SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
Faculty of Informatics and Information Technologies

Application of deterministic Ethernet in autonomous vehicle

Bc. Dávid Buhaj

Bc. Marek Číkoš

Bc. Pavol Gočal

Bc. Martin Ilavský

Bc. Milan Urminský

Supervisors: Ing. Ondrej Perešíni, Ing. Lukáš Kohútka

Content

1. INTRODUCTION

Problem of classic Ethernet network is delivery based on „best effort “. This means that there is no guaranty of packet delivery (for example during network congestion). Deterministic Ethernet is solution for this problem. Deterministic Ethernet has been developed by TTTech. This technology can guarantee packet delivery during network congestion. That is why deterministic Ethernet is widely used in real time application like automotive industry, space industry and others.

The main purpose of this project is to use deterministic Ethernet in autonomous vehicle. The vehicle should be able to avoid obstacles on the road, navigate, recognize signs. This document aims to show reader basic information of our work. It consists of several sections like analysis, solution design, implementation and experiment.

This project was developed in an iterative and incremental agile software development framework called Scrum. It means that every member of team has their tasks to do and also reports to write.

1.1 Winter semester goals

Winter semester goals can be divided in the following points:

- assignment specification and application selection
- choice of software tools for communication, code storage and task management
- component analysis
- order of components
- vehicle construction
- communication design
- basic communication between devices

Assignment specification and application selection

Our team project was a big deal that it was not directly specified. For us it meant that we had to invent our own application. It was the first and main goal of the first weeks of the semester.

Choice of software tools

The goal of the selection of software tools was to find the tools that will help us in developing our product. Tools were related to communication, code sharing and task management.

Component analysis

The goal of the analysis was to analyze the components from which the vehicle will be constructed. It is also important for compatibility in order to avoid the scenario that the two components cannot communicate with each other.

Order of components

The output from the analysis of components were devices that we have been chosen for vehicle construction. Since we cooperate with Austrian company TTTech all of the purchases has to be done via this company. Buying process begins with provided list of components and prices for company approval. If the list is approved, then the components are ordered by company. If the list it's not approved, we have to make deeper analysis and suggest new components. The whole process is under the direction and control of our external supervisor. This process takes a lot of time and it was our main slow factor.

Car construction

Based on the purchase of components, the objective is to have the physical vehicle to the end of the winter semester.

Communication design

Our autonomous car will communicate through local network. From that reason, solution requires protocol, which will enable communication. The goal of the communication design is to implement new protocol, which makes communication among end devices possible.

Basic communication between devices

One of the milestones of the winter semester is to create basic communication between devices on the local network of autonomous vehicle. This means that the objective is to send information from one device to another and vice versa.

1.2 Summer semester goals

Summer semester goals can be divided into following parts:

- Vehicle construction
- Control unit implementation
- Laser implementation
- GPS and compass implementation
- Camera implementation
- Testing of vehicle

Vehicle construction

Finish all the necessary parts on the vehicle. Mainly powering various boards, switches, wheels and so on. It is necessary to make a customized solution to ensure power supply. We will use 2 types of batteries. In addition, all the components and various other things that will occur during the project will be required.

Control unit implementation

The aim is to complete the program on the Raspberry Pi, based on the prototypes created during the summer semester. The most important part is decision-making on vehicle motion based on received data from various sensors. The goal is also to implement manual control that can be used during autonomous mode to avoid dangerous situations.

Laser implementation

Laser will be used to detect obstacles and to avoid them. It will be placed in an elevated position in the center of the vehicle so that nothing can affect it. It is necessary to program the processing of data from the laser according to the specified specification and the sending of this data to the central control unit.

GPS and compass implementation

Pomocou GPS a kompasu sa auto bude môcť pohybovať po definovanej trase. Takisto budú dáta z týchto senzorov rozhodovať spolu s dátami s lasera a smere obídenia prekážok. Dáta z týchto senzorov je nutné spracovať podľa špecifikácie, čím sa zaistí presná navigácia vozidla od jedného bodu k druhému.

Camera implementation

Using the GPS and compass, the car will be able to move via defined route. Also, the data from these sensors will be used together with laser data for obstacle avoidance. The data from these sensors must be processed according to the specification, which ensures accurate navigation of the vehicle from one point to another.

Vehicle testing

Demonstration of the functionality of our solution will be tested in real conditions near the building of FIIT using various test scenarios to test the functionality of all sensors.

1.3 General view of project

Goal of our project was to create autonomous vehicle based on deterministic Ethernet which uses deterministic TTTech switches. Requirements included vehicle size, number of engines and used boards. Four sensors are used to autonomous steering. GPS, compass, laser and camera.

1.3.1 GPS and compass

GPS is used to determine position of vehicle and to navigate it via predefined path. Compass is measuring vehicle heading. Also, a correction for GPS degree must be calculated because GPS measures geographical position and degree is computed against geographical north pole and compass measures degree to magnetic north pole. This difference between angles is called declination angle which is the 'Error' of the magnetic field in our location. Value of this angle for specific location can be found online. Its value for Bratislava is 0.07504916 in radians. GPS and compass module are located on Arduino board and they collaborate to compute the relative degree to next destination point. For further information about GPS and compass behavior and implementation see sections 3.4 and 4.1.

1.3.2 Laser

A laser scanner rotates around itself scanning the surroundings of the vehicle to avoid collision with other objects. Each rotation creates a number of ranges between which the vehicle can safely move. For further information about laser behavior and implementation see sections 3.5 and 4.5.

1.3.3 Camera

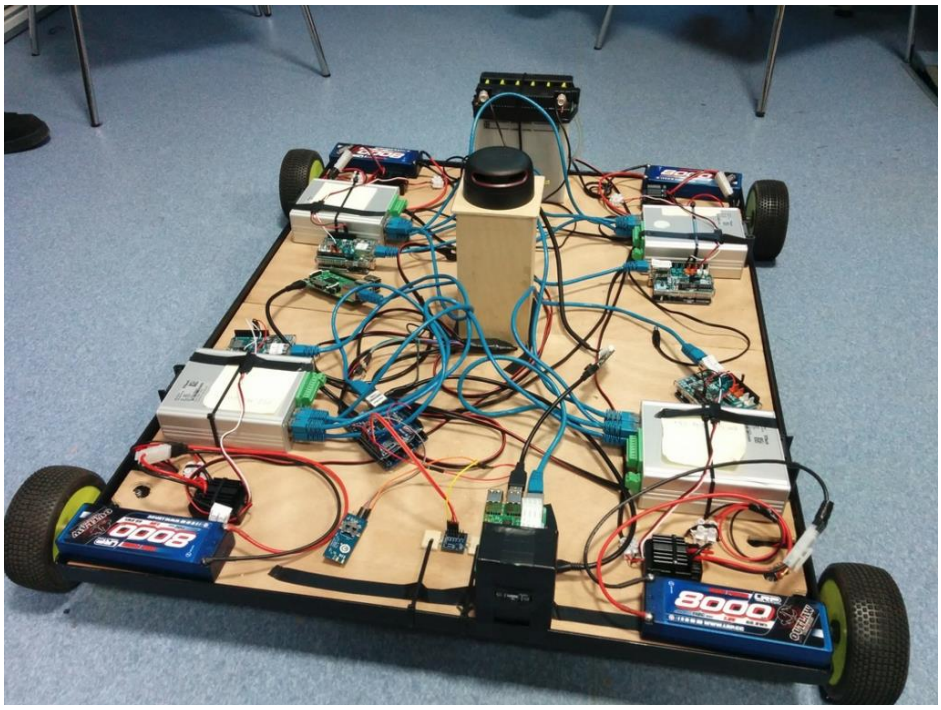
A camera records the road to navigate the vehicle on it. Each recorded frame is being processed searching for the road side. After it is found, an angle is calculated by using the reversed tangent function (arctan) on the coordinates of two points that we get on each frame. The frame processing is mostly built on the *floodfill* method. For further information about camera behavior and implementation see sections 3.7 and 4.6

1.3.4 Control unit

The control unit is the brain of our vehicle. It makes all the decisions and controls the movement of the vehicle. All data from sensors are sent to this device and movement orders are then sent to the engines. The control unit not only allows autonomous movement of the vehicle, but also manual control.

1.3.5 Vehicle construction

We have constructed our vehicle from ordered parts and tested its movement control. Current state of our vehicle can be seen on picture below. Design can be found in chapters 3.1 and 3.2 and car wiring in chapter 4.4. Experiments from movement testing are in chapters 5.1 and 5.2.



Constructed vehicle

2. ANALYSIS

This section contains analysis of components which some of them are part of autonomous car. Components are divided into sensors, batteries, car platforms, boards, cameras.

2.1 GPS

GPS receivers use a constellation of satellites and ground stations to compute position and time almost anywhere on earth. At any given time, there are at least 24 active satellites orbiting over 12,000 miles above earth. The positions of the satellites are constructed in a way that the sky above your location will always contain at most 12 satellites. The primary purpose of the 12 visible satellites is to transmit information back to earth over radio frequency (ranging from 1.1 to 1.5 GHz). With this information and some math, a ground based receiver or GPS module can calculate its position and time.

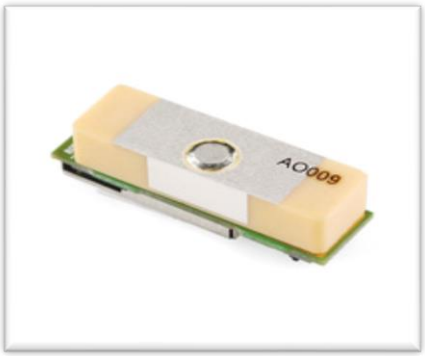
The data sent down to earth from each satellite contains a few different pieces of information that allows your GPS receiver to accurately calculate its position and time. An important piece of equipment on each GPS satellite is an extremely accurate atomic clock. The time on the atomic clock is sent down to earth along with the satellite's orbital position and arrival times at different points in the sky. In other words, the GPS module receives a timestamp from each of the visible satellites, along with data on where in the sky each one is located (among other pieces of data). From this information, the GPS receiver now knows the distance to each satellite in view. If the GPS receiver's antenna can see at least 4 satellites, it can accurately calculate its position and time**.** This is also called a lock or a fix.

All of GPS modules can be compared according to parameters. Here are some of them.

Accuracy - GPS accuracy varies but you can usually find out where you are, anywhere in the world, within 30 seconds, down to +/– 5 meters. The +/– is there because accuracy can vary between modules, time of day, clarity of reception, etc. Overall, to get the best accuracy from GPS, it must be in clear view of the sky and moving.

Antenna - GPS module is receiving signals from satellites about 12,000 miles away, so for the best performance, we need a clear path between the antenna and most of the sky. Weather, clouds, snow storms, shouldn't affect the signal, but things like trees, buildings, mountains, the roof, will all create unwanted interference and GPS accuracy will suffer.

There are many antenna choices, but these are some of the most common.

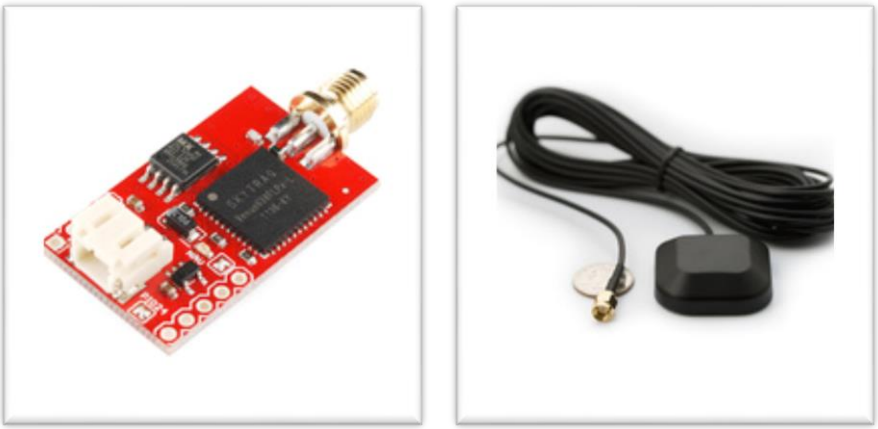


Ceramic patch antenna.

This antenna is low profile, inexpensive, and compact, but it has lower reception compared to other types of antennas. This antenna needs to face upwards with a clear view of the sky to get good a good signal, so the gain of the antenna is greatest when facing up.

Helical antenna

This antenna can take up more room than the ceramic patch, but the shape of the antenna allows for a better signal in any orientation, at the expense of slightly lower gain in any one specific orientation.



Module with a SMA antenna attachment

The SMA attachment gives the ability to mount antenna in a different location than our main circuit. This can be beneficial if main system is not in good view of the sky. For example, inside of a building or in a car which can be our case.

Baud Rate - GPS receivers send serial data out of a transmit pin (TX) at a specific bit rate. The most common is 9600bps for 1Hz receivers but 57600bps is becoming more common.

Channels - The number of channels that the GPS module runs will affect time to first fix (TTFF). Since the module doesn't know which satellites are in view, the more frequencies/channels it can check at once, the faster a fix will be found. After the module gets a lock or fix, some modules will shut down the extra blocks of channels to save power.

Chipset - The GPS chipset is responsible for doing everything from performing calculations, to providing the analog circuitry for the antenna, to power control, to the user interface. The chipset is independent of the antenna type, therefore you can have a range of different antennas for GPS modules with specific chipsets. Common chipsets are ublox, SiRF, and SkyTraq and all contain very powerful processors that allow for fast acquisitions times and high reliability. The differences between chipsets usually falls on a balance between power consumption, acquisition times, and accessibility of hardware.

Gain - The gain is the efficiency of the antenna in any given orientation. This applies to both transmitting antennas and receiving antennas.

Lock or Fix - When a GPS receiver has a lock or fix, there are at least 4 satellites in good view and you can get accurate position and time.

NMEA - This is a common data format that most GPS modules use. NMEA data is displayed in sentences and sent out of the GPS modules serial transmit (TX) pin. The NMEA sentences contain all of the useful data, (position, time, etc.).

Power - On average, a common GPS module, with a lock, draws around 30mA at 3.3V.

PPS - Pulse per second. This is an output pin on some GPS modules.

Start-up Times (Hot, Warm, and Cold) - Some GPS modules have a super-capacitor or battery backup to save previous satellite data in volatile memory after a power down. This helps decrease the TTFF on subsequent power-ups. Also, a faster start time translates into less overall power draw.

- Cold Start - If you power down the module for a long period of time and the backup cap dissipates, the data is lost. On the next power up, the GPS will need to download new almanac and ephemeris data.
- Warm Start - Depending on how long your backup power lasts, you can have a warm start, which means some of the almanac and ephemeris data is preserved, but it might take a bit extra time to acquire a lock.
- Hot Start: A hot start means all of the satellites are up to date and are close to the same positions as they were in the previous power on state. With a hot start the GPS can immediately lock.

TTFF - Time to first fix. The time it takes, after power-on, to accurately compute your position and time using at least 4 satellites. If you are in a location with a bad view of the sky, the TTFF can be very long.

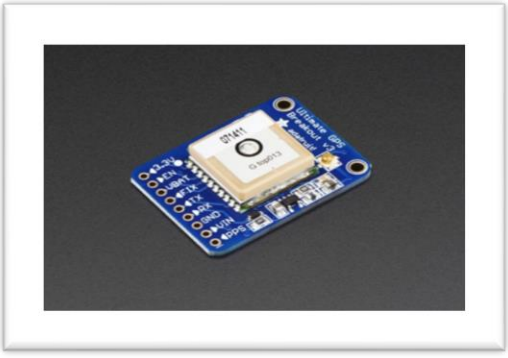
Update Rate - The update rate of a GPS module is how often it calculates and reports its position. The standard for most devices is 1Hz (once per second). UAVs and other fast vehicles may require increased update rates. 5 and even 10Hz update rates are becoming available in low cost modules.

2.1.1 GPS modules

We have been deciding mainly among two boards for Raspberry Pi. Adafruit ultimate GPS breakout and RasPiGNSS. The main decision factors were ease of installation, detailed documentation and existing projects on forums. After selecting Altera board a PMOD GPS Receiver (SKU: 410-237) was chosen.

2.1.1.1 Adafruit Ultimate GPS Breakout

This module has detailed step by step guide for Arduino and raspberry pi boards and that is the main reason we have chosen this GPS module. It offers very good specifications according to its price.



Adafruit Ultimate GPS Breakout

It supports also DGPPS, WAAS, EGNOS, jammer detection and reduction and multi-path detection and compensation. Output of this module is in standard NMEA format with 9600 baud rate.

Parameters	
Price	+/- 40 Euro
Weight	8.5g
Dimensions	25.5mm x 35mm x 6.5mm
Sattelites	22 tracking, 66 searching
Patch Antenna Size	15mm x 15mm x 4mm
Update rate:	1 to 10 Hz
Position Accuracy:	< 3 meters
Velocity Accuracy:	0.1 meters/s
Warm/cold start:	34 seconds
Acquisition sensitivity:	-145 dBm
Tracking sensitivity:	-165 dBm
Maximum Velocity:	515m/s
Vin range:	*3.0-5.5VDC*

2.1.1.2 RasPiGNSS

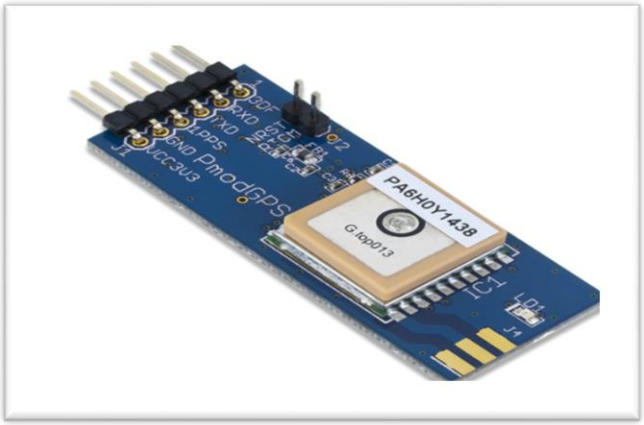
This module was tested for compatibility with raspberry pi 3 B and has updated installation guide according to it. For 170 euro, it is not very much better than the previous Adafruit ultimate GPS breakout.

RasPiGNSS

Parameters	
Price	+/- 170 Euro
Weight	22g
Channels	32
Patch Antenna Size	15mm x 15mm x 4mm
Update rate:	1 to 10 Hz
Position Accuracy:	< 2 meters
Velocity Accuracy:	0.05 meters/s
Warm/cold start:	25 seconds
Tracking sensitivity:	-160 dBm
Maximum Velocity:	500m/s
Maximum Acceleration	5G

2.1.1.3 PMOD GPS receiver

The Pmod GPS can provide satellite positioning accuracy to any embedded system. By communicating through UART with the GlobalTop FGPMMOPA6H GPS antenna, users may benefit from the 3 meters accuracy for any long term traveling. Due to an end of life notice on the Gms-u1LP antenna module, the PmodGPS will be using the FGPMMOPA6H module.



PMOD GPS receiver

Parameters	
Price	+/- 39.99 Euro
Weight	22g
Channels	66, 22 tracking
Patch Antenna Size	50mm x 20mm x 4mm
Update rate:	1 to 10 Hz
Position Accuracy:	< 3 meters
Velocity Accuracy:	0.1 meters/s
Warm/cold start:	33 / 35 seconds
Tracking sensitivity:	-165 dBm
Maximum Altitude	18 000 meters
Maximum Velocity:	515m/s
Maximum Acceleration	4G

2.1.1.4 Brief analysis of other modules

Gps add-on

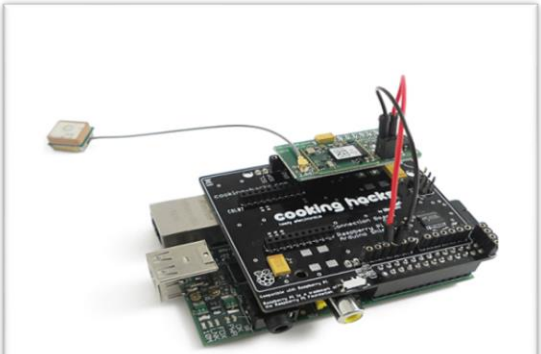
The 25.75€ add-on for Raspberry Pi B is based on the NEO-6 GPS module. With an input voltage of 3.3V and UART interface, the module returns information such as the current location and time. The add-on is also compatible with the Raspberry Pi Model B+.

GPS expansion board

Specially designed for Pi Model B+, the GPS board provides general information about the position and time. At a price of 47.00€, the board is based on the low power usage and high-performance positioning module called Ublox MAX-M8Q.

USB GPS Dongle

The easiest way to turn your fruit-named single board computer into a navigation device is to use a USB GPS dongle. At a price of 39.00€, the small piece of hardware supports Linux and ARM architecture. Also, it’s based on the high sensitive GPS chipset called SiRF Star III.



GPS shield

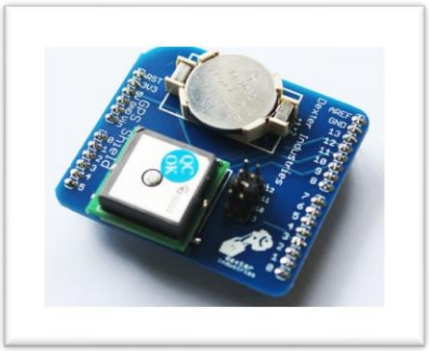
Using the standard NMEA protocol to provide information like speed, position and altitude, the GPS shield works great both inside and outside. It is available at a price of €82.00 and enables the data via serial port.

EM-506

The €35.00 GPS module is another receiver based on the SiRF StarIII chipset. Like the USB GPS dongle described above, the EM-506 provides the position very accurate even in urban canyon and dense foliage environment. The features include a position accuracy of 2.5m, and without any network assistance, it can predict for up to three days the satellite positions.

3G/GPRS shield

The 3G/GPRS shield is a device designed for Internet of Things applications. And because we are talking here about GPS data, the shield also provides the location and stay connected to the 3G network. The price is huge, about €149.00, and it's compatible with Pi, Intel Galileo and Arduino boards.



Dexter Industries GPS

With an accurate position of 2.5 meters and a velocity of 0.1 m/sec, the Dexter Industries GPS is a good solution to build an all-in-one tracking application. The €39.00($45.00) shield can work on Raspberry Pi only with the Arduberry shield. The Arduberry shield is compatible with the Raspberry Pi and allows you to attach the receiver shield.

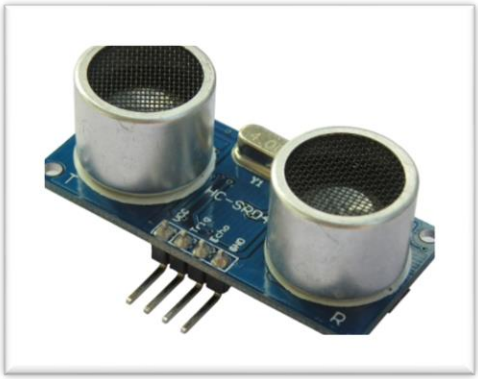
2.2 DISTANCE SENSORS

2.2.1 Ultrasonic sensors

- **Advantages**
  - do not use much electricity;
  - simple in design;
  - relatively inexpensive;
- **Disadvantages**
  - density, consistency, and material can distort an ultrasonic sensor's readings.

