Curriculum to Use the Sphero RVR in IT-Adventures.

sddec21-14: Dakota Berbrich, Aaron Goff, Noah Berkland, Nolan Jessen
Advisor: Doug Jacobson
Client: IT-Adventures

Project Plan

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 and Functional Requirements

Solution

Create a set of cheap, engaging curriculum for schools to use that will teach students basic programming skills, starting from any skill level.

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.

The kits must be cheap and easily supplied.

Other Constraints and Considerations

The challenges will be programmable within the period of a month, so they cannot be too challenging to implement and test

Most of the students will have little to no programming experience, so *everything* must be taught (or have an option to be taught)

For students who have a little more experience, there will be a secondary challenge that is slightly increased in difficulty

Each week, students are only expected to work 1-2 hours

The challenges should build upon each other, preparing students for the final challenge in April

Market Survey / Related Work

FIRST

Similar in promoting STEM for K-12 students, these programs are much more time-intensive and are much more expensive for schools to participate in

LEGO® Robotics

In comparison with the RVR, the LEGO® series is much more customizable. However, this platform is older and cannot accommodate the Smart-IT program well

Potential Risks and Mitigation

1. Creating a feasible final challenge
   - A final challenge will require multiple robots on the same playing field at the same time, a several-month-long development period, and an interesting purpose
   - Subject to extra reviews and feedback, this challenge will be thoroughly examined to mitigate as much risk as possible

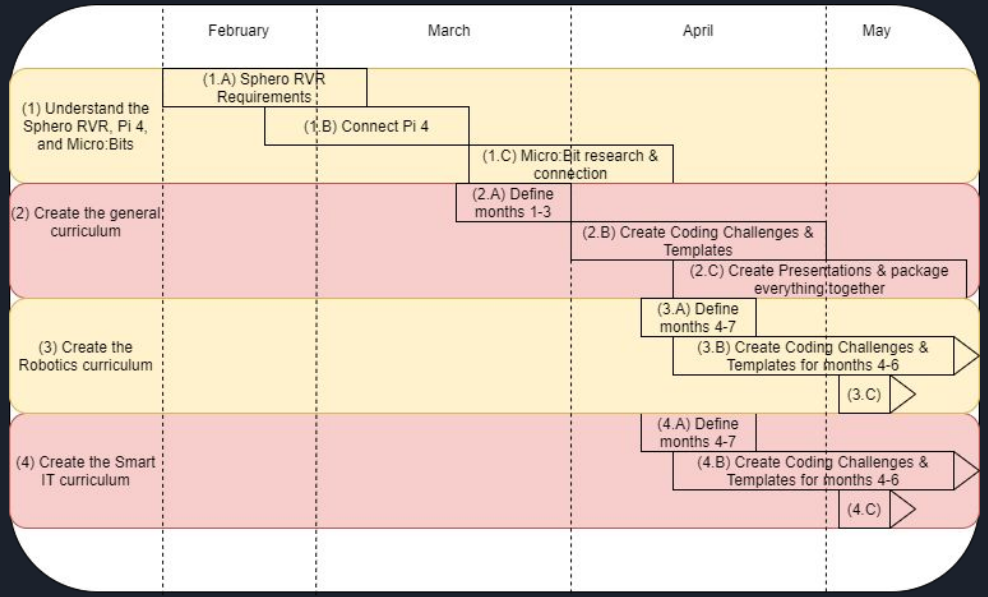
2. Making multiple systems work together reliably
   - Programming to hardware, there are several systems in both programs that must work together reliably
   - Pre-selected systems remove some risk
   - Working with Sphero to iron out any common errors that we run into, in hopes of preventing students from encountering them

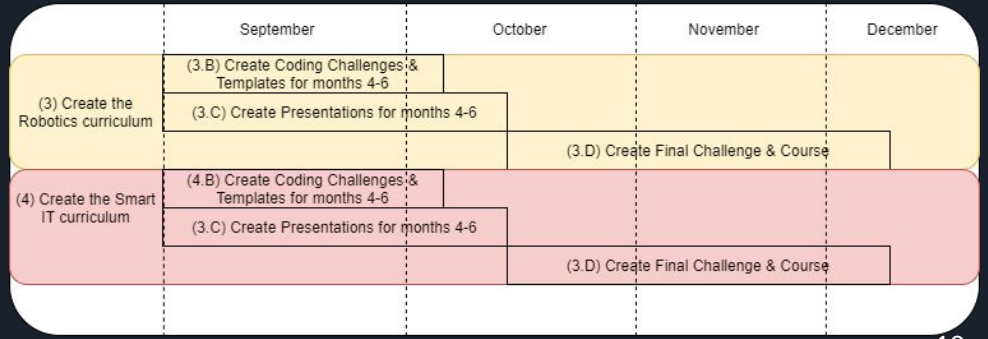
Resource Estimate

Robotics	
Sphero RVR	$250
littleBits Topper Kit	$125
micro:bit	$20
Total:	$395

Smart-IT	
Sphero RVR	$250
Raspberry Pi 3B+	$40
Wires and Mounting	$10
Total:	$300

