Design and implementation report for IARRC 2018 - Team Selfie

Mateusz Perciński, Mikołaj Marcinkiewicz, Mateusz Grudzień, Mateusz Mróz, Kornelia Łukojć, Mateusz Marczuk, Maciej Krasa, Krzysztof Gawin, Łukasz Jakubowski, Michał Jarzyński

Students' Robotics Association, Faculty of Power and Aeronautical Engineering
Warsaw University of Technology
Warsaw, Poland
Contact e-mail: mateusz.percinski@gmail.com

***Abstract*—This report presents the design and implementation system of autonomous car built to compete in International Autonomous Robot Racing Competition 2018. It describes selected components of platform and hardware, provides overview of system architecture and prepared software. The implemented methods are profiled to maximize performance of vehicle in competition events. Computer vision system is designed to enable high-speed localization on race lane and minimize computing load and latency in output data delivery. Image processing algorithms were selected to handle unstable outdoor environment. Localization data processed collectively with obstacle detection inputs enable optimal car trajectory planning. High-speed control is realized by application implemented in real-time operating system. It is running precise PID controllers of speed and steering angle. The report emphasizes how safety engineering and monitoring best practices were implemented.**

I. Introduction

The main purpose of this paper is to introduce the architecture of the system of small scale autonomous car - requirements-based design approach and implementation of ready-to-go system.

As market admittance of automated vehicles is coming closer and closer, questions regarding safety and system validation of algorithm are getting more and more important. Many approaches need to be tested properly in non-impacting environment. Challenges concern marking detection [1], obstacles avoiding with sensors [3], task handling reliability [2]. Technologies used in autonomous cars are very likely to become breakthroughs in other fields [4].

Small scale vehicles are low-cost platforms that allows testing most of aspects of localization and navigation algorithms for real scale autonomous cars. Students competitions like IARRC provide a space for testing ideas, competing and know-how sharing for students team.

(a) Model prepared with CAD software

(b) Final appearance of built car without bodywork

Figure 1. Vehicle built for International Autonomous Robot Racing Competition 2018

II. Platform Design

To comply with competition regulations vehicle is based on 1:8 chassis of RC car. Default mechanism connecting servo and wheels of front axle realizes Ackermann steering geometry. Developing and testing on small scale chassis with one working axle enable scalability of system for possible future reimplementation in bigger-scaled projects. Most of the mounting elements and electronics grippers are printed in 3D technology. Some critical elements were prepared manually by team members from metal, carbon fibre (camera stick) or

Research done cooperating with the Faculty of Power and Aeronautical Engineering, Warsaw University of Technology within the framework of Najlepsi z Najlepszych 2.0!" program.

rubber (front and rear bumpers). Figure 1 presents design of vehicle: 3D model (Figure 1a) and assembled ready-to-go car without bodywork (Figure 1b).

Particular model of purchased chassis Kit is RC8B3.le by company called *Associated Team*. The biggest advantage of this construction is small turning circle and hydraulic suspension system (it is operating smoothly with or without significant additional load). For steering angle control precise servomotor by *Sevox* was used. For this choice winning factors were high operating speed and full set of metal gears, which is not a common setup in this class standard servomotors, was used. Powerful motor *Reedy Sonic* with complying ESC (Electronic Speed Controller) was chosen. As it it brushless motor, not only it is able to easily achieve high speeds, but also operate smoothly in low rotational speeds.

Detailed information about models and costs of all component are provided in appendix A (Table I).

Dimensions of car after all modifications are:

- height: 370mm
- width: 320mm
- length: 650mm

III. Hardware Design

A. Hardware architecture

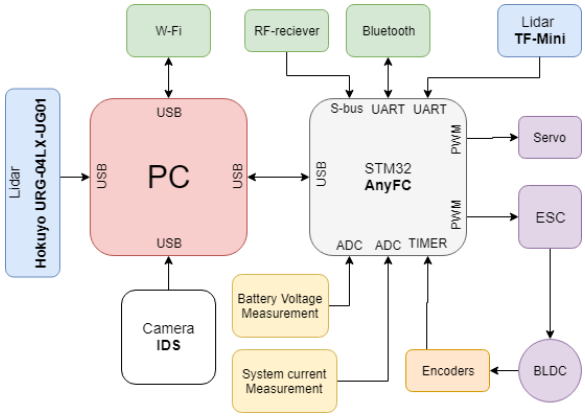


Figure 2. Map of communication between system elements

From high-level perspective Vehicle system is divided into 2 main processing subsystems and other peripherals. Figure 2 presents connections schema and points peripherals used for handling communication between main controller particular system elements.

