke automatisch
5. Die Anzeige „Abstand: 23cm" stellt den Abstand Zentimetergenau zu einem Hindernis (Wand, Tür) dar, welcher mit einem Ultraschallsensor ermittelt wird.

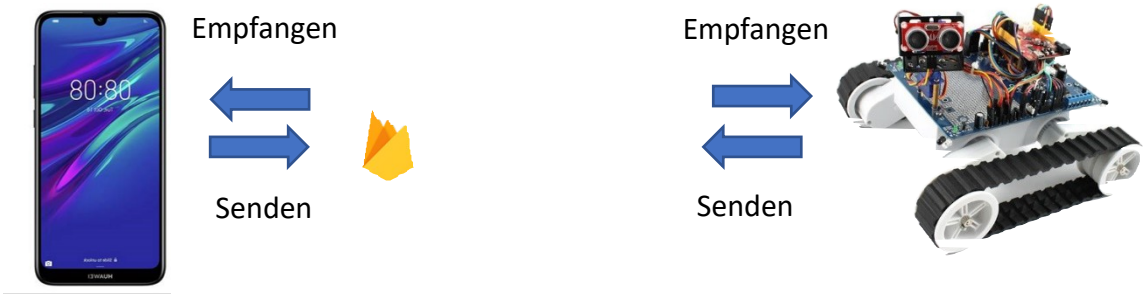
Die Latenz also vom drücken des Buttons bis Bewegung der Motoren beträgt bei normal schnellem Internet ca. 30ms.

Wenn sie den Raspberry Pi mit einem Mobilfunkhandy verbinden, können sie auch problemlos das Auto draußen oder im Wald von Zuhause aus fernsteuern.

Unter Dauerlast der Motoren können sie außerdem ca.2 Stunden fahren.

Funktionsweise

App Datenbank Raspberry Pi



App

Sobald ein Button gedrückt wird, wird eine Nachricht in eine Firebase Datenbank geschrieben.

In diesem Fall geht es um den Button_vor für das Vorwärtsfahren. So lange dieser Button gedrückt wird, wird die Funktion Send() mit dem Parameter “Vor“ ausgeführt und die Nachricht “Vor“ in die Datenbank geschrieben. Wenn der Button losgelassen wird, wird “Stehen“ in die Datenbank geschrieben.

Bei dem Button_Foto wird dann “Foto“ in die Datenbank geschrieben und bei Button autonom “autonom“.

```
1 final Button button_vor = findViewById(R.id.button2);
2 button_vor.setOnTouchListener(new View.OnTouchListener() {
3     @Override
4     public boolean onTouch(View v, MotionEvent event) {
5         switch (event.getAction()) {
6             case MotionEvent.ACTION_DOWN:
7                 Log.d(TAG, "Vor");
8                 Send("Vor");
9                 return true;
10
11            case MotionEvent.ACTION_UP:
12                Log.d(TAG, "Stehen");
13                Send("Stehen");
14                return true;
15        }
16        return false;
17    }
18 });
19
```

Funktion für Vorwärts-fahren

Bei der Funktion Send() wird erstmal eine Verbindung mit der Firebase Datenbank hergestellt. Der Login-Key für diese Datenbank ist aus Sicherheitsgründen in der google-services.json Datei im Verzeichnis Modules versteckt. Falls ein Angreifer diese App Dekompiliert und dadurch den Code einsehen kann wird der Login-Key nicht zu sehen sein.

Danach wird der Pfad angegeben, in der die Nachricht “msg“ reingeschrieben werden soll.

```
1 public void Send(String msg) {
2         FirebaseDatabase database = FirebaseDatabase.getInstance();
3         DatabaseReference myRef = database.getReference("/car/message");
4         myRef.setValue(msg);
5 }
6
```

Funktion Send()

Mit dieser Funktion Get() wird die Entfernung aus dem Eintrag “entfernung“ ausgelesen, in der Variable text2 gespeichert und danach in dem TextView “text“ angezeigt. Die Strings “Abstand:“, text2 und “cm“ werden addiert und es entsteht eine Informative anzeige, welche in Echtzeit aktualisiert.

```
1 public void Get() {
2         FirebaseDatabase database2 = FirebaseDatabase.getInstance();
3         DatabaseReference myRef2 = database2.getReference("/car/entfernung");
4
5         ValueEventListener postListener = new ValueEventListener() {
6             @Override
7             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
8                 Integer text2 = dataSnapshot.getValue(Integer.class);
9                 Log.d(TAG, String.valueOf(text2));
10
11                final TextView text = findViewById(R.id.textView);
12                text.setText(String.valueOf("Abstand: " + text2 + " cm"));
13            }
14        };
15        myRef2.addValueEventListener(postListener);
16 }
```

Funktion Get()

Datenbank

Die Firebase Datenbank von Google ist Open Source und kann von jedem genutzt werden.

