Project-based, Collaborative, Algorithmic Robotics for High School Students: Programming Self-driving Race Cars at MIT

Sertac Karaman, Ariel Anders, Michael Boulet, Jane Connor, Kenneth Gregson, Winter Guerra, Owen Guldner, Mubarik Mohamoud, Brian Plancher, Robert Shin, John Vivilecchia

Massachusetts Institute of Technology

***Abstract* – We describe the pedagogy behind the *MIT Beaver Works Summer Institute Robotics Program*, a new high-school STEM program in robotics. The program utilizes state-of-the-art sensors and embedded computers for mobile robotics. These components are carried on an exciting 1/10-scale race-car platform. The program has three salient, distinguishing features: *(i)* it focuses on *robotics software systems:* the students design and build robotics software towards real-world applications, without being distracted by hardware issues; *(ii)* it champions *project-based learning*: the students learn through weekly project assignments and a final course challenge; *(iii)* the learning is implemented in a *collaborative fashion*: the students learn the basics of collaboration and technical communication in lectures, and they work in teams to design and implement their software systems. The program was offered as a four-week residential program at MIT in the summer of 2016. In this paper, we provide the details of this new program, its teaching objectives, and its results. We also briefly discuss future directions and opportunities.**

Index Terms – High school STEM education; hands-on robotics; project-based learning; team-oriented learning.

I. Introduction

Robotics is a thriving emerging field that already has had tremendous impact. The future of robotics is even brighter: Self-driving cars may revolutionize transportation by bringing down costs by an order of magnitude, while substantially enhancing safety, perhaps even eliminating fatal traffic accidents; Aerial drones may finally enable affordable same-hour delivery of goods, transforming the way we shop; Autonomous underwater vehicles may explore our oceans, while robotic rovers roam around in Mars and beyond. These robotics applications and many more are expected to unfold during the next few decades. Those who will build these robotic systems are studying in high schools today. *How can we best prepare them now so that they can build this future?*

It is widely accepted that robotics is an exciting direction for high school students [1], and robotics competitions, camps, and clubs all increase interest in Science, Technology Engineering and Mathematics (STEM) [2], [3]. Many high-school robotics programs are organized as competitions [4]. Among the best known are the FIRST Robotics Challenge, the FIRST Lego League [5], BEST Robotics [6], National Robotics Challenge (NRC), and EARLY Robotics. These programs are all tailored for pre-college students, spanning a large variety of robotics hardware for students to build and utilize. For instance, on the one end of the spectrum, students build hundred-pound robots for the FIRST Robotics Challenge. Teams with budgets beyond $15,000 are fairly common [4]. On the other end of the spectrum, the BEST Robotics Competition provides all hardware material required for a small fee of $500 for the students to build palm-size robots. Independently of the cost of the platforms, almost all existing robotics competitions for high school students require the students to focus on building and integrating the hardware that make up the robot, at the expense of designing and implementing complex algorithms and software.

The emerging applications of robotics are much more complex for several reasons. Firstly, they are software heavy. In fact, most advanced robotics applications are enabled by software that is built from hundreds of thousands of lines of code that is professionally implemented, often in an object-oriented, low-level programming language, such as C++. The software is often so large and so complex that a large team of robotics engineers and software engineers implement it together. Hence, it is essential to *design the software system* given the project requirements, and *implement it as a team utilizing essential software collaboration tools.* Second, in most modern robotics systems applications the *sensor data is massive in size, unstructured and noisy*. Furthermore, for most robotic systems, there is not one sensor that the robot solely relies on; often, data from multiple sensors (*e.g.*, cameras, laser range finders, inertial measurement units, global positioning system, and more) are utilized together. Hence, algorithms and software for proper *sensor fusion* is key to making most of emerging applications of robotics.

Most existing high-school computer science and robotics programs are unable to address these challenges on their own. On the one hand, existing robotics programs focus on the mechanics; as a result, they do not have room for students to design and implement relatively complex software systems, as noted above. On the other hand, most existing computer programs are fairly conceptual, and they often do not work with real-world data and not consider real-time processing.

In this paper, we describe the pedagogy behind a new program, called the *MIT Beaver Works Summer Institute Robotics Program*. The new program complements the existing computer science and robotics programs as follows.

First, the new program focuses entirely on robotics software. Students do not engage in hardware design or development. They are given a hardware kit that includes state-of-the-art sensors and computers, including a scanning laser range finder, a high-resolution stereo camera, and an Nvidia Jetson Tegra X1 embedded computer with a 256-core General-Purpose Graphics Processing Unit (GP-GPU). The whole sensing and computation platform is fitted on 1/10-scale race car, making it an exciting platform for students.

Second, the new program is project based. Students build their skills through several "mini projects," each of which require the design of a relatively complex software and its implementation. Lectures and laboratory exercises are tailored to help the students think towards their projects. The program also features a final course challenge, which requires the students to design and implement robotics software for fully-autonomous racing. The fastest car that abides by the rules of the road wins! We believe project-based learning that involves exciting projects helps motivate the students and improves their system-level thinking and system design skills, in this case focusing on software systems.

