Elektronik Grabner

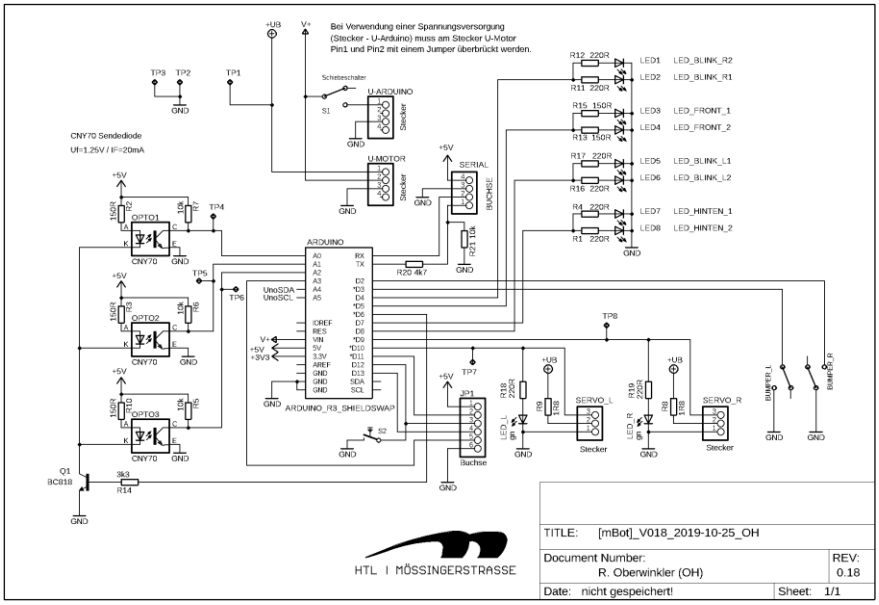
Inhalt

1.1 Plannungsphase

1.1.1 Idee

Der mBot ist ein „Selbst fahrender“ Roboter. Er kann Hindernisse durch zwei Bumper „erkennen“ und dann um sie herumfahren. Durch die Lichtschrankensensoren kann er auch Linien Folgen. Das Ganze wird von einem Arduino gesteuert und von 2 AA-Batterien betrieben.