2.2.1.1 HC-SR04

- Working Voltage DC 5 V
- Working Current 15mA
- Range: 2cm - 400cm
- Accuracy: 3mm
- Dimension: 45*20*15mm
- Price: about $2/1 piece



HC-SR04

Ultrasonic sensors are popular for their price and reliability. Laser rays can be in some outdoor environment disrupted and in these cases, ultrasonic sensors can take a place. With this type of ultrasonic sensor (HC-SR04) is really easy to work and implement. They work very well with Arduino microcontrollers. Though we have chosen 360° laser sensor, which is described further, these sensors can be buy in next phase of project, if there will be some problems with laser sensor.

Sharp GP2Y0A02YK0F

2.2.2 Infrared sensors

- **Advantages**
  - can detect infrared light from far distances over a large area;
  - operate in real-time;
  - can receive infrared light that is irradiated from both living and non-living objects.

- **Disadvantages**
  - incapable of distinguishing between objects that irradiate similar thermal energy levels.

2.2.2.1 Sharp GP2Y0A02YK0F

- Analog output varies from 2.8V at 15cm to 0.4V at 150cm
- Distance measuring range: 20 to 150 cm
- Package size: 29.5×13×21.6 mm
- Supply voltage: 4.5 to 5.5 V
- Price: €14.15

This type of infrared sensor is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector), IRED (infrared emitting diode) and signal processing circuit. The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method. This device outputs the voltage corresponding to the detection distance. So, this sensor can also be used as a proximity sensor. It is also suitable for robot applications.

2.2.3 Laser sensors

- **Advantages**
  - Higher accuracy
  - Fast acquisition and processing
  - Higher speed of measurement
- **Disadvantages**
  - Higher costs

2.2.3.1 RPLIDAR 360° A2

- 360 degree laser scanner development kit
- Omnidirectional laser scan
- User configurable scan rate
- Ideal Sensor for robot localization & mapping
- Price: €412.61

RPLIDAR 360° A2

RPLIDAR A2 is the next generation low cost 360 degree 2D laser scanner (LIDAR) solution. It can take up to 4000 samples of laser ranging per second with high rotation speed. The system can perform 2D 360-degree scan within a 6-meter range. The generated 2D point cloud data can be used in mapping, localization and object/environment modeling. The typical scanning frequency of the RPLIDAR A2 is 10hz (600rpm). Under this condition, the resolution will be 0.9°. And the actual scanning frequency can be freely adjusted within the 5-15hz range according to the requirements of users. The RPLIDAR A2 adopts the low cost laser triangulation measurement system, which makes the RPLIDAR A2 has excellent performance in all kinds of indoor environment and outdoor environment without direct sunlight exposure. Meanwhile, before leaving the factory, every RPLIDAR A2 has passed the strict testing to ensure the laser output power meet the standards of FDA Class I.

It is suitable for applications as obstacle avoidance, autonomous mapping, route planning or navigation. From that reason, we have chosen this type of distance sensor for our application to measure distance from obstacle around the car robot.

2.3 BATTERIES

2.3.1 Gogen Power Bank 12000 mAh black-gray

Key features

Capacity 3.7 V - 12000 mAh / 44.4 Wh

Input: Micro USB 5 V / 2 A

Output: 2 x USB (5 V / 2.1 A and 5 V / 2.5 A)

LED charge status indicator

LED flashlight

Power on the road

Thanks external rechargeable battery GoGEN high capacity 12000 mAh will make your travel easier, because your device as smartphone, MP3 and MP4 player, GPS navigation, outdoor camera, camera or tablet will be able to use a much longer without the fear that the end of the stay You will no longer have the "juice" to operate the device. Conventional mobile phones and recharge at least 5 times, depending on the capacity of the battery being charged devices. 5 V outputs with 2.1 and 2.5 and allows you to charge the device with higher current consumption such as tablets and mobile phones from Apple.

Fast charging

Thanks to the current 2.5 A can with UPS GoGEN charge your device up to two times faster than conventional chargers. The battery can use almost any mobile device, which can be powered via the USB port. LED will show the remaining power of the backup source.

Gogen Power Bank 12000 mAh black-gray

Price: 25.99 Eur

2.3.2 USB Battery Pack for Raspberry Pi - 10000mAh - 2 x 5V outputs

Description

A large-sized rechargeable battery pack for Raspberry Pi or anything else that uses 5V. This pack is intended for providing a lot of power to an GPS, cell phone, tablet, etc. But we found it does a really good job of powering other miniature computers and micro-controllers.

Inside is a massive 10,000mAh lithium ion battery, a charging circuit (you charge it via the USB cable attached), and two boost converters that provide 5VDC, 1A and 2A each via a USB A port. (The markings indicate one is good for 1A and one is good for 2A) The 2A output is best for charging tablets or other power-hungry devices. But either can be used for when you want to power a Beagle Bone or Raspberry Pi, Wi-Fi adapters, maybe even small displays.

The charging circuit will draw 1A from a 5V supply (plug a microUSB connector into the pack and then to a computer or wall adapter). You can charge and power something at the same time but the output switches to the USB input when charging so the output voltage may fluctuate. Its not good as a 'UPS' power supply for an embedded linux board, although microcontrollers like Arduino may not care about the voltage drop as much. Also, there's ~80% efficiency loss on both ends so if you charge it at 1A and draw 1A at the same time, the battery pack will eventually go empty. However, if you're powering something that’s 500mA or less, you can keep it topped up no problem. Also, when you start and stop charging the pack, it will flicker the output, this can cause a 'power sensitive' device like the Pi or an iPhone to reset on the power supply. If using it with a low current load, say under 100mA, the pack may 'fall asleep' - you can use this circuit to keep the pack awake.

2 USB Battery Pack for Raspberry Pi - 10000mAh

2.3.3 Xiaomi Power Bank 20000mAh White

This modern and elegant Power Bank features an ultra-high capacity of 20 000 mAh which allows for multiple recharge of most devices. This external source can be used for charging mobile phones, iPods, GPS navigation, MP3 players, cameras and digital cameras. Appreciate the large capacity of this battery on long trips, road trips and holidays, wherever there is no access to power network.

Key Features

Power Bank with a lightweight and durable aluminum body
A pair of USB to charge two devices simultaneously
Rounded edges for easy grip
High capacity sufficient for multiple phone recharge
Support for fast charging (DC 5V / 2A 9V / 2A 12V / 1.5A)
LEDs indicating the battery status
Compatible with all USB devices
Li-Ion battery with a capacity of 20,000 mAh
Practical and Neat

Integrated Li-Ion battery can be easily recharged via the USB connector. LEDs will indicate the actual state of your battery. Cutting-edge control microchips offer safe protection against over-voltage, over-heating, over-charging and discharging. This Power Bank features a sleek white casing with an elongated shape that fits comfortably to your palm. With this excellent backup battery your devices will always be ready to use.

Xiaomi Power Bank 20000mAh White

2.4 CAR PLATFORMS

2.4.1 GEARS SMP Mobile Platform

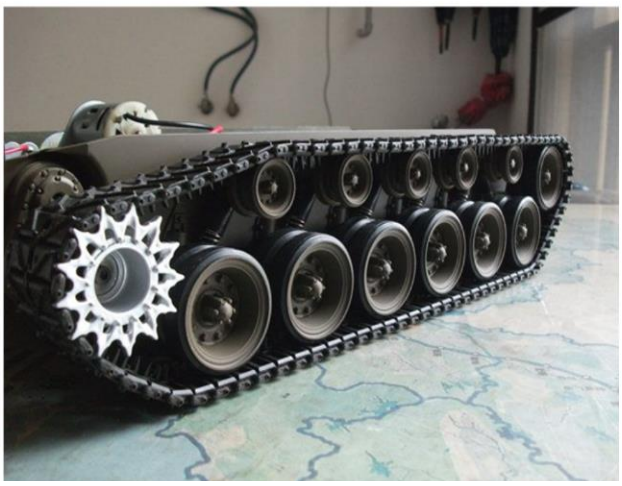
- Robust aluminum chassis for your RC or autonomous robot experimentation
- Innovative suspension system
- Customizable using GEARS aluminum parts
- Additional payload capacity: 8,1 kg
- Does not include encoders
- 0.46 m x 0,50 m x 0,33 m
- Price: €1,500.07 (without encoders)

GEARS SMP Mobile Platform

The GEARS-SMP Mobile Platform was conceived for educational programs looking to integrate robotic sensors and control in a robust mobile platform. The platform can be used indoors and outdoors and navigates the terrain using an innovative suspension system and skid steering. Although the standard version has a 4.5" ground clearance, the entire platform can be customized, raised or lowered or added to using GEARS-EDS parts sold separately.

2.4.2 Track Chassis

- Motor voltage: 6V-12V
- 0.96 x 0.55 x 0.25 m
- Price: $88.00
- Motor speed:
    - 3V 6915 turn 0.52A
    - Turn 0.66A 6V
- Drive gearbox ratio: 39.25: 1



Track Chassis

Big tank chassis provide much better driving control with used tracks. This platform also has a lot of space for placing our components on the top of the chassis. The seller did not give a lot of information about this platform.

2.4.3 C37 4WD Car

- Car body: Aluminum Alloy
- 400* 300*130 mm
- Working voltage: 12V
- The Car whole weight: 2 kg
- Working carrying capacity: 18kG
- Price: $128.34

3 C37 4WD Car

The platform has four robust wheels, which can provide better stability in rough surface. Although overall chassis dimensions are not too big, there is possibility to mount bigger chassis above the wheels, where we can put all components.

2.4.4 6WD ATR RC with 90mm motors

- IG90 gear motors
- ATR frame: 0.35 x 0.91 m
- Length of chassis: 1.04 m
- 35Ah battery
- Price: $2,499.00


6WD ATR RC with 90mm motors

This is a rugged frame made of 0,47cm thick aluminum to accept IG90 gear motors. The aluminum is all laser cut and CNC bent for an exact fit-up.

The IG90 ATR frame is 0,35m wide x 0,91m long. The sides are 5 cm high (bottom edge to top surface). As configured with 0,33cm tiller tires shaft and wheel sets it has a ground clearance of about 12,7cm and total width (wheel edge to wheel edge) of about 0,69m. The length of chassis is about 1,04m (wheel edge to wheel edge). The overall height is just the wheel diameter of 33cm.

2.4.5 Pre-Built 4WD IG52

- Fully assembled and ready to run
- IG52 24VDC 285 RPM Gear Motors
- Spektrum Transmitter and Receiver 2.4GHz
- 12 Volt 18Ah Sealed Lead Acid Battery (run in series for 24V)
- Speed: 3 mph
- 0.74 m x 0,74 m
- Payload of approximately 18-27 kg

Pre-Built 4WD IG52

This is a robot series that is designed to drive over just about any terrain for use with surveillance, academic research, and most practical robotic applications. It works on any indoor surface and most outdoor surfaces. The platform is already configured, but custom configuration is possible.

2.4.6 GRIZZLY (RUV)

- industry-leading robot workhorse
- four independently driven wheels
- 1750 x 1282 x 811 mm
- Max Payload: 600 kg
- Speed: 18 mph
- 16-degree front axle articulation
- 200 Ah, 48V sealed lead acid battery pack
- User Power: 5V, 12V, 24V and 48V
- Unknown price

GRIZZLY (RUV)

Grizzly is a large all-terrain robotic utility vehicle that offers the performance of a tractor and the precision of an industrial robot. This all-electric workhorse has a maximum continuous drawbar force of 1400 lbf and a payload capacity of 600kg.

Grizzly is built for the most demanding outdoor environments, making it ideal for mining, military and agricultural research. It interfaces with a variety of payloads, including single-point hitch implements as well as all of Clearpath's sensing, computing and manipulator packages.

Grizzly has four independently driven wheels, each with high-resolution (2500 counts per revolution) encoders and finely tuned closed-loop control. The result is precise linear position control even at low speeds.

2.5 BOARDS

In this section, analysis of available boards is presented. We have chosen four boards - Raspberry Pi 3 Model B, DE1-SoC-MTL2, Banana Pi-M2+ and Arduino Uno.

2.5.1 Raspberry Pi 3 Model B

Raspberry Pi 3 is the third-generation Raspberry Pi. For Raspberry Pi 3, Broadcom have supported us with a new SoC, BCM2837. This retains the same basic architecture as its predecessors BCM2835 and BCM2836, so all those projects and tutorials which rely on the precise details of the Raspberry Pi hardware will continue to work. The 900MHz 32-bit quad-core ARM Cortex-A7 CPU complex has been replaced by a custom-hardened 1.2GHz 64-bit quad-core ARM Cortex-A53. Combining a 33% increase in clock speed with various architectural enhancements, this provides a 50-60% increase in performance in 32-bit mode versus Raspberry Pi 2, or roughly a factor of ten over the original Raspberry Pi.

Raspberry Pi 3 Model B Specification

Processor Chipset	Broadcom BCM2837 64Bit Quad Core Processor powered Single Board Computer running at 1.2GHz
Processor Speed	QUAD Core @1.2 GHz
RAM	1GB SDRAM @ 400 MHz
Storage	MicroSD
USB 2.0	4x USB Ports
Max Power Draw/voltage	2.5A @ 5V
GPIO	40 pin
Ethernet Port	Yes
WiFi	Built in
Bluetooth Low Energy (BLE)	Built in
CSI camera port/DSI display port	Built in/Built in

2.5.2 DE1-SoC-MTL2

The DE1-SoC-MTL2 Development Kit is a comprehensive design environment with everything embedded developers need to create processing-based systems. The DE1-SoC-MTL2 delivers an integrated platform including hardware, design tools, and reference designs for developing embedded software and hardware platforms in a wide range of applications. The fully integrated kit allows developers to rapidly customize their processor and IP to best suit their specific application. The DE1-SoC-MTL2 features a DE1-SoC development board targeting Altera Cyclone® V SX SoC FPGA, as well as a capacitive LCD multimedia color touch panel which natively supports five points multi-touch and gestures.

The all-in-one embedded solution offered on the DE1-SoC-MTL2, in combination of a LCD touch panel and digital image module, provide s embedded developers the ideal platform for multimedia applications with unparallel processing performance. Developers can benefit from the use of FPGA-based embedded processing system such as mitigat ing design risk and obsolescence, design reuse, lowering bill of material (BOM) costs by integrating powerful graphics engines within the FPGA.

DE1-SoC-MTL2 Specification

Cyclone V SE SoC—5CSEMA5F31C6N	Dual-core ARM Cortex-A9 (HPS) 85K programmable logic elements 4,450 Kbits embedded memory 6 fractional PLLs
Memory Device	64MB (32Mx16) SDRAM for the FPGA 1GB (2x256MBx16) DDR3 SDRAM for the HPS microSD card socket for the HPS
Peripherals	Two port USB 2.0 Host UART to USB (USB Mini B connector) 10/100/1000 Ethernet PS/2 mouse/keyboard I2C multiplexer
Connectors	Two 40-pin expansion headers One 10-pin ADC input header One LTC connector (one Serial Peripheral Interface (SPI) master ,one I2C bus, and one GPIO interface)
Display	24-bit VGA DAC
Sensors	G-Sensor on HPS
Switches, Buttons and LEDs	5 user keys (4 for the FPGA and 1 for the HPS) 10 user switches for the FPGA 11 user LEDs (10 for the FPGA and 1 for the HPS) 2 HPS reset buttons (HPS_RESET_n and HPS_WARM_RST_n) Six 7-segment displays
Power	12V DC input

2.5.3 BANANA Pi-M2 +

Banana Pi M2 is a second-generation single board computer with an upgraded SoC to provide even more power for computing tasks. It features high performance quad-core SoC, 1GB of DDR3 SDRAM, Gigabit Ethernet, 4 USB, and HDMI connection. It can run on a variety of operating systems including Android, Lubuntu, Ubuntu, Debian, and Raspbian.

CPU	A31S ARM Cortex-A7™ Quad-core 256KB L1 cache 1MB L2 cache
GPU	PowerVR SGX54MP2 Comply with OpenGL ES 2.0 OpenCL 1x,DX9_3
Memory	1GB DDR3 (shared with GPU)
Storage Support	MicroSD Card(up to 64GB)
Onboard Network	10/100/1000 Ethernet RJ45
WiFi	WiFi 802.11b/g/n
Video In	Parallel 8-bit camera interface
Video Out	HDMI,LVDS/RGB (no composite video)
Audio Out	3.5 mm Jack and HDMI
Audio In	On board microphone
Power Source	5V DC @ 2A (4.0mm/1.7mm barrel plug - centre positive) or USB OTG
USB Ports	4x USB 2.0
Buttons	Power/Reset: next to Camera Connector
GPIO	GPIO,UART,I2C BUS,SPI BUS,ADC,PWM,+3.3V,+5V,GND
LED	Power key and RJ45
OS	Android and Linux etc.OS

2.5.4 Arduino UNO

Arduino/Genuino Uno is a microcontroller board based on the ATmega328PIt has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

2.6 Camera

For our vehicle, we will need 2 cameras to monitor the surroundings for good navigation on the road. One of these cameras is supposed to record the traffic signs, while the other will be used as an infrared sensor for road lines. The size of the cameras is very important, because we are working with a fairly small vehicle with limited space and carrying capacity. Also our budget is not too big, so the price has to be reasonable.

2.6.1 Outdoor Full HD WDR PoE Day/Night Bullet Network Camera - DCS-7513

Camera Hardware Profile

- 1/2.8” 2 Megapixel progressive CMOS sensor
- 30 meter IR illumination distance
- Minimum illumination: 0 lux with IR LEDs on
- Built-in Infrared-Cut Removable (ICR) Filter module
- 10x digital zoom
- 3 to 9 mm motorised varifocal lens
- Aperture: F1.2
- Angle of view (16:9)
- (H) 121.2° to 38.1°
- (V) 62.1° to 21.3°
- (D) 148.4° to 43.8°

Camera Housing

- IP-66 compliant weatherproof housing
- Cable management bracket

Image Features

- Configurable image size, quality, frame rate, and bit rate
- Time stamp and text overlays
- Configurable motion detection windows
- Configurable privacy mask zones
- Configurable shutter speed, brightness, saturation, contrast, sharpness, zoom, focus, and aperture

Video Compression

- Simultaneous H.264/MPEG-4/MJPEG format compression
- H.264/MPEG-4 multicast streaming
- JPEG for still images

Video Resolution

- 16:9 at 1920 x 1080,1280 x 720, 800 x 450, 640 x 360, 480 x 270, 320 x 176, 176 x 144 up to 30 fps
- 4:3 at 1440 x 1080, 1280 x 960, 1024 x 768, 800 x 600, 640 x 480, 320 x 240, 176 x 144 up to 30 fps

Audio Support

- G.726
- G.711

External Device Interface

- 10/100 BASE-TX Ethernet port with PoE
- 1 DI / 1 DO
- DC12 V, 100 mA output
- SD/SDHC card slot
- Audio input/output
- DI/DO connector 12 V DC output

Network Protocols

- IPv6
- IPv4
- TCP/IP
- UDP
- ICMP
- DHCP client

- NTP client (D-Link)
- DNS client
- DDNS client (D-Link)
- SMTP client
- FTP client
- HTTP / HTTPS
- Samba client
- PPPoE
- UPnP port forwarding
- RTP / RTSP/ RTCP
- IP filtering
- QoS
- CoS
- Multicast
- IGMP
- ONVIF compliant

Security

- Administrator and user group protection
- Password authentication
- HTTP and RTSP digest encryption

System Requirements for Web Interface

- Browser: Internet Explorer, Firefox, Chrome, or Safari

Event Management

- Motion detection
- Event notification and uploading of snapshots/video clips via e-mail or FTP
- Supports multiple SMTP and FTP servers
- Multiple event notifications
- Multiple recording methods for easy backup

Remote Management

- Take snapshots/video clips and save to local hard drive or NAS via web browser
- Configuration interface accessible via web browser

Operating Systems

- Windows 7/Vista/XP/2000

D-ViewCam™ System Requirements

- Operating System: Microsoft Windows 7/Vista/XP
- Web Browser: Internet Explorer 7 or higher
- Protocol: Standard TCP/IP

D-ViewCam™ Software Functions

- Remote management
- Control and manage up to 32 cameras
- View up to 32 cameras on one screen
- Management functions provided in web interface
- Scheduled, motion detection, and manual recording triggers

Dimensions

- 223.5 x 97.5 x 90.7 mm

Weight

- 2050 g (with bracket and sunshield)

External Power Adapter

- Input: 100 to 240 V AC, 50/60 Hz
- Output: 12 V 1.25 A

Power Consumption

- 11.02 watts ± 5 %

Temperature

- Operating: -40 to 50 °C (-40 to 122 °F)
- Storage: -20 to 70° C (-4° to 158° F)

Humidity

- Operating: 20% to 80% non-condensing
- Storage: 5% to 95% non-condensing

Certifications

- CE
- CE LVD
- FCC
- C-Tick

Price: 700€

This is a surveilance camera made for outdoor building security. It provides images and also streams of footage in full hd resolution. Being an IP camera, it sends the recorded data using the ethernet connection. Does provide an infrared illumination, so it can work at night.

Since this camera is quite big, it might prove to be a problem to install it on our vehicle. It's price is very high considering our budget. It's built for windows operating system and since we are going to work with the Raspberry Pi motherboard, we'll be working in linux OS.

Outdoor Full HD WDR PoE Day/Night Fixed Bullet Network Camera - DCS-7513

2.6.2 TRENDnet Indoor/Outdoor (TV-IP312PI)

Lens

- Focal length: 4 mm
- Focal depth: 20 cm+
- Aperture: F2.0
- Board lens
- Sensor: 1/3" progressive scan CMOS

Viewing Angle

- Horizontal: 77°
- Vertical: 42°
- Diagonal: 90°

Zoom

- User-defined digital zoom

Minimum Illumination

- IR off: 0.19 lux
- IR on: 0 lux
- 50 meter IR illumination distance
- Smart IR reduces close object overexposure

Video

- D-WDR: 0-100 scale
- 3D Digital Noise Reduction (DNR)
- Shutter speed: 1/3 - 1/100,000
- H.264: 2048 x 1536 up to 20 fps
- MJPEG: 704 x 480 up to 30 fps

Hardware Standards

- IEEE 802.1X
- IEEE 802.3
- IEEE 802.3u
- IEEE 802.3x
- IEEE 802.3af

Device Interface

- 10/100 Mbps PoE port
- Power port (for non-PoE installations, power adapter sold separately (12VDC1A))
- Integrated adjustable mounting base
- LED indicator

Housing

- Weather rating: IP66
- Adjustable sun visor

Network Protocol

- IPv4, IPv6, UDP, TCP, ICMP, ONVIF v2.2, DHCP, NTP, DNS, DDNS, SMTP, FTP, SNMP (v1, v2c, v3), QoS
- NFS, SMB/CIFS
- HTTP, HTTPS
- PPPoE
- UPnP, RTSP, RTP, RTCP, SSL

Operating Temperature

- -30 - 60 °C (-22 - 140 °F)

Operating Humidity

- Max. 95% non-condensing

Certifications

- CE
- FCC
- UL 60950

Dimensions

- 104 x 104 x 243 mm (3.9 x 4.1 x 9.6 in.)