Third, the program champions collaboration. The students work in teams for their projects. In each project, the students design the software system as a team. They partition their implementation into Robot Operating System (ROS) nodes, often each student implementing more than one ROS node. They connect the nodes utilizing the ROS messaging environment. The students also utilize the software version tracking tools, which are common in large software engineering projects in the robotics industry. Given the substantial amount of collaboration involved in the course, we have implemented lectures and instruction that teaches students effective communication and collaboration.

The program was implemented as a four-week residential program at the Massachusetts Institute of Technology (MIT) in the summer of 2016. A total number of 46 students were invited to the MIT campus for the program, 22 of which came from Massachusetts while the remaining 24 came from across the U.S. and stayed in the Boston Area during these four weeks. We devote this paper to a detailed description of the pedagogy of this program and its results.

Most lectures and laboratory exercises are modeled after MIT's flagship robotics course, taught by Karaman, entitled *Robotics: Science and Systems*, jointly offered by the Department of Electrical Engineering and Computer Science (under course number 6.141) and the Department of Aeronautics and Astronautics (under course number 16.405).

Let us note that there have been some other courses that utilized state-of-the-art hardware in robotics education. The most notable contemporary example is the ZERO Robotics Challenge [7]-[10]. Teams from all over the United States compete do develop software for a space robotics hardware. They first compete in simulation systems, and then winning teams compete with real robots at the International Space Station in the zero-gravity environment. The program is an excellent example of the utilization of state-of-the-art robotics equipment. It has also emphasized collaboration and teamwork since its inception [11], [12]. However, almost all teams are entirely made up of undergraduate students, rather than high school students. Furthermore, the state-of-the-art hardware for space robotics differs significantly from the hardware for ground/aerial robotics. Our program utilizes hardware for the latter, and it is specifically tailored for high-school students. A relevant example is from 2002. A joint program offered by the Carnegie Mellon University (CMU) and National Aeronautics and Space Administration (NASA) invited 30 high-school students to NASA for a residential program to learn robotics using a rover utilized by NASA [13]. The authors note that the goal for this course was to "provide selected high school students with an immersive exploration of mobile robotics using leading-edge technologies." Our program's aim for 2016 was similar, with two exceptions: the focus on high school education, and the focus on mobile ground robots (rather than space robotics).

For the mobile robotics domain, most of the research focused on the development of low-cost platforms, with the motivation of reaching out to a large number of students. A recent survey is given by Irigoyen [14]. Most notable examples include the use of Lego kits [5], [15]-[18], kits based on the iRobot Create fitted with a Gumstix embedded computer [19], [20] or fitted with just a laptop computer [21]. These platforms serve the important purpose of reaching as many students as possible, some of whom may have very limited budgets. However, almost all of them lack state-of-the-art sensors and computers that are an essential part of many contemporary robotic systems. Our platform includes the state-of-the-art hardware utilized in real-world robotic vehicles today, while remaining within the budget of high schools participating in, *e.g.*, the FIRST Robotics Challenge.

This paper is organized as follows. In Section 2, we describe the open-source hardware and software of our platform. In Section 3, we describe the major activities in the program and the details of the technical instruction. In Section 4, we provide the data on the students' self-assessment along with a discussion of the results. We devote

the Section 5 to a discussion on potential opportunities to extend this program and its pedagogy in the future. We conclude the paper with remarks in Section 6.

II. Hardware And Infrastructure Software

The hardware was initial developed for an MIT hackathon in January 2015 by the authors. It was updated for another hackathon in January 2016. The same version was also used for teaching MIT's flagship undergraduate robotics course. The same robotics hardware was used for teaching this high-school program.

Our hardware platform is an exciting autonomy-capable mini race car. The fully-assembled hardware platform is shown in Figure 1. Its most essential components are shown in Figure 2. The vehicle is based on the 1/10-scale Traxxas RC Rally Car. We use the vehicle chassis, which one electric drive motor that drives the wheels, and one electric servo motor that steers the front wheels. The reported speed of the Traxxas RC Rally Car is 40 mph. However, throughout this high-school program, we limited the speed the vehicle to 5 mph through programming the firmware of an open-source Electronic Speed Controller (ESC), called the VESC [22].

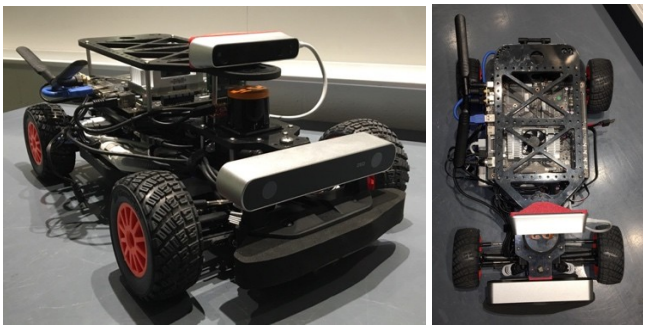

FIGURE I
THE MIT RACECAR PLATFORM.

