Lab Course: Hardware/Software Co-Design with a LEGO Car

Abdallah Attawia, Moritz Dötterl, Heiko, Berkay
Technische Universität München, Garching, Department of Informatics
Boltzmannstr. 3, 85748 Garching, Germany

This document explains the work done on a lego car to detect and follow a line autonomously. Connection and communication between the hardware parts is explained. Furthermore, collision avoidance and line detection algorithms are thoroughly described.

I Introduction

An autonomous car is one that can drive from point A to B independently without any human input. It can accelerate, brake and steer itself as well as sense the environment and navigate to avoid crashing. The development of autonomous vehicles is rapidly growing and a lot of big and small companies are working towards making autonomous driving a reality.
In this project a small scaled model, a lego car, along with some cheap components are used to test a real autonomous driving function, i.e. following a line on the street. This documentation starts with a brief explanation of the different hardware components. Then the cabling and communication between the components is described. Lastly, the collision avoidance technique and and the line detection and following are explained. The code, attached to this report, carries out the functions explained here and is clearly documented.

II Hardware Parts

The hardware components used are two controllers, the DEO-Nano and the Raspberry Pi, two types of actuators, namely speed motors and servo motors, as well as two types of sensors, which are the ultrasound sensors and the USB-camera.

II.A DEO-Nano

The DE0-Nano board is a compact-sized FPGA based development platform manufactured by terasIC. Its heart is a Cyclone IV FPGA from Altera. Furthermore it features on board memory, LEDs, Buttons, and most importantly three GPIO headers. On this FPGA we flashed the provided Nios-II image. This is an image of an adaptable processor allowing for custom specializations such as multiple hardware UART handlers. Now it is possible to wright code for this processor in C using the hardware you desire, without any trade off. We are using four UART units to connect to three ultrasonic sensors as well as to the Raspberry Pi. Furthermore we use two PWM modules with each two channels allowing us to generate PWM signals at four pins. Those are used to control the drive motor and the steering servo.

II.B Raspberry Pi

For this project the newly released Raspberry Pi 3 is used. Raspberry Pi is a small-sized one board computer. In this work it is used to connect the camera with the Nano-board and to run the line following code.
First, the micro SD-card for the Pi is prepared with the Raspbian Operating System installed on it. Then the Pi is boots the Raspbian and the computer is connected to the Raspberry Pi using Secure Schell (SSH) either via the Ethernet port, or using the build in Wi-Fi of the Pi 3 and the LRZ Wi-Fi network. Then the camera is connected using one of the USB ports and OpenCV is installed to run the line following code. Finally UART communication is enabled to connect the Raspberry Pi with the Nano-board. In Raspberry Pi 3 you need to disable Bluetooth in order for UART communication to work. To disable onboard Pi3 Bluetooth and restore UART over GPIOs 14 and 15, modify the file "/boot/config.txt":

- sudo nano /boot/config.txt
- Add to the end of the file:
  "dtoverlay=pi3-disable-bt"

II.C Actuators

- Speed motors
- Servo motor

II.D Sensors

- Ultrasound sensor
  The ultrasound sensor measures the distance from any detected object within its range. As shown in Figure 1 the ultrasound sensors detects objects by sending out ultrasound waves and receiving an echo, if an object is detected. The sensor measures the time of flight between transmitting and receiving the signal and calculates the distance according to the following formula (1).

$$L = \frac{340(m/s)x \triangle t(s)}{2} \qquad (1)$$

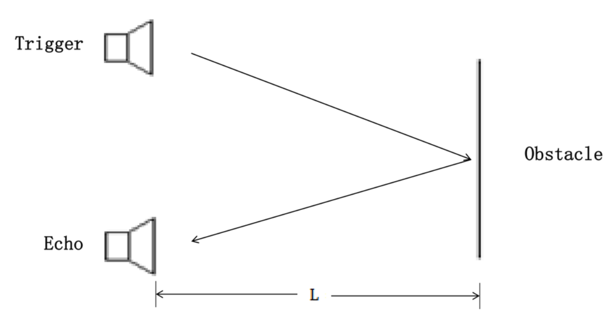

Figure 1: Theory of operation of the ultrasound sensor

In this course, the KS103 ultrasound sensor is used. It has a range up to 11m which can be specified by sending a command from the Nano-board. This sensor can operate using two digital communication protocols; I2C and Uart. It has an accuracy between $1mm$ and $10mm$

In order for the ultrasound sensor to operate, it is first initialized and set to the desired range. Like the raspberry pi, it communicates with the Nano-board using uart communication, which has to be initialized as well. The Nano-board then sends a command to the sensor to send the data using uat. The Nano-board finally receives date from the sensor. The data read from the sensor is multiplied times 170 to get the distance from detected objects in micrometer.

- USB Camera

III Cabling

IV Communication

V Collision Avoidance

In order to avoid collisions, an algorithm based on the idea of the adaptive cruise control is implemented. Adaptive cruise control is a cruise control function that allows the vehicle to adapt its speed according to the traffic and to maintain a safe distance from the vehicles ahead.
In real cars, a long range radar sensor is used to detect the cars, here we use an ultrasound sensor to measure the distance from the vehicles ahead. The following algorithm is implemented to avoid the collision with other cars or objects and to adapt the speed to the surrounding traffic.

- if distance from ultrasound sensor $\leq$ a minimum set distance ($distance_{min}$)
  -car stops

- if distance from ultrasound sensor $\geq$ a maximum set distance ($distance_{max}$)
  -drive with maximum speed

- else (distance lies within $distance_{min}$ and $distance_{max}$)
  -adapt speed according to distance:

$$CurrentSpeed = MaximumSpeed * \frac{distance}{distance_{max}} \tag{2}$$

The speed of the car is set according to the aforementioned algorithm, where the current speed is the duty cycle that is assigned to the speed motors.

VI Image processing for line detection and following

The task of the whole project was to build a Lego Car that can autonomous follow a line on the ground. For this purpose we are using an ordinary web cam on the front of the car giving us an image of the ground up to approximately 20 cm in front of the car. We developed an algorithm that detects the line in the image and calculates how the car needs to steer in order to stay on that line. The calculation is entirely done on the Raspberry Pi using the image processing framework OpenCV. The following sections explain how the image processing works and how we optimized it for the Raspberry Pi.

VI.A Code

VI.B Optimization

VII Conclusion

To conclude, the car followed a closed track without getting out of line in both directions. The maximum set speed was about 85% of the maximum possible speed. With a few modifications on the camera, e.g. a higher frame rate of 30

FPS, or a wider viewing angle, or a higher computation power on the Raspberry Pi, the full motor speed could had been reached.

Acknowledgments

References