Weight

- 835 g (1.8 lbs.)

Power

- Input: PoE (802.3af)

- Consumption: 9 Watts max.
- Optional Power Supply (Sold separately)
- Output: 12 V DC 1 A
- 5.5 mm barrel connector
- TRENDnet power adapter, model 12VDC1A, sold separately

Management Interface

- Multi-language support: English, French, German, Russian, and Spanish
- IP address filter
- QoS traffic prioritization
- Time, date, and text overlay
- Image settings: brightness, contrast, saturation, sharpness, smart IR, exposure time (1/3 – 1/100,000), video standard, day/night switch, sensitivity, switch time, mirror, D-WDR, white balance, digital noise reduction
- D-WDR enhances video quality in high contrast daytime lighting
- 3D Digital Noise Reduction enhances night vision quality
- Scheduled recording: continuous and motion detection
- Video storage: to computer, NAS, CIFS/SAMBA share or through software
- Motion detection fields: define custom motion detection areas, motion sensitivity, and dynamic motion analysis
- Privacy masks: define custom privacy mask areas
- Tamper detection: email notification if the viewing field darkens suddenly
- Video playback: advanced playback functionality with visual timeline displaying detected motion and scheduled recordings
- Alert messages: storage full, storage error, and illegal login
- Snapshot: real time snapshot, motion detection with schedule, video tamper detection with schedule
- Supported dynamic DNS services: Dyn.com and NO-IP.org
- Management Setting: maximum 32 user accounts
- Supports remote management
- Storage logs: Alarm, Exception, Operation, and others
- Compatibility: Internet Explorer® 9.0 or higher, Firefox® 13.0 or higher, Safari® 4.0 or higher, Chrome™ 24.0 or higher

Price

110€

This is another surveillance IP camera and like the first, it records in high resolution and also has an infrared filter that can be turned on and off. But also has the size and compatibility issues, however the price is a lot lower and actually reasonable.

TRENDnet Indoor/Outdoor (TV-IP312PI)

2.6.3 Raspberry Pi Camera Module V2 - 8 Megapixel,1080p

Number of Channels	1
Supported Bus Interfaces	CSI-2
Maximum Supported Resolution	3280 x 2464
Maximum Frame Rate Capture	30fps
Dimensions	23.86 x 25 x 9mm
Length	23.86mm
Width	25mm
Height	9mm
Connection	15cm ribbon cable for CSI port
Maximum Operating Temperature	+60°C
Minimum Operating Temperature	-20°C

Price

21€

This camera module is made specifically for the raspberry Pi motherboard. Provides high definition images/footage. With it's really tiny size it can fit anywhere, it has practically no weight at all. Camera accessible using libraries, e.g. Picamera. Does not have infrared vision.

Raspberry Pi Camera Module V2

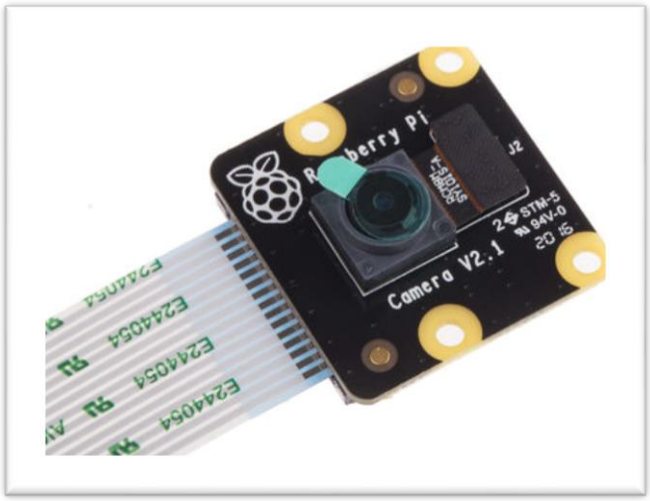
2.6.4 Raspberry Pi PiNoir Camera V2 Video Module

Number of Channels	1
Supported Bus Interfaces	CSI-2
Maximum Supported Resolution	3280 x 2464
Maximum Frame Rate Capture	30fps
Dimensions	23.86 x 25 x 9mm
Length	23.86mm
Width	25mm
Height	9mm
Connection	15cm ribbon cable for CSI port
Maximum Operating Temperature	+60°C
Minimum Operating Temperature	-20°C

Price 21€

The specifications for this camera are the same as the one before. The only difference between these two camera modules is the absence of the infra-red filter in the lens. To work in

the night, this module would need an infrared illuminator, however i think for our purposes it is not needed.



Raspberry Pi PiNoir Camera V2 Video Module

2.6.5 D8M-GPIO Terasic

- Package Interface: 2x20 GPIO with 3.3V I/O standard
- MIPI Camera Module:
    - Chip P/N: OV8865
    - Color Filter Arrangement: Bayer Pattern
    - View Angel: 70 degrees
    - Lens Type: 1/3.2 inch
    - Pixels: 3264x2448 (8-megapixels)
    - Frame Rate: Maximal 60 frame per second at 1408 x 792 resolution and 30 frame per second at 3268 x 2448 resolution
    - Support Focus Control
    - Interface: MIPI
- MIPI Decoder:
    - Chip P/N: TC358748XBG
    - MIPI CSI-2 Compliant
    - MIPI to Parallel Port Converter
    - Supports up to 4 data lanes
- Package Size: 73.4x60.0 mm

D8M-GPIO Terasic

Price 99$

This camera module is suitable for our project thanks to its direct compatibility with the Altera boards. It connects directly to the FPGA board via the GPIO interface. Its price is reasonable and properties are meeting our demands.

2.7 Conclusion from sensor analysis

In our project, we use Arduino Uno and Raspberry Pi boards. The reason we decided to use these boards is that they support every sensor we use; it is easy to write programs in their environment. Also, compatibility with sensors is very good.

From the GPS sensors, we selected PMOD GPS Receiver. We have chosen this module because it can be used with both Altera and Arduino boards. Compatibility with Altera is for future implementation of real time TT frames in FPGA. Our implementation will use this GPS receiver on Arduino UNO.

As a supplement to GPS receiver PMOD 3-axis digital compass was chosen to determine the car heading. This compass can also be used with Altera and Arduino boards for the same reason as in GPS.

As for the cameras we chose the Raspberry Pi PiNoir Camera V2 Video Module for its low price, compatibility with the Raspberry Pi and sufficient properties and the D8M-GPIO Terasic for its compatibility with the Altera boards.

Every device placed in an autonomous car needs a power supply. We decided to use 7.2V LiPo batteries for powering up four Arduino boards located near wheels. These LiPo batteries also power up electronic speed controllers and motors which are used to move the car. For other devices we decided to use 12V motorcycle Pb battery. Advantage of this battery is its large capacity. This 12V battery is used for powering up all four switches and every other device located in the car.

After discussion with all of us, we realized that none of the analyzed car platforms are suitable for our project. They have either too small dimension, or the custom configuration will bring a lot of problems. Therefore, we have decided to build our own car platform. It means we must buy all parts (wheels, motors, chassis) independently and put it together.

For lack of time, it is easier if our solution will not provide steering by turning front wheels. Changing direction will be allowed by moving wheels on one side (the wheels on the second side can move in opposite way).

The chassis should have dimension around 800x600 mm. Our member Marek has experiences with welding, so he can construct chassis with our requirements.

2.8 DE-Hermes Switch 3-1 BRR

The DE-Switch Hermes 3-1 BRR is a combined switch ECU that is designed for application development and evaluation of Deterministic Ethernet for in-vehicle network architectures considering multiple communication standards, including:

- Audio-Video Bridging (AVB),
- Time-Sensitive Networking (TSN), and
- Time-Triggered Ethernet in combination with a BroadR-Reach® physical layer.

Deterministic Ethernet enables the convergence of critical and non-critical application data streams on one network. The DE-Switch Hermes 3-1 BRR enables the evaluation of in-vehicle network requirements for diagnostics, control applications, infotainment and advanced driver assistance systems (ADAS).

It can be deployed to show the full potential for the next generation of Ethernet-based domain architectures using Deterministic Ethernet.

Specifications:

Dimensions (L x W x H):	146.6 x 92 x 38 mm
Weight:	315 g (with housing)
	400 g (weight of cable harness)
Power Supply:	Nominal: 12 V / 24 V
	Absolute maximum ratings: 6 to 36 V

Fields of Application:

- Automotive
- Buses and trucks
- Farming and off-highway

The DE-Switch Hermes 3-1 BRR

External Interfaces

The DE-Switch Hermes 3-1 BRR has

- **3** BroadR-Reach® physical layer interfaces that enable **100 Mbit/s full-duplex** communication over unshielded twisted single pair (UTSP) cabling,
- **one 1-Gbit/s** Ethernet port (100/1000Base-Tx),
- **1** RS-232 serial interface

The DE-Switch Hermes 3-1 BRR has the following standard interfaces, such as CAN and FlexRay™, and digital and analog I/Os for customized evaluation projects.

- **3** CAN interfaces (125 kbit/s up to 1 Mbit/s),
- **1** FlexRay™ interface (channel A and B),
- **4** analog inputs (0 to 5 V or 4 to 20 mA, 0 to 10 V provided by DE-Switch Hermes 3-1 BRR),
- **2** digital timer inputs (0.1 Hz to 20 kHz),
- **4** digital high-side PWM outputs:
    - 3 A permanent
    - 4 A peak
    - 5 A overall maximum

Standards Compliance

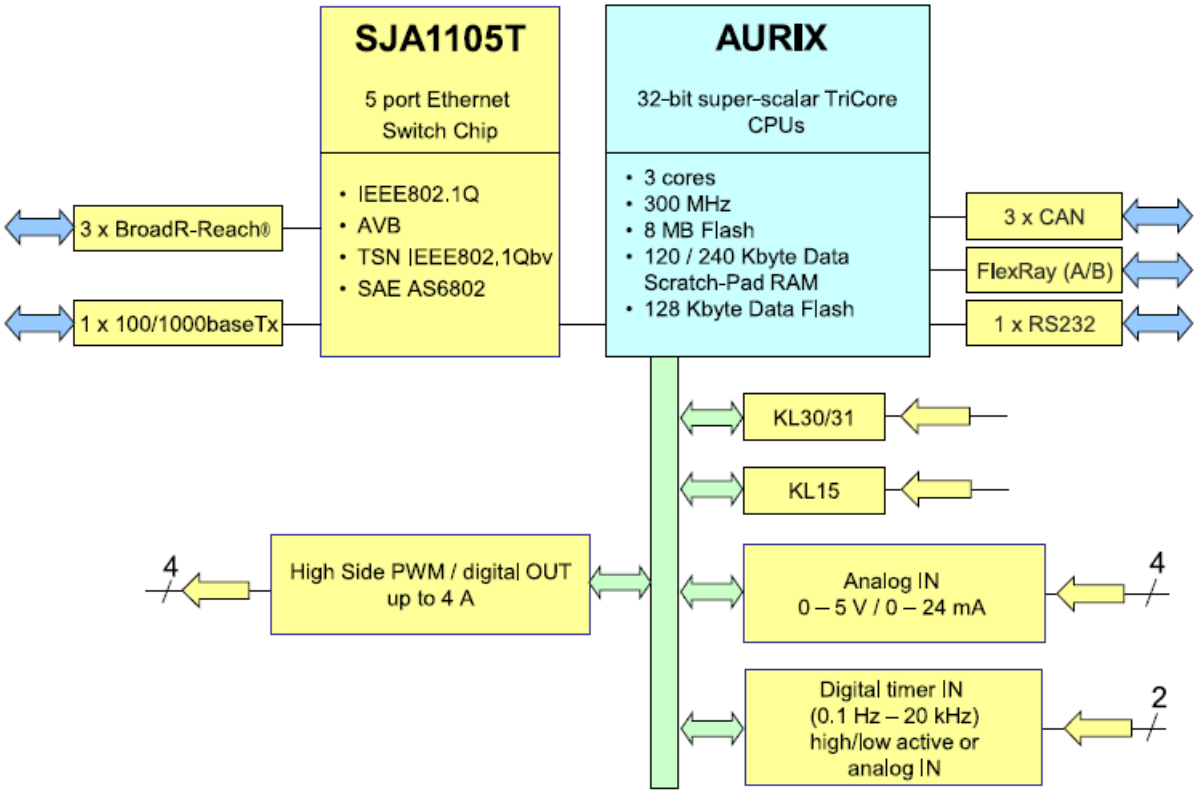
- **IEEE 802.1D™**-2004 (layer 2 switching)
- **IEEE 802.1Q™**-2011 (VLAN support)
- QoS handling based on IEEE 802.1Q PCP bits
- Support for SR Class A, Class B and Class C traffic
- **IEEE 802.1AS™**-2004 (Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks)
- **IEEE 802.1Qbv™**-2015 (Enhancements for Scheduled Traffic)
- **SAE AS6802** (Time-Triggered Ethernet)
- The switch forwards **best-effort traffic** in compliance with IEEE 802.3-2005 (switching).

- The switch forwards **VLAN-tagged frames** according to IEEE 802.1Q (VLAN core capabilities).

Functional Description

- The DE-Switch Hermes 3-1 BRR provides Ethernet for in-vehicle network architectures and implements network switching functionality that is implemented on the NXP SJA1105T automotive Ethernet switch.
- The DE-Switch Hermes 3-1 BRR has 3 x BroadR-Reach® physical layer interfaces that enable 100 Mbit/s full-duplex communication over unshielded twisted pair cabling in addition to one 100/1000Base-Tx port.
- A management CPU is connected with the Ethernet switch via a 100 Mbit Ethernet interface and an SPI configuration interface. The management CPU runs the switch management protocols (for RSTP and IEEE802.1AS). For customized evaluation projects, external interfaces, such as CAN, FlexRay™, it is possible to use analog and digital I/Os.

Following figure with block diagram gives an overview of the main features of the DE-Switch Hermes 3-1 BRR:



Block diagram of the DE-Switch Hermes 3-1 BRR

Primary Components

- **Ethernet Switch**: The Ethernet Switch is an automotive-compliant 5-port Ethernet switch. The device contains a variety of cross-wire media-independent interfaces to connect any kind of physical layer. The control interface is a serial peripheral interface (SPI), which is necessary to read and write internal registers of the switch chip. Four ports are connected via physical interfaces to the ECU connector, and one port is connected to the management CPU.
- **Management CPU**: The management CPU covers all the control and monitoring features of the system. The CPU also loads and stores the configuration for the Ethernet Switch. The device is responsible for the correct setting of the peripherals, which includes the configuration of the switch and the physical layer and the control of the digital and analog I/Os and communication interfaces.
- **100base-T1 BroadR-Reach® Physical Layer**: The physical layer is an OPEN Alliance BroadRReach ®-compliant Ethernet physical layer that is optimized for automotive use cases. The device provides 100 Mbit/s transmit and 100 Mbit/s receive capability over a single unshielded twisted single pair (UTSP) cable, supporting a cable length of at least up to 15 m. The system has three physical layers. The MII of each physical layer is connected to the Ethernet Switch, whereas the medium-dependent interface (MDI) is connected to the ECU connector via an analog front end.
- **Gigabit Physical Layer**: A Gigabit Ethernet transceiver implements the Ethernet physical layer portion of the 100BASE-TX and 1000BASE-T standards. The reduced Gigabit mediaindependent interface (RGMII) is connected to the Ethernet Switch. As the PCB does not have a standard RJ-45 connector, the MDI is connected to the ECU connector via dedicated magnetics HX5008NL.
- **Power supply and reverse polarity protection**: The power supply and reverse polarity protection block contain all the parts that are necessary to provide proper supply voltages for the board electronics. The input voltage range, which is between 6 V and 36 V is protected against reverse polarity. The nominal voltage range is 12 V or 24 V.
- **Communication Interfaces**: The DE-Switch Hermes 3-1 BRR provides additional communication interfaces beside the Ethernet functionality:

    - An RS-232 interface is also connected to the ECU connector as a user interface and for debug purposes.
    - CAN and FlexRay™ interfaces are implemented.
- **Digital and Analog I/O**: The DE-Switch Hermes 3-1 BRR has several control and monitoring features to combine network-control functionality with electronic control functionality in one ECU.
- **4 high-side PWM output** stages up to 3 A with current measurement and digital feedback provide availability to control relays and engines. A summed current of 5 A at the same time is the limit.
- **4 analog inputs** and **2 digital timer** inputs can be used for different sensor applications.

Ethernet ports

The Ethernet Switch has 5 independent ports. The ports are configured as follows:

Port	Description
0	1 Gbit/s Ethernet port (100/1000Base-Tx)
1	BroadR-Reach® channel 0 (100 Mbit/s full-duplex communication over UTSP cabling)
2	BroadR-Reach® channel 1 (100 Mbit/s full-duplex communication over UTSP cabling)
3	BroadR-Reach® channel 2 (100 Mbit/s full-duplex communication over UTSP cabling)
4	100 Mbit/s MAC-MAC interface to management CPU

We are going to use four Hermes switches in our application. Each switch will be connected to one wheel. Switches will be connected to each other in circle topology.

3. SOLUTION DESIGN

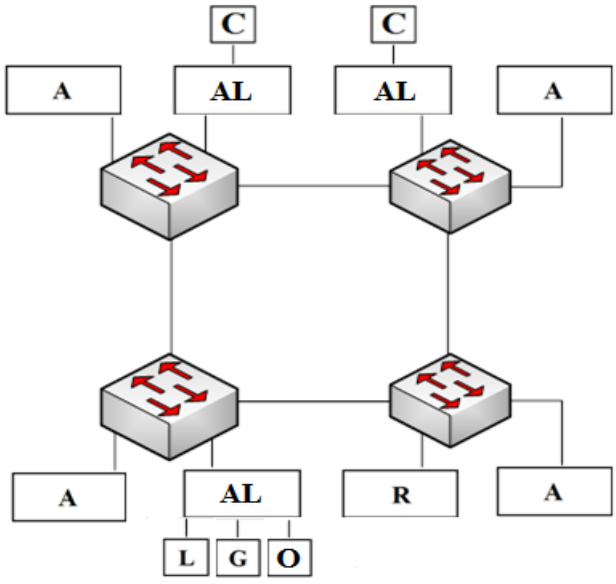
3.1 Logical design

Logical design of car is shown on picture. We will use all four available deterministic Hermes switches, four Arduino Uno boards (A), one Raspberry Pi 3 model B board (R), three Altera boards (AL), two cameras (C), laser (L), GPS sensor (G) and digital compass (O).

Each of the Hermes switches has four available Ethernet ports. Redundancy is achieved with ring topology, so in case of one link failure, other switches will still be able to communicate. This topology is not resistant against two or more links failures. Full mesh topology would solve this problem but we are limited by the count of physical ports on the switches.

Each Arduino board will control one wheel, but to connect them to the switches, Ethernet shields are required, because Arduinos do not have Ethernet port by default. Raspberry Pi will send signals to control the speed of each wheel.

Raspberry Pi will act as central point and other boards with sensors will send control information to it. On the front of the car, two cameras will be used. One will handle line assist feature, and the other one will handle road signs recognition. On the top of the model will be 360° degree laser sensor for detections of the obstacles. For car navigation, we will use GPS sensor. Navigation with GPS is described in section 3.3.



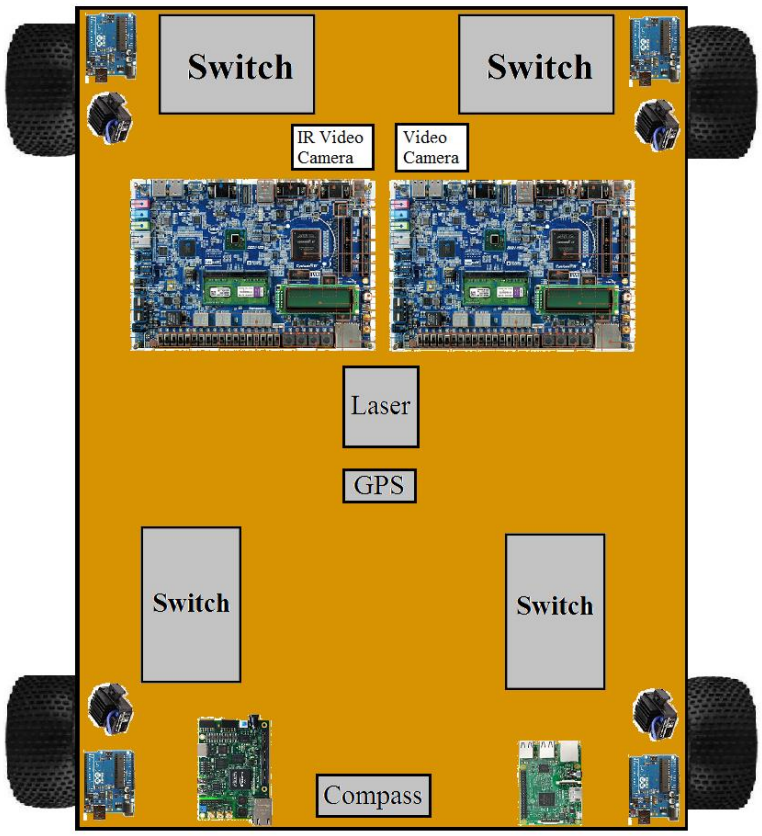
Logical design of car model

3.2 Physical design

Physical design of the car is shown in the picture. There are four wheels which were bought in RC shop. These wheels are from large RC buggy so larger payload on this car is not a problem for them. In every edge of the car an Arduino and regulator are located. Every Arduino controls one motor through a regulator. Therefore, we can achieve smooth speed regulation and reverse drive as well.

Large more powerful Altera boards are located in front part of the car. The first Altera board on the left controls IR video camera for line checking and the second Altera board on the right controls normal video camera for road signs recognition.

Laser is located on the top, in the middle of the car. It is because laser must not have any obstacle in its field of view and it is easier to compute a distance from obstacle if laser is located in the middle of the car. GPS sensor has to be in the middle of the car as well because the car is a larger one so GPS positions in the edges of the car may be different. A compass is located in the back part of the car. There are also located a smaller Altera board and a master raspberry pi board. The batteries in the picture are not shown. They will be placed to suitable position later in this project.