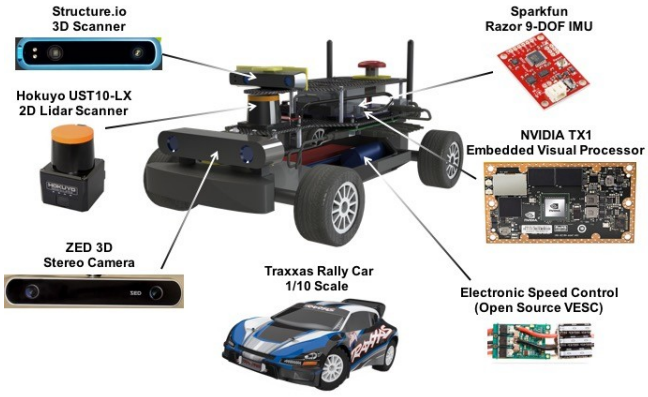

FIGURE II
THE MIT RACECAR PLATFORM COMPONENTS.

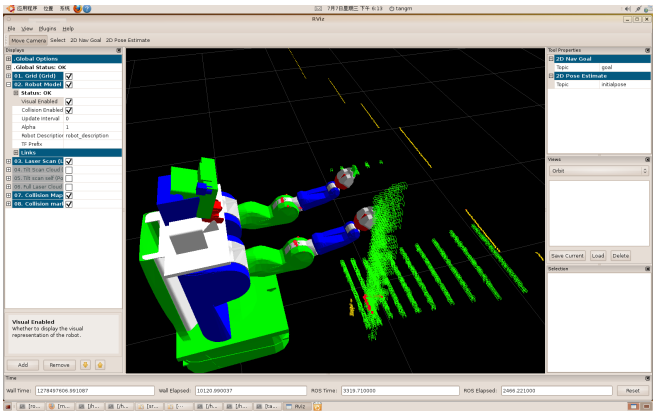

FIGURE III
THE ROBOT OPERATING SYSTEM VISUALIZATION SOFTWARE, RVIZ.

The main computing element is the Nvidia Jetson Tegra X1 embedded supercomputer (from here on called the, Jetson TX1). The Jetson TX 1 features a GP-GPU that delivers 1 Teraflops of computation power, using only 10 Watts of electrical power. It also boosts a quad-core Central Processing Unit (CPU). The CPU is clocked at 2 GHz.

The hardware platform includes three main sensors. The first major sensor is the Structure.io RGB-D camera. The sensor provides RGB color along with depth. The depth sensing is based on the structured light method, similar to the method used by the Microsoft Kinect sensor. The Structure.io sensor provides RGB-D images at the VGA resolution (640 by 480 pixels) at a rate of 30 frames per second. It perceives depth in the range of 0.4 to 3.5 meters. Its reported accuracy is less than a centimeter at the 0.4-meter range and around 3 cm at the 3-meter range. The second major sensor is the Hokuyo UST10-LX planar laser range finder. It features one laser range finder element that rotates at 40 Hz, providing 270-degree field of view at ¼-degree resolution. Finally, the third major sensor is the Stereolabs ZED stereo camera, which provides synchronized video from cameras. The two images from the two separate cameras can be utilized to re-construct depth by utilizing standard stereo matching techniques. The Stereolabs SDK implements a semi-global matching algorithm that runs on GPU-based computers, such as the Jetson TX1. The sensor suite also includes an inertial measurement unit, specifically the Sparkfun Razor 9-DOF IMU. Finally, the open-source electronic speed controller, called the VESC, allows us to sense the speed of the drive motor, which is a measurement of the vehicle's speed.

These components are arranged on two pieces of custom-made plates, which were cut using a laser cutter. The lower piece houses the main embedded computer and the inertial measurement unit. The RGB-D camera is mounted on the upper piece. The laser range finder and the stereo camera are mounted directly on the vehicle chassis.

The main computer runs the Ubuntu Linux operating system. The computer also runs the Robot Operating System (ROS). The ROS environment allows robotics software to be modularized. For instance, the feedback control systems