B. Computer Vision System

After initial testing and building proof of concept two crucial design decisions were made. First observation was that minicomputers like RaspberryPi and Odroid were insufficient (in terms of computing power) for ongoing image processing and trajectory planning. That is why custom computer unit was assembled. It is based on PC equipment.

Second observation was that the crucial factor for high speed localization and control is number of frames per second captured by camera. Even with high-speed camera this value strongly depends on lighting in testing environment and camera parameters setup. Several models of cameras have been tested. Particular models of cameras and lenses was chosen as a compromise between angle of view and level of distortion.

Computer vision equipment:

1) Computer Unit:
   - Motherboard ASRock H110M-ITX
   - Processor Intel Core i3-7100T, 3.4GHz, 3MB
   - Memory: GoodRam DDR4, 4GB, 2.4GHz, CL12 + SSD Silicon Power A55 64GB
2) Camera:
   - Camera Kurokesu C1 - Resolution: 1920x1080, Max fps: 30,
   - Lens - focal length: 2.8 –12 mm, Angle of view: 93 degrees
3) **OR**
   - Camera IDS UI-1220LE - Resolution: 752 x 480, Max fps: 87.2 fps, 0.36 MPix, 1/3"
   - Lens - Topacc, focal length: 2,1 mm

(a) camera IDS (b) Camera Kurokesu (c) Lens

Figure 3. Camera equipment used in project

C. Controller

As main controller AnyFCF7 board is utilized. It is equipped with 32-bit micro-controller STM32F745 (processor ARM® Cortex-M7®, 216Mhz). AnyFC board was chosen because of its small size, relatively low price and ready-to-use connectors. Also built-in gyroscope is used. Figure 4 presents photo of AnyFC F7 controller.

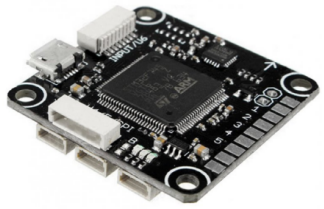


Figure 4. Controller AnyFC F7

D. Overall sensor setup

Goal of project was to minimize number of sensors used maintaining full vehicle ability to perform in competition events.

The camera in charge of road lane perception is mounted in the highest point of car.

Planar lidar sensor and laser distance sensor were provisioned for obstacle detection and high speed braking in drag race.

Magnetic encoder is mounted directly on BLDC motor. It enables actual speed measurement and feeds speed controller.

Additional equipment is gyroscope (anyFC built-in IMU is used). It is used for diagnostics and securing straight ride in drag race (additional guard on vision system)

Overall sensor setup is presented in figure 5.

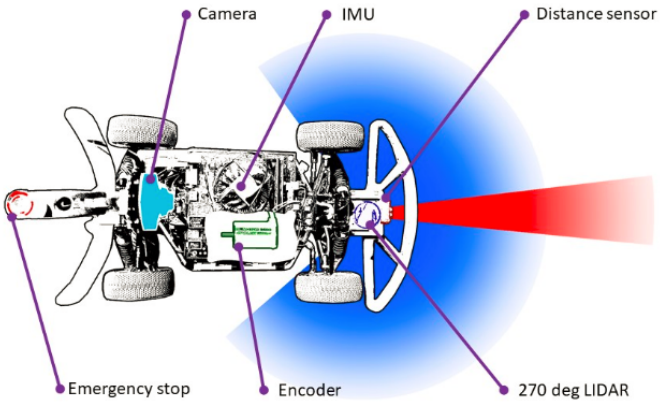
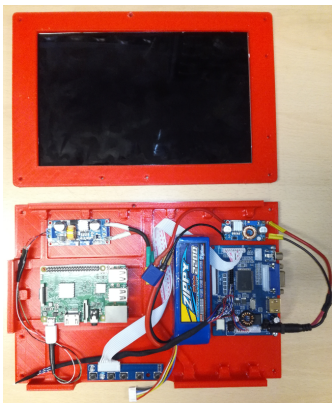


Figure 5. Placement of sensors used in vehicle

E. Human-Machine Interfaces



(a) Portable remote stream viewer

(b) FrSky Taranis - remote controller

Figure 6. Human - Vehicle interfaces used in project

Vehicle communicates with outside world through 3 channels.