1.1.2 Schaltplan



1,1.3 Partlist

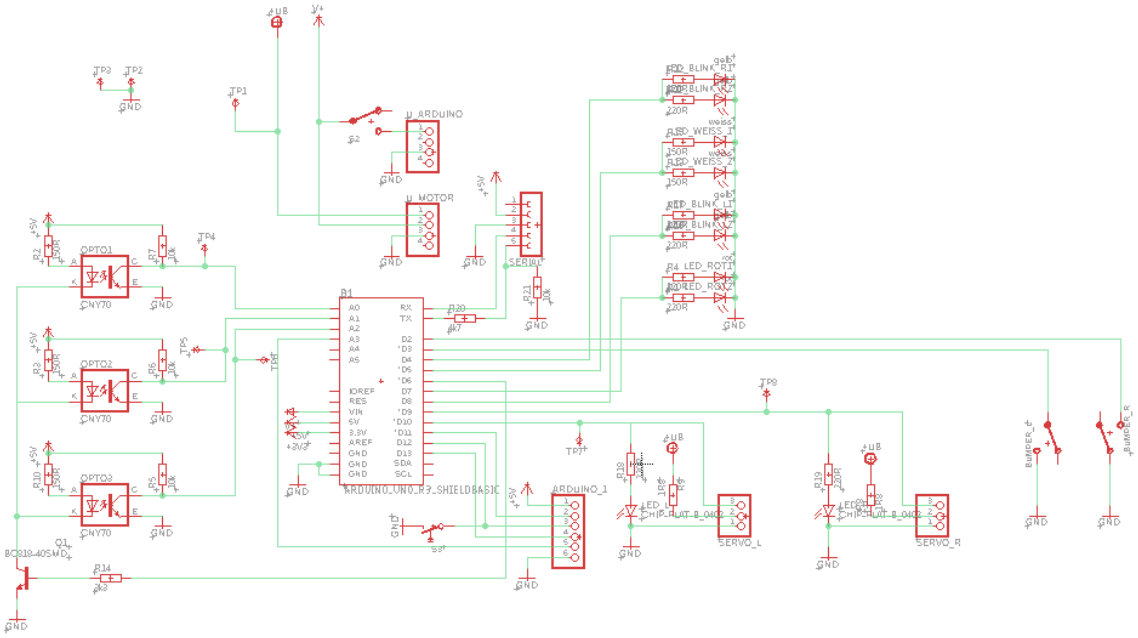
Pos	Anz	Part	Value	Lagerort	Package	Library
1	5	R2 R3 R10 R13 R15	150R	S16-A-1B	R1206	rcl
2	8	R1 R4 R11 R12 R16 R17 R18 R19	220R	S16-A-1G	R1206	rcl
3	5	R14, Servohack	3k3	S16-A-1J	R1206	rcl
4	4	R5 R6 R7 R21	10k	S16-A-3D	R1206	rcl
5	2	R8 R9	1R8	012W(OH)	R1206	rcl
6	1	R20	4k7	S16-A-2D	R1206	rcl
7	4	LED_BLINK_L1,L2,R1,R2	gelb	S10-A-5B	LED5MM-LGD	led
8	2	LED_ROT1,ROT2	rot	S10-A-5A	LED5MM-LGD	led
9	2	LED_WEISS_1,WEISS_2	weiss	S10-A-6A	LED5MM-LGD	led
10	2	LED_L, LED_R	rt od. gn	S10-G-1B	CHIPLED_1206	led
11	3	SERIAL, Batterieclip	Buchsenl. 4pol		1X04	pinhead
12	2	U_ MOTOR, U_Arduino	Stiftl. 4pol		1X 40	pinhead
13	2	SERVO_L, SERVO_R	Stiftl. 3pol			pinhead
14	1	Arduino	Stiftl. 6/8/8/10			SparkFun-Boards
15	2	Batterieclips		S10-L-2C		
16	1	Batteriehalter 4xAA				
17	1	S1	MFP113D		MFP113-KNITTER	Schalter_V12
18	1	S2			MINITAST2	Schalter_V12
19	1	Bumper_R			Mikroschalter_H_R	Schalter_V12
20	1	Bumper_L			Mikroschalter_H_L	Schalter_V12
21	1	Q1	BC818-40SMD	S10-B-4D	SOT23-BEC	transistor-npn
22	3	Opto1, Opto2, Opto3	CNY70	S10-B-6C	4	cny70
23	8	TP1 bis TP8			B2,54	testpad

pos	Anz	Schrauben
24	6	M2x12
25	4	M2x8
26	10	Mutter M2
27	2	Scheibe M2

pos	Anz	Kunststoffteile
28	2	Servo
29	2	Servohalter
30	1	Bumperset
31	1	Kugelknopf
32	2	Räder