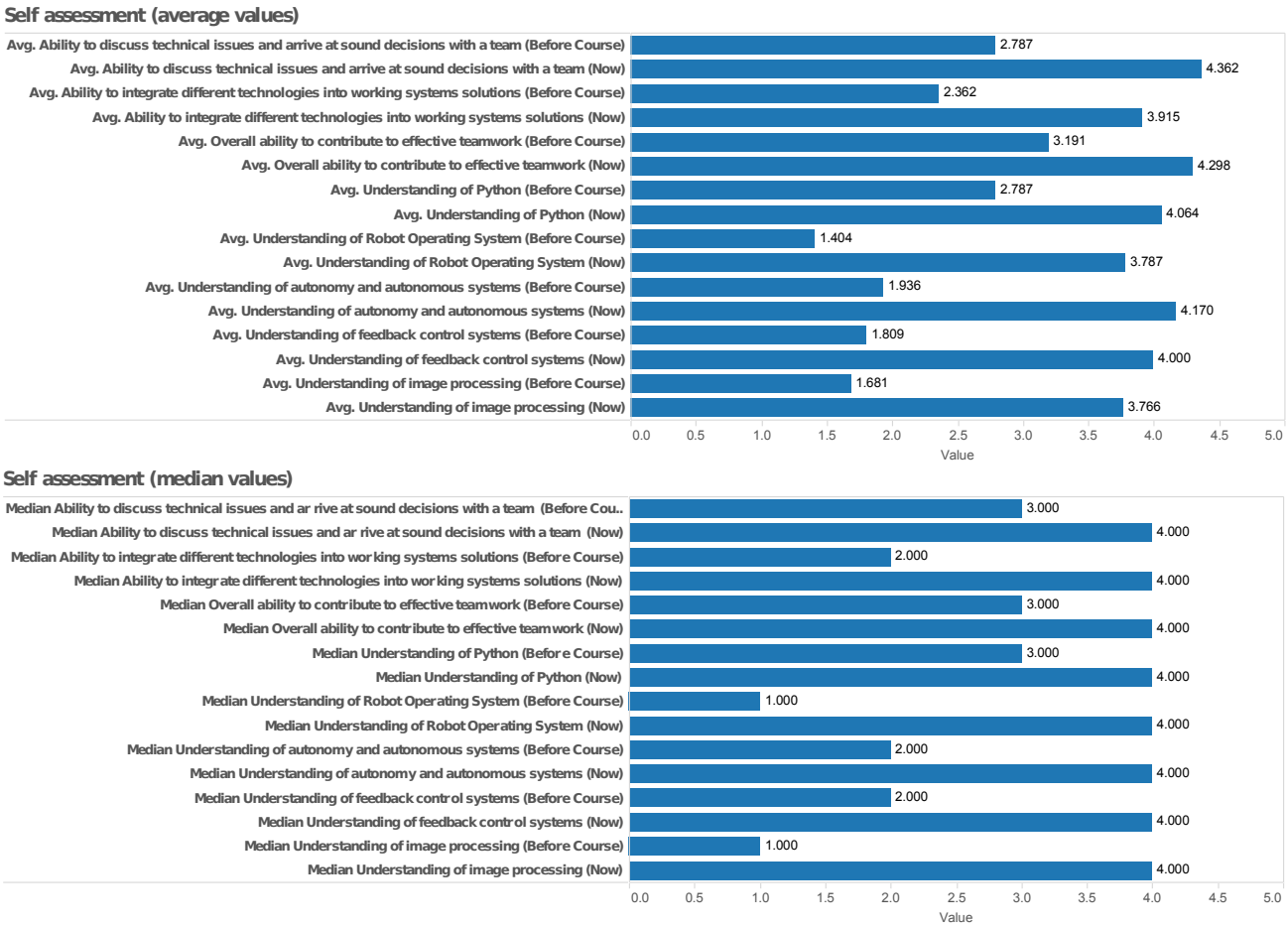

FIGURE VIII
SELF-ASSESSMENT OF COLLABORATION AND TECHNICAL SKILLS.

software, motion planning system software, computer vision system software, and other perception system software can be separated into their own software modules. Each software module is called a "node". The nodes share information using "messages." The ROS environment also provides tools to visualize data that is being sent between the nodes. See Figure 3. As seen in the figure, the software allows the visualization of the sensor data in real time along with other features.

We have implemented the various drivers for hardware integration. Specifically, our team has implemented software that interfaces with all sensors and software that governs the VESC. Our team has also developed a simulation system based on the Gazebo simulator. All this infrastructure software is made available open source through the following URL: http://github.mit.edu/mit-racecar/. More information about this and other classes taught by this platform can be found in the following URL: http://racecar.mit.edu.

III. ACADEMIC PROGRAM

The academic program included the following: *(i)* lectures that convey theoretical underpinnings of robotics, *(ii)* laboratory exercises that allow the students to practice hands-on skills, *(iii)* lectures on collaboration and communication that help students work in teams, and *(iv)* technical seminars that broaden the students' vision in science, engineering, research, entrepreneurship and beyond.

The agenda is as follows. Each day starts with a one-hour lecture that teaches the foundational the topics. The rest of the day is largely devoted to laboratory exercises that allows students to practice hands-on development skills. Each week ends with a mini project. Near the end of the week, the students are given time to go through the software system design exercise, and implement the software for their projects. The final week of the program is devoted to the course challenge. During the final week, the lectures focus on case studies. Instead of the laboratory exercises, the students focus on design and implementation for the course challenge.

The technical program is split into four modules. We named them: *(i)* Move, *(ii)* Explore, *(iii)* Learn, and *(iv)* Race. Each module takes one week of time to implement, and they collectively cover the four weeks in this order. The first three modules provide the basics of robotics, and end with a mini project. The last module focuses on the course

challenge.

The first module, which we call *Move*, teaches the basics of commanding the vehicle actuators, such as steering and drive motor. The students first learn how to work with the Robot Operating System and work with the platform. In particular, they learn how to acquire data from the laser range finder, the inertial measurement unit, and the wheel odometer (through the VESC). Later in the week, they learn the basics of control systems in the lectures, and they experiment with the design of PID control systems in the laboratory exercises. The mini project for this module is to design and implement a software system for wall-following drag racing. Specifically, the students must design a perception system that detects the wall from laser range finder measurements, and a controller that steers the system to align with the wall. During the mini race, we start two cars at the same time side by side, and declare the first car to finish as the winner. We rank the teams in a tournament style race.