Physical design of the car

3.3 Communication protocol

Communication is based on UDP protocol. The reason why preferring UDP to TCP is that we do not require reliable packet delivery because of „real time“ application. On the contrary, we require to deliver packet fast and without delay to meet requirements of real time. Also, UDP protocol supports broadcast, which can be useful. The communication protocol is shown on screen below.

TYPE	SOURCE BOARD	NUMBER OF SOURCE BOARD	DESTINATION BOARD	NUMBER OF DESTINATION BOARD	TYPE OF MESSAGE	DATA
1B	1B	1B	1B	1B	2B	

Communication protocol

3.3.1 Type field

Field called TYPE is used to distinguish if packet is send from central unit or to central unit. It has size of 1B.

TYPE	message from central unit	message to central unit
IDENTIFICATOR	0x00	0x01

Type field

3.3.2 Source board field

Source board field is used to distinguish which board send packet. Value 0 stands for Arduino, 1 stands for Raspberry and 2 stands for Altera.

SOURCE BOARD	arduino	raspberry	altera
IDENTIFICATOR	0x00	0x01	0x02

Source board field

3.3.3 Number of source board

The field is set to value of source board that sends packet. So far, we do not have more than four boards so the field can obtain values from 1 to 4.

NUMBER OF SOURCE BOARD	0x01	0x02	0x03	0x04

Number of source board field

3.3.4 Destination board field

Destination board field is similar to source board field. It identifies destination board which packet is send to. Value 0 stands for Arduino, 1 stands for Raspberry and 2 stands for Altera.

DESTINATION BOARD	arduino	raspberry	altera
IDENTIFICATOR	0x00	0x01	0x02

Destination board field

3.3.5 Number of destination board field

This field is similar to field called number of source board field. The field is set to value of destination board that packet is send to. The field can obtain values from 1 to 4.

NUMBER OF DESTINATION BOARD	0x01	0x02	0x03	0x04

Number of destination board field

3.3.6 Type of message field

Message can have different meaning. Value 0 represents notifying of IP address. This type of message is send when IP address is assigned to inform central board. Value 1 stands for instruction from central board to others. Value 2 defines message which servers for acknowledgment of received instruction. Value 3 defines data from infrared camera. Value 4 stands for road side camera and value 5 represents laser data. Value 6 is for data from GPS.

TYPE OF MESSAGE	notifying IP address	instruction	instruction ACK	infrared camera data	road side camera data	laser data	GPS
IDENTIFICATOR	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006

Type of message field

3.3.7 Laser data message

When laser data are sent, message is as show on screen no 8. First field represents number of available ranges. Next bytes define specific range of angles.

number of ranges (1B)	range #1 (8B)				range #2 (8B)				range #n (8B)			
	start angle (2B)	start distance (2B)	end angle (2B)	end distance (2B)	start angle (2B)	start distance (2B)	end angle (2B)	end distance (2B)	start angle (2B)	start distance (2B)	end angle (2B)	end distance (2B)
n	α	x	β	y	α	x	β	y	α	x	β	y

Laser data message

3.3.8 Road side camera data message

Data from road side camera are sent in format shown on screen no. 9.

Number of road signs	Road sign x1	Road sign x2	Road sign xn
n	#1	#2	#n

Road side camera data message

3.3.9 Infrared camera data message

Data from infrared camera are sent in format shown on screen no. 10.

Angle of direction change
x

Infrared camera data message

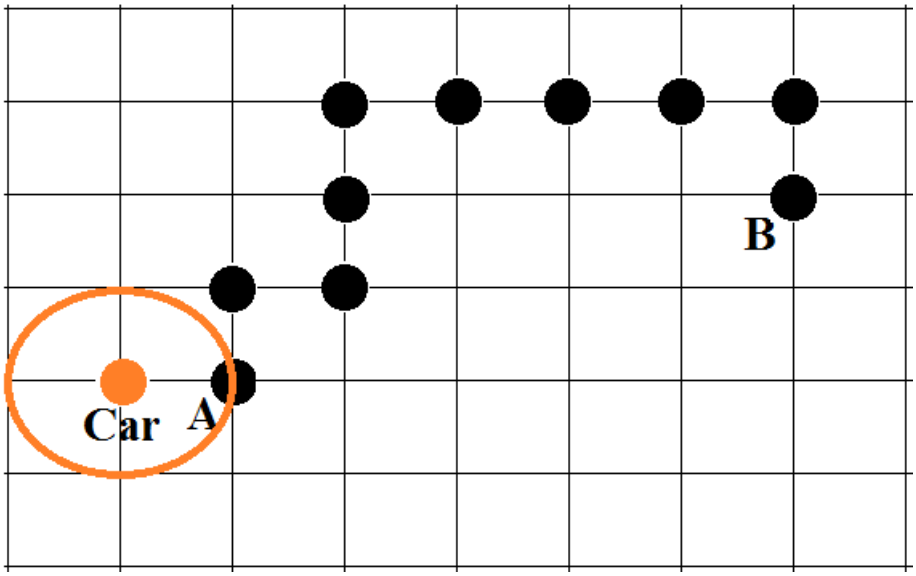
3.4 Navigation with GPS

GPS receiver and compass will be used on Arduino UNO board. At first we must determine vehicle position. GPS generates NMEA output sentences which contain all measured GPS information. These NMEA sentences are GGA, GSA, GSV, RMC and VTG. Our solution will use GGA sentence which contain information about latitude and longitude. Format of GGA sentence: *$GPGGA,064951.000,2307.1256,N,12016.4438,E,1,8,0.95,39.9,M,17.8,M,,*65* is shown in the table below.

Format of GGA NMEA sentence

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	064951.000		hhmmss.sss
Latitude	2307.1256		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12016.4438		dddmm.mmmm
E/W Indicator	E		E=east or W=west
Position Fix Indicator	1		0 - Fix not available 1 - GPS fix 2 - Differential GPS fix
Satellites Used	8		Range 0 to 14
HDOP	0.95		Horizontal Dilution of Precision
MSL Altitude	39.9	meters	Antenna Altitude above/below mean-sea-level
Units	M	meters	Units of antenna altitude
Geoidal Separation	17.8	meters	
Units	M	meters	Units of geoids separation
Age of Diff. Corr		second	Null fields when DGPS is not used
Checksum	*65		
<CR> <LF>			End of message termination

GPS navigation will collaborate with digital compass which returns 3 axis values. These values are used to compute the compass heading degree. Assume simplified version of map shown on picture below. Black dots represent nodes of optimal path from node A to node B.



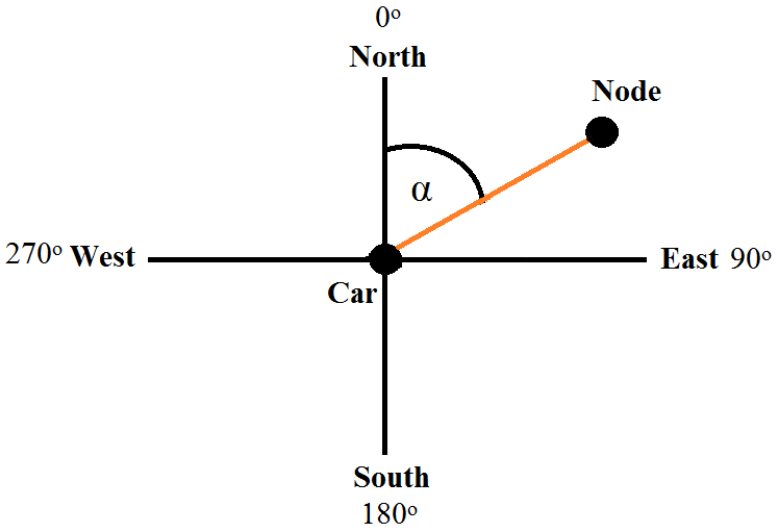
Simplified navigation map

Before navigation, a path from current position to defined destination must be known. This path will be hardcoded in first versions of navigation. For the future implementations tools for finding optimal path (for example based on OpenStreet maps) can be used to compute the path.

Data processing

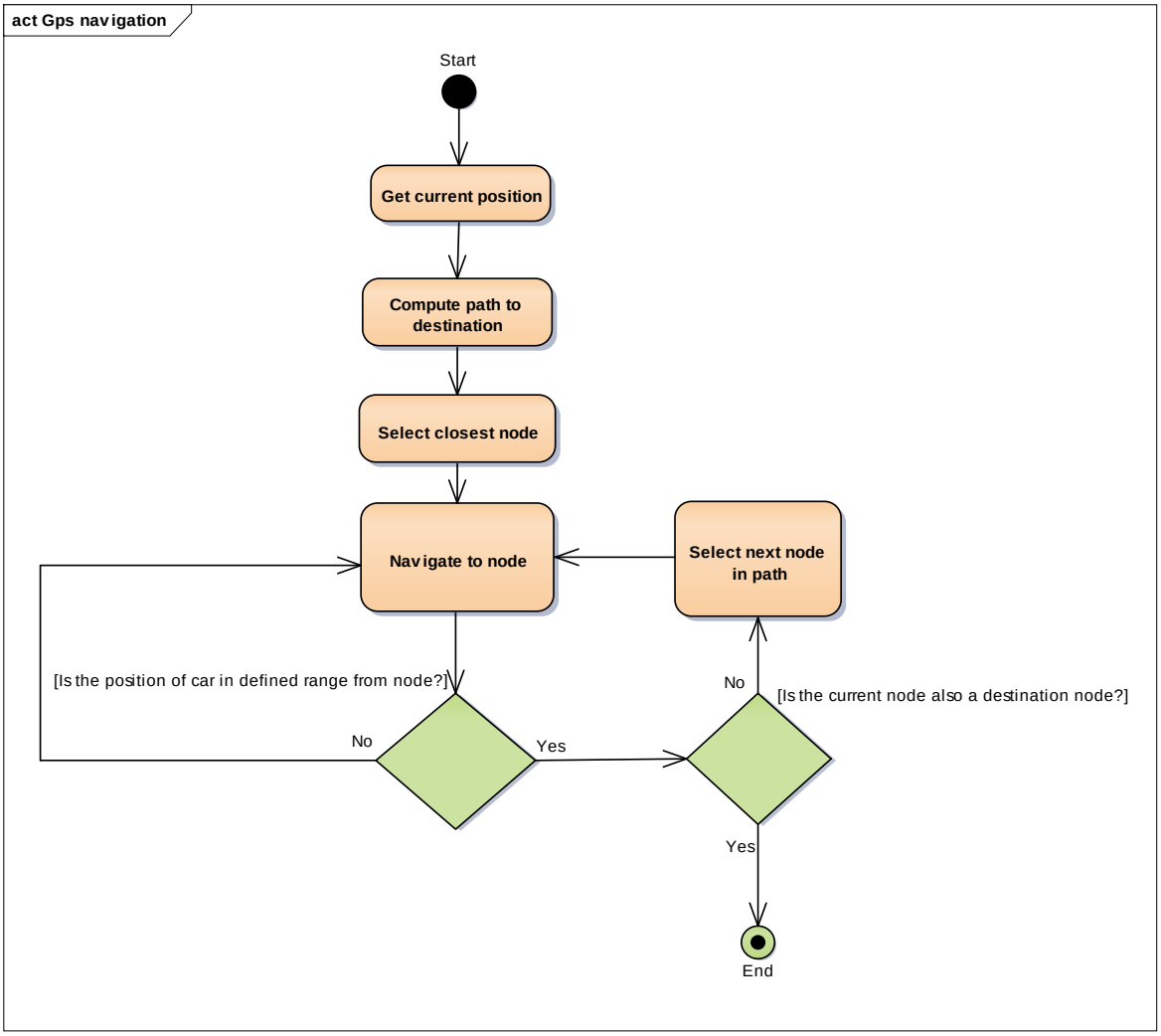
After the path is known, the navigation process begins. Every place in the world can be defined with latitude and longitude. The precision of GPS sensor is around 5 meters, therefore we can only approximately tell where our vehicle is. Defined path will be represented by nodes on the real road. Distance between these nodes must be greater than GPS precision, so we can navigate from one node to another.

Car first needs to determine the closest node of the path. Then the degree between actual position and closest node position will be calculated (see picture below). This degree is then transformed into relative degree. Relative degree is computed also with compass data. Direction of car heading is always represented as 0, so the computed degree from GPS must be transformed according do this heading so the vehicle know which side it should turn. This calculated relative degree is sent to central processing unit (Raspberry Pi), which will control the movement of the car to the selected node based on other signals from other sensors. As mention earlier, we cannot accurately tell, if the car is in the selected node. We will consider node reached, when the car will be in the defined range from the selected node.



Calculation of navigation degree

After car reaches the node, then the next node from the path is selected and the navigation process begins from start. Whole process of navigation is shown on diagram below.



Activity diagram of GPS navigation

Communication Protocol

Data field of the designed communication protocol will contain 2 additional bytes for sending computed relative degree. Two bytes are used because of theoretical maximum value of 360 which cannot be represented with 1 byte. For example, value 261 will be represented as below.

Example of data payload from GPS and compass

# relative degree byte 1	# relative degree byte 2
0x01	0x05

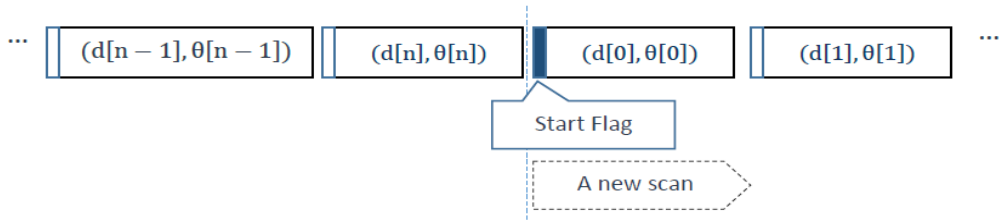
3.5 Laser data processing

RPLIDAR A2 can take up to 4000 samples of laser ranging per second with high rotation speed. The range scan data can be processed via the communication interface of the RPLIDAR and control the start, stop and rotating speed of the rotate motor via PWM.

During the working process, the RPLIDAR will output the sampling data via the communication interface. And each sample point data contains the information in the following table.

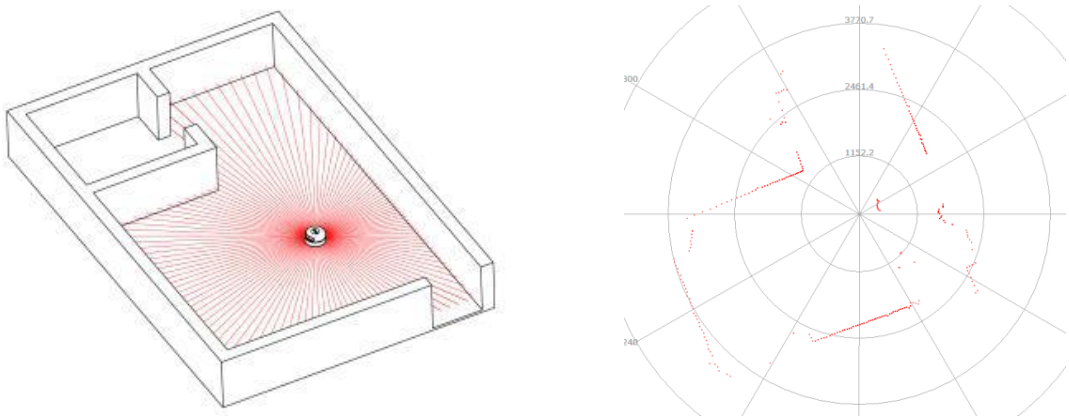
The RPLIDAR Sample Point Data Information

Data type	Unit	Description
Distance	mm	Current measured distance value between the rotating core of the RPLIDAR and the sampling point
Heading	degree	Current heading angle of the measurement
Start Flag	(Bool)	Flag of a new scan
Checksum		The Checksum of RPLIDAR return data



The RPLIDAR Sample Point Data Frames

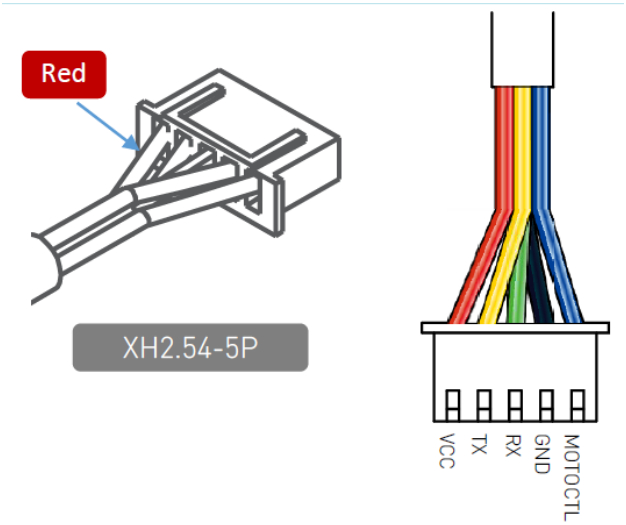
The RPLIDAR outputs sampling data continuously and it contains the sample point data frames in the above figure. Host systems can configure output format and stop RPLIDAR by sending stop command.



The Obtained Environment Map from RPLIDAR Scanning

Communication interface

The RPLIDAR A2 uses separate 5V DC power for powering the range scanner core and the motor system. And the standard RPLIDAR A2 uses XH2.54-5P male socket. Detailed interface definition is shown in the following figure:



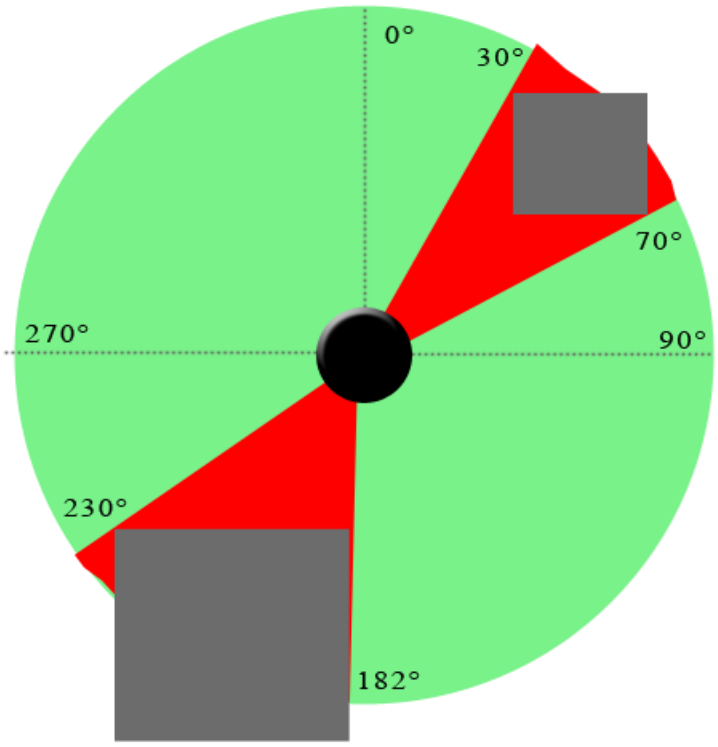
RPLIDAR Power Interface Definition

Color	Signal name	Type	Description	Min	Typical	Max
Red	VCC	Power	Total Power	4.9V	5V	5.5V
Yellow	TX	Output	Serial port output of the scanner core	0V	3.3V	3.5V
Green	RX	Input	Serial port input of the scanner core	0V	3.3V	3.5V
Black	GND	Power	GND	0V	0V	0V
Blue	MOTOCTL	Input	Scan motor /PWM Control Signal (active high, internal pull down)	0V	3.3V	5V

RPLIDAR External Interface Signal Definition

Communication Protocol

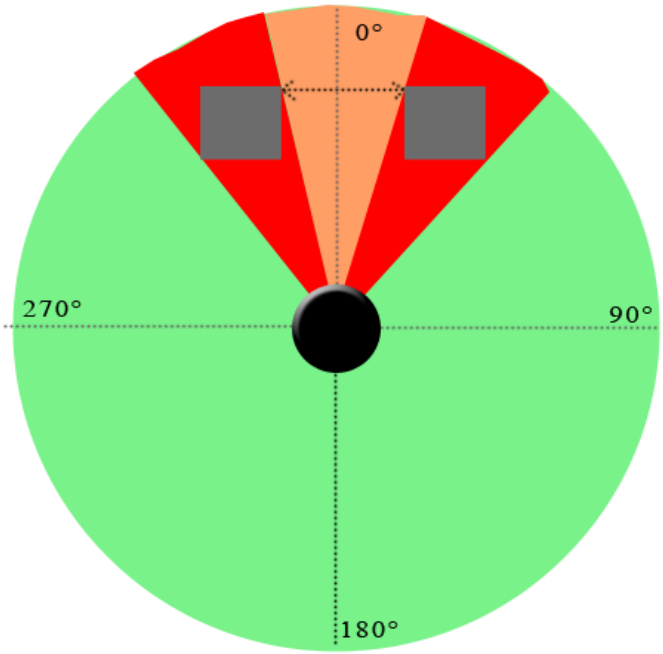
After receiving data from sensor, these data are processing, counting and evaluating for next processing. All counting is performed in microcontroller responsible for communication with laser sensor. This microcontroller sends data to control unit with ranges of angles, where is enough space for passing through this area. With this information, control unit can decide which side to choose in next move. This principle is shown in next figure:



Detecting obstacles around the laser sensor

The microcontroller sends only ranges with enough distance to pass through. In figure above, there are only two ranges with enough space.

For smaller range of angles, there is also necessary to count, if car robot can fit into this space.



Exclusion of angles with short distance between obstacles

The frame structure with open ranges is shown in next figure:

# ranges of ranges	range #1		range #2		range #3		...	
	start angle	end angle	start angle	end angle	start angle	end angle	start angle	end angle
n	x_1	x_2	x_3	x_4	x_5	x_6		

Frame structure

The worst-case scenario is with 180 open ranges. It means, that every second angle meets the distance condition and there is enough space. And vice versa, every second angle does not meet the condition, and there isn't enough space. In this case, it is obvious that robot must stop, because in any direction it hasn't enough space.

So, for number of ranges (value *n*) must be reserved 1 byte. Start and end angle can have value in range from 0 to 360, so there must be reserved 2 bytes.

If there are less than 180 open ranges, the rest of ranges are filled with zeroes. In the example below, there are only 2 open ranges, so only range number 1 and 2 are filled with some values. Rest of them are filled with zeroes. Field number of ranges ensures that these fields are not evaluating.

If the first number is bigger than second one, it means there is overflow and range of angles exceeds zero angle.

