IOWA STATE UNIVERSITY

Department of Electrical and Computer Engineering

Curriculum to Use the Sphero RVR in IT-Adventures

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

Presentation Overview

- Problem Statement
- Robotics
    - Curriculum Overview
    - Lessons and Final Challenge
    - Design Challenges
- Smart-IT
    - Curriculum Overview
    - Lessons and Final Challenge
    - Design Challenges
- Conclusion

Problem Statement

IT Adventures seeks to inspire kids to pursue STEM and IT-related fields through three programs:

- Robotics
- Smart-IT
- Cyber Defense

In the last year, they updated their kits for Robotics and Smart-IT, so we've been tasked with creating new curriculum and challenges for the Robotics and Smart-IT programs.

IT-ADVENTURES
MAKING IT FUN

First Curriculum Area

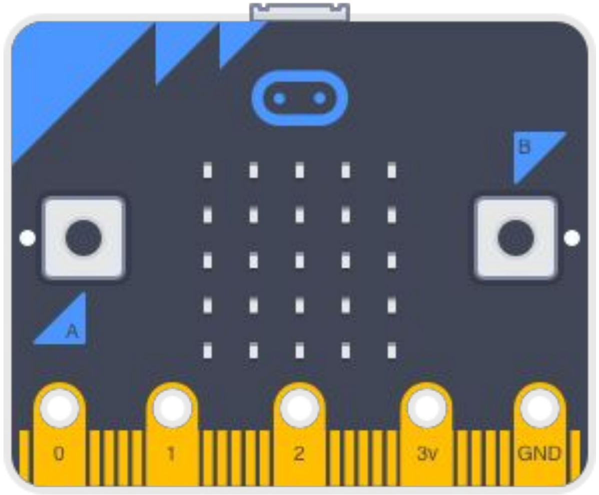
Robotics

Robotics Curriculum Overview

Curriculum generally focused on very basic programming techniques; made for students who have never programmed.

Objectives:

1) Teach basic programming concepts
2) Engage students in variety of activities to explain concepts
3) Prove that new platform works and create final challenge to utilize it



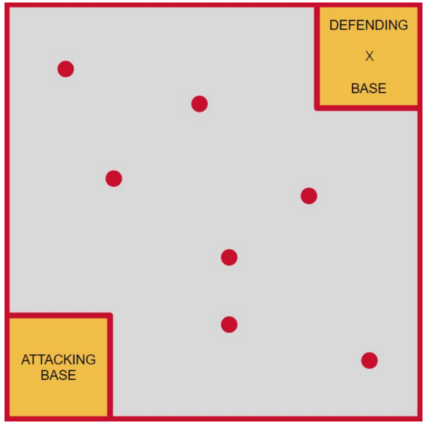
Lessons and Final Challenge

Lessons teach basic programming concepts, and have a standard format:

- Conceptual overview
- Code example
- Primary and secondary challenge

Final challenge is an application in all that has been learned, with creative design elements and requirements:

- Comprehensive review
- Semester-long design period
- Competitive and engaging



Lesson Demonstration

Link to Video: https://youtu.be/ocpYhCETXh0

Design Challenges and Solutions

Problem: Lessons are feasible for small classes or home schooled students to complete on their own without previous subject knowledge

Solution: Lessons ramp in difficulty, build on previous topics, and have examples to walk through vital concepts

Problem: Due to past limitations in memory, functionality of the RVR is limited with the micro:bit as a controller (embedded IR proximity, color, and light sensors)

Solution: Lesson plans offer expandability into these areas if updates happen to the SDK by teaching students to use the supported devices and touching on different types of sensors

