MATLAB Tool Allowing Wireless Control of Arduino Robot for Early Introduction of Robotics into Curriculum

Mr. Connor Daniel Olsen, University of Utah

Connor Olsen is Ph.D. Student and NSF Fellow at the Univeristy of Utah working in the Utah NeuroRobotics Lab. His work centers around the creation of a wrist-worn smart device that uses electromyography and other sensors to detect hand gestures to control a smart home environment, which can be used by paralyzed individuals to increase autonomy in their own living space and encourage rehabilitation

Dr. Amy Verkler, University of Utah

Amy Verkler is an assistant professor (lecturer) in the electrical and computer engineering department at the University of Utah. She completed a PhD focused on engineering education at Stanford University in 2021.

Daniel S. Drew, University of Utah
Jacob A. George, University of Utah

MATLAB Tool Allowing Wireless Control of Arduino Robot for Early Introduction of Robotics into Curriculum

Connor D. Olsen, Amy V. Verkler, Daniel S. Drew, Jacob A. George

Abstract
In modern Electrical Engineering degree programs, MATLAB is often one of the first coding experiences a student is exposed to. Most introductory robotics courses that combine hardware and software require students to understand C (typically learned during junior year) or require part of the course to teach coding syntax. In order to introduce robotics and cyber-physical systems earlier in the curriculum, we have developed an interface to allow students to remotely control a wireless microcontroller (e.g., Arduino MKR 1010) using MATLAB. This interface comprises two halves: 1) a MATLAB class that abstracts UDP commands transmitted over Wi-Fi, and 2) a custom C++ library for receiving, parsing, and responding to commands over UDP, as well as streaming data back to the client. The interface leverages students' existing knowledge of MATLAB and bypasses the need for C programming, allowing students to get early exposure to hardware-software integration, signal processing, edge computing, end-to-end platform development, and systems engineering. Our interface facilitates data observation, recording, manipulation, and analysis. Students have access to live data streams, real-time plots of sensor values, and the ability to use the command window to run and test individual commands outside of scripts. We deployed this system in an introductory class where students perform various mechatronic lab exercises and complete a final project where their robot navigates a maze then collects and classifies objects using sensor data and neural networks. We surveyed two semesters of students at the end of the course, and students reported that using this interface enhanced their learning experience despite varied responses about the difficulty of implementation. With the growing importance of data science in electrical engineering, tools like our interface play a crucial role in exposing students to cutting-edge robotics and cyber-physical systems earlier in the degree program. Our interface has been made available on GitHub for any who wishes to implement it.

Introduction

The advent of Arduino microcontrollers has provided a more user-friendly and approachable method for introducing topics in robotics and embedded programming [1]. It is common to find Mechanical Engineering departments teaching mechatronics courses that cover Arduino programming alongside basic circuitry, sensors, and actuation [2]. These courses are intended for lowerclassmen (freshman, sophomore) and require the students to learn the basics of programming and Arduino syntax, while occurring early enough in the student's career that they are introduced to these exciting topics while still discovering their interests [3].

In contrast, Electrical and Computer Engineering (ECE) students typically learn C/C++ from the Computer Science department before later learning how to use with a focus on low-level programming of embedded systems [4]. Many ECE departments lack a course with a low barrier to entry that introduces the exciting topics covered in Mechanical Engineering's mechatronics course. Although there has been debate in the academic community about the effectiveness of using Arduinos to teach embedded programming, many universities have successfully implemented them in the classroom with positive response from the students [5], [6]. While many ECE students are familiar with Arduino within our department, this familiarity generally comes from personal experience gained outside the classroom. In our ECE department, students begin learning MATLAB in their introductory courses. Our department's decision to use MATLAB was influenced by its graphics capabilities and the university-wide license providing free access for faculty and students. Additionally, MATLAB's existing Toolbox add-ons provide significant utility for the curriculum, particularly for later courses like Antennas and Signal Processing. Although students gain early familiarity with MATLAB, they lack formal education on programming Arduinos. Arduino programming can be accessible to a beginner, but manipulating and visualizing sensor data is non-trivial, and higher-level functions (e.g., machine learning) are challenging to implement. For example, Plata et al. successfully demonstrated the use of Arduinos for an early introduction to topics in robotics but dedicated eight of the sixteen class sessions to "getting to know Arduino" [7].

