

EXAMINING THE IMPACT OF CURRICULAR AND ROBOTIC INTERVENTIONS

By

Ayana A. Wild

Thesis

Submitted to the Faculty of the  
Graduate School of Vanderbilt University  
in partial fulfillment of the requirements  
for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

December 18, 2021

Nashville, Tennessee

Approved:

Taylor T. Johnson, Ph.D.

Julie L. Johnson, Ph.D.

## **ACKNOWLEDGMENTS**

To my parents, for being inspirations and support throughout.

# TABLE OF CONTENTS

	Page
<b>ACKNOWLEDGMENTS</b> . . . . .	<b>ii</b>
<b>LIST OF TABLES</b> . . . . .	<b>v</b>
<b>LIST OF FIGURES</b> . . . . .	<b>vi</b>
<b>I Introduction</b> . . . . .	<b>1</b>
I.1 Concept Inventories . . . . .	1
I.2 Motivation . . . . .	1
<b>II Literature Survey</b> . . . . .	<b>3</b>
II.1 Computing Education Research Overview . . . . .	3
II.2 Equity . . . . .	4
II.2.1 Accessibility . . . . .	4
II.2.2 Disparities . . . . .	5
II.2.2.1 Gender . . . . .	7
II.2.2.2 Racial . . . . .	8
II.3 Computing in Education . . . . .	9
II.3.1 Using Robotics . . . . .	14
II.3.2 Novice Programmers . . . . .	16
II.4 Measure Validation . . . . .	17
II.5 Surveys . . . . .	17
<b>III Evaluation of Student Interest in Computer Science After Early Exposure to Cyber-Physical Systems (CPS)</b> . . . . .	<b>19</b>
III.1 Introduction . . . . .	19
III.2 Methods . . . . .	21
III.2.1 Participants . . . . .	22
III.2.2 Surveys . . . . .	22
III.2.2.1 Questions . . . . .	22
III.2.2.2 Intentions . . . . .	23
III.2.3 Course Setup . . . . .	23
III.3 Results . . . . .	24
III.3.1 Results of Pre-Survey . . . . .	25
III.3.1.1 Programming Background Results . . . . .	25
III.3.1.2 Student Background Results . . . . .	26
III.3.1.3 Class Expectations Results . . . . .	27
III.3.2 Results of Post-Survey . . . . .	27
III.4 Discussion . . . . .	30
III.4.1 Threats to Validity . . . . .	30
III.5 Conclusions . . . . .	31
III.6 Future Work . . . . .	31
<b>IV Tello Interaction Interface</b> . . . . .	<b>32</b>

IV.1	Takeoff and Land	32
IV.2	Shapes Interface	32
IV.2.1	Circle Interface	32
IV.2.2	Square Interface	33
IV.2.3	Triangle Interface	33
IV.2.4	Cube Interface	33
IV.3	Camera Interface	35
<b>V</b>	<b>ES 140X Curriculum Redevelopment Results</b>	<b>36</b>
V.1	Research Questions	36
V.2	Study Design	36
V.2.1	Data	36
V.2.1.1	Dependent Variables	37
V.2.1.2	Independent Variables	37
V.2.1.3	Data Collection	38
V.3	Preliminary Results	38
V.3.1	ES 140X Study	38
V.3.1.1	Questions	38
V.3.1.2	Intentions	42
V.3.2	Pre-Test Results	44
V.3.3	Post-Test Results	49
V.4	Curriculum	51
V.5	Contributions	53
V.6	Potential Future Directions	53
<b>VI</b>	<b>Conclusion</b>	<b>55</b>
	<b>References</b>	<b>56</b>
<b>A</b>	<b>Pre-Test</b>	<b>63</b>
<b>B</b>	<b>Post-Test</b>	<b>77</b>
<b>C</b>	<b>Skeleton Code for Tello Interface Given to Students</b>	<b>89</b>
C.1	Main File	89
C.2	Take Picture File	103
C.3	Record Video File	113

## LIST OF TABLES

Table		Page
V.1	Distribution of programming languages previously known and used by the participants. . .	45
V.2	Distribution of majors declared by the participants on the pre-test. . . . .	45
V.3	Student Race and Ethnic Identity as reported on the Pre-Test (N = 70) . . . . .	47
V.4	Student Race and Ethnic Identity as reported on the Pre-Test (N = 70) broken down by gender. . . . .	47
V.5	Question Correctness (N = 70) . . . . .	48
V.6	Distribution of majors declared by the participants on the post-test. . . . .	49
V.7	Student Race and Ethnic Identity as reported on the Post-Test (N = 42) . . . . .	49
V.8	Student Race and Ethnic Identity as reported on the Post-Test (N = 42) broken down by gender. . . . .	50
V.9	Post-Test Question Correctness (N = 42) . . . . .	51

## LIST OF FIGURES

Figure	Page	
III.1	The Nao robot used in (34). . . . .	20
III.2	The Lego Mindstorms used in (34). . . . .	20
III.3	The drone connection shown as a Cyber-Physical System. . . . .	23
III.4	The DJI Tello Edu quadcopter drone used by the students in the course. . . . .	24
III.5	An example Python program executed on students' laptops that sends control commands to, and receives sensory and other information from, the Tello drones over 802.11. This particular program (1) connects to the drone, (2) has it take off into the air, (3) has it move downward at half (50%) maximum velocity for approximately 5 seconds, and (4) lands, where commands are sequenced in time using waits of 5 seconds. It additionally concurrently displays sensor information as messages are received through the handler function. . . . .	25
III.6	Distribution of programming languages previously known and used by the participants. . . . .	26
III.7	The distribution of the declared majors of the participants on the Pre-Survey. . . . .	26
III.8	The distribution of the racial/ethnic identities of the participants as reported on the Pre-Survey (N = 101) broken down by gender. . . . .	27
III.9	The results of the t-Test (Paired Two Samples for Means) of the participants whose self-reported CS interest remained the same on the Post-Survey (N=57; broken down by racial/ethnic identity and gender identity). . . . .	28
III.10	The results of the t-Test (Paired Two Samples for Means) of the participants whose self-reported CS interest increased on the Post-Survey (broken down by racial/ethnic identity and gender identity). . . . .	28
III.11	The distribution of the participants whose self-reported CS interest remained the same on the Post-Survey (N=57; broken down by racial/ethnic identity and gender identity). . . . .	29
III.12	The distribution of the participants whose self-reported CS interest increased on the Post-Survey (broken down by racial/ethnic identity and gender identity). . . . .	29
IV.1	The main menu of the Tello Interaction Interface. . . . .	32
IV.2	The Shapes Interface of the Tello Interaction Interface. . . . .	33
IV.3	The Circle Interface, where the user designates the radius of the circle. . . . .	33
IV.4	The Square Interface, where the user designates the side length of the square. . . . .	34
IV.5	The Triangle Interface, where the user designates the side length of the triangle. . . . .	34
IV.6	The Cube Interface, where the user designates the side length of the cube. . . . .	34
IV.7	The Camera Interface of the Tello Interaction Interface. . . . .	35
V.1	Quasi-Experimental Study-Specific Design. . . . .	36
V.2	Drone connection as Cyber-Physical System. . . . .	44
V.3	DJI Tello Edu quadcopter used by the students in the course. . . . .	44
V.4	Distribution of programming languages known by participants. . . . .	44
V.5	Graph showing the participants' interest in Computer Science. The mean of the data, on a scale of 0 to 100, is 77.13 with a standard deviation of 18.06. . . . .	46
V.6	Graph showing the participants' interest in programming/coding. The mean of the data, on a scale of 0 to 100, is 77.80 with a standard deviation of 17.42. . . . .	46
V.7	Graph showing the participants' interest in interacting with drones/robots. The mean of the data, on a scale of 0 to 100, is 68.79 with a standard deviation of 20.47. . . . .	46
V.8	Graph showing the participants' interest in interacting with the Nao Robot. The mean of the data, on a scale of 0 to 100, is 68.64 with a standard deviation of 19.67. . . . .	46
V.9	Graph showing the participants' interest in interacting with the iRobot Create. The mean of the data, on a scale of 0 to 100, is 62.63 with a standard deviation of 19.60. . . . .	46
V.10	Graph showing the participants' interest in interacting with the Tello drone. The mean of the data, on a scale of 0 to 100, is 76.41 with a standard deviation of 16.19. . . . .	47

V.11	Graph showing the participants' correctness with respect to the question categories. . . .	48
V.12	Graph showing the participants' interest in interacting with the Nao Robot. The mean of the data, on a scale of 0 to 100, is 69.90 with a standard deviation of 20.20. . . . .	50
V.13	Graph showing the participants' interest in interacting with the iRobot Create. The mean of the data, on a scale of 0 to 100, is 60.57 with a standard deviation of 19.11. . . . .	50
V.14	Graph showing the participants' interest in interacting with the Tello drone. The mean of the data, on a scale of 0 to 100, is 68.17 with a standard deviation of 18.74. . . . .	51
V.15	Graph showing the participants' correctness with respect to the question categories on the post-test. . . . .	51

# CHAPTER I

## Introduction

This work discusses techniques used for recruiting and retaining students into STEM (Science, Technology, Engineering, and Mathematics) majors, more specifically Computer Science; builds upon those techniques to include the student's interest; and surveys the efficacy of the proposed techniques after they have been implemented. Through the integration of drones and programming, it is hypothesized that the retention rate would increase as well as the overall interest in the area. Cyber-Physical Systems (CPS) are the integration of computational power and physical sensors into one system or robot (9; 8), such as drones, autonomous cars, and smart cities. Applications of cyber-physical systems extend to Medicine (47; 27), which could help interest biomedically-inclined students. CPS Education papers have focused on aspects of courses (43; 54; 60; 59), student's learning (37; 65), and robotics (48; 58; 83) (not specifically drones). Research into curriculum development for cyber-physical systems includes undergraduate courses (intro courses to more advanced courses) (85; 21; 66; 13; 35; 46; 45; 81; 84; 51; 20), using hands-on (rarely using robotics, except (12)) teaching approaches (17; 12; 25; 70; 36), and graduate curricula (79).

### I.1 Concept Inventories

One of the main aspects of the pre-and post test is a concept inventory of Python programming, adapted for Python from (86). Previous concept inventories focus on CS1 (63; 32; 19; 18) and CS2 (88), rarely looking at a course designed for first-semester freshmen undergraduate students. This is novel in the fact that it applies the concept inventory to an introductory engineering course, not specifically a Computer Science course.

### I.2 Motivation

As mentioned above, the usage of quad-copter drones in Computer Science Education and Computing Education research is few and far between. Computing Education Research is the study of teaching and learning computing and finding ways to determine the effectiveness of teaching methods (Com). The main goal of Computing Education Research is to improve upon the way computing is taught and received (26). Thus improving the quality and knowledge of the future Electrical Engineering and Computer Science environment and community.

According to (Pay), Electrical Engineering and Computer Science majors are the second highest-paid major, which could be an incentive for persisting through the major. Many papers discuss class difficulty and non-supportive professors as being indicative of attrition (74).



In this work, a curriculum for engaging students with drones is proposed, a pre- and post-test based empirical research study, and an interface through which the students interact with Tello drones. Both the pre- and post-test have a concept inventory aspect, to gauge the students' baselines. The term engagement in this work is defined as either impacting the student's interest in Computer Science or impacting their enrollment status, i.e., declaring a Computer Science major or minor.

In Chapter II, the literature on the related topics of Computing Education Research such as equity: accessibility and disparities (gender and racial); computing in education: using robotics, novice programmers; measure validation; and surveys. The results of the pilot study in an introductory engineering course are discussed in Chapter III. Then, in Chapter V, the redevelopment of the curriculum for said introductory engineering course is detailed. In Chapter IV, The user interface that the students use in the curriculum is described. Finally, Chapter VI, brings the thesis to a summit.

## CHAPTER II

### Literature Survey

The papers examined below are grouped into papers covering general Computing Education research, and papers that examine engagement, retention, and recruitment of students. Overall, the work described in the aforementioned papers inspired the work described in later chapters.

#### II.1 Computing Education Research Overview

In (24), the authors discussed the main questions being looked at by the two branches of Computing Education research. One branch mainly inspects research through an equity lens whilst the other branch focuses on issues with the foundations such as integrating computing into education. The papers at the first Future Directions in Computing Education Summit asked questions in five main areas: Broadening participation, K-12 computing, STEM education with computing, issues with students and their learning, and tools.

In (89), the authors discuss the literature over the last 15 years with respect to surveys in Computer Science education. All the surveys discussed ask questions about the student's demographic information, computing abilities and experiences, computer science perceptions, and overall motivation or interest in Computer Science. The main issues with the surveys discussed are the lack of question similarity, which makes comparing results more difficult, and the lack of validation, which allows for bias and ambiguity in the questions and prohibits study authentication and replication. Out of the 42 surveys looked at, only one survey was validated.

In (52), the authors examined the theoretical constructs used in computing education research published in International Computing Education Research (ICER) conference, ACM Transactions of Computing Education (TOCE), and Computer Science Education (CSEd) between 2005 and 2015. First, they identified the 65 theoretical constructs that were prevalent in the publications. Then, they determined the areas of focus for the theoretical concepts discussed in the papers, finding that the majority focused on learning/understanding and study choice. With respect to qualitative methodologies, they found that almost a third used phenomenography or grounded theory. With respect to quantitative methods, the most common approach was regression. The four main usages of the theoretical concepts are using them to discuss results, using them to predict results, using them to inform teaching and testing the results, and using them for data analyses.

In (62), the authors examine the benefits and costs of using theory heavily in Computing Education Research. Theory usage can both exhibit and inhibit progress in the Computing Education Research field. The main ways in which theory inhibits progress in design are:

- Explanation and design tensions

- the goals of design undermining the goals of explanation and
- the goals of explanation undermining the goals of design
- Domain-specific theory neglect, and
- Peer review publication bias.

To help mitigate the aforementioned inhibitions, the research community could focus on design and be guided by theory, invest more effort in Computing Education Research-specific theories and learning efficacy validation measures, novel design publication (working or not), and peer review support and audits.

## **II.2 Equity**

### **II.2.1 Accessibility**

In (11), the authors discuss the incorporation of accessibility concepts and education into computer science and computing courses. They found that the main techniques for integration of accessibility are employing it as the main topic of a course, incorporating it as a theme in a preexisting course, or having it as an addition in the course. With respect to what information about accessibility is covered in the course, the main approaches are awareness, technical knowledge, empathy, and potential endeavors. The main concepts that are integrated are Universal Design, Accessibility Guidelines, and General Disability Knowledge. The common courses that have accessibility in the syllabus are Human-Computer Interaction and Web Design/Programming, courses that require people to interact with the resulting products. Core classes such as Intro Programming and Data Structures, whilst they lack the direct human interaction factor, should still incorporate accessibility concepts. Pedagogic practices such as in-class activities, projects, and lectures are most common, while research and guest speakers with disabilities are rare. Overall, (11) lists numerous possible future directions such as:

- diverse ways to integrate accessibility,
- incorporating the core topics of accessibility and the learning objects in required Computer Science and Computing courses,
- improve guidelines and testing tools, and
- student learning impacted by accessibility introduction.

In (56), the authors discuss the disparities in data reporting for the K-12 Computing Education research field and provide recommendations and standards to rectify the problem. To determine the most important data reporting standards, the authors analyzed the standards from major organizations such as American

Psychology Association (APA), American Educational Research Association (AERA), What Works Clearinghouse (WWC), and the CONSolidated Standards of Reporting Trials (CONSORT). The main standards they found and focused on are:

- Treatment of data loss (22% reported)
- Evaluation instrument validity and reliability confirmation (39% reported)
- Justification and definition of used statistical analyses (44% reported)
- Sample sizes for all groups and subgroups (69% reported fully)
- Means (73% reported) and standard deviations (50% reported) for all evaluated areas
- Non-effective data (53% reported)
- Effect size, odds ratio, regression coefficient (42% reported), or effect size precision (8% reported)

Overall, the majority of papers did not fully report the standards mentioned above. The authors provide recommendations of how to satisfy the aforementioned standards in Table 2 in their paper.

### **II.2.2 Disparities**

In (49) (written in 2019), the authors acknowledge the lack of diversity in the computing field. They specifically examine the effects of communal values, demographics, and sense of belonging. Female, Asian, Black, Latinx, or First Generation students were found to have higher communal goals than their Male, White, or Continuing Generation counterparts. Also, Female, Asian, or Black students had a lower sense of belonging in computing than their Male, White, or Latinx colleagues. A student's communal goals and perception of computing's possible societal benefits are best for predicting their sense of belonging in computing. Students who have average communal goal affordance in computing perceptions and stronger communal goal orientation are less likely to feel like they belong in computing. Finding a way to showcase or highlight instances where society benefits from computing could help diversify the computing field.

In (42), the authors talked about the factors that can help teachers better predict a student's success in Computer Science. The factors that they believe are indicative of success in an introductory Computer Science course are preparation, tutorial time, experimentation, proficiency and accuracy of programming, mathematical materials approach, and unfamiliar learning territory approach. They explored the impact of gender and race on the aforementioned factors. Only 28% of the test population is female and only 28% is African American. They exclude any other minority groups from the population that is tested. The gain scores were positively correlated to the post-test scores, student's aptitude, and experimentation. They were

negatively correlated to the pre-test scores, time measurements, and accuracy. The course grade was best predicted by the pre- and post-test scores, aptitude, and time measures. It was shown that students with a higher math SAT score finished quicker and spent less time on the tasks than other students. The use of the SAT scores to measure aptitude is unfair, due to the fact that standardized tests are not accurate measures of everyone's cognitive abilities. They expose and are reliant on the social inequalities and emotional states of the students taking them. Not surprisingly, the more previous programming courses the students had taken, the higher their pre- and post-test scores were and their accuracy/proficiency in programming was also higher. With respect to race and gender, African Americans were found to perform the lowest, implying the need for Computer Science preparation class in high school. Women and men performed similar with foci on different programming aspects.

In (76), the authors describe the results of a survey about the Imposter Phenomenon among over 200 students in Computer Science courses. Both undergraduate (65%) and graduate (32.5%) students were surveyed. The gender proportion was 73.4% male and 26.1% female participants. The majority of participants were either Asian (76.3%) or White (8.9%). Overall, they found that more than half of Computer Science students experience Imposter Phenomenon and that females experience Imposter Phenomenon more than their male counterparts. Also, students in Computer Science are affected by Imposter Phenomenon more than students in other fields. Also, can be replicated.

In (67), the authors report on the contributing factors to retaining students in Computer Science. The factors examined are students' perceptions of teaching and their perceptions of cooperation in learning. The studied population was 83% male, 17% female, and represents the academic levels evenly. The surveys were given thrice in the semester. The first survey was given at the beginning of the course, the second around week 8, and the last survey was given the week before finals. They found that Computer Science majors and those considering a Computer Science major were 45 and 22 times more likely to enroll in a Computer Science course than non-majors, respectively. Overall, the instructional practices in individual classes did not predict student retention.

In (71), the authors discuss the benefits of intersectional computing and the analysis of interviews with fourteen Black women in various stages of the computing ecosystem. Intersectional computing is a deeper understanding of experience of groups that are marginalized in computing who also have to deal with racism, sexism, classism, xenophobia, heterosexism, ableism, etc. The women surveyed were professionals early in their careers, graduate students, and tenure-track faculty members. Three of the women have Associate degrees, thirteen have bachelor's degrees, nine went on to get master's degrees, two have certificates in software development, and two obtained doctoral degrees. The majority of those surveyed received bachelor's degrees from a Historically Black College or University (HBCU). Most of them had no prior programming

experience before their degrees. Some of their main experiences are that they are the only Black female student, Black women tend to be excluded, HBCUs prepare their students well, professors at predominant white institutions (PWIs) seemed to not care as much, a sense of community and mentoring at HBCUs, and becoming desensitized to comments as a coping mechanism. Intersectional computing enables and fosters a more collaboration and less competition based environment.

### **II.2.2.1 Gender**

In (14), the authors present data that show the gender gap in science, technology, engineering, and mathematics (STEM) in the form of a U.S. Department of Commerce brief. Women only make up 24% of the STEM workforce and 48% of the general workforce, whereas men comprise 76% of the STEM workforce but only 52% of the overall workforce. Women also constitute 49% of the college educated workforce. Overall, for both genders, the higher the education level attained, the higher the likelihood of working in STEM. A common concept in the public eye is the wage gap. In STEM jobs, there is a 14% gender wage gap, which is lower than the 21% wage gap in Non-STEM jobs. After adjusting the gender wage gap using regression, the STEM wage gap shrinks to 12% overall, with the lowest of 7% being in Engineering jobs. The majority of women (57%) in STEM have physical or life sciences degrees. 48% of men in STEM have engineering degrees instead. Only 26% of women with a STEM degree work in a STEM occupation, in comparison to 40% for men. Women who have both a STEM degree and a STEM occupation make 29% more than women who neither have a STEM degree nor occupation.

In (10), the authors discuss the results of examining 10 years worth of data on students who took Computer Science courses. They looked at the effect of gender on the attrition, withdrawal, and failure rates along with grades. They found that as the course sequence progresses, the attrition rates decrease. Even if a female student passes the course, they are much more likely, than their male counterparts, to not take the next course in the sequence. Interestingly, men are failing more than women, but women are withdrawing more often. They found that women receive lower grades in the classes than men, but overall have similar or higher cumulative GPAs (including non-STEM courses). Both gender and grades were shown to have a large effect on attrition. Should try replicating this study.

In (15), the authors explore the various factors that could explain the absence of women in Computer Science. The main factors that they look at are quantitative ability, goals and interests with respect to education, computer experience, Computer Science stereotypes and knowledge, confidence, personality, support and encouragement, issues with finances and stress, discrimination by gender, and attitudes towards Computer Science academia. The study population was 43% female and 82% white, demonstrating a lack of diversity in the examined population. The participants' quantitative abilities were measured through looking

at their GPAs and ACT scores. Computer Science majors were shown to have scored significantly higher on the science part of the ACT and slightly better overall. No gender differences were found. With respect to educational interests, there was no shown gender difference in Computer Science major interest and everyone saw Computer Science as a worthwhile major. However, Computer Science students were shown to spend more time on school work. Men, overall, had higher educational aspirations. Non-CS majors were more likely to select a career involving interpersonal interaction. Female Computer Science majors were shown to value extrinsic rewards the least, and male non-majors valued them the most. Computer Science majors spent significantly more time using computers than non-majors. Males believed the loner Computer Scientist stereotype and love of number more often than women. Everyone saw Computer Science as a good career for having a family due to flexibility. Interestingly, female Computer Science majors had less confidence in their own computer skills than male non-majors had in their skills. Non-majors were shown to be more open to new experiences than Computer Science majors. Men's gender roles were more "masculine" but male and female gender roles did not differ in femininity. Females were shown to have more interpersonal attachments. Female non-majors were reassured about their competence less than female Computer Science majors. Non-majors were more likely to have financial certainty than Computer Science majors. Men were less likely to perceive gender discrimination. Interestingly, Computer Science majors found the Computer Science environment less positive than non-majors.

