



WRO2021 Elementary Open Category

**NeuroPower: wireless charging system of electric vehicles**

Name: Aruzhan Kuanyshbai, Yelzhan Tama

Almaty 2021

Content

[Introduction 2](#_Toc83743408)

[Solution 3](#_Toc83743409)

[Code 4](#_Toc83743410)

[Conclusion 9](#_Toc83743411)

# Introduction

Nowadays, one of the global problems is CO2 emissions, and therefore a person is looking for different solutions, one of which is the use of electric cars. However, not every city has a charging infrastructure. For example:

In Britain, transport is considered the main source of CO2 emissions, the government is going to ban sales of new cars with gasoline and diesel engines from 2030, but the demand for electric vehicles is limited by a lack of charging infrastructure.



Unlike conventional cars, it takes 5 minutes to fill gasoline, electric cars take a really long time to charge, the same famous Tesla charges in as much as 24 hours. BUT! Thanks to our system, we can safely charge cars on the go and there will be no need for these 24 hours. This is really amazing!

To encourage people to protect our land and move to green energy, we offer the NeuroPower system that can charge electric cars remotely.



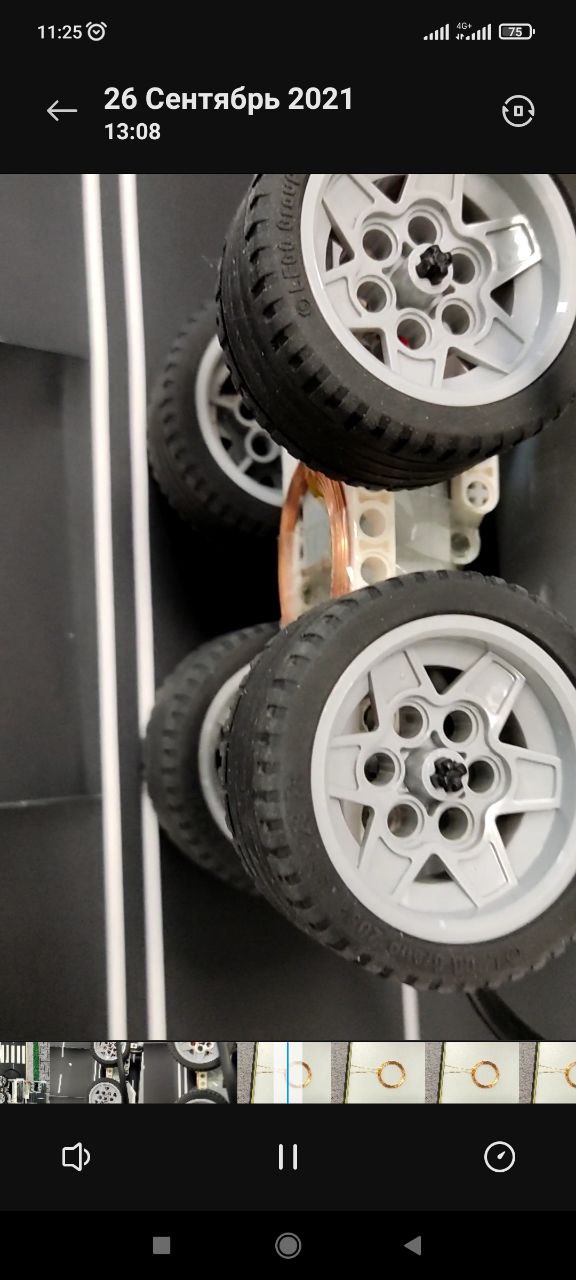
# Solution

NeuroPower is a system of autonomous and automatic charging of cars on the go, which is a system consisting of 2 components:

