Documentation

Idea

The idea of this project is, to build a parking aid system for a small RC car. Parking aids are advanced driver assistance systems designed to make parking easier. Such systems monitor an area of between roughly 20 and 180 cm in front of and behind the vehicle and warn the driver about any obstacles.

Furthermore, sharp changes in the acceleration which indicates emergency brakes shall be detected and shown via led indication. An additional feature is the automatic light which turns on, when it gets dark and dizzy. Also, the temperature is measured, which could be used to overwatch the battery temp of the RC car. After capturing the sensor data by our board, it gets send to a smartphone app which visualizes the data.

The distance sensors for the actual parking procedure have been implemented which ultrasonic sensors. However, the detailed hardware description can be found in the following chapters.

User Guide

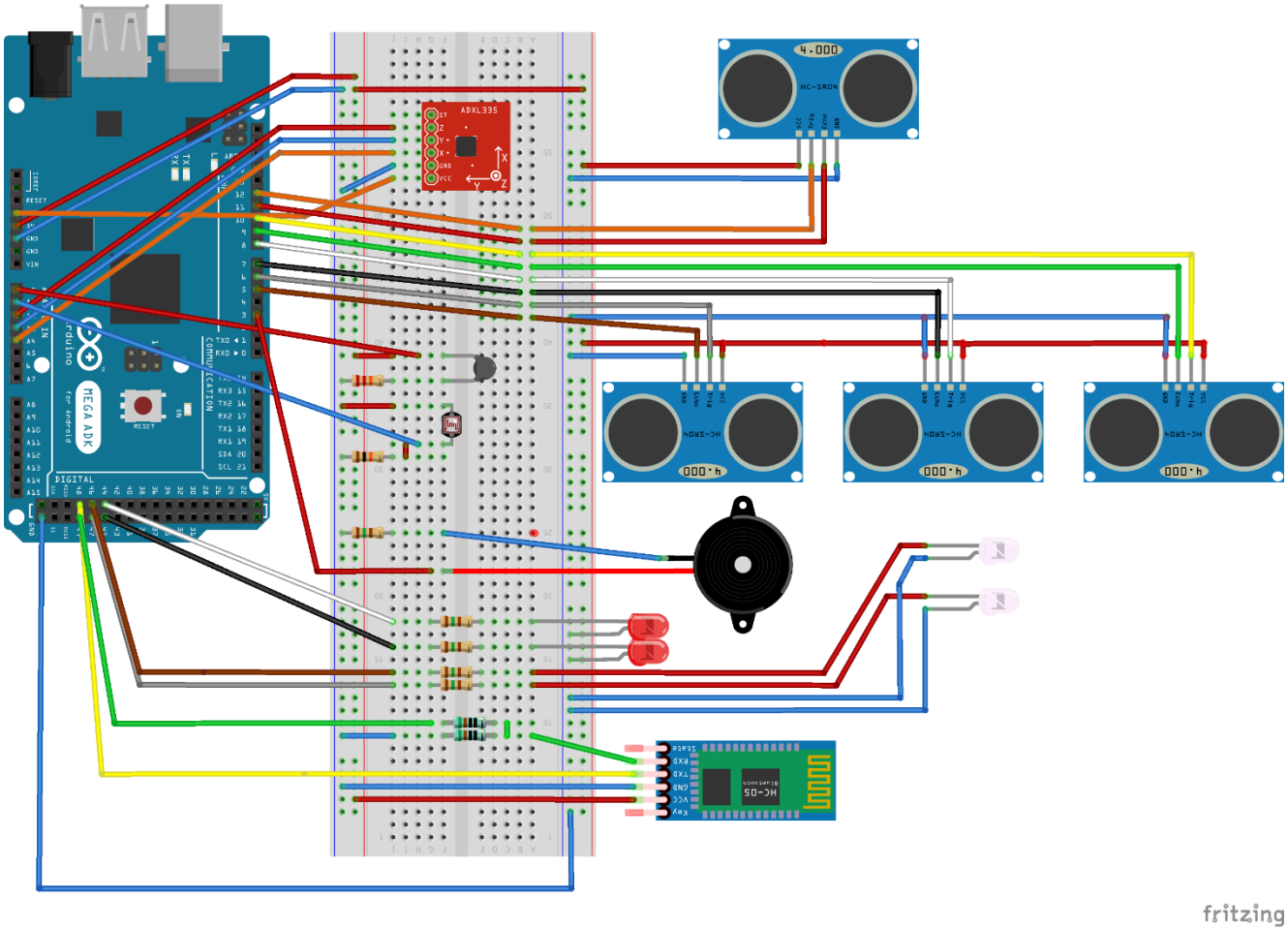
For starting up the sensors/Arduino, just connect the USB cable to a USB port with 5V, 1A output. The Arduino now is in an idle state and waits for the Bluetooth module to be connected to a respective device via the provided android application. After successfully connecting, the main routine starts. In this routine, Arduino threads[1] are started which handle basically all sensor in “parallel”. This Library helps to maintain organized and to facilitate the use of multiple tasks. We can use Timers Interrupts, and make it powerful, running "pseudo-background" tasks under the rug.