2,4GHz Remote Control
: It gives possibility to take over control remotely or just invoke emergency stop. All the communication is transmitted in radio-frequency of 2,4GHz to receiver mounted on car. It is communicating with control unit (AnyFC) using SBUS protocol. Figure 6a presents radio remote controller utilized in this project.

Bluetooth
: Bluetooth communication is designed to enable global variables (battery voltage, driven distance) monitoring and controller terms setting.

Wifi streaming
: Camera captured frames are streamed for testing and presentation purposes. Streaming is not typically used as it is big-size load for computer vision computation unit. For comfortable outdoor testing portable RaspberryPi-based computer was assembled. Figure 6a presents stream viewer device - wired and ready to be closed in 3D-printed box.

IV. Software Architecture

The main assumption on software design was clear role separation between Computer Vision Processing Unit and anyFC board as main controller and communication manager. All the software running on computer and controller is developed in C++ language.

A. Computer

The computer unit is custom PC with Linux Ubuntu system installed. Four programs are prepared to run on computer. First one is capturing camera frames, process them and write coordinate of points lying on lines (edges of the road) to shared memory. Separate application is responsible for fetching coordinates of obstacles detected by planar lidar. They are also written to shared memory. Third application in charge of reading all the points available in shared memory, trajectory planning, deriving current setpoint for speed and steering angle and sending these values to controller (AnyFC - STM32) via serial connection.

B. Controller

High-level control the concept of state machine is realized by controller. Every state involves low-level control operations that need to be performed. Figure 7 presents chart of state machine realized by controller

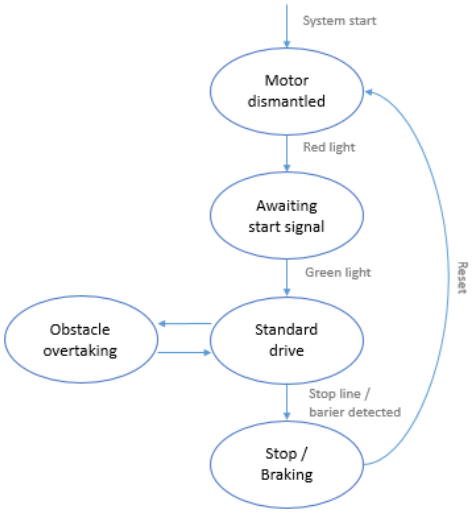


Figure 7. Chart of state machine realized by controller

As in case of autonomous cars driving in real traffic, also in case of small scale agents there is strong demand to operate fulfilling hard real-time conditions. To provide such deterministic and safe behavior Real-Time Operating system is used. Implementation of FreeRTOS, open-source real-time operating system, is installed on STM32 micro-controller. Application running in FreeRTOS integrates and governs gathering of sensor inputs, setpoints from computer unit and setting signals for actuators. It also realizes speed controller loop.

FreeRTOS enables setting priorities for particular tasks. Figure 8 presents hierarchy of tasks realized in design application in regards to priority. The top priority is assigned to receiving information from remote control equipment. It enables to take over the control of the car in every situation. Task responsible for remote battery diagnostics has the lowest priority. If few tasks have the same priority the Round Robin algorithm is responsible for their threads scheduling and time sharing.

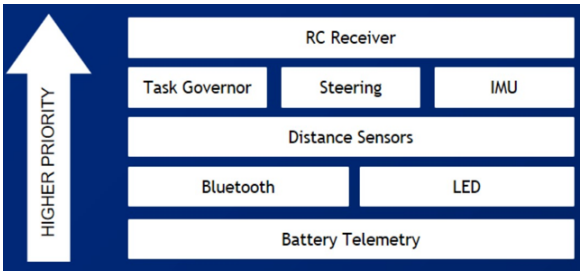


Figure 8. Prioritisation of tasks realized in FreeRTOS application

Implementations of data reading and writing differ across the peripherals used for particular systems element. All of them are using DMA (Direct Memory Access) mechanism to access.

