Irvington Robotics Summer Camp

Nurture Kids - July 15- July 26 Age groups: 7-9, 10-13 9AM - 1PM, 2PM-5PM Monday - Friday —

Genius Kids - July 8-19 Age groups: 7-9, 10-13 2PM-5PM, Monday - Friday

Document Key:

Important generally/for whole camp

Headers

Screenshare/Visuals

Important specific/for teaching curriculum - bolded means even more important

Super Important specific/for teaching curric - bolded means super duper important

Instructors teaching (@ nurture or genius):

- DAY CAMP FULL DAY

- 1 WEEK FULL DAY

-ANY 7-9 year olds

-half day one week camps

MODIFY CURRICULUM to fit the schedule, and share the modified curric w/Ansh to confirm it!

Camp Curriculum/Agenda

Property of **Ansh Verma & Sandesh Shrestha & Patrick Ding(barely)**

Lead Curriculum Developer: Ansh Verma

Collaborators: Patrick Ding (barely a collaborator) and Sandesh Shrestha

Irvington High School Robotics

Ansh -

510-512-0863

asapansh@gmail.com

Sandesh -

510-364-2502

sandeshshrestha02@gmail.com

Patrick

Timeline (by day - 3 hrs per day) -

1) Circuits
2) Circuits/build robot
    a) never take apart bot after build it!
3) CAD (3:00)
4) (30 min to finalize design for part)-CAD, (1:30) intro to arduino/arduino IDE, start intro to programming (1hr)
5) Full Day Intro to Programming w/examples (2hr) - PWM w/examples (LED finish, maybe finish smart car motors - only concern is building the schematic)

    Note - for all the smartcar programs that are applicable(infrared, ultrasonic) - make obstacles/games that kids can use to test that code works

6) Schematics 2,3 (schematic three, first time do smth like this - 2 hrs)
7) Schematics 4/5
8) 6/7 (IR/ultrasonic and line following)
9) Give them 3d printed part, a day to modify their code/circuitry with knowledge they know to create a bot that can….
    a) Explain ur competition
        i) Auton that avoids balls
        ii) Auton that moves through maze
            (1) After maze, then goes to ir remote - again, see if possible for the kids
        iii) IR driver control phase (make sure for each kid, use diff control if IR signals are the same)
    b) give them their 3d printed part,have them attach
    c) Bring field from this day - emphasize driving is super important - practice before just go on - test code before and modify to make driving easiest
10) Competition day!
    a) Avoid balls/ walls - ultrasonic (part of skills phase)
        i) Criteria - robot must be moving (analog write at atleast 100 - or smth like that)
    b) Complete a maze, then go into pushing balls into corners (part of skills phase and driving phase!)
        i) Completing going through the maze is like the autonomous phase!
        ii) If dont complete the maze - never get to the ball pushing part!
            (1) Ultrasonic again!
        iii) IR remote to ultrasonic - see if this even works - schematic wise
    c) Give certificates and awards
    d) Group picture/pack all stuff up as they are taking it home

Timeline/Daily Activities Notes-

Note- programming days are gonna get really tight...so if you finish electronics/cadding material/intro to Arduino curric early...not a bad thing (like half a day early or smth)... obv ONLY if all ur kids understand! This is #1 thing.

IMPORTANT - ur first impression on these kids really matters! It will base how they interact/respect you throughout the camp - ie. if they don't like you, they won't learn. When introducing yourself/this camp/kids to other kids, be excited, engaged, and without being awkward, create a friendly environment where they are serious enough to learn, but relaxed enough to ask questions whenever it pops in their mind/not feel tense.

Small, but important things, like saving files, accessing files, etc - if kids don't know how to do, go over whenever comes up/appropriate - DONT ACT LIKE WTF HOW DO U NOT KNOW HOW TO DO THIS plz - again, ur relationship with them matters

- **FOR ALL BREAKS, TALK TO THEM (if feels appropriate)/HAVE THEM PLAY WII/XBOX**
- **All instructors should teach in the same order, use the same visuals/examples, ask similar questions(cant predict all cuz some questions are obv spontaneous)**
    - **So all kids get as similar an education as possible**
- **Wear APPROPRIATE clothing, make sure you do not show excess skin (SEEMS INFORMAL)**
- **Use APPROPRIATE language (formal language) DON'T USE SLANG**

Way to teach:

- **activities alongside lessons to reinforce knowledge in fun way**
- **Ask questions alongside lesson to keep kids engaged**
- **Use appropriate visuals for each topic**
- **Use screen share for examples**
- **Give helpsheets at beginning of each topic, use as a reference for them while teaching (ie. a little like powerpoint slides they can follow while explaining), and use as a reference for when at home**

Camp Overview-

This summer camp is designed to teach students essential engineering knowledge through building, modeling, and programming a four wheel smartcar!. Students will be be taught basic circuitry, how to use Inventor/Tinker CAD (Computer Aided Design) Software (inventor for the 10-13 age group, tinkercad for the 7-9 age group), how to use the arduino microcontroller system, and the basics of C programming. The end goal of this summer camp is to leave students with fundamental engineering knowledge that will aid them in any future endeavours, and to provide an avenue to enter competitive robotics/technology competitions.

Note - Project Based Learning and Documentation are essential to this camp

- in this camp, we are focused on project-based learning after introducing them to the concepts/curriculum. With project based learning, documentation is a necessity that we will be introducing at various levels in our camp.

Documentation:

10-13 year olds:

Programming (**If needed, teach how to use computer first, ie. how to save files, etc)**

- file documentation (organizing projects under a folder, each version of code as a separate file)
- Programming documentation, writing comments in code to explain what program does
- when introduced to a new concept and try to write a program to practice the concept, comment an explanation of the concept to understand in the future

3D-Design

- File documentation (make folders, fusion will version files (multiple iterations) for you, know how to roll back and recover old versions.)
- Hierarchy for project management - projects, folders, files
- Design history and browser to look how made part, what tools used, so know how build part of come back after a long time and want to keep working on project

Circuitry:

- provide handouts to draw physical circuits -- common circuit symbols practice with this/provide these symbols in helpsheet
- Helpsheet for explanation of concepts in case forget again
- Tinkercad to make circuits/understand series vs. parallel / simulate circuits before make --- save under 1 file for this again, practice good computer organization
- Have them draw out every circuit before physically make for on breadboard/real live, practice circuit common symbols

Mechanical Design:

- Helpsheet with common practices (symmetry, etc)

7-9 year olds:

- Prolly for most, teach them how to use computers, save files, etc
- Same thing as older group, obv less advanced info and lower expectations, go at a slower pace
    - Depending on skill level (assume none), prolly just basic documentation of what basics programs do, nothing else

DAY 1

1. 9-10 (Beginning Introduction)
    - Divide Students into groups as specified on the following google spreadsheet, with one teacher per group for more individualized focus on each student.
        - Put into premade groups based on skill level/parent requests
        - 10 min introduction in teams,
            - **school, name, one interesting thing about them**
                - **(if applicable to group - ie. they don't know eachother before hand)**
                - **Will be working a lot with eachother on engineering projects, so will get to know eachother well**
        - Set up offline LAN network- file downloading/screensharing
        - Discuss Summer Camp Project (make sure to have different explanations for different age groups/camp structures (3 day full day vs. two week half day) as they will be learning subjects to different levels - 7-9, 10-13 groups)
            - **GO OVER TOPICS AS DEPICTED BY NUMBER (1-5)**
            - intro to arduino **(1)**
                - What the whole camp/smartbot is centered around
                - **Screenshare picture of arduino**
                - Explain what it is, what a "Microcontroller" is, essentially the brain of any machine, your programs are the instructions that the brain acts upon
                - Using it in this camp to control robot, manipulate mechanical/circuit/3d printed parts of robot with programming that arduino executes
            - C programming 10-11 **(2)**
                - Essentially a set of instructions write for a computer to do

        - Many programming languages - remind this isn't only language but is good to know
            - Just like regular languages - different words/letters,but similar ideas/instructions can be communicated
                - Computer understand many different types of languages to do commands
            - **Screenshare the many different programming languages**
                - **Screenshare a regular line telling someone to do something, equivalent in C, don't worry about how this line in C communicates the same idea, we will cover later**
            - Use the C programming language for arduino, commonly used in robotics
- CAD (3D Virtual Design) **(3)**
    - (10-13, fusion)(7-9, tinker)
    - Allows us to model our designs virtually before we build them in real life, anticipate any problems before they happen
    - Visualize a design by making it virtually
    - For this camp, we will be virtually building parts, and then 3d printing them to be used on the smartcar!
        - Can create custom parts to solve small design issues, like we will be doing in this camp to optimize your smartcars design for our end of camp game! - dont tell now
    - Show them 3d printer, 3d design you will be 3d printing your own parts!
        - **Screenshare to show the part**
- Basic Circuitry **(4)**
    - Circuits allow our programming/set of instructions to be used for the smartcar
        - Uses programming to send out signals to control/change objects
        - Signals - electricity - what is electricity? Take from the part of this curriculum later if they dont know - can give a review/overview of electricity later then

            - Objects - like turning on and off a motor - **instructors take out and show a motor**
    - For documentation during this camp (necessary in any engineering) **(5)**
        - Please refer to Documentation section above
        - We will be learning/using specific types of documentations for each of the previous aspects **(1-4)** of this camp to keep our projects organized/understandable so if you, or anyone else were to go back to your work after the camp, you could pick off where you left off and understand what you did
            - Even if a whole year later
        - **Screenshare example - programming documentation, all our files of one project/sub-section of a project under a folder, all with appropriate names**

NOTE- Introduce all parts they will be working with (their kits) as they reach the appropriate part of the curriculum

Practice organization with the kits - first time they open it, please emphasize to keep everything as it was when you openned it every time put it back to avoid loosing anything!