While the read sensor values are updated in the background, the Bluetooth sending routine is handled in the loop method. For sending data, data are kept in packages with start and stop byte, command byte and separators. Therefore, it is easier for the android application to parse the data and visualize them.

INFO: The code has been written in C++ and therefore the different exercises has been split into different files / classes.

[1] https://github.com/ivanseidel/ArduinoThread

Electronic Circuit

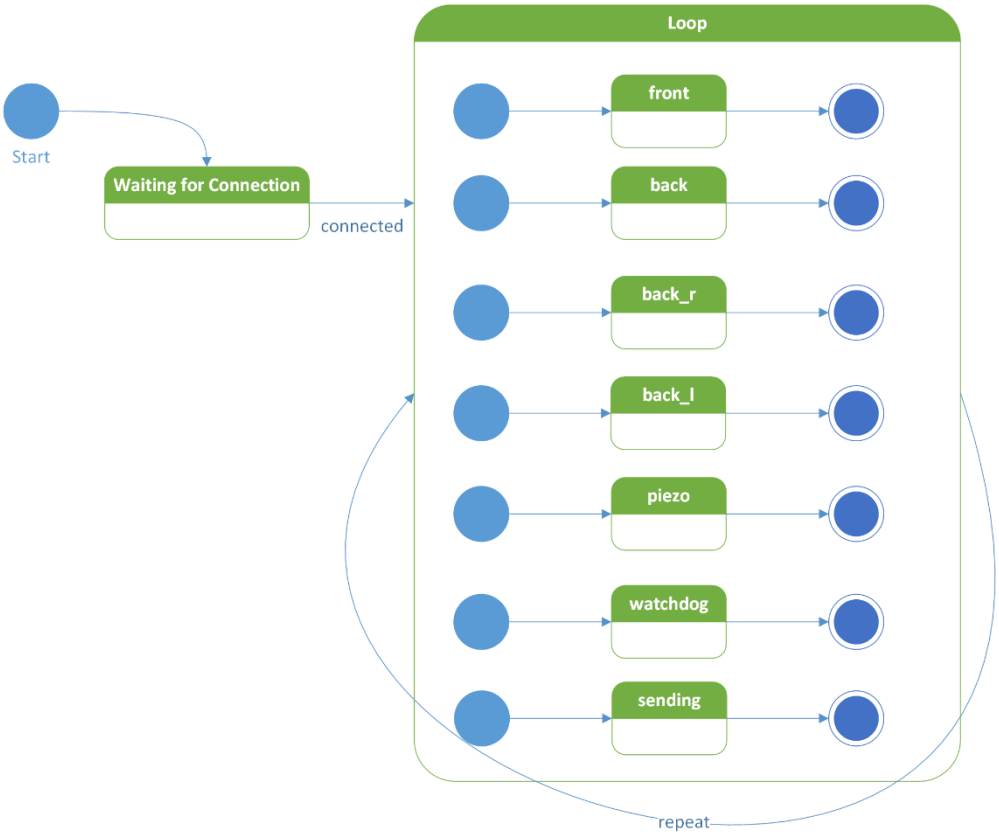


In the picture above, an Arduino Mega is shown. For our implementation we used similar hardware which is comparable with the Mega. Our hardware of choice was the SainSmart Mega which is a lot cheaper than the original one.

