Influence and Philosophical Reflection on ChatGPT in the Media Industry

Alejandro Gallego Schmid

Introduction

Recently there has been a rising popularity in the use of robotics as a vehicle to expose K-12 students to the STEM disciplines. A common practice is to have the students build remote control robotic vehicles for competitions such as the various First USA Robotics Competitions[1]. However, the robotics profession is not just focused on remote controlled mobile robots but rather involves stationary industrial arms running autonomously doing repetitive tasks. Our approach is to go beyond having the students simply build the robot and control it using a remote control to teaching them more advanced engineering concepts more closely related to the profession. We still aim to introduce STEM, and engineering in specific, to K-12 students using robotics but we are doing it in a different way.

In this work we are taking a robotics kit designed for K-12 students to build remote control mobile robots, combining it with an educational robotics software tool designed to support students in higher education and creating a robotics educational system designed to teach higher level engineering concepts to K-12 students. This is an ambitious challenge with promising results.

The system introduced in this paper allows the students to design, model, simulate, build, and program their robotic arm. Emphasis is placed on the process of modeling and simulating their design in order to assure correctness before physical construction and physical control of the arm. The system includes a Pitsco Tetrix Prime robotics kit [2], designed for K-12 robotics activities, an Atmel XMEGA-A3BU XPlained microcontroller board [3] along with a custom circuit board, both designed to provide electrical signals to the servo motors in the kit given commands from the robotics software tool and the software tool itself. The robotics software tool [4] was developed by our research team for the purpose of supporting undergraduate and graduate introductory robotics courses.

a. The SRO Summer Camp

The system was used in the Summer Research Opportunity [5] camp offered at Florida Gulf Coast University's Whittaker Center for STEM Education. This summer camp was offered as a two- week camp to middle school students who participated in the Thomas Alva Edison Regional Science Fair. The students were asked to design a robotic arm that could hold a pencil and write their name on a sheet of paper. The summer camp had 29 students participate. Ten kits were used and the students formed groups of 3 each using one kit. The software tool was given to each student so they can work independently in the camp or at home.

b. The Learning Objectives

This system is designed to teach students about basic industrial robotic arms and how to program them. However, we are introducing the concept of modeling and simulation. Using this tool we are teaching the students how to model their design, create a virtual arm based on their model and program the virtual arm. Once their programs are fully tested and correct then they can run their program on the physical arm which they will or would have built. Specifically the students are to learn the engineering concept of using a tool to implement a virtual version of their arm to reduce development effort. The mathematical modeling uses the standard Denavit and Hartenberg (DH) parameters [6,7].

The following are the learning outcomes for this system. The students will learn to:

- design a robotic arm that can perform a specific task,
- determine a set of D-H parameters to represent their physical arm,
- model their arm virtually by properly entering their D-H tables using the software tool,
- write a program to control the arm to accomplish the specific task,
- simulate their software program using the

virtual arm,
- build the physical arm based on their design, and
- control the physical arm running their software program.

c. Similar Approaches

Perhaps the largest K-12 robotics programs are the various FIRST competitions. In these competitions the students build their mobile robot and control it using the remote control. There is an autonomous section to the competition however in the author's experience it is observed that the autonomous section is very small and most teams do not participate. More closely related to summer camps, He et. al [8] offered a robotics summer camp which had an autonomous component however their mobile robots were controlled using a remote control unit for a competition at the end of the camp. Tetrix Prime for NI myRIO [9] is a similar system. It also uses a combination of the Tetrix Prime kit with National Instrument's myRIO [10] to teach advanced engineering concepts to students. The students program the myRIO device using the LabVIEW [11] programming language. The myRIO device is linked to the servomotors of the robot to control the robot. The robot can be an arm or a mobile robot. The myRIO is a Field Programmable Gate Array (FPGA) device which runs balancing software provided by the system. The robotics projects are based on this balancing technology. It differs from ours in that the programming of the FPGA and the balancing software are too complex for the students to understand and so they simply include premade packages the students include into their robot. The student learns what engineering can do in a cool project. Our approach is to teach robotics programming software using a realistic industrial robot programming language which is simple and the students can write all the necessary code to fully control their arm. Our approach also introduces modeling and simulation.

The Educational Experience

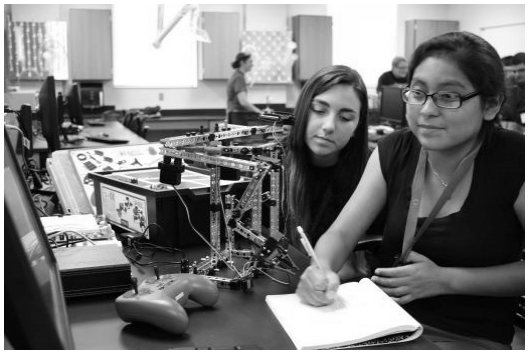
The following is the experience this system is designed to give the students. The students will be given a task that is appropriate for the kit. Next the students will receive a presentation of the kit and the capabilities of the kit followed by some suggestions they may follow to help them design the arm. In our camp the task was to have the arm draw on a piece of paper a message or a figure of the team's choice. It is feasible to design an arm using the kit to accomplish this task. Other tasks may be used as well. The students will then design an arm to accomplish this task. Ideally they will think about the design, draw their design on paper, measure the D-H parameters, enter them into the tool thus creating a virtual arm, and then finally move the arm using the tools slider controls to see if their design is capable of accomplishing the task. Experience from our camp showed that the students went straight to the kit and started building from the very beginning.