Bei dieser Echtzeit Datenbank wird dann anschließend unter dem Eintrag “message“ die Nachricht “Stehen“ reingeschrieben. Die Entfernung als Abstand zu einem Objekt misst der Ultraschallsensor und der Raspberry Pi schreibt dann diesen Wert in Zentimetern in diese Datenbank.

Dadurch, dass diese Datenbank Online ist und auf den Servern von Google gehostet wird, ist es möglich, das Auto von überall auf der Welt zu steuern. Es muss lediglich das Handy und der Raspberry Pi mit dem Internet verbunden sein.

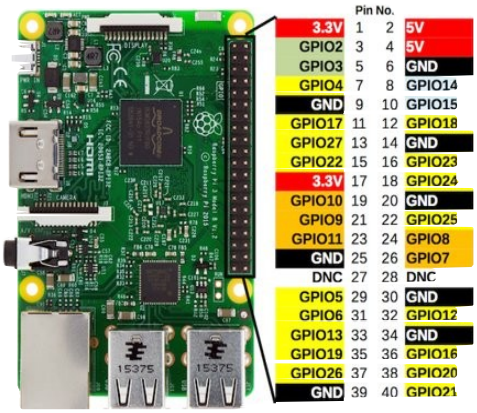
Raspberry Pi

In diesem Projekt ist der Raspberry Pi an einem Chassis Bausatz angebaut und steuert über die GPIO Pins (General Input Output), welches Signal an die Motoren übergeben werden soll.

In diesem Fall geht es um die Funktion vor(), um nach vorne zu fahren. Dabei wird auf den Pins In3(Input3) und In1 ein High-Pegel angelegt. D.h. diese werden als 1 codiert und die anderen beiden Pins als 0, was dazu führt, dass alle vier Motoren in die gleiche Richtung drehen ⟹ vorwärts fahren

```
def vor():
        pwmA.start(100)
        gpio.output(In3, True)
        gpio.output(In4, False)
        pwmB.start(100)
        gpio.output(In1, True)
        gpio.output(In2, False)
        abstandmessen()
```