At the University of Utah, we have developed a new course for sophomore-level ECE students to introduce topics related to robotics and cyber-physical systems that use Arduino hardware paired with a unique MATLAB interface (the "MATLAB-MKR Interface") to bypass the need to learn C in class. At the sophomore level, enrolled students have at least completed an introductory MATLAB course and an introduction to circuits. This allows us to take advantage of the user-friendly nature of the Arduino hardware while leveraging students' preexisting familiarity with MATLAB. By removing the time spent learning coding from our course, we were able to add an additional unit (alongside the traditional robotics units covering sensing and actuation), which covers the basics of intelligence and machine learning (See Table 2). This addition has been the most notable difference of our course and other similar courses taught by other departments at our university.

Though MATLAB has a toolbox to communicate with Arduino microcontrollers in real-time, when this interface was created (Summer 2021), this toolbox only worked reliably with wired microcontrollers. The toolbox did not fit the needs of our course, where dynamic wireless interfacing with the microcontroller was required to complete the final project, where the wheeled robot navigates an obstacle course. Unfortunately, the toolbox did not function correctly to allow for wireless communication, specifically with the MKR 1010, and at the time, there was scarcely any online support available. For this reason, we chose to build our own interface. Our interface allows students to collect data (either individual samples or with a buffered stream) and filter, manipulate, and process it using MATLAB's more comprehensive toolkits. Additionally, it allows the students to access peripheral devices and General-Purpose Input/Outputs (GPIOs) of their microcontroller wirelessly, run commands outside the script using the MATLAB command window, and take advantage of MATLAB's debugging and workspace tools. By using this interface, we expanded the range of topics beyond what most similar courses teach, allowing for a greater focus on computational approaches and implementations of digital signal processing, data manipulation, and machine learning [2], [7].

Students reviewed the course positively, indicating that they were more interested in the topics because of this method of introduction and that they were more likely to pursue robotics in the future. Though the student responses varied when asked about the difficulty of using the interface (and no control method was assessed), students generally agreed that this new tool enhanced the learning experience overall.

Methods

<u>Interface Architecture:</u>
The MATLAB-MKR Interface allows for wireless communication between the student's computer and the Arduino MKR. Once connected to power, the Arduino MKR establishes a wireless access point and configures a static IP address for itself. The student connects their computer to the MKR's access point and begins sending UDP commands using MATLAB. A MATLAB class encompasses all accessible commands, manages initial setup and communication, and abstracts the available functions. The class contains a callback function that asynchronously reads all incoming UDP messages from the MKR. The MKR itself is programmed to wait for the initial UDP connection, and then execute the incoming commands as they are received (see Figure 1A).