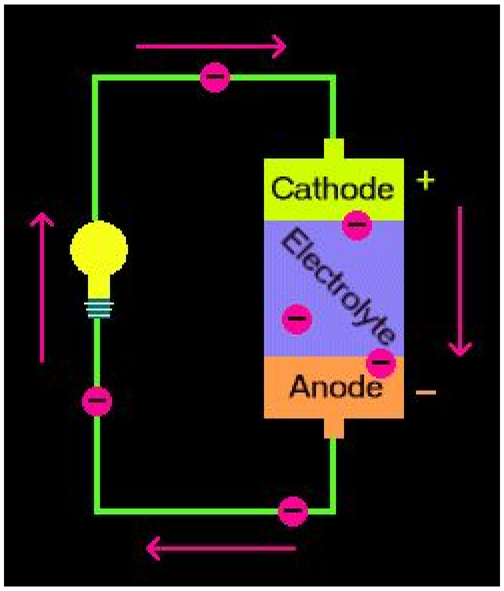
Give them all the things they will be using in this camp:

- **Folders and Labels, tell them to label folder with name and IHS Robotics Camp**
    - **Will have them put their helpsheets in this**
        - **Refer to helpsheets routinely throughout lecture**

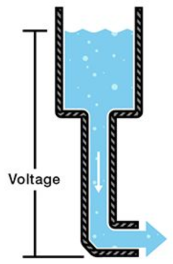
2. 10-12 (Intro. To Circuits)
    - **note - will be using the separately ordered batteries/battery holders for this part of the camp**
        - **We have a premade, simple circuit with battery, LED, resistor(use a 1k resistor and explain why we use this (bc battery voltage - if voltage was more, we would need stronger resistor)) , to demonstrate to them, make/add on to our circuit as they do**
            - **When discuss increase in current decreases voltage, the circuit laws, use a multimeter to show this!**
                - **Ex. can add a resistor, us a multimeter to measure pre and post, show how the current decreases and resistance increases**
    - **Whole explanation, inc water pipe analogy, should take 30 min**
    - **Types of questions to keep for engagement (obv during appropriate times in lecture, when reach topic)**

        - If same amount of electricity flows over smaller period of time, does current increase or decrease
        - Will an increase in voltage increase current? Why or why not
            - Before introduction to Ohms Law
        - If the diameter of the wire in which the electricity is running through decreases, will the current increase or decrease? Why or why not?
            - Water pipe analogy, less area to flow, so flows less
    - **Give them help sheets, have them follow along as explain/give lecture**
    - What is electricity?
        - Movement of electrons to a positive charge
        - Electrons inherently have a negative charge, and opposites attract - the movement of electrons as they “flow” to the positive charge is electricity
            - Show the premade circuit, let them know that part of the battery is positive, part is negative, and when connect, electricity flows
                - LED turning on shows electivity is flowing
    - What is an electron- a fundamental building block of all life - atoms, the fundamental building blocks of everything, have electrons in them, basically a super small particle with a negative charge that helps keep atoms together
    - Electrons following through a wire over a period of time is current
        - Kind of like speed of cars, faster cars and slower cars, faster cars are like faster current, slower cars like slower current
    - Battery Supplies Current to a circuit
        - What is a battery?
            - Take out batteries to look @it
                - Refer to it in ur explanation
            - A battery has 3 main components:
                - Anode - negatively charged ions (have more electrons than protons)
                - Cathode - positively charged ions (have less electrons than protons) - ELECTRON HOLES, THEREFORE LACK OF NEGATIVE, @ POSITIVE
                    - Do NOT confuse with electrons attracting to protons, not true, its electrons attracting to lack of electons!
                - Electrolyte Medium (in between Anode and Cathode, buffer, so nothing can pass in between)
                    - So when battery is not connected to circuits, the build up of electrons/lack of electrons (negative and positive) dont mix

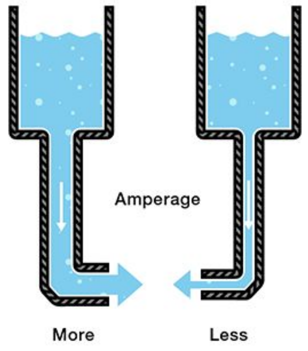
            - If did mix, less current when connects through circuit as less strong negative and positive on each side (less potential difference, but this is an easier way to say it)
        - When a closed circuit is made between an anode electrode and cathode electrode, the potential (or in this case) chemical energy is converted into electrical energy
            - Potential Energy? The energy that CAN be created if a trigger sets a system in motion
                - **an object 10 feet above the ground vs 100 feet above the ground, which one falls faster?**
                    - **100 feet bc more build up - same with a battery, more electron build up on one side is like being higher above the ground, faster fall with more power when hits ground**
                        - **Intro to voltage this way**
                - Chemical Energy? A form of potential energy, based on molecules and their POTENTIAL to react through collision or attraction
            - The higher concentration of electrons in the anode compared to cathode show a build-up of electrons, that, with a trigger, will flow to the empty cathode side
                - The closed circuit is the trigger!
- Voltage is the measure of the potential energy from the anode to cathode
    - Transition from the height example directly above in blue

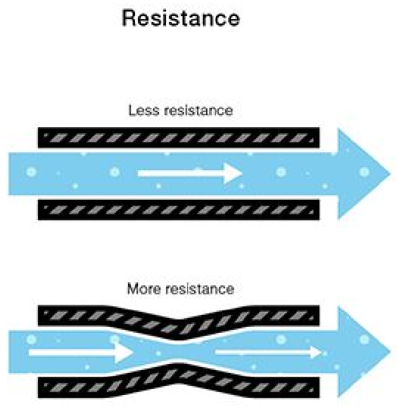


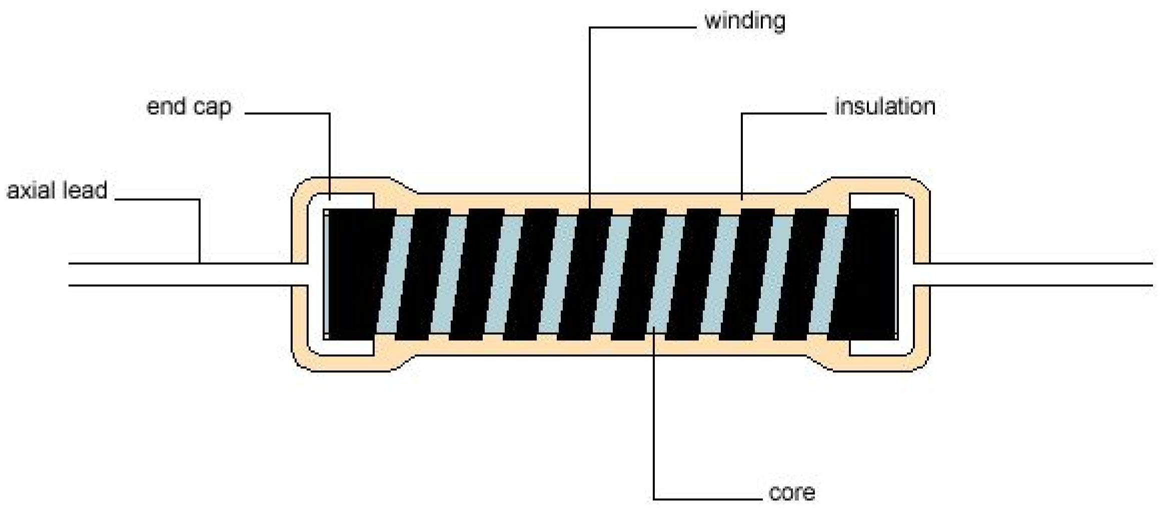
- Water Pipes Analogy
  - Reinforcment of concepts electricity/introduction to a few new ones



  - Reservoir of water can be compared to a battery
    - More buildup of water in reservoir, faster water will flow down pipes due to pressure
      - **What does this mean in terms of voltage and current? -- question to kids**
        - **Ans. buildup in reservoir -like buildup of electrons - more voltage and current**
  - Speed of water flow can be compared to current
  - Wider pipe vs. thinner pipe with same reservoir of water, thinner pipe will flow slower
  - Same with a circuit, smaller wires vs, larger wires with same battery, larger wires have bigger amperage/current
  - Temporary "kinks" in pipes will slow down amperage/current also, like circuits with "resistors," basically temporary kinks in circuit that decrease current flowing in circuit

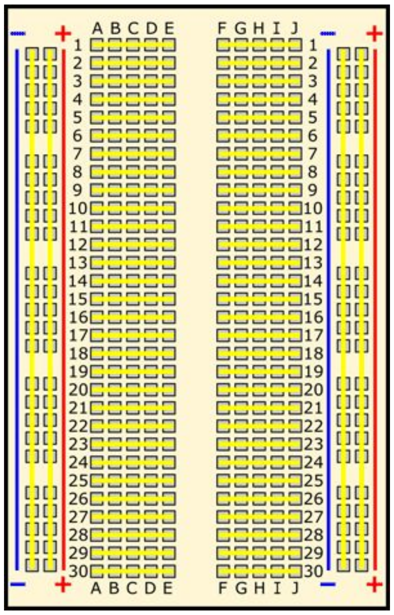






- How do Resistors work? - coils, more coils on resistor, longer the moving electrodes have to travel,
    - Have them take it out and use as a visualization tool
        - Take out resistors to look @ it
            - Refer to it in explanation
    - Moving the anode and cathode farther away from eachother, increase in distance effectively decreases potential energy/voltage between the two as less attraction between the two, therefore slower flow/current
        - **Magnet analogy- two opposites attracting (positive and negative, like anode and cathode of battery) - if farther apart, less attraction between magnets/less likely they will join, for circuits, slower current becasue decreased attraction**
    - Also think of it as more kinks in the circuit, slows down current

- BreadBoards **(time to apply our knowledge!)**
  - **U should already have a breadboard out for your LED circuit, but now have them take it out, refer to it through explanation**
  - Big problem circuits is keeping track of wires
    - Breakboard link many wires together, provide easy way to make/keep track of circuits
  - Rows are internally connected, all on same circuit
    - **Horizontal connections**
  - Left and right (with +/- signs), vertically connected,
  - If you connect the anode and cathodes of a battery to the +/- collumns open/closed?
    - **Open bc still no connection between postive and negative cathode/anode**
    - **Common practice to put batteries on side vertical rails - not neccesary tho**