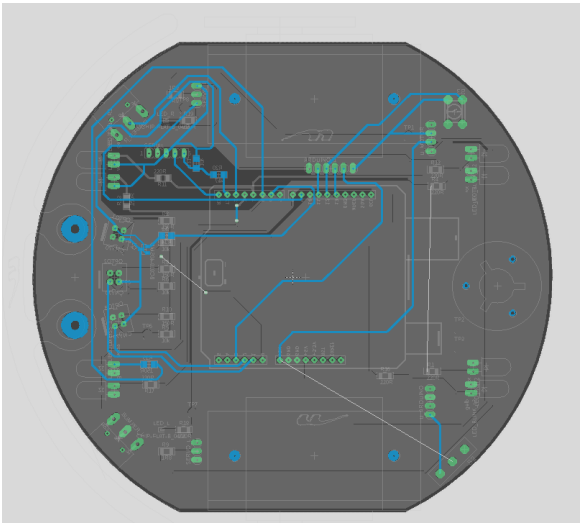
1.2 Realisierung

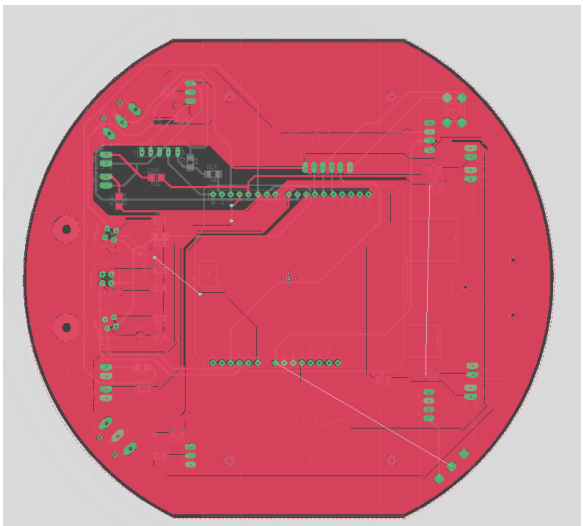
1.2.1 Schematic



1.2.2 Board-Layout

Hier sollte eine Rundung gemacht werden.
Rechteck von X/Y -1300/-2700mil bis 1300/2700mil
mit gedrückter <CTRL> Taste die linke Seite bis -3000/0, die rechte bis 3000/0 ziehen (Bogen)
Bei der Platzierung war die Position weniger kritische Bauteile schon gegeben.
Es sollte ein GND-Polygon auf die Fläche angewendet werden.
Auf Top-Bottom Seite achten, keine Drahtbrücken verwenden, nicht die Lichtschranken verdecken





1.2.3 Leiterplatte

Die Leiterplatte wurde nicht in der Hauseigenen Leiterplattenfertigung gemacht,
sondern über eine Professionelle Leiterplatten Manufaktur bestellt.

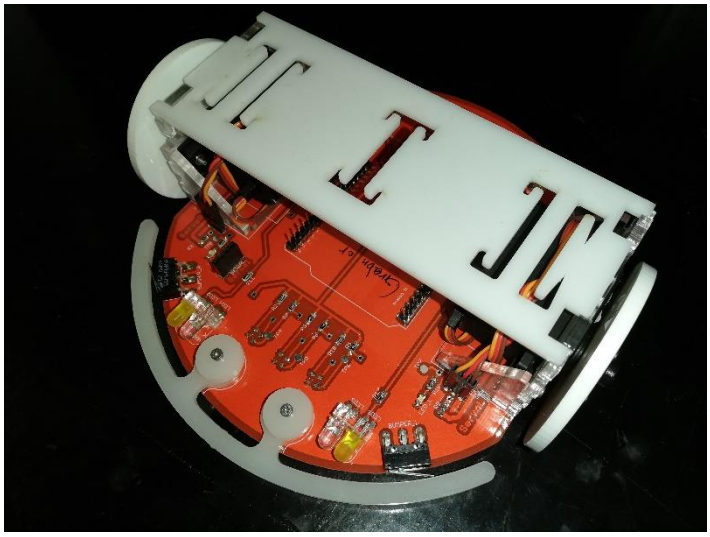
1.2.4 Bauteile

Die elektronischen Komponenten wurden Online-Bestellt,
die Bumper und andere Teile wurden im Laser Cutter ausgeschnitten.

1.2.5 Kostenaufstellung

1.2.5 Bestückung und Zusammenbau

- Die Platine wurde auf Fehler überprüft. Da die Platine bestellt wurde war das
- Zuschneiden der Rundung schon erledigt. Auch die Bohrungen waren schon perfekt.
- Zuerst wurden die SMD-Bauteile aufgelötet, hier muss man auf den Transistor auf der Rückseite achten.
- Dann wurden Steckerleisten, LED‘s und Lichtschrankensensoren (Polung beachten) verlötet.
- Die Plastik Motoren Halterung wurde mit Superkleber zusammengeklebt und an der Platine festgeschraubt.
- Die Motoren wurden dann in die Halterung eingeschraubt (2 Muttern verwendet) und mit Rädern versehen.
- Das Power-Kabel wurde an die vorgesehene Steckerleiste gelötet und mit einen Schrumpfschlauch versteckt.
- Zuletzt wurde eine Plastikbrücke angebracht, auf der sich der Batterieclip befindet