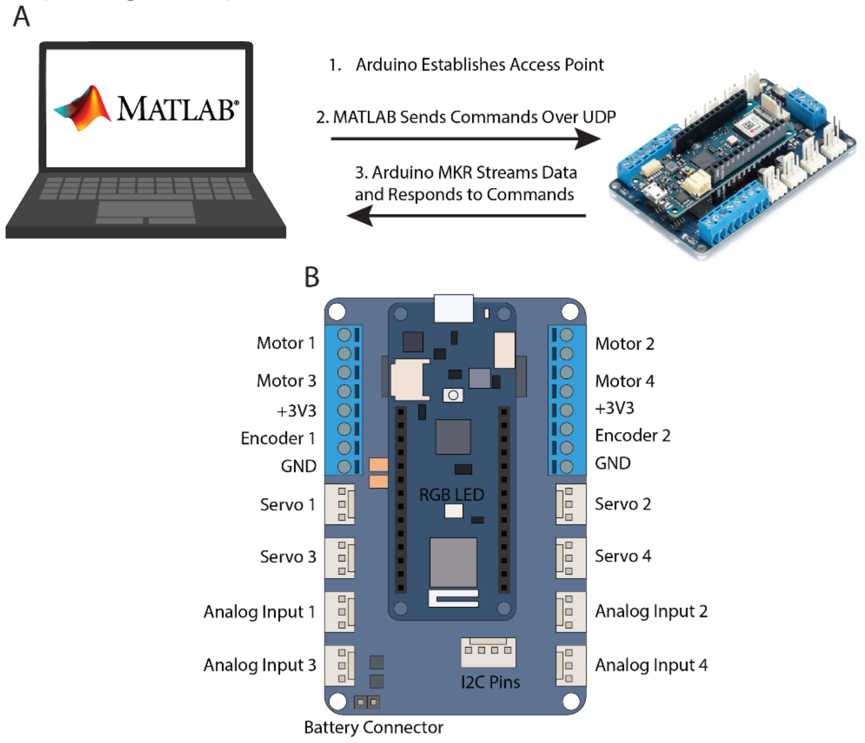

Figure 1: MKR-MATLAB Interface Overview. A) The MKR 1010 connection sequence with MATLAB. B) Pinout of the Arduino MKR 1010 mounted to the MKR Motor Carrier.

<u>Hardware Design</u>
The Arduino MKR 1010 was selected as the microcontroller for this course. The MKR 1010 possesses an onboard Wi-Fi module, which enables wireless communication between the computer and the microcontroller. It has 256 KB of flash memory and has a clock speed of 48

MHz. It is a compact board, measuring 61.5 mm long by 25 mm wide [8]. It also has an RGB LED mounted to the center of the board, which can be used as a diagnostic tool or an output. It is also relatively inexpensive (about 30 USD) and can be reused from semester to semester.

A major advantage of the MKR class Arduino boards is that they can be easily mounted to the Arduino MKR Motor Carrier (see Figure 1B) [9]. The MKR Motor Carrier allows for higher power actuator usage, facilitating control of up to four motors and four servos, and contains ports to attach I2C peripheral devices and two encoders. This board eliminates the need for several typical cyber-physical system components (e.g., H-bridge motor drivers) and keeps the hardware together in a small form factor.

The MKR Motor Carrier reserves pins A3, A4, D2-D6, and D11-D12 for their functionality; the MATLAB class checks for proper usage to prohibit and/or notify students when they are using reserved pins. Throughout the course, students attach a variety of different passives, sensors, and peripherals to the MKR Motor Carrier, including resistors, potentiometers, FSRs, motors, servos, encoders, accelerometers, Hall-effect sensors, ultrasonic sensors, infrared reflectance sensors, and photoresistors.

Software Design

The Arduino MKR was programmed to establish a wireless access point and await commands over UDP from an external device (e.g., a student running MATLAB on a laptop or classroom desktop). The MKR remains waiting, responding to commands as they are received.

When a command is received to read from a peripheral device or a GPIO pin, for example, the Arduino responds with the value. Several data streams have been established to facilitate data transfer when several different data values are needed, which will continuously measure and transmit data without being polled each time. The Arduino code defines the following four different data streams: *Analog*, which sends the integer values of the four available analog input pins; *Inertial Measurement Unit*, which sends the X, Y, and Z accelerometer float values; *Ultrasonic Sensor*, which streams the time-of-flight integer value from the ultrasonic sensor; and *Infrared Sensor*, which returns the four integer values from an infrared reflectance sensor array. These streams are turned on using a 'start' command and will continue to stream data until a 'stop' command is received. These streams may be used individually or in any combination with each other. In most cases, broadcasting all four data streams simultaneously is unnecessary. Activating them all together can result in the Arduino operating below its designed 200 Hz loop speed.

The MATLAB code represents the part of the interface that the student interacts with and has been built with input error checking to ensure that only viable commands can be sent to the Arduino. After connecting to an access point, the student can establish communication with the Arduino by constructing an instance of the class type *MKR_MotorCarrier*. Once the connection is established, the RGB LED on the MKR will turn green to indicate it is ready to receive commands. The commands available to the student are listed in Table 1, where bolded commands are syntactically identical to native Arduino functions in order to facilitate future skill transfer. The students were also provided with a demo file, which demonstrated all the functionality of the MATLAB-MKR Interface and served as a way to verify hardware during debugging.

Table 1: Commands available to the students through the MATLAB interface.