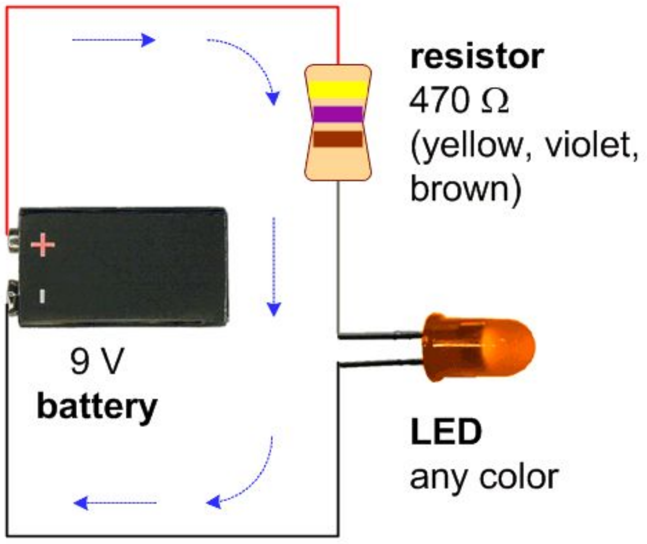
- Creating Simple LED Circuit
  - Create a circuit with LED/Battery, and then LED/Battery/Resistor, show how resistor affects LED brightness and reinforce how to use a breadboard
  - LEDs have a positive, and negative side
    - Longer leg is positive, shorter leg is negative
    - **U already have LED/battery/resistor**
      - **Have them make off urs, explain importance of resistor on not blowing up LED**
      - **Getting about 3 volts from 2 double a's, small LED can only take about 3 volts, so to be safe add resistor**
        - **We will see exactly how it effects the current going into the circuit later (with ohms law!)**
        - **Add 1K resistor next**
      - **Add another resistor to show how dims more, what does this mean?**
        - **Decreased voltage and current- less power, and less speed**

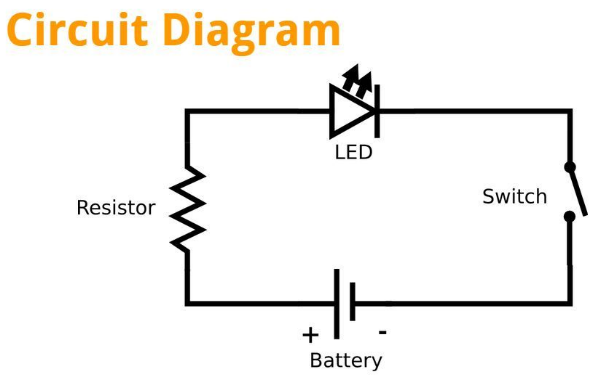
- **Emphasize that voltage and current are NOT mutually exclusive**
- **Come hand in hand - obvious decrease in voltage(power) to LED also means decrease in current**

○ **Add switch to circuit,show on and off**
  - **Conceptually - explain to them, when button is pressed, connects the two sides of the button (each side is on part of the circuit, so connecting this connects the circuit overall)**

○ DO NOT TAKE IT APART UNTIL END OF THE DAY

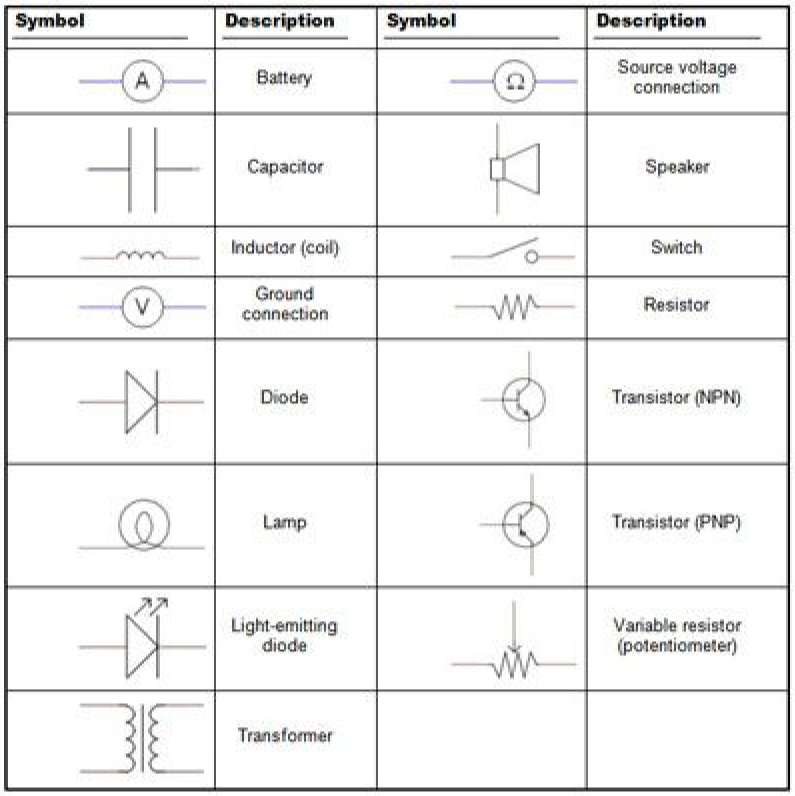
3. Schematic Diagrams
   - **Emphasize this is the form of documentation for circuitry! neccesary**
   - Used to model circuits, with universal symbols to make it easier for everyone to understand
     - Help you get a visual of your circuit first before building, a form of documentation for circuits, as they show circuit composition and properties
   - In general, draw schematic first before make circuit, this time is an exception as we are learning





- Given schematic symbols for circuit components given to you in helpsheet, try to draw schematic of circuit you made!
  - Do on back of helpsheet

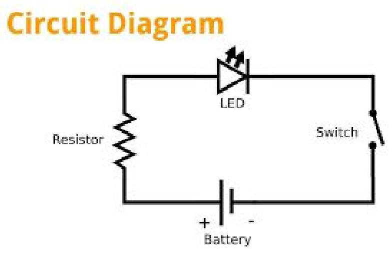
- Common Schematic Symbols -- will be using in drawing more schematics/making more circuits later!
  - Some of these, transformer, capacitor, will not need to worry about in this camp, but all relevant symbols point out



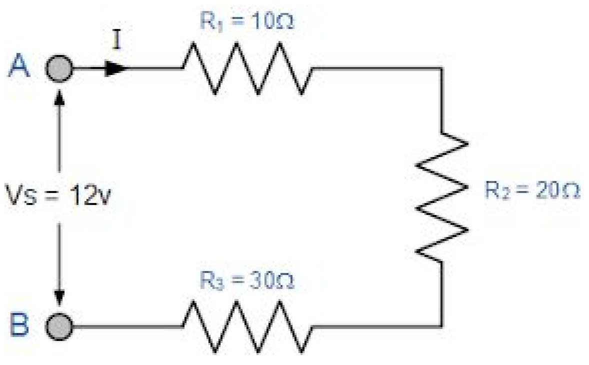
Symbol	Description	Symbol	Description
A	Battery	Ω	Source voltage connection
	Capacitor		Speaker
	Inductor (coil)		Switch
V	Ground connection		Resistor
	Diode		Transistor (NPN)
	Lamp		Transistor (PNP)
	Light-emitting diode		Variable resistor (potentiometer)
	Transformer		

Series Circuits:

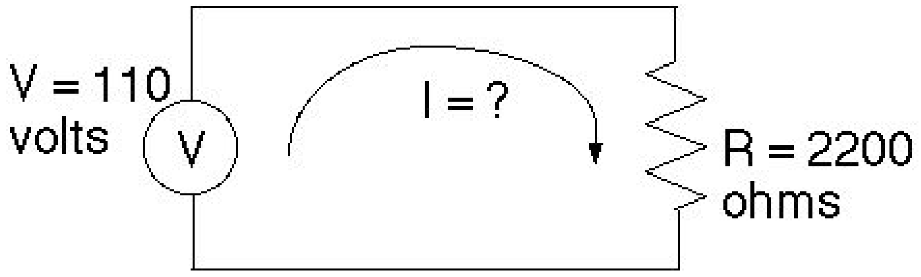
- Use the example they have infront of them - @ series circuit :)
  - Series circuits
  - all components of circuit are connected directly to the battery
  - If one connection is taken out, circuit is open, doesn't run
    - Demonstrate with buzzer/resistor - use LED to visually see current still running through the circuit
  - Our previous circuit/schematic was a series circuit!
  - Do u think ur house appliances are in a series circuit? Running the toaster, fridge/ oven, all at the same time or no?
  - Lets add a few more resistor, show how led's all dim
    - **emphasize components are interconnected, changing part of the circuit changes the state of other components**



- Series Circuit Characteristics:
    - LED's/any electronic component adds resistance to circuit as circuit is now longer
        - Therefore, in series increasing resistance decreases current, vice versa, why (hint, talked about this earlier today)
    - "magnets are farther apart analogy"
        - Farther distance, less attraction between cathode/anode, therefore slower electron flow
- The relationship between voltage, current and resistance in a series circuit can be demonstrated through **ohm's law, V = IR - fundamental law**
    - **Have them refer to their help sheets for this**
    - Young ppl, explain math at a very elementary level
    - pretend V is constant, ie, it never changes
    - If I decreases, what does R have to do to keep V the same?
    - If R decreases, what does I have to do to keep V the same?
- **Total resistance in a series circuit is the addition of all resistances in the circuit**
    - **On help sheet**
    - **Quick example problem**
        - **With picture to the right, calculate total resistance**



- Calculate current given the following diagram



    - resistance to 1100 ohms and recalculate
        - What does the resulting current from both calculations tell you about the relationship between current and voltage (discussed before, but shown mathematically now)
        - Show this with multimeter, compare accuracy of calculation and show relationship between current and resistance with multimeter
- Lunch Break - 12-12:30 -- **Play wii/xbox as eat**

    - 12:30-1, or earlier if finish earlier,
        - Circuit game!
        - Draw a schematic/given a breadboard/9 volt battery, build a circuit that can turn on/off 3 LEDS without affecting the other ! (transition into parallel circuits, next day)
            - Give hints and clues cuz deriving parallel circuits r lw hard
            - Build alongside them to help ease difficulty
        - If have time, start parallel circuits
            - **Therefore, keep all help-sheets handy and all instructors should be ready to start teaching this**