1.3 Testen

Der mBot wurde mit einen Linienfollower Arduino Test Programm getestet.
Der mBot konnte Linienfolgen (Lichtschranken) und Hindernisse Ausweichen (Bumper)

```
void setup()
{
  pinMode(L_front, OUTPUT);
  pinMode(L_brems, OUTPUT);
  pinMode(L_links, OUTPUT);
  pinMode(L_rechts, OUTPUT);
  pinMode(S_links_pin, OUTPUT);
  pinMode(S_rechts_pin, OUTPUT);

  pinMode(B_links, INPUT_PULLUP );
  pinMode(B_rechts, INPUT_PULLUP );

  pinMode(LF_enable, OUTPUT);
  pinMode(LF_links, INPUT);
  pinMode(LF_mitte, INPUT);
  pinMode(LF_rechts, INPUT);
  digitalWrite(LF_enable, HIGH);

  Serial.begin(9600);
}
```

```
////////////////// HARDWARE //////////////////////////////
// Pin-Definitionen fuer LEDs
const int L_front = 5;
const int L_brems = 7;
const int L_links = 8;
const int L_rechts = 4;

// Pin-Definitionen fuer Servos und Bumper
const int B_links = 3;
const int B_rechts = 2;
const int S_links_pin = 10;
const int S_rechts_pin = 9;

// Pin-Definitionen fuer Linienfolger
const int LF_enable = 6;
const int LF_links = A0;
const int LF_mitte = A1;
const int LF_rechts = A2;

////////////////////////////////////////////////////////////
```

```
switch (richtung)
{
  case 0: // stop
    if (S_links.attached()) S_links.detach();
    if (S_rechts.attached()) S_rechts.detach();
    digitalWrite(L_front, LOW);
    digitalWrite(L_brems, HIGH);
    digitalWrite(L_links, LOW);
    digitalWrite(L_rechts, LOW);
    break;
  case 1: // vorwärts
    S_links.write(180);
    S_rechts.write(0);
    digitalWrite(L_front, HIGH);
    digitalWrite(L_brems, LOW);
    digitalWrite(L_links, LOW);
    digitalWrite(L_rechts, LOW);
    break;
  case 2: // rückwärts
    S_links.write(0);
    S_rechts.write(180);
    digitalWrite(L_front, HIGH);
    digitalWrite(L_brems, HIGH);
    digitalWrite(L_links, LOW);
    digitalWrite(L_rechts, LOW);
    break;
  case 3: // rechts
    S_links.write(180);
    if (S_rechts.attached()) S_rechts.detach();
    digitalWrite(L_front, HIGH);
    digitalWrite(L_brems, HIGH);
    digitalWrite(L_links, LOW);
    digitalWrite(L_rechts, HIGH);
    break;
  case 4: // links
    if (S_links.attached()) S_links.detach();
    S_rechts.write(0);
    digitalWrite(L_front, HIGH);
    digitalWrite(L_brems, HIGH);
    digitalWrite(L_links, HIGH);
    digitalWrite(L_rechts, LOW);
    break;
  }
}
```

```
void loop()
{
  int links = analogRead(LF_links);
  int mitte = analogRead(LF_mitte);
  int rechts = analogRead(LF_rechts);

  Serial.print("Links: ");
  Serial.print(links);
  Serial.print(" Mitte: ");
  Serial.print(mitte);
  Serial.print(" Rechts: ");
  Serial.println(rechts);

  if (mitte > 1000)
  {
    // auf Spur
    fahren(1);
  }
  else
  {
    if (links > 1000)
    { // Rechts von der Spur -> nach Links fahren
      fahren(4);
    }
    if (rechts > 1000)
    { // Links von der Spur -> nach Rechts fahren
      fahren(3);
    }
  }
}
```