To ensure synchronization of access to data (read/write operations) Interrupts Manager is utilized.

V. Localization

The localization service is provided by image analysis. Position of the vehicle in regard to road edges is being determined. All the computations for this purpose are needed to be done in the real-time drumbeat. They are performed on the computer unit.

3 separate approaches for road lane perception were tested. All of them are based on lines that are marking edges of the road. Methods' description and assessment is provided below, in sections V-A and V-B.

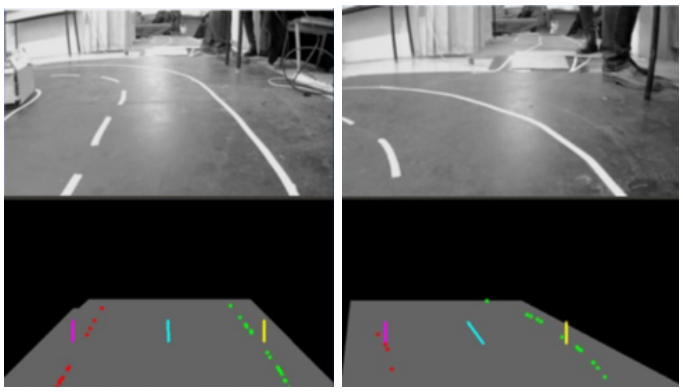
A. Edges slopes analysis

The main task of computer vision application is vehicle localization on road lane and providing information about lane curvature.

In general every captured frame is processed in following steps:

1) Camera frame capturing in YUYV2 format
2) Getting Y channel (gray-scaled image)
3) Applying Gaussian blur for noise elimination
4) Binary thresholding to distinct candidates for lines
5) Applying dynamic mask to shrink Region of Interest basing on lines captured in previous frame
6) Finding short white lines and calculating slope for each one
7) Dividing lines to associated with right and left edge of the road basing on calculated slopes
8) Determining horizontal position of car on road in regards to road edges
9) Calculating slope of the road as average of slope of its edges
10) Sending information to main controller

Figure 9 is presenting input (top) and output (down) frames of algorithm in straight lane 9a and turning lane 9b scenarios.



(a) Straight road scenario (b) Turn scenario

Figure 9. input (top) and output (down) frames of road lane perception algorithm

Above algorithm enables extraction of proper information from captured frames despite the fact that road edge can be continuous, dashed or broken. This image processing method works only in very structured environment - white lines on dark background. As all the control is based on current input frame (no trajectory planning), this approach provides strong results in very deterministic environment, but it is not noise-resistant and fails easily when unexpected object occurs in the frame. It is good way to drive in prepared indoor scenarios, but not in the outdoor world

B. Moving ROIs

To significantly decrease computational load only specific Region of Interests are searched for line. They are defined based on the past captures.

After initial line detection in whole captured frame and receiving start signals (start lights handling is described in section VIII-A), system is working with incoming frames in following way:

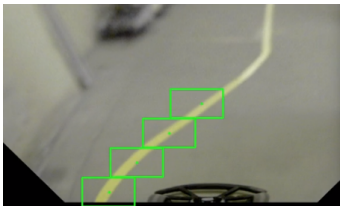
1) Applying Gaussian blur for noise elimination
2) Bird-eye transformation for obtaining the real-scaled road map
3) Color space transformation from RGB to HSL (Hue, Saturation Lightness)
4) Separation of HSL channels
5) Adaptive thresholding for channels (threshold value is obtained for blocks of 20x20 pixels. In daylight the best

results were achieved using S channel to yellow line detection and H or L channel for white one.

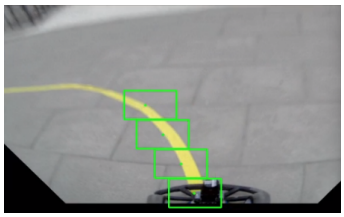
6) Conjunction on thresholded binary frames for noise reduction (only line left)
7) Morphological dilatation
8) For all the ROIs defined from the previous frame:
   - Horizontal histogram calculation
   - Obtaining histogram peak - getting its global coordinates
   - Saving coordinates of histogram peak as center of the ROI for the next frame processing
9) ROIs setup validation - initial check for detected lines consistency
10) Inverse bird-eye transformation (for visualization)
11) Coordinates of obtained points (labeled as yellow or white) are written to shared memory to feed algorithm for trajectory planning.