# ranges of ranges	range #1		range #2		range #3		…	
	start angle	end angle	start angle	end angle	start angle	end angle	start angle	end angle
2	160	280	300	120	0	0	0	0

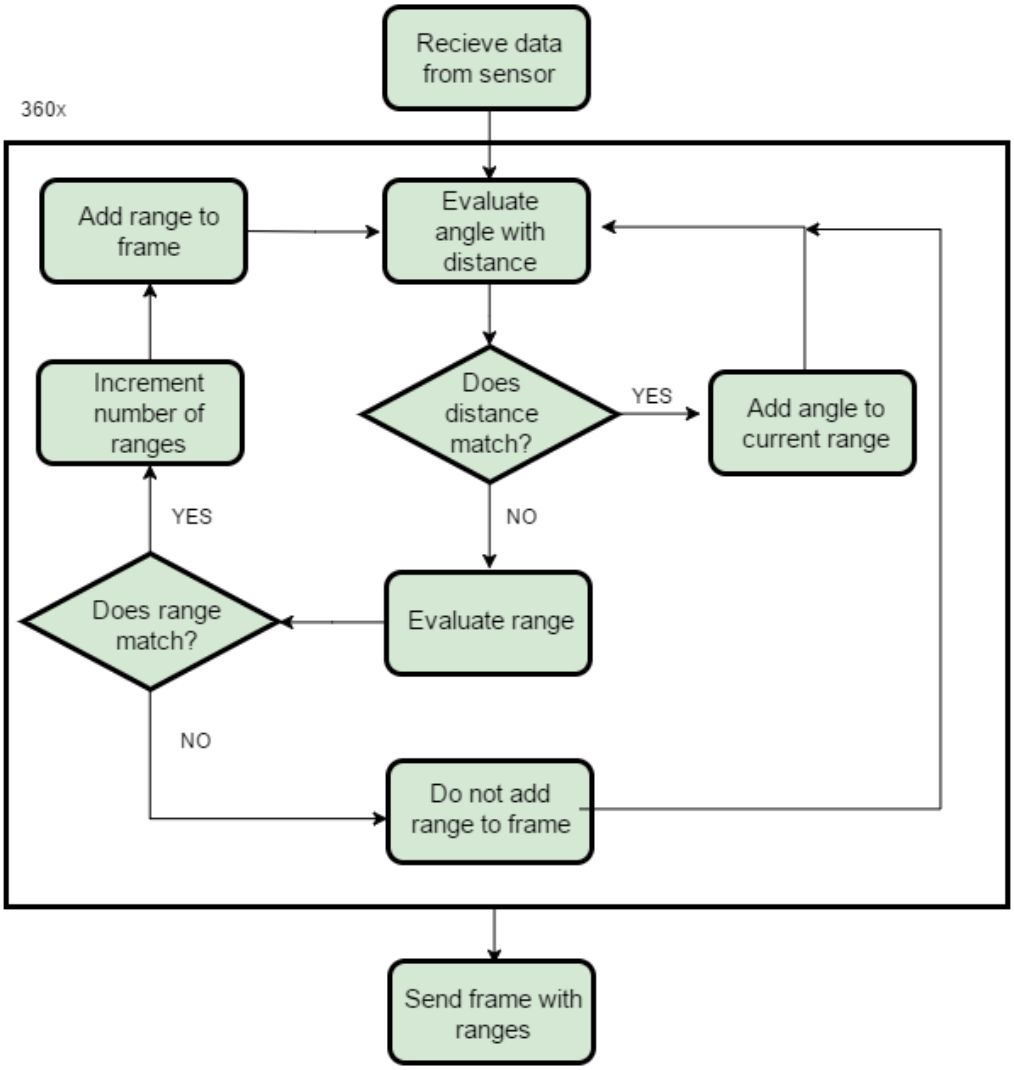
Example with two open ranges

Data processing

Diagram below illustrates data processing from receiving data to sending frame with open ranges.

After receiving data from sensor, each angle is evaluated by distance. If distance matches, it means the distance is less then certain limit distance, angle is added into open range. If not, range is closed. After that, range is evaluated, if there is enough space for passing through. It is counted with trigonometry functions. If range is enough big, it is added into frame with start and end angle.

When all angles are evaluated, the frame can be send to control unit. Since evaluating begins with zero angle, first and last range can be merged before sending.

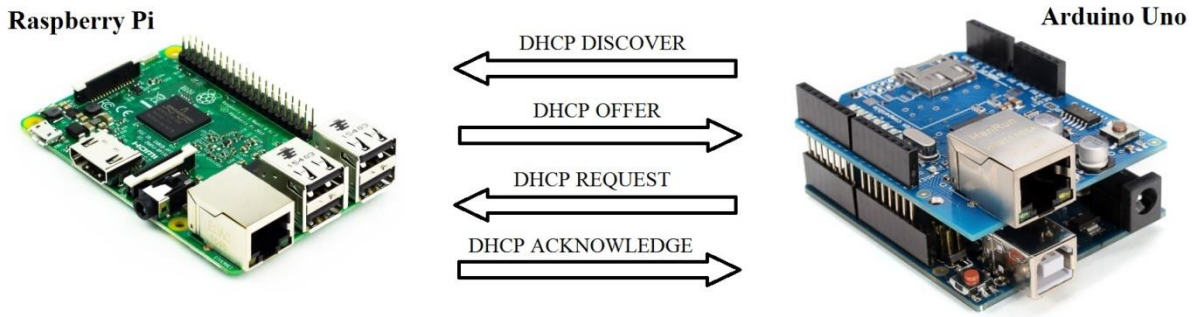


Data processing from laser sensor

3.6 Communication Raspberry Pi - Arduino

In this section a communication between Raspberry Pi and Arduino is described. In our autonomous vehicle, there are four Arduino devices. Every single device controls one motor which is connected to one wheel. After system startup, Arduinos have no IP addresses and they cannot communicate with master Raspberry.

Arduinos are therefore configured as DHCP clients and master Raspberry is configured as DHCP server. After system startup, every Arduino send DHCP DISCOVER message. Then master Raspberry sends DHCP OFFER message. Arduino sends DHCP REQUEST message and master Raspberry sends DHCP ACKNOWLEDGE message. Now every Arduino has its own IP address.



DHCP communication

After this process all Arduinos send initial message to Raspberry. In this message, Arduino tells its IP address and its number. This number is important because master Raspberry use it to recognize which Arduino controls which motor. For example, number 1 means that it is an Arduino which controls front right motor.

Now everything is ready and Raspberry can send instructions to Arduinos. In single instruction there is defined type of message which is in case of instruction ‘01‘ and after this code there is a number in range from 0 to 255 which represents new motor speed. Numbers from 127 to 0 are used to reverse the vehicle. Number 128 is used if we want our vehicle to stay at one place without movement. Last part of numbers from 129 to 255 are used for forward speed regulation.

If master Raspberry wants to send an instruction to Arduino, it uses UDP protocol where there are instruction numbers encoded. When Arduino receives an instruction, it executes this instruction (sets a new wheel speed) and after that it sends an acknowledge message via UDP.

This message is important for Raspberry to know. It can detect failure on Arduino. If battery runs out, Arduino does not send acknowledge messages. This is how Raspberry knows that something happened. If there is at least one Arduino which does not respond, Raspberry immediately stop the vehicle to prevent from any damage.

3.7 Camera data processing

3.7.1 Infrared camera

This camera is recording the road in front of the vehicle in order to navigate it on the road. The navigation is done by checking the roadside and informing about movement changes needed to keep in certain distance from it.

This is done by firstly marking the road, by expanding the area around a point in front of the camera according to its approximate color and then searching for an area to the right of the road that we can consider to be the roadside.

After that we keep an approximate distance from the roadside by calculating the angle in which the vehicle should move to increase / decrease the distance between them.

Road image

Floodfilled image

In this image we can search for the roadside thanks to the fact that the road is marked with a specific color. Everything that doesn't belong to the floodfill range to the right of the center-bottom of the image, can be considered as the roadside.

3.7.2 Communication protocol

As was mentioned above, the result of the image processing is an information about the direction change needed to improve the position of the vehicle. According to the rotation, the angle used is sent to the control unit, where the front of the vehicle is equal to 0°.

Frame structure

Angle of direction change
x

Where x belongs from –90° to 90°.

3.7.3 Road signs camera for future work

This camera is taking pictures of the road surroundings, namely the area to the right of the road, to record any road signs. The images should be in good quality to be able to perform a correct analysis. That might delay the framerate of the processing, however that is not a problem in this case. The motherboard to which the camera is connected then compares these

images with a stored list of road signs (image recognition). When it finds a match, it sends the recognized road sign as an **integer** to the device in charge of control. This also means that the control device has to have a custom list of these road signs (in number form) with special behavior for each of them.

Frame structure

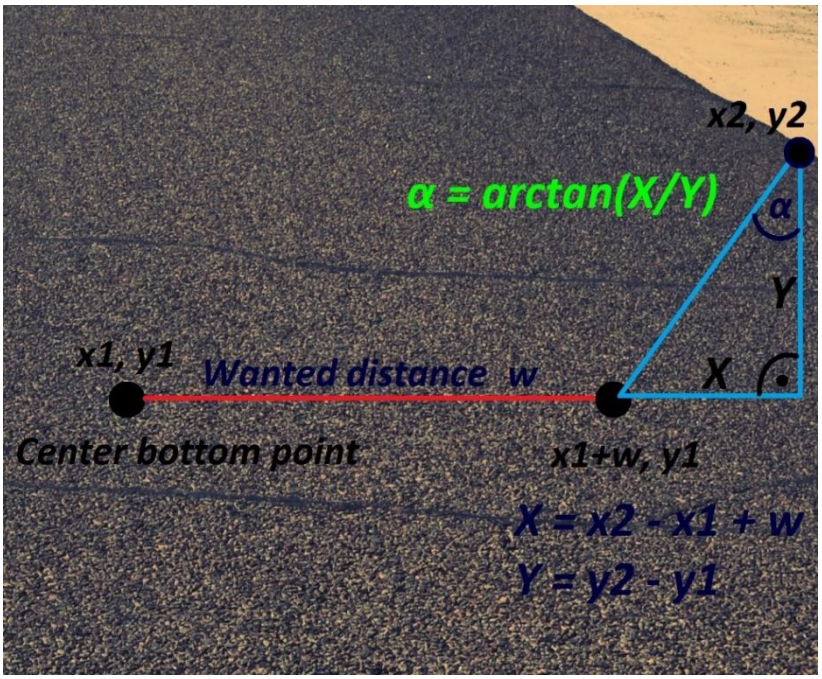
Number of road signs	Road sign x1	Road sign x2	Road sign xn
n	#1	#2	#n

3.8 Angle measuring

When it comes the angle measurement. We calculate the angle of movement change by finding the roadside to the right of the vehicle. When it's found we get the point coordinates for it and since we have the coordinates for our initial center bottom point with a preset distance we want to keep from the roadside (80px in our case).

Specifically, both points are given by their coordinates (location on horizontal line and location on vertical line). So, one point in 2D space is given by *x* (horizontal) and *y* (vertical).

As there are needed only two points, computation includes four values. These two points create right triangle. Thus, by using trigonometric functions it is possible to count angle between two points. In this case, it is used tangent equation. Since the angle must be calculated, inverse function is used (arcus tangent).



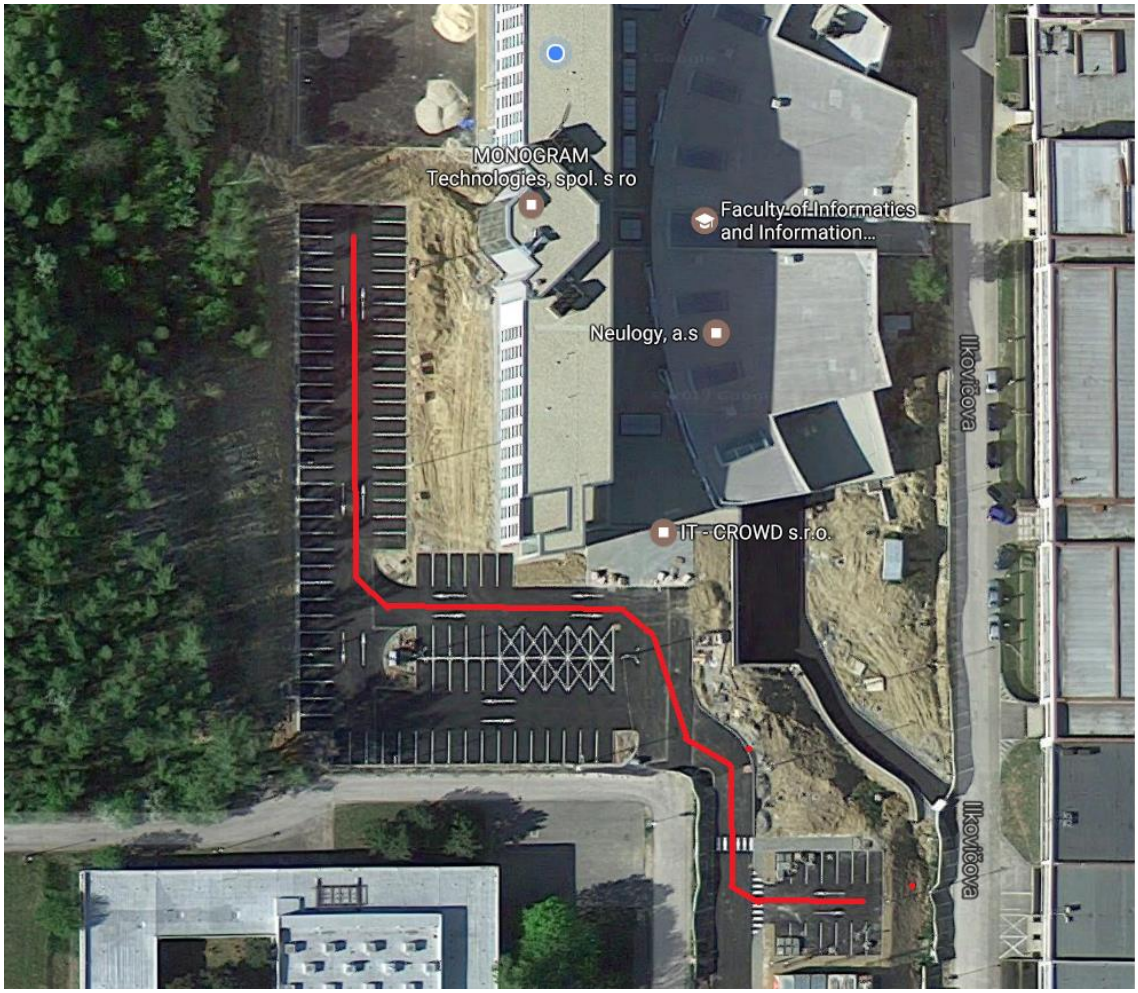
Principle of angle measuring

4. IMPLEMENTATION

4.1 GPS and compass navigation

4.1.1 General information

GPS and compass are implemented on Arduino UNO board. Program has hardcoded map that consists of 23 navigation points. Map is represented by arrays of structures. Structure represents one navigation point and consists of float latitude and longitude values. Created map is shown on the picture below.



Testing map for vehicle

To use the compass, hold the module so that Z is pointing 'up' and you can measure the heading with X and Y axis. These axes are labeled on the chip. Heading is then calculated using arctangents of X and Y axis.

Also, a correction for GPS degree must be calculated because GPS measures geographical position and degree is computed against geographical North Pole and compass measures degree to magnetic north pole. This difference between angles is called declination angle

which is the 'Error' of the magnetic field in our location. Value of this angle for specific location can be found online. It's value for Bratislava is 0.07504916 in radians.

Code is implemented in Arduino IDE. Tables below shows most important global variables and constants with their description and also most important functions with their description.

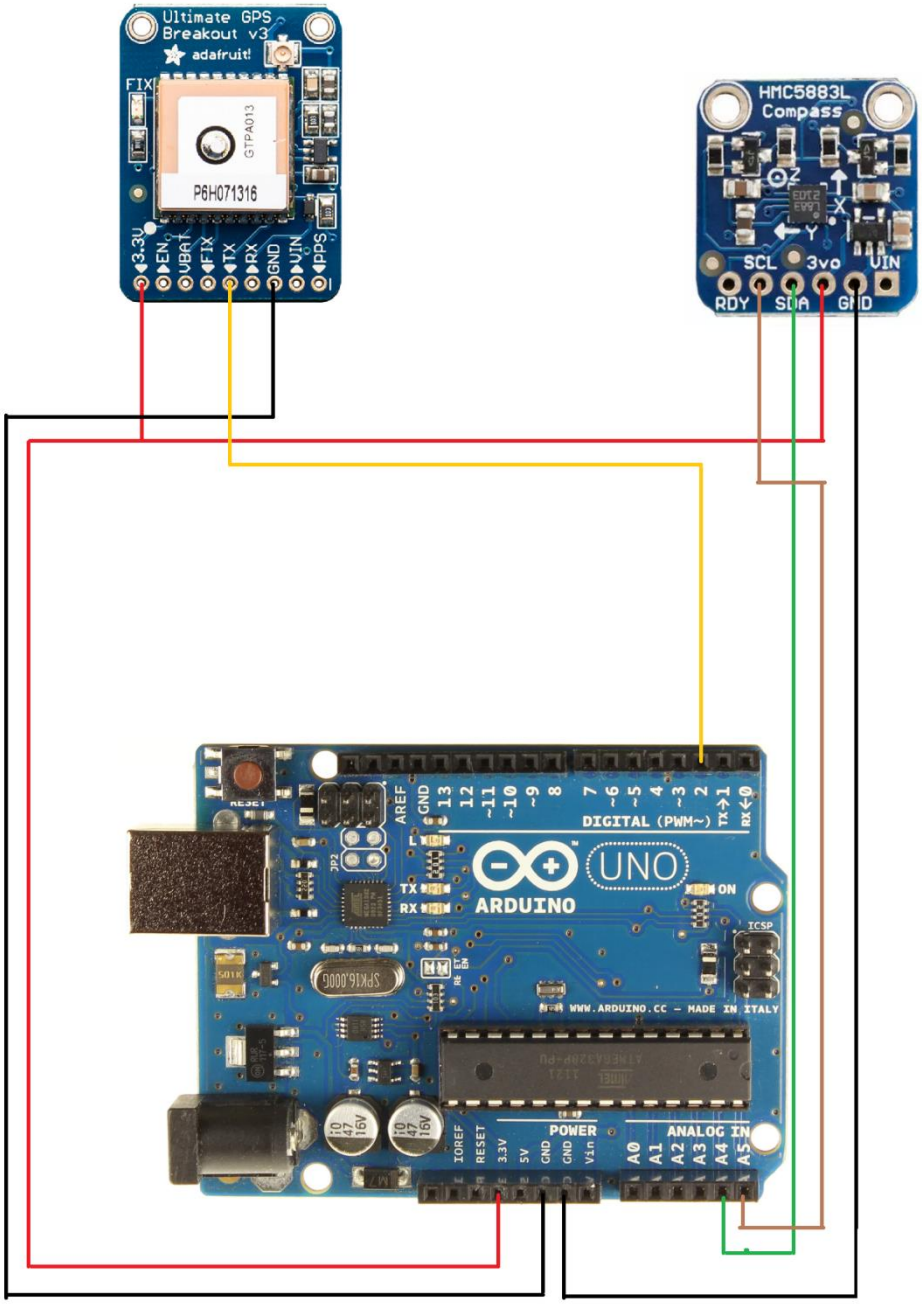
Most important global variables and constants in GPS navigation

Global variables and constants	Description
ACCURACY	Variation in which vehicle is considered in the point
MAP_LENGTH	Length of the navigation map
ARDUINO_NUMBER	Number of Arduino to use in protocol
MASTER_RPI_IP	IP address of control Raspberry PI
localPort	Port used for communication between Arduino and RPI
notifyMessage[]	Message sent for RPI when Arduino obtains new GPS and compass data and calculates relative angle (last two zeros will be changed)
nav_map[]	Navigation map
mac[]	MAC address of Arduino

Most important functions in GPS navigation

Function	Description
int find_closest_point(nav_point actual)	Function that finds closest point to actual position. Returns index of closest point in navigation map
int degree_based_on_quadrant(float x, float y, float degree)	Function for degree calculation based on quadrant
float calculate_relative_degree(float compass, float directionValue)	Function to calculate relative degree of next node, current compass degree is represented as zero on vehicle
float calculate_compass_degree(nav_point start,nav_point endPoint)	Function to calculate direction degree between two nodes
bool is_in_node(nav_point actual,nav_point dest)	Function to determine if actual node is in defined range of end node

4.1.2 Connection description



Wiring scheme for Arduino, compass and GPS sensor

In the picture above is wiring scheme for Arduino, compass and GPS sensor. Both GPS and compass are powered up by 3.3V from Arduino. There was a small problem because Arduino has only one 3.3V pin. Cutting a wire and duplicating it in the middle of the wire solved this issue. For ground there were enough pins on Arduino so it was simple to connect both devices to it.

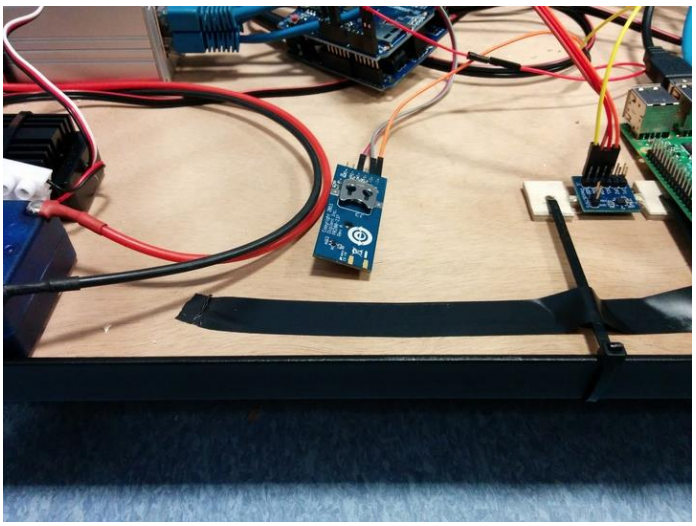
GPS sensor uses UART communication protocol to communicate with Arduino. Implementation in the picture above is quite specific. UART protocol uses TXD a RXD pins

to communicate between devices. If both TXD and RXD pins are connected, both devices can communicate in both directions. However our car needs only information about its position. There is no need to send messages to GPS sensor. Therefore only TXD pin is used to send information about position from GPS to Arduino. In Arduino's code there is defined pin 2 to be RXD pin of UART communication protocol. The result is only one way communication between GPS sensor and Arduino board possible.

Compass uses I2C communication protocol to transfer information about its direction. SDA pin on compass is connected to A4 pin on Arduino. SCL pin on compass is connected to A5 pin on Arduino. Arduino board allows I2C communication only through A4 and A5 pins.

After powering up the devices compass is ready to use instantly. GPS sensor, however, tries to find satellites at first. It takes variable amount of time depending on obstacles nearby. If GPS sensor does not have any GPS signal, green led diode blinks. Arduino board begins to send relative angle to main Raspberry board only after GPS sensor has established connection to satellites so position information is valid.

Arduino sends relative angle to Raspberry board every second. This process cannot be speeded up because GPS sensor cannot inform Arduino about position more often (there is 1Hz frequency). Sensors with fixed compass are located in the front of the vehicle as shown on the picture below.



