Automated Firetruck-Robot

Reiner Katharina   
*University of Applied Sciences Hof*Hof, Germany  
katharina.reiner@hof-university.de

Schlitter Moritz  
*University of Applied Sciences Hof*Hof, Germany  
moritz.schlitter@hof-university.de Wilfert Luca  
*University of Applied Sciences Hof*Hof, Germany  
luca.wilfert@hof-university.de

Manig Ralf  
*University of Applied Sciences Hof*Hof, Germany  
ralf.manig@hof-university.de Raithel Tim  
*University of Applied Sciences Hof*Hof, Germany  
tim.raithel@hof-university.de

*Abstract*—As part of the lecture “Applied Robotics” at the university of applied science Hof, we will build an intelligent firefighting robot. The robot is supposed to drive autonomously to the destination, there it should recognize a picture of a flame, then put out that flame with water and lastly return to the fire station. This functionality is implemented based on the chassis of an RC car with a single board computer that uses the input of a camera to navigate. The car contains a water tank with a pump to spray the water on the flame picture. Additionally, the vehicle has working blue light and can play a siren sound.

Keywords— intelligent, Roboter, Löschen, Feuerwehr

# STATE OF THE ART

## Einleitung

TODO

## What already has been done regarding the problem

There already were multiple projects done at the university of applied science Hof, in which the same type of electric vehicle was used. The practical work „Aufbau und Implementierung eines Elektrofahrzeugs mit REST API zur Steuerung und Kamerabildübertragung”, written by Daniel Hanik, on which this project is based on, describes steps to build up the car. Amongst other things, the paper includes the structure and configuration of all the hardware components, the installation of the operating system and the software components and the Python code, that controls the servomotors. The steps of these paragraphs can be adapted to fit this project.

# Approach

## Goal

The goal of this project is to build an intelligent firefighting robot, based on the vehicle, camera and Nvidia Jetson Xavier NX, provided by the university. The robot is supposed to look like a firefighter truck with working flashing blue light and siren sound. When the user manually activates an alarm, the vehicle autonomously follows a yellow line on the flour, by evaluating the camera input and controlling the servomotors accordingly. During that time two blue LEDs are flashing alternately and the siren sound file is played via a small Bluetooth speaker. Concurrently an AI application for image recognition monitors the camera input, looking for a previously determined symbol of a flame and a fire station. When the AI application recognizes symbol of the flame, the vehicle stops and begins to put out the flame. It does that, by activating a small water pump inside a water tank within the body of the car for a few seconds. The water will spray out of a jet pipe mounted on the roof of the truck. After the vehicle finished this job, it will switch off the blue light and the siren and turn around. It then autonomously follows the yellow line back, until the AI application recognizes the fire station symbol. Here the vehicle will come to a hold and the process is finished.

In case, the implementation of the goal stated above can be finished early in the time space of the project, there are some additional goals, which can be implemented afterwards. For example, the signal to start the whole process could be send autonomously by another image recognition software, that sends an alarm when recognizing a flame symbol. The user would not have to start the alarm process manually. Furthermore, the robot’s ability to navigate could be enhanced. A possibility are multiple possible destinations, that means the robot would have to decide where to go when encountering a crossing in the yellow line. Autonomous parking in a defined area or Avoiding obstacles on the yellow line are additional possibilities. On top of that the aiming with the water jet while putting out the flame can be enhanced. The vehicle would position itself autonomously, so that the water lands in the desired container. Additionally, the image recognition software can be expanded, to recognize different types of fires, and put out a message containing the correct method to extinguish this kind of flame.

## Schritte

### Preparing the Hardware

#### Installing the Operating System

TODO

#### Daten auf SSD verschieben

TODO

#### Softwarekomponenten installieren

TODO

#### Chassis

Like in the practical work referenced above, the given chassis with the pre-mounted servomotors is used. Additionally, a wooden platform was placed on four stud bolts on top of the chassis. This platform serves as a foundation for the 3D-printed body parts, which give the vehicle the look of a firefighter truck. The body is composed of three parts, that are fixated on the platform with multiple small blocks of wood, that are glued on the platform. The blocks keep the body parts from sliding horizontally, yet still the parts can be removed by lifting them up.

The front part is used as a mount for the camera, so that the camera points downwards in a 45° angle.

The middle part contains the water tank, and has lids on both sides, to grant easy access to the tank. The water tank itself is a lunch box, with the water pump inside. There are two holes in the tank, to allow the cable and hose from the pump to run to the outside. From there the hose runs through the roof of the middle part, inside a 3D-printed jet pipe. In the front of the hose is a small 3D-printed outlet, which reduces the diameter of the hose. This allows greater range of the water jet.

The back part contains electronic devices like the Jetson, the voltage converter and the Bluetooth speaker. It is separated from the middle part, to prevent water from leaking near the electronics. A Trunk lid allows easy access to the electronic parts.

