RaspyLab: A low-cost remote laboratory to learn programming and physical computing through Python and Raspberry Pi

Authors Here

***Abstract*— This article describes the development and assessment of *RaspyLab* which is a low-cost Remote Laboratory (RL) to learn and teach programming with Raspberry Pi and Python language. The RL is composed of 16 stations or nodes that contain hardware components such as display LCD, robotic arm, temperature sensor, among others, and two modes of programming (graphical and text-based) for the students to experiment with their designed algorithms. The concept of the RL was conceived as a pedagogical tool to support the students of Engineering and Computer Science (CS) in an online learning format, given the context of the COVID-19 pandemic. The laboratory has been used by (*n=30*) CS students during the second semester of 2020 in the subject of mathematical logic through the methodology of Problem-Based Learning (PBL). To evaluate preliminary the laboratory, it was used a survey with 3 open-ended questions and 12 closed-ended questions on a Likert scale according to the Technology Acceptance Model (TAM). The outcomes show a good reception of the laboratory, an enhancement of the students' learning regarding the concepts addressed in the course, and an interest of the students for the laboratory to be included in other subjects of the curricula.**

***Index Terms*— Remote laboratory, Problem-Based Learning (PBL), physical computing, programming, Raspberry Pi, Python.**

I. Introduction

REMOTE Laboratories (RLs) have served to support teaching and learning activities during the last two decades. Aspects such as the increase of the students in the classrooms, the reduction of the costs regarding the infrastructure in laboratories, and the need for the modernization and flexibilization of the curricula have led to the popularization of these tools in engineering and CS education [1]. Nowadays, the COVID-19 pandemic and its lockdown periods have meant the reduction or elimination of the in-person classes that have been replaced by blended or virtual learning methodologies in a short period of time [2], [3]. This fact has strengthened the usage of RLs in higher education as a temporal solution to provide experimentation and quality education to the students. Nonetheless, not all universities or schools have counted with an important budget and resources to generate this quick transition, which generates a lack of competencies, skills, and motivation to learn in the students. If the problem continues, the students can get high failures or even drop out of their programs, aggravating the educational effects produced by the pandemic. Besides, from the pedagogical perspective, there are a set of questions about the pertinence and impact of the online classes in the educational process of the students.

Given these elements, this article describes the development and implementation of *RaspyLab* available at [4], which is a low-cost remote laboratory to learn and teach programming with Raspberry Pi and Python programming language. The concept of the RL has been thought not only as a technical tool but also as a pedagogical instrument that serves instructors or teachers to create methodologies that enhance learning in the students. The RL counts with two programming modes and 16 working stations or nodes. In the first programming mode, the students can utilize graphical blocks (Algorithm Visualizations AVs) that have been designed to be interfaced with the General-Purpose I/Os (GPIOs), peripherals, and hardware devices of each node. In the second programming mode, the students can program directly in Python language. These modes provide support to students who are novices and intermediate in programming.

The RL has been assessed with 30 CS students of the first year in the subject of mathematical logic during the second semester of 2020 and their opinions have been collected with a survey according to the TAM model [5], [6]. The students designed several algorithms collaboratively in groups of two and three people with the described modes of programming through the educational methodology of Problem-Based Learning (PBL). The results show mainly the following points: (1) A good reception and acceptance of the RL by the students. (2) An increase in the motivation and interest to learn the concepts addressed in the course. And (3) an intention to use the RL by the students in the future in free spaces of time, or in other subjects of the curricula.

Our interest in the RL is twofold. On one hand to provide a quality technological tool to benefit the learning of the students concerning programming and physical computing [7].

Manuscrito recibido el día de mes de año; revisado día de mes de año; aceptado día de mes de año. English version received Month, day-th, year. Revised Month, day-th, year. Accepted Month, day-th, year.

F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (corresponding author to provide phone: 303-555-5555; fax: 303-555-5555; e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, (e-mail: author@nrim.go.jp).

There exists a Spanish version of this article available at XXXX DOI (Digital Object Identifier) Pendiente

PREPRINT

On the other hand, in the study, we proposed a low-cost and quick architecture for RLs that could be adapted by the researchers or educators in function of their interests in engineering and CS education. The architecture only needs a service of Virtual Private Server (VPS) in Linux with some Python packages, Raspberry Pis, hardware devices such as sensors, displays LCD, or low-cost robots made in 3D printers to operate.

While several remote laboratories are focused on robotics, control systems, Internet of Things (IoT), or communications networks, there exists a current lack of laboratories to teach and learn programming and physical computing even from scratch, gathering both graphical and text-based modes. Similarly, the cost of the infrastructure for remote laboratories can be a constraint for their deployment in higher education settings. With the low-cost architecture proposed in this study, we try to address this matter. These two aspects framing the main contributions of this work.

The remainder of the paper is organized as follows. Section 2 exposes the background of this study. Section 3 describes the design and development of the RL from the perspectives of software and hardware. Section 4 depicts the educational methodology, participants, and instruments adopted for this study. Section 5 shows the results of the study and discusses the educational implications of the RL. Finally, Section 6 outlines the conclusions of this study.

II. Background

This section addresses related works and the key concept in the educational foundations of the RL which is physical computing.

A. Related works

Concerning the developments of Remote Laboratories (RLs) for teaching and learning programming, de Lima et al. [8] describe a RL that integrates block and text-based coding to teach introductory programming. The proposal was evaluated for 18 high school students and 5 teachers. The RL had a good acceptance, and the students were satisfied with the Graphical User Interface (GUI) and the hardware devices to interact. In [9], the authors present a RL with Python to learn to program in CS1 and CS2 courses. 77 students were enrolled in the methodology of the study and the authors indicate that the interactive experiments with the RL offered a high-quality learning experience that motivated the students. Similarly, in [10] is presented a RL called *Block.ino* focused on Arduino for learning purposes in programming. The RL was employed by 144 students in the levels of high school and higher education. The authors highlight the feasibility, and scalability of the platform due to the usage of Free and Open Source Software (FOSS).

