Curriculum to use the Sphero RVR in IT-Adventures

FINAL DESIGN DOCUMENT

sddec21-14
Information Assurance Center
Doug Jacobson

Dakota Berbrich – Meeting Facilitator and SCRUM Master
Noah Berkland – External Contact Facilitator
Aaron Goff – Systems Engineer
Nolan Jessen – Meeting Scribe and Report Manager

sddec21-14@iastate.edu
http://sddec21-14.sd.ece.iastate.edu/

Revised: December 7, 2021/Version 4.0

Executive Summary

Development Standards & Practices Used

- CSTA K-12 CS Standards
- Agile Workflow (Jira)
- Version Control (Iowa State ECpE GitLab server)
- IEEE Coding Standards
- ISU Competencies
    - Engineering Knowledge
    - Continuous Learning
    - Quality Orientation
    - Planning
    - Teamwork
    - Communication
    - Professional Impact
    - Customer Focus

Summary of Requirements

- There should be two sets of curriculum provided, one which focuses on introductory coding and the other which focuses on a more advanced level of coding.
- The curricula should be accessible to students with any amount of coding experience.
- The curricula should be rather hands-off for teachers with their role focusing mostly on support.
- The challenges must ramp up in difficulty slowly / at a standard pace.
- The final course must be reliable and easy to put together.
- The kits must be affordable and easily supplied.

Applicable Courses from Iowa State University Curriculum

- CPR E 185: Introduction to Computer Engineering and Problem Solving I
- CPR E 288: Embedded Systems 1
- CPR E 294: Program Discovery
- CPR E 394: Program Exploration

- COM S 227: Object-Oriented Programming
- COM S 228: : Introduction to Data Structures
- COM S 309: Software Development Practices
- ENGL 314: Technical Communication

New Skills/Knowledge acquired that was not taught in courses

- Python
- Scratch
- micro:bit API
- Raspberry Pi API
- Jira
- Raspberry Pi 3 & Raspbian

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

To start, the team would like to thank Doug Jacobson and the Information Assurance Center for their generous donation of the Sphero RVR, Raspberry Pi 3, and micro:bit kits to each member of the team. This has allowed each member to physically work with the kits that the high school students will use, while maintaining a safe workplace and working remotely during the COVID-19 pandemic. Additionally, we would like to extend our gratitude to all of the members of IT-Adventures for supporting us and guiding us throughout this project.

1.2 Problem and Project Statement

General Problem Statement

Some Iowa high schools lack effective coding programs for students who wish to learn how to program. As programming continues to grow in importance as a general skill for the future workforce, these students fall behind in what careers they may pursue or be aware of. Although there are existing programs for introducing students to the field, such as FIRST® LEGO® League, these are often expensive and hard to manage for schools with no knowledgeable staff to lead them.

Proposed Solution

Our team aims to develop a new engaging set of curricula to fulfill the needs of these students. Through IT-Adventures, an established extracurricular program for high school students, we will develop two different venues to accommodate students new to programming and those who already have a background in the field. Both venues will utilize the Sphero RVR, an approachable and expandable platform.

The first venue, Robotics, will focus on learning the basics of programming, but also work with expanding the capabilities of the Sphero RVR using a set of magnetic circuits and components called littleBits. These components and the RVR will be controlled by a small microcontroller called a micro:bit that can be programmed using Python, Scratch, and JavaScript.

The second venue, Smart-IT, will focus on more advanced programming practices and utilize a Raspberry Pi to interact with the RVR. This venue will utilize the RVR but require students to complete programming puzzles and problems to interact with it.

Through the IT-Adventures curricula, students of all skill levels will learn how to program and debug different programs while tackling fun challenges. Both curricula will be monthly, teaching both the basics of programming and having a series of challenges that apply what has been learned.

1.3 Operational Environment

The operational environment will primarily consist of high school classrooms and club rooms. This area, indoors, should be a relatively stable operating environment. By the end of the curriculum, teams will be using the robots on well-defined challenge courses, which will be built on solid

surfaces (i.e. plywood ramps) and will be easily maintained. Therefore, there are not any concerns that the Sphero RVR (which is specifically designed with durability and outdoor activities in mind) will have any challenges in these environments with standard care.