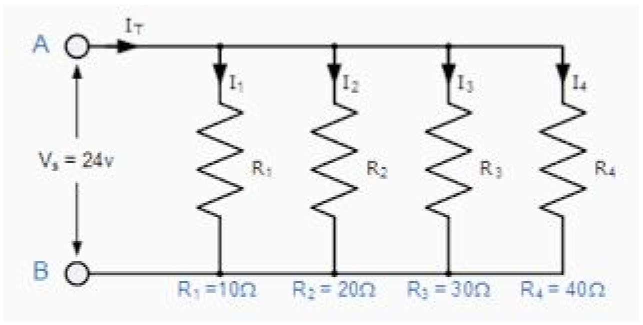
END OF DAY 1

DAY 2

9AM - 10:30AM

    - **Provide helpsheet for kids!**
        - **This content is only for 10-13 year olds, would go way over younger kids heads**
    - **VISUAL OF BASIC PARALLEL CIRCUIT THROUGH WHOLE EXPLANATION - each complement @ 1 LED and switch per channel, leave space to later add resistors/other components to discuss parallel component properties**
        - **Have them build their own visual too! Continuation of yesterdays activity**
    - **use multimeters on your, and their circuits so they can see what's happening! - engaging visual, shows whats actually going on in the circuit**
    - Transition from yesterday's activity, ie. if didnt finish that yet, then finish it, and start this! - say smth like oh this is called a parallel circuit! Lets talk about its characteristics/why it allows u to separately control all three LED's
- Parrallell circuits
    - Weird/Unique, has a lot of applications in daily life

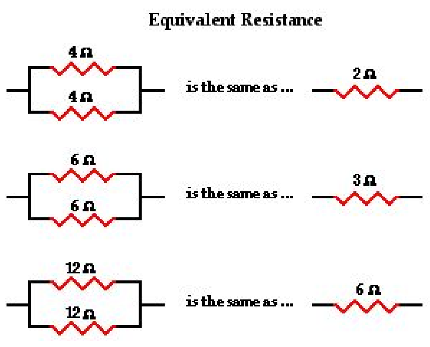
- Essentially, multiple components to a circuit, each component gets electrical current once at a time, so not all necessarily get at once
  - **Kind of obvious they all don't have to get electricity at same time.**



- **Each LED/part of component not connected to eachother, one "column" in a parallel circuit is not connected to other columns**
  - Give visual example with a 3 component parallel circuit, put switches in each circuit to show that if turn off 1 component LED, doesn't effect the rest
    - **Activity ended with yesterday, but show again for re-enforcement**
  - **Emphasize negative to positive flow for electricity, map it with finger on physical circuit**
  - Switch causes circuit between negative to positive to break in component, why LED turns off
    - In a series circuit, the switch would have turned off the whole circuit as the anode and cathode had no connection anymore. However, with a parallel circuit, the anode and cathode can still be connected through other pathways, which is why the circuit still has electricity flowing!
    - **On ur visual, only press the button on one of the components, parallel or series?**
      - **Series! Technically only one way electricity can go from anode to cathode! Other components are physically there, but bc not connected, they do NOT provide ways for electricity to flow from anode to cathode, and therefore do not count as components**
- Each component can be treated like as series circuit
  - **With one component, - if electrons already in a component, only one way it can get to the cathode - since only one pathway, why has properties of series**
- Make schematic of built circuit, utilize component symbols
  - **Emphasize necessary as form of documentation again!**
  - 3 component circuit with LED and switch for every component
- Through experimentation, ppl figured out that as a whole parallel circuit, an increase the resistors, increase the current also!

- **Dont worry about why, just trust the law as it was experimentally found**
- **Show them this by having them add resistors into each component, measuring new current in circuit**
  - **Using multimeter!**
    - More resistors === less resistance in a parallel
    - Note, it is extremely important to understand resistance decreases for the overall parallel circuit, but not each component!
      - **Each component is a series circuit, so if analyzing each component individually, an increase in resistance in one component will also decrease resistance in that component!**
  - experimental conclusion
    - Formula for resistance of overall circuit

$$\frac{1}{R_T} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \ldots$$



  - **Note, every component (even LED's) - add resistance, but they are so small compared to the resistance the resistors add that they are not factored into the calculation**
  - Try out formula to see that more resistors == less overall resistance
    - Simple calculation, remember level of students teaching
      - Simplest way- if they are not familiar with the math yet, just tell them how to do it/what to do,
        - **In calc, put in resistance 1 to ....resistance n on denominator of fractions and add using calculator**
        - **Then take the inverse of both sides(flip the top and bottom) - and then you have total resistance on the right**

        - Do calculations to the right picture,with doing calculations, emphasize the conceptual side of it --- **more resistors decreases overall resistance/usually the average of the two resistances**
    - **more resistors is less resistance, ohms law still applies!**
        - **As long as treat R as resistance, not amount of resistors, should be fine**
        - More resistors, Less resistance, more current with V = IR
            - **Measure with multimeter and prove**
        - Note, before current goes into the components and when it comes out, the current is the same, t**herefore the current distributed into all the branches, if added up, equals the total current of the parallel circuit**
            - **Current, is not evenly distributed between all components, but rather is distributed based on luck (electrons move very randomly), and which components have the most "space" to hold electrons (longer components - ie, more resistors, have more space)**
    - **More components == more resistance and more current also, bc each component adds more distance the current needs to travel from beginning to end, and bc we established this is what resistors do, this means more components adds more resistors, less overall resistance, and therefore more current with ohms law!**
        - More pathways, less crowdedness in each path, therefore more electrons can fill each pathway/increase in current overall as a result (more electrons can fill the components over the same unit time, which by definition increases current (current is movement of electrons over time))
        - **Have kids make new schematic with 4 components, add their circuit another component, and measure new current/resistance! With multimeters!**
            - **Can teach them now or earlier, whenever appropriate, and let them measure themselves!**
- **Do you think this is the circuit that is used at home with all different appliances? What do you think the "switches" you use at home are?**

- ACTIVITY - now that "understand" parallel circuits, can you create a 3 component parallel circuit with only LEDS, and try to make the LED's brighter on one of the components?
    - **Secret, add resistors on the circuit to the two components that you want to be dimmer, as each component is still a series circuit!**

10:30-10:45 -- BREAK

10:45-12, **BUILD SMARTCAR**

- **Kids will 100% not know a majority of the parts when they build the chassis, but you should explain them as you go**
    - **Ex. this is a motor, which turns the wheels of the smartcar - show wheels**
        - **We will be creating programs to turn the motors, and move the smartcar different directions later**
- **Build step by step (ie. make sure all kids are on the same step before you move on)**
    - **After build, no need to do wiring as only will become relevant later (when start programming section)**
- **Purpose is to fit in CADDing earlier so we have enough time to print all their part**

12-12:30 - lunch break

Note - can use screenshare during the cadding section to show them how do create project, folder, or use tools, but also can try to explain/walk around to help

START USING SCREENSHARE FOR THIS!

After lunch, start cadding, get them used to the software

- **how to projects, folders, new files**
    - **Organization and documentation with this - important to be organized so when come back after long time, can understand what did and keep working as if nothing happened**

Start on Day 3 curriculum if time

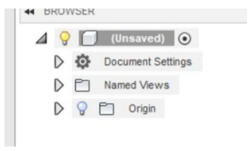
Day 3:

Note to instructors - you will need to be super comfortable with fusion software/be well knowledged where all the 3D modeling tools are as these are not detailed in the following curriculum

Provide helpsheet to kids - use to reference when explaining 3D design concepts

Note - can use screenshare during the cadding section to show them how do create project, folder, or use tools, but also can try to explain/walk around to help

CAD day - basics of how to use software - with an added project



- **As discussed yesterday, the project, folder, and file organization for projects, and the implications of organization, reiterate**
- **The folder with all the projects for the kids to do will be shared with you in fusion. Right click each part and "add to offline cache" to save it offline.**

Note - cadding will purely learning by doing - as make projects, learn about the tools, etc that can use to help make parts.

TIP - if a tool isn't working as you would like, press escape

- **Creating parts**
    - Sketches - **start on xy for consistency (not labeled in fusion, top left plane if you are in home view)**
        - Basic geometry
            - "L" for lines
            - "C" for circles
            - "P" for project geometry
        - Dimensioning ("D")
        - Construction lines ("sketch palette" on left side of screen)
        - Constraints (only use to reposition your lines, optimally you don't need to use these on 2d geometEditing sketches - double click to edit sketch, not make a new sketch (actually stress this a lot)
    - Controls
        - Shift + middle mouse for free orbit
        - Middle Mouse to pan
    - Profiles (define these to kids, will be used for definitions later, any sort of closed shape)
    - Extrusions
        - Needs at least 1 profile
    - Rotations
        - Needs at least 1 profile and exactly 1 axis (can select origin axis)
    - Sweeps
        - Need a profile and path
    - Lofts
        - Needs 2 or more profiles (regardless of planes, depended on how many profiles selected per loft)
    - Filets and chamfers
        - Needs an edge
    - Construction planes
        - Various types (only explain offset and angled for now)

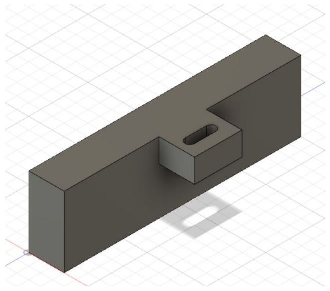
Day 4:

9AM - 10AM

As past days, screenshare is again an option! - show them and explain what this part will do, go through design process of example part we have for them

Designing part to be put on smart car chassis

- Using smartcar and CAD Knowledge from previous day, design a part that will aid your robot in pushing balls
    - **Introduction to end of camp competition**
        - **Leave off on discussion of the details (what is the exact competition, how many different challenges there will be, etc until later) - if go into this, it will take them WAY more than an hour to design this!**
        - **Give them general things to look out for based on understanding of equipment we will be using**
            - **Ie. whatever design ur making shouldn't be very close to the ground as it will hit polls that keep balls in a corner**
                - **You will be attaching the design to the polls (standoffs on which the ultrasonic rests on)**
- **Give students example design and constraints (size and thickness (so they can factor into their build - emphasize integrity is important too! It should not break if it gets in contact with other robots as during the competitions it will!))**
    - Give them the strength of the part with an analogy
- **Design: it just has to have a rectangle with a long hole (this is where the standoffs will go)**
- **size/contraints: dont go over 50k mm^3, cannot go more than 150 mm. in any direction.**
- Tell them to take 10/15 minutes to brainstorm, pros and cons, once have a design, tell them to you, once you approve it will work, have a small discussion with them about the implications of the pros and cons, and then have them build
    - Discussion of pros and cons, focus on consulting/analyzing idea like in a robotics team setting to help them
    - Dont want kids to have a bad idea and spend an hour building the bad design - waste of time!
- When cadding, walk around,view making of part, catch anything early on, help them with any issues, once cad is done, check to see if size wise will work out correctly (fit onto robot and will print in time)
    - Check browser to see if part is made correctly/will actually print
- EXAMPLE PIECE:



