Outreach Training

Girls Inc. Eureka Workshop

Profs. Sunghoon Lee and Phillipa Gill

College of Information and Computer Sciences
UMass Amherst

Outreach Training Agenda

1. Introductions
2. Event Description and Goals
3. Event Logistics and schedule for the day
4. Volunteer role, goals and best practices
5. Scratch Crash Course
6. The *Play It/Code It* activity

Event Description

- The event is held at the Design Building.
- 15 high school girls (12 – 18 years old) and teachers from Holyoke area will participate.
- Long-term goal is to contribute to broadening the participation of women in the technology workforce.
- Immediate goal is to give girls a sense of education and career opportunities in engineering and CS (STEM in general).
- The goal of our activity is to provide a fun and engaging introduction to basic ideas in computational thinking, programming, and embedded systems.

Event Logistics and Schedule

- Setup (8:50-9:50)
- Activity Sessions (9:50-11:50)
- Teardown (11:50-12:20)
- Room configuration

Logistics and Schedule: Setup

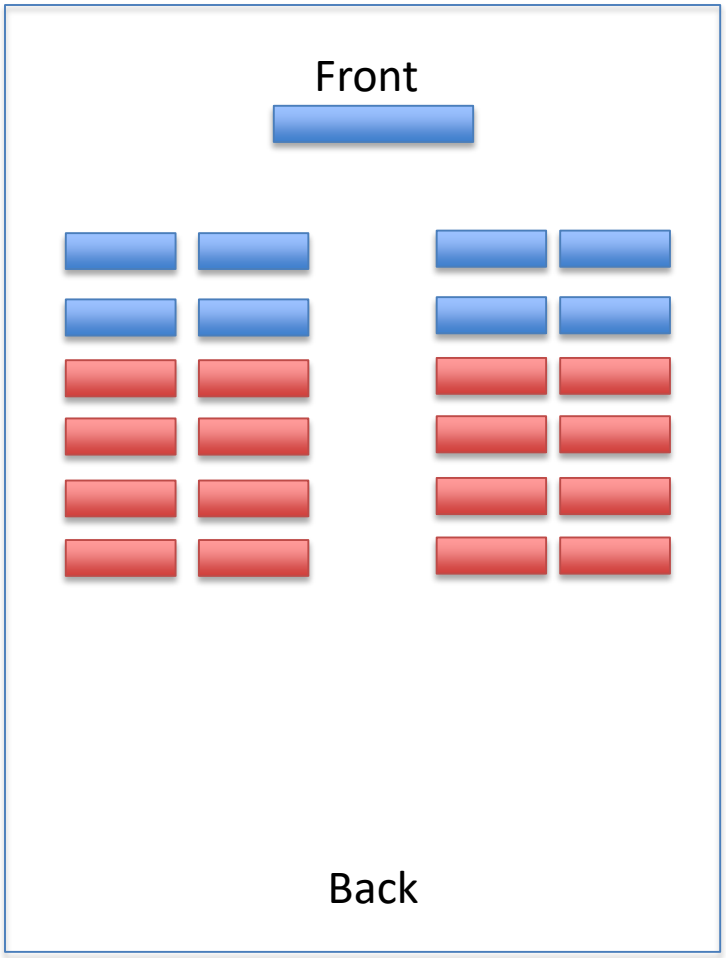
Setup Team 1

- **8:50:** Meet at the Design Building room 162.
  Laptop will be available in the room.
  Verify arrangement of tables, chairs, power, AV.
- **8:55-9:40:** Unpack, plug in, boot-up, log-in, and install drivers & SW on 8 laptops. Verify all laptops working.
- **9:40-9:50**: Setup team break. Get name badge at registration table.

Logistics and Schedule: Activity Sessions

- **9:50-10:00**: Introduction & Demo to students
- **10:00 – 11:50**: Activity Session
- **11:50-12:10**: Remove code from laptops, power down, pack up all gear.
- **12:10-12:20**: Return laptops (??)