1.4 Requirements

Functional Requirements

- There should be two sets of curriculum provided, one which focuses on introductory coding and the other which focuses on a more advanced level of coding.
- The curricula should be rather hands-off for teachers with their role focusing mostly on support.

Non-Functional Requirements

- The curricula should be accessible to students with any amount of coding experience.
- The challenges must ramp up in difficulty slowly / at a standard pace.
- The final course must be reliable and easy to put together.

Economic Requirements

- The kits must be cheap and easily supplied.

1.5 Intended Users and Uses

The intended users are high school students who are new to programming and those who already have experience. While learning the basics of coding, students will break into small groups to tackle the challenges presented to them. These challenges, over the span of a regular school year, will build in intensity, until the students are attempting the final challenge. This final challenge will consist of students quickly programming the robot to autonomously run through some course for a few minutes. In general, the robots will be used as learning tools to convey the power and limitations of programming.

1.6 Assumptions and Limitations

- Assumptions
    - The teams will consist of high-school students within the state of Iowa
    - The RVR will be flexible, with different microcontrollers and swappable peripherals to control the vehicle and how it interacts with its environment
    - The teams will be working on limited budgets
    - The students will have little to no programming experience
- Limitations
    - The teams will work on limited budgets (fit within a high school budget), so the final kits should work to contain only the RVR, a microcontroller, and necessary peripherals
    - The challenges will be programmable within the period of a month, so they cannot be too challenging to implement and test

1.7 Expected End Product and Deliverables

The first deliverable will be the specification of the robotics kit. Each kit will include a Sphero RVR for teams to use, as well as other accessories for ease of use and full programmability. The goal will be to get the kits as cheap as possible while including all the necessary accessories. The kit specifications will be delivered in the middle of April.

The next deliverable will be the general curriculum, which will be used for the first few months of both programs (Smart-IT and Robotics). Even though these are separate curricula, they will be developed in parallel because of their similarities. The goal of the general curriculum is to teach basic coding skills; to do this, each month will contain at least one presentation and multiple coding examples and tests for the students. This is intended to be run from September through November of each school year. Therefore, it will be delivered at the end of the spring semester, in May, so that it may be sent to the schools in time for teams to use it this fall (2021).

The Robotics curriculum is the first curriculum, and is intended to focus on a lower level of coding, utilizing the Sphero kits. This will teach students basic coding skills and start to prepare them for different challenges using the Sphero RVR. The first several months of the program (from September through December) will consist of weekly lessons, each teaching a basic skill. The spring semester will then see students working on the final challenge, which will culminate in the spring competition, pitting teams against each other. They will also be given a challenge on the day of the competition, which will be solvable using similar techniques to the individual challenges that were faced earlier in the year. Since the curriculum is split up by month, each piece of curriculum will be delivered at least 3 months in advance, after being thoroughly tested and ensured to work.

Similar to the Robotics curriculum, the Smart-IT curriculum will be a focused curriculum on a higher level of coding knowledge, again utilizing the Sphero kits. This will teach students more advanced coding skills and introduce different challenges using the Sphero RVR, especially focused on the possibilities of network and wireless opportunities (and challenges). Each month will contain at least one presentation and multiple coding challenges and examples. Again, these months of lessons and challenges will lead to a final challenge at the end of the school year. Since this set of curriculum is also split up by month, these will also be delivered at least 3 months before the unit in question is to be used.

2 Project Plan

2.1 Task Decomposition

The primary task at hand is developing a curriculum, using the Sphero RVR, that allows high school students to learn code and compete in the IT-Adventures challenges. This can be broken down into the following set of primary tasks and subtasks:

1. Understand how the Sphero RVR, Raspberry Pi 3, and micro:bits may link together

To teach coding through this platform, it must first be understood by the team.

   a. Understand the general requirements and functionality of the Sphero RVR
   b. Connect the Raspberry Pi 3 to the Sphero RVR and link their code bases
   c. Connect the micro:bit kit and figure out what value it has versus the Pi

2. Create the general coding curriculum

For the first few months of the curriculum (2-3 months), both sets of curricula will have the same theme. The goal will be to teach the basics of coding (i.e., defining what variables are). Assuming all students have no base in coding, this will be general concepts like loops, conditionals, and variables.