2.0 Erweiterung mBot

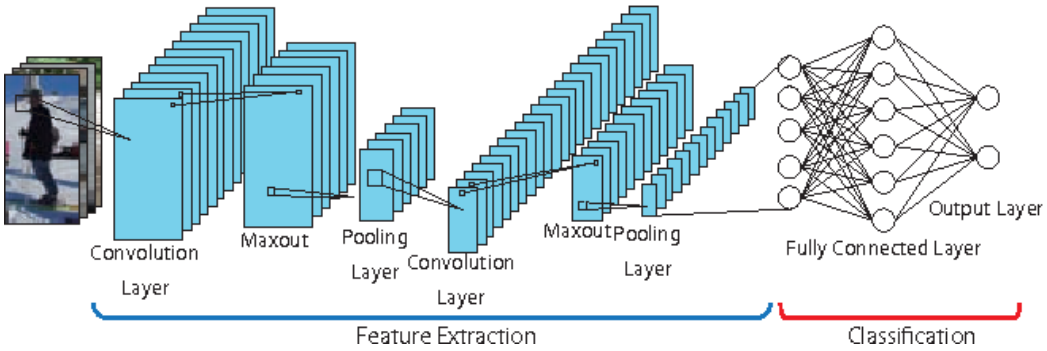
2.1 Idee

Der mBot ist kein wirkliches selbstfahrendes Auto. Er kann Linien nur Folgen, wenn sie durchgehend und direkt unter dem Fahrzeug sind.
Auch das er Hindernissen ausweichen kann ist eher trivial gelöst.
Er fährt erst gegen das Objekt und korrigiert dann erst. Die Idee der Erweiterung ist das der mBot schon im Vorhinein Hindernisse erkennen kann und diesen dann ausweicht. Auch die Linefollowing Funktion soll Implemtiert werden. Der Input für beide Funktionen kommt von einer Kamera.

2.2 Code

5.2.1 Erklärung Idee:

Das Ganze soll von einen Künstlichen Neuronalen Netzwerk gesteuert werden.
Genauer gesagt von einen CNN welches Trainingsdaten nimmt diese evaluiert Details findet, die in jedem Bild vorhanden sind und diese in neuen Bildern findet. Das Ganze wird auf der Basis von Keras Tensorflow gebaut.



2.2.2 Input Algorithmus

Da die Bilder in Unterschiedlichen Größen kommen (Verschiedene Datasets), müssen wir einen Algorithmus Implementieren der alle Bilder zu einer Größe Hinunterschneidet. Wenn man den Input nicht auf eine Größe Hinunterschneidet, kann das CNN diese nicht übernehmen (Interne Fehler in der Funktion).

2.2.2.1 Externes Training-Set:

2.2.2.2 Input Algorithmus Teil.1

Die Bilder werden geladen und auf eine gewisse Größe (px) geschnitten.
Danach werden die Bilder mit den neuen zugeschnittenen Bildern überschrieben
und ein Blocked Array (X=Bilder, Y=Beschriftung) und ein Free Array mit der
Länge der beiden erstellt.

```
blocked_X = [f for f in os.listdir("DataSet/Blocked") if os.path.isfile(os.path.join("DataSet/Blocked", f))]
print("Working with {0} images".format(len(blocked_X)))
blocked_Y = len(blocked_X)

free_X = [f for f in os.listdir("DataSet/Free") if os.path.isfile(os.path.join("DataSet/Free", f))]
print("Working with {0} images".format(len(free_X)))
free_Y = len(free_X)
```

2.2.2.2 Input Algorithmus Teil.2

Die zugeschnittenen Bilder werden eingelesen und in Farbwerte umgewandelt.
Danach werden sie in den zuvor erstellten Arrays gespeichert.
Hinter den ‚#' findet man Test Funktionen die das Array oder die Bilder
mit ihrer Überschrift ausgeben.
Zuletzt wird noch die Anzahl an Blocked und Free Images angezeigt.

```
for x in range (0,(len(blocked_X))):
    blocked_Y = [0] * len(blocked_X)
    name = blocked_X[x]
    image = Image.open("DataSet/Blocked" + "/" + name)
    #display(_Imgdis(filename= "DataSet/Blocked" + "/" + blocked_X[x], width=240, height=320))
    blocked_X[x] = asarray(image)
    #print(blocked_X[x].shape)
    #print(blocked_Y[x])


for x in range (0,(len(free_X))):
    free_Y = [1] * len(free_X)
    name = free_X[x]
    image = Image.open("DataSet/free" + "/" + name)
    #display(_Imgdis(filename= "DataSet/free" + "/" + free_X[x], width=240, height=320))
    free_X[x] = asarray(image)
    #print(blocked_X[x].shape)
    #print(free_Y[x])

dataset_X = np.concatenate([blocked_X, free_X])
dataset_Y = np.concatenate([blocked_Y, free_Y])
```