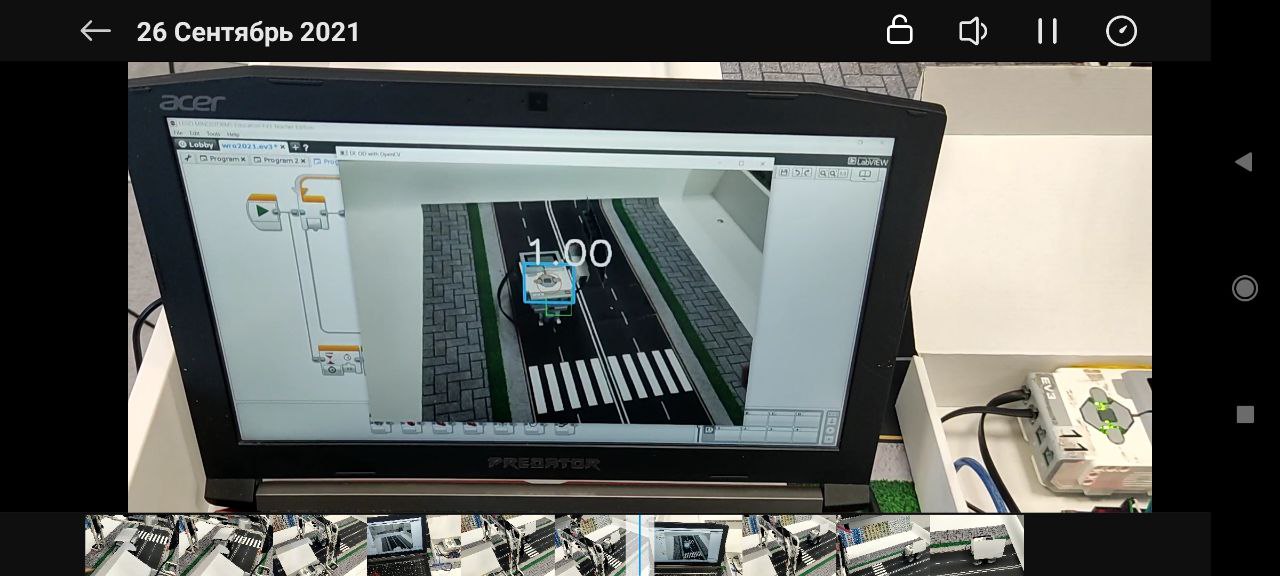
1. Camera that monitors electric cars, namely the roof of electric cars, on which solar panels will be located. Thanks to this, the lasers will be easily guided on the panels and thereby charge our electric cars on the move.
2. Inductors that will charge electric cars using electromagnetic induction.

Let's take a look at the mechanism of our project:

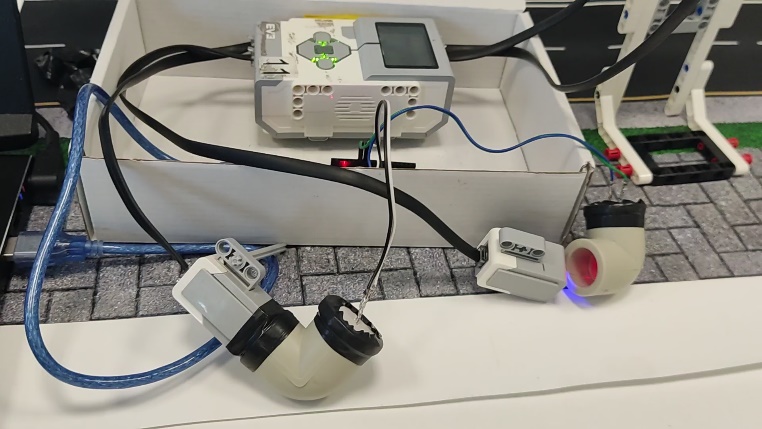
1. Firstly, these are inductors, that is, by placing the transmitters on the road and the receivers on the bottom of the car, they will transmit electricity using electromagnetic induction. Thus, we provide wireless charging of cars from the road. We simply supply electricity to 1 coil, and then it transfers electricity to the second.



1. Secondly, this is a laser camera that will monitor the roofs of electric cars. And then the laser will charge our electric cars from the air. In the project, we used the yolov4 convolutional neural network. We trained our neural network by making a dataset of 1000 photos and then ran the code to train the neural network. Then we got weights. These weights are the parameters for the neural network that are needed for object recognition. Our python code takes a frame from a video stream and then finds our object on it. Then it returns the coordinates of the object. Using these coordinates, we aim the camera so that the object is always in the center of the camera.