Compass and GPS sensors

4.2 Control Unit

Control unit code is written in python and contains 2 main threads. One thread is responsible for receiving packets and updating data in sensors objects. Second thread is used for vehicle manual controlling.

4.2.1 Global variables

In code, there are several global variables used in processing, decision making and moving vehicle. For speed control, there are:

- MOVE_FORWARD – value sent to vehicle to move forward
- SPEED_NEUTRAL – value sent to vehicle to set neutral speed
- MOVE_BRAKE – value sent to vehicle to stop vehicle
- MOVE_BACKWARD – value sent to vehicle to move backward

For laser processing, there are:

- MIN_DISTANCE – value used for object minimal distance from vehicle
- DIRECTION – direction of vehicle
- ANGLE_CONSTANT – used in processing for range of angles

4.2.2 Manual control

Manual control has been developed to overcome auto control. User can manually control vehicle with w, a, s, d keyboard letters. W stands for moving forward, A for turning left, S for moving backward, D for moving right. Also there is Q option for stopping vehicle. Manual control is possible only in command line. While holding W, speeds are sent to wheels from 91 to 150 incremented by 1. If holding S, speeds are sent to wheels from current value to minimal 30 decremented by 1. If there is transition from 91 to 90, first MOVE_BRAKE value is sent, then SPEED_NEUTRAL value is sent and then 89 value is sent. After that, value is decremented by 1 till it reaches 30.

4.2.3 Processing packets

Processing packets is done in separate thread. In while loop, control unit listen for packet on port 5000 and IP address 192.168.1.200. After control unit is started, it waits until all wheels send their IP addresses assigned by DHCP server (which is configured on control unit). Till all IP addresses are received, there is no decision making.

After all IP addresses are received, control unit stars to listen to communication in network. Based on communication protocol, control unit differs every packet. When packer is received, control unit processes it and make decision if suitable.

4.2.4 Moving vehicle

Vehicle can move in four directions – forward, backward, left, and right. It can also stop if necessary. For moving forward, control unit sends a message to wheel specifying speed value. Range of speed is from 91 to 150 – MOVE_FORWARD constant.

If vehicle needs to stop, control unit sends two messages to every wheel. First message is sent with value of MOVE_BRAKE constant and second message with value of SPEED_NEUTRAL constant.

If vehicle is turning, two wheels are going faster than other two. For example, if turning left – right wheels have speed value equal 130 and left wheels have speed value of 100. If turning right, the concept is the same.

For backward moving, the concept is the same as for moving forward. The difference is if there is manual controlling (section 4.2.2).

4.2.5 Decision making

Decision of direction is done based on received data from laser, GPS, compass and camera. Two directions are computed using these data. First is calculated with laser, GPS and compass and second is calculated with camera. Final direction is then decided based on these two computed directions. Laser, GPS and compass direction has highest priority because it is used to for collision avoidance and navigation. Camera direction has lower priority as it is used for keeping vehicle in the driving lanes. This direction is taken into consideration only if the higher priority direction is straight. Table below shows all possible combinations and the final direction.

Decision making based on received data

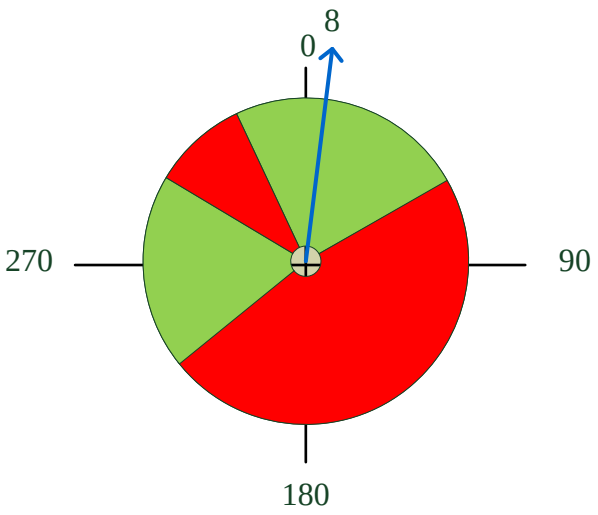
Laser, GPS, Compass	Camera	Final direction
↑	↑	↑
↑	→	→
↑	←	←
→	↑→←	→

←	↑→←	←
X	↑→←	X

First direction is computed with laser, GPS and compass data. Data from GPS and compass on Arduino are transferred into relative degree value which is received by central control unit. This degree tells where the next navigation point is. Laser data contains ranges of free angles (without obstacle). Only ranges which are big enough for vehicle are received.

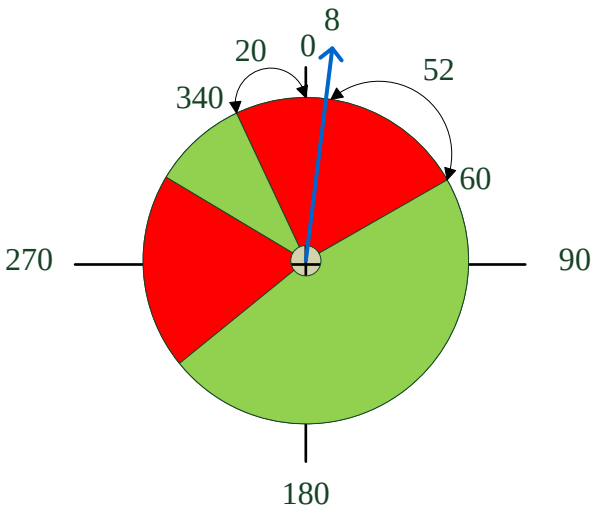
We need to specify range for the straight movement because we can't go straight only if the relative angle is 0. We specified range of 20 angles for this purpose. There are several cases for determining direction based on these values. On the following pictures is relative direction shown as blue arrow, free angles as green areas and restricted angles as red areas.

Case 1: Relative direction is in range <350, 10> degrees and this angle is in free zone based on data from laser. Decision: straight.



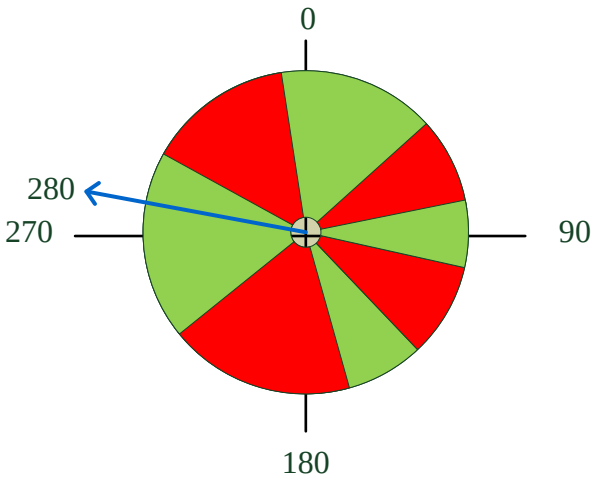
Case 1 in decision making

Case 2: Relative direction is in range <350, 10> degrees and this angle is in restricted zone based on data from laser. Decision: find closest free range to relative degree and turn that way. In the picture below angle 340 is closer, so the decision will be left.



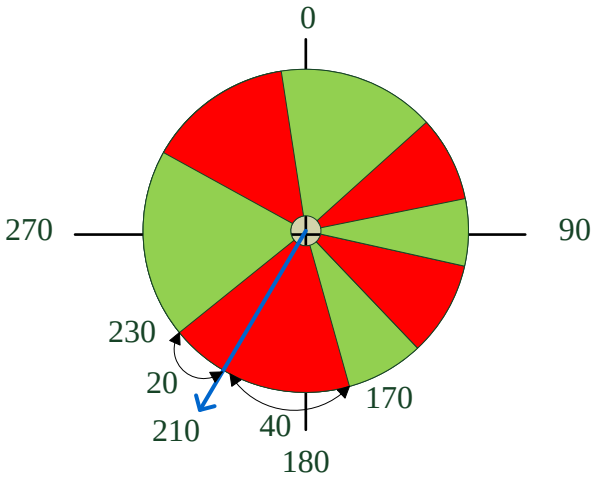
Case 2 in decision making

Case 3: Relative direction is not in range <350, 10> and direction angle is in free zone. Decision: turn to direction. Turn for direction in range <0,180> will be right and for range <181,360> it will be left.



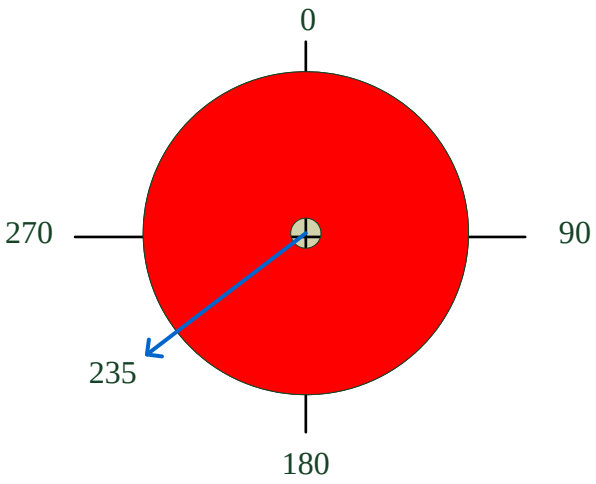
Case 3 in decision making

Case 4: Relative direction is not in range <350, 10> and direction angle is in restricted zone. Decision: find closest free range to relative degree and turn that way. In the picture below degree 230 is closer so the decision will be right.



Case 4 in decision making

Case 5: There is no free zone received from laser. Decision: stop.



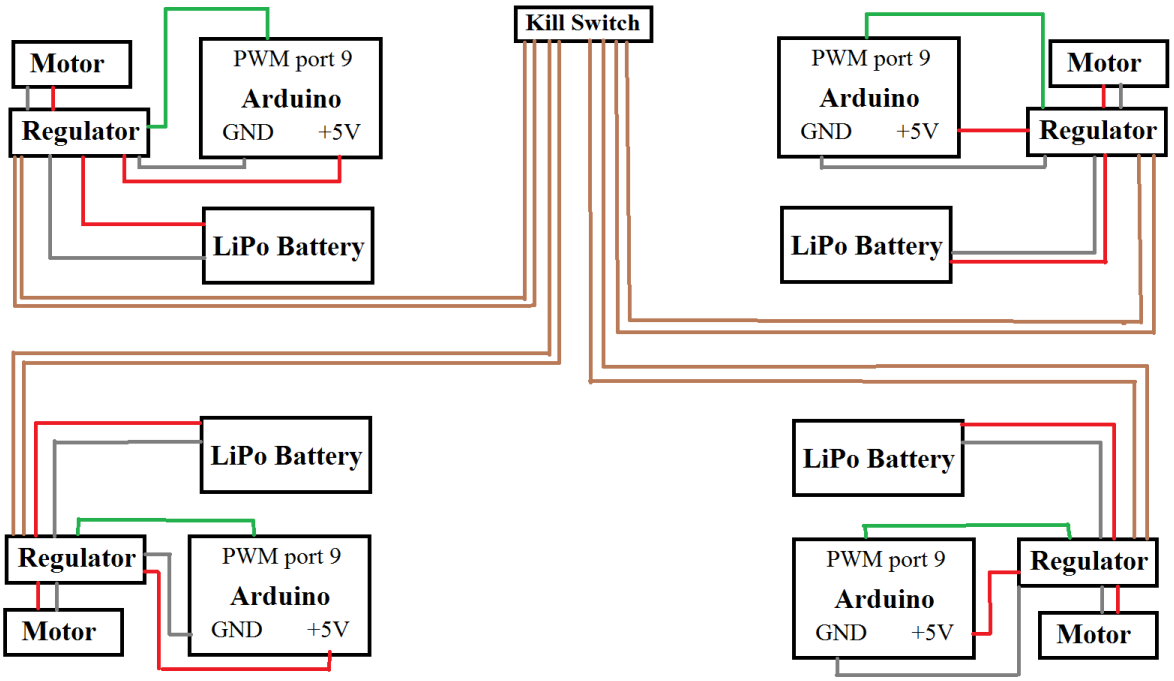
Case 5 in decision making

Camera decision is much simpler than decision from laser, GPS and compass. Only one degree is received from camera that is responsible for keeping vehicle in road lines. This angle is in range <-90, 90> degrees. Values in range <-20, 20> are represented as straight direction. Other positive values results to right direction and other negative values results to left direction. For the future improvements turning intensity for each degree can be specified for more precise correction of steering.

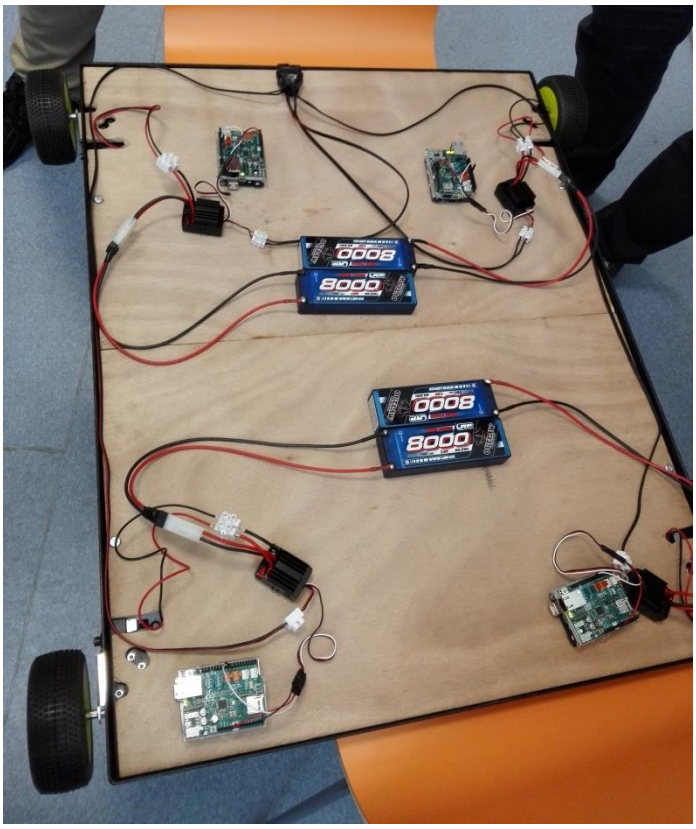
4.3 Car wiring

In the picture below is shown wiring diagram which describes how cables and devices are connected within the car. Every Arduino use 3 wires to connect to regulator. First cable is +5V (red), second cable is ground (grey) and third cable is PWM signal which controls a motor movement. Every regulator is connected to its battery which provides a power supply

for an Arduino and a motor. There is also a kill switch in case of emergency. Kill switch switches of every regulator in the car so the car will stop immediately after a kill switch was pressed.

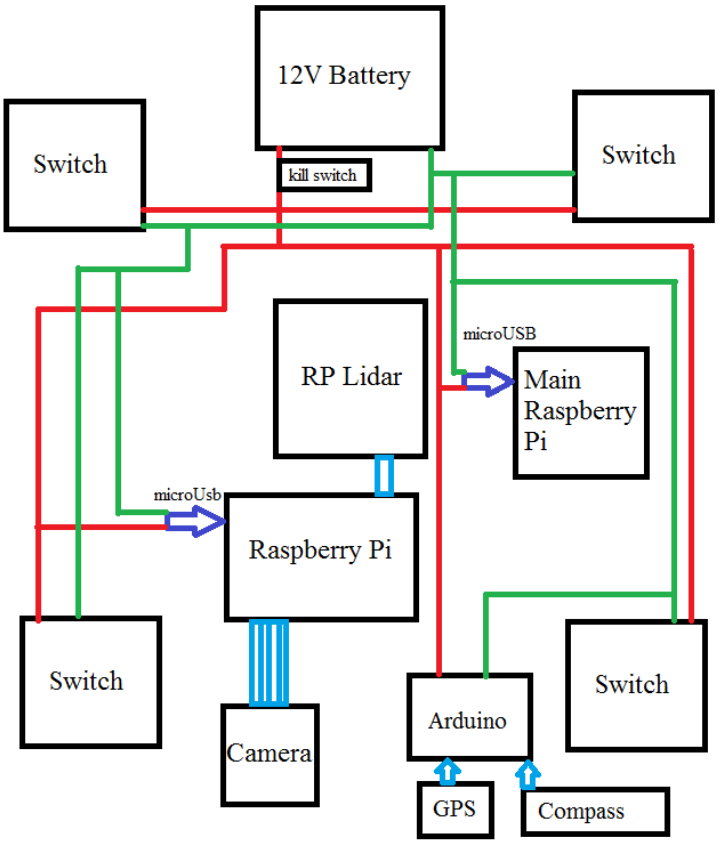


Wiring diagram of the first circuit in autonomous car

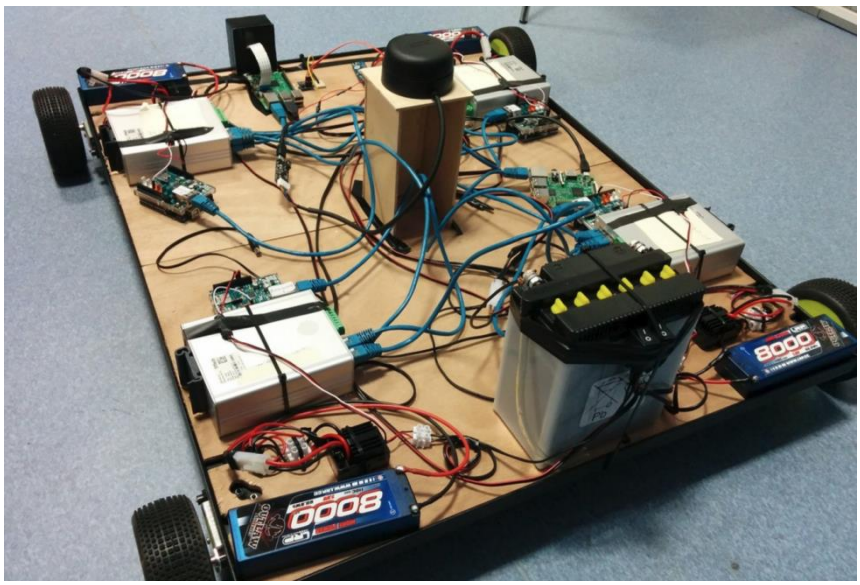


Wiring of first circuit in real vehicle

In the picture below is shown the second wiring diagram which describes how 12V battery powers up another devices placed in the car. Red cable represents +12V and green cable represents ground. There are also used voltage limiters to convert 12V to 5V power supply. Parts connected with light blue cables use specific cables for their connection.



Wiring diagram of the second circuit in autonomous car

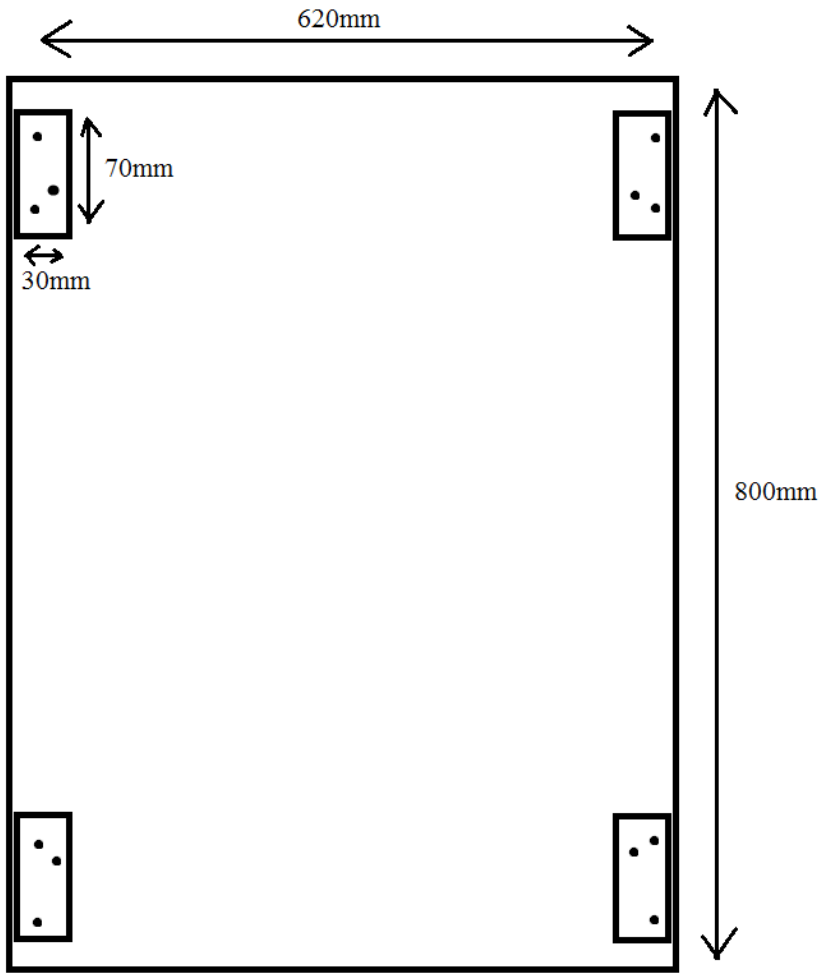


Wiring of the second circuit with first circuit connected together

4.4 Car construction

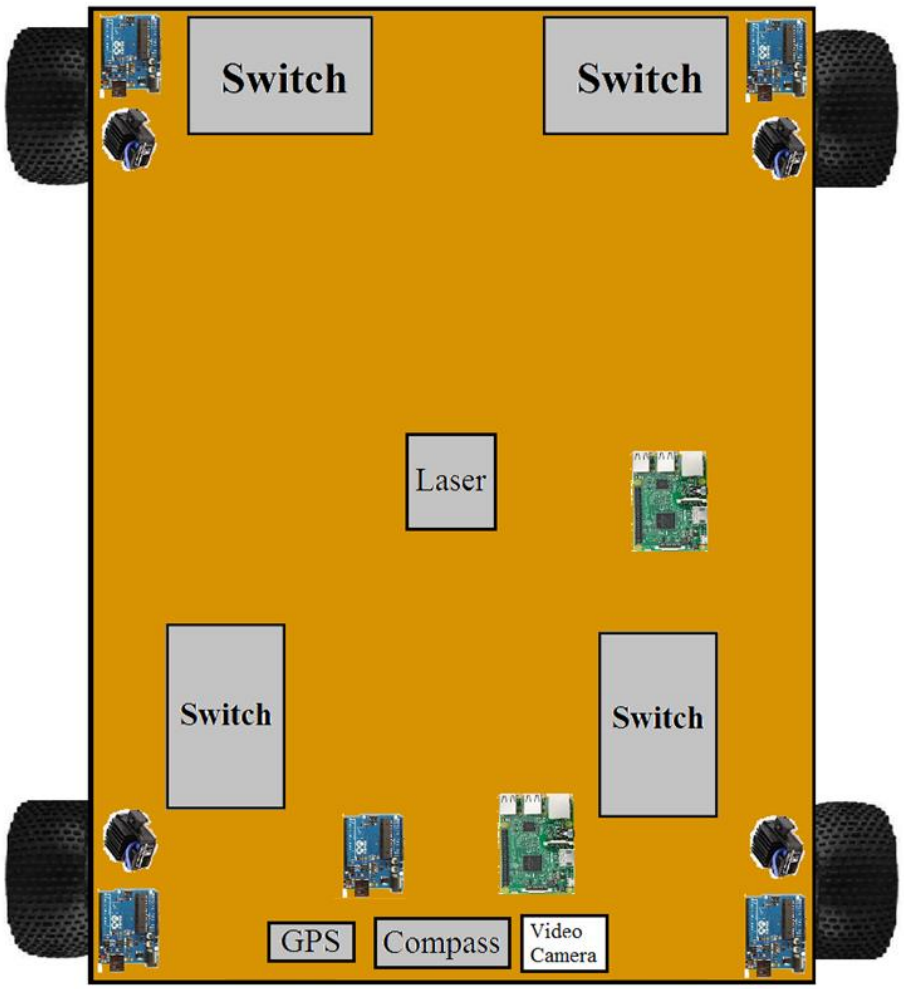
A car construction can be seen in the picture below. A basic rectangle is made of L - shaped metal with dimensions 20x20mm. It is welded in every corner. Four small platforms were

placed inside the main rectangle to hold motors with transmissions. Every platform has 3 drilled holes with 3mm diameter. These holes are used to mount motor to platform. Platforms are also welded to main rectangle. Inside the rectangle there is a wooden surface which is used to store every device used in the car. The whole car weights around 7 kilograms.