The second module, called *Explore*, delves into working with cameras. Specifically, the students learn the basics of blob detection in images as well as visual servoing control systems. The project for the week is to design and implement a software system that drives the robot towards a fork, detects the color of a marker right at the fork location, turns right if the marker is green and turns left if the marker is red.

July 12 (Tue):	Jaya Narain, MIT Mechanical Engineering Hacking AT (Assistive Technology)
July 13 (Wed):	Prof. Paulo Lozano, MIT Aeronautics and Astronautics Space Micro-Propulsion
July 14 (Thurs):	Luke J. Skelly, MIT Lincoln Laboratory 3D Photo Counting Laser Radar
July 15 (Fri):	Dr. Long Phan, Founder & CEO, Top FLight Technologies Be a Leader, Make Robots
July 18 (Mon):	Claudia Loy, Dr. Claudio Heller, Continental Corporation Environmental Perception for Driver Assistance and Automated Driving
July 19 (Tue):	Chris Peterson, MIT Admissions Office How to apply to MIT (and other colleges) as a Maker
July 20 (Wed):	Dr. Eric D. Evans, Director, MIT Lincoln Laboratory MIT Lincoln Laboratory Overview
July 21 (Thu):	Dr. Farzana Khatri, Matthew Willis, MIT Lincoln Laboratory A Laser Communications Link to the Moon!
July 22 (Fri):	Dr. Katherine A. Rink, MIT Lincoln Laboratory My Adventures in Engineering
July 25 (Mon):	Prof. Julie A. Shah, MIT Aeronautics and Astronautics Humans and Machines of Like Mind: Augmenting Humans through Collaboration in Decision-Making Tasks
July 26 (Tue):	Prof. Timothy M. Swager, MIT Chemistry Molecular Electronics for Chemical Sensors
July 27 (Wed):	Prof. Sangbae Kim, MIT Mechanical Engineering MIT Cheetah: New Design Paradigm Shift toward Mobile Robots
July 28 (Thu):	Prof. Alex "Sandy" Pentland, MIT Media Lab Social Physics
July 29 (Fri):	Dr. Idahosa A. Osaretin, MIT Lincoln Laboratory Micro-sized Microwave Atmospheric Satellite
August 1 (Mon):	Prof. Russ Tedrake, MIT Electrical Engineering and Computer Science Humanoid Robots and Robot Birds
August 2 (Tue):	Prof. Nick Roy, MIT Aeronautics and Astronautics Autonomy and Navigation for UAVs in the Urban Environment
August 3 (Wed):	Seth Kaufman, Amazon Robotics How to Make Your Mark in Robotics

FIGURE V
THE SEMINAR SCHEDULE.

The third module, called *Learn*, teaches the basics robot perception and robot motion planning. In the first half of the week, the students learn the basics of robot perception algorithms, including localization and mapping. In the second half of the week, the students learn the basics of robot motion planning algorithms, such as potential field-based, search-based, and sampling-based planners. The project is to develop software for a robotic vehicle that explores its surroundings without colliding with obstacles.

The final module, called *Race*, focuses entirely on the course challenge. The course challenge includes two main activities: Racing challenge and an exploration challenge. Both challenges must be completed *fully autonomously*.

The race involves the design and implementation of a software system which allows the vehicle to travel through the race course as fast as possible. A picture of the race course is shown in Figure 6 with cars and students along the race course. Notice that there is a fork at the top part of the course. Right at the fork, there is a visual marker. The cars must detect the marker and make a turn if the marker is red, drive straight if the marker is green. First, we allow each team to run their car on the track. We then place multiple cars on the track at the same time. A view from the race is shown in Figure 7; Autonomous mini race cars are chasing each other.

The exploration challenge involves navigating in an environment that is unknown *a priori*. The environment includes several visual markers framed in a pink frame. The challenge is to design and implement a software system that requires the robot to find as many markers as possible without colliding with obstacles within a fixed amount of time.

FIGURE VI
THE COURSE CHALLENGE EVENT ARENA.

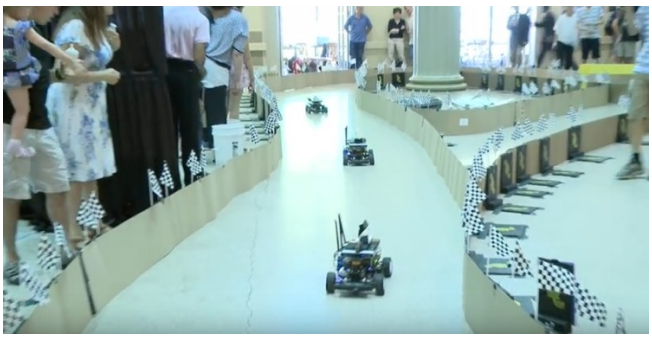

FIGURE VII
A VIEW FROM THE RACE EVENT.