Besides, the researchers have utilized robotics in the design of the RLs to foster learning and comprehension of the students. For instance, robotics can reduce the gender gap regarding programming skills in introductory courses as well as it has demonstrated to be preferable for women in these ones [11], [12]. In this line, the studies [13], [14] show proposals about RLs that incorporate robotics. In the first study, the researchers describe a Web App oriented towards autodidactic learning of programming through AVs. The study was carried out with 46 students between 11 to 14 years old. A preliminary survey with 20 questions was provided to the students to characterize their knowledge in programming. Four modules were proposed with the usage of the RL, and the authors indicate that the students considered the RL interesting to learn programming in an autodidactic way. Moreover, the authors report an improvement in the students' skills in each module. The second study proposes an educational robot with a mobile platform for STEM areas. The robot is composed of a microcontroller, motors, battery, and sensors. The authors discuss the technical components of the platform and the advantages for learning in the aspects of improvement of programming skills and problem-solving.

As for proposals that include AVs for introductory programming, the studies [15], [16] illustrate how these kinds of tools can influence learning and motivation in the students. The first study explores how the AVs can improve learning and grasping in female students in secondary school. The study was carried out with 24 German female students in two workshops, utilizing two AVs tools (*mBlock* and a Web-based Programming Application (WPA)). To analyze the data, it was performed a study with inferential statistics to find significant differences in programming abilities and motivation with both AVs tools. The results indicated that the students found enjoyable programming with the tool (*mBlock*) that provided a higher level of ease-of-use of programming. The authors state that the Scratch-based design is a productive environment for female students with no prior programming experience. The second study describes the employment of Scratch to teach introductory programming in parallel with the classical method in the course COMPE112 (Programming computer in C language). Firstly, the results were compared in 2010 and 2011 in terms of enrolled students vs. failed students with the methodology. Secondly, 55 students filled-out a questionary in a Likert Scale where its analysis showed that Scratch makes programming more enjoyable and visual, helping to understand the concepts in this area.

These studies evidence that the assumed approach in this work is coherent from the educational perspective in the selection of elements such as the programming modes, the usage of hardware devices that have been included in the design and deployment. The next sections discuss these components in detail.

B. Physical computing

Traditionally, the courses of programming have been addressed with an important load of theoretical components instead of hands-on activities that help to improve learning and grasping in the students. This trend has been modified by the gradual incorporation of physical computing as an important cornerstone concept in the curricula of engineering and CS. Physical Computing (PhyC) is not a new concept in CS, it has been in the educational arena for almost twenty years. Hodges et al. [7] define PhyC as a combination of

PREPRINT

hardware and software to build physical systems that sense and interact with the real world. Greenwold [17] states that PhyC is a type of human interaction with machines in which these manipulate real objects and spaces. O'Sullivan and Igoe [18], maybe the first to coin this term, indicate that PhyC is an interaction, where the term interaction is understood as "an iterative process of listening, thinking, and speaking between two or more actors". PhyC is essentially an activity with computers to manipulate objects in the real world. This interaction is mainly created by transductors which are elements that transform the sensed variables to an electric equivalent in voltage or current. Moreover, from a constructivist point of view, PhyC benefits the cognitive, perceptual, and social skills of the students [7].

The educational tenets of the PhyC are mainly based on constructivism. Educators have felt the need to make the curricula of engineering and CS more inclusive and interesting for the students. Aspects such as high failure and the drop out of the courses in programming have required methodologies and curricula to integrate more actively with the students. Constructivism starts from the tenet that knowledge and learning are created actively by the students in a process of interaction [19], for example, with the technology, peers, and teachers. Between the advantages to include PhyC in the curricula are highlighted the increasing motivation and self-confidence, fostering creativity, collaboration, inclusion, learning by doing, and engagement [7], [20]. The approach of the PhyC has allowed the incorporation of robotics in the courses of programming to improve learning in the students. As described in the section of related works, robotics has contributed to reducing the gender gap in programming skills, and it is preferable by women in these courses. In addition, students learn better when they can construct their robots and they can program them [21]. In this process, students can transform black-box designs which are preconfigured with restricted functions into white-box designs that can be modified, repaired, and improved according to the learning requirements. Also, PhyC is suitable to be integrated with educational methodologies as PBL where the work is oriented to problem-solving in small groups with the guidance of the tutor as a facilitator in the learning process [22].

III. Design and Development of the Remote Laboratory *RaspyLab*

Given the conditions for the design of the RL, the following generalities both technical and educational were considered:

1. Selecting a quick software architecture that provides access to the different services to the actors in the laboratory: students, teachers, and administrative staff. The architecture must be installed in a Linux server.
2. Initially, creating 16 nodes or working stations for the students. Each station must count with a Raspberry Pi 4 (2GB), a camera (8MP or 5MP) and the different hardware devices to interact with the algorithms. With the installed capacity, between 30 and 40 students can perform the different activities proposed by the teacher or instructor in a course.
3. Providing a user interface for the students. The interface must be accessed through https protocol in the browsers that support the video codecs for the camera protocols.
4. For the students' interface, two modes of programming were created. The first one with graphical blocks (block-based) for novice programmers and the second one in a text-based mode for intermediate programmers.
5. The software architecture must employ Free and Open-Source Software (FOSS).

With the previous requirements, it was created the software architecture and the hardware setup for the laboratory. Each one of these components will be explained in the next subsections.

A. Software architecture

For the software architecture, it was chosen a VPS in Linux Ubuntu due to constraints in budget and time in the project. With the VPS, we guarantee consistency and robustness in the laboratory service and a centralized platform for the webmaster and the programmers without the need of a physical server. Also, in the VPS were used some Python packages for the functions of user's access, processing of the students' code, linking with the relational databases and communication with the Raspberry Pi in each node. We selected these packages to accelerate the process of construction of the laboratory, providing flexibility and scalability to include additional laboratories in future studies. Fig.2 shows the overall architecture for the RL.

1) Student Interface

For the student interface, it was deployed an Apache server in the VPS. The Apache server hosts the web pages for the students' access and the type of selected programming in the modes of graphical blocks or text-based. Fig.1 illustrates the user interface for the blocks-based mode of programming.