#### Connecting electronics

Similar as in the previously mentioned practical work, the Nvidia Jetson Xavier NX, as well as the servomotors are powered by the given 4200mAh battery pack. Since the Jetson requires an input voltage of 18-19 volts, a voltage converter was used as described in the paper. The ground (GND), power supply (VCC) and signal cables of the servomotor used for steering, are connected to the corresponding pins of row 2 on a servo driver. While the cables of the electronic speed controller similarly are connected to the first row. The servo driver itself is connected to the Jetson as described in Daniel Haniks’s paper. The ground pin is connected to any ground pin of the Jetson, the SCL and SDA pins are connected to the corresponding pins of the I2C1 port (Pin 5 and 3) on the Jetson and VCC is connected to one of the two 3.3V power supply pins.

The camera and the water pump are both connected via USB. It is important here, that the pump is connected to one of the two outputs of USB hub closer to the power supply (Port 1 or 2), while the camera is connected to either USB port 3 or 4. This is because in order to switch off the pump, the whole USB hub is cut from power and this would affect the camera as well if connected to this hub.

The Jetson is capable of Bluetooth connection, which is used to communicate with the small speaker, which plays the siren sound.

The two blue LEDs are each connected to a ground pin with their black cable and to Pin 10 or respectively 12 with their red cable.

### Adressing each Hardware component

#### Flashing light

To implement the flashing light of the firetruck there are two blue LEDs on top of the car body.

The LEDs have integrated resistors and are addressed by powering the I/O pins of the Jetson on and off. They are wired to two different I/O pins and a ground pin each.

The default state of both LEDs has to be off. When powered on, they flash alternately. For this exists a python script that utilizes the `RPi.GPIO` library.

#### Water pump

The fire extinguishing process is implemented by using a small USB-powered water pump that pumps water from a small tank via a hose to a nozzle. For addressing the water pump one of the USB-ports of the Jetson is used.

By default, the pump must be powered off. This is achieved by running a bash script using `uhubctl` which powers off the used USB-port before plugging in the pump.

When the Robot has to release some water, there is another script that turns the respective USB-Port on. After enough water has been pumped out the same script as mentioned in the beginning can be used again to turn the pump back off.

#### Speaker

The implementation of the siren is achieved by using a small Bluetooth speaker which plays a mp3 file of a siren.

First, the Jetson must be initially paired with the speaker via Bluetooth. This is a one-time configuration task in the beginning which has to be performed manually.

After the devices have been paired once, the connection-process can be automated via the command `bluetoothctl connect`. This process is supported by another script checking regularly if the device is still connected.

For playing the mp3-file the `playsound`-library is used in a python script.

### Camera

#### Kameraanalyse: Linie folgen

TODO

#### Kameraanalyse: Symbole erkennen

TODO

### Integration

#### Wrapping the components with shared base class

A “GlobalController” is supposed to bring together all sub-components and realize the requirements.

The functionalities to control each hardware component are unified as follows. The controlling elements inherit from a shared base class “Component”. All of these require to be start- and stoppable on demand. When the car starts to drive to a fire destination, flashing light and sirens need to be activated. As soon as the destination is reached, they need to be stopped.

The base class “Component” encapsulates the usage of a thread including its start and stop process. This enables the GlobalController to manage the standardized interface according to the application.

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

The previously implemented operations to address the hardware are each integrated into a wrapping class. The logic can be realized in a function that later can be used as the thread’s target. Additional clean up code before stopping the thread can be linked into the stop-process. In this way blinking of the LED ends with the output of a LOW-signal before actually killing the thread.

#### Identifying the destinations

Ids are used to Identify which destination the firetruck should drive to. To distinguish between driving to a fire and the drive back the home-id is statically set to 0. Based on this information the controller can decide whether to use privileges on a ride or not. This attribute is defined as follows:

Is the destination id the id of a fire location, flashing light and sirens must be set to active while driving. As soon as the destination is reached those components stop. In return the water pump is activated. Recognizing a successful extinguish for now is simulated by waiting three seconds. Subsequently, the pump component is stopped. Finally, the firetruck turns round and drives back to the fire station by setting the destination to the home id.

Is the destination id the home-id, neither privileges are used while driving nor the extinguishing process needs to be started on reaching the destination.

#### Communication with Control-API

Steering the car still is realized by the camera analysis. To be able to pass current observations, the controller provides a communication interface. For each possible occurring event, a dedicated method is available. These each process the related signals.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

If a direct stop or steering event is identified, the API adapter is used to send appropriate POST-requests. The camera analyst continuously sends observations. To avoid overloading the REST interface, the last messages received are cached and only new signals are forwarded to the API.

For possible future updates there also is an interface to process the recognition of intersections. Therefore, the signal is forwarded to an “IntersectionGuide” which internally determines the direction to take and returns a corresponding instruction to the controller.

If the reach of a destination is observed, the following actions are executed as described in b).

# Experimente

TODO

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

# CONCLUSION

TODO

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.