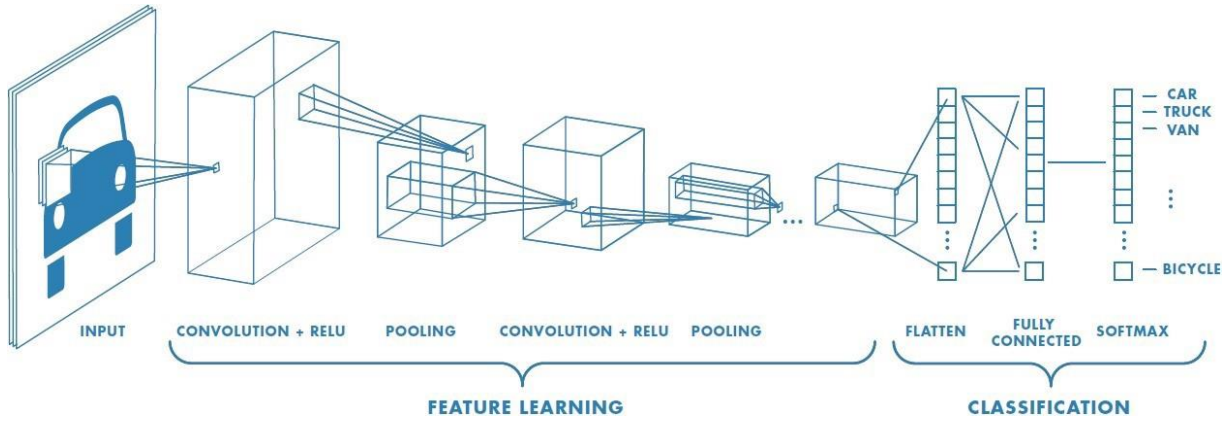
```
0
(224, 224, 3)
0
(224, 224, 3)
0
(224, 224, 3)
1
(224, 224, 3)
1
(224, 224, 3)
1
```

```
dataset_X[1]

array([[[115,  91, 107],
        [123,  99, 115],
        [123,  96, 113],
        ...,
        [136, 116, 144],
        [138, 118, 146],
        [144, 125, 155]],

       [[119,  95, 111],
        [127, 100, 117],
        [127, 100, 115],
        ...,
        [139, 119, 147],
        [168, 148, 176],
        [146, 125, 156]],

       [[122,  95, 110],
        [128, 101, 116],
        [133, 106, 121],
        ...,
        [168, 148, 176],
        [147, 127, 155],
        [121, 101, 129]],

       ...,
```

```
(224, 224, 3)
0
```

2.2.3 Training-Code

Der Haupt-Code, hier werden alle Bilder verarbeitet.
Im model.fit Teil wird das Neural Network mit 32 Bilder pro Input
auf 200 Epochen trainiert. Der erste große Teil sind Layers.
Diese helfen die accuracy zu steigern. Am Ende wird das ganze nochmal
Compiled und zusammengefasst.




```
from keras.models import Sequential
from keras.layers import Dense, Flatten,Conv2D , MaxPooling2D, Dropout
import tensorflow as tf

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(224, 224, 3), activation="relu", padding="same"))
model.add(Conv2D(32, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(Conv2D(64, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(Conv2D(64, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(Conv2D(128, kernel_size=(3, 3), activation="relu", padding="same"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(32, activation="relu"))
model.add(Dense(16, activation="relu"))
model.add(Dense( 2, activation="softmax"))

model.compile(optimizer='RMSProp', loss="sparse_categorical_crossentropy", metrics=['accuracy'])
model.summary()

model.fit(dataset_X, dataset_Y,batch_size=32, epochs=200, shuffle = True )
```

```
Epoch 1/200
7/7 [==============================] - 16s 2s/step - loss: 10.8233 - accuracy: 0.5101
Epoch 2/200
7/7 [==============================] - 16s 2s/step - loss: 0.7054 - accuracy: 0.4091
Epoch 3/200
7/7 [==============================] - 19s 3s/step - loss: 0.6933 - accuracy: 0.5051
Epoch 4/200
7/7 [==============================] - 18s 3s/step - loss: 0.6964 - accuracy: 0.5354
Epoch 5/200
7/7 [==============================] - 19s 3s/step - loss: 0.6907 - accuracy: 0.5303
```

```
Epoch 49/50
7/7 [==============================] - ETA: 0s - loss: 0.0505 - accuracy: 0.9798
Epoch 00049: saving model to cp.ckpt
7/7 [==============================] - 17s 2s/step - loss: 0.0505 - accuracy: 0.9798
Epoch 50/50
7/7 [==============================] - ETA: 0s - loss: 0.1685 - accuracy: 0.9747
Epoch 00050: saving model to cp.ckpt
7/7 [==============================] - 20s 3s/step - loss: 0.1685 - accuracy: 0.9747
<tensorflow.python.keras.callbacks.History at 0x27aad7a5820>
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 224, 224, 32)      896
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 224, 224, 32)      9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 112, 112, 32)      0
_________________________________________________________________
dropout_4 (Dropout)          (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 112, 112, 64)      18496
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 112, 112, 64)      36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 56, 56, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 56, 56, 64)        36928
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 56, 56, 64)        36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 28, 28, 128)       73856
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 28, 28, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               6422784
_________________________________________________________________
dense_7 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_9 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_10 (Dense)             (None, 16)                528
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 34
=================================================================
Total params: 6,827,442
Trainable params: 6,827,442
Non-trainable params: 0
```