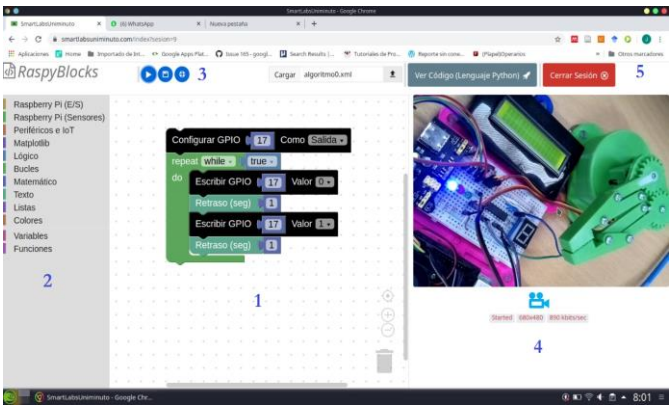


Fig. 1. Student interface for the option of blocks-based programming. 1. Working area. 2. Blocks' palette. 3. Tool bar with the options of execution, save, and upload. 4. Video frame. 5. Buttons to show code in Python and close session.

PREPRINT

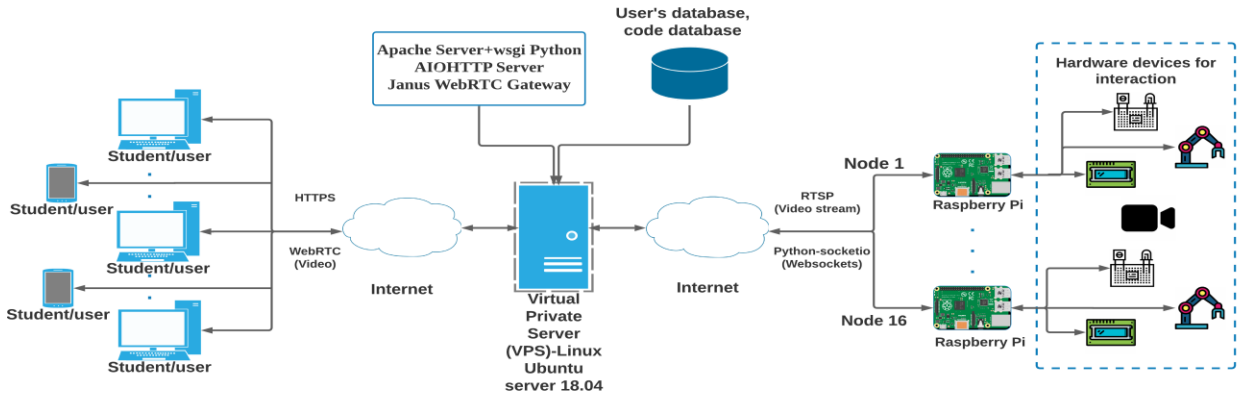
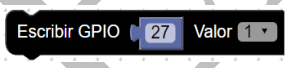
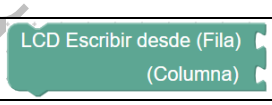
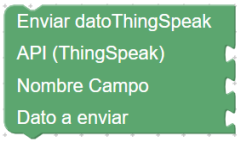
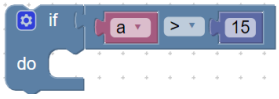


Fig. 2. Network architecture proposed for the RL *RaspyLab*.

Regarding the blocks-based mode, we modified the core of the Google tool *Blockly* [23], which is recognized in education as a web-engine Algorithm Visualization (AV) tool. *Blockly* transforms each block towards a respective equivalent, in this case, in Python programming language according to the requirements of the Raspberry Pi. The students have the options of execute, save, upload, and see the respective code in Python language for their AVs. Likewise, we have developed several categories of blocks to handle the different hardware devices, General Purposes I/O (GPIOs) and peripherals in the Raspberry Pi. Among the most relevant categories are the following: *Raspberry Pi GPIOs, delays LCD, peripherals, servomotor, Internet of Things (IoT), logic, and loops*. Table I shows a description of these categories.

TABLE I.
Description of the main categories for the graphical programming mode.

Category	Description	Block Example
Raspberry Pi (I/O)	Blocks to configure (Input, Output, and Pulse Width Modulation (PWM)), GPIO writing, GPIO reading, and delays in secs.	
Sensors	Blocks to read sensors and Analog to Digital Converter (ADC) such as (TMP102, HDC1008, ADS1115, INA219, TSL2591).	
LCD	Blocks to handle a Liquid Crystal Display (LCD) (2*16 characters).	
Peripherals and IoT	Blocks for Universal Asynchronous Receiver-Transmitter (UART) and Internet of Things (IoT) through MATLAB *ThingSpeak* [24].	
Logic	Blocks for conditional and logic statements.	
Loops	Blocks for loops (while and for).	

As mentioned, the graphical blocks were selected for novice students or learners who want to learn from scratch. Several studies [25], [26] in the field of CS have demonstrated that the AVs can enhance the learning process of the students in algorithmic creation and understanding of the processes and concepts immerse in them. Shaffer et al. [27] argue that the AVs could be used for the instructor as part of a lecture, a laboratory, or an assignment to teach a concept. Nonetheless, some researchers and educators have found that AVs can serve as a "gateway" to learn text-based languages [28]. Given these points, we also decided to employ the option of text-based programming to the students contrast their graphical algorithms or designs directly in this mode. We installed the different libraries in Python language for the Raspberry Pi in each node to provide the functionality of the hardware devices for both modes of algorithm creation.

In the same line of the graphical mode, we developed a console for the students to directly write the code in Python. Some studies in CS have shown that Python helps to reduce the student drop out and it improves understanding and interest in introductory programming [29], [30]. The interface is basically the same that in the graphical mode only with a change of editor in the working area. In the interface, it was utilized the code editor *ACE* available at (https://ace.c9.io/) to highlight the different statements in the programming language to the students recognize them. Fig.3 depicts the interface for this mode. Concerning the laboratory access, it was employed the Python packages *Flask* and *Flask logging* [31] to manage the user login and the actions to take when a student sends a code for a specific node. Afterwards, the student selects a node (1 to 16) and the type of programming that desires. Fig.4 illustrates an example of access to the laboratory and Fig.5 depicts the overall diagram for the events of login and logout of a student.

The student or user must log in with an accredited email and password. Personal data of the students are saved in a restricted database in the VPS. Data are provided by the university in terms of students' names, emails, and identification code (ID), and this information is hosted in the mentioned database. Besides, the Python package *Flask logging* manages the access to the students and limit the usage of the sessions by non-permitted users.