GPIO-Pins

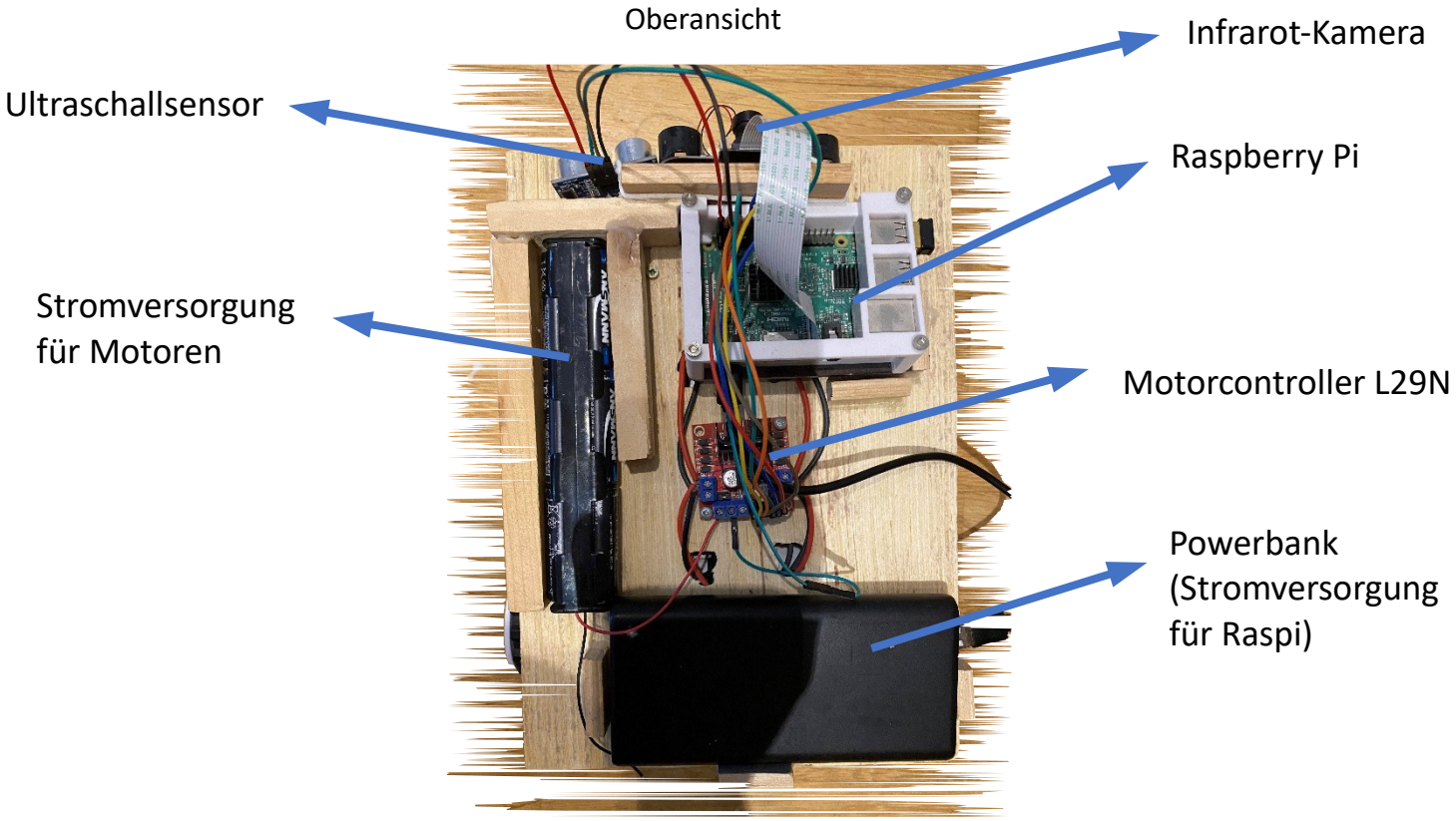


Außerdem ist am Raspberry Pi (Raspi) noch eine Infrarot-Kamera angeschlossen, die es erlaubt Nachtaufnahmen zu machen.

Mit dem Ultraschallsensor kann die Distanz nach vorne Zentimeter genau ermittelt werden, um z.B. eine Kollision zu verhindern und das Auto sofort zum Stehen zu bringen. Diese Distanz wird dann vom Raspi in Echtzeit in die Datenbank geschrieben.

Der Raspi loggt sich also in die Datenbank ein und schaut, welche Nachricht enthalten ist. Steht “Vor“ drinnen, wird demnach sofort die Funktion vor() ausgeführt.

Oberansicht



Hier sehen sie den Python Code für den Ultraschallsensor, der technisch wirklich ziemlich interessant ist.

1.) Hier wird der Pin Trigger um 10 µs aktiviert und dadurch werden Schallimpulse in die Umgebung ausgesendet.

2.) Anschließend wird ein Timer gestartet, der misst, wie lange es braucht, bis an dem Pin echo ein reflektierter Schall empfangen wird.

3.) Die Zeit, die die Schallwelle für den Hin- und Rückweg gebraucht hat wird dann mit der Schallgeschwindigkeit 343 m/s multipliziert und danach nochmal mit 100, um den Wert in cm auszugeben. Zu guter Letzt wird alles durch 2 geteilt, um nur die Wegstrecke zu erhalten.

4.) Wenn die Entfernung hier kleiner als 15cm beträgt fährt das Auto sofort zurück, um eine Kollision zu verhindern.

```python
def abstandMessen():
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        while gpio.input(echo) == 0:
                pass

        start = time.time()

        while gpio.input(echo) == 1:
                pass

        ende = time.time()
        entfernung = int(((ende - start) * 34300) / 2)

        file2 = db.child("car")
        file2.update({'entfernung' : entfernung})

        if entfernung < 15:
                print("zurück")
                back()
```

1

2

3

4

Anwendung

Für dieses Auto gibt es zahlreiche Anwendungen wie z.B. das bequeme Erkunden der Umgebung, ohne selber herumzulaufen zu müssen oder auch das leise Ausspähen von Familienmitgliedern.

Außerdem könnte ich, während ich hier in München sitze, dieses Auto in Echtzeit Problemlos in Bangladesch Steuern und Fotos von besonderen Tierarten oder sonstige Wartungsarbeiten machen.

Wenn man jetzt etwas weiter denkt, wäre es wirklich sinnvoll, dieses Auto in Krisen oder Radioaktiven Gebieten zu verwenden. Ein Vorteil ist, dass dadurch keine Menschen gefährdet werden und, dass man relativ Sicher sich einen Überblick über die Situation verschaffen kann.

Quellen

Firebase Realtime Database | Firebase Documentation (google.com)

Objekte hochladen | cl | Google Cloud

Developer Guides | Android Developers

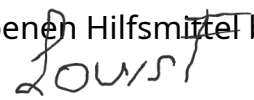
Retrieve Image From Firebase | Load Image From Firebase Android Studio - YouTube

Rheinwerk Technik – Raspberry Pi Das Umfassende Handbuch

Datenbank

Firebase (google.com)

Hiermit erkläre ich, dass ich meine App selbständig programmiert habe und keine
anderen als die angegebenen Hilfsmittel benutzt habe.



München, 12.10.22

_____ _____

Ort, Datum Unterschrift