(a) Input frame (sample 1)

(b) Input frame (sample 2)

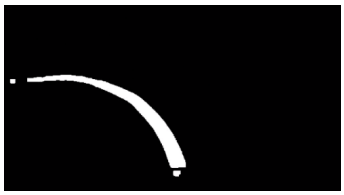


(c) Bird-eyes transformation (sample 1)

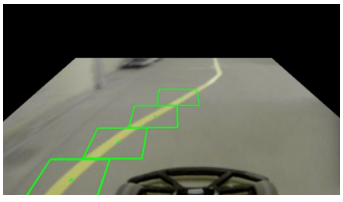


(d) Bird-eye transformation (sample 2)

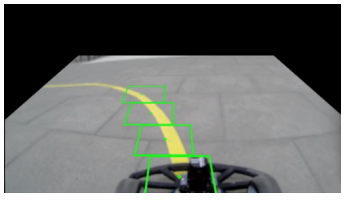
(e) Morphological dilatation (sample 1)



(f) Morphological dilatation (sample 2)



(g) Inverse bird-eyes transformation



(h) Inverse bird-eye transformation

Figure 10. Major steps in line detection algorithm

VI. Trajectory planning

Trajectory planing is realized by separate application running on the computer unit. It takes coordinates of characteristic points written to shared memory by localization application. The data sharing via shared memory is very efficient, well-constrained and generates no latency.

Having coordinates of points on lines and objects detected by lidar sensor, trajectory planning process is in charge of deriving setpoint for speed and a steering angle and sending these values to controller via serial communication.

Map of points is scanned horizontally with rectangular window. When concentration of characteristic points is detected, rectangular region is saved as line containing (pinky rectangles on figure 11). To speed up the whole process, next rows of the frame are scanned only in neighborhood of previously saved rectangular area. Above approach ensure proper filtering out false positives brought to shared memory by computer vision application.

Spline curves are built on centers of rectangular areas to approximate the edges of the road. The final trajectory is defined as the spline built on the center of the vehicle and set of points that are in equal distance from both edges. Typically up to 5 points is enough to draw a reliable trajectory. If one edge of the road is not visible in current camera capture, algorithm is navigating the vehicle in the way to keep the constant distance for one line

To determined current setpoint for steering angle 2 tangents are built: one on the position of the car and the second in some distance ahead. Weighted average of their angle (measured from vertical line) is setpoint for steering angle that is sent to controller unit.

Trajectory planning algorithm is parametrized to be adjustable to testing environment.

Figure 11 presents visualization of trajectory planning and current direction at a given point of time.

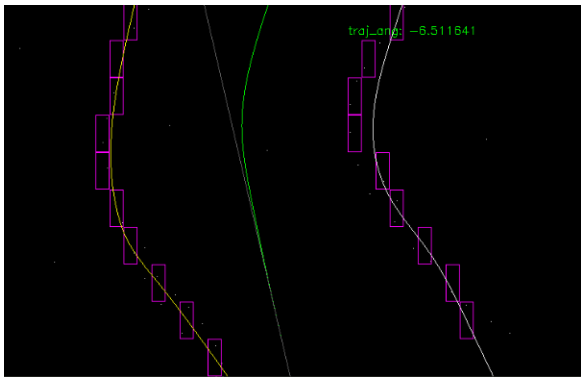


Figure 11. Sample trajectory planning and current direction calculation

A. Collision Avoidance

As one of requirements for circuit race is to avoid physical contact with the competitor vehicles or static obstacles, such functionality was implemented. Looking for the simple and light (in terms of computing load) solution two approaches for collision avoidance were provisioned and tested.