PREPRINT

2) Video Streaming

One of the critical elements in the laboratory construction was the video streaming. Each node counts with a Raspberry Pi Camera that shows the functioning of the hardware elements. To test the latency of the video streaming for the users or students, we used the servers and gateways described in Table II. The video is encoded by the *FFmpeg* tool [32] in the Raspberry Pi and sent through Real-Time Streaming Protocol (RTSP) or User Datagram Protocol (UDP) for further processing in the VPS.

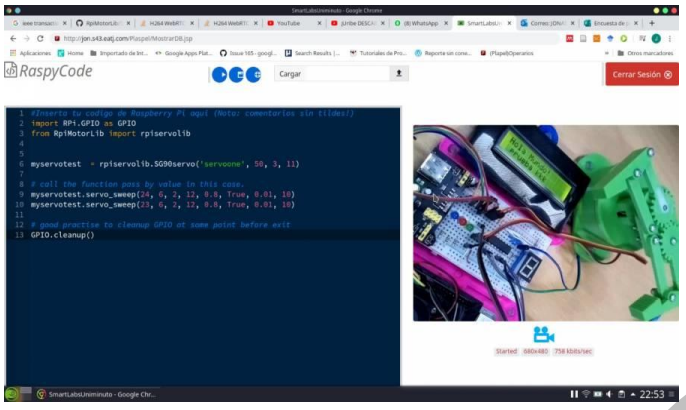


Fig. 3. Student interface for the text-based programming option.

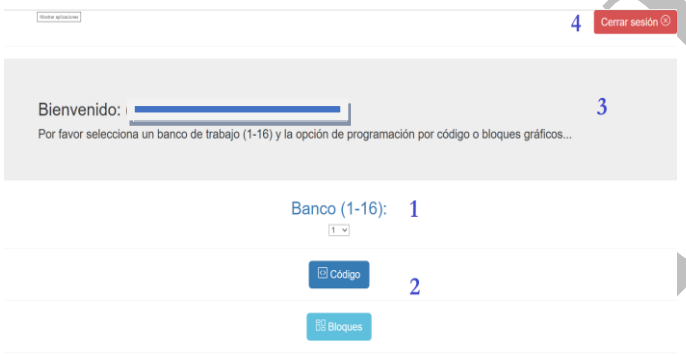


Fig. 4. Options of access for a student in the RL. 1. Node selection. 2. Programming options (blocks-based and text-based). 3. Welcome message. 4. Button to close session.

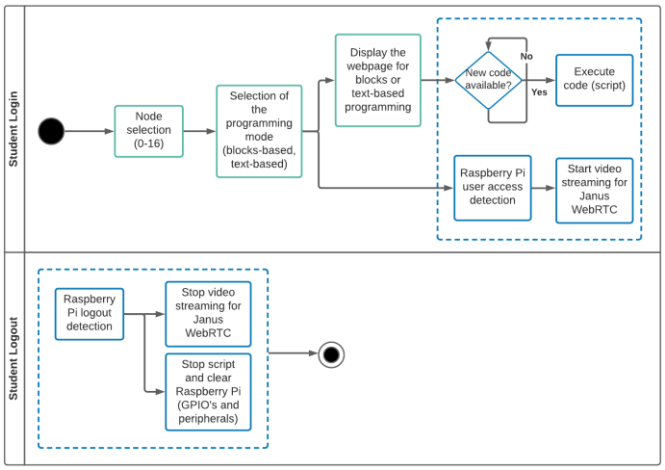


Fig. 5. Diagram for the student's access in the events of login and logout. In blue (functions executed in the Raspberry Pi), green (functions executed in the VPS).

Although the latency is injected by the client, it represented a critical issue for the efficiency and robustness of the laboratory. The video is on-demand which means that in function of the user and the selected node, the video is launched. With this method, we guarantee a low bandwidth consumption and a reduction of RAM and processor usage in the VPS.

TABLE II.
Video streaming servers and gateways tested for the RL.

Server or Gateway	Description	Experimented Latency
Nginx + HLS protocol	A web server, reverse proxy, load balancer, and HTTP cache.	10 to 20 secs
Node-Media-Server + HTTP FLV	A Node.js implementation of RTMP/HTTP-FLV/WS-FLV/HLS/DASH Media Server	5 to 12 secs
Janus WebRTC Gateway	A WebRTC Gateway for general purposes.	0.5 to 2 secs

Regarding Table II, the first test was performed with a *Nginx* Server with HLS protocol [33] which generated a latency that oscillated between 10 to 20 secs. Due to this high latency for the students' interaction, we tried other alternatives until the selection of Janus WebRTC gateway [34], [35]. Janus transforms the encoded stream of the Raspberry Pi camera in UDP protocol to WebRTC standard for the users. As a concept, Web-based Real-Time Communication (WebRTC) is a relatively new standard for real-time peer-to-peer communications that has several advantages for gaming, video streaming and sensor data feeds [36], [37]. WebRTC is fully compatible with browsers such as Chrome and Mozilla Firefox. However, WebRTC uses the UDP protocol which can yield to breaks and freeze in the video transmission for several milliseconds.

3) Raspberry Pi server connection

Each Raspberry Pi connects with the VPS performing a long-polling request. Due to restrictions in the network where the nodes are located, we cannot access directly to each Raspberry Pi. Instead, the Raspberry Pi makes a long-polling request through WebSockets employing the asynchronous server (AIOHTTP) [38]. When a student sends his code, this is temporally saved in a small database for further processing. This event is detected for the Raspberry Pi that extracts the code (script) to execute. All these events are synchronized with the login and logout actions that one user performs in the RL according to Fig.5. Throughout the process, a watchdog timer monitors the code execution in the Raspberry Pi, if there is a problem in the execution or the Raspberry freezes, the watchdog restarts the Raspberry Pi and the service connection with the VPS.

B. Hardware setup

For the hardware setup, it was used the components depicted in Table III. These components allow the students to make physical computing, experimenting in real-time with their algorithms. In the hardware, we employed the single-board computer Raspberry Pi 4. There are some reasons for

this selection. First, this version provides several options of Random-Access Memory (RAM) (2GB and 4GB) that are superior to its predecessor the Raspberry Pi model 3. These options of memory are more suitable to provide high efficiency for the requirements of video and processing that demand the remote laboratory. Second, with these memory options and features, for instance, in processor (Quad-Core Cortex A-72 @ 1,5 GHz), GPIOs and protocols, we guarantee scalability for further developments, e.g., in image processing, robotics, control systems, among others.