   a. Define what will be learned in months 1, 2, and 3 at a high level
   b. Create basic coding challenges and templates for each month
   c. Create presentations to go along with code for each month

3. Create the Robotics Curriculum

The Robotics curriculum will follow the general curriculum, as the students will use the Sphero RVR in tandem with the micro:bit and added peripherals to tackle increasingly complex problems.

   a. Define the challenges for months 4 through 7 at a high level
   b. Create basic code challenges and templates for months 4 through 6
   c. Create presentations for each challenge set
   d. Create final challenge and course

4. Create the Smart-IT Curriculum

The IT curriculum, also following the general curriculum, will split off into a series of more complex challenges than the Robotics curriculum. This will be aimed at students who may have some experience coding and are ready for more advanced challenges.

   a. Define the challenges for months 4 through 7 at a high level
   b. Create basic code challenges and templates for months 4 through 6
   c. Create presentations for each challenge set
   d. Create final challenge and course

2.2 Risks And Risk Management/Mitigation

The primary risks come with the final challenge. Leading up to that point, each month's worth of curriculum should be relatively simple to develop, building off the previous month. The risk for each month of curriculum is therefore quite low, around 0.1 (0.2 for some of the later months, where deadlines may start to slip). However, the final challenge (in both curricula) will pose some challenges in identifying the correct skill level and appropriateness of the challenge. Therefore, it is initially pegged as a 0.7 risk level. To lower this, the team will employ an external review to ensure

that it is not too difficult, at least one month before it is due. This will give ample time to remedy any issues that pop up in testing the final challenge.

An additional area of risk is getting the multiple systems to work together. This is well-demonstrated within the Robotics sphere, where the students have to create a system that utilizes the Sphero RVR, the micro:bit controller, and the littleBits topper kit to create a cohesive robot. Any time a system requires multiple components from different manufacturers, there are going to be challenges in getting them to all work together (initial risk level estimated to be 0.5). However, there are a few advantages to this system already. The team has decided that students will use Microsoft MakeCode to develop their applications, which easily connects with the micro:bit. To connect the micro:bit to the RVR, all that is necessary is connecting a microUSB cable from the micro:bit to the RVR. This system was specifically chosen to work together automatically, which should greatly reduce the risk that students have issues getting it all to work. In addition, the team and the IT-Adventures group is working directly with the Sphero group to resolve any issues, technical or otherwise, that come up throughout the development process. Considering these risk-mitigation measures that are implemented, this risk should be adequately addressed.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Most of the milestones will be relatively straightforward. For each month, there should be a completed PowerPoint presentation and a set of 3-4 tasks (at minimum) that demonstrate what has been learned that month. These tasks should have an example or a code template to follow, where the students can then implement it themselves.

For the final challenge, there will be a set of 3 to 4 requirements for teams to follow. This should be easily tackled within a few hours of learning the final challenge and should have an intended runtime between 3 and 10 minutes.