#### **II.2.2.2 Racial**

In (38), the authors discuss their preliminary findings of the Glitch Game Testers program. Their two main goals are to enable the viewing of the computation behind the video games the testers play and to create a computing curriculum that is contextualized. The researchers found that all the students would consider taking classes in programming in the future, the testers views of gaming shifted, they felt that they assisted in the development of video games, and that the length of the program appeared to be a deterrent rather than an incentive.

Further expanding on (38), in (39), the authors discuss the results of pre- and post-surveys of students who participated in the Glitch Games Testers program. In the program, young African American men test video games and learn about Computer Science in the process. The part of Computer Science covered by the program are coding, documentation of code bugs, and the process of software development. The results of the survey show that after participating in the Glitch program, members were more likely to see classmates, teammates, and Glitchmates as technical resources. Participants' access to technical expertise was shown to increase significantly. Students who participated showed that they believe their own technical expertise increased due to the program. Prior to Glitch, many students were unlikely to talk about their technical skills,

but Glitch made them feel more comfortable and confident discussing their skills.

### **II.3 Computing in Education**

In (64), the author exhibits the work of Code.org. Code.org has four main foci:

- Hour of Code,
- educational tools,
- professional development, and
- policy and regulation reform.

Hour of Code is a world-wide program for students to spend one hour becoming more familiar with and experience Computer Science. Millions of students of color and girls participated, further addressing and demolishing stereotypes of the Computer Science field. The main educational tools created are a Computer Science curriculum, materials for teaching Computer Science in K-12, and a Computer Science learning platform. The curricula and partnerships were broken down by grade level. Thinkersmith will be used in Elementary education, Bootstrap and Project GUTS in Middle School education, and Exploring CS in High School. Ideally, there will be multiple units for Elementary education, algebra and science integration units and a semester-long Computer Science course in Middle School, and a full-year introduction to Computer Science course and a full-year AP Computer Science Principles course in High School. They offer free professional development for High School and Middle School teachers. They have trained more than 10,000 teachers in teaching Computer Science, mostly Elementary School teachers. Also, they are working with multiple state governments to make Computer Science part of the core curriculum, due to its prevalence in the real world, not just an elective.

In (75), the author illustrates the impact of intersecting poetry and education, specifically Information and Communication Technology (ICT) education. The use of poetry to present and discuss topics related to the education field is impressive and should be expanded. IT allows students to not only make use of their logical sides, but also their creative sides. The author focuses on the information integrity, impact on society, and humorous aspects of utilizing poetry.

*Take your feet off the ground and let your mind wander.*

In (41), the authors explore the three main frames through which K-12 Computer Science education is exhibited. The computational thinking frames are cognitive, situated, and critical. The optimal perspective of computational thinking is a dialogue and integration of all three. Cognitive computational thinking is where students receive an education of the key computational concepts, practices, and perspectives, which



emphasizes skills and competencies for future learning and careers. Situated computational thinking focuses on the process of design and programming digital artifacts that are shareable while developing computational fluency during the creative process. Critical computational thinking is where computational thinking is used with an emphasis on oppressive power structure examination and resistance and creating digital multimedia products to address important societal, moral, and ethical issues. The frames that are expanded to include the basic three's dialogue are computational participation, computational making, and computational action.

In (30), the authors discuss the results of a meta-analysis of published STEM undergraduate education comparing the effects of active learning methods with the traditional lecturing style. Overall, they found that active learning increases performance on examinations and that lecturing causes an increase in failure rates by 55%. The increase in performance and learning holds in all STEM disciplines and all class sizes and course types and levels. Students in classes with fewer than 50 other students benefit the most. Concept inventory performance also increased with active learning.

In (69), the authors explore the current trends of metacognition and self-regulation (together called cognitive control) in the study and research of programming education. Metacognition is generally used to discuss thinking about one's own thinking, whereas self-regulation is more about one's thought process while solving a problem or learning. The three categories of papers they examined are passing (cognitive control mentioned in one sentence), peripheral (used to support the paper), and depth (centered paper on metacognition or self-regulation). 123 of the papers referenced cognitive control in passing, 53 papers were peripheral, and only 31 papers were depth papers. The main four cognitive control theories found to be commonly used both outside of and in the computing education research community are:

- Flavell's model:
  - Defines metacognition as a cognitive activity that regulates any aspect of the cognitive enterprise.
  - Later added self-regulation as a difficult concept for novices.
- Bandura's work:
  - Self-regulation is comprised of self-observation, self-judgment, and self-reflection.
  - Focuses on the influence of cognitive control on behavioral factors from society and the environment.
- Zimmerman's work:
  - Self-regulation as a process with conscious planning, practicing, and progress evaluation phases.
- Pintrich and de Groot's work:

- Focuses on self-regulation’s relationship with motivation.
- Phases:
  - \* Forethought, planning, and activation
  - \* Monitoring
  - \* Control
  - \* Reaction and reflection
- Each phase has cognitive, motivating, behavioral, and contextual components.

Revised Bloom’s Taxonomy and Ertmer & Newby’s model are other theoretical bases used in Computing Education research. Self-efficacy is a commonly studied construct. According to the authors, the main underrated cognitive control theories in Computing Education research are Boekaert’s Adaptable Learning Model, Efklides’ Metacognitive and Affective Model of Self-Regulated Learning, Winne & Hadwin’s model, and Hadwin et al.’s Socially Shared Regulated Learning (SSRL) model. The main measures of cognitive control are the Motivated Strategies for Learning Questionnaire (MSLQ), Classroom Assessment Techniques (CATs), Self-Efficacy Scales, Self-Regulation Questionnaire (SRQ), Student Perceptions of Classroom Knowledge Building (SPOCK), Epistemological Beliefs Questionnaire (EBQ), and Achievement Goals Framework Questionnaire. Commonly, professors use reflective activities as both post-assignment/exam exercises (7 papers) and as activities themselves (8 papers). For visual learners, some educators use visualizations (4 papers) to show student progress. While completing a task, some professors use cognitive control scaffolding approaches. Only two of the depth papers used active learning. Very few papers focused on cognitive control awareness (1 paper). Pre- and post-surveys are common evaluation techniques (5 papers). Six of the papers used standardized tests and analyzed the results of different instruments. Lastly, cognitive control was approached through secondary data analysis by three of the depth papers.

In (73), the authors discuss an introductory Computer Science course, at Georgia Institute of Technology, which is designed to address gender issue concerns about Computer Science and appeal to women. The course was called *Introduction to Media Computation* and focused on incorporating creativity, relevance, collaboration, and restricting registration to only non-computer science and non-engineering majors. The class allowed students to see first hand how multimedia manipulations can be done through coding as opposed to using an existing program. The course still was able to cover traditional Computer Science concepts, but through the application lens. The class was 2/3 female and had a withdrawal, failure, or D (WFD) rate of only 11.5% compared to the traditional Introduction to Computer Science course with a WFD rate of 27.8%. In the course, 60% of female students said that they would likely take an advanced media computation course, but only 10% of the female students said they would continue on to more Computer Science courses. Due

to the makeup of the course, the students interviewed said that they felt ease when asking questions, whilst saying the presence of Computer Science majors would significantly increase their anxiety. Collaboration helped foster success and enjoyment in the course. They also interviewed female students in the traditional Introduction to Computer Science course. Of those interviewed, they expressed disinterest in computing, intimidation, and material irrelevance. The media computation course engages students on both an emotional level and a logical level, which is ideal.

In (72), the authors discuss three methods for teaching Machine Learning and the effects of those methods. The three methods are Facts Condition, Impersonal Condition, and Personal Condition. For Facts Condition, the instruction was the basics of Machine Learning systems and linear regression while leaving out data. The Impersonal Condition incorporated the data of a hypothetical person. The student's own personal data was used in the Personal Condition. The results of their experiments showed that the Personal Condition participants overall paid more attention to Machine Learning mechanisms and were more likely to ground their self-advocacy arguments in Machine Learning mechanisms but did not necessarily have a better ability to apply any Machine Learning mechanism to their lives. Great research!

In (90), the author outlines future research in computing education. The courses that they will be surveying and on which they will be testing their teaching methods are both in-person at Aalto University and online using Amazon's Mechanical Turk platform. They are hoping to develop a framework for teaching programming for both environments that is also scalable.

In (87), the authors describe their results of integrating computational thinking into high school mathematics courses through the utilization of R for mathematically modeling real-world problems. Only 32% of states in the United States have policies enabling Computer Science high school courses. The project, called Computing with R for Mathematical Modeling (CodeR-4MATH), collects learning, assessment, and tutoring resources. The school in which the project was implemented is in a community where the median household income is \$170,000. The participating students are from a Discrete Mathematics course, 28 females, and 14 males. 57% of females intended on majoring in Humanities subjects in college, and none of the students planned to major in Computer Science or Mathematics. 93% of the students have no previous programming or Computer Science experience. The students looked at two questions from the real-world: Meal Plan vs. Pay-as-You-Go and Driving for Gas. They took a post-activity assessment to gauge the impact of the activity on their interest in Computer Science or programming in the future. For the Meal Plan vs. Pay-as-You-Go task, the students were looking for a model that would maximize the number of students it would help in a given period of time and find the best individual solutions. They were encouraged to collaborate with each other on ideas. The Driving for Gas task required the students to execute the given template R model code, interpret the execution results, evaluate and comment on the model, and propose and implement improve-

ments to the model. The exit questionnaire asked for feedback (impressions, thoughts, and suggestions) on the modeling activities and their experience learning in the activities. Only 86% of the students completed the assessment. 33% of the students were either a little more interested or much more interested in computer programming after the activities, whilst 62% of students were either still not interested or even less interested. Some reasons to explain the increased interest are that the students saw coding as a powerful tool, as a way to look at real-life questions with the mathematical skills acquired at school, and as a novel experience for them. Some reasons for the disinterest in programming are that some students see themselves as not good at Computer Science or Mathematics, they were uncomfortable with the hands-off curriculum design, and that they saw programming as boring and not useful. Approximately 78% of students who became less interested or were not interested in programming were female.

In (44), the authors describe the idea of interactive learning and show the results of applying it to software engineering courses. Also, reducing the delay between theory and practice was shown to be helpful towards information retention. Interactive learning is defined as when educators teach a concept then have an exercise on that same topic within minutes of teaching and provided student exercise feedback immediately. The iterative phases are theory, example, exercise, solutions, and reflection. With respect to individual completion, some exercises are multiple choice quizzes, tutorials that have interactive step-by-step instructions, coding challenges that are interactive, and modeling exercises that are also interactive. For completion in teams, projects where the major aspects are communication and collaboration are vital. They used interactive learning in a Software Engineering II: Project Organization and Management class and a Patterns in Software Engineering course. The Project Organization and Management course used quizzes, interactive tutorials, and project work exercises, whereas the Patterns in Software Engineering courses incorporated quizzes, interactive tutorials, interactive coding challenges, and interactive modeling challenges. They examined participation, improved learning, and scalability with respect to interactive learning. For the Project Organization and Management course, they had four main tutorials about agile methods, branch and merge management, continuous integration, and continuous delivery. In the Project Organization and Management course, they found that students felt like interactive learning increased not only their knowledge of the techniques, but also their comfort with applying the techniques to future projects. Interactive learning was shown to be scalable, increase student's participation in classes, and improve their performance on exams.

In (33), the authors discuss a survey administered at three universities that asked students to directly and indirectly assess themselves about moments in their programming processes. There were a total of 214 students who participated in the survey study. The demographics for each population are:

- University 1: 78 students total

- 58% Female
- 6% African American, 35% Asian, 4% Latin American, 42% White, 4% Other, and 8% Two or More Races.
- University 2: 57 students total
  - 32% Female
  - 7% African American, 23% Asian, 10% Latin American, 47% White, 5% Other, and 5% Two or More Races.
- University 3: 79 students total (the most diverse institution)
  - 17% Female
  - 7% African American, 24% Asian, 31% Latin American, 24% White, 8% Other, and 6% Two or More Races.

They both surveyed and performed follow-up interviews. Through those measures, they found that at all the universities, students assessed themselves negatively. Students were also found to comprehend and sympathize with the vignettes, and the higher the rate of self-assessment, the lower the self-efficacy.

In (91), the authors describe the results of a course for first year Electrical Engineering and Computer Science students that is instructed by upperclassmen. They covered topics such as YouTube videos, Machine Learning and YouTube’s recommender systems, Web and Network security, OS interface and applications, building a computer, computer physics, microfabrication, circuits (analog and digital), signals and systems, and Electrical Engineering and Computer Science ethics. Overall, they found that the course satisfied its intended purpose of enabling first year students to make informed educational choices in Electrical Engineering and Computer Science. It also increased interest in less popular Electrical Engineering and Computer Science topics.

### **II.3.1 Using Robotics**

In (31), the authors present the results of a 2-week program for engaging high school freshmen, sophomores, and juniors in Cyber Physical Systems. The project is the Obstacle Avoidance Roombas project, and has four main sub-problems:

- Program a Roomba to follow a set of user-entered coordinates.
- Program a Roomba to acquire and optimally plan a route to the target.

- Program a Roomba to avoid a single obstacle en-route to an acquired target.
- Program a Roomba to avoid multiple obstacles while travelling to a target location.

In the first week, the lectures focus on the necessary mathematics, physics, sensors, programming, and scientific process concepts that the students would need for their projects. The Cyber Physical Systems design phases are conceptual modeling, simulated testing, controlled environment testing, and real-world deployment. For conceptual modeling, the students learned the aforementioned concepts and applied them to create possible algorithms that will be peer reviewed. After peer review, their algorithms for each sub-problem will be validated using computer simulations. They used a common program for simulating robots called Player/Stage package and C/C++ to perform the sub-problems. They also used a test bed called SimVille to perform the controlled environment tests. They did not get to the real-world deployment phase due to time constraints. Overall, the students enjoyed the program. Some students would have preferred a more collaborative environment and utilizing the sensors about which they learned. This program is repeatable.

In (34), the authors discuss the results of using both the Lego Mindstorms robot and a more humanoid robot, Nao, in an introductory programming course for Computer Science non-major freshmen. The two contexts in which the robots were applied are Information Management and Corporate Communications (IMCC) and Information Management Automotive (IMA). IMCC is a female-dominated field, while IMA is male-dominated. The Nao robot is more expensive and less common than the Lego Mindstorms robot, which is more versatile. Each student was given tasks for each robot. The tasks were

- Nao
  - Main Task: Conversation and reading the student's motions
  - Additional Task: Face recognition
- Lego Mindstorms
  - Main Task: Collision avoidance
  - Additional Task: Darkness detection and displaying something on the screen

Overall, they found that the majority of students were more motivated by using Nao, a humanoid robot, than using the Lego Mindstorms robot. Not surprisingly, the IMA students were more motivated by the Lego Mindstorms robot, whilst the IMCC students were more motivated by the Nao. Also, they found that, during the tasks, women had less of a feeling of competence than men.

In (77), the authors explain the setup and results of surveys of students before and after interactions with a robot. They examined the effects of social facilitation on the performance of both male and female students

on both easy and hard arithmetic tasks. For some of the experiences (robot-first), they introduced the robot for Survey 1, the robot administered it, and stayed in the room during the arithmetic tasks but left before Survey 2. For the other experiences (robot-second), the robot was not visible during Survey 1 and the arithmetic tasks, but administered Survey 2. The questions on both surveys included personal questions and robot-related questions. To assess bias in the responses, the last five questions on both surveys were Marlowe-Crowne Scale items. Survey 1 included the 28 remaining Marlowe-Crowne Scale items. With respect to the robot-related questions, when alone, females were pretty negative and males were pretty positive about robots. When the robot was present, females were pretty positive and males were somewhat negative towards robots. Leaving out gender, when the students were alone, they were more honest on the personal items. The students were subject to social pressure. For Survey 2, the results showed that the robot's presence had no effect on the student's honesty on the personal questions. But, female students did view the robot less positively, possibly due to carried over attitudes or responses. In the context of the arithmetic tasks, male students did more poorly when the robot was present than when they were alone. The robot did have a distinctly male voice, which could have had an impact.

### **II.3.2 Novice Programmers**

In (50), the authors discuss reflecting during programming tasks for novice programmers to understand their own processes. Reflecting on one's processes of programming allows for self-regulation of the process better and results in program efficiency. Teaching students to think about their own thinking is a vital skill. However, some of the students in the study struggled with being aware of their own programming practices or of the assignment goals. Some students also found reflection during the process to be distracting. Overall, the self-regulation and reflection seems helpful for understanding how novice programmers become more experienced. The paper discusses many future directions for research.

In (16), the authors discuss the impact of the first five years of Blackbox, a large-scale, continuous, novice programming data project. First, they performed a mapping study of 18 publications that used data from Blackbox. Through that study, they found that most papers covered the topic of errors in code (13 out of 18 papers). The code segments are all Java code extracted from a novice-programmer Java IDE, BlueJ. They also looked at a variety of methodologies used on the data set. Some used the data to construct a model for further exploration, while others replicated previous work. Second, they performed and examined the results of a survey of programming researchers, some of whom were not previously aware of the Blackbox project. The demographics and the corresponding number of survey respondents are:

- Research Area:

- Computing Education (12)
- Software Engineering (7)
- Role:
  - Permanent Academic Staff (11)
  - Postgraduate Student (8)
- Blackbox Relationship:
  - Signed up and used data (9)
  - Signed up but never used data (4)
  - Heard of project but not signed up (6)
  - Never heard of the project before (2)

On the survey, the most important features with respect to the respondent's work are large scale of data set, access to source code, and the ability to see edit history for each user. The most problematic features are the lack of user task information, the need for creating one's own program for data extraction, and the need for one to create their own analysis for the Java code. The main components that would make the data more useful are enabling languages other than Java, user task/assignment information availability, and demographic information. The data is 2.31 terabytes in size and thus not easily downloaded on to one's personal machine. Overall, the data set has been shown to have a reasonably large impact, versatility, large amount of data, and a learning curve.

#### **II.4 Measure Validation**

In (53), the author discusses the implications and principles of the Spatial Encoding Strategy (SpES) theory. The theory implies that honing one's spatial skills helps one develop strategies that are generalizable for nonverbal information encoded into mental representations, such as useful landmarks' identification for orientation. The theory's main components are spatial skills, mental representation encoding, and landmark identification. The theory still needs to have its validity evaluated. Using validated measures to assess each component of the theory would overall validate the theory.

#### **II.5 Surveys**

In (26), the authors discuss the results and impact of a two-stage survey of both computing education researchers and practitioners. The paper defines the goal of Computing Education Research as improving upon



how computing is taught and learned. They apply a modified version of the didactic triangle, which includes researchers and pedagogical content knowledge. The categories and number of questions per category on the survey are:

- Pedagogy (Computing): 41 questions
- Curriculum: 36 questions
- Pipeline: 32 questions
- Languages and Tools: 31 questions
- Pedagogy (General): 25 questions
- Student Understanding: 23 questions
- Inclusivity: 23 questions
- Assessment and Predictors: 21 questions
- Student Behavior: 20 questions
- Teacher Development: 16 questions
- Awareness and Broader Impact: 16 questions

It was interesting to see that the largest number of questions focused on Computing pedagogy. The questions about student access to computers and broadband at home indicates that the population of schools that were surveyed are privileged, if not private. 70% of respondents stated that teaching is their primary activity and that 83% perform formal teaching, whilst only 5% are teaching outside of a formal setting. The questions that were ranked most interesting were in the Student Behavior, Student Understanding, Pedagogy (Computing), and Pedagogy (General) categories. Whilst, the questions seen as least interesting were in the Assessment and Predictors, Awareness and Broader Impact, Curriculum, Inclusivity, Languages and Tools, and Pipeline categories. The Teacher Development category was neither most interesting nor least interesting. Shockingly and dismayingly, inclusivity was seen as uninteresting to practitioners and researchers, even though there is a serious representation gap in the computing community. Researchers and practitioners are interested in the completely different questions about computing education.

## CHAPTER III

### **Evaluation of Student Interest in Computer Science After Early Exposure to Cyber-Physical Systems (CPS)**

Developing interactive and engaging educational experiences and measuring their effectiveness in both learning gains and student retention and engagement is a common goal in Computing Education Research (CER). In this chapter, a case study looking at the pre-and post-survey data from 105 participants is presented. The participants took the surveys about their experience in an introductory Engineering course that acquainted them with Computer Science and specifically Cyber-Physical Systems (quad-copter drones). Students self-reported their programming background, demographics, opinions of the concepts covered, and expectations for the class. Overall, the data demonstrated that the participants were either more interested or had the same level of interest in Computer Science and programming after the experience. This study served as a pilot study for deeper research into the knowledge gains and retention of students in Computer Science after experiencing Computer Science through the lens of Cyber-Physical Systems.

#### **III.1 Introduction**

Trying to understand how best to engage with students is a common question in Computing Education Research, e.g., (24; 26; 92; 28; 57; 68) and vital to recruitment and retention (22; 82; 78; 80). No one will argue that enrollment in Computer Science courses or selection of CS as a major is on the rise. Read any account of the phenomenal growth of the Computer Science major, and you can see that the current trend is far out-pacing that of the dot-com boom of the late 90s (3; 40). Then why are researchers and practitioners still focusing on recruitment and retention? Demand and diversity.

The demand for Computer Science related jobs continues to grow both in the US and around the world (6; 23). And though no one is able to predict the future, it seems clear that this trend will continue. Yet despite the rise in CS enrollments and the seemingly inexhaustible number of job opportunities, the growth in the number of women and underrepresented minorities in Computer Science has not kept pace (7; 5). Among the recommendations made by the National Academies of Sciences, Engineering, and Medicine in their 2018 report, is the urgency for institutions to, “. . . leverage the increasing interest in Computer Science and related fields, among both non-majors and intended majors, to engage, recruit, and retain more women and underrepresented minorities into the field and help address the diversity problem proactively.” (61) To this end, contributing to the ongoing research in these areas is vital.