Logistics and Schedule: Room Configuration



- We will use the first 4 tables.
- You will be assigned to cover 2 teams (4 girls).

Volunteer Role and Goals

- **Role:** Facilitate, guide, and coach.
- **Goals 1:** Ensure all the girls have a **positive experience** with the activity.
- **Goal 2:** Keep the girls **engaged and on-task** for the full session.
- **Goal 3:** Try to **keep frustration levels low**.
- **Goals 4:** Convey basic material about **computational thinking** and programming including how to break down a problem into a sequence of steps, and how to use basic control flow (looping and conditionals).

Volunteer Best Practices

- **Break the ice**. Introduce yourself. Ask some questions about what school they are from, if they have programmed before.
- For questions about how to complete the exercises, try to **guide them in terms of how to think about the problems**. Try prompting questions like: “How could you break this down into steps?”, “Can you think of what the first step could be?”, “What should the next step be?”
- You can **directly answer any questions about the ArduBlock interface**: where blocks are; how to drag, drop stack, rearrange, remove blocks; how to change between the sprites to add code to them.

Volunteer Best Practices

- For questions about what blocks do, or what a piece of code might do, **try to encourage experimentation** with prompts like “why don’t you try it out and see what happens.”
- When an initial solution doesn’t work, **encourage them to debug**: figure out where the problem is and try something else.
- You need to **watch out for frustration levels** if a group is having trouble, particularly groups with no prior experience. You should try **encouragement first, then give hints, then use redirection** (“why don’t you go on to the next exercise and come back to this one later).

Volunteer Best Practices

- **Let advanced groups work independently** to the end of the main exercises and get them to demo their solutions for you. If they didn't get the most efficient solutions, encourage them to think a bit more.
  - Maybe teaming up those students who have programming experience?
- For groups that complete the main exercises, they can continue to modify the program we provide by adding enhancements. Get them to explain what they want to try, and check in periodically. They can do anything within the structure of the game. **Suggest enhancements and challenge them with harder tasks until time is up.**

Volunteer Best Practices

- **Ask for help** if you get into a situation that you're not sure how to deal with, or a question you don't know how to answer, ask for help from Ivan or Phillipa.
- As a general rule, **volunteers do not touch laptops during hands-on sessions**. Your goal is to talk participants through problems and guide them to come up with solutions on their own. If something outside the activity comes up (like a software crash or a hardware issue), then you can take over the laptop to get them back on track with the activity as quickly as possible.