2.4 Project Timeline/Schedule

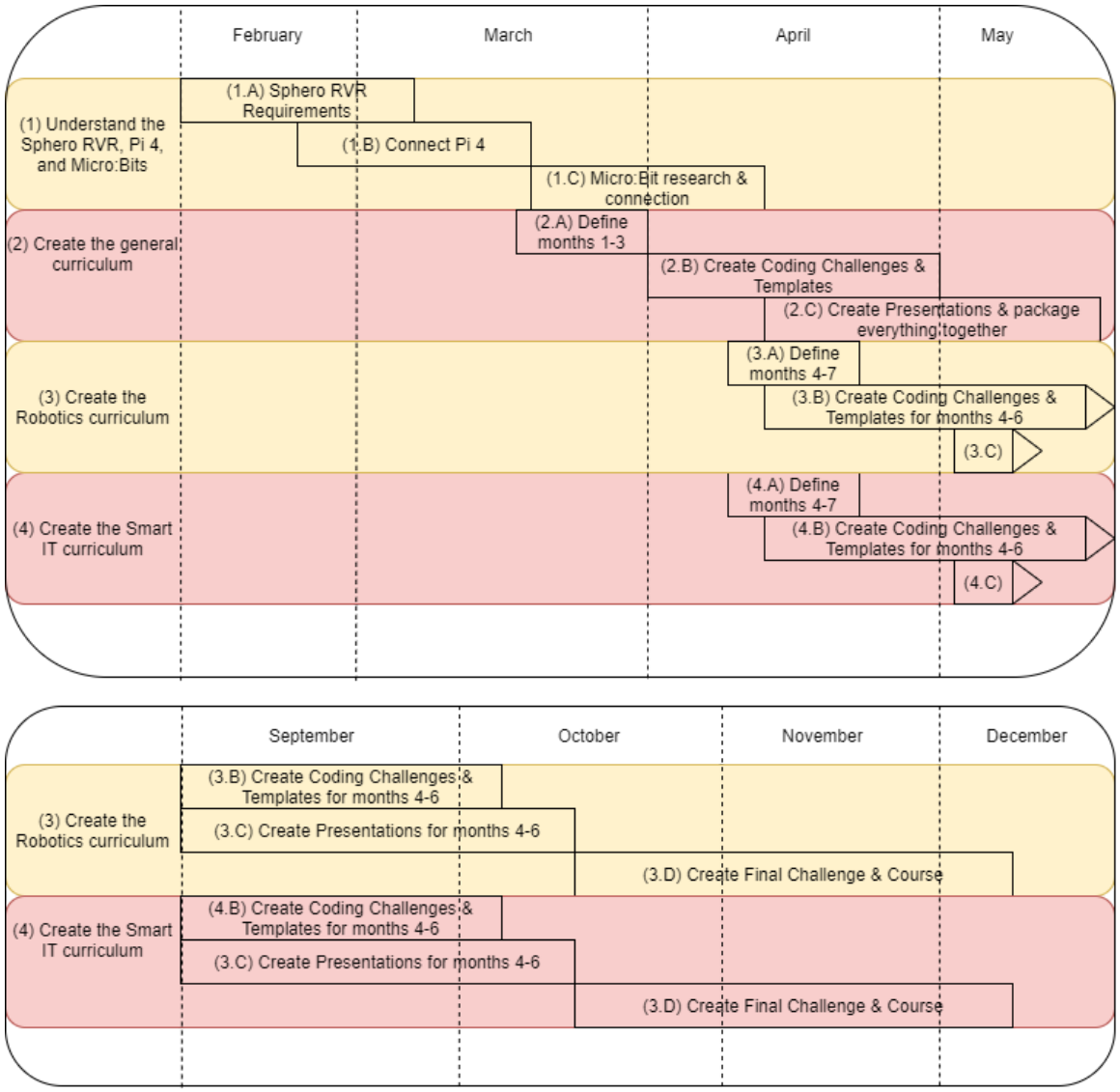

Figure 1: Gantt Chart of Project Timeline

In the Gantt chart above, the primary tasks and subtasks are laid out. This highlights how the two main curricula (Robotics and Smart-IT) will be developed in parallel. In addition, this is split into the two semesters. The main deliverables for this project will be the General Curriculum, the Robotics Curriculum, and the Smart-IT Curriculum. The General Curriculum will be delivered by the end of the spring semester (in May). This is because it will be used this fall in schools, and therefore must be ready by the beginning of the fall. The two specific curriculums, Robotics and Smart-IT, will then be delivered in mid-October (as teams will transition to those when they are done with the general curriculum, around the end of November). The final deliverable will be the final challenges, which will be delivered at the end of the fall semester. These challenges will be used the following spring, around April 2022, so they are going to be the final developed product.

2.5 Project Tracking Procedures

The team will be using the Iowa State ECE GitLab for developing code bases for the Sphero RVR. These bases will be sent out with each month of curriculum and are intended to be starting points for the high school students using these robots.

Alongside GitLab, the team will be using Jira to track development of each set of curriculum and other tasks that need to be done. This is linked with the GitLab, so individual commits can be linked to tasks or bugs in Jira. By using Jira, the team is also able to set up both a backlog of tasks and look ahead, developing a roadmap for the months ahead.