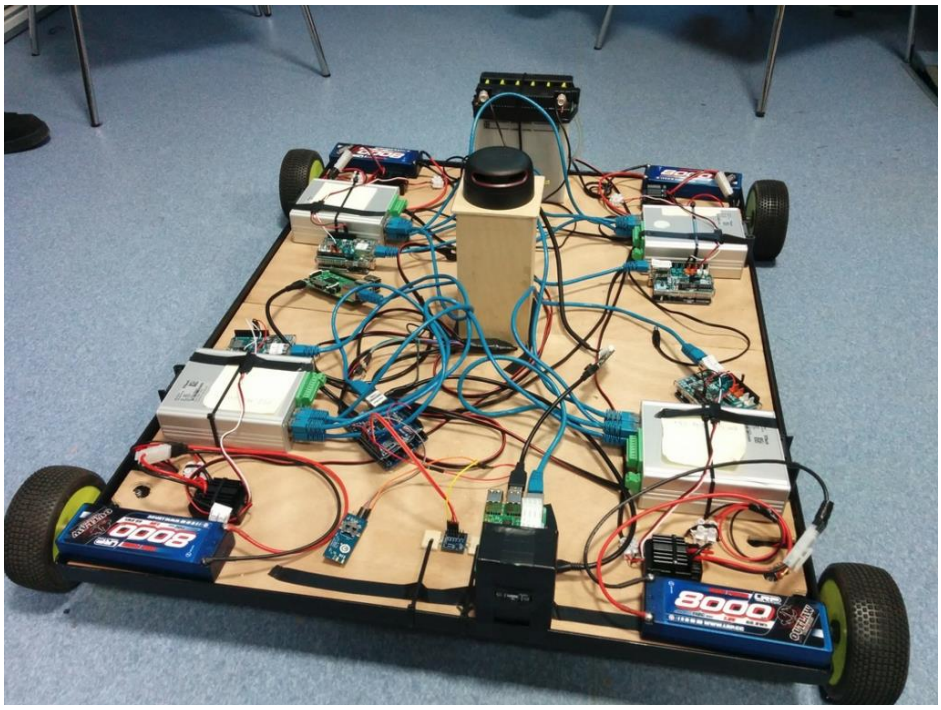
Car construction scheme

There were also made some changes to physical design of the car. Updated design can be seen in the picture below. The biggest change was made thanks to Altera boards. We decided not to use these boards so we replaced them with another Arduino and Raspberry Pi. GPS and compass were connected to the fifth Arduino and camera with laser were connected to another Raspberry Pi. There is also only one camera instead of two cameras in solution design. This is because we only recognize lines in the road, not road signs.



Updated physical car design

In the picture below there is the final physical design of the autonomous car. We used zip-ties to catch all the hardware to the construction. Laser is placed on wooden tower which is high enough to properly read distances from obstacles nearby. In front of the car there is a camera. It is taped to paper box from Arduino board. This solution is lightweight and stable. Compass is screwed to the wooden floor of the car to ensure its proper direction. GPS sensor is placed freely to the front of the car, because it does not need to be in precise location in the car.



Final physical design of the autonomous car

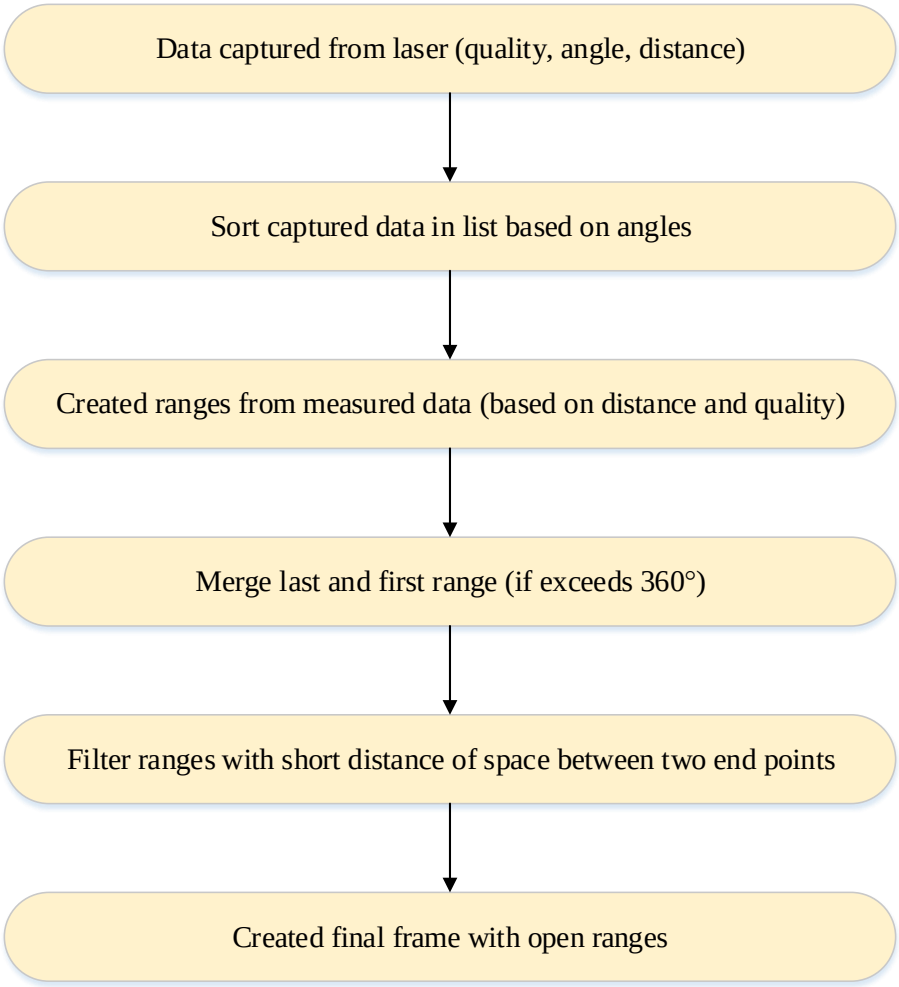
We used a loctite glue to lock the wheels to transmission axes. Motors with great transmissions provide great amount of torque so wheels need to be loctited. In the rear part of the car there is located large heavy Pb 12V battery. This battery is used to power everything except from motors and hardware which is used to control motors. Converters from 12V to 5V power supply are placed freely in the middle of the car.

4.5 Laser data processing

Processing of laser data is written in Python programming language using pre-defined libraries. Libraries include functions, which provide easy working with laser sensor (e.g. connect to laser, start motors, iterate through measurements or scans, stop motors, get info about laser, set PWM frequency).

Developing of laser processing was made in IDE for Python - Pycharm Community Edition. For processing data some external libraries had to be included into project: *numpy* for creating matrixes and *math* for mathematical operations, e.g. cosine or square root.

Diagram below illustrates processing of one sample with measured data:



Processing of measured data from capturing data to create final frame to send

This section of implementation includes description of each part from diagram above.

4.5.1 Data capturing

Laser libraries include functions for iterating through scans or measurements. For this purpose, it is better to iterate through scans, where each scan contains three values:

- **Quality** – if quality equals zero, given measurement was not successful – due to either object is too close to laser, or too far from laser.
- **Angle** – angle from specific measurement.
- **Distance** – distance from specific measurement.

```
def iter_scans(self, max_buf_meas=500, min_len=5, force=False):

scan = []
iterator = self.iter_measurments(max_buf_meas, force)
for new_scan, quality, angle, distance in iterator:
    if new_scan:
        if len(scan) > min_len:
            yield scan
        scan = []
```

```
    if quality >= 0:
        scan.append((quality, angle, distance))
```

Function returns list of the measurements. Each measurement is tuple with following format: (quality, angle, distance). After that, the list is evaluating.

4.5.2 Data sorting

Since measurements in list are sorted based on when they are captured (it means some measurements can be reversed), list of scans need to be sorted based on angles. For next processing and creating ranges it would be difficult to work with this original not sorted list.

```
 s_scan = sorted(scan, key=lambda x: x[1]) #sorting of measured data based
on angle
```

4.5.3 Creating of ranges

This is the most complex part of algorithm. Ranges represents angles, which have all distances between two end angles longer than limited distance, and distance of space between these two points are longer than limited distance of space. Detail description of ranges are described is here: 3.5 Laser data processing.

First condition in loop through all measured data is checking of quality. We suppose if quality is zero, there is enough space and assign this angle maximal measured space. It is mainly because of losing of many data, since application is designed for outdoor environment, where distances of objects are bigger than maximal measured distance from sensor. However, laser is in the middle of chassis, so it avoids of measuring objects, which are too close from sensor.

```
for meas in s_scan:
    if(meas[0] == 0):  #if quality is zero, suppose distance as max
measurable
        distance = MAX_DISTANCE
    else:       #otherwise, assign original distance
        distance = meas[2]
```

After checking of quality, distance can be checked. If actual distance is less than limited distance, range must be closed (only if previous angle has longer distance than limited). If range must be closed, array representing range is created. Array includes four values – begin angle, distance in begin angle, end angle, distance in end angle. This range is then append to list of all open ranges.

```
if(distance < LIMIT_DISTANCE):  #if distance is less then limited distance
    if(previous_bigger == 1):   #if in previous angle was distance bigger
then limited, close range
```

```
        range = [begin_angle, begin_distance, s_scan[j-1][1], s_scan[j-
1][2]] #create range (beg-angle, beg-dist, end-angle, end-dist
        ranges.append(range)    #append created range to array of ranges
        previous_bigger = 0;    #set control variable to zero
```

For remembering previous measurements, help variables are keeping – previous_bigger, begin_distance and begin_angle.

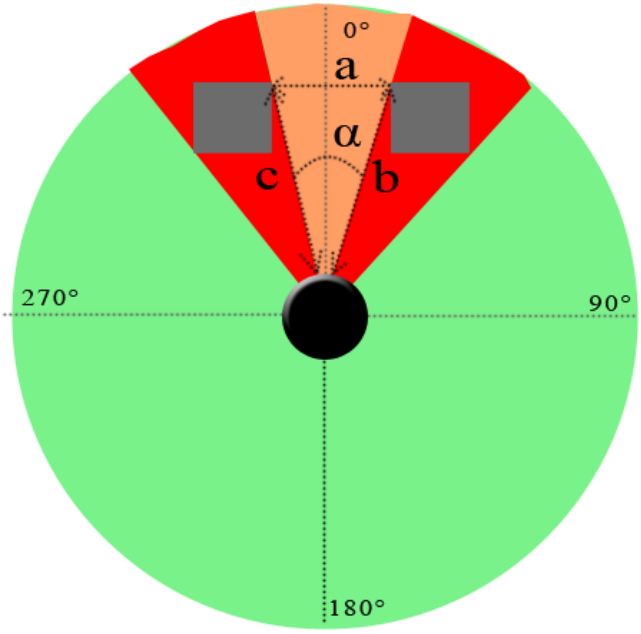
4.5.4 Merging last and first range

Since processing of measurements begins with angle zero, first range can only start from zero degree (if distance in that angle is bigger than limited). Similarly, last range can end only with angle 359. From first processing this is represented as two angles, since in real life it is only one angle. From that reason, if this situation occurs, these two ranges must be merged into one (e.g. begin angle is 270 degrees and end angle is 30 degrees). Python provides feature for accessing last element in array – index -1. If merging is needed, first range is updated – begin distance and angle are changed for begin distance and angle from last range, and last range is removed from array of ranges.

```
if(ranges[0][0]>0 and ranges[0][0]<2 and ranges[-1][2]>357 and ranges[-
1][2]<360):  #if range exceeds 360 degrees, merge first and last range
    ranges[0][0] = ranges[-1][0]    #edit angle in first range -> change
for begin angle from last range
    ranges[0][1] = ranges[-1][1]    #edit distance in first range -> change
for begin distance from last range
    del ranges[-1]                  #delete last range
```

4.5.5 Filtering ranges with short distance of space

Created ranges usually includes angles, which are too short. For their deletion, cosine sentence is used. Known are two sides (distances from begin angle and end angle) and angle between these two sides (difference between begin and end angle). From cosine sentence, third side can be count. Only ranges with big enough space between two end points to pass between obstacles are assigned to new created array.



Illustrated values for cosine sentence

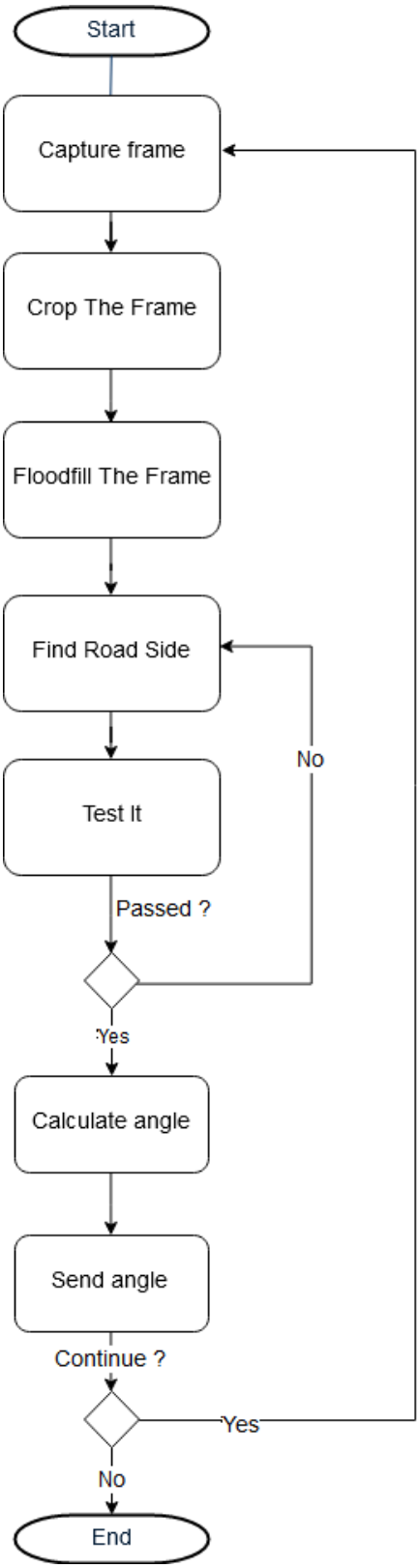
```
def count_distance_of_space(range):

alfa = range[2] - range[0]  #getting angle alfa
val = math.pi / 180.0   #pre-counting for converting radians to degrees
ret = math.cos(alfa * val) #countig of cosinus alfa
return math.sqrt(square(a) + square(b) - 2 * a * b * ret)  #cosinus
sentence - known b,c, alfa, return side c

def remove_short_ranges(ranges):

if(space > LIMIT_SPACE):
    new_ranges.append(r)     #if space is bigger then limited distance for
space, add to new ranges
```

4.6 Image processing for roadside detection

Image processing is written in Python language using the Open CV library (*cv2*) as well as *numpy* library for matrices and mathematical calculations, *time* for timeouts, *socket* for udp communication and data sending and *picamera* for Raspberry Pi camera module control.

The camera continuously captures frames which are then being processed one by one.

4.6.1 Frame processing



Processing of camera output

The frames are being captured using the *picamera* module which is also used to setup the wanted resolution and the expected framerate. After that a small timeout is used to ready-up the camera for continuous capture of frames (video).

```
camera = PiCamera()
camera.resolution = (320, 240)
camera.framerate = 20
rawCapture = PiRGBArray(camera, size=(320, 240))
time.sleep(0.1)

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
```

Each frame is then processed by Open CV flip, crop and floodfill functions. Flip ensures the frame orientation, according to the camera module positioning.

```
frame = frame.array
frame = cv2.flip(frame, 0)
frame = cv2.flip(frame, 1)
```

Crop cuts out parts of the frame, to lower the size of the frame, fasten the processing and avoiding of unimportant parts of the frames, since we are only interested in the road which is situated in the lower part of the frame we cut the frame in half vertically.

```
if not frame == None:
   h, w = frame.shape[:2]

   cropped = frame[int(h/2):h,0:w]
```

Floodfill is a filter that repaints the frame given by filling the area around a point given, that falls in the color range to a specific color.

Firstly a point in the bottom center is chosen which color is used as the floodfill base color for comparison (RGB). Next a value has been chosen for the higher and lower threshold to the base color. Floodfill then overwrites the current frame with the “repainted” colors. The color to which it repaints is defined in the parameters, in our case we use a grey color (150,150,150).

```
point = (int(w/2),h-10)
mask[:] = 0
value = 10
lo = (value,value,value)
hi = (value,value,value)
flags = 4
```

```
cv2.floodFill(flooded, mask, point, (150, 150, 150), lo, hi, flags)
```

After floodfill we find the roadside by looking for a lighter color then our custom grey we added in the last procedure. We start from the very right side of the frame going up and looking for a different color.

```
def get_angle (point, frame):
    b,g,r = frame[point[1]][point[0]]
    h, w = frame.shape[:2]
    distance = 0

    for j in range(5,100,5):
        b2,g2,r2 = frame[h-j-1][w-1]
        if b2 > b or g2 > g or r2 > r:
            break
```

If it is found, we then search from the center to the right side of the frame for the roadside.

```
    for i in range(int(w/2),w,3):
        b2,g2,r2 = frame[h-j-1][i]
        if b2>b or g2 > g or r2 > r:
```

The we test the found roadside with the *test_area* method. Which checks a 3x3 area from the found roadside to ensure we didn't find a random light spot.

```
            boolean = test_area (h-j-1,i,frame,b,g,r)
```

If the area seems correct, we calculate the angle in which the vehicle should move to keep an approximate distance of 80px from the roadside. We use the arctan goniometric function and the known coordinates of our points.

```
            if boolean == True :
                distance = i - int(w/2) - 80
            if distance < 20 and distance > (- 20) :
                return 0
            else :
                angle = int(np.rad2deg((np.arctan((distance)/(j)))))
                print (angle)
```

Calculated angle is then converted to unsigned byte, since our angle values can only be between (-90°, 90°).

```
                if angle < 0 :
                    angle = angle + 128
                return angle
    return None
```

After we got the angle we send it to the control unit by UDP using our custom protocol.

```
def send_camera_angle(angle):
```

```
# 01 - message to central unit
# 01 - source board is raspberry pi
# 02 - # of source board
# 01 - destination board is raspberry pi
# 01 - # of destination board
# 0011 - type of message is infrared_camera_data
if angle == None:
    angle = 0
data = bytearray([1, 1, 2, 1, 1, 0, 3, angle])
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(data, ("192.168.1.200", 5000))
```

5. Testing

5.1 First experiments

After we constructed vehicle, we tested its movement. This test was performed indoors. First time we tested vehicle movement in our lab, where we it was constructed. There were two types of tests:

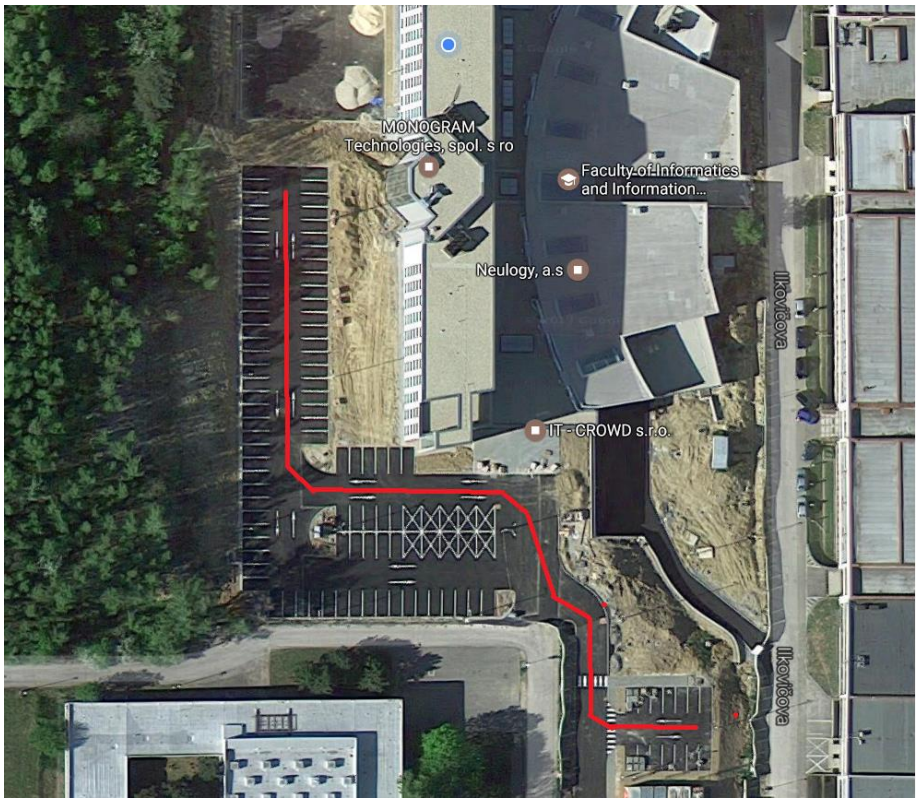
- straight movement,
- turning.

Purpose of straight movement test was to prove that vehicle is able to go straight without human control. This test was successful, despite its high noise. Unfortunately, there is no way we can fix high noise.