TABLE III.
Hardware components employed in each node of the RL.

Component	Description
Raspberry Pi 4 (2GB)	Single-Board Computer Raspberry Pi version 4 with 2GB of RAM.
Camera	Raspberry Pi Camera (8MP or 5MP).
LCD 2*16 Alphanumeric	Liquid Crystal Display (2*16 characters).
7-segment display	7-segment numeric display.
HDC1008	High-accuracy digital humidity sensor with temperature sensor.
Protoboard	Protoboard with 800 points.
Robotic Arm	3D ensembled robotic arm with two degrees of freedom.
LEDs	LEDs for GPIO usage.
DC Adapter	Adapter 5V, 3A for the Raspberry Pi.
SD Card	Micro SD Card (32GB).
Camera support	Folding camera support for Raspberry Pi made in a 3D printer.

We accelerated and reduced the costs in the process of hardware construction designing several components such as cases, robotic arm, and camera support with a 3D printer as it is illustrated in Fig.6. With this hardware setup, the instructor or teacher can create different types of educational methodologies that could be adapted to the learning needs of the students. Thus, students can understand the concepts, e.g., in programming and basic robotics. The total cost of the hardware components per node in Table III is U$111.

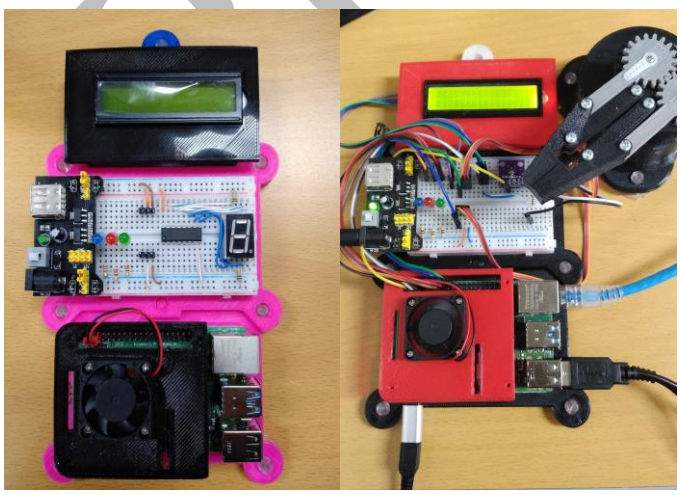


Fig. 6. Example of the hardware setup for the RL.

IV. EDUCATIONAL METHODOLOGY

Educational foundations of the RL are based on constructivist learning theory [19], [39] and physical computing. In the constructivism, learning and knowledge are constructed actively by the students through interactions with their peers, but also with the technology. In this interaction with hands-on activities, the students construct their algorithms to manipulate artifacts which help to enhance the comprehension of concepts in order to create abstractions and generalizations [21]. Nonetheless, the mediation between technology and learning requires time to be recognized as an important part of the education process of the students, in this case, in programming and algorithmic thinking. For this reason, the results presented in this study are preliminary. Taking these elements into account, Fig.7 shows the methodology with the usage of the RL which has been elaborated with Bloom's taxonomy in the cognitive learning domain [40]. The methodology has five steps which will be explained as follows:

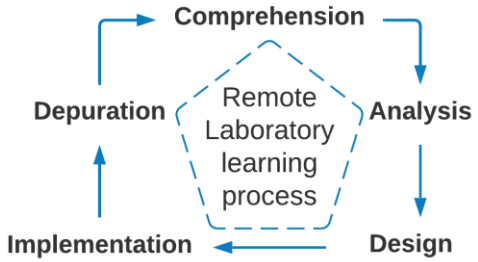


Fig. 7. Proposed scheme of learning with the RL.

1. *Comprehension*: Students understand the concepts addressed in the course. Initially, the students identify the uses of these concepts in problem-solving within real situations or case studies. The instructor can use synchronous or asynchronous methodologies for learning such as videos, online lectures, tutorials, etc.
2. *Analysis*: The teacher or instructor proposes a problem that gathers knowledge and skills acquired by the students. Subsequently, the students analyze how to develop a solution utilizing these elements collaboratively.
3. *Design*: The students start the design of their algorithm for the problem situation. The algorithm could be represented

PREPRINT

in terms of a pseudo code, a flow diagram, etc. Also, the students understand the constraints and limitations of the design.

4. *Implementation*: With the design, the students implement their algorithm in the RL selecting the most suitable programming mode according to the learning needs and curricula requirements.
5. *Depuration*: The students evaluate the problems, errors, or possibilities to improve their algorithms with the RL. The students fix these errors, and reimplement the designs in the RL.

These steps were assumed in the activities of the course of mathematical logic which is offered to first semester students of CS. The selected educational approach in the course was Problem-Based Learning (PBL) in where a set of problems were solved by the students collaboratively in groups of two and three people, as mentioned with the help of the RL. The subject encompasses topics in propositional logic, sets, logical operators, and formal languages which the students were encouraged to learn with the RL. The laboratory was utilized by each student during 6 hours per week for 4 weeks. Students utilized both modes of programming, contrasting their graphical algorithms with the syntax in Python language.

A. Participants

30 first semester CS students participated of the methodology and the testing phase of the RL. The mean age was 18.5 years old with a students' distribution of 90% male and 10% female.

B. Instruments

To collect the information about the perceptions of the students, a survey with 12 closed-ended questions in the Likert Scale, and 2 open-ended questions was designed. The range of the closed-ended questions was from *1: Strongly disagree to 4: Strongly agree*. The survey was based on the Technology Acceptance Model (TAM) proposed by Davis et al. [6], [41] to evaluate the use of the RL. The TAM approach is a good predictor of the intentions in using the RL and it has been employed in diverse studies in education [5], [42]–[44]. TAM is determined for three factors and the external stimulus-response, which is fostered by aspects such as motivation, interest, engagement, anxiety, among others. The relationship between these constructs is illustrated in Fig.8.