The main goal of the research discussed in this work is to determine the efficacy of grounding an intro-



Figure III.1: The Nao robot used in (34).



Figure III.2: The Lego Mindstorms used in (34).

ductory Computer Science course within the field of Cyber Physical Systems. In this novel approach, the models used in CPS are leveraged to foster curiosity among the students and more specifically to measure their self-reported increase in their interest in pursuing Computer Science as a major or minor.

Previous work on introducing students to Cyber Physical System often centered on the study of Robotics. They used humanoid robots, such as Nao and Lego Mindstorms, to increase students' engagement. In (77), the authors examined the effects of social facilitation on the performance of both male and female students on easy and hard arithmetic tasks. The research showed that male students were more positive toward the robot than female students, and that both groups were subject to social pressure when the robot was present.

In (34), the authors discuss the results of using both the Lego Mindstorms robot (shown in Fig. III.2) and a more humanoid robot, Nao (shown in Fig. III.1), in an introductory programming course for Computer Science non-major first-year undergraduates. The two contexts in which the robots were one female-dominated field and a male-dominated field. The Nao robot is more expensive and less common than the Lego Mindstorms robot, which is more versatile. Students were asked to program different tasks for each type of robot. The authors found that the majority of students were more motivated by using Nao than using the Lego Mindstorms robot. In the female-dominated field, the students were more motivated by the Nao robot, whilst, in the male-dominated field, the students were more motivated by the Lego Mindstorms robot. Their research also revealed that, during the tasks, women reported having less of a feeling of competence than men.

In (55), researchers found that among students as young as 6 years old, the perception that girls were not as proficient as boys when it came to robotics and programming had already taken hold. Yet an intervention measured against a control group showed that among the students who were given a single 20-minute lesson

in programming and interacting with a pet-like robot, the girls in the group viewed themselves as “good at robots” and that “programming was fun” nearly as often as boys did. Whereas among the 6-year-olds in the control groups, the stereotypes still dominated.

Self-assessments are used in (33) during the programming process, through the administration of a survey at three universities that asked students to directly and indirectly assess themselves about moments in their programming processes. There were a total of 214 students who participated in the survey study. Through those measures, they found that at all the universities, students assessed themselves negatively. Students were also found to comprehend and sympathize with the vignettes, and the higher the rate of self-assessment, the lower the self-efficacy.

In (67), the authors report on the contributing factors to retaining students in Computer Science. The factors examined are students’ perceptions of teaching and their perceptions of cooperation in learning. The studied population was 83 percent male, 17 percent female, and represents the academic levels evenly. The surveys were given thrice in the semester. They found that Computer Science majors and those considering a Computer Science major were 45 and 22 times more likely to enroll in a Computer Science course than non-majors, respectively. Overall, the instructional practices in individual classes did not predict student retention.

These prior studies and many more motivated us to expand this research to other models of Cyber Physical systems, more specifically, quad-copter drones. Would the use of drones with students new to Computer Science have the same effect that robots do, increasing interest and improving self-efficacy in some students? Viewed in the broader purpose of our research, are drones another avenue for recruitment and retention?

The setting for our study was a 4-week course which served as an introduction to the field of Computer Science for first-year Engineering students, which was repeated 4 times (once in Fall 2019 and thrice in Fall 2020). Using quad-copter drones and simple Python programming, students engaged in first-hand experience with Cyber-Physical Systems. The following are examined: the influence of the drone interaction on the students; interest in the field; and attempt to draw some conclusions on the effects to subgroups separated by gender and ethnicity.

In Section 2 of this chapter, the research methods are described, including measurement instruments used. In Section 3, the quantitative results of the work are presented, which included the pre- and post-surveys. Section 4 is a discussion of the results, which also details some threats to the validity of the study. The chapter concludes with findings in Section 5 and further studies proposed in Section 6.

## **III.2 Methods**

The survey questions were designed with the following research question in mind:

- What impact on self-reported interest would interacting with and programming a quad-copter drone have on first-semester undergraduate engineering students?

### **III.2.1 Participants**

The students who participated in the survey were all first-semester first-year undergraduate Engineering majors. There were 101 participants: 69 males, and 32 females. Since the survey was administered at a predominantly white institution (PWI), indicative of a significant population of White/Caucasian students.

### **III.2.2 Surveys**

The surveys mentioned in this section were approved by the institution's Internal Review Board (IRB). Survey participation was voluntary.

#### **III.2.2.1 Questions**

The Background Survey was broken into three subcategories: Programming Background, Student Background, and Class Expectations. The Programming Background questions asked about their previous experiences, declared major, and programming language familiarity. The questions were:

- Do you have any programming or Computer Science experience?
- What programming languages have you used?
- How experienced are you with using and programming computers?
- What major are you planning on declaring?
- Do you have any robotics or embedded systems experience?

The Student Demographics collected were racial/ethnic identity and gender identity. The questions were:

- What is your ethnic background?
- What is your gender identity?

The question about the Class Expectations asked What do you hope to get out of this class?

The student feedback survey asked for feedback about the coverage of Python and the Tello drone, as well as their self-reported interest. Some questions were:

- The coverage of python was \_\_\_\_\_.
- The usage of Tello drones was \_\_\_\_\_.

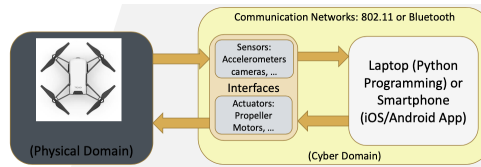


Figure III.3: The drone connection shown as a Cyber-Physical System.

- The Tello drones were introduced \_\_\_\_\_ in the module.
- Compared to the beginning of this course, do you have more, less, or about the same level of interest in Computer Science, now that the module is ending?

### III.2.2.2 Intentions

The intention of the background survey was to ascertain what skills the students have coming into the class. The student feedback survey is to help each module improve based on the feedback from the previous one and to gauge the self-reported change in level of interest, from the beginning of the module compared to the end of the module.

### III.2.3 Course Setup

At this particular institution, the first-year undergraduate Engineering students are required to take three different 4-week modules of Introductory Engineering during their first fall semester. Each of the modules is discipline-specific based on the students' academic interests. The disciplines are: Biomedical Engineering, Chemical Engineering, Civil Engineering, Computer Engineering, Computer Science, Electrical Engineering, and Mechanical Engineering. The students choose the modules they are interested in, fostering a more informed decision around declaring their majors/minors. The main idea behind this course design framework is to allow the students to make informed decisions when choosing their majors.

The Computer Science module surveyed is designed to be a broad survey course of Computer Science principles. In order to cement some principles, the students program in Python to control mobile robots such as the one shown in Figure V.3. The topics covered, at an introductory level, in the course are: Overview of Computer Science; Data Structures; Control Structures; Design and Analysis of Algorithms; Software Engineering; Python Programming; Embedded systems and Cyber-Physical Systems (CPS); Robotics; Feedback Control Systems and Control Theory; Sensors and Actuators; and Networking.

The students start out by learning how to compose computer programs in Python. The students are placed in pairs where at least one student has previous programming knowledge to foster a peer-learning environment. They are then required to complete a small robotics control project together using the Tello



Figure III.4: The DJI Tello Edu quadcopter drone used by the students in the course.

drone, shown in Figure V.2—a cyber-physical system. They use the drone’s various sensors to interact with the world, and utilize its Bluetooth and Wi-Fi capabilities to interact with their laptops. The students send commands to the Tello to control the actuators, such as the propellers and motors. The programs are sent to the Tello drone over Wi-Fi, allowing the students to experience networking and communications between computers and physical objects.

In the example shown in Fig. III.5, the main component is the `droneTakeoffAndLand()` function. In that function, `drone` is an object of type `Tello`. If the function is able, it will execute all the commands in the `try` block. The `drone.subscribe(drone.EVENT_FLIGHT_DATA, handler)` command subscribes to printing out data about the current drone flight. `drone.connect()` initializes the connection with the drone by sending a connection request to the drone. `drone.wait_for_connection(60.0)` makes the program pause until either there is a connection with the drone or the timeout limit is reached, which in this case is 60 seconds. If there is not a timeout exception thrown, the block continues to the `drone.takeoff()` command, which tells the drone to liftoff and start flying. To make the drone’s movements more individual and succinct, the `sleep(5)` command postpones the current thread from being executed for 5 seconds. `drone.down(50)` tells the drone to descend at half speed and sets `self.left_y = -0.5`. The `drone.land()` command tells the drone to land on the ground. These drones also have the ability to land on the palm of a person’s hand. Unless there is an exception thrown, the program should run as mentioned and then utilize the `drone.quit()` function to stop the internal threads that are running.

### III.3 Results

Of the potential 105 participants, 101 students responded to the pre-survey and only 57 students filled out the post-survey. The racial and ethnic backgrounds were only reported by approximately 75 percent of participants on the pre-survey and 98 percent of participants on the post-survey. Many of the participants reported multiple racial and ethnic identities, accounting for the larger number of responses. The ethnic and racial distribution is similar to that of the overall undergraduate population at the institution. Similarly, the

```

1 from time import sleep
2 import tellopy
3
4
5 def handler(event, sender, data, **args):
6     drone = sender
7     if event is drone.EVENT_FLIGHT_DATA:
8         print(data)
9
10
11 def droneTakeoffAndLand():
12     drone = tellopy.Tello()
13     try:
14         drone.subscribe(drone.EVENT_FLIGHT_DATA, handler)
15
16         drone.connect()
17         drone.wait_for_connection(60.0)
18         drone.takeoff()
19         sleep(5)
20         drone.down(50)
21         sleep(5)
22         drone.land()
23         sleep(5)
24     except Exception as ex:
25         print(ex)
26     finally:
27         drone.quit()

```

Figure III.5: An example Python program executed on students' laptops that sends control commands to, and receives sensory and other information from, the Tello drones over 802.11. This particular program (1) connects to the drone, (2) has it take off into the air, (3) has it move downward at half (50%) maximum velocity for approximately 5 seconds, and (4) lands, where commands are sequenced in time using waits of 5 seconds. It additionally concurrently displays sensor information as messages are received through the handler function.

demographics of the pre-survey participants (51.32 percent White/Caucasian, 26.32 percent Asian, and 68.32 percent Male) are congruous with the demographics of the post-survey (59.65 percent White/Caucasian, 24.56 percent Asian, and 68.42 percent Male).

Overall, the results of the survey are indicative that using a cyber-physical system in an introductory course could assist in increasing interest in Computer Science.

### III.3.1 Results of Pre-Survey

The pre-survey had three main sections and the resulting implications are mentioned below. Overall, the pre-survey's response rate was 96 percent.

#### III.3.1.1 Programming Background Results

Most of the students (75.25 percent of participants) reported having previous experience with either programming or Computer Science, whilst only about a quarter did not have any previous experience. As shown in Fig. III.6, the programming languages used most often by students prior to taking this course were Java (61 students) and Python (48 students). In the survey, some participants reported knowing multiple programming languages, which accounts for the numbers summing to greater than the number of participants. Most re-



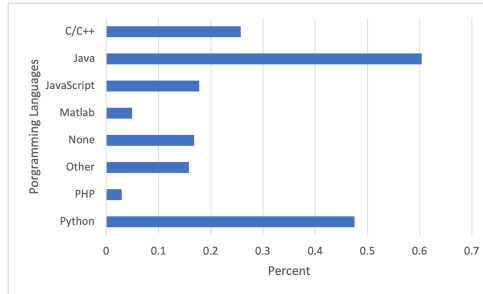


Figure III.6: Distribution of programming languages previously known and used by the participants.

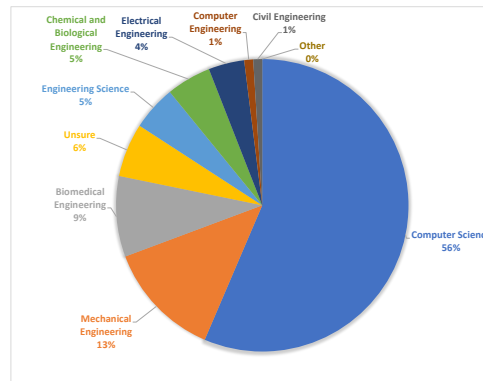


Figure III.7: The distribution of the declared majors of the participants on the Pre-Survey.

spondents reported having some experience (74 students) with programming or using computers, whilst only 27 students reported having no experience.

The top major declared by the students was Computer Science (57 students), as shown in Fig. III.7. Most of the participants (79 students) reported having no previous experience with either robotics or embedded systems, while only 22 participants did have previous experience.

### III.3.1.2 Student Background Results

The racial/ethnic information is shown in aggregate form, according to (29), and broken up by gender identity in Fig. III.8.

As previously mentioned, the demographics of the studied population are consistent with the demographics of the institution. As only 76 participants (approx. 75 percent of the studied population) provided their racial or ethnic identity, the survey showed that a majority of the participants identify as White/Caucasian (39 students) or Asian/Asian-American (20 students). Also, 68 percent of the studied population identify as male and 32 percent identify as female.

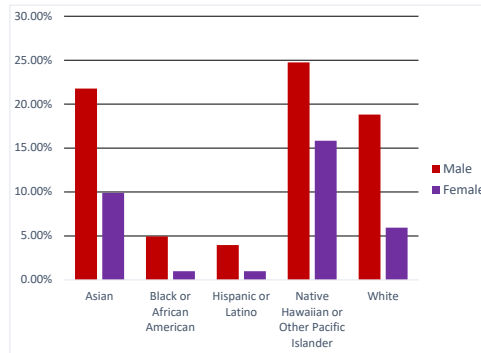


Figure III.8: The distribution of the racial/ethnic identities of the participants as reported on the Pre-Survey (N = 101) broken down by gender.

### III.3.1.3 Class Expectations Results

The variety of responses in this subsection of the survey showed a few main trends:

- Insight into and more experience with programming
- Computer Science with non-traditional hardware real-world applications (drones)
- Knowledge about Python, robotics, and computational logic
- Familiarity with Cyber-Physical Systems
- Cement their interest in Computer Science
- Work on and experience collaborative programming projects
- Machine learning applied to robotics

Overall, the students seemed to be seeking a further understanding of their existing knowledge or the creating and absorbing new information and skills.

### III.3.2 Results of Post-Survey

The post-survey asked about the students' experience in the module and for feedback on what could be changed. The resulting implications are mentioned below. Overall, the post-survey's response rate was 54 percent (57 participants), which is a significant decrease, potentially due to the lack of interest towards the end of the semester or due to the online system/not in-person classes. Other potential contributing factors could be that the pre-survey was required, in order for the professor teaching the course to perform experience-level based pairing, whereas the post-survey was not required but highly encouraged. As specified in the

t-Test: Paired Two Sample for Means		
Same Interest Level		
	Male	Female
Mean	6.73%	1.75%
Variance	0.51%	0.12%
Observations	6	6
Pearson Correlation	0.93361115	
Hypothesized Mean Difference	0	
df	5	
t Stat	2.995856352	
P(T<=t) one-tail	0.01512147	
t Critical one-tail	2.015048373	
P(T<=t) two-tail	0.030242939	
t Critical two-tail	2.570581836	

Figure III.9: The results of the t-Test (Paired Two Samples for Means) of the participants whose self-reported CS interest remained the same on the Post-Survey (N=57; broken down by racial/ethnic identity and gender identity).

t-Test: Paired Two Sample for Means		
Increased Interest Level		
	Male	Female
Mean	3.22%	2.05%
Variance	0.17%	0.08%
Observations	6	6
Pearson Correlation	0.601750559	
Hypothesized Mean Difference	0	
df	5	
t Stat	0.877058019	
P(T<=t) one-tail	0.210295513	
t Critical one-tail	2.015048373	
P(T<=t) two-tail	0.420591027	
t Critical two-tail	2.570581836	

Figure III.10: The results of the t-Test (Paired Two Samples for Means) of the participants whose self-reported CS interest increased on the Post-Survey (broken down by racial/ethnic identity and gender identity).

IRB application, the survey participation was completely voluntary for the students, which could explain the pre-survey response not being 100 percent.

The main major declared by the students on the post-survey was Computer Science, with 31 students self-reporting.

As 56 participants (approx. 98 percent of the post-survey population) provided their racial or ethnic identity, the post-survey showed that a majority of the participants identify as White/Caucasian (34 students) or Asian/Asian-American (14 students). Similarly to the pre-survey, 68 percent of the post-survey population identify as male and 32 percent identify as female.

With respect to Python, most students (68 percent surveyed) thought that the module covered it well. Similarly, a majority of the respondents (67 percent) thought that the usage of the Tello drones was sufficient. Most of the students (75 percent) were satisfied with the introduction of the drones in the course calendar.

Of the 40 students that responded (70 percent response rate) to the question evaluating the change in their interest level in Computer Science, a slight majority of students (55 percent) said that they had about the same level of interest after the module and the other students (45 percent) expressed more interest in Computer Science.

In the tables displayed in Figures III.9 and III.10, the results of t-Tests ( $\alpha = 0.05$ ) comparing the male participants to the female participants with respect to their self-reported change in level of interest are shown.

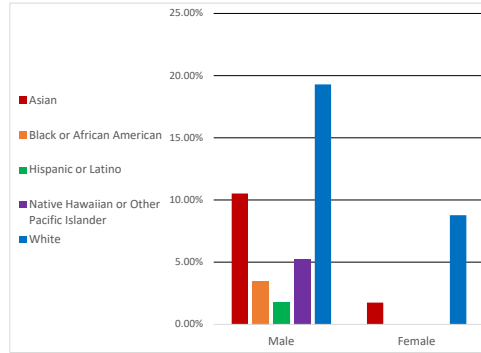


Figure III.11: The distribution of the participants whose self-reported CS interest remained the same on the Post-Survey (N=57; broken down by racial/ethnic identity and gender identity).

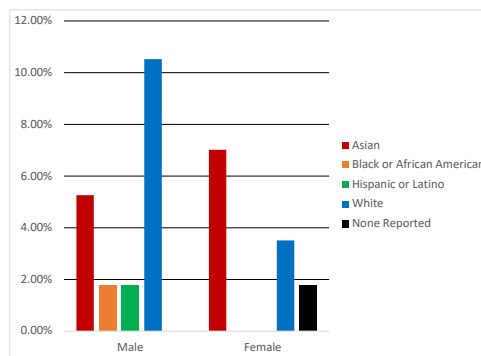


Figure III.12: The distribution of the participants whose self-reported CS interest increased on the Post-Survey (broken down by racial/ethnic identity and gender identity).

For those participants with the same level of self-reported interest in Computer Science, the mean percent of the sample population that is male was 6.73 percent and that is female was 1.75 percent, with variances of 0.51 percent and 0.12 percent, respectively. For those participants with an increased level of self-reported interest in Computer Science, the mean sample population percentage that is male was 3.22 percent and that is female was 2.05 percent, with variances of 0.17 percent and 0.08 percent, respectively. In this sample, there were 5 degrees of freedom. The null hypothesis is that gender is indicative of interest in Computer Science. The tests did not consider racial/ethnic identity. The p-value for the same level of self-reported interest is  $p = 0.015$ , which indicates that the null hypothesis should be rejected, hence gender is not indicative of self-reported interest in Computer Science for those whose interest remained the same. The p-value for the increased level of self-reported interest is  $p = 0.21$ , which indicates that the null hypothesis should not be rejected, hence gender is potentially indicative of self-reported interest in Computer Science for those whose interest level increased. The aforementioned statistical values could be impacted by the small sample sizes. The data that was analyzed was the self-reported change in Computer Science interest from the beginning of the module to the end. It was broken down by racial/ethnic identity and gender identity.

The breakdown of the students, by demographic data, who indicated stationary interest post-CPS interaction is shown in Fig. III.11, whilst Fig. III.12 shows the breakdown of the students who indicated increased interest. Interestingly, Asian Females were the most inspired females by the hands-on Cyber-Physical Systems experience.

### **III.4 Discussion**

For the previous coding experience, the majority of students have used the open-source/free-to-use programming languages, e.g., Python, Java, C/C++. Thus, the lack of MATLAB experience could be attributed to the lack of availability of licenses. For the students who had never programmed before, there was more of a learning curve in the course, and they generally were paired with a student who did have previous programming experience, as detailed in Section III.2.3. The multitude of Computer Science majors in the modules corresponds to the fact that the module is Computer Science focused and therefore attractive to intended majors.

After interacting with a Cyber-Physical System and learning more about programming, the students either had the same level of interest or an increased level of interest in Computer Science. No student surveyed reported a decreased interest. The interaction provided evidence indicating that the likely answer to the main research question is a positive impact, i.e., increased interest in Computer Science among some students. Further work needs to be done to confirm that indication.

Due to the limited scope and time constraints of the modules, covering topics such as other programming languages and machine learning would not have been feasible. Additionally, at the institution, the students would be able to learn about those topics in upper-level Computer Science courses.

#### **III.4.1 Threats to Validity**

As this survey was administered only at a PWI, the population studied is not representative of the overall population and therefore would need to be replicated at a more diverse institution to gather more data and be a better representation of the effectiveness. Also, the population included only cisgender individuals, thus not representing transgender and non-binary individuals. The participants did not have identification numbers, thus tracking the populations between the pre- and post-surveys was not possible.

Additionally, the survey has only been administered to 4 sections of the class module for a total of 105 students. Also, some students did have an advantage in having previous programming knowledge while others did not. The various familiarity levels with programming, robotics, and Computer Science, in general, could have an impact on the results.

### **III.5 Conclusions**

After surveying undergraduate students in an Introduction to Engineering course focusing on Computer Science and Cyber-Physical Systems (CPS), it can be shown that their interest in Computer Science either increased or remained the same. The aforementioned conclusion can be used to address the motivating research question. In the course, the students used Python to program a quad-copter drone as a method of engagement. The course allowed the students to experience and interactive with a physical application of Computer Science.

### **III.6 Future Work**

In future work, the threats to validity, mentioned previously in this chapter, will be addressed by gathering data from other institutions that are also using quad-copters as a means by which to introduce Engineering students to the study of Computer Science. Though this would introduce more variables into our experiments, collecting qualitative data from different sources would be a step towards building a stronger hypothesis to then be tested. The research to measure the effect our chosen model (the quad-copter) had on self-reported interest in Computer Science, as part of a larger effort to quantify what it is about the chosen model (Lego robot, quad-copter, pencil and paper, etc.) that inspires a students' curiosity, particularly those in underrepresented groups? Or is it simply the environment in which these models are introduced that is the ultimate cause for the increase observed? As goals of diversity and levels of demand continue, I would like to contribute to an overarching body of work that seeks to distill these experiments to a general model for introductory tools and environments that are proven effective in recruitment and retention of Computer Science enthusiasts.