Turning test was held in the hallway. The purpose of this test was to prove that vehicle is able to turn. This test was also successful. Thanks to our software implementation and vehicle construction, vehicle was able to turn almost on the spot.

5.2 GPS and compass outdoor testing

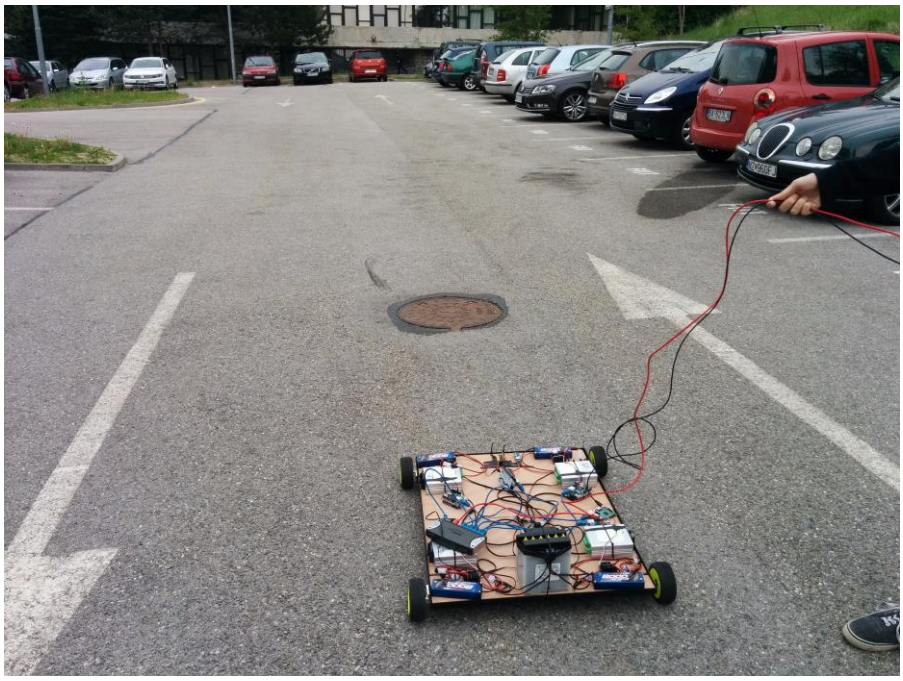
Outdoor testing was held near FIIT faculty on parking lot. For auto controlling and moving vehicle we created map of vehicle road on parking lot shown on the picture below. This map consisted of few turns to confirm the functionality of GPS and compass decision algorithm. Vehicle is supposed to go from one point of map to another till it reaches last point (end of road).



Testing map

Compass precision showed minimum deviations. It was only affected when then vehicle was right above metal channel hatches. Bigger deviations were detected in GPS accuracy which is essential for precise navigation. This accuracy can be affected by near buildings, weather conditions and number of tracked satellites. NMEA messages from GPS contains information about satellite numbers and deviation. These fields can be used to prevent any action until favorable conditions are met. However, they can never met and vehicle will just stand on place.

During many tests, we finally got precise GPS position and navigation could be tested. Vehicle was successfully navigated via predefined route and also to make a successful turn so the functionality of navigation based on GPS and compass were proven. Picture from this testing can be seen below and video of successful turn is located on the enclosed CD.



Outdoor testing

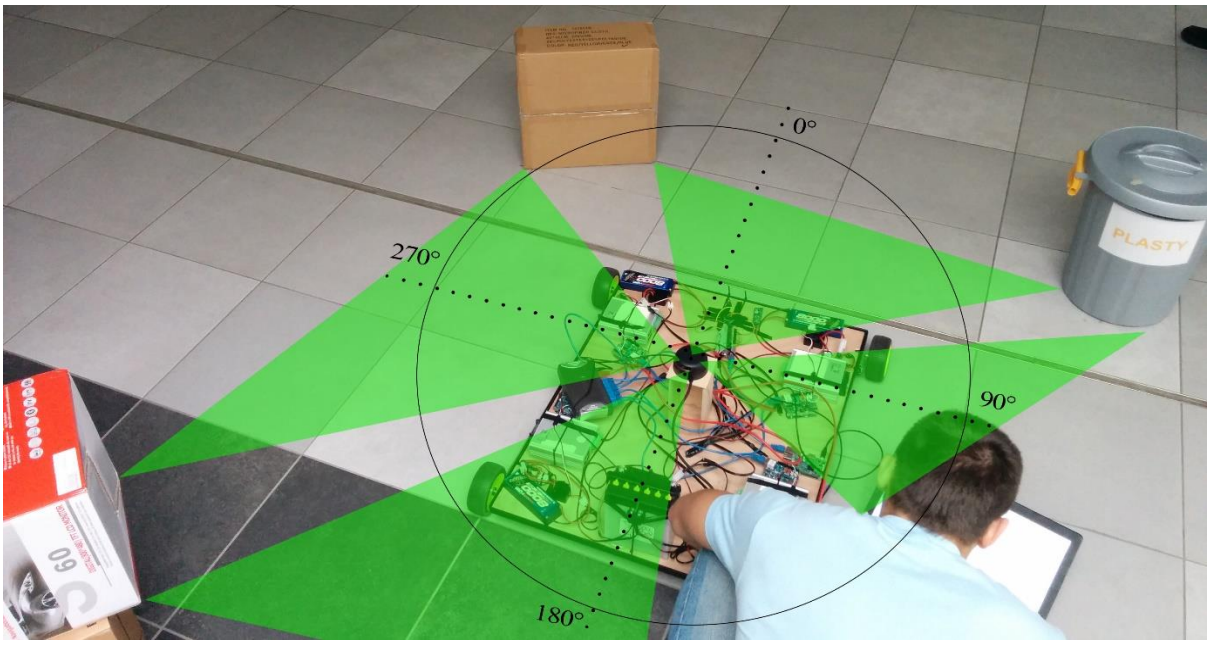
5.3 Manual control testing

For manual testing we used area near FIIT. We started our vehicle and using WASD we move it from one point to another. Testing was successful. The only thing we had to change was speed of wheels. We set it so vehicle goes in appropriate speed. We tested moving forward, moving backward and also turning vehicle to left and right. All test cases worked.

5.4 Laser testing

Laser processing was tested in area of FIIT faculty in big hall. Near vehicle few boxes which simulated obstacles were placed, as is shown in figure below. After turning program on, we were watching what ranges are sent and what is the behavior of vehicle (control system). We could see some weaknesses, mainly caused some wrong measurements from laser.

Laser processing tests were provided also without existence of vehicle. Correct choice of ranges could be tested without another devices.



Laser process testing

Here are sent ranges from measurement:

- Range #1: [350.9, 3128.5, 3.3125, 6697.25]
- Range #2: [4.28125, 6688.75, 58.078125, 2000]
- Range #3: [77.109375, 2000, 115.390625, 2000]
- Range #4: [120.15625, 2000, 142.15625, 2000]
- Range #5: [192.140625, 2000, 229.734375, 2000]
- Range #6: [244.84375, 6499.25, 333.109375, 6716.25]

From given test scenario we expected 4 ranges. Nevertheless 6 ranges were detected and sent. First range ends in 3,3° and next open range begins in 4.2°. Between them there was some wrong measurement, but quality bit was higher than 0, so this scan is considered as valid:

Fourth wrong scan from measurement

# of scan	angle	distance	quality
4	3,5	0,0	13

If first two ranges were merged, probably it could represent angle between brown box and trash bin (begin angle = 350°, end angle = 58°).

Similar situation occurred in next range, where even four ranges in row had distance zero and quality higher than 0.

Another wrong scans from measurement

# of scan	angle	distance	quality
118	116.4	0,0	13
119	117.3	0,0	13

120	118.2	0,0	12
121	119.2	0,0	11

Similarly, if range #3 and range #4 were merged, there could arise range, which probably represents real range between trash bin and laptop (begin angle = 77°, end angle = 142°). Range #5 and range #6 are almost real representation from given scenario. There are deviations with figure above, but it is caused by wrong drawn coordinate system.

The control variables for algorithm were set as follow:

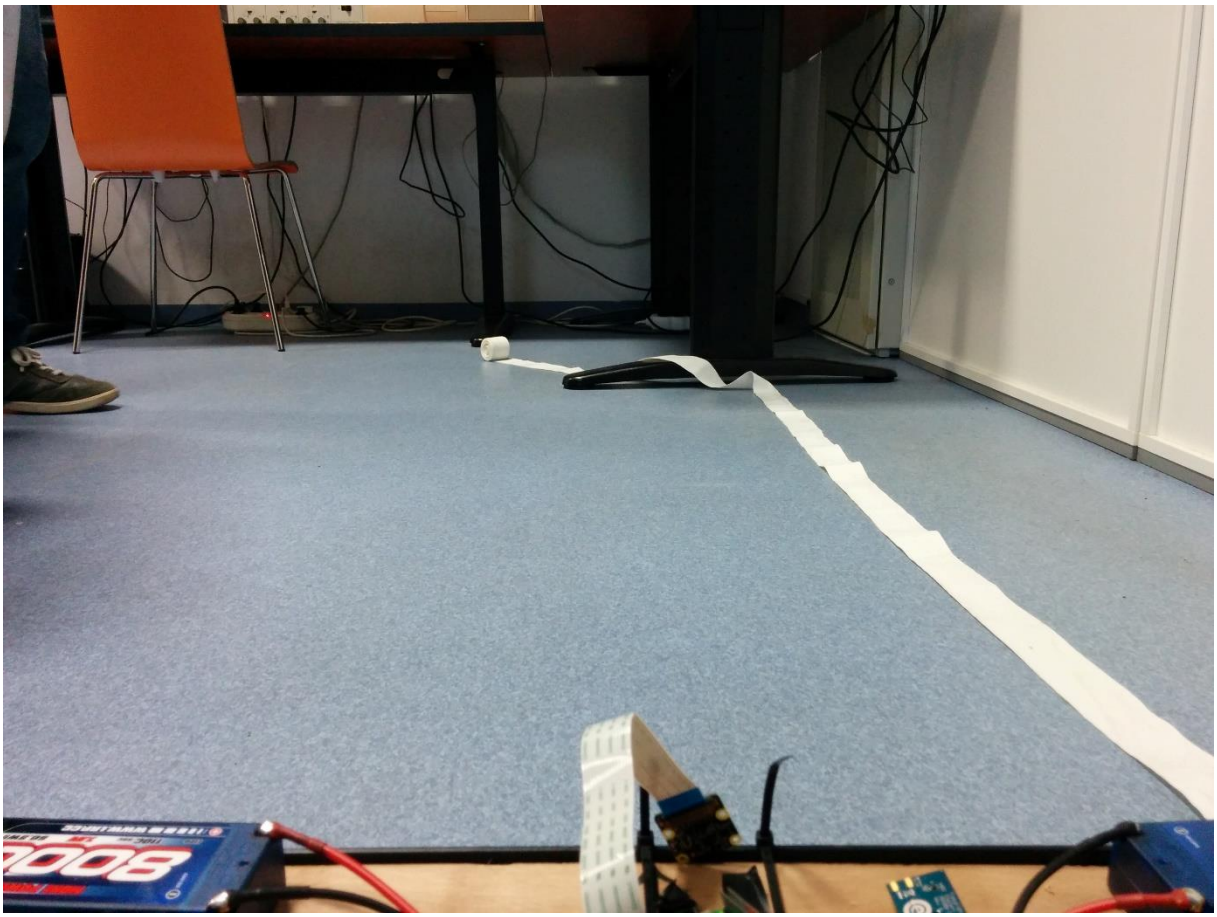
- LIMIT_DISTANCE = 1500 - limit distance in front of obstacle (in millimeters)
- MAX_DISTANCE = 2000 - maximum distance, which is assigned when measurement is not successful (in millimeters)
- LIMIT_SPACE = 500 - maximum length of space between two end points (in millimeters)

Most ranges have start or end distance equaled 2000 mm. This is caused by quality bit, which is equaled zero. In that case, given scan was not successful and we supposed obstacle is far away from sensor, so for that angle, maximum measurable distance is assigned.

With provided ranges, there were made also tests for functional behavior of vehicle. Decision maker on control unit was tested to all cases according to section 4.2.5 Decision making. All cases were simulated with change of DIRECTION variable and all decisions were processed according to specification.

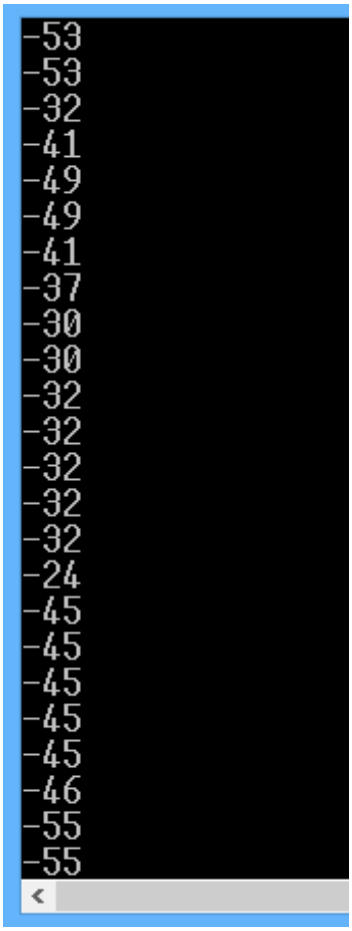
5.5 Camera testing

Camera processing was tested inside the faculty. A white roll of textile was used to simulate the road side and the behavior and sent angles have been monitored.



Normal picture of the tested area (taken from behind the camera)

Processed frame from camera view



Angles calculated during the camera manipulation

6. Future work

This section contains other ideas for future work development and upgrades.

6.1 WiFi manual control

Hotspot with private network can be deployed on control Raspberry Pi to provide wireless connectivity to all boards of the vehicle to provide simplified control of vehicle. Hotspot should use private range of IP address space with additional DHCP server (careful one DHCP server is already running on this board to assign IP addresses to Arduino boards). Connectivity between this network and internal 192.168.1.0/24 network must be ensured. This can be done by tunneling between wlan0 and eth0 interfaces.

6.2 Increased GPS precision

GPS precision is essential for correct navigation, therefore some new additions to increase GPS precision should be applied. Combinations of more GPS boards with computed triangulation should solve the problem. If this will not fix the issue, GPS should be only supplementary and camera should be used for track recognition. Probability of turn could be then computed with GPS, but it must be confirmed by the camera. Keep in mind that GPS is the more precise the faster is vehicle moving.

6.3 Fix issues with TTTech switches

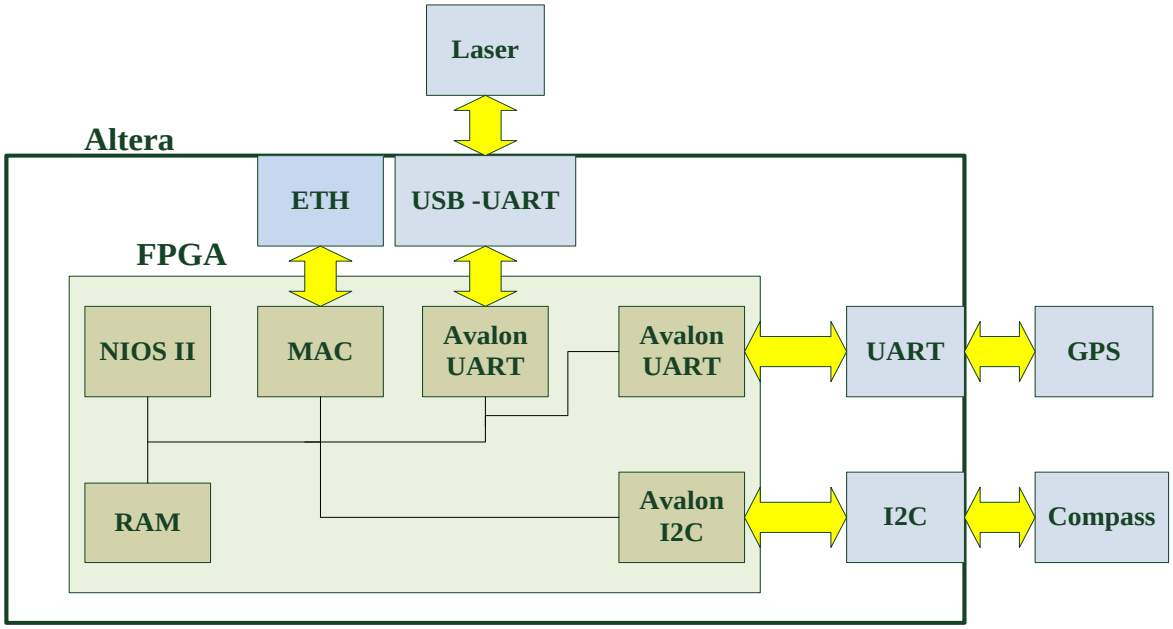
Large problems have arisen with TTT switches that do not always work as they are meant to. Turning switches on with the same topology of cabling has not always lead to full switch functionality. Long debugging sessions were needed to somehow fix the issues which later again appeared. This long spent time was a big complication with every testing. Our own small switch was sometimes used to fix these issues so we could test our programs. In future work this issue is should by primary to fix. Possibilities without interference of TTTech are very limitless. Another small issue is big power consumption of switches (4 switches 1.5 A).

6.4 Altera FPGA extension

For future work FPGA on provided Altera boards can be programmed to contain IP cores shown on picture below. It will allow communication with all three sensors via UART and I^2C protocol. Cores are made in Altera Quartus Prime program using QSys tool.

- NIOS II – free version of 32-bit embedded-processor architecture designed for smallest possible logic utilization of FPGAs. Contains JTAG debug module and up to 256 custom instructions. Acts as avalon mm master.
- RAM – random access memory. Acts as avalon mm slave.

- MAC - is IP core for communication with Ethernet interface with deterministic Ethernet support from TTTech. Acts as avalon mm slave.
- AVALON UART - The UART core with Avalon interface implements a method to communicate serial character streams between an embedded system on an Altera FPGA and an external device which is in our case Laser and GPS. Acts as avalon mm slave.
- AVALON I2C - simple two-wire, bidirectional interface developed for I2C communication with any I2C slave device with compatible pins in our case with compass. Acts as avalon mm slave.



Realization of FPGA

6.5 Bodywork

Bodywork should provide protection from weather conditions, crashes and also for aesthetical appearance. Design should consider easy access to all batteries, boards and also space for camera and laser. Camera and compass must have clear vision to road and surroundings of vehicle.

6.6 Obstacle detection

Since laser sensor can detect obstacles in its angle of measurement (which is less than 1°), obstacles below and above the sensor cannot be detected. With this solution, we cannot detect for examples curbs, through which our vehicle cannot pass. This problem can be solved by implementation at least two ultrasonic sensors to front and back part of vehicle's chassis.

While we have used low cost laser sensor RPLIDAR, we could see some inaccurate measurements from RPLIDAR sensor. Obstacle detection can be improved by replacing RPLIDAR for some better laser range finder, e.g. Hokuyo URG-04LX-UG01. However, its price is above 1000$.

6.7 Road sign detection

From the start of the project we planned to use a second camera for road sign detection, but due to various problems and difficulties we did not have enough time to implement this feature with the expected Altera boards. The hardware is bought, only the software is missing.

7. Technical documentation

7.1 Starting procedure

1. Connect all devices to appropriate ports and power supplies.
2. Turn on witch using moto battery. It takes about 30 – 60 seconds to start the switches to operating state.
3. Connect to Raspberry PI with laser and camera using VNC and start their python programs.
4. Connect to control Raspberry PI and start DHCP server using command sudo service isc-dhcp-server restart.
5. Start laser program, view section 7.4.
6. Start camera program , view section 7.5.
7. Start control unit program.
8. Turn on Arduinos for wheel control with external switch. They will request IP addresses and vehicle is ready to operate.
9. Vehicle will not move until some data from laser, GPS and camera are received. Manual control can be done in program on control unit.

7.2 GPS, Compass

Program with GPS and compass control will start automatically after powering up Arduino. Blinking red LED on GPS means that signal from satellites is not acquired, you need to wait until the led will stop to blink.

7.3 Control Unit

IP address of control unit is 192.168.1.200/24 and can be used to connect to it via VNC. Preinstalled DHCP server is isc-dhcp-server for assigning IP addreses to Arduino boards. Vehicle is not functional without working DHCP, because control unit is waiting for all Arduino boards (means 4 wheels are available). Other DHCP servers can be used as well.

Program handler.py located in PycharmProjects can be run from command line using command python handler.py. Python 3 is required to run this program. Vehicle will start when it receives all 4 messages from Arduinos.

7.4 Laser processing

For laser processing, firstly laser sensor must be connected to USB port. After that, correct USB port name with connected Lidar sensor must be chosen (for Windows *'\\\\.\\com3'*, for Linux *'/dev/ttyUSB0').* This could be done in main file *laser_process.py* by rewriting argument in constructor for creating instance of data object Lidar. Program is executed by same python file. If Lidar is connected and correct port is chosen, basic info about Lidar sensor should appeared and measurement with sending open ranges should start.

7.5 Camera processing

For camera processing, the camera should be connected to a Raspberry Pi device. On the device a virtual environment is needed to be selected, since we used a virtual environment for the OpenCV installation. Following commands need to be executed : *source ~/.profile* then *workon cv* and after that you just need to start the main script *play_video.py* (by using python of course).

Conclusion

This document summarizes work on team project on Faculty of Informatics and Information Technologies in Slovak University of technology. Goal of the project was to create autonomous vehicle based on prototype deterministic Ethernet switches from TTTech Company.

Proposed solution was design with specification of number of used switches, number of engines, measurements and others. One of the main problems was little knowledge about electronics. Despite of that we have built vehicle with power supplies for all different components which considers all voltage and power requirements.

GPS, compass, laser and camera are used for autonomous movement. Data from these sensors are processed by Raspberry Pi control unit which controls movement of all four wheels. Solution was tested in real environment near the building of FIIT.

Project is supposed to be extended by other team projects and bachelor thesis on Faculty of Informatics and Information Technologies and Faculty of Electrical Engineering and Information Technology of Slovak University of technology.

Bibliography

[1] https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/

[2] http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=930

[3] http://www.banana-pi.org/m2.html

[4] https://www.arduino.cc/en/Main/ArduinoBoardUno

[5] https://www.intorobotics.com/14-gps-modules-navigate-track-movements-raspberry-pi-project/

[6] https://www.adafruit.com/product/746

[7] http://store.digilentinc.com/pmod-gps-gps-receiver/

[8] https://www.alza.sk/

[9] https://www.raspberrypi.org/forums/viewtopic.php?f=28&t=150138

[10] http://www.micropik.com/PDF/HCSR04.pdf

[11] http://www.robotshop.com/en/rplidar-a2-360-laser-scanner.html

[12] http://www.sharpsma.com/webfm_send/1487

[13] http://www.robotshop.com

[14] https://www.clearpathrobotics.com/grizzly-robotic-utility-vehicle/

[15] http://www.superdroidrobots.com/