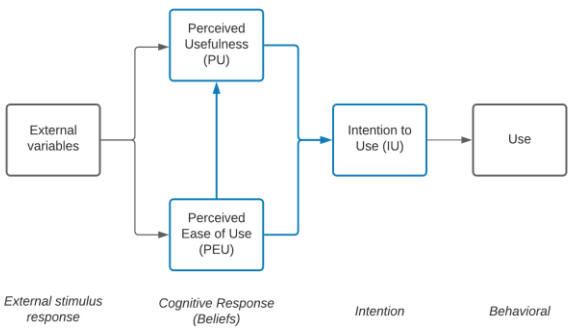


Fig. 8. Technology Acceptance Model (TAM) adapted from [41].

1. *Perceived Usefulness (PU)*: Define the degree in which the students found that the RL contributed to enhancing their learning in the concepts and topics addressed in the course.
2. *Perceived Ease of Use (PEU)*: Indicate the degree in which the students easily used the features and interface of the RL without excessive effort or confusion.
3. *Intention to Use (IU)*: Define the degree in which the students are aware to utilize the RL in the future to learn programming and physical computing.

This model is suitable to evaluate preliminary the RL because it provides insights concerning the perceptions that the students had with the employment of the RL as well as it indicates the acceptance degree of the RL. However, an assessment for a longer time is required to analyze the educational implications of the RL in the students.

Then, 20 students (66.66%) answered the final survey. The data were analyzed with IBM SPSS Statistics v.23. Table IV shows the survey with the respective Cronbach's alpha value for each one of the factors in the TAM. In all factors, the value of the Cronbach's alpha is over 0.7 which demonstrates the reliability of the instrument [45].

TABLE IV.
Survey for the students' perceptions of the RL according to the TAM (*n=20*).

TAM factor	Question	Cronbach's Alpha
PU	**Q1**. Did the RL help me to understand how the topics of the course could be applied in real situations?	**0.782**
	Q2. Did the programming modes in the RL help me to understand and apply the concepts of the course?	
	Q3. With the RL, Was I able to experiment and learn in a self-paced way?	
	Q4. Do I think that the RL is a tool that benefits my learning?	
	Q5. Through the RL, Was I able to contrast the graphical blocks with the respective code in Python language to learn its syntax?	
PEU	**Q6**. Do I consider that the access to the RL was easy?	**0.759**
	Q7. Do I consider that it was easy to learn the RL interface?	
	Q8. Was I able to run my algorithms and observe their functioning with the real-time video?	
	Q9. Do I consider that the RL interface is suitable to create my algorithms?	
	Q10. Do I consider that it is easy to learn the features of the RL?	
IU	**Q11**. Would I like that the RL was included in other subjects of the curriculum in the future?	**0.772**
	Q12. Would I use the remote lab in some of my free spaces to continue experimenting and learning?	

PREPRINT

V. Results and Discussion

This section describes and discusses the experimental results for the methodology. Moreover, it discusses the advantages and limitations regarding the RL. Table V depicts the descriptive statistics for the responses of the students in the survey.

TABLE V.
Mean and Standard Deviation (SD) by question of the survey (*n=20*).

TAM Factor	Question	Mean	SD
PU	Q1	3.6	0.503
	Q2	3.5	0.513
	Q3	3.45	0.510
	Q4	3.75	0.444
	Q5	3.35	0.489
PEU	Q6	3.05	0.759
	Q7	3.4	0.598
	Q8	3.25	0.786
	Q9	3.6	0.502
	Q10	3.5	0.606
IU	Q11	3.75	0.444
	Q12	3.7	0.470

With this information, it is calculated the mean, median, and standard deviation of each factor in the TAM. Table VI shows these values.

TABLE VI.
Mean, median, and Standard Deviation (SD) of the TAM factors (*n=20*).

TAM factor	Mean	Median	SD
PU	3.53	3.4	0.363
PEU	3.36	3.4	0.47
IU	3.725	4	0.41

Based on these values, students have a good Perceived Usefulness (PU) of the RL. The students indicated that the RL helped them to learn and apply the concepts in the course in real-time. Likewise, they argued the importance of this kind of laboratory despite the distance and the lockdown problems associated with the COVID-19 pandemic. The programming mode that more students liked was the blocks-based. On this respect, some students commented the following:

S1. *The experience to use a tool like this had a very positive impact to learn the topics in the course.*

S2. *It was interesting the way in which I was able to program. The results were displayed directly in the real-time video.*

S3. *For me, it was interesting the fact that the functions were executed in real-time in the remote laboratory.*

S4. *With the real-time video, I realized several mistakes in my codes and I fixed them.*

S5. *It was very interesting the way to modify and contrast the codes in blocks and in the programming language.*

S6. *It was interesting how I could experiment, putting and modifying different instructions and codes in the interface and seeing their response in real-time.*

The PEU factor was the lowest scored in the survey. Some students argued difficulties in the access to the RL and in the real-time video due to internet and video codecs troubles. As mentioned, the WebRTC standard has some problems with freeze and breaks in some cases. However, the reconnection of the client is guaranteed by the Janus gateway and several scripts designed in Python language. In addition, the problems in the access of the RL were fixed and the experimented robustness troubles in the video transmission will be fixed to work with future students in other courses. These problems influenced the perception of the students in this factor. Despite these problems, the students evaluated the RL in a scale from 1 to 5 as it is depicted in Fig.9. The mean of these scores was 4.25.

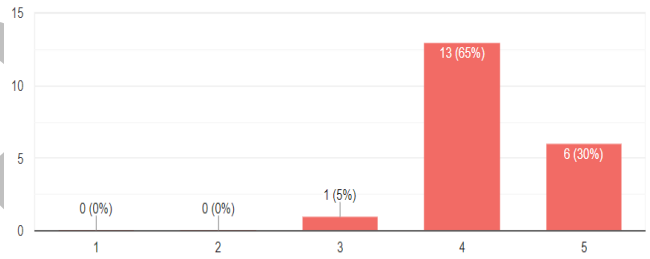


Fig. 9. Overall evaluation of the RL (*n=20*).

The IU factor is the higher scored in the survey, despite the described technical problems in the platform. The students stated that they want the laboratory to be included in other subjects in the curricula. Even, the students are willing to employ their spare time in learning other concepts and topics of CS with the RL. This fact demonstrates that the laboratory had a good reception and acceptance from the students, and it contributed to enhance their learning. With these factors, it was calculated Pearson's correlation matrix to know the relationships between the TAM factors. Table VII illustrates these values.

TABLE VII.
Results of Pearson's correlation for the TAM factors (*n=20*).