## CHAPTER IV

### Tello Interaction Interface

The interface that the students will be mainly interacting with is the Tello Interface shown in Figure IV.1. As shown in Figure IV.1, the graphical user interface (GUI) has five buttons on the main menu. The takeoff and land buttons make the drone do what their names indicate. The Shapes button guides the user to a menu for selecting the shape they would like to draw. The options are a Circle, Triangle, Square, and Cube. The Camera button allows the user to either take a picture or record a video. Overall, the students will be given skeleton code for the interface to fill and make functional.

#### IV.1 Takeoff and Land

The takeoff and land buttons, shown in Figure IV.1, send the takeoff and land commands, respectively. Communicating with the drone is done using UDP addresses and the Tello SDK (4).

#### IV.2 Shapes Interface

The shapes button, shown in Figure IV.1, loads the Shapes Interface window, shown in Figure IV.2, allowing the user to draw a triangle, square, circle, or cube with the drone.

##### IV.2.1 Circle Interface

The Circle Interface, shown in Figure IV.3, allows the user to designate the radius of the circle that they want the quadcopter to fly.

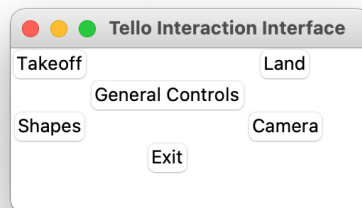


Figure IV.1: The main menu of the Tello Interaction Interface.

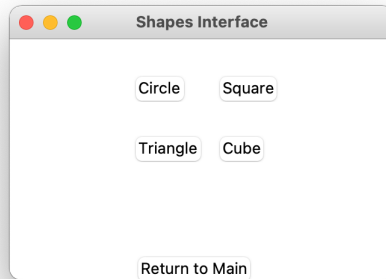


Figure IV.2: The Shapes Interface of the Tello Interaction Interface.

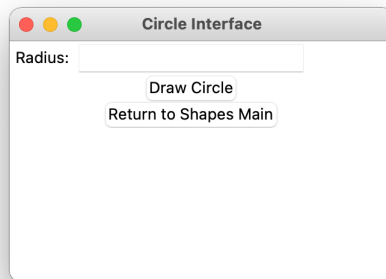


Figure IV.3: The Circle Interface, where the user designates the radius of the circle.

#### **IV.2.2 Square Interface**

The Square Interface, shown in Figure IV.4, allows the user to designate the length of the sides of the square that they want the quadcopter to fly.

#### **IV.2.3 Triangle Interface**

The Triangle Interface, shown in Figure IV.5, allows the user to designate the length of the sides of the triangle that they want the quadcopter to fly.

#### **IV.2.4 Cube Interface**

The Cube Interface, shown in Figure IV.6, allows the user to designate the length of the sides of the cube that they want the quadcopter to fly.



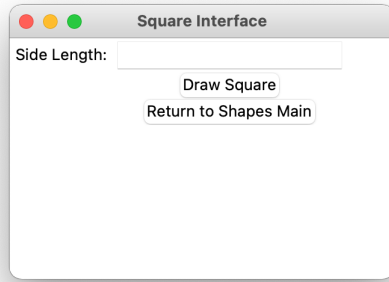


Figure IV.4: The Square Interface, where the user designates the side length of the square.

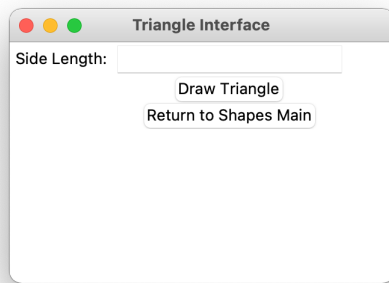


Figure IV.5: The Triangle Interface, where the user designates the side length of the triangle.

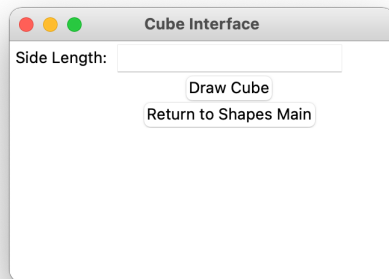


Figure IV.6: The Cube Interface, where the user designates the side length of the cube.

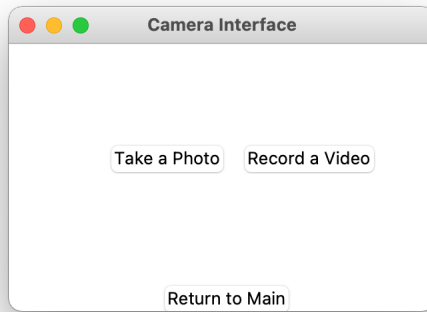


Figure IV.7: The Camera Interface of the Tello Interaction Interface.

### **IV.3 Camera Interface**

The Camera Interface, shown in Figure IV.7, allows the user to either take a photo with the camera on the quadcopter or record a video while in flight.

## CHAPTER V

### ES 140X Curriculum Redevelopment Results

*The main purpose of this research is to determine the effect of a new curricular design in an introductory engineering course on Python programming knowledge and enrollment in future Computer Science courses.*

#### V.1 Research Questions

1. Will students improve their knowledge of Python programming after a 4-week module using Python in conjunction with quad-copter drones?
2. Are students more likely to enroll in CS1/CS2 after a 4-week module using Python in conjunction with Cyber-Physical Systems?

For both of the questions above, I will examine the following participant characteristics:

- Gender identity (Male, Female, Non-Binary)
- Ethnic identity (Underrepresented, Well-represented)
- Enrollment status (Computer Science Major or Computer Science Minor or Non-Computer Science Major)

#### V.2 Study Design

The design would be a quasi-experiment with a single group pre- and post-test, with the setup illustrated in Figure V.1.

##### V.2.1 Data

The data collected is broken into dependent and independent variables.



Figure V.1: Quasi-Experimental Study-Specific Design.

### **V.2.1.1 Dependent Variables**

The dependent variables in the study are the participants' knowledge of Python programming and likelihood of enrolling in future Computer Science courses (e.g., CS1/CS2). To examine the participants' knowledge of Python programming, I will test it using a concept inventory pre- and post- intervention, ideally, using a Python programming or Tello validated concept inventory. To indicate the future Computer Science course enrollment, the participants will self-report intentions on the pre- and post-test, and I will look at enrollment data. The potential question on the pre- and post-test is "Are you planning on enrolling in a Computer Science course next semester?" Possible responses to the question are similar to a 7-point Likert scale: Definitely, Very Probably, Probably, Unsure, Possibly, Probably Not, Definitely Not. When examining the enrollment data, I will see whether the participants enrolled in a Computer Science course the next semester or even considered it (put in their cart but did not enroll).

### **V.2.1.2 Independent Variables**

The independent variables in the study are the participants' gender identity (Female, Male, Transgender, Non-binary/non-conforming, Prefer not to respond), ethnic identity (Underrepresented, Well-represented), and enrollment status (Computer Science Major, Computer Science Minor, or Non-Computer Science Major). To acquire the gender identity of the participants, they will self-report on the pre- and post-test with the possible selections of Female, Male, Transgender, Non-binary/non-conforming, or Prefer not to respond. To ascertain the ethnic identity of the participants, they will self-report on the pre- and post-test. The question on the pre- and post-test will be "What is your ethnic or racial identity?" The possible selections of Asian or Pacific Islander; Black or African American; Hispanic or Latinx; Native American or Alaskan Native; White or Caucasian; Multiracial or Biracial (with place to specify); or A race/ethnicity not listed here (with place to specify). To gather the enrollment status of the participants, they will self-report their major on the pre- and post-test, and I will look at the enrollment data to see if the participants are Computer Science majors or minors already. The question on the tests will be "What major(s) are you planning on declaring?" The possible selections will be Biomedical Engineering, Chemical and Biological Engineering, Civil Engineering, Computer Engineering, Computer Science, Electrical Engineering, Engineering Science, Mechanical Engineering, Undeclared, and Other (fill in the blank). To determine their minor status, they will self-report on the pre- and post-test with the possible selections of Computer Engineering, Computer Science, Data Science, Electrical Engineering, Energy and Environmental Systems, Engineering Management, Environmental Engineering, Materials Science and Engineering, Nanoscience and Nanotechnology, Scientific Computing, None, or Other (fill in the blank).

### **V.2.1.3 Data Collection**

The pre-test will be administered during the first class meeting of ES 1401, ES 1402, and ES 1403 on dates August 25, 2021, September 27, 2021, and November 1, 2021, respectively. The post-test will be administered during the last class meeting of ES 1401, ES 1402, and ES 1403 on dates September 24, 2021, October 27, 2021, and December 10, 2021, respectively. There are approximately 30 participants per module (approx. 90 participants in total). The study population are first-semester Freshman students in an introductory engineering course, specifically the Computer Science-focused module. The population will be broken down by gender identity, ethnic/racial identity, and enrollment status.

## **V.3 Preliminary Results**

After running the pilot study in Fall 2019 and 2020 and participating in the DEERS workshop, the pre- and post-tests were revamped for running the study in Fall 2021.

### **V.3.1 ES 140X Study**

Thus far, I have examined the impact on interest in Computer Science of using quad-copter drones in combination with Python programming in an introductory engineering course. In that study, the effects on subgroups based on gender and ethnicity are also examined.

First-semester freshmen students were surveyed. The survey questions were designed with the research questions discussed in Section V.1 in mind.

The students who have thus far participated in the survey are all first-semester first-year undergraduate Engineering students. Of the 56 respondents thus far, one did not want their responses to be used for research, four were under 18 years old, and three responses were incomplete. Thus, there are 48 participants so far: 28 males and 20 females.

#### **V.3.1.1 Questions**

In order to be able to track the responses across the pre- and post-tests, I ask the respondents for their Commodore ID as I am unable to look it up, therefore unable to identify the students. The Pre-test starts with a question asking if the participant's responses can be used for research, and their response is either Yes or No. The pre-Test was broken into four main subcategories: Programming Background, Participant Background, Class Expectations, and Concept Inventory. The Programming Background questions are:

- Do you have any programming or Computer Science experience? (Yes or No)
- If any, what programming languages have you used?

- Python
  - Java
  - MATLAB
  - C/C++
  - JavaScript
  - PHP
  - Scratch (or other block-based languages)
  - Other (please specify)
  - None
- How experienced are you with using computers? (Sliding scale from Not at All to Extremely)
  - How experienced are you with programming computers? (Sliding scale from Not at All to Extremely)
  - What major(s) are you planning on declaring?
    - Biomedical Engineering
    - Chemical and Biological Engineering
    - Civil Engineering
    - Computer Engineering
    - Computer Science
    - Electrical Engineering
    - Engineering Science
    - Mechanical Engineering
    - Undeclared
    - Other (please specify)
  - What minor(s) are you planning on declaring?
    - Computer Engineering
    - Computer Science
    - Data Science
    - Electrical Engineering

- Energy and Environmental Systems
  - Engineering Management
  - Environmental Engineering
  - Materials Science and Engineering
  - Nanoscience and Nanotechnology
  - Scientific Computing
  - None
  - Other (please specify)
- Do you have any robotics or embedded systems experience? (Yes or No)
  - How interested are you in Computer Science? (Sliding scale from Not at All to Extremely)
  - How interested are you in programming/coding? (Sliding scale from Not at All to Extremely)
  - How interested are you in interacting with drones/robots? (Sliding scale from Not at All to Extremely)
  - How interested are you in interacting with the Nao robot shown below? (Sliding scale from Not at All to Extremely)
  - How interested are you in interacting with the iRobot Create robot shown below? (Sliding scale from Not at All to Extremely)
  - How interested are you in interacting with the Tello drone shown below? (Sliding scale from Not at All to Extremely)

The Participant Background questions are:

- What is your racial or ethnic identity?
  - Asian or Pacific Islander
  - Black or African American
  - Hispanic or Latinx
  - Native American or Alaskan Native
  - White or Caucasian
  - Multiracial or Biracial (please specify)
  - A race/ethnicity not listed here (please specify)

- Prefer not to respond
- What is your gender identity?
  - Female
  - Male
  - Transgender
  - Non-binary/Non-conforming
  - Prefer not to respond
- Are you 18 years old or older? (Yes or No)
- Is there anything else you would like us to know about you? (Free Response)

The questions about the Class Expectations are:

- What do you hope to get out of this class? (Free Response)
- How likely are you to enroll in a Computer Science course next semester? (Sliding scale from Definitely Not to Definitely)
- How likely are you to major in Computer Science? (Sliding scale from Definitely Not to Definitely)
- How likely are you to minor in Computer Science? (Sliding scale from Definitely Not to Definitely)

The Concept Inventory questions asked about Python topics including variables, conditionals, loops/iteration, and a basic understanding of algorithms.

The Post-Test asked the Programming and Participant Background questions from the Pre-Test along with the Concept Inventory. Additionally, it asked for feedback about the concept coverage in the course and the efficacy of the intervention. The concept coverage questions are:

- Prior to this module, how interested were you in Computer Science? (5-point Likert scale)
- Prior to this module, how interested were you in programming/coding? (5-point Likert scale)
- Prior to this module, how interested were you in interacting with drones/robots? (5-point Likert scale)
- Now, after completing the module, how interested are you in Computer Science? (5-point Likert scale)
- Now, after completing the module, how interested are you in programming/coding? (5-point Likert scale)



- Now, after completing the module, how interested are you in interacting with drones/robots? (5-point Likert scale)
- How interested are you in interacting with the Nao robot shown below? (Sliding scale from Not at All to Extremely)
- How interested are you in interacting with the iRobot Create robot shown below? (Sliding scale from Not at All to Extremely)
- How interested are you in interacting with the Tello drone shown below? (Sliding scale from Not at All to Extremely)
- What did you like about the material covered in the module? (Free Response)
- What would you change about any aspect of the module? (Free Response)
- Is there any other feedback you would like to share with us? (Free Response)

The efficacy of the intervention questions are:

- The coverage of Python was \_\_\_\_\_. (5-point Likert scale)
- My experience with Python programming was \_\_\_\_\_. (5-point Likert scale)
- The coverage of the user interface was \_\_\_\_\_. (5-point Likert scale)
- My experience with the user interface was \_\_\_\_\_. (5-point Likert scale)
- My experience with the Tello drones was \_\_\_\_\_. (5-point Likert scale)
- The coverage of the Tello drones was \_\_\_\_\_. (5-point Likert scale)
- How likely are you to enroll in a Computer Science course next semester? (7-point Likert scale)
- How likely are you to major in Computer Science? (7-point Likert scale)
- How likely are you to minor in Computer Science? (7-point Likert scale)

### **V.3.1.2 Intentions**

The intention of the pre-test was to ascertain an idea of with what skills the students come into the class. The post test is to help each module improve based on the feedback from the previous one.

At this particular institution, the freshmen Engineering students are required to take three different modules of a course throughout their first fall semester. Each of the modules is discipline-specific based on the

student's academic interest. The disciplines are Biomedical Engineering, Chemical Engineering, Civil Engineering, Computer Engineering, Computer Science, Electrical Engineering, and Mechanical Engineering. The main idea behind this course design framework is to allow the students to make educated decisions when choosing their majors.

The Computer Science module that was surveyed has been designed to be a broad coverage course of Computer Science principles. In order to cement some principles, the students program in Python to control mobile robots such as the one shown in Figure V.3. The topics covered, at an introductory level, in the course are:

- Overview of Computer Science
- Data Structures (Arrays, Lists, Classes, etc.)
- Control Structures:
  - Conditionals (*if* statements)
  - Iteration (*for* and *while* loops)
  - Functions
- Design and Analysis of Algorithms
- Software Engineering
- Programming in Python
- Embedded systems and Cyber-Physical Systems (CPS)
- Robotics
- Feedback Control Systems and Control Theory
- Utilizing Sensors and Actuators
- Networking

For the modules, the students start out by learning how to compose computer programs in Python. The students are required to complete a group assignment where they are randomly placed in pairs without previous programming knowledge in mind to foster a peer-learning environment. As shown in Figure V.2, the Tello drone, that the students interact with, is a cyber-physical system. It uses its various sensors to interact with the world and employs its Bluetooth and Wi-Fi capabilities to interact with the student's laptop or

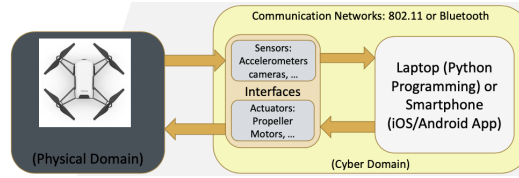


Figure V.2: Drone connection as Cyber-Physical System.



Figure V.3: DJI Tello Edu quadcopter used by the students in the course.

smartphone. Using either a Python program on the student’s computer or the iOS or Android application, the students send commands to the Tello to control the actuators such as the propellers and motors. Since the programs are sent to the Tello drone over Wi-Fi, the students are learning a little about networking and communications between computers and physical objects.

### V.3.2 Pre-Test Results

The pre-test had four main sections and the resulting implications are mentioned below. The pre-test’s response rate was 92.11%. Most of the participants (85.71% of participants) reported having previous experience with either programming or Computer Science. The distribution of the programming languages is shown in Table V.1 and Figure V.4. The top 5 responses for the programming languages that most participants used

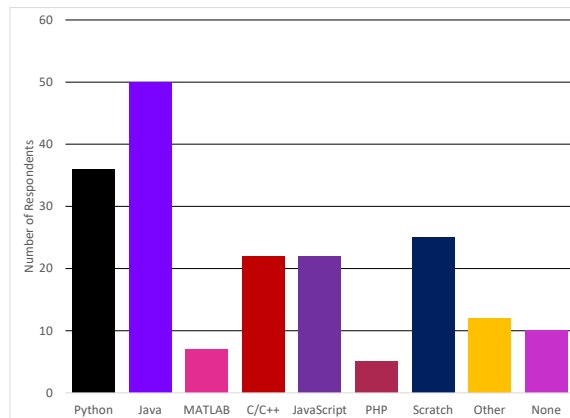


Figure V.4: Distribution of programming languages known by participants.

<b>Programming Language</b>	<b>Num. participants</b>	<b>Percent (%)</b>
Python	36	51.43
Java	50	71.43
MATLAB	7	10.00
C/C++	22	31.43
JavaScript	22	31.43
PHP	5	7.14
Scratch	25	35.71
Other	12	17.14
None	10	14.29

Table V.1: Distribution of programming languages previously known and used by the participants.

<b>Undergraduate Major</b>	<b>Participants</b>	<b>Percent (%)</b>
Biomedical Engineering	8	11.43
Chemical and Biological Engineering	3	4.29
Civil Engineering	2	2.86
Computer Engineering	6	8.57
Computer Science	43	61.43
Electrical Engineering	4	5.71
Engineering Science	4	5.71
Mechanical Engineering	8	11.43
Other	6	8.57
Undeclared	1	1.43

Table V.2: Distribution of majors declared by the participants on the pre-test.

previously were Java (50 participants), Python (36), Scratch or Other Block-Based programming languages (25 participants), JavaScript (22 participants), and C/C++ (22 participants). Some participants know multiple programming languages, which accounts for the numbers summing to greater than the number of participants.

On a scale from 0 to 100, the mean value of experience with using computers was reported as 66.40 whilst the mean value of experience with programming computers was reported as 44.07. That result is not surprising, as most people have experience using computers, but not programming them.

The major distribution of the participants in the pre-test is shown in Table V.2.

The main major declared by the participants was Computer Science (43 participants). Most of the participants (51 participants) reported having no previous experience with either robotics or embedded systems, while only 19 participants did have previous experience.

Figures V.5, V.6, and V.7 show the participant's interest data from the pre-test about Computer Science, programming, and drones, respectively.

As seen in Figures V.8, V.9, and V.10, students were most interested in interacting with the Tello drone, of the three shown. The mean interest in coding/programming was 77.80, whilst the mean interest in interacting with drones/robots was only 68.79.

The racial and ethnic information is shown in Table V.3. As previously mentioned, the demographics

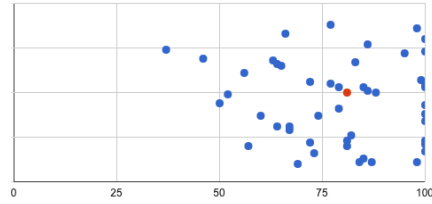


Figure V.5: Graph showing the participants' interest in Computer Science. The mean of the data, on a scale of 0 to 100, is 77.13 with a standard deviation of 18.06.

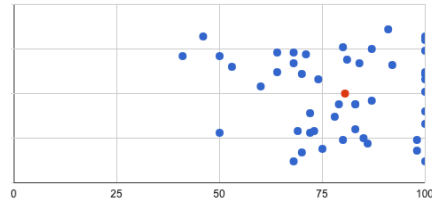


Figure V.6: Graph showing the participants' interest in programming/coding. The mean of the data, on a scale of 0 to 100, is 77.80 with a standard deviation of 17.42.

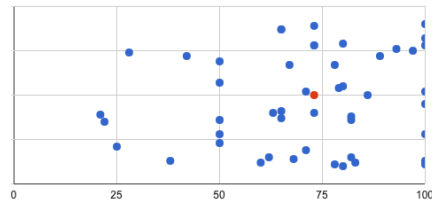


Figure V.7: Graph showing the participants' interest in interacting with drones/robots. The mean of the data, on a scale of 0 to 100, is 68.79 with a standard deviation of 20.47.

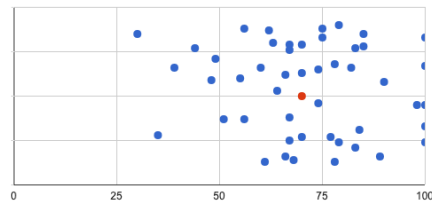


Figure V.8: Graph showing the participants' interest in interacting with the Nao Robot. The mean of the data, on a scale of 0 to 100, is 68.64 with a standard deviation of 19.67.

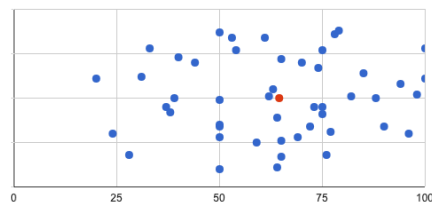


Figure V.9: Graph showing the participants' interest in interacting with the iRobot Create. The mean of the data, on a scale of 0 to 100, is 62.63 with a standard deviation of 19.60.