2.6 Personnel Effort Requirements

Task	Task Title	Description	Expected Man-Hours
1	**UNDERSTAND SPHERO RVR, RASPBERRY PI 3, AND MICRO:BITS**		
1.A	Understand Sphero RVR	*The goal is to understand what the basic capabilities and limitations of the Sphero RVR are.*	12
1.B	Connect Raspberry Pi 3 to Sphero RVR	*Connect the Pi to the RVR and understand how one can control the RVR via the Pi (and how it can be autonomously run).*	24
1.C	Connect micro:bit to Sphero RVR	*Understand the basics of the micro:bit kit and determine whether it is useful for running the RVR (if it is worth the price to include it in the final kit).*	12
2	**CREATE THE GENERAL CODING CURRICULUM**		
2.A	Define months 1-3 curriculum	*At a high level, define what should be learned each month.*	4
2.B	Create coding challenges for each month	*Create a minimum of 3 coding examples and challenges for each month, intended to teach and practice the skills of what is being learned.*	40
2.C	Create presentations for each month	*Create a basic presentation to teach the why and how of the skills for the month.*	9
3	**CREATE THE ROBOTICS CURRICULUM**		
3.A	Define months 4-7 curriculum	*At a high level, define what should be learned each month.*	8
3.B	Create the coding challenges for months 4-6	*Create coding examples and challenges for each month. As these grow more complex and specific to the curriculum, they will take more time to develop and test.*	60

3.C	Create presentations for each month	*Create a basic presentation to teach the why and how of the skills for the month.*	10
3.D	Create final challenge and course	*Create the final challenge for the curriculum, make sure it can be completed within a few hours and is a simple set up, and create the course to go alongside it.*	60
4	**CREATE THE SMART IT CURRICULUM**		
4.A	Define months 4-7 curriculum	*At a high level, define what should be learned each month.*	8
4.B	Create the coding challenges for months 4-6	*Create coding examples and challenges for each month. As these grow more complex and specific to the curriculum, they will take more time to develop and test.*	60
4.C	Create presentations for each month	*Create a basic presentation to teach the why and how of the skills for the month.*	10
4.D	Create final challenge and course	*Create the final challenge for the curriculum, make sure it can be completed within a few hours and is a simple set up, and create the course to go alongside it.*	60
		TOTAL TIME	377

2.7 Other Resource Requirements

The primary resources required to complete the project are the Sphero RVR kits, the Raspberry Pi 3 (and accessories), and the micro:bit kits (which include the littleBits topper kit). As these are what the students will be using, it is important that the team understands how they work.

2.8 Financial Requirements

There are no financial requirements required to directly run this project, but one of the major goals is to keep the kits as cheap as possible, so that schools can easily participate without a huge financial commitment. This is especially important within the Smart-IT realm, where beyond the Sphero RVR and the Raspberry Pi 3, the kits are not as well defined. In contrast, the Robotics curriculum will be limited to the RVR, the micro:bit microcontroller, and the littleBits topper kit, so there is no concern about going over budget in that realm.

3 Design

3.1 Previous Work And Literature

There have been multiple groups around the nation and world that have implemented some form of high-school coding curriculum. One of the most prominent groups is FIRST®, which has programs available for students from kindergarten age to senior year of high school. FIRST® focuses on a combined program of hardware, software, and outreach (where groups work within their communities). The primary difference with our program is that it is a much smaller time commitment (making it more feasible for students to enter and teachers / school aides to run) and that it has a much lower financial barrier to entry. With this smaller scope, we are also focused on teaching students more of the basics in the beginning, rather than leaving them to explore more on their own.

Beyond this external group, we are basing our curriculum on what was previously developed by IT-Adventures. IT-Adventures has previously developed programming curricula. However, previous years used the LEGO® NXT robot. While this robot platform was versatile and powerful for demonstrating different programming concepts, the older platform was not as expandable as IT-Adventures wished. The team decided to switch to the Sphero RVR, and a new curriculum must be built around this platform (for both the Robotics and the Smart-IT programs). This new platform is more powerful and can interface with many different computing platforms (such as the Raspberry Pi or the micro:bit). Previously, the team relied on LEGO®-provided lessons, which are no longer available with the Sphero RVR. Instead, we are working on a logical progression of lessons for students to learn how to code from, starting with the basics of getting motors to run. The goal is that the students will reach certain levels each month, signifying that they have learned a certain skill set. At the end of the year, they are then prepared for the competition.