2.2.4 Evaluation, Testen, Speichern

Zuletzt wird das Model nochmal Compiled und Evaluiert. Dann kann man dem Neural Network ein Bild geben welches es dann Predicted (Free/Blocked). Am Ende speichert man das System als HDF5 gespeichert.

```
model.compile(optimizer='RMSProp', loss="sparse_categorical_crossentropy", metrics=['accuracy'])
model.summary()

checkpoint_path = "cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1)

model.fit(dataset_X, dataset_Y,batch_size=32, epochs=200, shuffle = True, callbacks=[cp_callback])
```

```
model.evaluate(dataset_X, dataset_Y)
results = model.predict(dataset_X[40].reshape(-1, 224, 224, 3))
print(results)

dataset_Y
dataset_X
```

2.3 Hardware

5.3.3 CPU

Durch die geringe „CPU-Leistung“ des Arduinos müsste man auf einen Raspberry PI 4 umsteigen. Die Implementierung der Live Kamera wäre dadurch auch viel einfacher.

2.3.4 Kamera

Die PI-Cam wurde ausgesucht für das Live Video.
Sie kann jedoch auch durch die Wireless Funktion sehr gut zum Trainingsdaten sammeln eingesetzt werden.

Cam und Controll Code:

```
#from picamera import PiCamera
from keras.models import load_model
import numpy as np
import random
import numpy as np
import os, sys
from IPython.display import display
from IPython.display import Image as _Imgdis
from PIL import Image
from numpy import asarray
from matplotlib import image
from matplotlib import pyplot

#import RPi.GPIO as GPIO

M1 = 23
M2 = 19
#GPIO.setmode(GPIO.BOARD)

#GPIO.setup(M1, GPIO.OUT)
#GPIO.setup(M2, GPIO.OUT)

model = load_model('mBot_KI.hdf5')

i = 0


while i != 1:


    #camera.start_preview()
    #camera.capture('/home/pi/Desktop/image%s.jpg')
    #camera.stop_preview()

    #image = Image.open('/home/pi/Desktop/image%s.jpg')
    image = Image.open("DataSet/Blocked/05199562-694f-11e9-86df-72b5f773b75d.jpg")
    image_data = asarray(image)
    results = model.predict(image_data.reshape(-1, 224, 224, 3))
    index_max = np.argmax(results)
    print(index_max)

    n = random.randint(1,2)


    if((index_max == 0) and (n == 1)):
        #Drive Left
        #GPIO.output(M1, False)
        #GPIO.output(M2, True)
        print("left")


    if((index_max == 0) and (n == 2)):
        #Drive Left
        #GPIO.output(M1, True)
        #GPIO.output(M2, False)
        print("right")

    if(index_max == 1):
        #Drive Straight
        #GPIO.output(M1, True)
        #GPIO.output(M2, True)
        print("straight")
```