Project Milestones





System Design

Functional Decomposition

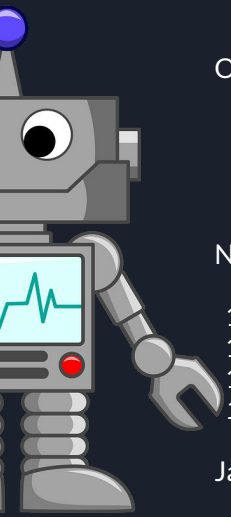
Two sets of curriculum: Robotics and Smart-IT

These are split into weekly curricula, with monthly challenges and a final challenge

Each week contains a new topic (i.e. loops)

Monthly challenges should review lessons from current month and previous months

Detailed Design



September - Initialization and Familiarization
1. Setting up the RVR and micro:bit connection
2. Moving the RVR
3. Basic micro:bit I/O
4. Variables

October - Boolean Logic & Loops
5. If Statements / Conditionals
6. While / For Loops
7. Arrays / For Loops
8. *Challenge of the Month*

November / December - littleBits Sensors & Catch Up
9. Button and Buzzer [Digital I/O - littleBits version]
10. Servo and Slide Dimmer [Analog I/O]
11. Remote Trigger & Latch [Remote I/O]
12. Proximity Sensor
13. *Challenge of the Month(s): Factory Robot*

January - April - Competition Prep

April - Competition
14. Final Challenge

September - Introduction and concepts
1. Setting up the RVR and Pi connection
2. Introduction to Smart IT, Python, and the Pi
3. If Statements / Conditionals
4. While / For Loops

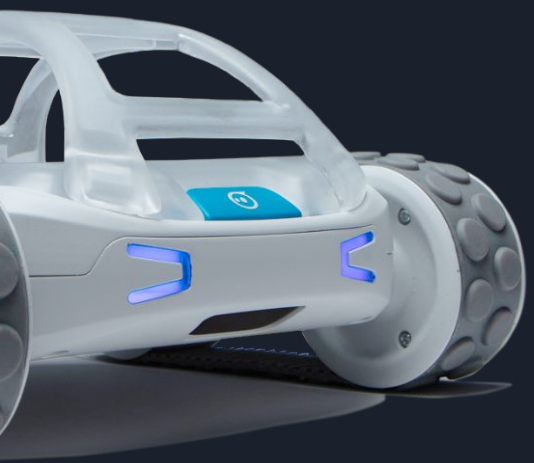
October - Arrays, Functions, and Files
5. Arrays & Strings
6. Functions
7. Import/Export Files
8. *Challenge of the Month*

November / December - Debugging, Dictionary, Classes
9. Fun with Debugging
10. Dictionary
11. Classes
12. *Monthly Challenge*
13. *Winter Themed Challenge (advanced concepts)*

January - April - Competition Prep

April - Competition
14. Final Challenge

Hardware Platforms



Sphero RVR

This durable robot can be programmed and combined with multiple platforms, making it a perfect fit for the program

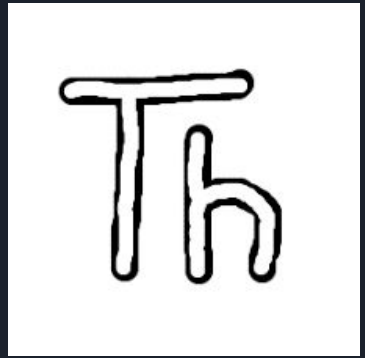
micro:bit and littleBits

A simple, easy-to-control microcontroller, the micro:bit interfaces directly with the RVR and the littleBits topper kit to create a flexible, interactive robot.

Raspberry Pi 3

The flexibility of a full computer in the size of a deck of cards. Easily attached to the topper plate and GPIO pins the Pi gives us the full functionality of the RVR's api including the motors, LEDs, and, unlike the micro:bit, the sensors.

Software Platforms



MakeCode

To develop the micro:bit code, students will use the Microsoft MakeCode IDE. This can download programs directly to the micro:bit and has a built-in simulator. This IDE also directly connects to the RVR SDK, making it a fantastic tool for our development.

Thonny

On the Pi students will use Thonny, a preinstalled IDE designed to be used by beginner python programmers. It offers all the basic features and tools for development and debugging all while remaining lightweight enough to be used on the Pi. Running the code on the Pi will also run the code on the RVR without need for flashing the code to the RVR's hardware.