Second Curriculum Area

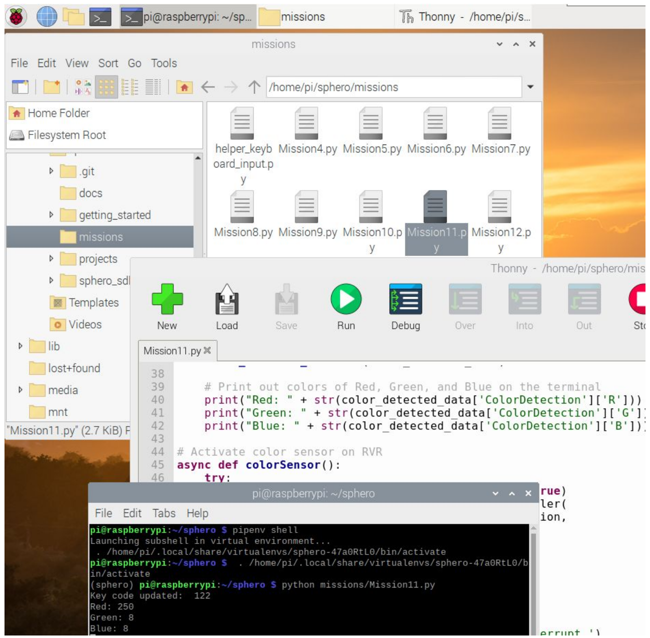
Smart-IT

Smart-IT Curriculum Overview

Curriculum is focused on intermediate programming techniques; and is made for students are familiar with object oriented programming.

<u>Objectives</u>:

1) Teach Linux & Python concepts
2) Engage students in variety of activities to explain concepts
3) Prove that new platform works and create final challenge to utilize it



```python
    # Print out colors of Red, Green, and Blue on the terminal
    print("Red: " + str(color_detected_data['ColorDetection']['R']))
    print("Green: " + str(color_detected_data['ColorDetection']['G']
    print("Blue: " + str(color_detected_data['ColorDetection']['B'])

# Activate color sensor on RVR
async def colorSensor():
    try:
```

```
pi@raspberrypi:~/sphero $ pipenv shell
Launching subshell in virtual environment...
 . /home/pi/.local/share/virtualenvs/sphero-47a0RtL0/bin/activate
pi@raspberrypi:~/sphero $  . /home/pi/.local/share/virtualenvs/sphero-47a0RtL0/b
in/activate
(sphero) pi@raspberrypi:~/sphero $ python missions/Mission11.py
Key code updated:  122
Red: 250
Green: 8
Blue: 8
```

Lessons and Final Challenge

Fall Lessons (Python):

Week 4-13: Variables, Inputs, Outputs, String conversions, Conditionals, Iterations, Lists, Strings, Functions, Modules, Objects, Classes, Dictionaries, and JSON.

Fall Lessons (Sphero):

Week 3-4: Setting up Raspberry Pi & Linux
Week 8-13: Sphero RVR Input and Output functions.

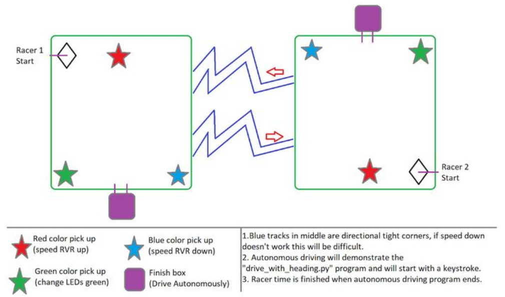
RVR Inputs: Color Sensor, Keyboard Controls, Infrared Sensor
RVR Outputs: LEDs, Motors

Spring Lessons (Sphero only):

Week 1-11: Keyboard control, speed control, rotation control, LED control, color sensor, color sensor on demand, autonomous driving, and combining lessons.

Final Challenge:

A head to head race utilizing all learned functionality. Students will have 2 hours to optimize their code to be used for this challenge.


Racer 1 Start
Racer 2 Start
Red color pick up (speed RVR up)
Blue color pick up (speed RVR down)
Green color pick up (change LEDs green)
Finish box (Drive Autonomously)

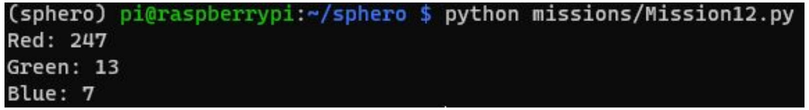
1.Blue tracks in middle are directional tight corners, if speed down doesn't work this will be difficult.
2. Autonomous driving will demonstrate the "drive_with_heading.py" program and will start with a keystroke.
3. Racer time is finished when autonomous driving program ends.