The academic program also features instruction on communication and collaboration. In these lectures, the students are instructed in forums, where they get a chance to actively participate in exercises that help them develop technical communication skills and provide them an opportunity to effectively work in teams.

Finally, the program features a one-hour seminars on each day. The seminars were given by researchers at MIT as well as the thriving robotics industry. The seminar schedule is shown in Figure 5. Each technical seminar is roughly 45 minutes of talk by the speaker and roughly 15 minutes of questions and answers session.

IV. ASSESSMENT AND DISCUSSION

In this section, we report our assessment based on student surveys, and our findings after analyzing the data.

At the completion of the course, the students were asked to complete a self-assessment survey. They were asked to evaluate their the following before and after the course:

- Understanding of the Python programming language;
- Understanding of the Robot Operating System;
- Understanding of feedback control systems;
- Understanding of image processing;
- Ability to integrate different technologies into a working systems solution;
- Ability to discuss technical issues and arrive at sound decisions with a team;
- Overall ability to contribute effective teamwork.

The ratings correspond to the following: 1. Very poor; 2. Somewhat poor; 3. Neutral; 4. Somewhat good; 5. Very good.

The results of the self-assessment survey are shown in Figure 8. The top figure shows the average rating for each question; the bottom figure shows the median. Notice that the students report substantial improvement in every item.

We also conducted a self assessment survey of communication and collaboration skills taught by the class in the middle of the program. We asked the students to evaluate their interpersonal, communication and collaboration skills, before the course and two weeks after the start of the course. The ratings correspond to the following: 1. Very poor; 2. Somewhat poor; 3. Neutral; 4. Somewhat good; 5. Very good.

In terms of averages, we observe a slight increase in the students' self assessment of these skills two weeks into the program. Specifically, assessment of interpersonal skills average raises from 3.26 to 3.71, that of communication skills raises from 3.36 to 3.78, and that of collaboration skills raises from 3.42 to 3.86. In all three categories, the median values are 3 before the class, while the median values improve to 4 after the class. Hence, most students report at most "neutral" for these skills before the class; they report that these skills improved to at least "somewhat good," after the course. For the reader to see the slight improvement individually, we refer the reader to Figure 9. Several comments are in order.

First, the students report relatively good programming skills (Python programming) even before starting the course; yet, they improve their programming skills over the course of this program. See Figure 8. The average reported rating for "understanding of Python" is 2.787 before the course, and rises to 4.064 after the course; the median rises from 3 to 4. We believe that the program contributes to students' programming abilities substantially, even for those who are good programmers. The students frequently quoted the valuable experience of designing and implementing software to handle real-world, real-time data, which we believe was one of the major reasons that helped the students sharpen their programming and software development skills.

Second, the students report very little understanding of advanced robotics concepts, such as robot operating system, feedback control systems, image processing, and autonomous systems; but, the program allows the students to build these skills substantially in a relatively short amount of time. Before the course, the average rating ranges between 1.5 and 2; the medians 1 or 2. Hence, most students report that their skills in these areas are "poor," before the course. After the course, the ratings improve substantially. Specifically, the average values range between 3.7 and 4.2; the median values all improve to 4. Hence, most students rate their skills in these areas as at least "good," after the course. We believe that the program's focus on these advanced robotics topics with hands-on practice in a collaborative project-based environment allows the students to learn these advanced skills fairly well in a very short period of time.

Third, even though the students report relatively strong communication and collaboration skills, we observe that the reported assessment of these skills also improve rapidly throughout the program. Even two weeks into the program, the students report improvement of these skills, as seen in Figure 9. By the end of the program, the students report a significant improvement on complex teamwork skills, such as "ability to discuss technical issues and arrive at sound decisions as a team," "ability to integrate different technologies into a working system solution," and "ability to contribute to effective teamwork." We believe that the assessment is a consequence of a number of aspects of the program. First, the program teaches a rigorous set of lectures on communication and collaboration. In forum-style lectures, the students learn the basics of working in teams. Second, the program provides the students several opportunities to practice these skills. Each lab exercise, each project, and the course challenge are all conducted in teams. We believe that the lectures and the hands-on practice for communication and collaboration that is embedded in this

class provides the results we observe in Figures 8 and 9.

