Autonomous **L**earning **I**ntelligent **V**ehicles **E**ngineering (ALIVE 1.0)

Jihene Rezgui, Émile Gagné and Guillaume Blain
Laboratoire **R**echerche **I**nformatique **M**aisonneuve (LRIMa)

Montreal, Canada
jrezgui@cmaisonneuve.qc.ca

Abstract—The increasing number of vehicles on roads brings more risks associated with vehicular travel. Nevertheless, with the massive attraction towards self-driving vehicles and the use of artificial intelligence, a trained physical Autonomous Vehicle (AV) is now a major part of transports future. This paper discusses the limitations of the related research based on autonomous vehicles; particularly those who are not taking into account the real-world physics. It also proposes an Autonomous Learning Intelligent Vehicles Engineering, called ALIVE to let each vehicle have additional information about its surroundings in order to get an extended perception of its environment. Moreover, ALIVE car sensors will gather in real-time the required data concerning the vehicles environment which are fused into a learning algorithm predicting the vehicle's response. We tested our algorithm through different mazes to evaluate its efficiency to avoid obstacles and its capacity to adapt to any type of terrain. This has been done to make ALIVE versatile, open source, low-cost and work in any environment. Preliminary results demonstrate the effectiveness of ALIVE in terms of obstacle avoidance and delay minimization. Besides, we hope that our project can be used by other researchers to test their artificial intelligence in the real world instead of keeping it in a simulation.

Keywords—IoT, AI, Arduino, Sensors, Self-Driving, Physical car, Obstacle avoidance, Wireless Communication.

I. INTRODUCTION

Nowadays cars are getting more and more reactive to their environment. In fact, the production of connected vehicles drastically increases every year and they become more and more intelligent. Those developments and advances in modern technology are what motivated us to develop our own cars and algorithms on a reduced scale in order to make the whole project flexible, physical, affordable and adaptive. The whole **ALIVE** car can be changed and adapted to any environment or any car skeleton as it does not rely on the recognition of specific colors or specifically made signs for the project, like some other researchers have done so far, as it relies on enormous amounts of training and algorithm optimization based on a variety of sensors giving the most possible information to the vehicle to allow it to know its environment to the best of its capabilities.

It is worth noting that the Learning Algorithm Optimization Platform LAOP and the Learning Algorithm Sharing Platform LASP [1-2] help researchers in the field of deep learning to develop their models by allowing them to easily test and compare them. The **ALIVE** project [3] aims to be fully trained to any circumstances and to any situation that could come up and it is expected to respond in the most human way possible, if not better.

With our **ALIVE** research project, we have achieved improvements on cars and their algorithms discussed in prior research in ways that are explained in the following section.

Our contributions in this paper can be summarized as follows: (1) We introduce our physical car and it's corresponding artificial intelligence algorithm, called "ALIVE"; (2) We explain related researches that contributed to our learning and gave us an idea of what was already made and what still had to be explored; (3) We highlight the effectiveness of the improvements made with ALIVE compared to the other researches that have been published before on the subject; (4) We couple the ALIVE car with the artificial intelligence part of the project instead of the onboard algorithm itself and (5) We provide open directions towards future possibilities to continue the project.

Section II provides a brief overview of the related work and compares them to our **ALIVE** project. In section III, we describe our whole platform and how it functions. In section IV, we present two algorithms, one coupled with our artificial intelligence and our preliminary results. Section V provides different solutions and propositions to extend the project and go further. Section VI concludes the paper.

II. RELATED WORK

This section explains different research projects and it compares them to ours by illustrating their main differences and showing what are the advantages and disadvantages in having those different characteristics and how seeing the other project influenced us and continues to give us ideas for the near future.

A. Duckie Town Project vs ALIVE project

First, we have found a similar study case named "Duckie Town", a study conducted by MIT researchers [4], which is a study aimed at providing easy to use and open source software and hardware for researchers to interact with the real world environment in cars and artificial intelligence. It is based on solely one camera and it has an on-board Raspberry 2 running all their processes.