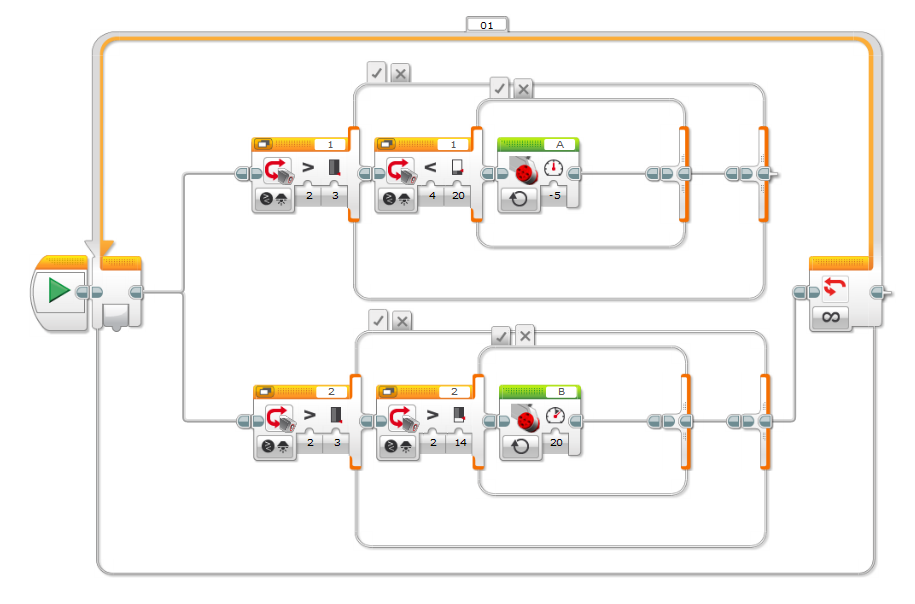


The connection between the laptop and the ev3 occurs through the Arduino microcontroller since the ev3 cannot process information from the COM port without additional libraries. By sending a signal to the Arduino, it transmits information about the rotation of the ev3 camera using LEDs. By changing the brightness of the LEDs, we can understand what ev3 needs to be done.



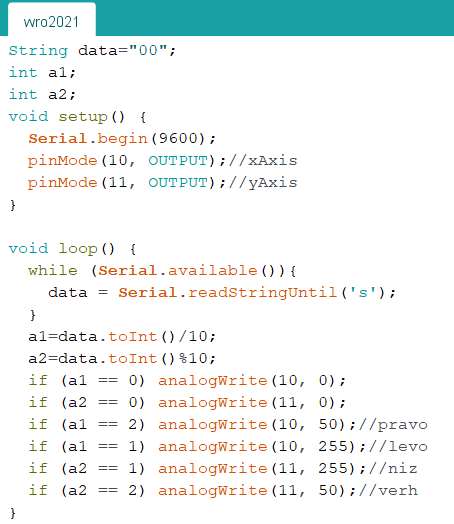
# Code

**EV3 loop code:**



Here we have infinite loop in which we continuously compare the brightness of two color sensors and control motors correspondingly

**Arduino code:**



Here we have continuously loop in which we receive data by the COM port from the laptop and change the brightness of LED correspondingly

**Python code:**

import cv2 as cv  
import numpy as np  
import serial  
import time  
ArduinoUnoSerial = serial.Serial('com8',9600)  
#Write down conf, nms thresholds,inp width/height  
confThreshold = 0.3  
nmsThreshold = 0.3  
inpWidth = 640  
inpHeight = 640  
a = ""  
a1 = ""  
a2 = ""  
  
#Load names of classes and turn that into a list  
classesFile = "coco.names"  
classes = None  
  
with open(classesFile,'rt') as f:  
 classes = f.read().rstrip('\n').split('\n')  
  