Outside of the IT-Adventures realm, the Iowa Legislature passed House File 2629 in 2020, which required a series of computer science curricula at schools around the state of Iowa. This bill utilizes the CSTA K–12 CS Standards as the standards for these curricula. To help schools meet these requirements, IT-Adventures wishes for the curricula to address as many of these standards as possible and logical.

3.2 Design Thinking

The most important aspects that were defined while laying out the curricula were focusing on the *who* and their skillset. The curricula are focused on high-school students, which limits what kind of language should be used and how the students should be treated. In addition, defining their skill level was extremely important. In Robotics, it was determined that they should have effectively no coding experience. This is the opposite of Smart-IT, where the students should have at least a basic idea of how to program and should be ready for some more advanced challenges. Finally, the challenges that the students would face were defined to be at least monthly. This also includes a larger end-of-the-year challenge that the students would face.

With the main defined limitations, the next step was to *ideate*, which was also limited because of the previously created curriculum that the team was building off of. Within this, the format was focused between two proposed designs (as discussed in 3.3). Both curriculum formats have a monthly challenge and could meet the needs of the students. These are debated and discussed further in 3.3.

3.3 Proposed Design

The design issue that we have faced is that the design itself is already relatively laid out. We have been tasked with creating a set of monthly curricula. Within this, we have been working on what programming concepts to teach, how to teach them, and what kinds of challenges the students will face in the future. The main ways that we considered splitting up the curricula are as follows:

1. Monthly Topics and Challenges
    a. In this format, the students would get one set of information a month (which would all tie together) and would have one primary, major challenge to complete. The upside to this approach is that this would be much simpler to implement and would have larger, potentially more interesting challenges for the students. The biggest downside that we considered was that it would be much easier for students to get off-track and either quickly finish the challenges (and get bored) or not finish the challenges and fall behind more easily.
2. Monthly Challenges; Weekly Topics
    a. In this more structured format, students would have a smaller weekly topic. They would then attempt to solve one or two small challenges related to the topic. At the end of the month, they would then be given a larger challenge to solve, integrating every topic they had learned over the previous weeks. While this creates more work for us, it provides a much more structured learning environment for the students. They still have the flexibility to learn and implement the challenges however they wish, but by focusing more on these weekly challenges, the concepts that they are learning should be better retained in the long-term.

With the above considerations and limitations, the team has decided to move forward with weekly topics. Monthly topics were originally attempted. However, the attempts to lay the months out and structure the topics in a way that made sense and moved at a solid pace were consistently flawed, either being too slow, too fast, too easy, or too hard. The switch to the weekly topics has made much more sense. The developing curricula is also much more flexible and easier to work with (as discussed further in 3.5).

3.4 Technology Considerations

The Sphero RVR is the primary robot that will be used in both Robotics and Smart-IT. However, the intention is that the Robotics program will use the micro:bit as the controller, while Smart-IT will use the Raspberry Pi 3. One question that was asked was why the Pi would not be used for the Robotics program as well. This led to the discussion where the Pi was viewed as much more controllable, but also much more complicated to get started on. While the Smart-IT venture will have students that have some experience programming, the Robotics program will not. It is instead intended for students who have literally no programming experience. Therefore, the micro:bit is a much better fit for that program.

In addition, the Pi 4 was originally being considered for the Smart-IT program. However, it was found to have too large of a power draw and drew more than what the RVR could provide. Therefore, the Pi 3 B and/or Pi 3 B+ will be a better fit.

3.5 Design Analysis

Overall, the proposed design from 3.3 has worked and been implemented successfully. The curricula have been laid out in a logical order and been verified and accepted (as discussed in 4.3, Acceptance Testing) by the client. Both areas of curriculum, within the "General Curriculum" area (as discussed in Section 2), are completed and running successfully.