Given this tendency the desired experience was modified as follows. After the presentation of the kit and its capability and the presentation of the task, the students will start building their arm using the kit, See Figure 1 (a). In our camp, with team of 3 students, each student took about 3 days (12 hours) to complete their initial arm. After this phase the students are given a presentation on robotics including robotics in the work place, industrial robotics, research and mobile robotics, robot applications, and studying robotics in undergraduate and graduate school as a profession. Next the process for determining the D-H parameters is presented. This includes what they are, what they measure, briefly how they are used in the field of robotics and finally how they can measure these parameters from their arm. Finally a presentation of the robotic programming language is given.

Next the students will measure the D-H parameters of their arm and enter them into the tool, see Figure 1 (b). They will verify that their virtual arm resembles their actual arm in terms of the way it can move. If the virtual arm does not move the way their real one moves then they did not measure the D-H parameters correctly. In the camp the students were not able to grasp the process of measuring these parameters and relied on the student helpers. Software has since been added to the tool to support this activity. Once the parameters are measured correctly the students will enter them into the tool and verify their correctness.

Next the students will program their virtual arm using the robotics language and the software tool to accomplish their task, see Figure 2 (a). Once they are satisfied with the simulated results they make all the connections between the arm and the PC, establish the communications and tell the software tool to control the actual arm by executing their program. At this point the real arm they would have built will move according to their program. The

(a)



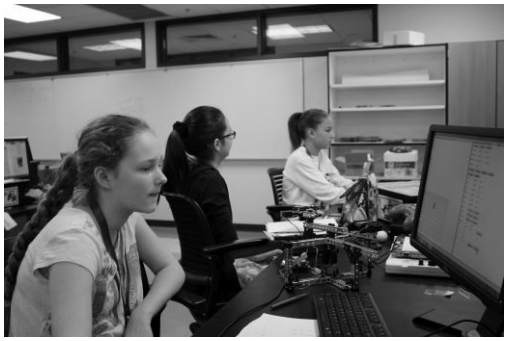
(b)

Figure 1: (a) the students build the arm, (b) the student measure the D-H parameter.

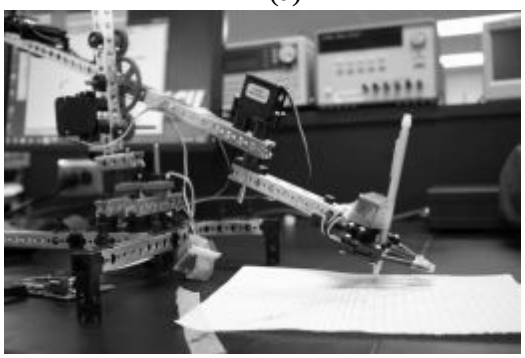
students can then make adjustments to their programs to account for characteristics that are not modeled such as the momentum in the movement, velocity limits, overshooting the target and so on, See Figure 2 (b).

The students in the camp took about 6 days (24 hours) for this task, however the tool had errors in it preventing the physical and the virtual arms from getting synchronized. This caused lots of frustration with the students. We estimate this task should take only about 3 days (12 hours) with the current software tool that has been fixed.

Finally, at the end of the camp each team will present their arm and give a demonstration of it performing the task, see Figure 3 (a). The students' parents and teachers will be invited to the presentations, Figure 3 (b). This part was the highlight of our camp as the students were very eager to show off their hard work to their families and their teachers.



(a)



(b)

Figure 2: (a) the students develop the program to control the arm, (b) the students adjust their program to account for the characteristics of the real arm.

(a)

(b)

Figure 3: (a) The students present their work, (b) the parents and teachers are invited to attend the presentations.

Programming the virtual arm first shows the students how they can use modeling and simulation to save time in creating a solution to an engineering problem. Programming the virtual arm before the actual arm saves time for several reasons. First each student has a copy of the software tool running their virtual arm so they each can program the arm concurrently. We only had one arm kit per team. Next, the dynamics of the virtual arm can be adjusted in the model so as to remove those constraints while the basic program is developed. For example, they can increase the power of the motors and move the arm at very fast speeds without the negative effects such as overshoot, and acceleration limits. They can adjust the dynamics to represent the real arm after they determined the basic programming steps are correct. This allows them to concentrate on one issue at a time. And finally the virtual arm does not model the imperfections of the real arm which cause delay in testing the programs. Eventually they will need to take into consideration all the characteristics of the real arm that are not included in the modeled. However, having a program that works with the virtual arm allows the students to concentrate only on adjusting their program to account for these realities knowing that the program is fundamentally correct. This generally includes slowing the arm to the point that the arm does not overshoot its destination location.