- EXPORT AS STL FILES, UPLOAD TO NAS
    - **Patrick will take all the files and start the printing process right after the files are uploaded**

10-10:15, break

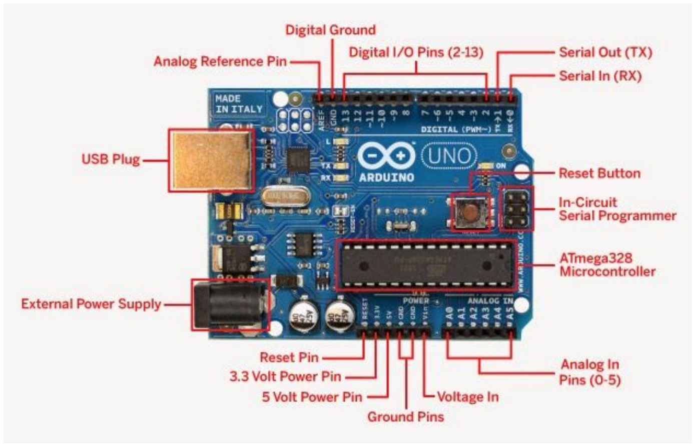
10:15 - 11:30 - INTRO TO ARDUINO

- **Take out arduino/use as a visual the whole time**
    - **U and the students both take out**
- **Provide help sheet with all info go over**
- What is an arduino? It is a microcontroller!
    - What would the arduino/microcontroller be if compared to your body?
        - **The brain**
    - Essentially the "brains" of the machine
    - How does a "microcontroller" control a machine/use your code?
        - Through transistors in the micro-controllers --- like cells in our brains!
        - Cells are fundamental building blocks of the brain, allow parts of the brain to communicate with one another, same with transistor
            - **Code, after you write it, is read by the computer, and the actions are performed by the computer with transistor communication**
                - **Ex. write code that tells your arduino to send a signal to a circuit, the transistors then send the signal!**
            - **Transistors allow microcontroller to communicate within itself, to other computers, or to send external signals that control circuits**
    - **Show the ports on the arduino, how transistors send these ports electricity when code prompts it**

- **Piece by piece introduction**

- **Provide examples of uses of each of these parts of arduino as describe them, examples already for some of the parts**
- **Note - for the below, obviously ask these questions when it comes appropriately in the discussion - goal is to refresh previous knowledge**
    - **With everything ask hella questions to remind them of previous content, ex, what is voltage again? Ie. what is the arduino really measuring when the digital pin is OFF when voltage is less than 0.8 ?**
    - **To stop the pins from receiving to much current and frying the arduino, in a series circuit what can you do? (answer, add more resistors) - why? Ohms law! - increase in ressistance decreases current**

- USB Plug -- upload code and power arduino -
    - Max 500 milliamps
    - **Redundant number, but shows again current used to communicate between arduino/computer - convert code into electric signals, vice versa**



- External power supply - when not uploading code - power supply
    - Max 3.3-5 amps(current)/7-20 volts
        - Safe is 12 volts
            - Any more, can FRY arduino - dont want that -
            - Any more "power" - what voltage is, when it reaches the transistors of the arduino, risks it breaks
                - **Transistors breaking, since they do all the tasks the code you make says to do, indicates a "fried" arduino as the arduino can't do any tasks anymore**

- Reset pin
  - **Reset button -- red button, restarts your code in arduino, led should flash a few times, same thing happens when uploading code to arduino**
- Volt pin is positive lead, ground is negative
  - Like battery in circuits we made previously, when using a breadboard, ground an positive pins must line up to positive and negative of the breakboard power rails
    - **Can power a circuit/project only using the voltage/current from the arduino - (anything 5 Volts or less do not because arduino can output, at max 5 volts, then use the arduino as the "battery" in the circuit)**
      - **Show the 5/3.3 power pins, let them know that when put with ground, this acts as the battery**
    - NOTE - **whenever create an arduino project using an external battery to power a circuit, etc, NEED to have the ground of the arduino connected to the negative lead of battery!**
      - **Attaching ground pin to circuit/thing arduino is controlling allows completion of circuit,** which as we learned before, is necessary
        - Adding arduino component in the closed circuit allows it to function/manipulate the circuit (what arduino is doing - reads the circuits (current/voltage), and based on what program installed in the arduino says it should be, manipulates it)
        - **This allows for the arduino to have a reference to in the circuit, so if there is a change, the arduino can detect the change and perform the corresponding action in the code for when the change occurs**
        - **Without reference, the arduino would never know when the change occurred,and because it never knows when a change occurs, it can never perform any action!**

        - (numbers without being compared to a reference number are meaningless)
        - Ex, 0,1,2,3...... , the only reason 10 has its value is because it is being compared to 0, so "10" is 10 **MORE** than 0! -- **same with ground on the arduino, whatever the beginning voltage/current in the circuit is, thats 0 now, the arduino uses this as a reference to output,ex, 5 more volts than the reference**

- Analog reference pin - sets reference voltage for arduino to refer to when seeing voltage of other analog pins
    - **Same as the ground, but there incase the ground pin is preoccupied**
    - In common practice, not used much
- **If want an analog pin to pick up voltage, only way arduino knows is by seeing that the analog pin voltage reading is higher/lower than reference - relative like talked about earlier**
    - **Important of a reference for input instead of output**
- Analog ports A0-A5
- Has ADC (Analog to Digital Converter) electronic circuit
    - Converts **voltage** to digital numbers
    - **Circuit is capable of converting voltage recieved in circuit at 1023 different digital levels -- why max number analog ports can read is 1023**
        - **Like a scale, stronger voltage has a higher number on the scale**
    - **Note - these pins have such "high" resolution, all the pins on the other side**
        - Point to the digital/PWM pins, are much lower, which we will read abt later
- Arduino understands digital levels/numbers 4 code
    - Use the numbers the arduino converts voltage to in code to refer to the voltage, manipulate voltage being sent out by arduino, will do later

        - Digital Input/Output Pins - pins 0-13
        - Difference this and analog - digital is only 1/0, while analog input pins is 0-1024
            - 1 is ON, 0 is OFF
                - For input and output, only send out two values, and for reading, 1/0 is a range of voltage that all corresponds to either one or zero
                    - See next page for more in depth explanation for kids
        - Pin 13 is connected to arduino LED - means if this pin is on, LED is on
        - Pins (3,5,6,9,10,11) have pwm capability
            - **Basically means these can be analog or digital pins with input or output, however resolution is limited**
                - **Same input reading (0-1023) output is only 0-255**
        - **Note, through code you can actually manipulate/change this resolution (number range arduino reads and sends voltage with analog),dont worry bout rn tho**
        - **Each pin, when input, can intake 40 mA, but recommended is 20 mA**
            - More than this, fry arduino! - make sure good before connect to arduino
                - Same thing as voltage, move through fast through transistors, the transistors will break
        - **Max current provide/take from all pins is 200 mA - output digital signals**
            - Good to know to make sure this current is significant enough to effect circuit/whatever you are manipulating
                - If not, can add electrical components like MOSFETS - amplify current, turn small amount of current into much larger - dw about details for rn
    - **Make sure follow ALL current/voltage max/min, don't want to risk burning/messing up arduino**
    - What does digital mean for these pins??
        - Represent voltage by 0 or 1 (off or on)

- Output
    - Gives voltage
    - **Set to 0 or 5 volts - (on is 5 volts, off is 0 volts)**
- Input
    - Voltage supplied by external device
    - Less than .08V is considered a 0 digital input
    - Greater than 2V is considered a 1 digital input
    - **Inbetween is undefined/neither 0/1 -- so make sure not here**
        - **Arduino just doesn't recognize/sets to nothing**
        - **Again, something to check and change in circuit!**
    - In code (we write later) - ON represents digital pin at 5V state (whether outputting 5V or receiving 5V, still at 5V state)
        - OFF represents a 0V state
    - Serial Communication - way computers/ microcontrollers talk to eachother, the **digital pin 0 -- (RX), and digital pin 1 --(TX)** are essentially serial ports, can be used for communication with software serial library (coding communication with other computers)
        - Can connect multiple devices to talk to arduino at the same time (in addition to the USB), not rlly useful for us, but good to know

BREAK 11:30-11:45

DOWNLOADING ARDUINO IDE from NASS/EXPLAINING HOW TO USE IT,

Note - now we will start to use screenshare to explain the IDE, so make sure it is ready by the beginning of the day!/set up during break

Use screensharing to explain everything ur trying to teach

Have all programs/screenshare visuals you will be using in the below lecture ready ahead of time

- Arduino IDE - This is a compiler, what is a compiler?