Function Name	Description	Arguments
General Functions		
pinMode	Sets a digital pin as either an input or an output	"Input" or "Output"
analogRead	Reads and returns the value on the given analog pin	Analog Pin Number
digitalRead	Reads and returns the value on the given digital pin	Digital Pin Number
digitalWrite	Sets the value of the specified pin to a provided value	Digital Pin Number
startStream	Starts a specified data stream	Specific Stream Type
stopStream	Stops a specified data stream	Specific Stream Type
checkFrequency	Tests the frequency of the data transfer	
livePlot	Generates a real-time plot of either the Analog or IMU data streams	Specific Stream Type
getAverageData	Returns the average data of the specified data stream	Specific Stream Type
getNewData	Returns only new data that received since the last time getNewData call	Specific Stream Type
startRecording	Starts recording all data received, if data is streaming	
stopRecording	Stops a recording and returns the recorded data	
setRGB	Sets the RGB LED on the MKR	Red Green and Blue values
getVoltage	Prints the battery level to the command window	
close	Ends communication with the MKR	
Motor Functions		
motor	Sets the value of the specified motor to a provided value	Motor Number and Speed
servo	Sets a value (-180 to 180) to the specified servo motor	Servo Number and Angle
readEncoderPose	Returns the position of the motor encoder	
readEncoderVel	Returns the velocity of the motor encoder	
resetEncoder	Recalibrates the motor encoder to 0	
External Peripheral Functions		
piezoTone	Sends a signal to the piezo buzzer for a specified duration	Frequency and Duration
reflectanceSetup	Initializes the IR reflectance sensor	
readReflectance	Returns the four IR reflectance values	
rgbRead	Returns the red, green, and blue values of the RGB sensor	
ultrasonicPulse	Returns the value of the ultrasonic sensor	

Once the connection with the MKR has been established, commands can be sent one at a time or stacked together in loops and conditional statements from within MATLAB. One of the significant advantages of sending commands over MATLAB (as opposed to programming in Arduino IDE) is the presence of a debugger, which lets students step through each line of code and more easily catch errors within the script. Additionally, manipulating and visualizing data can be done easily in the MATLAB workspace. Students can also take advantage of the Read, Evaluate, Print, and Loop (REPL) environment to test commands and pull sensor readings without the need to run the entire script. Above all, the interface enables students to dynamically alter the commands sent to the MKR without needing to recompile code and program the microcontroller.