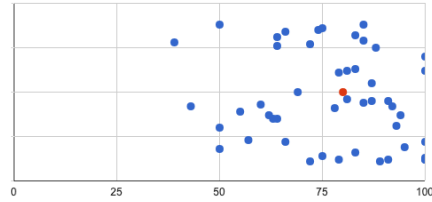


Figure V.10: Graph showing the participants’ interest in interacting with the Tello drone. The mean of the data, on a scale of 0 to 100, is 76.41 with a standard deviation of 16.19.

Identity	Num. participants	Percent (%)
Asian or Pacific Islander	20	28.57
Black or African American	9	12.86
Hispanic or Latinx	7	10.00
Native American or Alaskan Native	0	0.00
White or Caucasian	26	37.14
Multiracial or Biracial	6	8.57
A race/ethnicity not listed here	1	1.43
Prefer not to respond	1	1.43

Table V.3: Student Race and Ethnic Identity as reported on the Pre-Test (N = 70)

of the studied population are consistent with the demographics of the institution. As only 69 participants (approx. 98.57% of the studied population) provided their racial or ethnic identity, the test showed that a majority of the participants identify as Asian or Pacific Islander (202 participants) or White or Caucasian (26 participants). Also, 60.00% of the studied population identify as male and 40.00% identify as female.

The variety of responses in this subsection of the pre-test showed a few main trends:

- Exploring programming
- Computer Science general knowledge/applications
- Programming/Interacting with the drones
- Meet new people/peers

Identity	Male	Female	Male (%)	Female (%)
Asian or Pacific Islander	12	8	17.14	11.43
Black or African American	8	1	11.43	1.43
Hispanic or Latinx	5	2	7.14	2.86
White or Caucasian	12	14	17.14	20.00
Multiracial or Biracial (please specify)	4	2	5.71	2.86
A race/ethnicity not listed here (please specify)	1	0	1.43	0.00
Prefer not to respond	0	1	0.00	1.43

Table V.4: Student Race and Ethnic Identity as reported on the Pre-Test (N = 70) broken down by gender.

Question Category	Percent (%) Correct
Variables	92.14
Conditionals	88.57
Loops and Iteration	84.29
Algorithms	77.14

Table V.5: Question Correctness (N = 70)

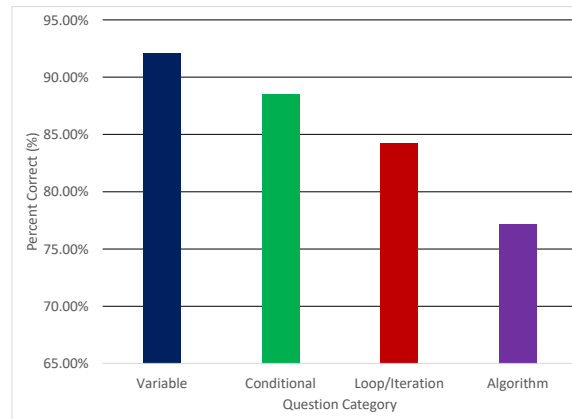


Figure V.11: Graph showing the participants' correctness with respect to the question categories.

- Cement their interest in majoring in Computer Science
- Work on control systems and with robotics

Overall, the participants seemed to be seeking a new experience in Computer Science and drive to major in Computer Science.

The likelihood of a participant taking a Computer Science course next semester had a mean of 86.74, on a scale of 0 to 100, and a standard deviation of 20.31. The likelihood of a participant majoring in Computer Science had a mean of 69.77 and a standard deviation of 29.16. The likelihood of a participant minoring in Computer Science had a mean of 45.83 and a standard deviation of 27.14, which makes sense due to the large population of Computer Science majors in the module.

For the concept inventory, the questions were broken into four categories: algorithms, variables, loops/iteration, and conditionals. As shown in Figures V.11 and V.5, most of the participants responded correctly to the basic concept inventory, which is not surprising since all respondents said they had previous experience with programming or Computer Science. Participants performed least well on the algorithmic analysis question.

<b>Undergraduate Major</b>	<b>Participants</b>	<b>Percent (%)</b>
Biomedical Engineering	7	16.67
Chemical and Biological Engineering	1	2.38
Civil Engineering	1	2.38
Computer Engineering	1	2.38
Computer Science	25	59.52
Electrical Engineering	1	2.38
Engineering Science	2	4.76
Mechanical Engineering	4	9.52
Other	6	14.29
Undeclared	1	2.38

Table V.6: Distribution of majors declared by the participants on the post-test.

<b>Identity</b>	<b>Num. participants</b>	<b>Percent (%)</b>
Asian or Pacific Islander	14	33.33
Black or African American	7	16.67
Hispanic or Latinx	4	9.52
Native American or Alaskan Native	0	0.00
White or Caucasian	13	30.95
Multiracial or Biracial	3	7.14
A race/ethnicity not listed here	0	0.00
Prefer not to respond	1	2.38

Table V.7: Student Race and Ethnic Identity as reported on the Post-Test (N = 42)

### V.3.3 Post-Test Results

The post-test had five main sections and the resulting implications are mentioned below. Thus far, the post-test's response rate was 82.35%.

The major distribution of the participants in the post-test is shown in Table V.6. The main major declared by the participants was Computer Science (25 participants).

The racial and ethnic information is shown in Table V.7. As previously mentioned, the demographics of the studied population are consistent with the demographics of the institution. As only 21 participants (approx. 95% of the studied population) provided their racial or ethnic identity, the test showed that a majority of the participants identify as Asian or Pacific Islander (9 participants) or White or Caucasian (5 participants). Also, 63.64% of the studied population identify as male and 36.36% identify as female.

The self-reported interest in Computer Science prior to the module had an average of 71.43 with a standard deviation of 13.95, while the self-reported interest in Computer Science at the end of the module had an average of 79.17 with a standard deviation of 18.43. Also, the self-reported interest in programming/coding prior to the module had an average of 73.21 with a standard deviation of 15.28, in comparison to an average of 80.95 and standard deviation of 19.36 at the end of the module. The self-reported interest in drone/robot interaction prior to the module had an average of 59.52 with a standard deviation of 9.71, compared to an end



Identity	Male	Female	Male (%)	Female (%)
Asian or Pacific Islander	8	6	19.05	14.29
Black or African American	6	1	14.29	2.38
Hispanic or Latinx	2	2	4.76	4.76
Native American or Alaskan Native	0	0	0.00	0.00
White or Caucasian	7	6	16.67	14.29
Multiracial or Biracial (please specify)	2	1	4.76	2.38
Prefer not to respond	0	1	0.00	2.38

Table V.8: Student Race and Ethnic Identity as reported on the Post-Test (N = 42) broken down by gender.

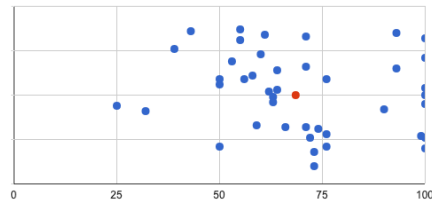


Figure V.12: Graph showing the participants' interest in interacting with the Nao Robot. The mean of the data, on a scale of 0 to 100, is 69.90 with a standard deviation of 20.20.

of the module average of 72.62 with a standard deviation of 15.67.

As seen in Figures V.12, V.13, and V.14, students were most interested in interacting with the Tello drone, of the three shown. The mean interest in coding/programming was 79.85, whilst the mean interest in interacting with drones/robots was only 71.02.

The likelihood of a participant taking a Computer Science course next semester had a mean of 92.46, on a scale of 0 to 100, and a standard deviation of 27.96. The likelihood of a participant majoring in Computer Science had a mean of 70.24 and a standard deviation of 14.38. The likelihood of a participant minoring in Computer Science had a mean of 39.29 and a standard deviation of 3.98, which makes sense due to the large population of Computer Science majors in the module.

For the concept inventory part, the questions were broken into four categories: algorithms, variables, loops/iteration, and conditionals. As shown in Figures V.15 and V.9, most of the participants responded cor-

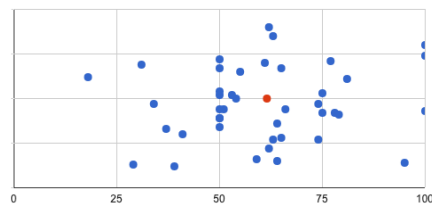


Figure V.13: Graph showing the participants' interest in interacting with the iRobot Create. The mean of the data, on a scale of 0 to 100, is 60.57 with a standard deviation of 19.11.

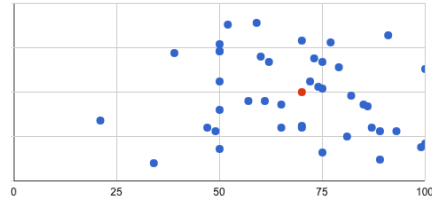


Figure V.14: Graph showing the participants' interest in interacting with the Tello drone. The mean of the data, on a scale of 0 to 100, is 68.17 with a standard deviation of 18.74.

Question Category	Percent (%) Correct
Variables	97.62
Conditionals	92.86
Loops and Iteration	90.48
Algorithms	88.10

Table V.9: Post-Test Question Correctness (N = 42)

rectly to the basic concept inventory, which is not surprising since all respondents said they had previous experience with programming or Computer Science. Participants had the performed least well on the algorithmic analysis question. With respect to the concept coverage in the course, most of the students reported either Good or Excellent coverage and experience with Python. Most of the students reported either Good or Excellent coverage and experience with user interface. Also, most of the students reported either Good or Excellent coverage and experience with Tello drones. Overall, the experience and coverage of Python out-shined the other two concepts (Tello drones and user interface).

#### V.4 Curriculum

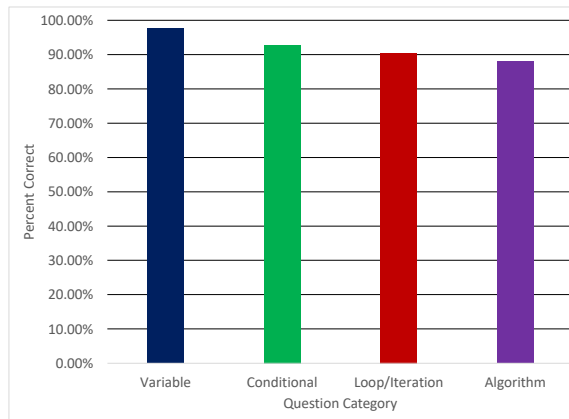


Figure V.15: Graph showing the participants' correctness with respect to the question categories on the post-test.

<b>Date/Class</b>	<b>Topic</b>	<b>Intervention</b>	<b>Tests</b>
Class 1:	Intro to Course, CS, and Real-Life CPS examples		Pre-Test Administered
Class 2:	Python Programming Introduction		
Class 3:	Python Programming		
Class 4:	Python Programming		
Class 5:	Python Programming (Lists)		
Class 6:	Python Programming (Dictionaries/Classes)		
Class 7:	Tello Drones Actuators (Propellers, Motors) and Sensors (Camera, Antennas, Vision Positioning System)	Potential Activity: look at sensing data and plot them/interact with them Concepts: time series data, physical modeling	
Class 8:	Tello Drones and Python Programming and File I/O		
Class 9:	Camera Intro/Intro to basic OpenCV work with Tello (Flying)	Potential Activity: photo then possibly video Concepts: file IO, basic computer vision, image/video format in a computer	
Class 10:	Camera Work (Camera)		
Class 11:	Camera Work (Interface Additions)		
Class 12:	Interface Intro	Activity: Fill in the skeleton interface code Concepts: Interface design, GUI, and basic control theory	

Class 13:	Interface Work	Activity: Continue filling in the skeleton interface code Concepts: Interface design, GUI, and basic control theory	
Class 14:	Interface Work Wrap Up/ CS possibilities presentation Activity: Finish filling in the skeleton interface code Concepts: Interface design, GUI, and basic control theory	Post-Test Administered	

## V.5 Contributions

- Developed a curriculum for ES 140X using the Tello Drones
- Finalized Tests and Questions for optimal and useful data collection
- Submitted IRB for:
  - Curriculum tests
  - Pre- and Post tests
- Apply designed curriculum to ES 140X modules (in progress)
- Survey the students in the ES 140X modules (in progress)
- Designed and implemented an interface for the students to use to interact with the Tello drone

## V.6 Potential Future Directions

- Submit IRB for:
  - Interviews of students
- Interview Students about their experience in ES 140X
- Drone Day:

- Small group of students (7 students) from ES 140X help plan/take lead planning
- See how accurate the self-reported interest truly is on the test
- Compare the efficacy of the tool (drone) on interest in main population vs. women identifying in engineering
- Reach out to various organizations in Engineering:
  - \* ACM-W
  - \* WiSE
  - \* Phi Sigma Rho (Engineering Sorority)
  - \* Engineering Council
- Interview the student organizers
- Ask: How would you get your (non-CS) roommate to come to the Drone Day?
  - \* Ask similar question on the post-test
- <https://ct-stem.northwestern.edu/curriculum/preview/118/>
- Introduce non-CS population to CPS
- Students from Fall ES140X plan the event

## **CHAPTER VI**

### **Conclusion**

In this thesis, the work of re-development of a curriculum and implementation of a user interface, designed with undergraduate engineering students in mind, were detailed and examined. This project and could be further explored by anybody interested in not only a chance to diversify the population and knowledge of Computer Science, but also in trying to make the world a better and more equitable place. It is hoped that the future directions detailed in Chapter V will be built upon with the goal of encouraging creativity, innovation, and acceptance in both Computer Science and Education research.

## References

- [Com] Computing Education.
- [Pay] The Highest-Paying Careers with a Bachelor's Degree for 2019.
- [3] (2017). Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006.
- [4] (2018). Tello sdk 2.0 user guide.
- [5] (2019). 2019 state of computer science education: Equity and diversity.
- [6] (2020). Computer and information technology occupations : Occupational outlook handbook.
- [7] (2020). Women and information technology by the numbers.
- [8] (2021). Cyber-physical systems (cps).
- [9] (2021). National science foundation - where discoveries begin.
- [10] Baer, A. and DeOrio, A. (2020). A Longitudinal View of Gender Balance in a Large Computer Science Program. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20*, page 23–29, New York, NY, USA. Association for Computing Machinery.
- [11] Baker, C. M., El-Glaly, Y. N., and Shinohara, K. (2020). A Systematic Analysis of Accessibility in Computing Education Research. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20*, page 107–113, New York, NY, USA. Association for Computing Machinery.
- [12] Baker, J. T. (2017). *Use of robotics in introductory computer science classrooms*. PhD thesis. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-06-25.
- [13] Bauer, K. and Schneider, K. (2012). Teaching cyber-physical systems: A programming approach. In *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education, WESE '12*, New York, NY, USA. Association for Computing Machinery.
- [14] Beede, D. N., Julian, T. A., Langdon, D., McKittrick, G., Khan, B., and Doms, M. E. (2011). Women in STEM: A Gender Gap to Innovation. *Economics and Statistics Administration Issue Brief*, (04-11).
- [15] Beyer, S., Rynes, K., Perrault, J., Hay, K., and Haller, S. (2003). Gender Differences in Computer Science Students. *SIGCSE Bull.*, 35(1):49–53.
- [16] Brown, N. C. C., Altadmri, A., Sentance, S., and Kölling, M. (2018). Blackbox, Five Years On: An Evaluation of a Large-Scale Programming Data Collection Project. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER '18*, page 196–204, New York, NY, USA. Association for Computing Machinery.
- [17] Burg, J., Pauca, V. P., Turkett, W., and Santago, P. (2016). A stem incubator to engage students in hands-on, relevant learning: A report from the field. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '16*, page 142–147, New York, NY, USA. Association for Computing Machinery.
- [18] Caceffo, R., Gama, G., Benatti, R., Aparecida, T., Caldas, T., and Azevedo, R. (2018a). A Concept Inventory for CS1 Introductory Programming Courses in C. Technical Report IC-18-06, Institute of Computing, University of Campinas. Partly in English, partly in Portuguese, 107 pages.

**englishAbstract** This work is a report related to the development and assessment of a Concept Inventory to Introductory Programming (CS1) Courses. A Concept Inventory (CI) is a set of multiple-choice questions addressing specific misunderstandings and misconceptions of the students. In previous works, through instructor interviews, exam analysis, online pilot test and interviews with students, we have identified a list of 33 misconceptions related to 7 programming topics in C language. On this report, we present a CI composed of 27 multiple-choice questions in C language. Each possible answer, besides the right one, was mapped to a previously documented misconception. Future work involves the CI submission to CS1 students and the analysis of its internal consistency and educational impact.

- [19] Caceffo, R., Wolfman, S., Booth, K., Gama, G., Garcia, I., Caldas, T., and Azevedo, R. (2018b). An Exploratory Questionnaire to Support the Identification and Assessment of Misconceptions in CS1 Courses Based on C Programming Language. Technical Report IC-18-16, Institute of Computing, University of Campinas. In English, 40 pages.  
**english Abstract:** This Technical Report is part of an ongoing work to develop and assess a Concept Inventory (CI) for Introductory Computer Programming Courses (CS1) based on the C programming language. A CI is a set of multiple-choice questions that can be used to assess the students' comprehension on some topic at some point during a course. Each incorrect choice corresponds to a specific misconception – an inaccurate line of thought students often follow. CIs exist for several knowledge areas, such as physics, chemistry, and statistics. In previous work we identified, through the analysis of open-ended exams and interviews with instructors, 19 misconceptions related to CS1 courses in the C programming language. The misconceptions were grouped into 7 topics: function parameter use and scope; variables; recursion; iteration; structures; pointers; and boolean expressions. In this work, we: a) present in details the 19 misconceptions identified on the previous work and; b) present the Exploratory Questionnaire created to assess the previous work's misconceptions and also allow the identification of new misconceptions. This Technical Report will be referenced by an article that explains the Exploratory Questionnaire design and the results of its online administration, including the student's performance and individual think-aloud interviews with selected participants.
- [20] Chen, H., Damevski, K., and Edwards, W. M. (2013). Infusing cyber-physical systems concepts into an introductory computer science course. *J. Comput. Sci. Coll.*, 28(6):26–34.
- [21] Cheng, A. M. K. (2014). An undergraduate cyber-physical systems course. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, CyPhy '14, page 31–34, New York, NY, USA. Association for Computing Machinery.
- [22] Cohoon, J. M. (2002). Recruiting and Retaining Women in Undergraduate Computing Majors. *SIGCSE Bull.*, 34(2):48–52.
- [23] Colombo, J. (2019). Computer Science Demand Is Soaring Due To Tech Bubble 2.0.
- [24] Cooper, S., Grover, S., Guzdial, M., and Simon, B. (2014). A Future for Computing Education Research. *Commun. ACM*, 57(11):34–36.
- [25] Crenshaw, T. L. A. (2013). Using robots and contract learning to teach cyber-physical systems to undergraduates. *IEEE Transactions on Education*, 56(1):116–120.
- [26] Denny, P., Becker, B. A., Craig, M., Wilson, G., and Banaszkiwicz, P. (2019). Research This! Questions That Computing Educators Most Want Computing Education Researchers to Answer. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 259–267, New York, NY, USA. Association for Computing Machinery.
- [27] Dey, N., Ashour, A., Shi, F., Fong, S., and Tavares, J. (2018). Medical cyber-physical systems: A survey. *Journal of Medical Systems*, 42:1–13.
- [28] Dorodchi, M. M., Dehbozorgi, N., Benedict, A., Al-Hossami, E., and Benedict, A. (2020). Scaffolding a team-based active learning course to engage students: A multidimensional approach. In *2020 ASEE Virtual Annual Conference Content Access*, number 10.18260/1-2-35174, Virtual Online. ASEE Conferences. <https://peer.asee.org/35174>.



- [29] Evans, L. (2015). NOT-OD-15-089: Racial and Ethnic Categories and Definitions for NIH Diversity Programs and for Other Reporting Purposes.
- [30] Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., and Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410–8415.
- [31] Gadepally, V., Krishnamurthy, A. K., and Özgüner, Ü. (2014). A Hands-on Education Program on Cyber Physical Systems for High School Students. *ArXiv*, abs/1408.0521.
- [32] Gama, G. A. R., Caceffo, R. E., Souza, R., Benatti, R., Aparecida, T., Caldas, T., Garcia, I., and Azevedo, R. (2019). Why do students make mistakes? an antipattern documentation about misconceptions related to CS1 introductory programming courses in Python. *Revista dos Trabalhos de Iniciação Científica da UNICAMP*, (26).
- [33] Gorson, J. and O’Rourke, E. (2020). Why Do CS1 Students Think They’re Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities. In *Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER ’20*, page 170–181, New York, NY, USA. Association for Computing Machinery.
- [34] Gressmann, A., Weilemann, E., Meyer, D., and Bergande, B. (2019). Nao Robot vs. Lego Mindstorms: The Influence on the Intrinsic Motivation of Computer Science Non-Majors. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli Calling ’19*, New York, NY, USA. Association for Computing Machinery.
- [35] Grimheden, M. E. and Törngren, M. (2014). Towards curricula for cyber-physical systems. In *Proceedings of the WESE’14: Workshop on Embedded and Cyber-Physical Systems Education, WESE’14*, New York, NY, USA. Association for Computing Machinery.
- [36] Helps, R. G. and Pack, S. (2013a). Introducing information technology students to cyber-physical systems using a lab experience. In *2013 ASEE Annual Conference & Exposition*, number 10.18260/1-2–19832, Atlanta, Georgia. ASEE Conferences. <https://strategy.asee.org/19832>.
- [37] Helps, R. G. and Pack, S. J. (2013b). Cyber-physical system concepts for it students. In *Proceedings of the 14th Annual ACM SIGITE Conference on Information Technology Education, SIGITE ’13*, page 7–12, New York, NY, USA. Association for Computing Machinery.
- [38] James, D. B., Mark, G., Tom, M., Charles, M., Kenneth, P., Corey, S., and S., B. A. (2009). Glitch Game Testers: African American Men Breaking Open the Console. In *DiGRA & #3909 - Proceedings of the 2009 DiGRA International Conference: Breaking New Ground: Innovation in Games, Play, Practice and Theory*. Brunel University.
- [39] James DiSalvo, B., Yardi, S., Guzdial, M., McKlin, T., Meadows, C., Perry, K., and Bruckman, A. (2011). African American Men Constructing Computing Identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’11*, page 2967–2970, New York, NY, USA. Association for Computing Machinery.
- [40] Johnson, S. (2018). Computer Science Degrees and Technology’s Boom-and-Bust Cycle.
- [41] Kafai, Y., Proctor, C., and Lui, D. (2020). From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education. *ACM Inroads*, 11(1):44–53.
- [42] Katz, S., Aronis, J., Allbritton, D., Wilson, C., and Soffa, M. L. (2003). Gender and Race in Predicting Achievement in Computer Science. *IEEE Technology and Society Magazine*, 22(3):20–27.