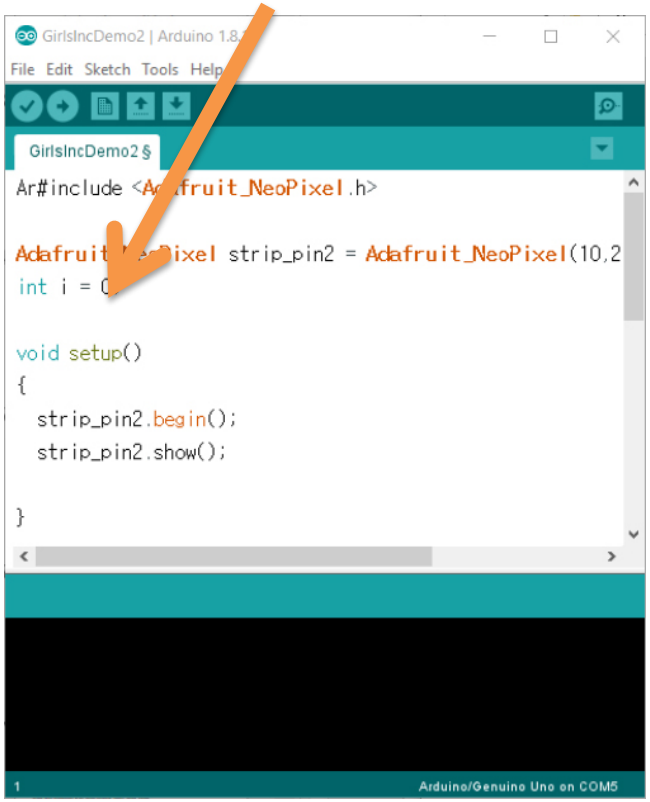
Arduino & ArduBlock

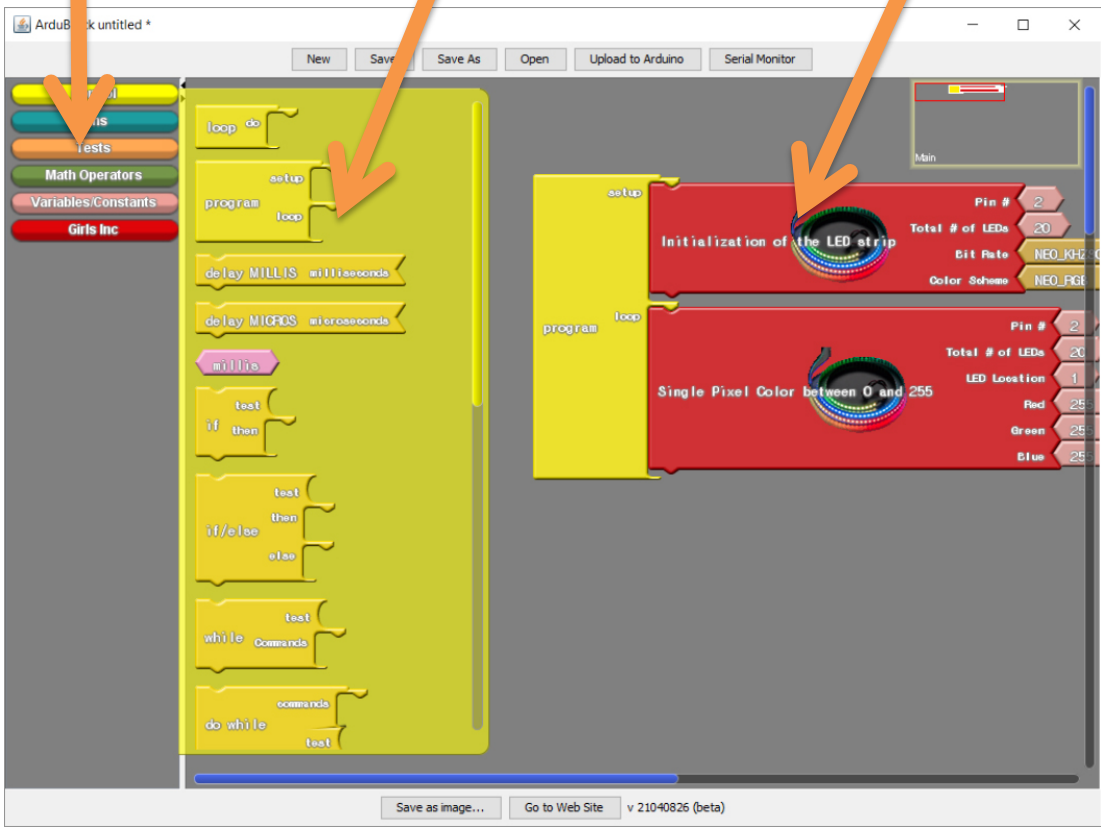
Arduino IDE

Block Categories

Blocks

Code





ArduBlock

- An Arduino “add-on tool” that is designed in JAVA based on MIT’s Scratch platform.
- It simply provides a graphical UI, which converts the code into the Arduino code
  - You will be able to see the converted “Arduino Code” every time you compile the “Block Codes”
- Programs are created by dragging command blocks from the blocks panel to the code area.
- Blocks are arranged in categories. We will mostly be using “control blocks” and custom blocks that we designed for “Grils Inc”.