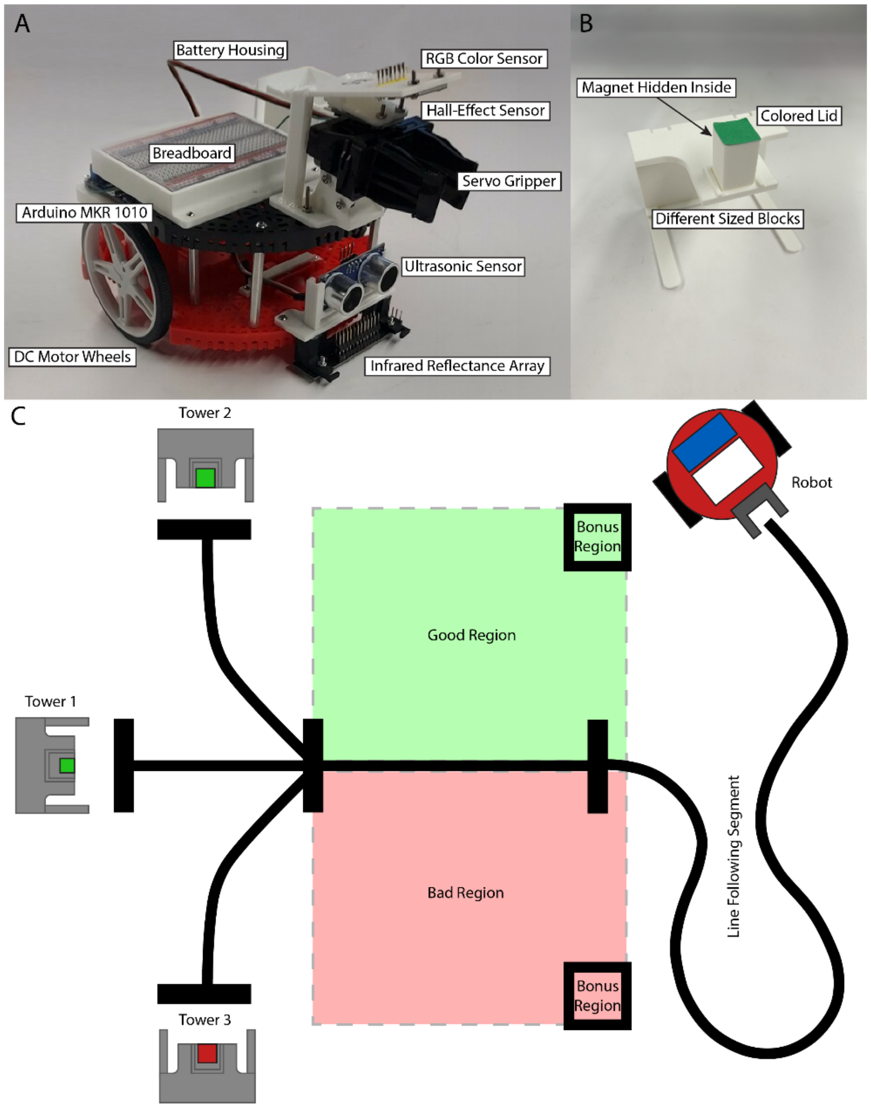

Figure 2: Course final project. A) Labeled model of the final project robot. B) Image of the towers holding the blocks to be gathered and classified by the robot. C) Map of the final project line following the course.

Labs and Activities

This interface was used for all the labs in this course. The activities where this interface provided the greatest benefit were those requiring continuous streaming of sensor data. For example, the Simultaneous Localization and Mapping (SLAM) module went over the basics of that field and introduced ultrasonic range sensors. Students were able to stream ultrasonic sensor data while walking about the classroom to see how this sensor could be used for obstacle avoidance. The Quadrotor Control and Path Planning Module introduced the idea of simultaneous sensorimotor loops, allowing students to draw on sensor data in real time (such as a force-sensitive resistor) to control an actuator (such as a servo motor).

The distinct addition of our class over other similar courses is the inclusion of five classes dedicated to intelligence modules, teaching students about machine learning and classification at varying levels of complexity (ranging from Linear Discriminate Analysis to Convolutional Neural Networks). In this module, students attached an accelerometer to the end of a 12” dowel and used it as a pen to draw numbers (0-9) in the air. Over the course of these modules, they would use different supervised learning techniques to train algorithms that could accurately predict which numbers were being drawn in real time.

Final Project

The final project was designed to allow the students to combine the experience gained throughout the course into a single robot mission. The students were each provided with all the needed materials and tasked with navigating a path to collect and classify different blocks. Students worked in groups of three or four, allowing them to divide the project among themselves and work on different parts in parallel. This project helped solidify the various concepts taught throughout the course and gave real-world examples for their implementation and use.

For this project, the MKR 1010 (mounted to the MKR Motor Carrier) was attached to a robot chassis (see Figure 2A) with mounted peripherals including an ultrasonic time-of-flight distance sensor, an infrared reflectance array, an RGB color sensor, and a Hall-effect sensor. The chassis had two DC motors with quadrature encoders to provide left and right directional control and a servo motor with an encoder that served as a gripper.

The students then wrote a script for the robot to navigate a course using infrared reflectance measurements to follow a line (see Figure 2C). Following the lines, the robot would encounter a three-way split and navigate all possible paths to find towers holding up blocks (See Figure 2B). The robot could approach the tower precisely using the ultrasonic sensor and grab the block with the gripper. Each block was either small (2.5 cm) or large (3.0 cm) and had a specific color and magnetic field polarity (established by permanent magnets fixed to the inside). The Hall-effect sensor, RGB color sensor, and gripper’s encoder would then measure the magnetic polarity, block color, and size, respectively. Based on these three metrics, the robot would classify the block as either “good” or “bad.” The robot would then navigate to and drop the block in the appropriate zone.

A training set of 25 blocks was provided to the students to create a model of good and bad blocks. When running the course to earn points, the three towers held blocks from a testing set of five different and previously unseen blocks that fit the same criteria as the training set. Students were free to choose their own kind of supervised learning model from the many learned about in class to classify the blocks.

Points were awarded based on the completion speed and classification accuracy. Partial points were awarded for meeting certain milestones (e.g., successfully navigating the curved line-following portion, approaching the tower, and grabbing a block). Students were allowed two days to run the course for points and were allowed to try as many times as possible.