Hardware Used

Hardware	***Quantity***	***Price in €***
SainSmart MEGA 2560 R3 AVR Microcontroller Board	1	17,99
Set of 3 Azdelivery 40-Piece Jumper Wire	1	6,49
DSD TECH Bluetooth 4.0 BLE-Slave UART Serial-Modul	1	7,99
Neuftech HC-06 Serial Wireless 4 Pin Bluetooth RF Transceiver Module	1	6,99
HC-SR04 Ultrasonic Module	1 x 5	8,99
		48,45

State chart



As mentioned above, the states are run in “pseudo” parallelism and therefore the UML state chart looks like this. The states front, back, back_r and back_l describes the ultrasonic sensors. The sensor values are fetched in these states. The piezo state checks permanently if an obstacle is to close to the RC car and signalizes it view a sound. The Watchdog permanently checks the Bluetooth connection and resets the board if no response is received from the android application. The sending routine basically just formats all the data into packages and passes them to the Bluetooth module.

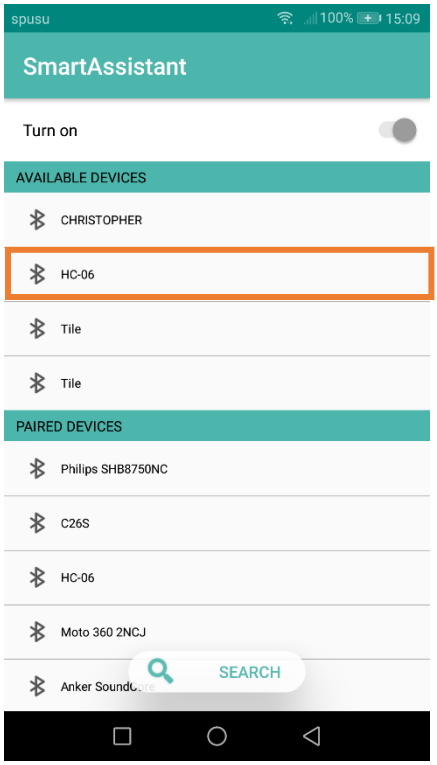
Protocol

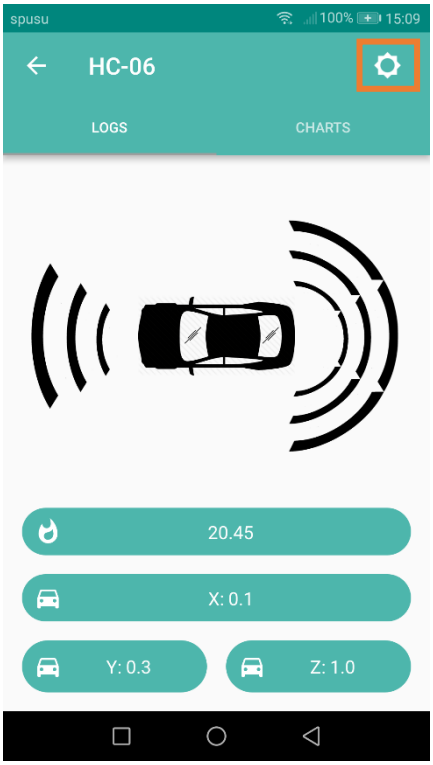
To send data over Bluetooth, an Android client has been used on the other side of communication. The android app receives the data and tries to parse them. In every message, a command is specified to signalize how the data should be parsed.