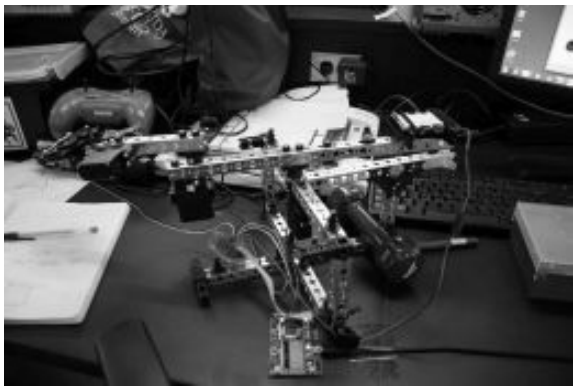
This modeling and simulation exercise also shows that a model is not perfectly accurate and can only be used to develop parts of the solution independent to these inaccuracies. The students need to understand the difference between the virtual and the actual arm and how they can adjust the model to allow them to address development complexities incrementally.

Details of the Student Activities

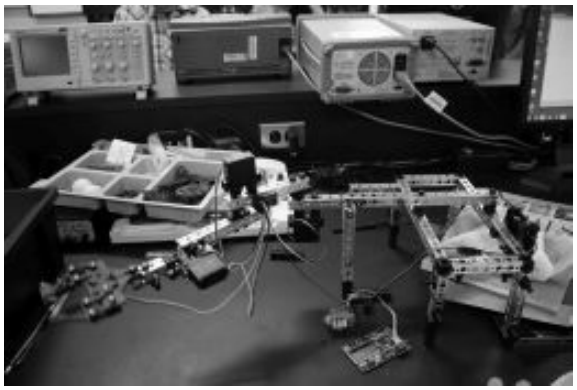
The following section describes the details of each major activity the students were asked to perform. These include building the arm with the kit, measuring the D-H parameters, creating the virtual arm, and programming the arm. The authors built a demonstration arm using the kit shown in Figure 5 to illustrate a complete example of these steps.

a. Building the Arm Using the Kit

The arm can be built using the kit. The students quickly figure out how to snap the parts together to form their designs. Since each kit included 3 servomotors they generally used 2 for the shoulder and elbow and one for the wrist to lift the pen off the paper. Figure 4 (a) and (b) shows two arms built by the students in the camp.



(a)



(b)

Figure 4: Photographs of two robotic arms built by the students participating in the summer camp.

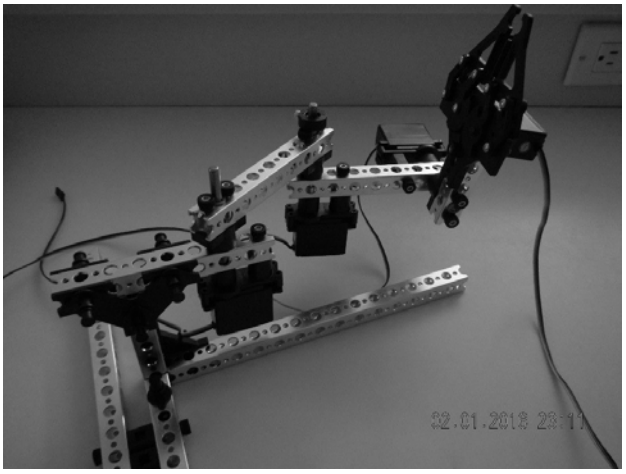


Figure 5: The demonstration arm built by the authors.

b. Measuring the D-H Parameters

The purpose of this section is not to help the reader understand how to determine the D-H parameters for an arm but rather to help the reader understand the complexity of the process. We feel that while this process is complex, middle school and older students can master this material if given sufficient time.

The four D-H parameters include the angle the link has relative to the link its attached to, Θ , the joint offset, d , the length of the link, a, and the joint twist, α. All are constants describing the physical characteristics of the link except for either Θ for revolute joints or d for prismatic joints which are variables representing the arm's current position. The kit can only support revolute joints so the joint offset, d, will be a constant. To represent a robotic arm's kinematic characteristics, we attach an imaginary reference frame to each link following the parameter placement algorithm. The links are ordered starting at the base of the arm and moving outwards towards the hand. The D-H parameters for each link are the distances its reference frame must move to reach the reference frame of the next link in the arm.

If the frames are placed following the D-H algorithm shown in Figure 6 then with only 4 motions one can move from one frame to the next. That is, suppose you will move a frame along its own axis to coincide with the frame of the next link. The 4 D-H parameters for that link describe these 4 motions.

The D-H parameters for the demonstration arm are shown in Figure 8. Figure 9 (a) shows the virtual demonstration arm and Figure 9 (b) shows the arm with all the frames drawn.

Determining the D-H parameters is not difficult once you understand the algorithm, however to "see" the process one must be able to visualize the arm and its reference frames in three dimensions. This 3D visualization is difficult but is an essential skill for any engineer.

For each link perform the following:

1. Assign a Z-axis to it and place it along the axis of rotation for that link. This is the joint attached to the link that moves the rest of the arm, not itself.
2. Determine the location of the common normal between the Z-axis of the previous link and its own Z-axis.
3. Place the origin of the frame at the intersection of this common normal and its Z-axis.
4. Point the X-axis along the common normal in the direction away from the previous frame. If the common normal has a length of 0 then place the X-axis perpendicular to its Z-axis and the Z-axis of the previous frame.

Figure 6: Algorithm to assign reference frames to the links.