The main difference is that the Duckie Town project works with images while ALIVE works with ultrasound sensors. We must admit that having a camera can bring different advantages compared to the sensors but also eliminates a lot of possible

978-1-7281-5628-6/20/$31.00 ©2020 IEEE

image interpreting errors by using physical sensors instead of image recognition algorithms. Although it does reduce the scope of possibilities since we cannot really interact with any road signs or the likes and our system is solely based on obstacle detection and avoidance, we have not excluded the possibility of adding an onboard camera to give more information and data to our algorithms. We plan on inspiring ourselves from that project by indeed adding a camera in the near future as the main detection tool and having the ultrasounds as complementary tools and data providers. That combination of tools could give us the opportunity to verify and cross examine all of our data as a whole and the sensors could give us good tips on how to interpret the data from the camera regarding distances and objects. We believe the addition of the camera to our current car would greatly increase the accuracy and would allow us to push it seriously further by adding some image recognition software and algorithms and starting to recognize different patterns such as road signs, traffic lights or just simply lines on the road that delimits the border of the street.

B. LAOP Project vs ALIVE project

In LAOP platform, in order to train the neural networks, the authors need to simulate cars in an environment that is as close as possible to the real world in order to be able to transfer the algorithm to a real car when it is done learning. We will later on compare: A) The machine-made algorithm (MMA) that only reacts to certain pre-set rules and B) Our custom-made AI we developed and trained using the ALIVE car. We had the idea for the project when we realized that most of the projects like LAOP do not really test their results and we have no clue if those researches would actually be true in a real world environment with all the variables that we might not account for while coding. By having a physical car, we add an important part that allows us to verify our algorithms with all variables accounted for.

C. Other Projects vs ALIVE project

Table.I Comparison of Other Projects vs ALIVE project

PRODUCT	COST	MODULAR HARDWARE	OPEN SOURCE SOFTWARE	MOBILE APP	PATH PLANNING	AVOIDANCE	TRACKING	DECENTRALIZED COORDINATION	VISUAL SERVOING	2D VISUAL REPRESENTATION OF HIS ENVIRONMENT	ULTRASOUND DETECTION
AEROBOT [5]	$20	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗
KILOBOT [6]	$50	✗	✓	✗	✓	✗	✓	✓	✗	✓	✗
3PI [7]	$100	★	✗	✗	✗	✓	✗	★	✗	✗	✗
ALIVE-BOT [3]	$105	★	✓	✓	✗	✓	✓	✓	✗	✓	✓
JASMINE [8]	$120	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗
LABRAT [9]	$120	✓	✓	✗	✗	✓	✗	✓	★	✗	✗
THYMIO II [10]	$130	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗
DUCKIEBOT [4]	$135	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
SCRIBBLER-3 [11]	$150	✓	✓	✗	✓	✓	✗	✗	★	✗	✗
BOE-BOT [12]	$180	✓	✓	✗	★	✓	★	✗	✗	✗	✗
ACTIVITYBOT [13]	$180	✓	✗	✗	★	✓	★	✗	✗	✗	✓
CREATE-2 [14]	$200	★	✗	✗	★	✓	✗	✗	★	✗	✗
BOT'N ROLL [15]	$200	✓	✓	✗	✗	✓	✗	★	✗	✗	✓
R-ONE [16]	$250	✗	✓	✗	✗	✓	✓	✓	✗	✗	✗
HEMISSION [17]	$250	★	✗	✗	✗	✓	✗	✗	★	✗	✓
PHEENO [18]	$270	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗

We compared our whole project (including the car and the platform) to various similar projects to better understand where our platform situates itself compared to the research already completed. This comparison table puts the project in perspective and aims at giving the project some guidelines to develop some features the other bots do not have. The comparison is illustrated in Table.I on the following.