Test Plan

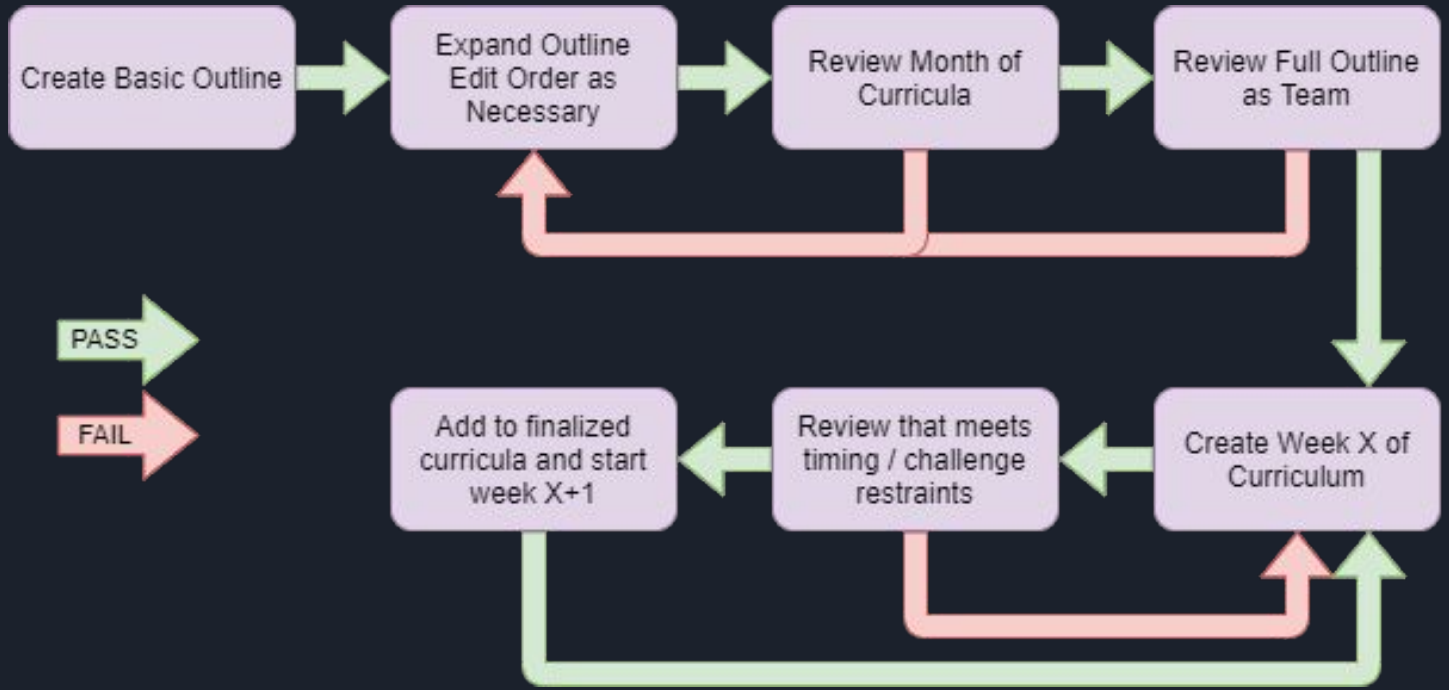
Uniquely, the deliverable is not a raw piece of hardware or software, but is instead curricula. This makes testing a bit unique. Testing should reveal any of the following flaws:

- Material that is not yet covered or learned
- Challenges are too long or too short
- Challenges are not possible with primary codebase

Above risks are mitigated and assessed in the following ways:

- Creating an extensive outline
  - Review that outline achieves general goals and does not skip lessons
- Creating all code challenges while creating the lesson, making both a solution key and ensuring that they are possible in the limited time
- Having an outside reviewer at every step of the process

Test Plan - Diagram



Prototype Implementations

Prototypes of first lesson of Robotics faced following challenges:

- How in-depth to go
- How much code to provide
- What platform to use
- How much guidance to give

Secondary Learning Challenge

Lets ramp up the difficulty a bit, can you figure out how to make the RVR move in the pattern below?

Makecode Project

To make a new program for the micro:bit and RVR, follow these steps:

1. Open MakeCode and select New Project
2. Name the project according to the lesson, in this case "Robotics Lesson 1", and click create
3. Import the RVR controls (Software Development Kit) by clicking Advanced and then Extensions
4. Make sure your computer is connected to the internet and search for RVR in the search bar.
5. Click on the result "sphero-sdk-microbit-mak..."
6. Start coding!

Conclusions

Current Project Status

Robotics Curriculum

- Outline and primary curriculum for the first semester is completed, including challenges and code examples
- Over the next week, the finalized written lessons will be delivered for these months

Smart-IT Curriculum

- Outline and primary curriculum for the first semester is completed, including challenges and a few early mock ups of lessons
- Over the next week, code examples and finalized written lessons will be created.

Task Responsibility & Contributions of Team Members

Team is split to tackle the two curricula separately:

- Robotics
  - Dakota has been working on both Robotics and overall team direction
  - Nolan has been working on Robotics and general documentation / project generation
- Smart-IT
  - Noah has been working on Smart-IT and external communication, reaching out to other groups when we need feedback or assistance in debugging a component
  - Aaron has been working on Smart-IT and ensuring that the website has the resources for schools and teams to effectively work

Second Semester Plan

Heading into the second semester, the main goal will be designing and finalizing the final project for both areas. This will be done in a few key steps:

1. Brainstorming
    - Generate as many ideas as possible, selecting the best few
2. Choosing an Idea
    - Select the most creative yet feasible project
3. Finalize Board
    - Create an arena that can be durable and easily set up
4. Test and Finish
    - Continuously test that the idea is feasible, and finalize all the components

Thank you for listening!

Questions or Concerns?