- [43] Kim, Y., Chen, A., Jadhav, S., Gloster, C. S., Le, T., and Hsu, P. (2016). Project based courses in control cyber physical system co-design. In *Proceedings of the 2016 Workshop on Embedded and Cyber-Physical Systems Education*, WESE '16, New York, NY, USA. Association for Computing Machinery.
- [44] Krusche, S., Seitz, A., Börstler, J., and Bruegge, B. (2017). Interactive Learning: Increasing Student Participation through Shorter Exercise Cycles. In *Proceedings of the Nineteenth Australasian Computing Education Conference*, ACE '17, page 17–26, New York, NY, USA. Association for Computing Machinery.
- [45] Laird, L. (2016). Strengthening the "engineering" in software engineering education: A software engineering bachelor of engineering program for the 21st century. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 128–131.
- [46] Laird, L. M. and Bowen, N. S. (2016). A new software engineering undergraduate program supporting the internet of things (iot) and cyber-physical systems (cps). In *2016 ASEE Annual Conference & Exposition*, number 10.18260/p.26192, New Orleans, Louisiana. ASEE Conferences. <https://strategy.asee.org/26192>.
- [47] Lee, I., Sokolsky, O., Chen, S., Hatcliff, J., Jee, E., Kim, B., King, A., Mullen-Fortino, M., Park, S., Roederer, A., and Venkatasubramanian, K. K. (2012). Challenges and research directions in medical cyber-physical systems. *Proceedings of the IEEE*, 100(1):75–90.
- [48] Levshun, D., Chevalier, Y., Kotenko, I., and Chechulin, A. (2020). Design and verification of a mobile robot based on the integrated model of cyber-physical systems. *Simulation Modelling Practice and Theory*, 105:102151.
- [49] Lewis, C., Bruno, P., Raygoza, J., and Wang, J. (2019). Alignment of Goals and Perceptions of Computing Predicts Students' Sense of Belonging in Computing. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 11–19, New York, NY, USA. Association for Computing Machinery.
- [50] Loksa, D., Xie, B., Kwik, H., and Ko, A. J. (2020). Investigating Novices' In Situ Reflections on Their Programming Process. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 149–155, New York, NY, USA. Association for Computing Machinery.
- [51] Loo, S. M. and Babinkostova, L. (2020). Cyber-physical systems security introductory course for stem students. In *2020 ASEE Virtual Annual Conference Content Access*, number 10.18260/1-2-34366, Virtual On line. ASEE Conferences. <https://peer.asee.org/34366>.
- [52] Malmi, L., Sheard, J., Kinnunen, P., Simon, and Sinclair, J. (2019). Computing Education Theories: What Are They and How Are They Used? In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 187–197, New York, NY, USA. Association for Computing Machinery.
- [53] Margulieux, L. E. (2020). Spatial Encoding Strategy Theory: The Relationship between Spatial Skill and STEM Achievement. *ACM Inroads*, 11(1):65–75.
- [54] Marwedel, P., Mitra, T., Grimheden, M. E., and Andrade, H. A. (2020). Survey on education for cyber-physical systems. *IEEE Design Test*, 37(6):56–70.
- [55] Master, A., Cheryan, S., Moscatelli, A., and Meltzoff, A. N. (2017). Programming Experience Promotes Higher STEM Motivation among First-Grade Girls. *Journal of Experimental Child Psychology*, 160:92–106.
- [56] McGill, M. M. and Decker, A. (2020). A Gap Analysis of Statistical Data Reporting in K-12 Computing Education Research: Recommendations for Improvement. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 591–597, New York, NY, USA. Association for Computing Machinery.

- [57] Meguid, E. A. A. and Collins, M. (2017). Students' perceptions of lecturing approaches: traditional versus interactive teaching. *Advances in Medical Education and Practice*, 8:229 – 241.
- [58] Miller, D. P. and Nourbakhsh, I. (2016). *Robotics for Education*, pages 2115–2134. Springer International Publishing, Cham.
- [59] Mäkiö, J., Mäkiö-Marusik, E., and Yablochnikov, E. (2016). Task-centric holistic agile approach on teaching cyber physical systems engineering. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6608–6614.
- [60] Mäkiö-Marusik, E. (2017). Current trends in teaching cyber physical systems engineering: A literature review. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 518–525.
- [61] National Academies of Sciences, Engineering, and Medicine (2018). *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. The National Academies Press, Washington, DC.
- [62] Nelson, G. L. and Ko, A. J. (2018). On Use of Theory in Computing Education Research. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER '18*, page 31–39, New York, NY, USA. Association for Computing Machinery.
- [63] Parker, M. C., Guzdial, M., and Engleman, S. (2016). Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. *Proceedings of the 2016 ACM Conference on International Computing Education Research*.
- [64] Partovi, H. (2015). A Comprehensive Effort to Expand Access and Diversity in Computer Science. *ACM Inroads*, 6(3):67–72.
- [65] Pester, A., Madritsch, C., and Klinger, T. (2015). Collaborative learning with cyber-physical systems. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, pages 184–188.
- [66] Peter, S., Momtaz, F., and Givargis, T. (2015). From the browser to the remote physical lab: Programming cyber-physical systems. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–7.
- [67] Peteranetz, M. S. and Soh, L.-K. (2020). A Multi-Level Analysis of the Relationship between Instructional Practices and Retention in Computer Science. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20*, page 37–43, New York, NY, USA. Association for Computing Machinery.
- [68] Peters, A.-K. (2018). Students' Experience of Participation in a Discipline—A Longitudinal Study of Computer Science and IT Engineering Students. *ACM Trans. Comput. Educ.*, 19(1).
- [69] Prather, J., Becker, B. A., Craig, M., Denny, P., Loksa, D., and Margulieux, L. (2020). What Do We Think We Think We Are Doing?: Metacognition and Self-Regulation in Programming. In *Proceedings of the 2020 International Computing Education Research Conference, ICER '20*, New York, NY, USA. Association for Computing Machinery.
- [70] Przybylla, M. (2016). Situating physical computing in secondary cs education. In *Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16*, page 287–288, New York, NY, USA. Association for Computing Machinery.
- [71] Rankin, Y. A. and Thomas, J. O. (2020). The Intersectional Experiences of Black Women in Computing. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20*, page 199–205, New York, NY, USA. Association for Computing Machinery.
- [72] Register, Y. and Ko, A. J. (2020). Learning Machine Learning with Personal Data Helps Stakeholders Ground Advocacy Arguments in Model Mechanics. In *Proceedings of the 2020 International Computing Education Research Conference, ICER '20*, New York, NY, USA. Association for Computing Machinery.

- [73] Rich, L., Perry, H., and Guzdial, M. (2004). A CS1 course designed to address interests of women. In *SIGCSE*.
- [74] Roberts, M. R. H., McGill, T., and Koppi, T. (2015). What students are telling us about why they left their ict course. *Innovation in Teaching and Learning in Information and Computer Sciences*, 10(3):68–83.
- [75] Rogerson, S. (2020). Poetical Potentials: The Value of Poems in Social Impact Education. *ACM Inroads*, 11(1):30–32.
- [76] Rosenstein, A., Raghu, A., and Porter, L. (2020). Identifying the prevalence of the impostor phenomenon among computer science students. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 30–36, New York, NY, USA. Association for Computing Machinery.
- [77] Schermerhorn, P., Scheutz, M., and Crowell, C. R. (2008). Robot Social Presence and Gender: Do Females View Robots Differently than Males? In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, HRI '08, page 263–270, New York, NY, USA. Association for Computing Machinery.
- [78] Sprint, G. and O'Fallon, A. (2018). Engaging programming assignments to recruit and retain cs0 students. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 1093, New York, NY, USA. Association for Computing Machinery.
- [79] Stankovic, J. A., Alemzadeh, H., Campbell, B., Lach, J., Feng, L., Fleming, C., Goodall, J., Odumosu, T., Quinn, D., Tian, Y., and Tobler, K. (2021). A graduate curriculum in cyber-physical systems. *IEEE Design Test*, 38(3):112–120.
- [80] Subramanian, K., Payton, J., and Saule, E. (2019). Retaining and engaging cs majors using bridges. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (NSF Showcase)*.
- [81] Tran, T. B. L., Törngren, M., Nguyen, H. D., Paulen, R., Gleason, N. W., and Duong, T. H. (2019). Trends in preparing cyber-physical systems engineers. *Cyber-Physical Systems*, 5(2):65–91.
- [82] Vandenberg, S., Small, S. G., Fryling, M., Flatland, R., and Egan, M. (2018). A Summer Program to Attract Potential Computer Science Majors. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 467–472, New York, NY, USA. Association for Computing Machinery.
- [83] Vicaire, P. A., Hoque, E., Xie, Z., and Stankovic, J. A. (2012). Bundle: A group-based programming abstraction for cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 8(2):379–392.
- [84] Vierhaus, H. T., Schölzel, M., Raik, J., and Ubar, R. (2014). Advanced technical education in the age of cyber physical systems. In *10th European Workshop on Microelectronics Education (EWME)*, pages 193–198.
- [85] Voas, J. and Laplante, P. (2017). Curriculum considerations for the internet of things. *Computer*, 50(1):72–75.
- [86] Wiebe, E., Rachmatullah, A., Akram, B., Boulden, D., Mott, B., Boyer, K., and Lester, J. (2020). Development and Validation of the Middle Grades Computer Science Concept Inventory (MG-CSCI) Assessment. *EURASIA Journal of Mathematics, Science and Technology Education*, 16(5).
- [87] Wiedemann, K., Chao, J., Galluzzo, B., and Simoneau, E. (2020). Mathematical Modeling with R: Embedding Computational Thinking into High School Math Classes. *ACM Inroads*, 11(1):33–42.
- [88] Wittie, L., Kurdia, A., and Huggard, M. (2017). Developing a concept inventory for computer science 2. *2017 IEEE Frontiers in Education Conference (FIE)*.

- [89] Zavaleta Bernuy, A. and Harrington, B. (2020). What Are We Asking Our Students? A Literature Map of Student Surveys in Computer Science Education. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 418–424, New York, NY, USA. Association for Computing Machinery.
- [90] Zavgorodniaia, A. (2020). Efficient Instructional Design of Programming Examples. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 581–582, New York, NY, USA. Association for Computing Machinery.
- [91] Zhang, N., Liang, J., Tomlinson, A., Boensch, F., and Sahai, A. (2020). Undergraduate-Led Survey Class to Improve CS Education for New Students. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 142–148, New York, NY, USA. Association for Computing Machinery.
- [92] Zhong, J. (2018). Actively engage students with diverse background using a more personalized approach. In *IEEE Frontiers in Education Conference, FIE 2018, San Jose, CA, USA, October 3-6, 2018*, pages 1–5. IEEE.

## **Appendix A**

### **Pre-Test**

# Pre-Test

Please complete the survey below.

Thank you!

---

Commodore ID:

---

---

May your responses be used in the research study?

- Yes
- No

## Programming Background

Do you have any programming or Computer Science experience?

- Yes  
 No

If any, what programming languages have you used?

- Python  
 Java  
 MATLAB  
 C/C++  
 JavaScript  
 PHP  
 Scratch ( or other block-based languages)  
 Other (please specify)  
 None

Please specify "Other":

\_\_\_\_\_

How experienced are you with using computers?

Not at all                      Moderately                      Extremely

=====

(Place a mark on the scale above)

How experienced are you with programming computers?

Not at all                      Moderately                      Extremely

=====

(Place a mark on the scale above)

What major(s) are you planning on declaring?

- Biomedical Engineering  
 Chemical and Biological Engineering  
 Civil Engineering     Computer Engineering  
 Computer Science  
 Electrical Engineering  
 Engineering Science  
 Mechanical Engineering  
 Undeclared     Other (please specify)

Please specify "Other":

\_\_\_\_\_

What minor(s) are you planning on declaring?

- Computer Engineering  
 Computer Science     Data Science  
 Electrical Engineering  
 Energy and Environmental Systems  
 Engineering Management  
 Environmental Engineering  
 Materials Science and Engineering  
 Nanoscience and Nanotechnology  
 Scientific Computing  
 None     Other (please specify)

Please specify "Other":

\_\_\_\_\_

Do you have any robotics or embedded systems experience?

- Yes     No

How interested are you in Computer Science?

Not at all                      Moderately                      Extremely

=====

(Place a mark on the scale above)







How interested are you in interacting with the Tello drone shown above

Not at all                      Moderately                      Extremely

\_\_\_\_\_

*(Place a mark on the scale above)*

---

**Participant Background**

---

What is your racial or ethnic identity?

- Asian or Pacific Islander
- Black or African American
- Hispanic or Latinx     Native American or Alaskan Native
- White or Caucasian
- Multiracial or Biracial (please specify)
- A race/ethnicity not listed here (please specify)
- Prefer not to respond

---

Please specify "Multiracial or Biracial":

\_\_\_\_\_

---

Please specify "A race/ethnicity not listed here":

\_\_\_\_\_

---

What is your gender identity?

- Female     Male     Transgender
- Non-binary/Non-conforming
- Prefer not to respond

---

Are you 18 years old or older?

- Yes
- No

---

Is there anything else you would like us to know about you?

\_\_\_\_\_

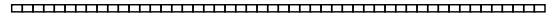
**Class Expectations**

What do you hope to get out of this class?

---

How likely are you to enroll in a Computer Science course next semester?

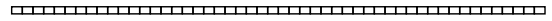
Definitely Not                      Unsure                      Definitely



*(Place a mark on the scale above)*

How likely are you to major in Computer Science?

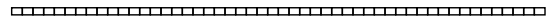
Definitely Not                      Unsure                      Definitely



*(Place a mark on the scale above)*

How likely are you to minor in Computer Science?

Definitely Not                      Unsure                      Definitely



*(Place a mark on the scale above)*

**Concept Inventory**

What are the values of x and y after the below code is run?

**x = 10**

**y = 5**

**y = x**

- x is equal to 10; y is equal to 5
- x is equal to 5; y is equal to 5
- x is equal to 10; y is equal to 10
- x is equal to 5; y is equal to x

---

What will be printed after running the code below?

```
x = 10

if x > 7:
    print("Inside the if")
else:
    print("Inside the else")

print("All done")
```

- 
- All done
  - Inside the if
  - Inside the if and All done
  - Inside the else and All done
  - Inside the if, Inside the else, and All done

---

To ensure "here" is printed 10 times, which number should be filled in the repeat block to replace t?

```
for i in range(2):  
    print("here")  
  
    for i in range(t):  
        print("here")  
  
    print("here")
```

- 
- 2
  - 3
  - 6
  - 10

---

Image 1

```
for i in range(3):  
    print("A")  
    print("B")  
print("C")  
print("D")
```

---

Image 2

```
for i in range(3):  
    print("A")  
    print("B")  
    print("C")  
    print("D")
```

Image 3

```
print("A")
print("B")
for i in range(3):
    print("C")
    print("D")
```

Image 4

```
print("C")
print("D")
for i in range(3):
    print("A")
    print("B")
```

Which lines of code above will result in the output printing "ABABABCD"?

- Image 1
- Image 2
- Image 3
- Image 4



---

Which of the following should replace \_\_\_\_\_ so that the code will print "Hello Ladies and Gentlemen"?

```
x = "Gentlemen"
```

```
y = "Hello"
```

```
z = "Ladies"
```

```
sentence = _____
```

```
print(sentence)
```

---

- z + "and" + y + x
- y + z + "and" + x
- y + x + "and" + z
- y + "and" + z + x

---

What will be printed after running the code below?

```
val = 10

if val < 40:

    print("Under 40")

if val < 21:

    print("And under 21")
```

- 
- Under 40  
 And under 21  
 Under 40 and And under 21  
 Nothing will be said

---

What will be printed after running the code below?

```
for i in range(3):

    print("apple")

    print("orange")
```

- 
- apple apple apple orange orange orange  
 apple orange  
 apple orange apple orange apple orange  
 Nothing will be printed

---

What does the code below do?

```
if x > 10:
```

```
    x = 10
```

```
else:
```

```
    if x < 5:
```

```
        x = 5
```

- 
- Makes sure the value of x is not equal to 10
  - Makes sure the value of x is less than 5
  - Makes sure the value of x is between 10 and 5
  - It always sets x equal to 10
  - This code will cause an error

## **Appendix B**

### **Post-Test**

# Post-Test

Please complete the survey below.

Thank you!

Commodore ID:

---

What major(s) are you planning on declaring?

- Biomedical Engineering  
 Chemical and Biological Engineering  
 Civil Engineering    Computer Engineering    Computer Science  
 Electrical Engineering  
 Engineering Science  
 Mechanical Engineering  
 Undeclared    Other (please specify)

Please specify "Other":

---

What minor(s) are you planning on declaring?

- Computer Engineering  
 Computer Science    Data Science  
 Electrical Engineering  
 Energy and Environmental Systems  
 Engineering Management  
 Environmental Engineering  
 Materials Science and Engineering  
 Nanoscience and Nanotechnology  
 Scientific Computing  
 None    Other (please specify)

Please specify "Other":

---

What is your racial or ethnic identity?

- Asian or Pacific Islander  
 Black or African American  
 Hispanic or Latinx    Native American or Alaskan Native    White or Caucasian  
 Multiracial or Biracial (please specify)  
 A race/ethnicity not listed here (please specify)    Prefer not to respond

Please specify "Multiracial or Biracial":

---

Please specify "A race/ethnicity not listed here":

---

What is your gender identity?

- Female    Male    Transgender  
 Non-binary/Non-conforming  
 Prefer not to respond

Are you 18 years old or older?

- Yes  
 No

Not at all

Slightly

Moderately

Very

Extremely

Prior to this module, how interested were you in Computer Science?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prior to this module, how interested were you in programming/coding?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prior to this module, how interested were you in interacting with drones/robots?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Now after completing the module, how interested are you in Computer Science?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Now after completing the module, how interested are you in programming/coding?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Now after completing the module, how interested are you in interacting with drones/robots?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

---



How interested are you in interacting with the Nao robot shown above?

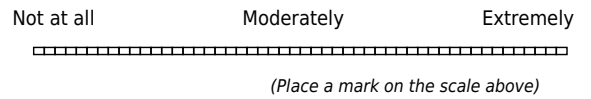
Not at all                      Moderately                      Extremely

=====

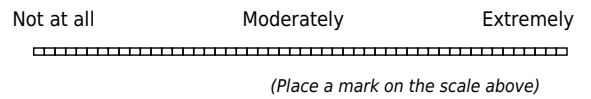
(Place a mark on the scale above)



How interested are you in interacting with the iRobot Create robot shown above



How interested are you in interacting with the Tello drone shown above



What did you like about the material covered in the module?

---

What would you change about any aspect of the module?

---

Is there any other feedback you would like to share with us?

	Very Poor	Poor	Fair	Good	Excellent
The coverage of Python was _____.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
My experience with Python was _____.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The coverage of the user interface was _____.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
My experience with the user interface was _____.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The coverage of the Tello drones was _____.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
My experience with the Tello drones was _____.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

	Definitely Not	Probably Not	Possibly	Unsure	Probably	Very Probably	Definitely
How likely are you to enroll in a Computer Science course next semester?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
How likely are you to major in Computer Science?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
How likely are you to minor in Computer Science?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



**Concept Inventory**

What are the values of x and y after the below code is run?

**x = 10**

**y = 5**

**y = x**

- x is equal to 10; y is equal to 5
- x is equal to 5; y is equal to 5
- x is equal to 10; y is equal to 10
- x is equal to 5; y is equal to x

---

What will be printed after running the code below?

```
x = 10

if x > 7:
    print("Inside the if")
else:
    print("Inside the else")

print("All done")
```

- 
- All done
  - Inside the if
  - Inside the if and All done
  - Inside the else and All done
  - Inside the if, Inside the else, and All done

---

To ensure "here" is printed 10 times, which number should be filled in the repeat block to replace t?

```
for i in range(2):  
    print("here")  
  
    for i in range(t):  
        print("here")  
  
    print("here")
```

- 
- 2
  - 3
  - 6
  - 10

---

Image 1

```
for i in range(3):  
    print("A")  
    print("B")  
print("C")  
print("D")
```

---

Image 2

```
for i in range(3):  
    print("A")  
    print("B")  
    print("C")  
    print("D")
```

Image 3

```
print("A")
print("B")
for i in range(3):
    print("C")
    print("D")
```

Image 4

```
print("C")
print("D")
for i in range(3):
    print("A")
    print("B")
```

Which lines of code above will result in the output printing "ABABABCD"?