III. Autonomous Learning Intelligent Vehicles Engineering (ALIVE)

Our project is the result of two different expertise fields. We decided to combine both electronics and programming to create a full-scale project that can be both appreciated and useful to mechanical engineers but also artificial intelligence programmers.

A. Architecture of ALIVE Platform

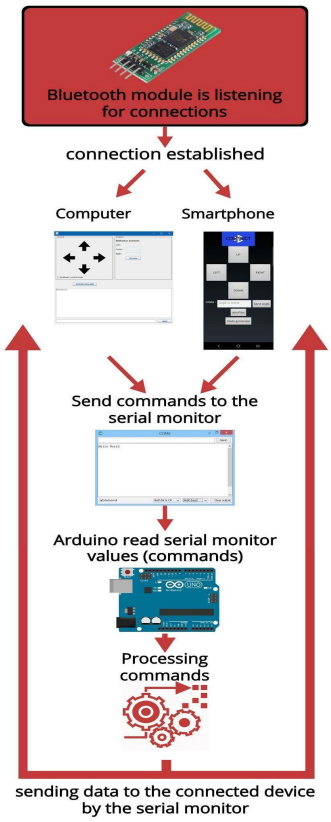

Fig.1 Architecture of ALIVE Platform

The whole architecture starts with the establishment of a Bluetooth connection between either a computer or a cellphone to the Arduino Bluetooth onboard module. Once the connection is established, the selected communication tool (Computer or cellphone) will be used to transmit commands and information through the Serial Monitor of the Arduino as shown in Fig.1. It will then interpret those commands and process them to make the car react accordingly to the decisions made by the computer

(in a scenario where the algorithm or AI is not onboard, otherwise the Bluetooth connection is mainly used to start the car and to activate the onboard autopilot while receiving the sensors data). Once the car did the proper manoeuvre following the reception of the command, it will then transmit the current onboard information back to the computer in order to let the computer make another decision.

B. ALIVE car characteristics

The ALIVE car is equipped with a L298N Dual H-Bridge to control 2 DC motors in a simpler and more effective way as shown in Fig.2.

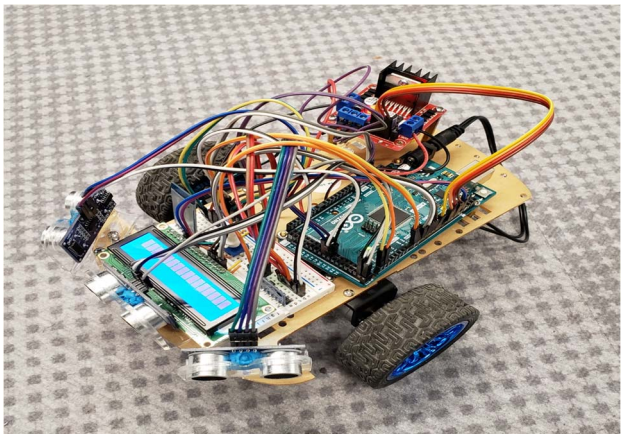

Fig.2 ALIVE car

This module is the main link we have between the Arduino microcontroller which manages the whole physical side of the project and the car motors.