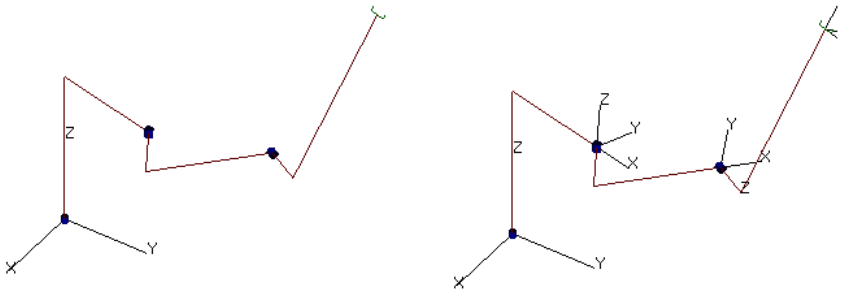
For each link perform the following:

1. Rotate the frame along its Z-axis until its X-axis aligns with the X-axis of the next frame. The angle rotated is Θ .
2. Translate the frame along its Z-axis until it reaches the common normal between its Z- axis and the Z-axis of the next frame. The distance traveled is d.
3. Translate along its X-axis (along the common normal) until it reaches the next Z-axis. At this time the origins and both X-axis will coincide. The distance traveled is a.
4. Rotate the frame along its X-axis until its Z-axis aligns with the Z-axis of the next frame. The angle rotated is α. .

Figure 7: Algorithm to measure the D-H parameters.

	Θ	*d*	*a*	α
0-1	0	85	60	0
1-2	0	-20	60	90
2-3	0	20	75	90

Figure 8: The D-H parameters for the demonstration arm. The thetas are variables and so we place zeros for them.



(a) (b)

Figure 9: (a) the virtual demonstration arm, (b) the arm with the frames attached.

c. Creating the virtual arm.

Once the D-H parameters are determined the student can enter them into the tool using the setup window as shown in Figure 10. The range of movement for each joint must also be entered as well as the maximum acceleration for each joint. When the student presses the Create Robot button the virtual arm appears in the left pane of the tool's window as shown in Figure 11.

d. Programming the Arm.

Programming the arm uses the tool's Robot Programming Window shown in Figure 12. The programming language is unique, however it is basically a simplified version of a typical industrial robot programming language. The basic program consists of a list of way points the hand must go through to reach the destination point together with their corresponding speeds along with other program control instructions for looping and conditional execution. Gonzalez et. al.[4] presents details on programming using this tool. The arm moves from point to point in traveling through a path. The code to move the arm consists of series of MOVE instructions that tell the arm to move from its current location to a specified point with a specified speed. To program the arm to perform a task the student must first determine a complete path. Then using either the virtual or real arm the student finds a set of points along the path. Then with these points the student writes the list of instructions to move the

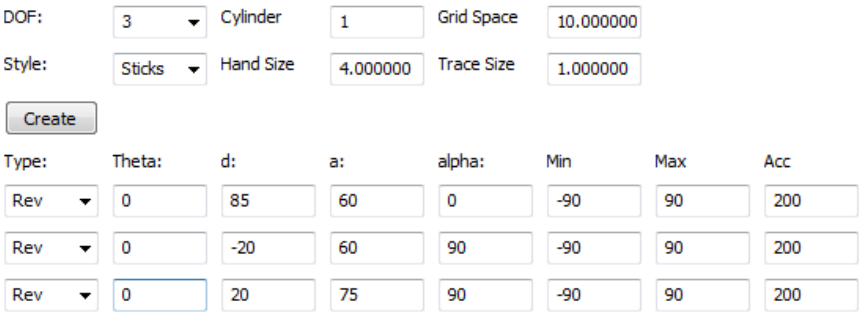


Figure 10: The window used to enter the specifications including the D-H parameters of the arm.

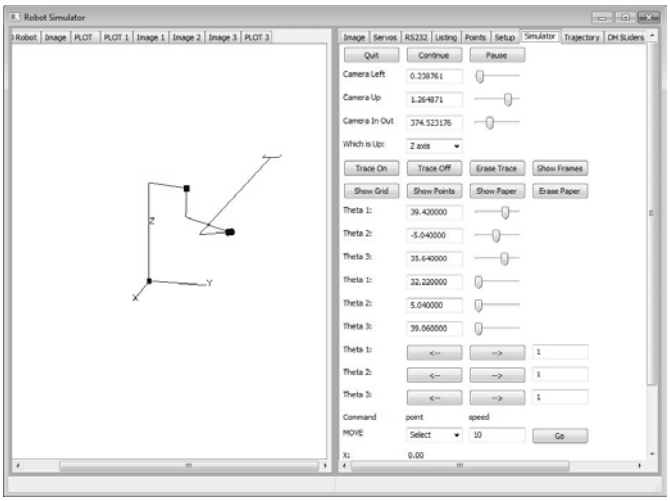


Figure 11: The window after the arm is created. On the left is the virtual arm and on the right is the menu for manual movements and other operations.