- Image 1
- Image 2
- Image 3
- Image 4

---

Which of the following should replace \_\_\_\_\_ so that the code will print "Hello Ladies and Gentlemen"?

```
x = "Gentlemen"
```

```
y = "Hello"
```

```
z = "Ladies"
```

```
sentence = _____
```

```
print(sentence)
```

---

- z + "and" + y + x
- y + z + "and" + x
- y + x + "and" + z
- y + "and" + z + x

---

What will be printed after running the code below?

```
val = 10

if val < 40:

    print("Under 40")

if val < 21:

    print("And under 21")
```

- 
- Under 40  
 And under 21  
 Under 40 and And under 21  
 Nothing will be said

---

What will be printed after running the code below?

```
for i in range(3):

    print("apple")

    print("orange")
```

- 
- apple apple apple orange orange orange  
 apple orange  
 apple orange apple orange apple orange  
 Nothing will be printed

---

What does the code below do?

```
if x > 10:
```

```
    x = 10
```

```
else:
```

```
    if x < 5:
```

```
        x = 5
```

- 
- Makes sure the value of x is not equal to 10
  - Makes sure the value of x is less than 5
  - Makes sure the value of x is between 10 and 5
  - It always sets x equal to 10
  - This code will cause an error

## Appendix C

### Skeleton Code for Tello Interface Given to Students

#### C.1 Main File



```

import math
from tkinter import *
import tellopy
from threading import Thread
import cv2, time
import os, numpy

#Global Variables
distanceEntry = None
sideLengthEntry = None
tello = tellopy.Tello()
stream_on = False
frameRead = None
capture = None

class Window(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master

        self.pack(fill=BOTH, expand=1)
        # Initializing the Buttons for the GUI
        exitButton = Button(self, text="Exit", command=self.clickExitButton) #
            Exit button
        takeoffButton = Button(self,
            text="Takeoff",command=self.clickTakeoffButton) # Takeoff button
        landButton = Button(self, text="Land", command=self.clickLandButton) #
            Land button
        shapesButton = Button(self, text="Shapes",
            command=self.clickShapesButton) # Shapes Menu button
        cameraButton = Button(self, text="Camera",
            command=self.clickCameraButton) # Camera Menu button
        generalControlButton = Button(self,text="General Controls",
            command=self.clickControlButton) # General Drone Control Menu button
        # Placing the buttons for the GUI in a grid layout
        takeoffButton.grid(row=0,column=1)
        landButton.grid(row=0,column=3)
        generalControlButton.grid(row=1,column=2)
        shapesButton.grid(row=2, column=1)
        cameraButton.grid(row=2, column=3)
        exitButton.grid(row=3,column=2)

        # Lands the drone and quits the program on the press of the Exit button
    def clickExitButton(self):
        global frameRead
        tello.land()
        tello.quit()
        exit()

```

```

# Loads the drone basic control window
def clickControlButton(self):
    global tello, distanceEntry
    # Brings the window to the front
    controlWindow = Toplevel(root)
    # Titles the window "Drone Control Interface"
    controlWindow.title("Drone Control Interface")
    # Sets the geometry of the window to 320 x 200
    controlWindow.geometry("320x200")
    # Creating the label and entry box for getting the desired distance
    distanceLabel = Label(controlWindow, text="Distance:")
    distanceEntry = Entry(controlWindow)
    # Initializing the Buttons for the basic drone controls: Up, Down,
    # Left, Right, Forward, Backward
    upButton = Button(controlWindow, text="Up", command=self.clickUpButton)
    downButton = Button(controlWindow, text="Down",
        command=self.clickDownButton)
    leftButton = Button(controlWindow, text="Left",
        command=self.clickLeftButton)
    rightButton = Button(controlWindow, text="Right",
        command=self.clickRightButton)
    forwardButton = Button(controlWindow,
        text="Forward", command=self.clickForwardButton)
    backwardButton = Button(controlWindow,
        text="Backward", command=self.clickBackwardButton)
    returnButton = Button(controlWindow, text="Return to Main",
        command=self.clickReturnButton)
    # Place the label and entry box for getting the desired distance
    distanceLabel.grid(row=0, column=1)
    distanceEntry.grid(row=0, column=2)
    # Placing the buttons in a grid layout
    upButton.grid(row=1, column=2)
    downButton.grid(row=3, column=2)
    leftButton.grid(row=2, column=1)
    rightButton.grid(row=2, column=3)
    forwardButton.grid(row=4, column=1)
    backwardButton.grid(row=4, column=3)
    returnButton.grid(row=5, column=2)
    self.controlWindow= controlWindow
    self.shapesWindow = None
    self.cameraWindow = None
    self.circleWindow = None
    self.triangleWindow = None
    self.cubeWindow = None
    self.squareWindow = None

# Function that activates when you click the up button
def clickUpButton(self):
    global tello, distanceEntry
    length = int(distanceEntry.get())

```

```
print("Going Up {}".format(length))
tello.connect()
tello.takeoff()
time.sleep(1)
self.moveUp(length)
tello.land()
```

```
#
```

```
-----
# Function that tells the drone to move upward, called by the clickUpButton
# TODO: Fill in code here
```

```
def moveUp(self, distance):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to distance upward command using the Tello SDK command
```

```
    # Sleep between commands for 2
    time.sleep(2)
    # Send up command to the tello
```

```
    # Sleep between commands for 2
    time.sleep(2)
```

```
# TODO_END: end of added code
```

```
#
```

```
-----
# Function that activates when you click the down button
```

```
def clickDownButton(self):
    global tello, distanceEntry
    length = int(distanceEntry.get())
    print("Going Down %d" %(length))
    tello.connect()
    tello.takeoff()
    time.sleep(1)
    self.moveDown(length)
    tello.land()
```

```
#
```

```
-----
# Function that tells the drone to move downward, called by the
clickDownButton
```

```
# TODO: Fill in code here
```

```
def moveDown(self, distance):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to distance downward command using the Tello SDK command
```

```
    # Sleep between commands for 2
    time.sleep(2)
    # Send down command to the tello
```

```

        # Sleep between commands for 2
        time.sleep(2)
    # TODO_END: end of added code
#-----

# Function that activates when you click the left button
def clickLeftButton(self):
    global tello, distanceEntry
    length = int(distanceEntry.get())
    print("Going Left {}".format(length))
    tello.connect()
    tello.takeoff()
    time.sleep(1)
    self.moveLeft(length)
    tello.land()

#-----

# Function that tells the drone to move left, called by the clickLeftButton
# TODO: Fill in code here
def moveLeft(self, distance):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to distance left command using the Tello SDK command

    # Sleep between commands for 2
    time.sleep(2)
    # Send left command to the tello

    # Sleep between commands for 2
    time.sleep(2)
    # TODO_END: end of added code
#-----

# Function that activates when you click the right button
def clickRightButton(self):
    global tello, distanceEntry
    length = int(distanceEntry.get())
    print("Going Right {}".format(length))
    tello.connect()
    tello.takeoff()
    self.moveRight(length)
    tello.land()

#-----

```

```

# Function that tells the drone to move right, called by the
  clickRightButton
# TODO: Fill in code here
def moveRight(self, distance):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to distance right command using the Tello SDK command

    # Sleep between commands for 2
    time.sleep(2)
    # Send right command to the tello

    # Sleep between commands for 2
    time.sleep(2)
# TODO_END: end of added code
# -----
-----

# Function that activates when you click the forward button
def clickForwardButton(self):
    global tello, distanceEntry
    length = int(distanceEntry.get())
    print("Going Forward {}".format(length))
    tello.connect()
    tello.takeoff()
    time.sleep(1)
    self.moveForward(length)
    tello.land()

# -----
-----

# Function that tells the drone to move forward, called by the
  clickForwardButton
# TODO: Fill in code here
def moveForward(self, distance):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to distance forward command using the Tello SDK command

    # Sleep between commands for 2
    time.sleep(2)
    # Send forward command to the tello

    # Sleep between commands for 2
    time.sleep(2)
# TODO_END: end of added code
# -----
-----

```

```

# Function that activates when you click the backward button
def clickBackwardButton(self):
    global tello, distanceEntry
    length = int(distanceEntry.get())
    print("Going Backward {}".format(length))
    tello.connect()
    tello.takeoff()
    time.sleep(5)
    self.moveBackward(length)
    time.sleep(1)
    tello.land()

# -----
# Function that tells the drone to move backward, called by the
# clickBackwardButton
# TODO: Fill in code here
def moveBackward(self, distance):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to distance backward command using the Tello SDK command

    # Sleep between commands for 2
    time.sleep(2)
    # Send backward command to the tello

    # Sleep between commands for 2
    time.sleep(2)

# Function that tells the drone to move clockwise
def moveClockwise(self, angle):
    # Using the global variable tello to communicate with the drone
    global tello
    # Set variable to angle to rotate clockwise command using the Tello
    SDK command

    # Sleep between commands for 2
    time.sleep(2)
    # Send clockwise command to the tello

    # Sleep between commands for 2
    time.sleep(2)
# TODO_END: end of added code
# -----
# Function that activates when you click the takeoff button
def clickTakeoffButton(self):
    global tello

```

```
tello.connect()
tello.takeoff()
time.sleep(2)
print("Takeoff")
```

```
# Function that activates when you click the land button
```

```
def clickLandButton(self):
    global tello
    tello.connect()
    tello.land()
    time.sleep(2)
    print("Land")
```

```
# Function that activates when you click the shapes button
```

```
def clickShapesButton(self):
    shapesWindow = Toplevel(root)
    shapesWindow.title("Shapes Interface")
    shapesWindow.geometry("320x200")
    circleButton = Button(shapesWindow, text="Circle",
        command=self.clickCircleButton)
    squareButton = Button(shapesWindow, text="Square",
        command=self.clickSquareButton)
    cubeButton = Button(shapesWindow, text="Cube",
        command=self.clickCubeButton)
    triangleButton = Button(shapesWindow, text="Triangle",
        command=self.clickTriangleButton)
    returnButton = Button(shapesWindow, text="Return to Main",
        command=self.clickReturnButton)
    returnButton.place(x=105, y=180)
    circleButton.place(x=103, y=30)
    squareButton.place(x=173, y=30)
    triangleButton.place(x=103, y=80)
    cubeButton.place(x=173, y=80)
    self.shapesWindow= shapesWindow
    self.cameraWindow = None
    self.circleWindow = None
    self.triangleWindow = None
    self.cubeWindow = None
    self.squareWindow = None
```

```
# Function that activates when you click the camera button
```

```
def clickCameraButton(self):
    cameraWindow = Toplevel(root)
    cameraWindow.title("Camera Interface")
    cameraWindow.geometry("320x200")
    photoButton = Button(cameraWindow, text="Take a Photo",
        command=self.clickPhotoButton)
    videoButton = Button(cameraWindow, text="Record a Video",
        command=self.clickVideoButton)
    returnButton = Button(cameraWindow, text="Return to Main",
        command=self.clickReturnButton)
```

```
photoButton.place(x=75,y=75)
videoButton.place(x=175,y=75)
returnButton.place(x=115, y=180)
self.shapesWindow = None
self.cameraWindow = cameraWindow
self.cubeWindow = None
self.triangleWindow = None
self.squareWindow = None
self.circleWindow = None
```

```
# Function that activates when you click the circle button
```

```
def clickCircleButton(self):
    print("Circle")
    global tello, sideLengthEntry
    # tello = tello.Tello()
    circleWindow = Toplevel(root)
    circleWindow.title("Circle Interface")
    circleWindow.geometry("320x200")
    sideLengthLabel = Label(circleWindow, text="Radius:")
    sideLengthEntry = Entry(circleWindow)
    returnButton = Button(circleWindow, text="Return to Shapes Main",
        command=self.clickReturnButton)
    drawButton = Button(circleWindow, text="Draw Circle",
        command=self.drawCircle)
    sideLengthLabel.grid(row=0, column=0)
    sideLengthEntry.grid(row=0, column=1)
    drawButton.grid(row=1, column=1)
    returnButton.grid(row=4, column=1)
    self.shapesWindow = None
    self.cubeWindow = None
    self.triangleWindow = None
    self.squareWindow = None
    self.circleWindow = circleWindow
    self.cameraWindow = None
```

```
# Function that activates when you click the square button
```

```
def clickSquareButton(self):
    print("Square")
    global tello, sideLengthEntry
    squareWindow = Toplevel(root)
    squareWindow.title("Square Interface")
    squareWindow.geometry("320x200")
    sideLengthLabel = Label(squareWindow, text="Side Length:")
    sideLengthEntry = Entry(squareWindow)
    returnButton = Button(squareWindow, text="Return to Shapes Main",
        command=self.clickReturnButton)
    drawButton = Button(squareWindow, text="Draw Square",
        command=self.drawSquare)
    sideLengthLabel.grid(row=0, column=0)
    sideLengthEntry.grid(row=0, column=1)
    drawButton.grid(row=1, column=1)
```



```
returnButton.grid(row=4, column=1)
self.shapesWindow = None
self.cubeWindow = None
self.triangleWindow = None
self.squareWindow = squareWindow
self.circleWindow = None
self.cameraWindow = None
```

```
# Function that activates when you click the triangle button
```

```
def clickTriangleButton(self):
    print("Triangle")
    global tello, sideLengthEntry
    triangleWindow = Toplevel(root)
    triangleWindow.title("Triangle Interface")
    triangleWindow.geometry("320x200")
    sideLengthLabel = Label(triangleWindow, text="Side Length:")
    sideLengthEntry = Entry(triangleWindow)
    returnButton = Button(triangleWindow, text="Return to Shapes Main",
        command=self.clickReturnButton)
    drawButton = Button(triangleWindow, text="Draw Triangle",
        command=self.drawTriangle)
    sideLengthLabel.grid(row=0, column=0)
    sideLengthEntry.grid(row=0, column=1)
    drawButton.grid(row=1, column=1)
    returnButton.grid(row=4, column=1)
    self.shapesWindow = None
    self.cubeWindow = None
    self.triangleWindow = triangleWindow
    self.squareWindow = None
    self.circleWindow = None
    self.cameraWindow = None
```

```
# Function that activates when you click the cube button
```

```
def clickCubeButton(self):
    print("Cube")
    global tello, sideLengthEntry
    cubeWindow = Toplevel(root)
    cubeWindow.title("Cube Interface")
    cubeWindow.geometry("320x200")
    sideLengthLabel = Label(cubeWindow, text="Side Length:")
    sideLengthEntry = Entry(cubeWindow)
    returnButton = Button(cubeWindow, text="Return to Shapes Main",
        command=self.clickReturnButton)
    drawButton = Button(cubeWindow, text="Draw Cube",
        command=self.drawCube)
    sideLengthLabel.grid(row=0, column=0)
    sideLengthEntry.grid(row=0, column=1)
    drawButton.grid(row=1, column=1)
    returnButton.grid(row=4, column=1)
    self.shapesWindow = None
    self.cubeWindow = cubeWindow
```

```
self.triangleWindow = None
self.squareWindow = None
self.circleWindow = None
self.cameraWindow = None
```

```
#-----
```

```
# Function that tells the drone to draw a Triangle
```

```
# TODO: Fill in code here
```

```
def drawTriangle(self):
```

```
    # Using the global variable tello to communicate with the drone, and
    # sideLengthEntry to get the value from the GUI
```

```
    global tello, sideLengthEntry
```

```
    # Save the sideLengthEntry to the length variable
```

```
    # Connect to the drone
```

```
    tello.connect()
```

```
    # Tell the drone to takeoff
```

```
    # Iterate 3 times
```

```
    x =
```

```
    for i in range(x):
```

```
        # Move forward the length
```

```
        # Sleep between commands for 2
```

```
        time.sleep(2)
```

```
        # Rotate clockwise by the appropriate angle
```

```
        # Sleep between commands for 2
```

```
        time.sleep(2)
```

```
    # Land the drone
```

```
    tello.land()
```

```
# Function that tells the drone to draw a Circle
```

```
def drawCircle(self):
```

```
    # Using the global variable tello to communicate with the drone, and
    # sideLengthEntry to get the value from the GUI
```

```
    global tello, sideLengthEntry
```

```
    # Save the sideLengthEntry to the radius variable
```

```
    # Calculate the side length based on the radius
```

```
    length = int((2*radius)*math.sin(math.pi/36))
```

```
    # Connect to the drone
```

```
    tello.connect()
```

```
    # Tell the drone to takeoff
```

```
    # Iterate 36 times
```

```
    x =
```

```
    for i in range(x):
```

```
        # Move forward the length
```

```

        # Sleep between commands for 2
        time.sleep(2)
        # Rotate clockwise by the appropriate angle

        # Sleep between commands for 2
        time.sleep(2)
    # Land the drone
    tello.land()

# Function that tells the drone to draw a Square
def drawSquare(self):
    # Using the global variable tello to communicate with the drone, and
    # sideLengthEntry to get the value from the GUI
    global tello, sideLengthEntry
    # Save the sideLengthEntry to the length variable

    # Connect to the drone
    tello.connect()
    # Tell the drone to takeoff

    # Iterate 4 times
    x =
    for i in range(x):
        # Move forward the length

        # Sleep between commands for 2
        time.sleep(2)
        # Rotate clockwise by the appropriate angle

        # Sleep between commands for 2
        time.sleep(2)
    # Land the drone
    tello.land()

def drawCube(self):
    # Using the global variable tello to communicate with the drone, and
    # sideLengthEntry to get the value from the GUI
    global tello, sideLengthEntry
    # Save the sideLengthEntry to the length variable
    # Connect to the drone
    tello.connect()
    # Tell the drone to takeoff

    # Move forward the length

    # Sleep between commands for 2
    time.sleep(2)
    # Move right the length

    # Sleep between commands for 2

```

```

time.sleep(2)
# Move backward the length

# Sleep between commands for 2
time.sleep(2)
# Move left the length

# Sleep between commands for 2
time.sleep(2)
# Move down the length

# Sleep between commands for 2
time.sleep(2)
# Move forward the length

# Sleep between commands for 2
time.sleep(2)
# Move right the length

# Sleep between commands for 2
time.sleep(2)
# Move backward the length

# Sleep between commands for 2
time.sleep(2)
# Move left the length

# Sleep between commands for 22
time.sleep(2)
# Land the drone
tello.land()
# TODO_END: end of added code
#-----
-----

# Function that activates when you click the Photo button
def clickPhotoButton(self):
    import takePictureStudent

# Function that activates when you click the Video button
def clickVideoButton(self):
    import recordVideoStudent

# Function executed when return to previous menu
def clickReturnButton(self):
    if self.cameraWindow is not None:
        self.cameraWindow.destroy()
        self.cameraWindow = None

```

```
elif self.shapesWindow is not None:
    self.shapesWindow.destroy()
    self.shapesWindow = None
elif self.cubeWindow is not None:
    self.cubeWindow.destroy()
    self.cubeWindow = None
elif self.triangleWindow is not None:
    self.triangleWindow.destroy()
    self.triangleWindow = None
elif self.squareWindow is not None:
    self.squareWindow.destroy()
    self.squareWindow = None
elif self.circleWindow is not None:
    self.circleWindow.destroy()
    self.circleWindow = None
elif self.controlWindow is not None:
    self.controlWindow.destroy()
    self.controlWindow = None
```

```
root = Tk()
app = Window(root)
root.wm_title("Tello Interaction Interface")
root.geometry("320x160")
root.mainloop()
```

## C.2 Take Picture File

```

"""Library for interacting with DJI Ryze Tello drones.
"""

# coding=utf-8
import logging
import socket
import time
from threading import Thread
from typing import Optional, Union, Type, Dict
import datetime

import cv2

threads_initialized = False
drones: Optional[dict] = {}
client_socket: socket.socket

class Tello:
    """Python wrapper to interact with the Ryze Tello drone using the official
    Tello api.
    Tello API documentation:
    [1.3](https://dl-cdn.ryzerobotics
    .com/downloads/tello/20180910/Tello%20SDK%20Documentation%20EN_1.3.pdf),
    [2.0 with EDU-only
    commands](https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%20
    .0%20User%20Guide.pdf)
    """
    # Send and receive commands, client socket
    RESPONSE_TIMEOUT = 7 # in seconds
    TAKEOFF_TIMEOUT = 20 # in seconds
    FRAME_GRAB_TIMEOUT = 3
    TIME_BTW_COMMANDS = 0.1 # in seconds
    TIME_BTW_RC_CONTROL_COMMANDS = 0.001 # in seconds
    RETRY_COUNT = 3 # number of retries after a failed command
    TELLO_IP = '192.168.10.1' # Tello IP address

    # Video stream, server socket
    VS_UDP_IP = '0.0.0.0'
    VS_UDP_PORT = 11111

    CONTROL_UDP_PORT = 8889
    STATE_UDP_PORT = 8890

    # Set up logger
    HANDLER = logging.StreamHandler()
    FORMATTER = logging.Formatter('[%(levelname)s] %(filename)s - %(lineno)d -
    %(message)s')
    HANDLER.setFormatter(FORMATTER)

    LOGGER = logging.getLogger('djitellopy')

```

```

LOGGER.addHandler(HANDLER)
LOGGER.setLevel(logging.INFO)
# Use Tello.LOGGER.setLevel(logging.<LEVEL>) in YOUR CODE
# to only receive logs of the desired level and higher

# Conversion functions for state protocol fields
INT_STATE_FIELDS = (
    # Tello EDU with mission pads enabled only
    'mid', 'x', 'y', 'z',
    # 'mpry': (custom format 'x,y,z')
    # Common entries
    'pitch', 'roll', 'yaw',
    'vgx', 'vgy', 'vgz',
    'templ', 'temph',
    'tof', 'h', 'bat', 'time'
)
FLOAT_STATE_FIELDS = ('baro', 'agx', 'agy', 'agz')

state_field_converters: Dict[str, Union[Type[int], Type[float]]]
state_field_converters = {key : int for key in INT_STATE_FIELDS}
state_field_converters.update({key : float for key in FLOAT_STATE_FIELDS})

# VideoCapture object
cap: Optional[cv2.VideoCapture] = None
background_frame_read: Optional['BackgroundFrameRead'] = None

stream_on = False
is_flying = False

def __init__(self,
             host=TELLO_IP,
             retry_count=RETRY_COUNT):

    global threads_initialized, client_socket, drones

    self.address = (host, Tello.CONTROL_UDP_PORT)
    self.stream_on = False
    self.retry_count = retry_count
    self.last_received_command_timestamp = time.time()
    self.last_rc_control_timestamp = time.time()

    if not threads_initialized:
        # Run Tello command responses UDP receiver on background
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        client_socket.bind(('', Tello.CONTROL_UDP_PORT))
        response_receiver_thread =
            Thread(target=Tello.udp_response_receiver)
        response_receiver_thread.daemon = True
        response_receiver_thread.start()

        # Run state UDP receiver on background

```



```

        state_receiver_thread = Thread(target=Tello.udp_state_receiver)
        state_receiver_thread.daemon = True
        state_receiver_thread.start()

        threads_initialized = True

drones[host] = {'responses': [], 'state': {}}

self.LOGGER.info("Tello instance was initialized. Host: '{}'. Port:
'{}'.format(host, Tello.CONTROL_UDP_PORT))

def get_own_udp_object(self):
    """Get own object from the global drones dict. This object is filled
    with responses and state information by the receiver threads.
    Internal method, you normally wouldn't call this yourself.
    """
    global drones

    host = self.address[0]
    return drones[host]

@staticmethod
def udp_response_receiver():
    """Setup drone UDP receiver. This method listens for responses of
    Tello.
    Must be run from a background thread in order to not block the main
    thread.
    Internal method, you normally wouldn't call this yourself.
    """
    while True:
        try:
            data, address = client_socket.recvfrom(1024)

            address = address[0]
            Tello.LOGGER.debug('Data received from {} at
            client_socket'.format(address))

            if address not in drones:
                continue

            drones[address]['responses'].append(data)

        except Exception as e:
            Tello.LOGGER.error(e)
            break

@staticmethod
def udp_state_receiver():
    """Setup state UDP receiver. This method listens for state information
    from
    Tello. Must be run from a background thread in order to not block