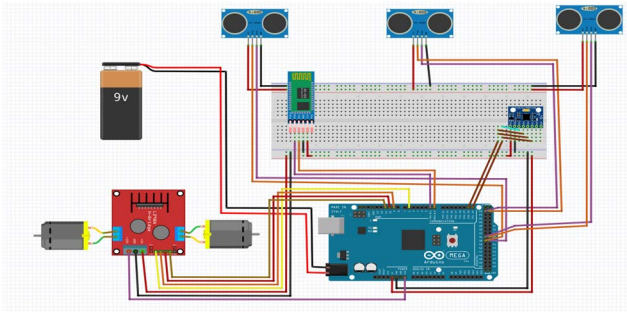

Fig.3 ALIVE Schematics

In fact, the Arduino is the core of the car and all the different modules are linked to it. Those modules are : 3 Ultrasonic sensors (HC-SR04), 1 Bluetooth module (HC-05), 1 L298N Dual H-Bridge, 1 accelerometer/gyroscope, an LCD Panel on the main vehicle, which is extremely useful to see the data in real-time on the car, and the core, an Arduino Mega. The reason behind the choice of the "Mega" model for the Arduino is that it has more pins available to connect the modules and it offers a simpler approach regarding the wiring as most of the pins on it are PWM (Pulse-Width Modulation) which are required in order to control the H-Bridge and the motors without problems. The whole car is powered using a 9 volts and 600 mA rechargeable battery and it is enough power for the car to be effective and responsive for an approximate of 30 to 45 minutes. The ALIVE car parameters are described in Table.II. All those components are wired as shown in Fig.3.

Table.II Parameters details of ALIVE car

System slot	Component	Cost (rounded)
Computation	Arduino Mega	20$
Frame + Motors	Chassis	20$
Motor control	L298N Dual H-Bridge	18$
Detection	Ultrasonic Sensors HC-SR04	2.30$ * 3 (7$)
Power	9V Battery (Rechargeable)	25$
Communication	Bluetooth module HC-05	14$
Total:	-	104$

***This is the cost for the prototype. If it were ordered in considerate quantities, the price would likely hover around ~70-80$.**

C. ALIVE Platform results and utilities

The choice for the Bluetooth module was motivated by the objective that we wanted fast, reliable one-on-one two-way communication. The other main option we considered was using Wi-Fi, but we came to the conclusion after thorough research that the Wi-Fi technology was not available everywhere (e.g. in the street or in any other outside environment) and that it required the current location to have a modem, router and all the required equipment for Wi-Fi. When that came to light, we realized our best option would be Bluetooth because it had about the same problems as the Wi-Fi (i.e. not practical for outside use because of the short range of Bluetooth) but the main difference is that with our current technology, it offers an easier gateway to the transition towards cellular data connection. We kept that in mind because we now know that the 5G technology developed by the main service providers across the world are working on will come out in the near future and we thought it would be a great opportunity to do the transition when the technology does come out.

We also thought about using the DSRC technology that would allow the communication between the different cars and also between the cars and the infrastructure but we quickly realized that it seemed like a deprecated technology as most automobile constructor have left it out instead of trying to use it. It also seemed as if there were major security problems linked to that technology which confirmed our decision to stay away from that specific technology.

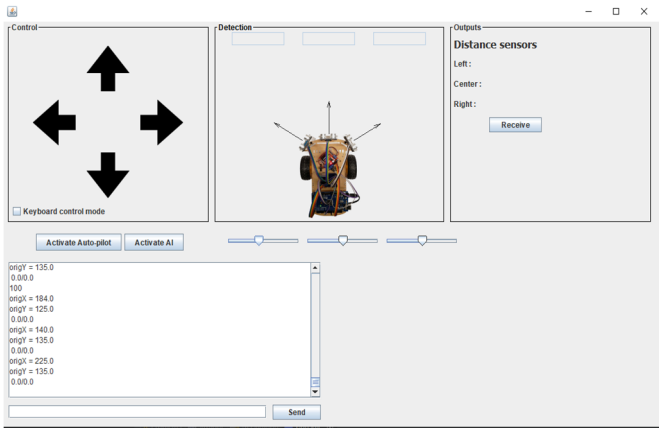

Fig.4 ALIVE Java application

We incorporated the "intelligent" adjective in our project description not only because we have implemented a complete AI in our car's infrastructure but also because we implemented different types of algorithms in the car via the Bluetooth communication and a computer running a Java coded application as shown in Fig.4.