    - **Computers read in binary,** essentially a series of 1's and 0's, when put in specific orders ,represent specific commands and instructions computers understand
    - Compilers, convert the instructions/program you wrote into binary! For computer to read
- Introduce to Arduino IDE!
- Have them go NASS and download
    - **For people who are not using their own computer, have them send stuff to their email once done**
- Explain how IDE works
        - **With screenshare, write some varaibles in the void setup, tell them they do NOT need to know what the varaibles mean right now, just that they are setup to be used by the void loop below**
        - Top "void" thing is to **define** all stuff using in program - void setup
        - What does that mean?
        - In a program, **we give names to numbers, and sometimes have a block of code that is repeated multiple times, and is therefore redundant to write out repeatedly, put this in the setup section so we only have to call with a keyword,** instead of rewrite every time in the main part of the code that the arduino executes
            - All this is written in the setup area, and used in the following void loop
        - Void loop
            - Where use premade sets of declarations to write instructions for arduino to follow, what the arduino executes
- Small summary of how programming works and its place in the engineering space
    - In the engineering/robotics world, programming is a tool used to manipulate hardware to accomplish certain tasks/solve problems
    - However, in the software industry, programming is used to create virtual solutions to issues, such as the internet!
        - Gave millions of people access to an unprecendented amount of information
- **Fully understand arduino IDE syntax/coding environment**
    - Main workspace w/ main code

- **Show all with screenshare**
  - Black Console output for code uploading and error msgs (at bottom of IDE)
  - Tab w/ code name - each tab is a different program file within the same folder
    - **Note- if multiple files in folder, can call the code from another file into the file currently working in**
      - **More applicable when have students write functions**
      - Way to organize code by writing in different file
  - Verify & Upload Btns, New,Open,Save buttons
  - Top Dropdowns:
    - Classic File tab
      - Open new files/old files
        - **When click this/opens a new arduino window means the programs written in this window are not transferable to the previous window open**
          - **Ie. cannot call functions made in other files to a previous window**
      - examples ! - lots of example programs we will be using later to help demonstrate certain programming concepts
    - Edit w/ commenting out and undo/redo
      - If want the compiler to ignore a line of code/not convert it into binary, just need to add two slashes (per line ignoring) - show where the slash button is
        - Under the edit tab, there is a shortcut command that does the double slash for u, even tho it isn't hard to do
        - **Show commenting with screenshare again**
          - **Two types of commenting**
            - **Long and short comments**

- One IMPORTANT thing to note, in this compiler, **"1 line" isn't what it looks like to us**
    - One line is whenever add a semicolon! Semicolons signify the end of a line for this compiler!
    - For example, if write these lines of code on one line as it looks to us (define a bunch of variables and write them in "1 line" with semicolons
        - **Do with screenshare**
        - Run it, it works! Unconventional to write as it is confusing to humans as humans definition of one line is not the same as the compilers, but nonetheless it can still be written this way!
        - The double slash ignores one human line of code, so multiple of the commented out lines
- Select all, go to line, and paste keywords
    - Go to line keyword really useful when long code, and want to **navigate to line of error fast as the black rectangle at bottom (console) indicates which line is wrong when shows errors**
    - Select all keyword useful when want to select whole program - for deleting
        - can just select all with mouse
    - Paste keyword is good to paste code, but really easy keyboard solution for this too
        - Show kids, **control/command v** - young ones so might not know yet
    - Find keyword really useful if have a large program, and need to change specific variable names (variables are things that are equal to somehing) - for example, say x =5, can use x in place of 5 for the whole programcvc c

            - Screenshare, do x=5 or a number to make it easier, then use in void loop and change x var with this tool
                - **Change variable name, just use the find function and change it**
        - Increase and decrease indent, again can just do this with the tab button or delete button, and compiler usually does it for you, but this **essentially is when we are using a certain type of programming structure/concept(control/logical loops, if, for, else)**
            - **For most languages, this spacing is essential for the complier to recognize which code to run, and which code not to run**
                - **Tell them not worry about this too much now - ie. what are control/logical loops**
                    - **Go back to it what this is when cover this in next day**
            - **For this language, spcing si not important, but for other languages it is!**

    - Sketch tab....
        - **w/ verify/compile & include library (talk about libraries l8r)**
        - General idea for libraries
        - Remember when in the void setup section I mentioned how sometimes you can write a piece of code you will repetitively use?
            - Libraries allow you to import a ton of these to use in your program!

- **Someone else wrote it, ur just using what they wrote to make programming easier for you**

- Verify - essentially check for errors in code, lets u know if there are any before compiling
- Compiling allows the compiler to convert program into **binary instructions for computer** - see "export compiled binary" - which means send binary instructions to the arduino
    - **Under the sketch tab also**
- Add file, if downloaded an arduino program from online or something, adding the file would allow you to open it in the arduino ide - another way of opening a file as seen under the file tab
    - **Not something you need to focus on**

- Tools
- w/ library manager, serial monitor (Show serial monitor shortcut btn), and platform options
    - Manage libraries, again since we dont need to worry too much about libraries, we will talk about later
    - Fix encoding and reload
        - **If compiling isn't working for some reason, the error is not with program, then click this**
    - Board and port - to let the arduino IDE compile the binary instructions correctly to the arduino, need to give the IDE what specific board and port using
        - For this camp, arduino/genuino uno - dont need to put port if computer has one port, if more than one then you need to specify
        - Serial Monitor and Serial Plotter

Note- have the programs going to screenshare below, ready ahead of time!

- **Hello world program - use this with serial monitor so smoother introduction when go**

understanding hello world program tmr - have arduino connected/use at the same time

- **Instead of print hello world, print 5, for serial plotter**

Note - u should be using screenshare this whole time!

- **Extremely important and useful when troubleshooting code**
  - When turn the serial monitor on (in the void setup)
    - Serial.begin(9600);
      - Tells serial monitor to begin reading at this many bits per second, 9600 is convention, just use
      - **Dont go into too much detail about how this works for now, will go into more detail when start learning programming**
  - Use the serial monitor (void loop)
    - Serial.print(value)
    - Serial.println(value)
      - Difference is that ln puts each data point on a new line
      - Putting on a new line makes really easy to read
  - Serial plotter
    - Will plot the data points gotten against time,
  - Purpose of serial monitor/plotter
    - allow you to see if you are getting the correct data you intended to

get to allow program to work, adjust program accordingly!

- **Everything else ignore for understanding the arduino IDE ignore for now, not really relevant**

LUNCH BREAK - 12:00-12:30, lunch break!

Spend rest of the day finishing the intro to arduino IDE - if don't finish, mark where you left off, and do it tmr/be concise to catch with this schedule as after this timing gets really tight!

Day 5

Continue Arduino IDE introduction if don't finish from yesterday - be concise tho bc from here on the curriculum gets tight

Screenshare the whole time - with hello world explanation, then when appropriate, transition into them going into IDE/writing their own code

Can explain concepts using screenshare, then when have them do a follow up example after, they write in IDE/comment in their code so they remember at the same time!

For programming..... (please go at a slow enough pace so everyone

understands…. Add more examples as appropriate until everyone has a clear understanding) at the same time….from this day on its get pretty tight…so go as fast as u can while making sure everyone understands

Note - below is additional things that were not explicitly covered in the above curriculum, but whenever appropriate, with examples, SHOULD be covered!

start

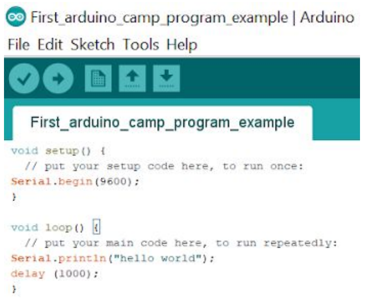
- Arduino library:
    - Basic Math:
        - +, -, *, / same as normal math
        - % for remainder in division (ex: 5 % 2 = 1)
        - Comparison with ==, >, <, >=, <=, !=
        - Compound Operators with ++, +=, --, -=
        - Math shortcuts
            - Addition
                - Shortcut: +=
                - value += 5
                - Adds 5 to the current value of the variable
            - Subtraction
                - Shortcut: -=

            - value -= 8
            - Subtracts 8 from the current value of the variable
        - Multiplication
            - Shortcut: *=
            - value *= 3
            - Multiples the current value of the vairba;e by 3
        - Division
            - Shortcut: /=
            - value /= 5
            - Divides the current value of the variab;e by 5
    -
- Logic:
    - ==, !, &&, ||
        - If necessary, explain logic gates And, Or, Nand, Nor, XOR
- Math Functions:
    - abs() for absolute value
    - sq() for squaring value & sqrt() for sq root
- Controlling Arduino I/O:
    - pinMode() to set up pinout for connections. Parameters: pin #, mode
        - Pin #: Pin being used on Arduino from Digital or Analog
        - Pin # should be set to an int variable. Ex: int LEDPin = 5;
        - Mode: INPUT or OUTPUT (NEEDS ALL CAPS). Ex: Input from button sensor or Output for controlling servo/motor or LED
        - Usually found in Void Setup(){}
    - digitalRead() to read value of input pin
        - Outputs HIGH or LOW, parameters pin #
            - In example set equal to a int var, so prolly 1/0
        - Ex: when button pressed on pin 5: digitalRead(5); will have value of HIGH
    - digitalWrite() to set value of output pin:
        - Can set to HIGH or LOW, parameters pin # and value
        - Ex: To set pin 3 to HIGH: digitalWrite(3, HIGH);
        - Using both digitalwrite and read can be useful to control items with if else / switch cases
    - analogRead() to find value of input pin:
        - Can send many different values
            - Increments of 0-1024 volts
                - Note, this depends on what the operational voltage is
                    - If 3.3 volts, increments so this is one more than max, if 5 volts, same incrementation
        - analogWrite(pin#, val between 0-255 - must be int)

- Number represents PWM duty cycle
    - Talk about this tmr with intro to PWM
    - 255 is 100% duty cycle, 0 is 0% duty cycle
        - Whats duty cycle? See curric for tmr

End of additional things to cover

- **Help everyone get their arduino's out, connect to computer, show kids what they need to write, say we will explain it later**
    - **When writing the "Hello World" prgm, have them write:**
        - Classic example used (can see in pic to the right)
            - In void setup()
                - Serial.begin(9600);
            - In void loop ()
                - Serial.println("hello world");
                - delay(1000);
                    - From a preimported libraries


First_arduino_camp_program_example | Arduino
File Edit Sketch Tools Help
First_arduino_camp_program_example

```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
Serial.println("hello world");
delay (1000);
}
```

        - **Upload Procedure to Arduino-**
            - Plug in usb cable to Arduino & PC
            - Press Verify & Upload
        - Make sure computer is connected to arduino for uploading purposes, then compline and upload, go to serial monitor and see if program worked!
            - **Make it clear, if a normal programming IDE, no need to connect arduino and then upload,** doing this because IDE specifically made to give code to the arduino, arduino runs it and IDE uses serial monitor/plotter to show results (if serial functions in code)
                - With this emphasize how this IDE works - create code to put into arduino, arduino runs it (not computer), and if computer connected, can display any values code is programmed to show
    - Explanation:

- What are the {} on the viod setup/loops?
  - **More detailed explanation later - when go over functions and stuff**
  - **For now, basically tells the compiler that code is in here**
    - Whenever u need to write a block of code, for set up (void setup) or execution (void loop), you need to include the {put code in between these}
    - Void setup runs once, and void loop runs continuously
- Explain later:
  - Essentially, necessary syntax for any compiler to recognize using two specific types of programming structures - loop control structure or creation of methods
    - Explain later, but the void loop and void setup stuff are essentially methods, mediums in which code is written, depending on the purpose of the method, either to be used in another method, in a command terminal, or for code to be executed in the code itself (info in void setup method used in void loop method, void loop method is executed!)