Figure 12: The programming window shown while executing a sample program. Instruction 4 is highlighted in red indicating that instruction is being executed. Debugging information is listed below. Note a = 2 and b = 2.

arm from point to point. The looping instructions can be used to perform segments of code more tha once. To define a point, the student moves either arm using the sliders to the point they are defining then tells the software to record that point along with a name that must also be provided. In Figure 12 the program moves from its current location to P1 then to P2 then back to P1 and P2 then to the home position. The repetitive movement was achieved with a simple loop using variables a and b. The screen while executing is shown in Figure 13.

The virtual arm moves in real time as the program executes. The program executes the next instruction only after the arm reaches the destination point for the current instruction. The speed parameter tells how fast to travel. The red button tells the current instruction that is executing. Once the program executes on the virtual arm correctly, the student establishes the communication link with the board and simply tells the tool to control the physical arm as well. The virtual and physical arm should move in synchronized fashion.

The Complete System

The complete system shown in Figure 14 was created by combining the two main components, the educational robotics software tool with the Tetrix Prime robotics kit and adding some interface components. The system is designed to support educational activities in the modeling, simulation and control of robotic arms as described by the educational experience presented above. The system

includes a Pitsco Tetrix Prime robotics kit, an Atmel XMEGA-A3BU Xplained microcontroller board along with a custom circuit board, both designed to provide electrical signals to the servo motors in the kit given commands from the robotics software tool and the software tool itself. The software tool provides the modeling and simulation of the arm. A feature was added to the software tool to allow it to control the physical arm. The interface between the software tool and the kit is implemented with the Atmel microcontroller board which communicates with the software and provides electrical control signals to the servo motors in the Tetrix kit.

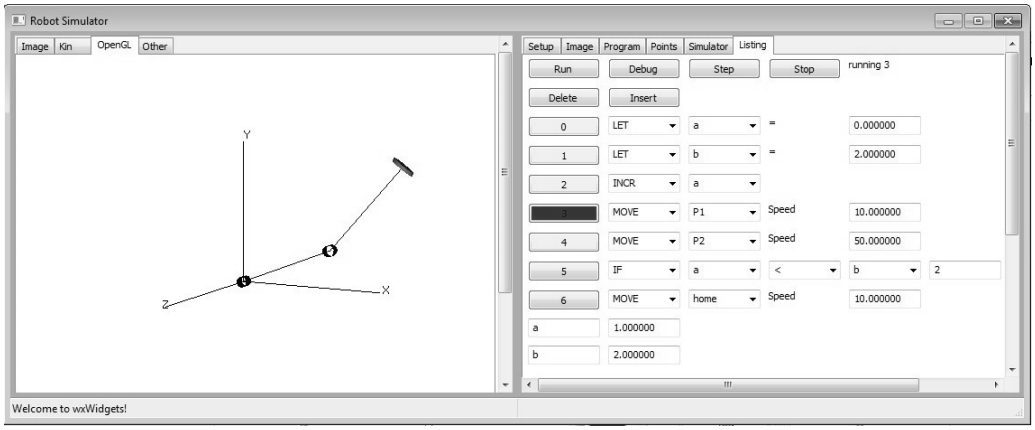


Figure 13: The complete tool screen after executing instruction 3. The red button tells the current instruction being executed. The arm moves in real time as the program executes.

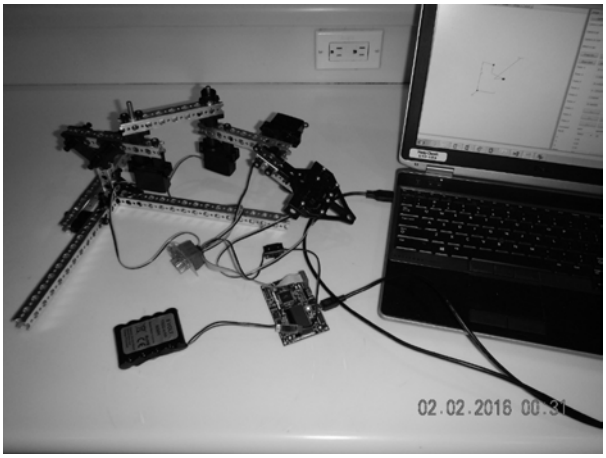


Figure 14: The complete system including the software tool running on the PC, the microcontroller board, the dedicated board, the USB cable, and an example arm built using the robotics kit.

a. The Software Tool

The robotics educational tool used in this work was developed by the authors and is specifically designed to teach the basic Introduction to Robotics undergraduate course. This course generally covers robotics fundamentals including history, robot types, and degrees of freedom, robot kinematics including the transformation matrix, forward and inverse kinematics, and the D-H parameters, differential motions, robot dynamics, trajectory planning, actuators and sensors, and robot vision. The tool displays a virtual robotic arm and a panel of controls (see Figure 11). The virtual arm is entered into the tool by specifying the arm's D-H parameters and the joint limits for each joint. With these numbers the tool can accurately represent a virtual arm with the precise kinematic motions of the physical arm it's modeling. The arm may not look like the real arm since it is represented using stick figures, however it has the same kinematic characteristics. Consider the way we represent the motions of humans using stick figures.

This tool supports teaching the following:

- The relationship between the standard corresponding arm.
- The forward kinematics equations.
- How to use the inverse kinematic equations to program the arm.
- How to program the arm at a high level by defining points and using the move instruction.
- How to design a trajectory

The tool is written in C++ using wxWidgets for its graphical user interface (GUI) and OpenGL for its 3D graphics.