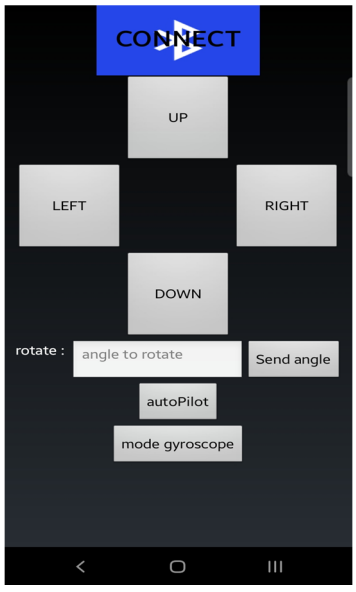

Fig.5 ALIVE android application

It is important to note that there is also a basic Android application available for debugging, testing and plain entertainment to control the car as shown in Fig.5.

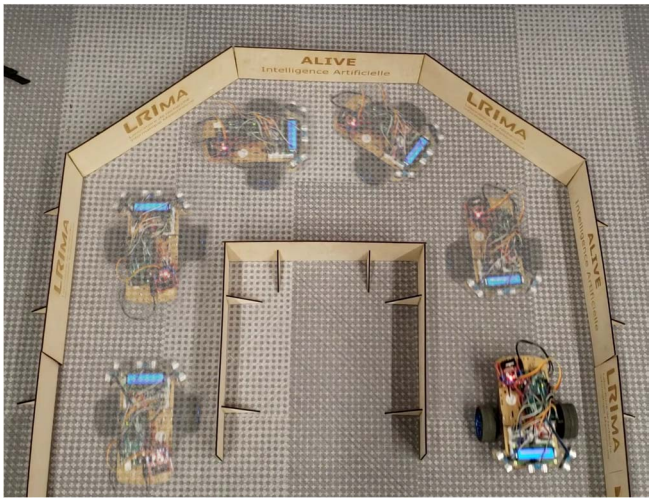

Fig.6 ALIVE Car in action

It is useful as it gives another platform to control the car from and it gives us a certainty of the portability of the project. The only downside in controlling it with a cellphone acting as a remote is that there are no AI that will be acting in the decision taking process of the car. We also added an interesting function where we create a 2D Map of the actual environment of the car. The map creates and displays itself by putting dots where the car **detects walls** and it places the car as a red moving dot. All that is made without GPS localization and is only based on physics and mathematics calculations considering acceleration, maximum speed and the angle the car is at compared to his starting angle. Finally, our AI performed well in avoiding collisions with walls as it responded well to the sensors data and was able to give our car the correct decisions almost all the time as shown in Fig. 6. The ALIVE car can avoid walls and get out of a maze without any problem.

IV. ALIVE algorithms

A. Obstacle Detection Algorithm

This obstacle detection algorithm currently developed and mounted in the Arduino module is our custom-made algorithm is based solely on the distance detected by the ultrasound detectors. Those detectors send a short signal of ultrasounds ahead of them and we calculate the distance between the detector and the closest object by multiplying the travel time of this wave by the travel speed of the sound, divided by two as following:

$$d = \frac{\tau \times 0.034}{2} \qquad \text{eq.1}$$

This result brings us the current distance between the sensor and the first object if encounters, in centimeters. Divided by two is there because the wave has to be sent and come back, which means it travels twice the actual distance that we want to know, hence the division. Once we have this distance, we can enable different conditions as to what speed should the motors be turning at and which motors should be active. (e.g., the right wheel clockwise and the left wheel counter clockwise will induce a turn to the left). This is how the basic controls of the car works and it's his core. We first check the current distances of the 3 different sensors and then we evaluate. If one of the sensors detects an object closer than a few centimeters away from him, we raise a flag and indicate the car to turn in the opposite direction until the object is no longer detected and the path is free. With that in mind, we did a lot of tests with different conditions to see which ones would suit the car and we came to the conclusion that we had to know the exact speed of the car in order to adapt the mathematics of the algorithm. We calculated the speed and it gave us a speed of 0,32 m/s. This was the maximum we could attain with our current hardware and our batteries. We moved on to create new algorithms, more adapted to the speed and dimensions of our vehicle and we developed a second version of our first algorithm (see Algorithm. I) which gave some valuable results, giving the car the ability to avoid almost any shape of obstacle in a closed-circuit different shape of walls, although mainly plane and straight walls. The vehicle was able to go through narrow paths and to angle itself right when coming at straight walls or even corners. The logic behind it mostly resorts to going in the most open direction, i.e. the direction from which the sensor gets the longest distance.