The biggest area of improvement that the team is looking with these schedules is determining how to teach the material to the students. Without any guidance, the students can easily become frustrated and lose interest. On the flipside, it is expected that the people volunteering to run the program (such as teachers or librarians) will not have any experience in coding either. This means that there must be some form of lesson that is not too intensive and that the students can follow along with easily.

3.6 Development Process

We are primarily following an Agile development process. This is because each month of both sets of curriculum relies on previously developed months. Therefore, the entire set must be first analyzed and designed as a whole, then the months are developed and verified in order (to ensure they are neither too easy nor too challenging). This model is both iterative and revisional, so an Agile process works best for our team.

3.7 Design Plan

Our individual units that we are developing are related to each week of curriculum, or each challenge (as happens in the later months, challenges extend for more than one week, but continue to be one "unit"). Therefore, the design plan is that, for each curriculum, we will first create a general outline. This outline will be discussed and revised multiple times, until it is accepted by all parties. By creating the outlines first, we will be ensuring that we have two sets of curriculum which follow a logical order and which are at different levels of difficulty. This will also satisfy the requirement that the challenges will ramp up in difficulty in a logical way and that they are accessible to students with little to no coding experience.

After creating the outline, the individual units will be created in chronological order, starting with the units in September and ending with the units in April, when the final challenge will occur. With these units being created, the individual unit testing (as discussed in 4.1) will occur. This testing will also ensure that the challenges can be completed with some ease by the students, but that they still teach the students and are an enjoyable experience.

To ensure that the modules build on one another, as intended by the outline, they will then face interface testing. The units do not directly rely on a working code base from the previous lesson, but the interface is that those lessons are learned, so that they can be applied in the current week. Therefore, the interface testing is necessary, and will be performed as described in 4.2. This will be followed by acceptance testing, which will verify that the curriculum (as a whole) meets the client's expectations.

This testing will also encompass the final challenge, wherein each component will be individually tested before interface testing. The interface testing will, similar to the lessons, ensure that all challenges can be solved by using lessons learned in the curriculum. Together, the testing as a whole will ensure that everything from the lessons to the challenges go off without a hitch.

4 Testing

4.1 Unit Testing

Our project is rather unique in that there are no major software or hardware units. Instead, the "units" could be considered individual weeks in the curricula, each of which cover a separate unit. While they tie into each other, each one has individual pieces of software or challenges for the students to code. Therefore, the testing is split into a few components:

- Ensure that the challenge is programmable
    - Create an example, running program for each proposed challenge. This code should be relatively simple (as it is intended to be used as an example by the teachers and to verify that the curricula work by the team) and should be free of unintended effects.
- Ensure that the lesson is easily understandable and can be completed in less than two hours
    - Using a person who was not involved in the creation of the lesson (and preferably someone who has little to no programming experience), walk through the lesson with them and note any changes that should be made

To add to this, the final challenge needs to be tested, but faces some slightly different requirements. Essentially, each component needs to be covered by previous lessons. This mostly falls under the purview of interface testing (as it focuses on the compatibility between those lessons and the final challenge) but ensuring that the individual components are valid falls under unit testing. Each component requires the following testing:

- Durability testing
    - The Sphero RVRs are fast and durable robots. However, the high probability of a bug in a team's software means that the field must be made to withstand repeated hits from these robots. Therefore, they are tested using programming to simply ram into them multiple times, with emergency stop software and padded surroundings to limit surround damages.
- Simplicity
    - These components will be put together by teams with a vast range of price ranges and knowledge levels. Therefore, the designs should consist of few components and have alternative designs, so that teams can choose the design that is best for them.

4.2 Interface Testing

Within our project, the main units are individual weeks of curriculum. Therefore, the main interface testing is ensuring that each week does not include any topic that has not been taught yet, as well as ensuring that the pace is neither too fast nor too slow. This is a simple lesson check, and errors are mostly prevented by the creation of the outline ahead of time. However, it is well-known in engineering that even the best preventative measures are not always sufficient. Therefore, with half of the team working on each curriculum, the other half of the team will be reviewing the curriculum to ensure that the flow does not have illogical jumps or knowledge leaps. This will be done by blind testing, where the team members will attempt to complete the curriculum themselves in a short period of time (recognizing that it will take the students longer than it will take a team member to complete).

