Curriculum to use the Sphero RVR in IT-Adventures

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: March 7, 2021/Version 1.0

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

Summary of Requirements

List all requirements as bullet points in brief.

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

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 4, 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.

1.2 Problem and Project 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.

Through the IT-Adventures curricula using the Sphero RVR, students of all skill levels will learn how to program and debug different programs while tackling fun challenges. This curriculum 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

- The kits must be cheap and easily supplied.
- The challenges must ramp up in difficulty slowly / at a standard pace.
- The final course must be reliable and easy to put together.
- The curriculum 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.

1.5 Intended Users and Uses

The intended users are high school students. Small student groups will be formed to tackle the challenges presented throughout the curriculum, while learning the basics of coding. 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). This will teach general 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 will be third and will be a focused curriculum on a lower level of coding for the Sphero kits. This will teach students more basic coding skills and start to adapt them to different challenges using the Sphero RVR. Each month will contain at least one presentation and multiple coding challenges and examples. As this curriculum is intended to directly follow the general curriculum, it will be delivered in the middle of October, so that students can start using it in December 2021.

The Smart IT curriculum will be fourth and will be a focused curriculum on a higher level of coding knowledge for 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. As this curriculum is intended to directly follow the general curriculum, it will be delivered in the middle of October, so that students can start using it in December 2021.

Finally, the final challenges for both Robotics and Smart IT will be delivered. These challenges must be in the scope of the curriculum. However, it must still pose a significant enough challenge to be the “finale” for the season. This will be delivered at the end of the fall semester, December 2021.

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 4, 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 4 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 be the same. 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 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.

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

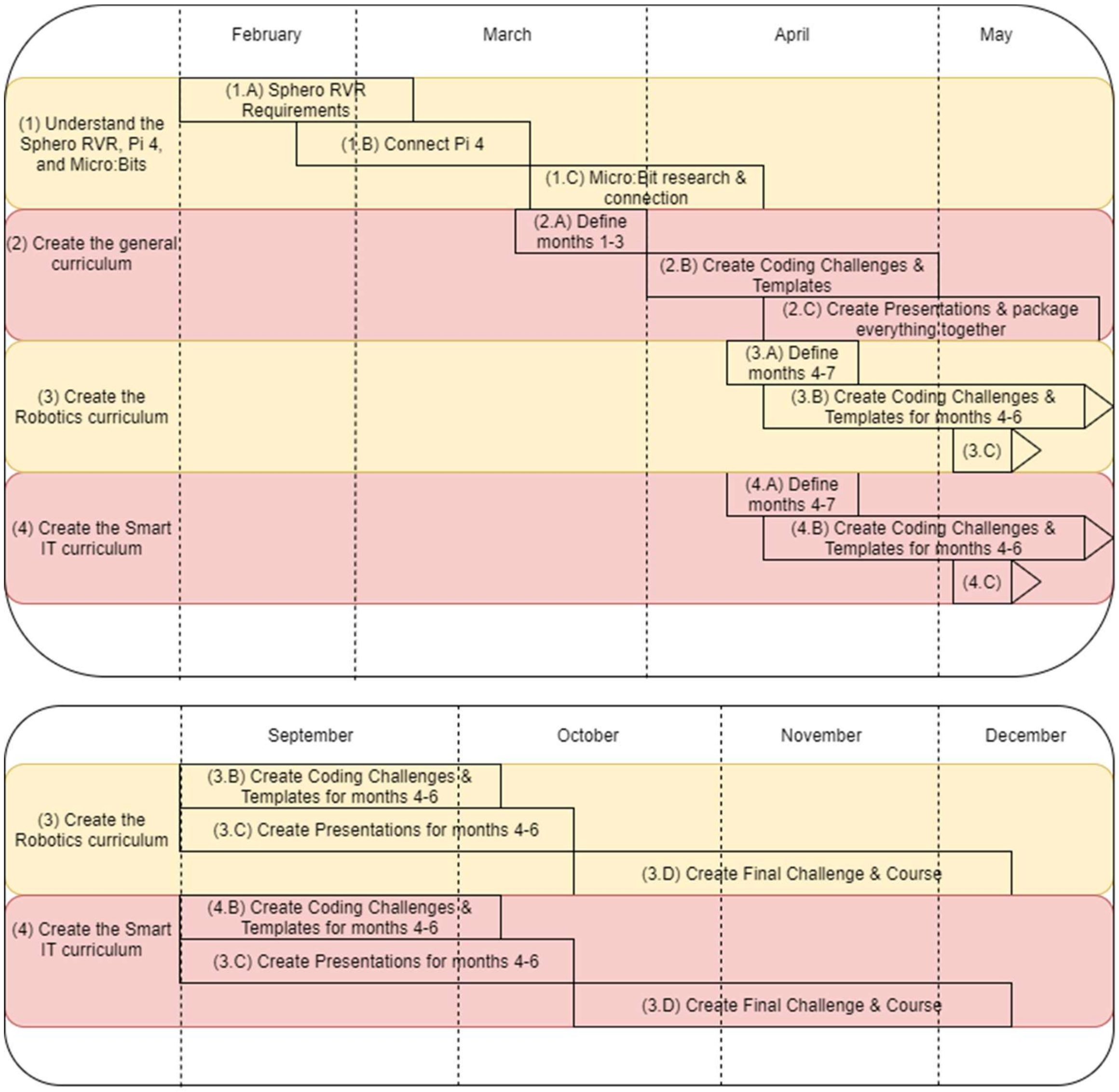

Figure 1: Gantt Chart of Project Timeline

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 4, 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 4 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 4 (and accessories), and the micro:bit kits. 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.

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 curricula. 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, it was showing its age. 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 skillset. At the end of the year, they are then prepared for the competition.

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 opposite to SmartIT, 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 is a better fit.

3.5 Design Analysis

Overall, the proposed design from 3.3 has worked thus far. 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. The team is still working on expanding the details of each week, though, and it is expected that this will continue over the next several weeks. The primary design for SmartIT is in development, and the outline and primary design for Robotics is complete.

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 a Waterfall 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).

3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

4 Testing

4.1 Unit Testing

Our project is rather unique in that there are not 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

4.2 Interface Testing

Within our project, there are no specific interfaces to be tested. Each "unit" is an individual piece of curriculum, so the only interface testing is ensuring that each week does not include any topic that has not been taught yet. 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.

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 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

As the team is still early in the testing phase, there are no results to report on yet.