Algorithm I. Auto Pilot
1. Java Application launches a command to the car to start the Auto Pilot Mode 2. Car receives command and loops in to test the sensors 3. If the sensor detects a distance < 10 cm, it is considered, and a decision is made based on the other sensors too 4. Java Application asks for sensors data to the car 5. Car sends back the data about angle, distances, etc. 6. Java Application displays the situation on a 2D Map

B. Artificial Intelligence Algorithm

When we built this project, we always had in mind to make it easy to use for others so they could test their own algorithms but we realized that giving examples of incorporation of algorithms is the best way to show others how to do it on their own. This is the reason why we added a general, easy-to-implement AI in our project and tested it on our cars, as illustrated in Algorithm. II. We wanted to give the researchers who plan on reusing our research to test their experiments on physical cars to just base their algorithm shape on our AI as it works extremely well and it's optimized to handle the inputs the best way possible and to give outputs in a simple and readable format for the car. It is hard to compare the pure data from LAOP to our own algorithm as they are genuinely different in so many ways. They do not learn the same way at all nor do they give similar outputs. In fact, our AI was trained on a large modular dataset that was created using our premade algorithm and it used two different activation functions, RELU and SoftMax [eq.2].

SoftMax:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad \text{eq.2}$$

Where $\boldsymbol{\sigma}(z)$ is the output vector, j is the index, k is the iteration variable.

We gave him the answer our algorithm would've taken and we let him try to understand what all those decisions correlated with the sensors data and the AI got to a point where it performed even better than the algorithm because it would try to find a solution in narrow situations rather than just back out of it as we instructed our algorithm to do in case everything else failed. The AI could be better trained if we trained the neural network differently such as using a genetic algorithm. This change would allow the car to get a better understanding of its environment and to develop new answers and new decisions that we could not predict in our algorithm which is one of the huge advantages that differences our AI and the premade algorithm that runs onboard.

Algorithm II. Artificial Intelligence Guidance
1. Java Application launches a command to the car to start the AI Mode 2. Car receives command and sends the sensors data 3. Java Application receives the car data and analyzes it by putting it through the AI 4. The AI gives a decision based on its training and the data 5. Java Application transmits the decision to the car 6. Car acts according to the decision

V. Hardware upgrade possibilities

A major upgrade that the car could benefit from would be going from an Arduino to a Raspberry Pi because this platform offers a better processing power and an operating system so we could run the artificial intelligence program locally on the car instead of doing it remotely by a computer which reduces the reaction latency. Another upgrade we did was going from 120

RPM motors and wheels to 280 RPM motors and wheels. This has led to some interesting discoveries as it increased the steadiness of the car and the overall feel. Finally, we considered using a similar platform, halfway from Arduino to Raspberry Pi, called Tipot [19]. That platform is similar to Raspberry as it has an onboard operating system but it is similar to Arduino as it is made and developed to automate tasks and let it run coupled with sensors. The main advantage of it is that the different "motes" can be linked together simply through low-frequency radio waves. It saves battery and facilitates the creation of Internet of Things meshes. It could really come in handy for us to connect the different cars together and communicate about the different walls and obstacles it has seen to the other cars in order to correlate that data after a few cars pass by there.