First one is based on planar lidar sensor inputs and optimal trajectory modification. Similarly to points symbolizing road lane borders, coordinates of detected objects provided by lidar sensor are being written to shared memory of processing

computer. Application responsible for trajectory planning takes them as an input and basing on minimum distance assigns them to left of right lane of the road. In this way vehicle is always navigating to drive through the widest slot between obstacle and edge of the road. In testing it occurred that this method works perfectly only in cases with single obstacle. It has problems with multiple objects and due to small range of planar lidar (up to 1 meter after filtering out noise) it is possible to used only with very low speeds. Due to that small range and high level of noise achieved in outdoor testing new solution that is not utilizing planar lidar was needed.

Second approach considered was far more simplified. It utilizes TFmini laser sensor and its operating range of over 8 meters. Idea was to always stick to trajectory that is closer to one road edge than another. After detection of obstacle in front of the car, priority is switched to second edge of the road and the trajectory is drafted closer to it. Similar switch is performed every time potential collision is assumed. This naive method has provided very strong results in the test sessions. Its main advantages are not only simplicity and low computing load, but also possibility to operate in high speeds due to sensor long range and high frequency (100 Hz). It is worth to mention that this solution could strongly decrease whole hardware costs, because it is based on cheap one dimensional sensor (TFmini - $40).

VII. High-speed control

A. Speed control

One of threads defined on STM32 controller is responsible for the speed control. It is periodically invoked by CPU on controller board (AnyFC - STM32). It realizes PID controller taking setpoint received from trajectory planner. Actual value is calculated based on magnetic encoders output. Encoder with proper filtering environment is mounted in the car. Source of magnetic field are magnetic rings mounted directly on the drivetrain shaft. With this setup ring is generating 10240 impulses per one shaft rotation. Due to gearshift existence, it gives even better resolution in terms of number of impulses per one rotation of the wheel. In iterational tuning it was possible to determine the proper terms for the controler. Figure 12 presents how speed of 2 $\frac{m}{s}$ in time of 0.7 second.

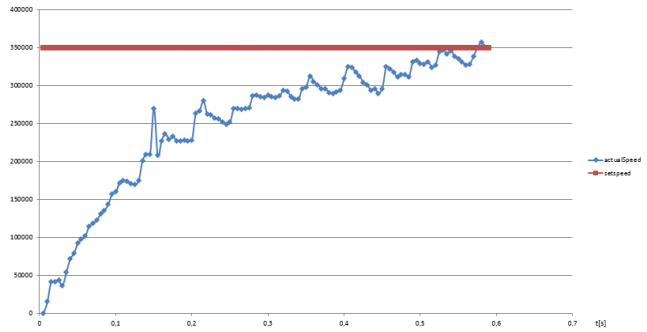


Figure 12. Response of speed controller for unit step (setpoint: $2\frac{m}{s}$)

VIII. Selection of other solutions

A. Starting lights handling

Starting lights handling is based on a differential algorithm. Vision system is sending start signal to vehicle controller immediately after recognition of significant difference between to consecutive frames. To avoid impact of background noise only selected Region of interest (where traffic lights are expected) is analyzed. Number of pixels that need to be perceived as changed is thresholded to filter out false positive signals and limited to avoid reaction for externally caused camera vibration.

The best result and repeatability was achieved during analyzing pixels values in HSV space. Recording changes in lightness channel it possible to detect simultaneous change of values for pixels representing both red and green light areas. When above scenario is recognized, start signal is send to car controller and race begins.

B. Attitude and heading reference system

o komunikacji z układem służy interfejs SPI1. Obsługę MPU6000zrealizowano w oddzielnym wątku GyroTask. Wątek napisany jest w sposób następujący:

IX. Conclusion

The designed systems are successfully implemented in vehicle. Initial assessment and testing suggest that vehicle is ready to present satisfying performance during competition. Team focus was put on building software for off-shelf hardware to not recreating well known electronic circuits. Simplicity of algorithm and repeatability of it results were the most important criteria in every conceptual decision.

In both design and implementation phase a lot of obstacles were overcome. Team members learned a lot not only in their technical domains, but also in area of system designing and team work coordination. It is irreplaceable experience. As key learning it is worth to mention importance of detailed definition of responsibilities assigned to particular system components, its input and outputs, Also how important is to optimize computation load to enable using more complex algorithm in high-speed operating robots.