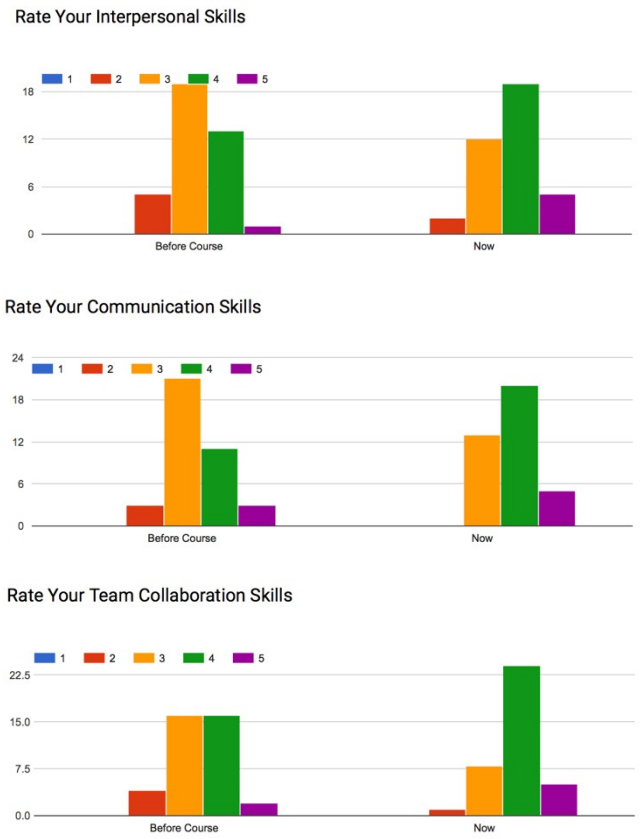

FIGURE IX
MID-COURSE SELF ASSESSMENT OF INTERPSERSONAL, COMMUNICATION AND COLLABORATION SKILLS.

V. FUTURE PLANS: SUPPORTING ONLINE COURSES AND A COMMON HARDWARE PLATFORM

There is ample room for improvement, towards updating the platform, the technical lectures, the laboratory exercises, the projects and the course challenge. However, we believe the most important area for future work is to fill in the essential ingredients that will allow us to scale this program up, to reach thousands of high school students. We propose two extensions: a set of supporting online courses and a common hardware platform. We envision the students will have access to both of these components before coming to MIT campus for the residential part of the program. In this section, we describe these extensions in details, and we survey some of the relevant work in the literature afterwards.

Supporting Online Courses: We believe one of the key ingredients is to support the program with *online material* that students can work with before they start the summer program at MIT. The online material will be shaped into a Massive Open Online Course (MOOC) program. The MOOC program will consist of a few short courses, including programming, software engineering, the basics of control systems, computer vision and robotics as well as a set of lectures on communication and collaboration.

We envision that the students will go through these courses the academic year before they come to MIT campus, when they can focus on projects that involve the design and implementation of complex robotics software systems.

These online courses will include a number of components: *(i) short video lectures* which the students can watch on their own pace, comment on and discuss through the online forums; *(ii) short technical exercises* which are automatically graded, so that the students can get rapid feedback on their learning of the complex foundational concepts; *(iii) hands-on programming exercises* which are also automatically graded through unit tests; *(iv) hands-on robotics software development exercises that run on a realistic simulation environment or run with real-world data collected using the experimental platforms*, through which the students can design and implement relatively complex robotics software systems and test them on their own computer on their own time and pace.

The implementation of these components requires the development of a few key platforms. First is an *online forum* that is open to all students and instructors. We envision this online forum to be used for any question that relates to the course. The second is an *open code repository* that students and instructors can share code. We envision instructors to share the solutions to various programming exercises, software components that students can utilize toward their work as well as certain infrastructure software such as robot simulation software. We envision that the students will use the code repository to share work-in-progress code snippets, their solutions to programming exercises, and code branches that include their own updates to the instructors' software posted in the code repository. We expect that the students, will help tremendously in improving the software developed by the instructors; hence, it is critical to allow students to provide feedback or even the opportunity to develop software for the course itself. Thankfully, modern software version tracking tools provide easy means to implement this vision. Finally, third is an *open data repository* which the instructors and students can utilize to share data, either from experiments on the robots or from the simulation environment.

Common hardware platform: To further strengthen this program, we also envision a common hardware platform, such as the mini race car described in Section 2, which several high schools have access to even before coming to the MIT campus for the residential portion of the program. We believe that having access to a common platform will allow the students download and execute software from the code repository. They will be able to try software developed by the instructors or the fellow students. They will also be able to upload their data from their experiments to the common data repository to share with fellow students and/or ask feedback from the instructors, or even have the data evaluated by automated data evaluation software to get rapid feedback on their latest experiment. Furthermore, we believe that access to a common platform

will allow the students from different high schools to work on common projects by utilizing the online resources, specifically the online forums, the open code repository, and the open data repository. We imagine the students will utilize state-of-the-art online video conferencing tools to talk to each other and collaborate. We, as the instructors, hope to learn from this experience and tightly integrate the utilization all these remote collaboration tools into our teaching of communication and collaboration, as we believe these tools will be essential to robotics software systems design and development in the coming years.

VI. Conclusions

We presented a new high-school robotics program, called the MIT Beaver Works Robotics Institute Robotics Program. The program focuses on robotics software, and allows the students to design and develop complex software systems to run on mini race platform. In particular, the course challenge involves the design and implementation of software systems that allow the mini race car to operate completely autonomously. The program champions project-based and team-oriented learning. The program was implemented for the first time in the Summer of 2016 on MIT campus as a four-week residential program with the participation of 46 students, 24 of which participated from outside of Massachusetts and stayed in the Boston Area during this program. The self-assessment suggests that the students benefit in many directions in terms of both technical and teamwork skills. Our future plans include the expansion of the platform, particularly with an online-supported teaching, in which the students follow online classes during the semester building up to the summer to understand the foundations, and then use their time at MIT to design and build complex robotic software systems. A short video describing the course can be found in the following URL: https://youtu.be/ozcBNbu7ogY.