VI. CONCLUSION

Finally, project ALIVE proposed a learning algorithm which we compared to several other similar projects. We demonstrated that our project outperforms those projects in terms of cost, environment recognition and decision taking. We are also the only project we found that provides a working mobile app to help debugging the car. The whole project was made open source to make it easier for other academics to inspire themselves from our car and platform and use it to test their own algorithms in the real world. However, we still presented open directions to further develop on the hardware and algorithm side, but we also plan to go further by adding other algorithms to further compare our results.

ACKNOWLEDGMENT

This research was financially supported by the "Fonds Québécois de la recherche sur la nature et les technologies (FRQNT)."

REFERENCES

[1] J. Rezgui, C. Bisaillon and L. Oest O'Leary, "Finding better learning algorithms for self-driving cars", in IEEE ISNCC 2019 18-21 June, Turkey.

[2] J. Rezgui, L. Oest O'Leary, C. Bisaillon and L. Chaari, "Training Genetic Neural Networks Algorithms for Autonomous Cars with the LAOP Platform", in IEEE IWCMC 2019 Tangier, Morocco.

[3] ALIVE prototype by Émile Gagné, Guillaume Blain and J. Rezgui on GitLab, https://github.com/Emilex108/ALIVE [last visit 05/01/2020: open source code].

[4] Duckietown team, "Duckietown: an Open, Inexpensive and Flexible Platform for Autonomy Education and Research", in 2017 IEEE International Conference on Robotics and Automation (ICRA) Duckietown: an Open, Inexpensive and Flexible Platform for Autonomy Education and Research

[5] M. Rubenstein, B. Cimino et al., "AERobot: An affordable one-robotper-student system for early robotics education," in IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 6107–6113

[6] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in 2012 IEEE International Conference on Robotics and Automation. I

[7] B. Thursky and G. Gaspar, "Using Pololu's 3pi robot in the education ˇ process." [Online]. Available: http://tiny.cc/zfluey

[8] S. Kernbach, "Swarmrobot.org - Open-hardware microrobotic project for large-scale artificial swarms," arXiv preprint arXiv:1110.5762, 2011. [Online]. Available: http://arxiv.org/pdf/1110.5762v1.pdf

[9] P. Robinette, R. Meuth et al., "LabratTM: Miniature robot for students, researchers, and hobbyists," in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2009, pp. 1007–1012.

[10] F. Riedo, M. Chevalier et al., "Thymio II, a robot that grows wiser with children," in 2013 IEEE Workshop on Advanced Robotics and its Social Impacts. Institute of Electrical & Electronics Engineers (IEEE), nov 2013. [Online]. Available: http://dx.doi.org/10.1109/arso.2013.6705527

[11] P. Inc. (2016) Scribbler s3 robot. [Online]. Available: https: //www.parallax.com/product/28333

[12] R. K. Cole, "STEM outreach with the Boe-Bot," Robots in K-12 Education: A New Technology for Learning: A New Technology for Learning, p. 245, 2012.

[13] Parallax. (2016) Activitybot robot kit. [Online]. Available: https: //www.parallax.com/product/32500

[14] M. Dekan, F. Duchon˘ et al., "iRobot create used in education," Journal of Mechanics Engineering and Automation, vol. 3, no. 4, pp. 197–202, 2013.

[15] Bot'n Roll. (2016) Bot'n roll ONE A. [Online]. Available: http://botnroll.com/onea en/

[16] J. McLurkin, A. McMullen et al., "A robot system design for low-cost multi-robot manipulation," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014, pp. 912–918.

[17] Robots in Course. (2016) Hemission. [Online]. Available: http: //www.robotsinsearch.com/products/hemisson

[18] S. Wilson, R. Gameros et al., "Pheeno, a versatile swarm robotic research and education platform," IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 884–891, July 2016.

[19] TIPOT TECHNOLOGIES INC. (Corporation# 9396276) is a federal corporation entity registered with Corporations Canada. The incorporation date is August 5, 2015. [https://www.tipottechnologies.com/ :last visit 05/01/2020].