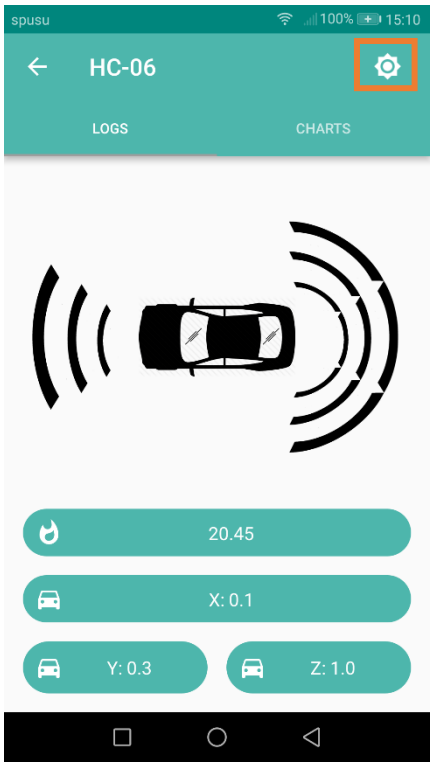
The protocol looks as follows:

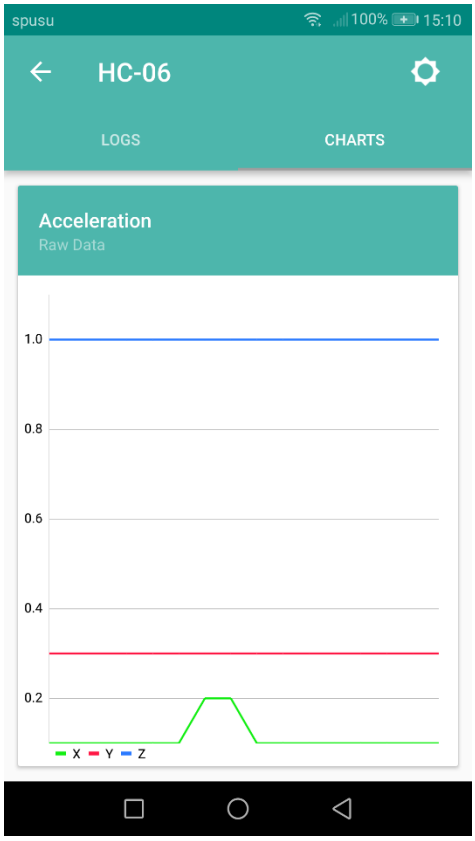
- Start Byte “$”
- Command Byte “[1-5]:”
- Separator “;”
- Data
- ACK “|”
- NACK “?”
- End Byte “*”

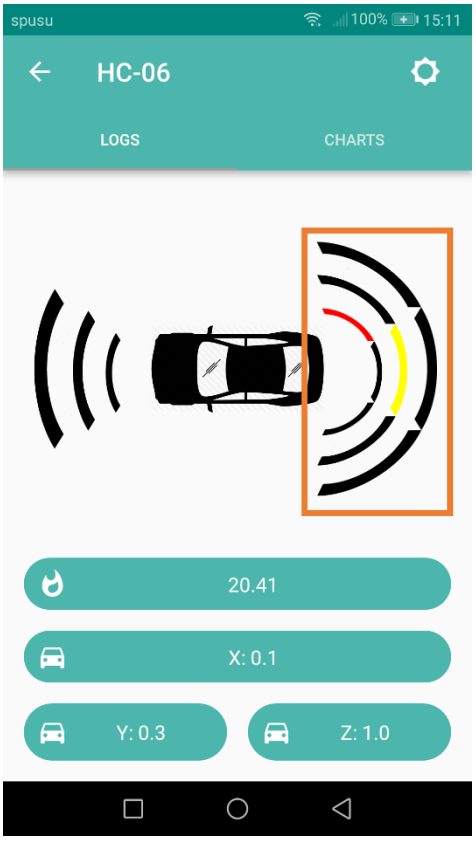
Screenshots Android








spusu
100%
15:10
HC-06
LOGS
CHARTS
Acceleration
Raw Data
1.0
0.8
0.6
0.4
0.2
X
Y
Z


spusu
100%
15:11
HC-06
LOGS
CHARTS
20.41
X: 0.1
Y: 0.3
Z: 1.0

Screenshots Hardware