Appendix A

Table I presents detailed list of costs incurred during building vehicle.

Table II lists all the people involved in this project. The team consist of 10 student members: 7 undergratudates and 3 graduates. They are students of Computer Science, Mechatronics and Robotics from 4 different faculties of Warsaw University of Technology.

Acknowledgment

Authors are members of Students' Robotics Association (Koło Naukowe Robotyków) at the Faculty of Power and Aeronautical Engineering, Warsaw University of Technology.

Research done cooperating with the Faculty of Power and Aeronautical Engineering within the framework of "Najlepsi z Najlepszych 2.0!" program.

Table I
COST OF VEHICLE COMPONENTS

Element	Model	Cost
Chassis	Associated Team RC8B3.1e	$500
BLDC Motor	Reedy Sonic 1512 1800kV	$150
Servo Motor	Sevox SC-1258TG	$60
ESCS (controller)	XERUN XR8-Plus	$140
Motherboard	ASRock H110M-ITX	$60
Processor	Intel Core i3-7100T, 3.4GHz, 3MB	$120
Memory	GoodRam DDR4, 4GB, 2.4GHz, CL12 + SSD Slicon Power A55 64GB	$75
Camera	Kurokesu C1 / IDS UI-1220LE	$100 / $330
Autopilot	AnyFC F7	$45
Laser sensor	TFmini	$40
Lidar	Hokuyo URG-04LX-UG01	$1000
Remote controller	FrSky Taranis Q X7	$120
Batteries + wiring	LiPol 4S	$40
Mechanical parts	ABS for 3D printing, metal, carbon fibre, screws	<$20
	SUM	**$2470 / $2700**

Table II
TEAM MEMBERS

Name	Role
Mateusz Perciński	Project Lead
Mikołaj Marcinkiewicz	Finance, Integration
Mateusz Grudzień	Computer Vision, Computing
Maciej Krasa	Computer Vision
Kornelia Łukojć	Computer Vision
Mateusz Marczuk	Trajectory Planning, Control Systems
Mateusz Mróz	Embedded Software, Control Systems, Electronics, Integration
Michał Jarzyński	Embedded Software, Control Systems
Krzysztof Gawin	Mechanics, 3D Printing
Łukasz Jakubowski	Mechanics, 3D Printing
Ph.D. Krzysztof Mianowski	Assistant Professor - Scientific Supervisor

REFERENCES

[1] B. S Khan, M. Hanafi, S. Mashohor *Automated Road Marking Detection System for Autonomous Car*. 2015 IEEE Student Conference on Research and Development (SCOReD), 2015, pp. 398 - 401

[2] Shimil Jose, Sajith Variyar V V, Soman K.P *Effective Utilization And Analysis Of ROS On Embedded Platform For Implementing Autonomous Car Vision And Navigation Modules*, 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 877 - 882

[3] A. Iqbal, S. S. Ahmed, M. D. Tauqeer, A. Sultan, S. Y. Abbas *Design of Multifunctional Autonomous Car using Ultrasonic and Infrared Sensors*, 2017 International Symposium on Wireless Systems and Networks (ISWSN), 2017, pp. 1 - 5

[4] M. Martinez, A. Roitberg, D. Koester, B. Schauerte, R. Stiefelhagen *Using Technology Developed for Autonomous Cars to Help Navigate Blind People*, 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 1424 - 1432

[5] G. S. Pannu, M. D. Ansari, P. Gupta *Design and Implementation of Autonomous Car using Raspberry Pi*, International Journal of Computer Applications (0975–8887) Volume 113-No.9, March 2015

[6] T. Wankhade, P. Shriwas, *Design of Lane Detecting and Following Autonomous Robot*, IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661 Volume 2, Issue 2 (July-Aug. 2012), pp. 45-48.

[7] X. Miao, S. Li, H. Shen, *On-Board lane detection system for intelligent vehicle based on monocular vision*, International Journal on Smart Sensing and Intelligent Systems, vol. 5, no. 4, December 2012, pp. 957-972.

[8] S. Tuohy, D. O'Cualain, E. Jones, M. Glavin, *Distance determination for an automobile environment using inverse perspective mapping in OpenCV*, Irish Signals and Systems Conference 2010.