Once the arm is created, the student can perform a variety of robotics activities of which one is to control the movement of the arm by running a program written in a robotic language. The student writes and enters a program to control the movement of the arm in order to perform some task. The tool runs the program and moves the virtual arm which is rendered in three dimensions.

The tool has an internal model of the joint motors in the virtual arm. To move a joint, the software must send a velocity command to each motor and allow it to move over time. This was designed to more accurately model a real arm. The joint model determines the angular position of each joint and the rendering routine then uses these angular positions and the forward kinematic equations to render the arm in its correct position. The kinematic equations are derived from the D-H parameters.

The tool was modified to send a copy of each velocity command to the controller board at the time it sends it to the simulated joint motors. In this way the physical arm can move in parallel to the virtual one. This communication was performed using the RS-232 protocol.

b. The Pitsco Tetrix Prime robotics kit

The Tetrix Prime robotics kit from Pitsco Education, see Figure 15 (a), is designed for middle school students to learn robotics and is used in competitions such as the FIRST Tech Challenge competition. This kit allows students to design and build a robot out of the parts of the kit. This kit was chosen because it has many parts designed to build a robotic arm and even includes a gripper. The wheels and 2 continuous rotating motors were not used in the workshop as industrial robotic arms are generally stationary. This kit includes 2 Hitec servomotors, See Figure 15 (b) that are used to move the links of the arm. An extra motor was purchased for each kit to control all 3 links.

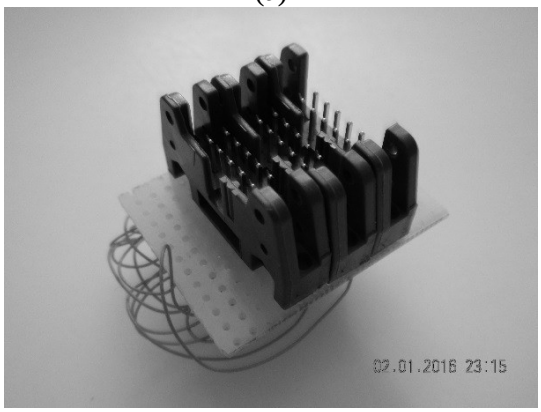
c. The Microcontroller Board

The Xplained microcontroller board shown in Figure 16 (a) is an educational board offered by Atmel to support learning how to use the Atmel XMEGA-A3BU microcontroller and related peripherals. A microcontroller is a small computer in a single chip generally designed to control physical equipment such as a microwave oven for example. This board is only used to physically control the HI-TEC servomotors included in the Tetrix kit. It receives instructions from the PC via its RS-232 serial port and produces electrical signals to control the servomotors. The use of an Arduino or Raspberry Pi microcontroller board may be used and perhaps would have resulted in an easier implementation, however many of these Xplained boards were already available and the authors were experienced using it. The dedicated board shown in Figure 16 (b) was made for this purpose and has the simple task of connecting the ribbon cable connected to the Xplained board to the individual connecters for each motor

(a) (b)

Figure 15: (a) The Tetrix Prime robotics kit by Pitsco Education, (b) The Hitec servo motor.

(a)



(b)

Figure 16: (a) The Xplained microcontroller board, (b) The custom board.

The Atmel microcontroller board was created to provide the electrical signals to each of the servomotors in the kit. It communicates with the software tool via an RS-232 port. The Atmel board comes with a Universal Synchronous Asynchronous Receiver Transmitter (USART) device that handles RS-232 communications. The software on the PC must communicate using the RS-232 protocol. Fortunately Atmel provides a driver that turns a USB port on the PC into an RS-232 port and the Atmel board comes with a USB connector so a standard USB cable connected between the PC and the board is all that is needed, see Figure 17.

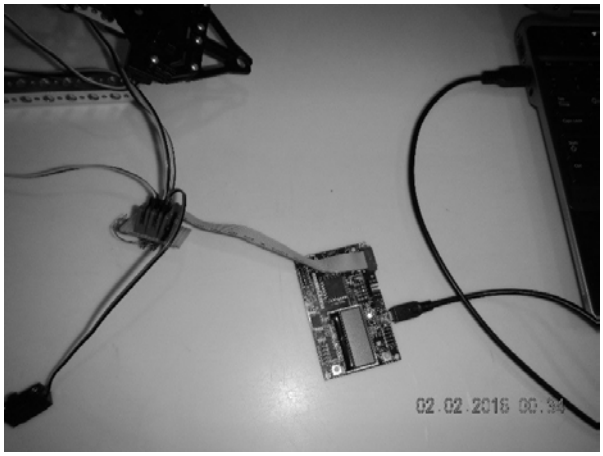


Figure 17: The USB connection between the PC and the Atmel board.