ArduBlock

- Blocks snap together like puzzle pieces in the code area to form instruction sequences.
- When you click on a block and drag it, it drags all blocks below it in the stack.
- You delete blocks by dragging them out of the code area to the “Block Category” panel
- You can run individual blocks or stacks by double clicking on them
- To start a program running, click the “Upload to Arduino” button on the ArduBlock IDE.

Activities

- A total of 5 parts with sub-activities and incremental difficulty.
- Part 1: Connecting wires together
- Part 2: Learn how to use ArduBlock
  - Activity 1: upload the code to Arduino
  - Activity 2: changing color of 1 LED
  - Activity 3: changing the position of the LED
- Part 3: How to get blocks to do more (how to program)
  - Activity 1: manually changing the color of an LED from Red to Green to Blue every 1 second
  - Activity 2: manually changing the position of an LED to the right 4 times.

Activities

- **Part 4**: Learning about loop & variables
  - **Activity 1**: using a loop to move the LED
  - **Activity 2**: using a loop and if-statement to change colors of all LEDs
- **Part 5**: Adding a sensor to control
  - **Activity 1**: Turn on/off the LEDs with a color of choice every 0.5 second
    - Students MUST use only 20 LEDs, otherwise Arduino disconnects itself from the USB, i.e. IDE does not recognize the board.
    - How to solve it: disconnect LEDs from the Arduino board, upload a dummy code, and reconnect the LEDs
  - **Activity 2**: use “analog read” to control the delay.

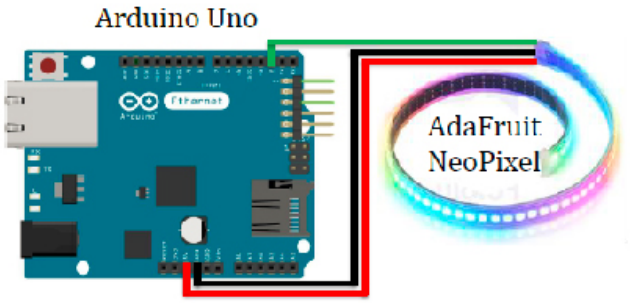
The Activity: Play It/Code It – Instruction Handout

Creative Programming With LEDs

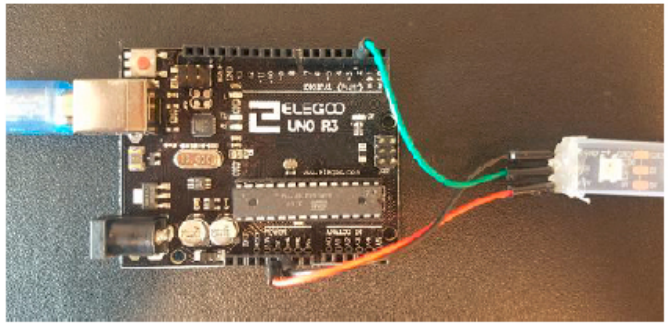
Girls Inc. Eureka Workshop U. Massachusetts July 2017

Part 1: Build it!

Objective: Building LED art often requires setting up custom hardware. For this lab we will use Arduino Uno and AdaFruit Neopixel, which needs to be connected as follows. The **green wire** will need to connect to 2, the **black wire** need to connect the GND pin (one of the two GNDs), and the **red wire** needs to connect to 5V



Your connections should look something like the following



Part 2: Getting familiar with the LEDs

Objective: The objective of this part of the activity is to become familiar with running code and making changes to existing code. You will learn how to change the LED's color and location.

Activity 1: Run your first program!

- Navigate to the Labs folder.
- Open the Lab 1 folder.
- Double click on "Exercise1.abp"

Welcome to your first Arduino program!

To run your program click "Upload to Arduino" in the menu that is above the sketch.

Based on what the LEDs are doing can you guess how the Single Pixel Color block is working?

Can you modify the code to make the LED change color?

Questions?