#Model configuration  
modelConf = 'Tiny/yolov4-tiny-custom.cfg'  
modelWeights = 'Tiny/yolov4-tiny-custom\_best.weights'  
  
def postprocess(frame, outs, a):  
 frameHeight = frame.shape[0]  
 frameWidth = frame.shape[1]  
  
 classIDs = []  
 confidences = []  
 boxes = []  
   
  
 for out in outs:  
 for detection in out:  
   
 scores = detection [5:]  
 classID = np.argmax(scores)  
 confidence = scores[classID]  
  
 if confidence > confThreshold:  
 centerX = int(detection[0] \* frameWidth)  
 centerY = int(detection[1] \* frameHeight)  
  
 width = int(detection[2]\* frameWidth)  
 height = int(detection[3]\*frameHeight )  
  
 left = int(centerX - width/2)  
 top = int(centerY - height/2)  
  
 classIDs.append(classID)  
 confidences.append(float(confidence))  
 boxes.append([left, top, width, height])  
 # cv.rectangle(frame, (220, frame.shape[0] - 150), (frame.shape[1] - 220, 150), (0, 255, 0), 1) left, bot, right, top  
  
 if centerX > 300:  
 a1="1"  
 if centerX < (frame.shape[1] - 300):  
 a1="2"  
 if centerY < 260:  
 a2="1"  
 if centerY > frame.shape[0]-260:  
 a2="2"  
 if 290<=centerX<=(frame.shape[1] - 290):  
 a1 = "0"  
 if 270>=centerY>=frame.shape[0]-270:  
 a2="0"  
 a=a1+a2+"s"  
 cv.circle(frame, (centerX, centerY), 20, (0, 0, 255), 1)  
  
 indices = cv.dnn.NMSBoxes (boxes,confidences, confThreshold, nmsThreshold )  
  
 indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)  
 for i in indices:  
 i = i[0]  
 box = boxes[i]  
 left = box[0]  
 top = box[1]  
 width = box[2]  
 height = box[3]  
   
 drawPred(classIDs[i], confidences[i], left, top, left + width, top + height)  
 ArduinoUnoSerial.write(str.encode(a))  
 print(a)  
  
  
def drawPred(classId, conf, left, top, right, bottom):  
 # Draw a bounding box.  
 cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)  
  
 label = '%.2f' % conf  
 cv.putText(frame, label, (left,top), cv.FONT\_HERSHEY\_SIMPLEX, 2, (255, 255, 255), 3)  
  
def getOutputsNames(net):  
 # Get the names of all the layers in the network  
 layersNames = net.getLayerNames()  
   
 # Get the names of the output layers, i.e. the layers with unconnected outputs  
 return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]  
  
  
#Set up the net  
  
net = cv.dnn.readNetFromDarknet(modelConf, modelWeights)  
net.setPreferableBackend(cv.dnn.DNN\_BACKEND\_OPENCV)  
net.setPreferableTarget(cv.dnn.DNN\_TARGET\_CPU)  
  
  
#Process inputs  
winName = 'DL OD with OpenCV'  
cv.namedWindow(winName, cv.WINDOW\_NORMAL)  
  
cap = cv.VideoCapture(2)  
  
while cv.waitKey(1) < 0:  
 #get frame from video  
 hasFrame, frame = cap.read()  
 a="00s"  
 #Create a 4D blob from a frame  
  
 blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop = False)  
  
 #Set the input to the net  
 net.setInput(blob)  
 outs = net.forward (getOutputsNames(net))  
 cv.rectangle(frame, (300, frame.shape[0]-260), (frame.shape[1]-300, 260), (0, 255, 0), 1) #left, bot, right, top  
 postprocess (frame, outs, a)  
  
 #show the image  
 cv.imshow(winName, frame)

Here we just take the frame of video stream and set it to the net. All of the steps are written in the comments of the code (#parts). We use opencv2 library to process the images.

We trained our neural network yolov4 using guides on the Internet by uploading the dataset with pointed data on the training program. Then we got weights file – parameters of neural network, like A+B\*x^2+…+C\*x^999. Where A B C are parameters. So, we used it and config file of yolov4, where layers of neural network are described.

# Conclusion

Thus, we can say that thanks to our system, we will be able to charge electric cars on the go and thus popularize them in our country, as well as around the world making it greener!