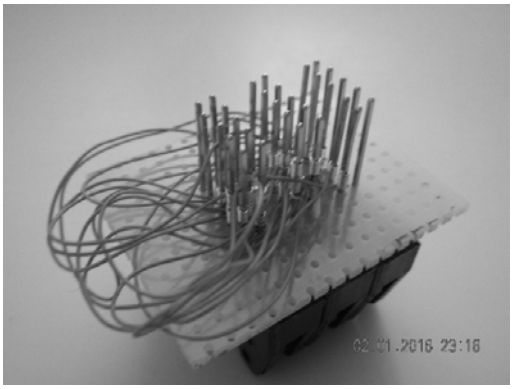
Once the communication link between the PC and the board is established, to move the arm, the PC sends a velocity and duration pair for each motor. So for each servo motor, the board receives an angular velocity that it must travel at and a destination angle that it must turn to. The board needs to remember the angle each servo motor is at in order to start from the correct position.

The board then must generate a Pulse Width Modulation (PWM) electrical signal to each servomotor. This signal tells the servo motor the angle to move to. The software on the board uses a periodic timer interrupt to create time slices or time events that occur periodically. At each time event it determines the angular increase each motor will be driven to before the next time event. The velocity of the individual motor along with the time event frequency are used to determine the angle increment the motor must move by before the next time event. By controlling the angular increment at each time event, the motor's velocity is precisely controlled.

d. The dedicated circuit board

Since the microcontroller board has a 10 pin double row header interface to its ports and the servomotors each have a 3 pin single row header, an interface board was created to connect the board to each motor and the power from the battery pack, see Figure 18 (a). Each motor has 3 wires, a positive and negative power source and the PWM signal input. The battery pack included with the kit provides the positive and negative power lines and the microcontroller provides the control signal. The microcontroller's port has 8 output lines of which 3 are used to deliver the 3 control signals to the servomotors. Wire wrap technology was used for its simplicity and development turnaround time however printed circuit boards will be made for future camps, Figure 18 (b).

(a)



(b)

Figure 18: (a) The dedicated board with the ribbon cables connected, (b) the wires that make the connections.

e. The Servomotors

The kit includes two Hitec HS-233 servomotors and we purchased an additional HS-485HB servomotor per kit. Each motor has 3 wires, a

positive voltage of 5V, a ground and the control signal. The power comes from a battery pack that's included in the kit. The control signal must be generated by the microcontroller. Both servomotor types are controlled by sending it a PWM signal through the control line. This signal tells the motor the angle to move to. The motor moves at full speed to the specified angle. The microcontroller board controls the speed by giving it incremental destination angles over the course of a specified time period. The control signal is a PWM wave with a 50 Hz frequency and a voltage between 3 to 5 volts. A PWM signal is a square wave where the percent of time the wave is high is called the duty cycle. Since the frequency is 50 Hz, the period, or time between waves, is 20 ms. The motor is capable of turning 180 degrees total (– 90° to 90°). The duty cycle of the signal tells the motor the angle to move to. The – 90° corresponds to a duty cycle of 4.5%, the center, 0° is 7.5% and 90° corresponds to a 10.5% duty cycle. Figure 19(a) shows an example of PWM signal that tells the servomotor to move to an angle of 35 degrees. Note in Figure 19(b) the duty cycle must be 8.65% which corresponds to 1.73 ms high and 18.27 ms low.

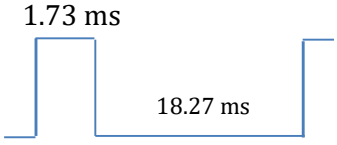
The Execution of the SRO Summer Camp

For over a decade, the Whitaker Center for STEM Education has sponsored a 40 hour, two week event for middle school students interested in STEM. These students must have participated in the Thomas Alva Edison Regional Science Fair. They must prepare an application and are chosen to participate in the Summer Research Opportunity at Florida Gulf Coast University offering them a hands-on experience of collaboration on a complete and genuine research problem from hypothesis generation and initial design, through field and laboratory data collection, and culminating in data analysis and interpretation. The program is staffed by FGCU faculty and Graduate Student Assistants and takes place during each summer.

In the summer of 2015, the students were asked to design a robotic arm that can hold a pencil and write their name on a sheet of paper. Since there were only 3 servo motors per kit and we needed all three to control the joints, we did not allocate a motor to the gripper but instead taped the jaw closed holding the pen. The summer camp had 29 participants. The camp was conducted by a team consisting of 3 middle school teachers, 4 undergraduate students, 2 students from the previous year's camp, and the instructor. Only 3 undergraduate students and the instructor had knowledge of robotics. Ten kits were used and the students formed groups of 3 each using one kit. The software tool was given to each student so they could work independently in the camp or at home.

The software system had errors in it at the time that prevented the physical arm from synchronizing with the virtual arm. The delay in the serial communications between the PC and the microcontroller board was not considered so when the virtual arm arrived at a point it immediately sent a command to the physical arm to start moving towards the next point. The arm would then abort the remainder of the path and immediately start moving towards the next point. As a result, the shapes drawn looked deformed. Figure 20 shows a sample drawing from one of the teams displaying the word "Hi."

This error cause frustration which is reflected in the surveys the students took at the end of the camp. However, even the distorted drawing they were able to produce resulted in the students displaying great excitement. Other factors such as a weak grip on the pen and physical play in the arm's joints also produced distortion however the students seamed to understand these characteristics.