```

```

the main thread.
Internal method, you normally wouldn't call this yourself.
"""
state_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
state_socket.bind(("", Tello.STATE_UDP_PORT))

while True:
    try:
        data, address = state_socket.recvfrom(1024)

        address = address[0]
        Tello.LOGGER.debug('Data received from {} at
state_socket'.format(address))

        if address not in drones:
            continue

        data = data.decode('ASCII')
        drones[address]['state'] = Tello.parse_state(data)

    except Exception as e:
        Tello.LOGGER.error(e)
        break

@staticmethod
def parse_state(state: str) -> Dict[str, Union[int, float, str]]:
    """Parse a state line to a dictionary
Internal method, you normally wouldn't call this yourself.
"""
    state = state.strip()
    Tello.LOGGER.debug('Raw state data: {}'.format(state))

    if state == 'ok':
        return {}

    state_dict = {}
    for field in state.split(';'):
        split = field.split(':')
        if len(split) < 2:
            continue

        key = split[0]
        value: Union[int, float, str] = split[1]

        if key in Tello.state_field_converters:
            num_type = Tello.state_field_converters[key]
            try:
                value = num_type(value)
            except ValueError as e:
                Tello.LOGGER.debug('Error parsing state value for {}: {}
to {}'.format(key, value, num_type))

```

```

                .format(key, value, num_type))
        Tello.LOGGER.error(e)
        continue

        state_dict[key] = value

    return state_dict

def get_current_state(self) -> dict:
    """Call this function to attain the state of the Tello. Returns a dict
    with all fields.
    Internal method, you normally wouldn't call this yourself.
    """
    return self.get_own_udp_object()['state']

def get_state_field(self, key: str):
    """Get a specific sate field by name.
    Internal method, you normally wouldn't call this yourself.
    """
    state = self.get_current_state()

    if key in state:
        return state[key]
    else:
        raise Exception('Could not get state property: {}'.format(key))

def get_udp_video_address(self) -> str:
    """Internal method, you normally wouldn't call this yourself.
    """
    address_schema = 'udp://@{ip}:{port}' # +
        '?overrun_nonfatal=1&fifo_size=5000'
    address = address_schema.format(ip=self.VS_UDP_IP,
        port=self.VS_UDP_PORT)
    return address

def get_video_capture(self):
    """Get the VideoCapture object from the camera drone.
    Users usually want to use get_frame_read instead.
    Returns:
        VideoCapture
    """

    if self.cap is None:
        self.cap = cv2.VideoCapture(self.get_udp_video_address())

    if not self.cap.isOpened():
        self.cap.open(self.get_udp_video_address())

    return self.cap

def get_frame_read(self) -> 'BackgroundFrameRead':

```

```
"""Get the BackgroundFrameRead object from the camera drone. Then, you
just need to call
backgroundFrameRead.frame to get the actual frame received by the
drone.
```

```
Returns:
```

```
BackgroundFrameRead
```

```
"""
```

```
if self.background_frame_read is None:
    address = self.get_udp_video_address()
    self.background_frame_read = BackgroundFrameRead(self, address) #
    also sets self.cap
    self.background_frame_read.start()
return self.background_frame_read
```

```
def send_command_with_return(self, command: str, timeout: int =
RESPONSE_TIMEOUT) -> str:
```

```
"""Send command to Tello and wait for its response.
Internal method, you normally wouldn't call this yourself.
```

```
Return:
```

```
bool/str: str with response text on success, False when
unsuccessful.
```

```
"""
```

```
# Commands very consecutive makes the drone not respond to them.
```

```
# So wait at least self.TIME_BTW_COMMANDS seconds
```

```
diff = time.time() - self.last_received_command_timestamp
```

```
if diff < self.TIME_BTW_COMMANDS:
```

```
    self.LOGGER.debug('Waiting {} seconds to execute command:
    {...}'.format(diff, command))
    time.sleep(diff)
```

```
self.LOGGER.info("Send command: '{}'.format(command))
```

```
timestamp = time.time()
```

```
client_socket.sendto(command.encode('utf-8'), self.address)
```

```
responses = self.get_own_udp_object()['responses']
```

```
while not responses:
```

```
    if time.time() - timestamp > timeout:
        message = "Aborting command '{}'. Did not receive a response
        after {} seconds".format(command, timeout)
        self.LOGGER.warning(message)
        return message
    time.sleep(0.1) # Sleep during send command
```

```
self.last_received_command_timestamp = time.time()
```

```
first_response = responses.pop(0) # first datum from socket
```

```
try:
```

```
    response = first_response.decode("utf-8")
```

```
except UnicodeDecodeError as e:
```

```

        self.LOGGER.error(e)
        return "response decode error"
response = response.rstrip("\r\n")

self.LOGGER.info("Response {}: '{}'.format(command, response))
return response

def send_command_without_return(self, command: str):
    """Send command to Tello without expecting a response.
    Internal method, you normally wouldn't call this yourself.
    """
    # Commands very consecutive makes the drone not respond to them. So
    # wait at least self.TIME_BTW_COMMANDS seconds

    self.LOGGER.info("Send command (no response expected):
    '{}'.format(command))
    client_socket.sendto(command.encode('utf-8'), self.address)

def send_control_command(self, command: str, timeout: int =
RESPONSE_TIMEOUT) -> bool:
    """Send control command to Tello and wait for its response.
    Internal method, you normally wouldn't call this yourself.
    """
    response = "max retries exceeded"
    for i in range(0, self.retry_count):
        response = self.send_command_with_return(command, timeout=timeout)

        if response.lower() == 'ok':
            return True

        self.LOGGER.debug("Command attempt #{} failed for command:
        '{}'.format(i, command))

    self.raise_result_error(command, response)
    return False # never reached

def connect(self, wait_for_state=True):
    """Enter SDK mode. Call this before any of the control functions.
    """
    self.send_control_command("command")

    if wait_for_state:
        REPS = 20
        for i in range(REPS):
            if self.get_current_state():
                t = i / REPS # in seconds
                Tello.LOGGER.debug("''.connect()' received first state
                packet after {} seconds".format(t))
                break
            time.sleep(1 / REPS)

```

```

        if not self.get_current_state():
            raise Exception('Did not receive a state packet from the
                Tello')

def streamon(self):
    """Turn on video streaming. Use `tello.get_frame_read` afterwards.
    Video Streaming is supported on all tellos when in AP mode (i.e.
    when your computer is connected to Tello-XXXXXX WiFi network).
    Currently Tello EDUs do not support video streaming while connected
    to a WiFi-network.

    !!! Note:
        If the response is 'Unknown command' you have to update the Tello
        firmware. This can be done using the official Tello app.
    """
    self.send_control_command("streamon")
    self.stream_on = True

def streamoff(self):
    """Turn off video streaming.
    """
    self.send_control_command("streamoff")
    self.stream_on = False

class BackgroundFrameRead:
    """
    This class read frames from a VideoCapture in background. Use
    backgroundFrameRead.frame to get the current frame.
    """

    def __init__(self, tello, address):
        tello.cap = cv2.VideoCapture(address)

        self.cap = tello.cap

        if not self.cap.isOpened():
            self.cap.open(address)

        # Try grabbing a frame multiple times
        # According to issue #90 the decoder might need some time
        #
        # https://github
        # .com/damiafuentes/DJITelloPy/issues/90#issuecomment-855458905
        start = time.time()
        while time.time() - start < Tello.FRAME_GRAB_TIMEOUT:
            Tello.LOGGER.debug('trying to grab a frame...')
            self.grabbed, self.frame = self.cap.read()
            if self.frame is not None:
                break
            time.sleep(0.05)

```

```

    if not self.grabbed or self.frame is None:
        raise Exception('Failed to grab first frame from video stream')

    self.stopped = False
    self.worker = Thread(target=self.update_frame, args=(), daemon=True)

def start(self):
    """Start the frame update worker
    Internal method, you normally wouldn't call this yourself.
    """
    self.worker.start()

def update_frame(self):
    """Thread worker function to retrieve frames from a VideoCapture
    Internal method, you normally wouldn't call this yourself.
    """
    while not self.stopped:
        if not self.grabbed or not self.cap.isOpened():
            self.stop()
        else:
            self.grabbed, self.frame = self.cap.read()

def stop(self):
    """Stop the frame update worker
    Internal method, you normally wouldn't call this yourself.
    """
    self.stopped = True
    self.worker.join()

date = datetime.datetime.now().strftime("%Y_%m_%d-%I:%M:%S_%p")
# -----
# -----
# TODO: Fill in code here
tello = Tello()
# Connect to the drone

# Turn on the video stream using "streamon"

# Read the current frame from the camera to the variable readFrame

# Save the image frame to file "picture_{date}.png"

# Turn off the video stream using "streamoff"
tello.streamoff()
# TODO_END: end of added code
# -----
# -----

```

### **C.3 Record Video File**



```

"""Library for interacting with DJI Ryze Tello drones.
"""
import logging
import socket
import time
from threading import Thread
from typing import Optional, Union, Type, Dict
import datetime

import cv2

threads_initialized = False
drones: Optional[dict] = {}
client_socket: socket.socket

# @enforce_types
class Tello:
    """Python wrapper to interact with the Ryze Tello drone using the official
    Tello api.
    Tello API documentation:
    [1.3](https://dl-cdn.ryzerobotics
    .com/downloads/tello/20180910/Tello%20SDK%20Documentation%20EN_1.3.pdf),
    [2.0 with EDU-only
    commands](https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%20
    .0%20User%20Guide.pdf)
    """
    # Send and receive commands, client socket
    RESPONSE_TIMEOUT = 7 # in seconds
    TAKEOFF_TIMEOUT = 20 # in seconds
    FRAME_GRAB_TIMEOUT = 3
    TIME_BTW_COMMANDS = 0.1 # in seconds
    TIME_BTW_RC_CONTROL_COMMANDS = 0.001 # in seconds
    RETRY_COUNT = 3 # number of retries after a failed command
    TELLO_IP = '192.168.10.1' # Tello IP address

    # Video stream, server socket
    VS_UDP_IP = '0.0.0.0'
    VS_UDP_PORT = 11111

    CONTROL_UDP_PORT = 8889
    STATE_UDP_PORT = 8890

    # Set up logger
    HANDLER = logging.StreamHandler()
    FORMATTER = logging.Formatter('[%(levelname)s] %(filename)s - %(lineno)d -
    %(message)s')
    HANDLER.setFormatter(FORMATTER)

    LOGGER = logging.getLogger('djitelloy')
    LOGGER.addHandler(HANDLER)

```

```

LOGGER.setLevel(logging.INFO)
# Use Tello.LOGGER.setLevel(logging.<LEVEL>) in YOUR CODE
# to only receive logs of the desired level and higher

# Conversion functions for state protocol fields
INT_STATE_FIELDS = (
    # Tello EDU with mission pads enabled only
    'mid', 'x', 'y', 'z',
    # 'mpry': (custom format 'x,y,z')
    # Common entries
    'pitch', 'roll', 'yaw',
    'vgx', 'vgy', 'vgz',
    'templ', 'temph',
    'tof', 'h', 'bat', 'time'
)
FLOAT_STATE_FIELDS = ('baro', 'agx', 'agy', 'agz')

state_field_converters: Dict[str, Union[Type[int], Type[float]]]
state_field_converters = {key : int for key in INT_STATE_FIELDS}
state_field_converters.update({key : float for key in FLOAT_STATE_FIELDS})

# VideoCapture object
cap: Optional[cv2.VideoCapture] = None
background_frame_read: Optional['BackgroundFrameRead'] = None

stream_on = False
is_flying = False

def __init__(self,
             host=TELLO_IP,
             retry_count=RETRY_COUNT):

    global threads_initialized, client_socket, drones

    self.address = (host, Tello.CONTROL_UDP_PORT)
    self.stream_on = False
    self.retry_count = retry_count
    self.last_received_command_timestamp = time.time()
    self.last_rc_control_timestamp = time.time()

    if not threads_initialized:
        # Run Tello command responses UDP receiver on background
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        client_socket.bind(('', Tello.CONTROL_UDP_PORT))
        response_receiver_thread =
            Thread(target=Tello.udp_response_receiver)
        response_receiver_thread.daemon = True
        response_receiver_thread.start()

        # Run state UDP receiver on background
        state_receiver_thread = Thread(target=Tello.udp_state_receiver)

```

```

        state_receiver_thread.daemon = True
        state_receiver_thread.start()

        threads_initialized = True

drones[host] = {'responses': [], 'state': {}}

self.LOGGER.info("Tello instance was initialized. Host: '{}'. Port:
'{}'.".format(host, Tello.CONTROL_UDP_PORT))

def get_own_udp_object(self):
    """Get own object from the global drones dict. This object is filled
    with responses and state information by the receiver threads.
    Internal method, you normally wouldn't call this yourself.
    """
    global drones

    host = self.address[0]
    return drones[host]

@staticmethod
def udp_response_receiver():
    """Setup drone UDP receiver. This method listens for responses of
    Tello.
    Must be run from a background thread in order to not block the main
    thread.
    Internal method, you normally wouldn't call this yourself.
    """
    while True:
        try:
            data, address = client_socket.recvfrom(1024)

            address = address[0]
            Tello.LOGGER.debug('Data received from {} at
            client_socket'.format(address))

            if address not in drones:
                continue

            drones[address]['responses'].append(data)

        except Exception as e:
            Tello.LOGGER.error(e)
            break

@staticmethod
def udp_state_receiver():
    """Setup state UDP receiver. This method listens for state information
    from
    Tello. Must be run from a background thread in order to not block
    the main thread.

```

```

Internal method, you normally wouldn't call this yourself.
"""
state_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
state_socket.bind(("", Tello.STATE_UDP_PORT))

while True:
    try:
        data, address = state_socket.recvfrom(1024)

        address = address[0]
        Tello.LOGGER.debug('Data received from {} at
state_socket'.format(address))

        if address not in drones:
            continue

        data = data.decode('ASCII')
        drones[address]['state'] = Tello.parse_state(data)

    except Exception as e:
        Tello.LOGGER.error(e)
        break

@staticmethod
def parse_state(state: str) -> Dict[str, Union[int, float, str]]:
    """Parse a state line to a dictionary
    Internal method, you normally wouldn't call this yourself.
    """
    state = state.strip()
    Tello.LOGGER.debug('Raw state data: {}'.format(state))

    if state == 'ok':
        return {}

    state_dict = {}
    for field in state.split(';'):
        split = field.split(':')
        if len(split) < 2:
            continue

        key = split[0]
        value: Union[int, float, str] = split[1]

        if key in Tello.state_field_converters:
            num_type = Tello.state_field_converters[key]
            try:
                value = num_type(value)
            except ValueError as e:
                Tello.LOGGER.debug('Error parsing state value for {}: {}
to {}'.format(key, value, num_type))

```

```

        Tello.LOGGER.error(e)
        continue

    state_dict[key] = value

return state_dict

def get_current_state(self) -> dict:
    """Call this function to attain the state of the Tello. Returns a dict
    with all fields.
    Internal method, you normally wouldn't call this yourself.
    """
    return self.get_own_udp_object()['state']

def get_state_field(self, key: str):
    """Get a specific state field by name.
    Internal method, you normally wouldn't call this yourself.
    """
    state = self.get_current_state()

    if key in state:
        return state[key]
    else:
        raise Exception('Could not get state property: {}'.format(key))

def get_udp_video_address(self) -> str:
    """Internal method, you normally wouldn't call this yourself.
    """
    address_schema = 'udp://@{ip}:{port}' # +
        '?overrun_nonfatal=1&fifo_size=5000'
    address = address_schema.format(ip=self.VS_UDP_IP,
        port=self.VS_UDP_PORT)
    return address

def get_video_capture(self):
    """Get the VideoCapture object from the camera drone.
    Users usually want to use get_frame_read instead.
    Returns:
        VideoCapture
    """

    if self.cap is None:
        self.cap = cv2.VideoCapture(self.get_udp_video_address())

    if not self.cap.isOpened():
        self.cap.open(self.get_udp_video_address())

    return self.cap

def get_frame_read(self) -> 'BackgroundFrameRead':

```

```
"""Get the BackgroundFrameRead object from the camera drone. Then, you
just need to call
backgroundFrameRead.frame to get the actual frame received by the
drone.
```

```
Returns:
```

```
BackgroundFrameRead
```

```
"""
```

```
if self.background_frame_read is None:
    address = self.get_udp_video_address()
    self.background_frame_read = BackgroundFrameRead(self, address) #
    also sets self.cap
    self.background_frame_read.start()
return self.background_frame_read
```

```
def send_command_with_return(self, command: str, timeout: int =
RESPONSE_TIMEOUT) -> str:
```

```
"""Send command to Tello and wait for its response.
Internal method, you normally wouldn't call this yourself.
```

```
Return:
```

```
bool/str: str with response text on success, False when
unsuccessful.
```

```
"""
```

```
# Commands very consecutive makes the drone not respond to them.
```

```
# So wait at least self.TIME_BTW_COMMANDS seconds
```

```
diff = time.time() - self.last_received_command_timestamp
```

```
if diff < self.TIME_BTW_COMMANDS:
```

```
    self.LOGGER.debug('Waiting {} seconds to execute command:
    {...}'.format(diff, command))
    time.sleep(diff)
```

```
self.LOGGER.info("Send command: '{}'.format(command))
```

```
timestamp = time.time()
```

```
client_socket.sendto(command.encode('utf-8'), self.address)
```

```
responses = self.get_own_udp_object()['responses']
```

```
while not responses:
```

```
    if time.time() - timestamp > timeout:
        message = "Aborting command '{}'. Did not receive a response
        after {} seconds".format(command, timeout)
        self.LOGGER.warning(message)
        return message
    time.sleep(0.1) # Sleep during send command
```

```
self.last_received_command_timestamp = time.time()
```

```
first_response = responses.pop(0) # first datum from socket
```

```
try:
```

```
    response = first_response.decode("utf-8")
```

```
except UnicodeDecodeError as e:
```

```

        self.LOGGER.error(e)
        return "response decode error"
response = response.rstrip("\r\n")

self.LOGGER.info("Response {}: '{}'.format(command, response))
return response

def send_command_without_return(self, command: str):
    """Send command to Tello without expecting a response.
    Internal method, you normally wouldn't call this yourself.
    """
    # Commands very consecutive makes the drone not respond to them. So
    # wait at least self.TIME_BTW_COMMANDS seconds

    self.LOGGER.info("Send command (no response expected):
    '{}'.format(command))
    client_socket.sendto(command.encode('utf-8'), self.address)

def send_control_command(self, command: str, timeout: int =
RESPONSE_TIMEOUT) -> bool:
    """Send control command to Tello and wait for its response.
    Internal method, you normally wouldn't call this yourself.
    """
    response = "max retries exceeded"
    for i in range(0, self.retry_count):
        response = self.send_command_with_return(command, timeout=timeout)

        if response.lower() == 'ok':
            return True

        self.LOGGER.debug("Command attempt #{} failed for command:
        '{}'.format(i, command))

    self.raise_result_error(command, response)
    return False # never reached

def connect(self, wait_for_state=True):
    """Enter SDK mode. Call this before any of the control functions.
    """
    self.send_control_command("command")

    if wait_for_state:
        REPS = 20
        for i in range(REPS):
            if self.get_current_state():
                t = i / REPS # in seconds
                Tello.LOGGER.debug("''.connect()' received first state
                packet after {} seconds".format(t))
                break
            time.sleep(1 / REPS)

```

```

        if not self.get_current_state():
            raise Exception('Did not receive a state packet from the
                Tello')

def streamon(self):
    """Turn on video streaming. Use `tello.get_frame_read` afterwards.
    Video Streaming is supported on all tellos when in AP mode (i.e.
    when your computer is connected to Tello-XXXXXX WiFi network).
    Currently Tello EDUs do not support video streaming while connected
    to a WiFi-network.

    !!! Note:
        If the response is 'Unknown command' you have to update the Tello
        firmware. This can be done using the official Tello app.
    """
    self.send_control_command("streamon")
    self.stream_on = True

def streamoff(self):
    """Turn off video streaming.
    """
    self.send_control_command("streamoff")
    self.stream_on = False

class BackgroundFrameRead:
    """
    This class read frames from a VideoCapture in background. Use
    backgroundFrameRead.frame to get the current frame.
    """

    def __init__(self, tello, address):
        tello.cap = cv2.VideoCapture(address)

        self.cap = tello.cap

        if not self.cap.isOpened():
            self.cap.open(address)

        # Try grabbing a frame multiple times
        # According to issue #90 the decoder might need some time
        #
        # https://github
        # .com/damiafuentes/DJITelloPy/issues/90#issuecomment-855458905
        start = time.time()
        while time.time() - start < Tello.FRAME_GRAB_TIMEOUT:
            Tello.LOGGER.debug('trying to grab a frame...')
            self.grabbed, self.frame = self.cap.read()
            if self.frame is not None:
                break

```



```

        time.sleep(0.05)

    if not self.grabbed or self.frame is None:
        raise Exception('Failed to grab first frame from video stream')

    self.stopped = False
    self.worker = Thread(target=self.update_frame, args=(), daemon=True)

def start(self):
    """Start the frame update worker
    Internal method, you normally wouldn't call this yourself.
    """
    self.worker.start()

def update_frame(self):
    """Thread worker function to retrieve frames from a VideoCapture
    Internal method, you normally wouldn't call this yourself.
    """
    while not self.stopped:
        if not self.grabbed or not self.cap.isOpened():
            self.stop()
        else:
            self.grabbed, self.frame = self.cap.read()

def stop(self):
    """Stop the frame update worker
    Internal method, you normally wouldn't call this yourself.
    """
    self.stopped = True
    self.worker.join()

def RecordVideo():
    global readFrame, record
    print("Recording a Video")
    height, width, _ = readFrame.frame.shape
    date = datetime.datetime.now().strftime("%Y_%m_%d-%I:%M:%S_%p")
    video = cv2.VideoWriter(f'video_{date}.avi',
        cv2.VideoWriter_fourcc(*'XVID'), 30, (width, height))
    while record:
        video.write(readFrame.frame)
        time.sleep(1 / 30)
    video.release()

# -----
# TODO: Fill in code here
tello = Tello()
# Connect to the drone

# Set the record variable to true

```

```
# Turn on the video stream using "streamon"

# Read the current frame from the camera to the variable readFrame

# Run the recorder on another thread with target of the RecordVideo function
  above

# Start the recorder

# Set the amount of time you would like to record using the time.sleep function

# Set the record variable to false

# Join the frames collected by the recorder

# Turn off the video stream using "streamoff"
tello.streamoff()
# TODO_END: end of added code
#-----
-----
```