Results

At the end of the semester, students were asked to respond to a questionnaire about their experience with the MATLAB-MKR interface and the general class. Of the 44 students who have taken the course (15 in the first semester and 29 in the second), 26 responded to the survey. When asked, "How difficult was it to program the robot through MATLAB," students responded on a 5-point Likert scale, with 1 being 'Difficult' and 5 being 'Not Difficult.' The average student response was 2.8, with a standard deviation of 1.34 (Figure 3A). When asked to agree or disagree with the statement "Programming the Arduino MKR in MATLAB enhanced my learning experience," students responded with a median response of "Agree" (Figure 4B).

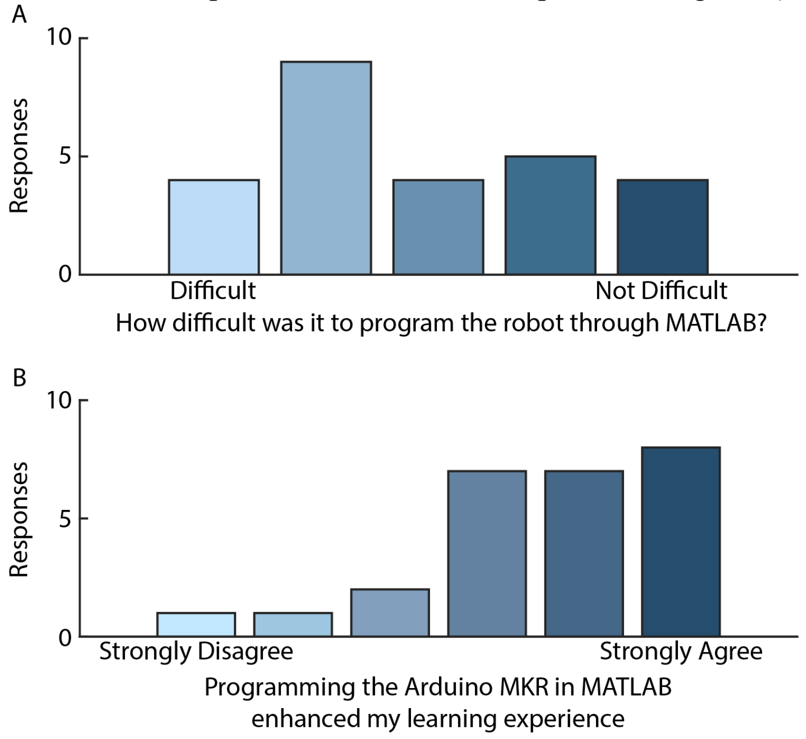

Figure 3: Student survey responses regarding the MKR-MATLAB Interface.

Regarding the class as a whole, students were asked to agree or disagree with the statements on a six-point scale (scale size based on university course feedback standards). When prompted, "I am more interested in pursuing robotics as a career now, having taken this course," the median response was "Agree" (Figure 4A). When prompted, "I would recommend this class to anyone who is interested in studying robotics," the students responded with a median answer of "Strongly Agree" (Figure 4B). When prompted, "the electronics materials in this class were appropriate for what we learned," the median response was "Strongly Agree" (Figure 4C). When prompted, "I achieved the learning outcomes from this class," the median response was "Strongly Agree" (Figure 4D).

At the end of the provided questionnaire, the students were asked about their favorite lab in the class and why. Responses varied evenly across each of the thirteen different labs. Of these, only three labs weren't selected as a favorite among the students (N = 26, See Table 2). When responding to why the selected lab was their favorite, 42% (11 of 26) of respondents mentioned that the lab was novel or provided exploration into fields beyond prior coursework, 15% (4 of 26) stated the lab offered practical and real-world relevance to important theoretical concepts, and 42% (11 of 26) stated the lab provided a deeper understanding of fundamental concepts, filling knowledge gaps.

Table 2: List of modules covered in the course and the percentage of students who voted that module as their favorite.