The other primary section of the curricula is the final challenge. For this, the interface testing is in place to ensure that all of the individual and combined portions of the challenge *can* be completed with what students have learned in the lessons.

4.3 Acceptance Testing

As the curricula mostly consist of non-functional requirements (and the functional are encased within the non-functional), the primary acceptance testing is ensuring that the curricula meet the expectations of the IT-Adventures group. The vast majority of the testing is done in 4.1, the Unit Testing portion, so it is expected that this should mostly pass. During our bi-weekly meetings with the client, we will be reviewing the general ordering within the curricula and requesting feedback. This feedback loop will ensure that we stay within the limitations and expectations placed on us by the client.

4.4 Results

With the project wrapping up, the team has successfully implemented two sets of curriculum – the Robotics and the Smart-IT curricula. With the teaching portions heavily weighted towards the first semester, the team has been able to solicit feedback and update the lessons to answer frequently asked questions, as well as identify blind spots. Looking ahead, the team has also proposed the final challenge in both areas and is finalizing the basic design. These have been designed with modularity in mind, which will allow the client to easily edit them as more capabilities become available with the RVR.

In addition, the team previously decided upon the main Robotics platform consisting of the micro:bit, littleBits topper kit, and the Sphero RVR. The Smart-IT team decided that the platform would be the Raspberry Pi 3 and the Sphero RVR. While there have been supply chain disruptions, all participating teams this year were able to acquire the hardware to compete.

5 Implementation

The team has successfully implemented both curriculums. This includes the lessons and individual challenges for each, alongside the final challenge in both instances. They have also been presented to the client, both in the completed form and with the templates for future accessibility.

In both curriculums, the lessons themselves were laid out early on. After the schedule was approved by the client, lessons were developed in a chronological order, ensuring that they would be completed by the time that students would be starting them. This included testing the lessons (as described in Section 4) to make sure that they met the requirements. During implementation, the challenging component was creating valid challenges and lessons that fit into the time requirements, which led to the two curriculums going in vastly different directions (format-wise). These have been future proofed by providing the master copies to the client. This will allow them to update the curricula as necessary, as the technology continues to improve.

Similar to the development of the early curricula, developing the final challenge followed an iterative process. In essence, this has meant that the team has, from scratch, been proposing and improving upon a final challenge for both curriculums. Over a two-month period, the team started by brainstorming ideas for the RVRs and discussing them with the client. After determining likely paths, the team worked to identify shortfalls and limitations with the RVRs that would prevent certain challenges from being implemented.

6 Updates From Semester 1

In the spring semester, the team was able to mostly complete the Robotics curriculum and outline the Smart-IT curriculum. This included identifying the necessary hardware components and overarching software goals. Since that point, both curriculums have been completed. Students have been working through both curriculums, showing that they are indeed working. The last major step – as the curriculums had been started in the spring – was the final challenge in both areas. As described in Section 5, implementing the final projects has gone well. With some limitations on sensors and expected knowledge, the team has worked hard to create final projects that are within the bounds of the students' abilities. However, this has also been planned by allowing the client to continue to modify the challenge, with some advised routes to make the challenges easier or harder (depending on feedback from the students). This will allow them to easily modify it to fit their needs. Overall, the team has done quite a bit of work to reach this point, and has shown it by completing the project.

7 Closing Material

7.1 Conclusion

Overall, the team completed the curricula, meeting the primary goal. Each month has weekly lessons and a monthly challenge, with the second semester being more wholly dedicated to larger challenges. The final challenges are also outlined and defined. This will let IT Adventures update them as appropriate, especially as more technology options become available. Overall, this design is also more productive than the other primary option, which only had monthly lessons and was much less structured. Success has already been seen with students learning from the lesson. The team is excited for the spring, in which students will tackle the final challenge, even as team members are technically separated from the project.

7.2 References

"CS Standards," *Standards | Computer Science Teachers Association*. [Online]. Available: https://www.csteachers.org/page/standards. [Accessed: 23-Apr-2021].

W. Hoffman, "PK-12 Computer Science," *Iowa Department of Education*, 2021. [Online]. Available: https://educateiowa.gov/pk-12/instruction/computer-science. [Accessed: 23-Apr-2021].