Lesson Demonstration

Mission 12:

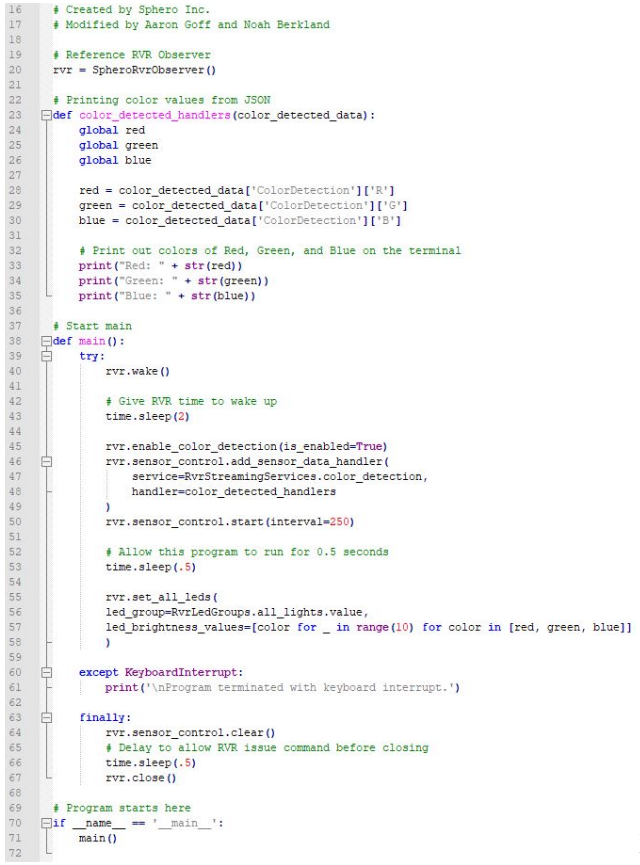
1. Wake Sphero RVR.
2. Activate color sensor pickup.
3. Print color sensor values.
4. Change all LEDs to the color the sensor detects.

Code running in Linux:



```
(sphero) pi@raspberrypi:~/sphero $ python missions/Mission12.py
Red: 247
Green: 13
Blue: 7
```

Image of example code:



```python
# Created by Sphero Inc.
# Modified by Aaron Goff and Noah Berkland

# Reference RVR Observer
rvr = SpheroRvrObserver()

# Printing color values from JSON
def color_detected_handlers(color_detected_data):
    global red
    global green
    global blue

    red = color_detected_data['ColorDetection']['R']
    green = color_detected_data['ColorDetection']['G']
    blue = color_detected_data['ColorDetection']['B']

    # Print out colors of Red, Green, and Blue on the terminal
    print("Red: " + str(red))
    print("Green: " + str(green))
    print("Blue: " + str(blue))

# Start main
def main():
    try:
        rvr.wake()

        # Give RVR time to wake up
        time.sleep(2)

        rvr.enable_color_detection(is_enabled=True)
        rvr.sensor_control.add_sensor_data_handler(
            service=RvrStreamingServices.color_detection,
            handler=color_detected_handlers
        )
        rvr.sensor_control.start(interval=250)

        # Allow this program to run for 0.5 seconds
        time.sleep(.5)

        rvr.set_all_leds(
        led_group=RvrLedGroups.all_lights.value,
        led_brightness_values=[color for _ in range(10) for color in [red, green, blue]]
        )

    except KeyboardInterrupt:
        print('\nProgram terminated with keyboard interrupt.')

    finally:
        rvr.sensor_control.clear()
        # Delay to allow RVR issue command before closing
        time.sleep(.5)
        rvr.close()

# Program starts here
if __name__ == '__main__':
    main()
```

Design Challenges and Solutions

Problem: Create a curriculum that allows multiple schools and home-schooled students to easily follow along and learn with little/no assistance.

Solution: Done by utilizing Powerpoint slides, pictures, video, and prewrite working code for challenges for students to reference if/when they get stuck.

Problem: Create an environment where students are able to proficiently use Linux and remote connection to Raspberry Pi.

Solution: Done by adding onto the IT-Adventures website and creating a step-by-step guide with example pictures and a video walkthrough.

Problem: Problematic instructions little/no reference code due to using relatively new educational equipment.

Solution: Solved through direct communication with Sphero RVR development team, and a lot of trial and error.

Results and Next Steps

Conclusion

Results

1) Curricula completed successfully, being run now

2) Robotics lessons easily transferable to python for advanced students

3) Tutorial videos and walkthroughs implemented for Smart-IT

4) Lesson have been executed and feedback from schools has been implemented

Next Steps

- Continue to implement feedback from course instructors to make the curricula more approachable

- Make a repository of programming examples for each lesson plan to reduce confusion and confirm student solutions