Unit	Module	% Reported as Favorite Module
Sensing	Microcontrollers and ADCs	3.8
	Transduction Principles	0
	MEMS Sensors and Communication Protocols	7.7
	Noise and Filtering	0
	Simultaneous Localization and Mapping (SLAM)	0
Actuation	DC Motors and Encoders	15.4
	Other Actuators and Drive Circuits	11.5
	Feedback Control Basics	11.5
	Quadrotor Control and Path Planning	3.8
	Wheeled Robot Modeling and Odometry	11.5
Intelligence	Linear Discriminate Analysis Classification	15.4
	Perceptron Classification	3.8
	Convolutional Neural Networks	15.4

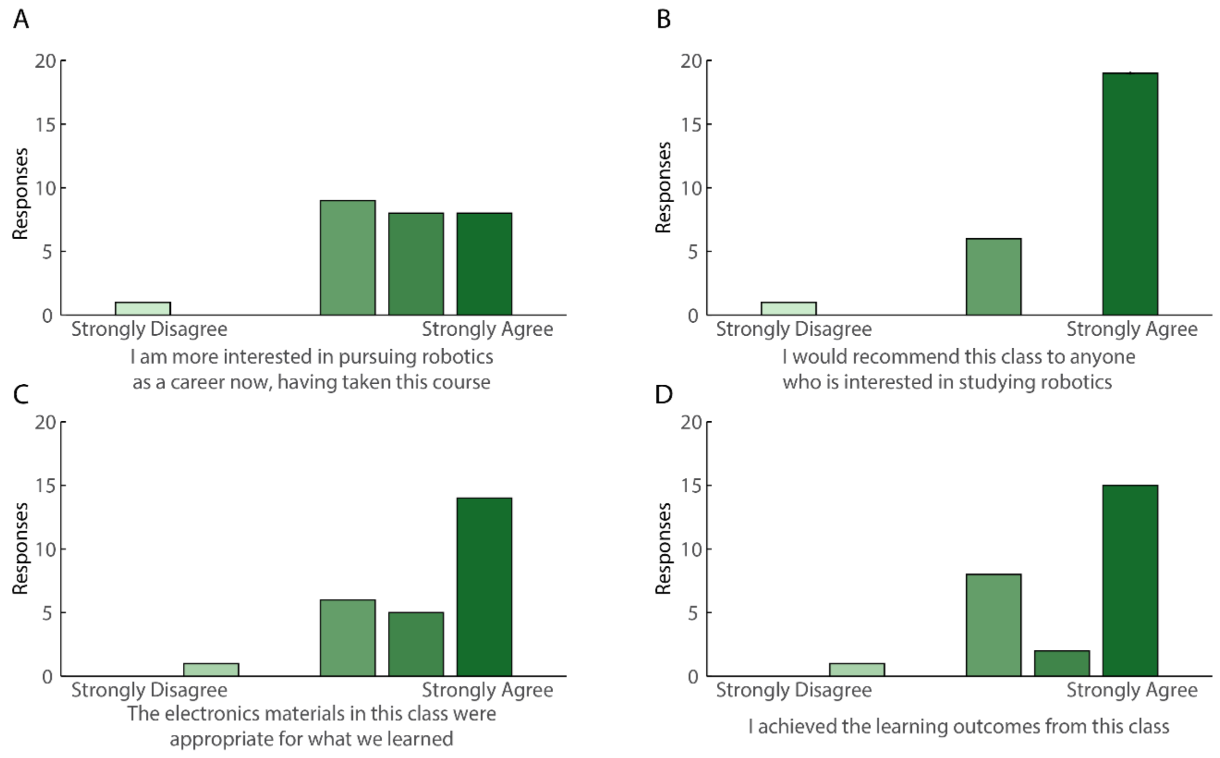

Figure 4: Student survey responses regarding the class.

Discussion

While the average response to the difficulty of the interface leaned more toward “difficult,” than “not difficult”, the class responses were well-distributed across this spectrum. Though not specifically asked, we speculate that a similar response distribution may have accompanied a question about the difficulty of programming the Arduino using the Arduino IDE and coding language. Another contributing factor to the reported difficulty may have been the wireless nature of the interface, which occasionally produced problems such as dropped connections or periods of high latency. Interestingly, despite the fact that the average student regards the interface as difficult, they also agree that using the interface enhances their learning experience. We speculate the reasons for this are: 1) Using the interface allowed them to quickly run and edit scripts without needing to reprogram to the robot, and 2) Running scripts through MATLAB allowed the students to take full advantage of MATLAB’s more robust debugging system and documentation.