- Syntax: Spacing, Periods, Parentheses, Quotation Marks, semicolon
  - Again, the compiler recognizes each **"line" by semicolon**, which is why we must put this at the end of EVERY line **(with exception to any control structures which use {}, the {} are telling the compiler - read everything inside the{}- so everything inside the {} must have (;) to be understood by the compiler, but not the control structures as they create a medium for lines of code with semicolons to be understood**
    - **Address why void loop and void setup don't have the ; behind it, anything that has the {} is considered an exception because the void loop and void setup are essentially "holding" all the lines of code, are not treated as one**

From this point on, stopped using any highlighting conventions.....its ok cuz all of it is important :)

- The Serial class keyword:
  - Any name/symbol given to an object to represent it!

    - In this case, the serial keyword represents the serial monitor/plotter, it contains a bunch of other keywords that do tasks for you
        - These keywords essentially hold code
- For right now the variable "serial" as the keyword refers to anything relating to the serial monitor/plotter (which helps you display values)
    - Dont try to go into classes and all that stuff with this little ones
- To access a specific command for the serial keyword, which represents anything serial, all you have to do is add a dot, and then a keyword for the command/function you want to execute (guess what, these "commands" are all methods - blocks of code written to be executed with the command word given to the method!)
    - .begin() - always need the () after the command keyword, as when these commands were made (you are using a block of code made by someone else, and this block of code is called by using a key word), the () was put just incase the block of code required an input value that the block of code manipulates to complete its task
    - The block of code does need input, a number ! represents how many bits per second the serial monitor should take in data from the arduino!
        - 9600 is the conventional number, just use this
        - This number is then manipulated by the block of code represented by the keyword to tell the computer, when u compile the code, that
            - The serial monitor is on
            - It should read arduino data at 9600 bits per second
    - This is all setup, turning the serial monitor on --- a setup step for the void loop (one continuously executed by the arduino --- to use

        - One note, the void loop continually executes, as in once the block of code in this loop reaches the bottom, it restarts again!
            - Think about it, would turning on the serial monitor every time the void loop rerun be practical, each time returned on, lost all past data, or would u like to have it on once, leave it like that forever? --- as a tool the void loop program can use?
    - In the void loop program....
        - Serial.println("hello world");
            - The "serial" variable with another method println
                - Dot to signify a method/command is being called (remember the key after dot represents a block of code another person wrote to complete a task)
            - () is present to take something for the code to use (code represented by keyword)
                - In this case, we want it to take the statement hello world, so it can be printed on the serial monitor when the arduino is running the program
                    - Why in quotations?
                    - We will talk about this soon! For now, any word/sentence/letter must be surrounded by quotations to be understood by the compiler
                    - Will print without the " ", only there for compiler to understand/recognize

        - note , this specific method can take in any type of input, word, number (don't mention primitive types yet) as this is how the code was designed - to take in anything essentially and display it on the serial monitor!
    - Semicolon present to signify the line of code is over,compiler can move on
    - @ln in name so we can create a new line for each time print hello world
    - Since this is in the loop the arduino runs when code is downloaded, if display serial monitor, it will be continually printed on the monitor
- delay(1000);
    - Semicolon so compiler knows line is done
    - Another keyword that runs of block of code
        - If this is a "keyword"/method, so often before we saw these behind a word and a dot, why not this?
        - Some methods can be referred by just calling them under certain circumstances, we can worry about it later
        - This block essentially stops the program for a given number of time put in the () - which again is to put any value that program needs to run
        - These "values" put in the () are called **parameters**
        - Numbers are inputted to the () by milliseconds

        - Inputting 1 is 1 millisecond delay in program
        - Inputting 1000 is 1 second delay in program as milli means 1 thousanth of a second, so 1000 1 thousanths is 1 second
    - Why do we include this delay statement?
        - If not, the "hello world" will be printed every time the compiler finishes going through the whole loop, will print really fast
        - Add delay statement to slow down the printing
        - Real life application, if programming arduino to turn LED on and off, need to add delay statements so the on and off is visible!
        - Commenting! - a **necessary form of documentation in programming**
- Note, commenting means that compiler will ignore, so u can write normally
- In real world
- @top, description of what program does, what parameters it takes in, what it outputs if anything, otherwise null
    - Every block of code that does a specific task to accomplish main task, on the side (conventionally right hand side after the line of code is done) write a description of what this part of the program does and what it outputs for other parts of program/inputs from other parts
- Have them comment their first code
- For this course.... every time we learn a new concept in a program u write/ example given to u, want u to comment on the side of the line

of code with the concept, what the concept is so u can look back and remember

- Two types of commenting
  - 1- //, one line of code ignored
    - Short comments
  - 2- */ and /*, from wherever the */ starts to when /* is, all stuff ignored
    - Useful for longer comments
- Commenting can also be useful when testing parts of code (see which parts work and which don't if make a program that has errors, or if need to modify a part of program to change overall task),
  - usually use 2 as commenting out multiple lines of code, long, but 1 also used if not much

○ Introduction to general engineering design process - specific to programming

- Whenever creating any product, including a program, should be an iterative process, in which create part of the product, test to see if working as intended, make design changes as needed, then work on next part that builds upon previous part, keep repeating process till done
- If build at all once, doesn't work, don't know where mistake is, will need to reverse engineer whole product and find all the mistakes, much less inefficient and harder to find errors as usually multiple
  - When edit to fix one part, often other stuff changes too that used to be correct

○ DoSaving & Naming procedure - now that understand program and have run it

- Give file for program a name that:
- Type of file this is for (usually put in a folder, but just in case)
  - For this - Arduino
- Indicates the project working on
  - Ex, SmartBot
  - Rn, Instruction
- Indicates the part of the program the file constrains, or version of the program

            - Ex. Version 1 or DriveCode_Version1
        - In between every part of name, put 1 _
    - Saving all of one file type under a folder, for organization-rlly important once u are using computer for multiple tasks
        - Under "documents" file preinstalled in computer, add an Arduino folder
            - Inside Arduino folder, and Arduino Summer Camp 2019 folder, inside this create Introduction to programming folder, and in here save the files making right now
            - Lots of steps, lots of folders, but organization is key

10-11

Again... use screenshare for them to see what ur doing, then write in IDE/make their own files/examples whenever appropriate

Note - for every concept have them do the activity specified below

- **Continuing to teach programming**
- Primitive types
    - **SCREENSHARE - ACTIVITY - TYPE OUT A BUNCH OF PRIMITIVE TYPES ON UR IDE**
        - **Explain what primitive types are while doing this**
        - **Incorporate knowledge they learnt through the initial example given to them to spice things up**
            - **Ex. given this method (delay()) , if put delay("1"), will this work?**
            - Stuff compiler can understand --- foundation of any program, ie. any program must use these data types
        - Int: Number - ex. 5 (anything without a decimal)
            - **Syntax: *int variable = 93;***
        - String: Text - note, must be in quotes as discussed earlier
            - **Syntax: String variable = "word";**
            - Any letter, word, or phrase
            - What if quotes in the string
            - What if backslash in string
        - Char: Character - single quotes to represent single items - 'I'

            - **Syntax: char variable = “s”;**
        - Float: Decimal ex. 4.00, 5.55
            - **Syntax: *float variable = 34.82;***
        - Boolean - true or false - key words true or false
            - These are used for some stuff we will talk about later (control loops)
            - **Syntax: bool variable = true;**
    - Using arduino/general controlling robots, only use int/float really, no need to add strings
    - Variables
    - Can put any of these primitive types equal to a variable instead of typing the primitive type over and over
    - After you create a variable, u can modify it through the program
        - Give example of u doing this
        - **Can u, for example do this?**
        - **Via screenshare show this example, ...edit difficulty for age ranges of students**
            - Int a = 5;
            - A(lowercase) = a + “mom”
                - why/why not?
            - Can u do this?
                - Int a = 5;
                - Int l = 5.5;
                - a = a + l;
                    - **Yes, because different number primitive types can still be added/subtracted - used to modify eachother**
                - U can cast tho, casting means turning one primitive type into another type
                - Cast by putting (new type)V
                    - Ex Int a = 5;
                    - Dou b = (float)a;
                    - Can cast from int to float, float to int

- What can variables be? - **Camelcase - give screenshare example w/ int myFirstVariable =6; - or anything you want really w/screenshare**
  - Numbers, letters
  - By convention, only use letters, first word is lowercase, every resulting word, to indicate a break in words, is uppercase
  - Ex. int myFirstVariable =6;
  - Note - all variables - name must be unique to indicate they represent a value
    - Easier to remember/understand when looking back through code
  - Spaces not allowed in var names
  - **Cannot be a "keyword name"** - produces an error

- name/letter/number cannot already be a premade variable in arduino, make sure its unique!
  - If premade, will get error, because compiler will not understand what you are trying to do
  - **Ex.int true = 4;**
    - True is a boolean value, can't make a boolean value into an int value
  - Note, the single = does not compare to see if two different values are equal, but rather sets the variable on the left hand side equal to the value on the right hand side, comparing is something we will get to later (relational operator - ==)
- Functions/methods, essentially made to do something in program, u will be able to make ur own later!
  - Puts a big block of code into one key word as said before, has () for parameter, whatever is put into that is used by program to complete its task
- Examples of functions
  - **Create a function - one that just makes x =5....super basic - but shows that keyword represents code**
    - **Then say this is not what functions would ever really be used for..... For doing things to inputted values and**

outputting values - do input value of 3, does input values times 2, outputs this

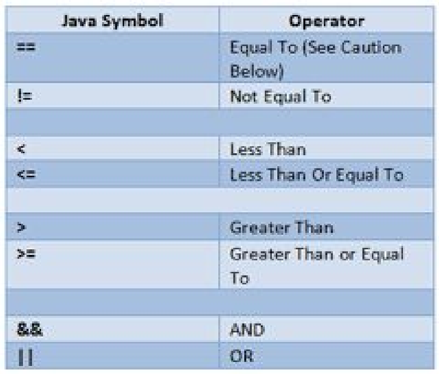
- **Function takes in something, outputs something!**