Acknowledgements

This material is based upon work supported under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Air Force. The program was supported by a number of sponsors, which are featured in the video link provided in Section VI. In addition, Prof. Karaman was supported through the NSF CAREER program through grant #1350685.

References

[1] E. Kolberg and N. Orlev, “Robotics learning as a tool for integrating science technology curriculum in K-12 schools,” presented at the Frontiers in Education Conference, 2001.

[2] A. Welch and D. Huffman, “The Effect of Robotics Competitions on High School Students' Attitudes Toward Science,” *School Science Mathematics*, vol. 111, no. 8, pp. 416–424, Dec. 2011.

[3] G. Nugent, B. Barker, N. Grandgenett, and G. Welch, “Robotics camps, clubs, and competitions: Results from a US robotics project,” *Robotics and Autonomous Systems*, vol. 75, no. Part B, pp. 686–691, Jan. 2016.

[4] R. T. Johnson and S. E. Londt, “Robotics Competitions: The Choice Is up to You!.,” *Tech Directions*, vol. 69, no. 6, pp. 16–20, 2010.

[5] C. Chalmers, “Learning with FIRST LEGO League,” *Society for Information Technology and Teacher …*, 2013.

[6] H. Fike, P. Barnhart, C. E. Brevik, E. C. Brevik, C. Burgess, J. Chen, S. Egli, B. Harris, P. J. Johanson, N. Johnson, M. Moe, and R. Olsen, “Using a robotics competition to teach about and stimulate enthusiasm for Earth science and other STEM topics,” *EGU General Assembly*, 2016.

[7] A. Saenz-Otero, J. Katz, and S. Mohan, “ZERO-Robotics: A student competition aboard the International Space Station,” *IEEE Aerospace Conference*, 2010.

[8] S. Nag, I. Heffan, A. Saenz-Otero, and M. Lydon, “SPHERES Zero Robotics software development: Lessons on crowdsourcing and collaborative competition,” presented at the IEEE Aerospace Conference, 2012, pp. 1–17.

[9] D. W. Miller, “ZERORobotics: a Student Competition Aboard the International Space Station,” presented at the Next-Generation Suborbital Researchers Conference, 2010.

[10] A. Saenz-Otero and J. Katz, “The Zero Robotics SPHERES Challenge 2010,” *IEEE Aerospace and Electronic Systems Magazine*, 2011.

[11] S. Nag, J. G. Katz, and A. Saenz-Otero, “Collaborative gaming and competition for CS-STEM education using SPHERES Zero Robotics,” *Acta Astronautica*, vol. 83, no. C, pp. 145–174, Feb. 2013.

[12] S. Nag, “Collaborative Competition for Crowdsourcing Spaceflight Software and STEM Education using SPHERES Zero Robotics,” MIT, 2012.

[13] I. R. Nourbakhsh, E. Hamner, and K. Crowley, “Formal measures of learning in a secondary school mobile robotics course,” presented at the IEEE International Conference on Robotics and Automation, 2004.

[14] E. Irigoyen, E. Larzabal, and R. Priego, “Low-cost platforms used in Control Education: An educational case study,” presented at the IFAC Symposium Advances in Control Education, 2013, vol. 46, no. 17, pp. 256–261.

[15] E. Danahy, E. Wang, J. Brockman, A. Carberry, B. Shapiro, and C. B, “LEGO-based Robotics in Higher Education: 15 Years of Student Creativity,” *International Journal of Advanced Robotic Systems*, pp. 1–16, 2014.

[16] L. E. Whitman and T. L. Witherspoon, “Using legos to interest high school students and imtrove k12 stem education,” presented at the 33rd Annual Frontiers in Education, 2003. FIE 2003., 2003, vol. 2, pp. F3A_6–F3A_10.

[17] A. Salamon, S. Kupersmith, and D. Housten, “Inspiring Future Young Engineers Through Robotics Outreach,” presented at the Proceedings of the Global Conference on Educational Robotics, 2008, pp. 1–7.

[18] E. Afari and M. S. Khine, “Robotics as an Educational Tool: Impact of Lego Mindstorms,” *IJIET*, vol. 7, no. 6, pp. 437–442, 2017.

[19] M. J. Mataric, N. Koenig, and D. Feil-Seifer, “Materials for Enabling Hands-On Robotics and STEM Education,” presented at the AAAI Spring Symposium Semantic Scientific Knowledge Integration, 2007, pp. 1–4.

[20] T. L. Crenshaw and S. Beyer, “UPBOT: A Testbed for Cyber-Physical Systems,” presented at the Proceedings of the International conference on Cyber security experimentation and test, 2010.

[21] J. Kelly, J. Binney, A. Pereira, O. Khan, and G. Sukhatme, “Just Add Wheels: Leveraging Commodity Laptop Hardware for Robotics and AI Education ,” presented at the Proceedings of AAAI Education Colloquium, 2008.

[22] Benjamin's robotics, VESC – Open Source ESC Project, URL: http://vedder.se/2015/01/vesc-open-source-esc/, retrieved, January 19, 2017.