(a)

$$7.5\% + 35° \frac{10.5\% - 4.5\%}{90° - (-90°)} = 7.50\% + 1.167\% = 8.67\%$$

(b)

Figure 19: (a) the PWM signal used to move the servomotor's position to 35 degrees, (b) the calculation for computing the required duty cycle for a 35 degree angle.

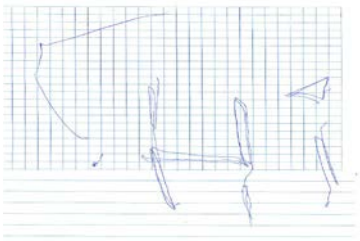


Figure 20: a sample drawing from one of the arms where the arm drew the word “Hi.”

The best way to find errors or weaknesses on a software product is to give it to a set of users. Young children are exceptionally good at finding error. Most of the errors discovered during the camp were fixed as soon as they were discovered. Every morning each team would start by getting a new copy of the software tool with all the errors discovered the previous day fixed.

Assessment

Observations show that the students were more comfortable designing their arm with the kit as opposed to first designing it on paper. They were eager to start building. It was hoped that the student would first model their theoretical design and test its effectiveness using a virtual arm based on their model before building. We acknowledge that putting a toy in front of a child and telling them not to use it yet is impossible. We also acknowledge that it’s much harder to measure the D-H parameters from a paper design than when the physical arm is present. However, they did prefer to program their virtual arm first before running their program on the physical arm. It was encouraging to see that they appreciated the use of their virtual arm in developing their programs. There were two reasons for this preference. The first is that the virtual arm was more reliable and did not require the setup time the real arm took before executing the program and the second reason is that all 3 members of the team could program the arm concurrently since they each had their own copy of the virtual arm and the software tool. By using and appreciating the use of the virtual arm they demonstrated the understanding of the concept of modeling and simulation in the development process.

It was also noted that the students were very motivated to work on their arm. Typically, not wanting to stop for lunch and wanting to stay at the end of the day. We believe the students were motivated by the challenge of having to develop a program to control their arm as opposed to just building it and controlling it manually with a remote control unit.

Although the students did seem to understand what the D-H parameters are, they were not able to learn how to measure these parameters and needed help from the undergraduate students. Learning how to measure these parameters is considered difficult for undergraduate and even graduate students.

At the end of the camp all teams had a working arm that could write. All teams were able to design a functioning arm that could perform the task and develop a program to move the arm.

The only assessment tool that was used was a survey the students filled at the end of the camp. It was designed to assess the camp itself and the effectiveness of the staff and not for assessing the robotic system. Twenty-two participants completed the survey. Eighteen of them indicated their favorite part of the camp was building the robot. Sixteen students indicated that their least favorite part was programming the arm, one student indicated it was finding the D-H parameters, and 2 indicated their least favorite part was starting over with a new design once their old design was proved ineffective. We believe the frustration with the programming was a result of the errors in the software tool. This frustration was observed during the camp. Six students indicated the lectures teaching them how to find the D-H parameters and how to program were a bit over their heads. And finally all the students who filled the survey indicated that they had fun.

Future Work

To help the students understand how to measure the D-H parameters, we plan on adding a feature that will guild the student through the process. Like a wizard that will display different options and the student can select the appropriate one. Error checking will be increased to prevent the program from entering invalid states.

It is envisioned that this tool will be made available to all who have an interest, however the software is not polished or fully tested at this time. The method for offering the software is also not decided at this time.

Conclusions

We presented a new robotics based educational system for K-12 students. Our goal was to go

beyond the traditional activity of building a mobile robot and control it using a remote control to an activity more closely related to the robotics profession including introducing the concept of modeling and simulation. We gave a summer camp to middle school students that showed they had some difficulty learning how to measure the D-H parameters of their arm but were able to use the virtual arm to develop their program before running it on the actual arm. The survey showed they had fun and observations showed that they did learn the modeling and simulation concept and were able to use it to speed the development of their project.

References

1. FIRST Robotics, Link: http://www.firstinspires.org/
2. Tetrix Robotics, Link: http://www.tetrixrobotics.com/
3. Atmel ATxmega256A3BU microcontroller based XPlained board, Link: http://www.atmel.com/tools/XMEGA-A3BUXPLAINED.aspx
4. F. Gonzalez, J. Zalewski, G. Pinzon, "An Educational Tool to Support Introductory Robotics Courses,"*Proceedings of the. 122nd ASEE Annual Conference*, Seattle WA. June 14-17, 2015. Paper No. 13128.
5. Florida Gulf Coast University's Whittaker Center for STEM Education, Link: http://www.fgcu.edu/WhitakerCenter/sro.html
6. Denavit, J., R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, June 1955, 215-221.
7. Saeed B. Niku, "Introduction to Robotics, Analysis, Control, Applications," 2nd Ed. Wiley, 2011.
8. He Shouling, Jefferson Maldonado, Alex Uquillas, Terry Cetoute, " Teaching K-12 Students Robotics Programming in Collaboration with the Robotics Club," *Proceedings of the 4th IEEE Integrated STEM Education Conference*, 2014.
9. Tetrix Prime for NI myRIO, Link: http://www.ni.com/white-paper/52707/en/
10. National Instrument's myRIO, Link: http://www.ni.com/myrio/
11. LabVIEW: http://www.ni.com/labview/