TAM Factor	PU	PEU	IU
PU	1		
PEU	0.710*	1	
IU	0.532**	0.428	1

*Correlation is significant at 0.01 level. Current p-value 0.00. **Correlation is significant at 0.05 level. Current p-value 0.016.

For this case, there are significant positive relationships, on one hand, between PU and PEU with it is demonstrated by the results analyzed previously. On the other hand, the IU factor is mainly determined by the its relationship with the PU factor.

PREPRINT

The students found useful the RL for their learning and this fact was more relevant than the PEU factor in the intention to use the RL.

In a similar way, the students were asked about positive elements and aspects to improve in the RL. A synthesis with some of these comments is presented in Table VIII.

TABLE VIII.
Synthesis of the positive aspects and elements to improve in the RL.

Description	Theme	Examples of comments
Positive aspects	Programming mode	"I think that the programming with blocks is more dynamic"
		"The programming with blocks is more graphic and intuitive for the students"
		"I consider that both forms to programming are interesting because they facilitate learning"
	Real-time video and processing	"It was interesting the real-time video because we could observe our failures in the code's structure"
		"The real-time camera was very interesting"
		"The laboratory has different tools and their functioning in virtual mode was interesting"
	Learning	"The experience to use a tool like this had a very positive impact to learn the topics in the course"
		"The fact of the distance doesn't be an obstacle to learn and see the functioning of the algorithms that one creates in home"
		"It was interesting how I could experiment, putting and modifying different instructions and codes in the interface of the RL"
Aspects to improve	Video and camera	"The laboratory is good but, in some occasions, it did not give video"
		"For me, many times the video generated errors which difficulted the work with the laboratory"
		"Sometimes, the video had few errors"
		"Maybe the camera position"
	Restarting functions	"It would be nice if the restart process will be faster in case of failure"

In sum, these elements show that the RL had a good acceptance and reception by the students who used it in function of their learning needs in the course. The TAM approach was a suitable starting point to know the perceptions of the students concerning the RL and how this was a technological tool that supported learning and motivation in the students.

VI. Conclusions and Further Work

In this article, we described a low-cost remote laboratory that was used to learn and teach programming and physical computing with Raspberry Pi and Python programming language. The concept of the RL arose from a quick transition from in-person classes towards an online learning format due to the lockdown restrictions that limited the access of the students to the laboratories which has consequences for their learning process. With the development of the RL, firstly, we proved that it is possible to create low-cost RLs in a short timeframe according to the learning needs of the students. The software structure with the VPS, the Python packages, and the usage of 3D printers in the hardware setup allowed to accelerate the design and construction of the RL. Secondly, the results demonstrated that the students took advantage of the RL's features for their learning and comprehension in the topics in the course. Likewise, the students manifested their intentions to use the RL in future subjects or even in their free time. This fact shows that the different components of the RL in software and hardware motivated the students to learn programming and experiment with hands-on activities. The RL was oriented not only as a technical tool but also as a pedagogical element to enhance learning and comprehension of the students. Further work will be focused on improving the technical functioning of the RL, especially, regarding the video transmission and the students' access. Also, a study of the educational implications will be conducted to analyze the impact of the RL on the skills and competencies of the students in engineering and CS. Complementary, the features of the laboratory will be improved to meet the requirements of the standard *IEEE 1876-2019 - IEEE Standard for Networked Smart Learning Objects for Online Laboratories*.

References

[1] I. Gustavsson *et al.*, "On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories," *IEEE Trans. Learn. Technol.*, vol. 2, no. 4, pp. 263–274, 2009, doi: 10.1109/TLT.2009.42.

[2] M. D. H. Rahiem, "The emergency remote learning experience of university students in indonesia amidst the COVID-19 crisis," *Int. J. Learn. Teach. Educ. Res.*, vol. 19, no. 6, pp. 1–26, 2020.

[3] C. Hodges, S. Moore, B. Lockee, T. Trust, and A. Bond, "The difference between emergency remote teaching and online learning," *Educ. Rev.*, vol. 27, 2020.

[4] UNIMINUTO, "SmartLabsUniminuto." [Online]. Available: https://www.smartlabsuniminuto.com/. [Accessed: 21-Jan-2021].

[5] I.-F. Liu, M. C. Chen, Y. S. Sun, D. Wible, and C.-H. Kuo, "Extending the TAM model to explore the factors that affect Intention to Use an Online Learning Community," *Comput. Educ.*, vol. 54, no. 2, pp. 600–610, 2010.

[6] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Q. Manag. Inf. Syst.*, vol. 13, no. 3, pp. 319–339, 1989, doi: 10.2307/249008.

[7] S. Hodges, S. Sentance, J. Finney, and T. Ball, "Physical computing: A key element of modern computer science education," *Computer (Long. Beach. Calif).*, vol. 53, no. 4, pp. 20–30, 2020.

[8] J. P. C. de Lima, L. M. Carlos, J. P. Schardosim Simão, J. Pereira, P. M. Mafra, and J. B. da Silva, "Design and implementation of a remote lab