Regarding the class, the students gave overwhelmingly positive responses. Students agreed that the course materials were appropriate and that they were more interested in the subject as a result. It should be noted that three of the four negative responses seen in Figure 4 all came from the same student. The varied responses about the students’ favorite labs are also encouraging, and we believe that this indicates that each lab module is well-balanced and structured in a way that provides an enjoyable and interesting learning experience. Though three labs weren’t selected as favorites, SLAM was mentioned by several students in their qualitative response as another lab from which they benefited greatly. We find it especially encouraging that 34.6% of students selected labs involving machine learning as their favorite. The machine learning labs arguably benefitted the most from using the MATLAB-MKR Interface since teaching machine learning concepts in C would otherwise require substantial class time dedicated to lower-level programming logic and syntax.

Because this is a new course in our department, there is no previous student performance with which we can directly compare. This makes it difficult to say whether the inclusion of this interface tool is what made the class so successful. However, student responses indicate that the course was a valuable elective, and the interface, though difficult at times, did not take away from the experience. As future cohorts of students take this course, further survey responses will highlight weaknesses in the lab design and the interface, which will provide vital feedback for continuing to develop effective teaching tools.

Conclusions

We developed a novel interface between MATLAB and an Arduino microcontroller to introduce undergraduate students to hands-on robotics concepts earlier in their education. Using this tool allowed us to eliminate class time devoted to teaching C programming in favor of covering a broader range of topics. Responses indicated that using this tool enhanced their learning experience despite sometimes being difficult to use. Responses about the course indicate the tool assisted in the success of the class, which students generally enjoyed and reviewed positively. We believe that the inclusion of this interface was successful in its design and that students are more interested in pursuing robotics education and future careers as a result. Additionally, we feel confident that this course has provided exposure to the relationships between data science and electrical engineering, as well as edge computing and systems engineering. Over the course of future semesters, we intend to further refine the interface by adding a method for wired serial connection and improving data streaming to allow for a better run-time experience. Additionally, we plan to update the hardware and move from the Arduino MKR 1010 to the Arduino Nano IoT to address the MKR Motor Carrier being discontinued. As the interface continues to improve, we believe it will become more intuitive to use, helping the students to achieve the desired learning outcomes with even more ease.

References

[1] G. Recktenwald and D. Hall, "Using Arduino as a platform for programming, design and measurement in a freshman engineering course," in *2011 ASEE Annual Conference & Exposition Proceedings*, Vancouver, BC: ASEE Conferences, Jun. 2011, p. 22.1609.1-22.1609.23. doi: 10.18260/1-2--18720.

[2] R. Grover, S. Krishnan, T. Shoup, and M. Khanbaghi, "A competition-based approach for undergraduate mechatronics education using the arduino platform," in *Fourth Interdisciplinary Engineering Design Education Conference*, Mar. 2014, pp. 78–83. doi: 10.1109/IEDEC.2014.6784685.

[3] R. Chancharoen, A. Sripakagorn, and K. Maneeratana, "An Arduino kit for learning mechatronics and its scalability in semester projects," in *2014 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Dec. 2014, pp. 505–510. doi: 10.1109/TALE.2014.7062606.

[4] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "Incorporating embedded programming skills into an ECE curriculum," *SIGBED Rev.*, vol. 4, no. 1, pp. 17–26, Jan. 2007, doi: 10.1145/1217809.1217813.

[5] P. Jamieson, "Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?," Apr. 2012, Accessed: Sep. 12, 2023. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.drpeterjamieson.com/html_papers/fecs_11.pdf

[6] J. Sheng, "Teaching Devices and Controls for Computer Engineering and Systems Students using Arduino and MATLAB/Simulink," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, Anchorage, AK: IEEE, Jun. 2018, pp. 318–323. doi: 10.1109/ICCA.2018.8444192.

[7] P. Plaza *et al.*, "Arduino as an Educational Tool to Introduce Robotics," in *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Wollongong, NSW: IEEE, Dec. 2018, pp. 1–8. doi: 10.1109/TALE.2018.8615143.

[8] "Arduino MKR WiFi 1010," Arduino Online Shop. Accessed: Sep. 15, 2023. [Online]. Available: https://store-usa.arduino.cc/products/arduino-mkr-wifi-1010

[9] "Arduino MKR Motor Carrier," Arduino Online Shop. Accessed: Sep. 15, 2023. [Online]. Available: https://store-usa.arduino.cc/products/arduino-mkr-motor-carrier