PREPRINT

for teaching programming and robotics," *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 86–91, 2016, doi: 10.1016/j.ifacol.2016.11.133.
[9] H. Guerra, A. Cardoso, V. Sousa, and L. M. Gomes, "Remote experiments as an asset for learning programming in Python," *Int. J. Online Eng.*, vol. 12, no. 4, pp. 71–73, 2016, doi: 10.3991/ijoe.v12i04.5278.
[10] J. B. da Silva, G. de Oliveira, I. N. da Silva, P. M. Mafra, and S. M. S. Bilessimo, "Block.ino: Remote Lab for Programming Teaching and Learning," *Int. J. Adv. Eng. Res. Sci.*, vol. 7, no. 1, pp. 41–47, 2020, doi: 10.22161/ijaers.71.6.
[11] M. M. McGill, "Learning to program with personal robots: Influences on student motivation," *ACM Trans. Comput. Educ.*, vol. 12, no. 1, pp. 1–32, 2012.
[12] M. A. Rubio, R. Romero-Zaliz, C. Mañoso, and P. Angel, "Closing the gender gap in an introductory programming course," *Comput. Educ.*, vol. 82, pp. 409–420, 2015.
[13] P. G. Feijóo García and F. De la Rosa, "RoBlock - web app for programming learning," *Int. J. Emerg. Technol. Learn.*, vol. 11, no. 12, pp. 45–53, 2016, doi: 10.3991/ijet.v11i12.6004.
[14] M. M. Rahaman, E. Mahfuj, M. Haque, R. Shekdar, and K. Z. Islam, "Educational Robot for Learning Programming through Blockly based Mobile Application," *J. Technol. Sci. Eng.*, vol. 1, no. 2, pp. 21–25, 2020.
[15] M. Seraj, E. S. Katterfeldt, K. Bub, S. Autexier, and R. Drechsler, "Scratch and google blockly: How girls' programming skills and attitudes are influenced," *ACM Int. Conf. Proceeding Ser.*, 2019, doi: 10.1145/3364510.3364515.
[16] D. Ozoran, C. N. Ercil, and D. Topalli, "Using Scratch in introduction to programming Course for Engineering Students," *Educ. Technol. Distance Educ. Eng.*, no. i, pp. 125–132, 2012.
[17] S. Greenwold, "Spatial computing," *Massachusetts Inst. Technol. Master*, 2003.
[18] D. O'Sullivan and T. Igoe, *Physical computing: sensing and controlling the physical world with computers*. Course Technology Press, 2004.
[19] M. Ben-Ari, "Constructivism in computer science education," *Acm sigcse Bull.*, vol. 30, no. 1, pp. 257–261, 1998.
[20] M. Przybylla and R. Romeike, "Physical Computing and Its Scope--Towards a Constructionist Computer Science Curriculum with Physical Computing.," *Informatics Educ.*, vol. 13, no. 2, pp. 241–254, 2014.
[21] D. Alimisis and C. Kynigos, "Constructionism and robotics in education," *Teach. Educ. Robot. Constr. Pedagog. methods*, pp. 11–26, 2009.
[22] M. Moallem, W. Hung, and N. Dabbagh, *The Wiley handbook of problem-based learning*. Wiley Online Library, 2019.
[23] N. Fraser, "Ten Things We've Learned from Blockly," in *2015 IEEE Blocks and Beyond Workshop*, 2015.
[24] Mathworks, "ThingSpeak - MATLAB & Simulink." [Online]. Available: https://www.mathworks.com/products/thingspeak.html. [Accessed: 23-Jan-2021].
[25] D. Mason and K. Dave, "Block-based versus flow-based programming for naive programmers," *Proc. - 2017 IEEE Blocks Beyond Work. B B 2017*, vol. 2017-Novem, pp. 25–28, 2017, doi: 10.1109/BLOCKS.2017.8120405.
[26] Y. Matsuzawa, Y. Tanaka, and S. Sakai, "Measuring an Impact of Block-Based Language in Introductory Programming - Stakeholders and Information Technology in Education," 2016, pp. 16–25.
[27] C. A. Shaffer *et al.*, "Algorithm Visualization: The state of the field," *ACM Trans. Comput. Educ.*, vol. 10, no. 3, pp. 1–22, 2010, doi: 10.1145/1821996.1821997.
[28] M. Noone and A. Mooney, "Visual and textual programming languages: a systematic review of the literature," *J. Comput. Educ.*, vol. 5, no. 2, pp. 149–174, 2018.
[29] A. Jayal, S. Lauria, A. Tucker, and S. Swift, "Python for teaching introductory programming: A quantitative evaluation," *Innov. Teach. Learn. Inf. Comput. Sci.*, vol. 10, no. 1, pp. 86–90, 2011.
[30] U. Nikula, J. Sajaniemi, M. Tedre, and S. Wray, "Python and roles of variables in introductory programming: experiences from three educational institutions," *J. Inf. Technol. Educ. Res.*, vol. 6, no. 1, pp. 199–214, 2007.
[31] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. O'Reilly Media, 2018.
[32] F. Korbel, *FFmpeg Basics: Multimedia handling with a fast audio and video encoder Paperback*. 2012.
[33] D. DeJonghe, *The Complete NGINX Cookbook*. O'Reilly Media, 2019.
[34] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Janus: A general purpose WebRTC gateway," *Proc. Conf. Princ. Syst. Appl. IP Telecommun. IPTComm 2014*, pp. 1–8, 2014, doi: 10.1145/2670386.2670389.
[35] A. Amirante, T. Castaldi, L. Miniero, and S. Pietro Romano, "Performance analysis of the Janus WebRTC gateway," *Proc. 1st Work. All-Web Real-Time Syst. AweS 2015 - Conjunction with EuroSys 2015*, 2015, doi: 10.1145/2749215.2749223.
[36] R. Manson, *Getting Started with WebRTC: Explore WebRTC for real-time peer-to-peer communication*. 2013.
[37] G. Inc., "WebRTC." [Online]. Available: https://webrtc.org/. [Accessed: 08-Jan-2021].
[38] "Welcome to AIOHTTP — aiohttp 3.7.3 documentation." [Online]. Available: https://docs.aiohttp.org/en/stable/. [Accessed: 13-Feb-2021].
[39] S. Hadjerrouit, "Constructivism as guiding philosophy for software engineering education," *ACM SIGCSE Bull.*, vol. 37, no. 4, pp. 45–49, 2005, doi: 10.1145/1113847.1113875.
[40] M. E. Hoque, "Three domains of learning: cognitive, affective and psychomotor," *J. EFL Educ. Res.*, vol. 2, no. 2, pp. 45–52, 2016.
[41] T. Teo, *Technology acceptance in education*. Springer Science & Business Media, 2011.
[42] Z. Shana and E. Abulibdeh, "Cloud computing issues for higher education: theory of acceptance model," *Int. J. Emerg. Technol. Learn.*, vol. 12, no. 11, pp. 168–184, 2017.
[43] M. Chow, D. K. Herold, T.-M. Choo, and K. Chan, "Extending the technology acceptance model to explore the intention to use Second Life for enhancing healthcare education," *Comput. Educ.*, vol. 59, no. 4, pp. 1136–1144, 2012.
[44] A. Tlili, F. Essalmi, and M. Jemni, "Improving learning computer architecture through an educational mobile game," *Smart Learn. Environ.*, vol. 3, no. 1, pp. 1–14, 2016.
[45] C. H. Yu, "An introduction to computing and interpreting Cronbach Coefficient Alpha in SAS," in *Proceedings of 26th SAS User Group International Conference*, 2001, vol. 2225, pp. 1–6.

PREPRINT