- Already lw went over earlier
- Serial.print() func: function "print" sends string("text") to serial monitor
- Void Setup(){} Func: Function that automatically runs at start of program to setup items needed later.
    - Ex: Naming/allocating digital & analog ports in setup
    - **Arduino has a map of which pins are what number, set variable equal to a number, use a pinmode() function later with the number to call the pin/set the pin to input/output, don't worry if don't understand this yet, will go over later**
    - **Note, initial setting variables equal to primitive types can be done outside both void loops on the top of the program, and should be done at the "top" of the program (outside both loops) to make it clear that they can be used within the whole program - scope - go over later or introduce as comes up**

- **ACTIVITY: create a program that turns the LED on the arduino on and off**
    - Note - if u don't know how to do this, look at the arduino IDE example, learn how to use the pinmode function!
    - Void setup
        - Find number of pin that is connected to LED, set up pin to being output
    - Void loop
        - Turn pin on and off with delay statements so it is visible
            - Another use of the delay statement

11:00-12:00

- Relational Operators:
    - **Examples with screenshare**
        - **Do all relational operators - including and/or!**
    - These are to compare two values/variables



Java Symbol	Operator
==	Equal To (See Caution Below)
!=	Not Equal To
<	Less Than
<=	Less Than Or Equal To
>	Greater Than
>=	Greater Than or Equal To
&&	AND
||	OR

- Most often used in creating conditions, which all of the control structures we are talking about soon will utilize
- Relational operators essentially create a boolean value, either true or false
    - **4 > 3 - true statement!**
        - **stop/go analogy - statement tells code whether to run the code, or skip over it, if the block of code controlled by this statment (4>3) did not get run, 4>3 was false, otherwise it was true!**
- **Screenshare the following!**
- **Questions**
    - **False == false, what is it?**
        - **True**
    - **False == true, is false, but is there any way we can make it true?**
- **We can use the not symbol (!) to make the boolean value the opposite of what the statement says**
    - **Ex. boolean l = !(false == true), l is true**
- Also have += and -=, really fast way to edit variables
    - Ex. a = a + l can be written as a += 1
    - a = a - l can be written as a -= 1

- Control Structures
    - if/else if/else
    - Note, all these statements use the {} blocks! When writing an if, if else,else,while,or for statement, no ; is needed after initially writing the statement, only ; is needed for all the lines of code within the if/else if /else condition
        - Syntax - if/else if /else (condition) {code }
        - These statements each have a condition attached to them, as long as the condition is met at the time when the compiler reads the statement, the code within the statement will execute
        - **Screenshare with if statements - if(true) - print smth**
            - **Upload to arduino, open serial monitor, and show!**

If/else if/else blocks

- **Ex with screenshare**
    - **Set variable = 4, account for all three scenarios, whether x greater than, less than, or = to 4 (print something to serial monitor with every outcome) - last condition, use else statement, because nothing else is possible, has to be it!**
        - **Make something w/4 conditions, have 1 if, 1 else if, 1 else, show that else will run on either of two conditions!**

        - If want to have a response to a number of predicted conditions, depending on what the values of variables that in the conditions are can use these blocks
        - One if statement, infinite if else statements, else statement at bottom (optional)
        - Once a condition is met and the code is run, then the whole block is skipped
        - Can have this block without all statement types, ie, can have if and else, must have if tho --- needed to initiate the block
- In programming, we have several loops
    - **With hella ex and explanation through screenshare**
        - **Every loop, give an example to help illustrate concept**
    - Loops are something that the compiler will run through as long as the condition for the loop is correct
        - Loops run continually until the condition for the loop is broken - ie. the compiler stays on the loop until it is broken, and the code within the loop is run top to bottom, and then repeated until the condition becomes false
    - Loops include While & For loops
    - **While loops - turn on three LED's**
        - **Make 3 diff series circuits with LEDS, using 3 digital pins right next to eachother, make a for loops that turns each of these pins ON, and therefore turns the LED on!**
        - While(condition){code}, For(condition){code}
        - **For loops are really useful to simplify the loop/shorten down the amount of code put in the body of the loop - in a while loop**
        - `for (initialization; condition; increment) {`
        - `  // statement(s);`
        - `}`
            - Note, for all these statements, no need to use ;
            - Initialization, creating a variable (usually representing a number)
                - Convention is to use the variable i for this part, if have a second for loop, use the variable j, etc
                - Ex. int i = 1

            - Condition, whenever u want the loop to stop/compiler to exit, and based on the initialization value, make accordingly
                - Ex. i <= 5
            - Increment, changes the initial value by x amount each time the bulk of the loop is run, keeps changing until the condition (second statement) becomes false
        - **Using a 4 loop, create a variable, and add onto that variable as many times as the for loop runs**
            - **Serial.println this value - run code, show them that the value will keep increasing beyond the end of one for loop….why (only if you define the initial varialbe outside of everything, and make it global)**
                - **Because the void loop() - keeps running, it is a infinite loop! - reinforce this concept**
        - Way a for loop runs, first initialization of the variable, then runs through the bulk of the code, then increments, then checks the condition, and if the condition is true it runs again (except no **re-initialization** when reruns)
            - Otherwise would be an infinite loop, as the condition to stop the loop would never be met
    - Void loop is your program, the arduino continually loops through this to do what you want it to do
        - Only difference is this loop never stops as no condition
    - Infinite loops, if condition is never met in the for or while loops, keeps going through code segment/never stops, bad for computer as amount of memory used up is a lot if while/for loop is outputing values (don't need to understand, just know it is bad)
        - In general, want to avoid, however in specific cases when infinite loops are neccesary (ex, in arduino - and if the microcontroller is built for this), its ok
- Global/local variables
    - **Hella ex as this is a lw confusing concept**
        - **Examples**
    - Global variables, usually initiated at the beginning of the program (before any of the loops), or at the beginning of any of the loops, exists/can be modified or used throughout the whole program

        - **Before both void loops to indicate it can be used by whole program**
            - **If in two {}, the variable is automatically restricted to only existing in that {}**
    - Local variables, ie, declared in loops themselves - if/for/while loops, are local to the loops themselves, only exist in them, not outside
        - **Hella screenshare ex!**
        - If global variable i, and create a local variable i (redeclare it and everything) , local variablel overrides global
            - **Ex - if declare int i = 10**
                - **Make a for loop where i becomes a max of 9**
                    - **For loop can do whatever it wants inside**
                - **And print i at the end.... i=9!**
                    - **For loop, even tho "reinstantiating" - int i = 1**
                        - **Since old i was of same type, treats as the same....overrides it**
        - Global variable exists outside, when loop is exited, global variable should be unchanged
        - If in a loop set global var equal to new val, then it is changed
        - If create **new** variable in a loop, anyting of {} - it is local to that...only exists in there!
- **Activities/problems want them to do for all individual concepts, one at a time and then integrate multiple concepts and increase difficulty, until the end of the day**
    - **All should have a physical effect on their arduino once they upload code**
- **For instructors to make, but some examples could be.......**
    - **Print all the letters in the alphabet**
    - **Print every number leading up to your current age**
    - **Harder ex..... See below**
    - **If want......and if you know how to do**
        - **Activity: Use serial.print() and analogRead() with potentiometer to see values outputted**
            - **Define analog pin for potentiometer**
            - **Connect 5V, Gnd, and analog i/o to breadboard and pot**
            - **Print all values**
    - **Turn an LED on ONLY if a variable is equal to 5**

        - In the final else condition, set the variable = 4
            - Else if is when varialbe = 4
                - In this one.... LED turns on slowly! - analogwrite!
                - Also sets variable = 5.... So after this.... First condition is now played
    - Making a function
        - Create a function that makes a nixie-tub print out the number 8
        - Another function that reads the value of an input pin (digital) - and returns that value
            - Put weak battery in digital slot so returns 1 (3 volts --- above the 2V threshold)
                - Negative in a ground slot
            - Practice setting functions to variables with same type that it returnec

After lunch, finish previous concepts with examples

DAY 6

Today is mostly to finish yesterday.... As you probably will not finish all the intended material

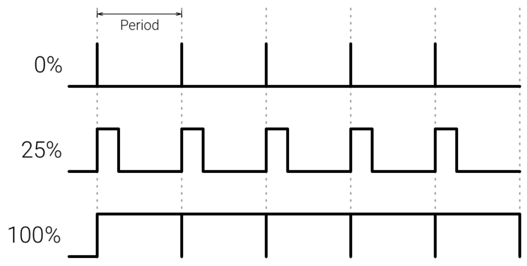
yesterday (please go at a slow enough pace so everyone understands.... Add more examples as appropriate until everyone has a clear understanding)

When you finish..... Below is mandatory curric

- **9-10**

  Discussion of PWM, Serial Communication, SPI, I2C, Aref, Interrupt, External Interrupt,

  - **Note - before go into topic, will need to teach them about periods (use sine/cosine graphs, and show how 1 of the repeating patterns is one period), review what percentages are (for calculating duty cycles)**
    - **Can use whiteboards for this**
    - **After explanation of PWM, show them an application**
    - **Show them a online video of PWM with an led, make after show vid**
  - PWM
    - Whats used to increase and decrease voltage/current given to a circuit
      - Increase in voltage increase in current bc ohms law



    - Within one period.......... (one period is the time it takes for one of the fluctuation cycles to occur)
      - **Arduino periods/cycle length is always 500 Hz (wavelength unit)`**
        - **Good technical knowledge**
      - The duty cycle (part that grows through the picture), is the high voltage section of the period, the blank section is the low part

of the period (alternates between whatever "high" voltage is set and "low")

- Note, for arduino uno, default high voltage is 5 Volts
- Percentage of the wave cycle compared to the low voltage/off is the duty cycle
- Percentage decides average "power" - or voltage, going to circuit
    - **Average is high voltage times its percentage**
        - **Do example problems with 25,50,75 percent 5 volt**
        - This average voltage, changes power circuit provides, used to control speed of motor (change amount of voltage/current provided to it)
    - Dim LEDS/make them bright
    - The PWM pins on the arduino are the only ones that can do this
        - Only analog pins can do this
    - Other applications outside of controlling voltage/current in circuits
        - Sends the high/low voltage, other devices read the high/low voltages as 1's/0's, interpret electrical signals into instructions to do actions/complete tasks

Note - for PWM w/motors, see next day as this will be schematic one for the smartcar - implement 2nd after LED

Also.... If you can get STARTED ON LINE TRACKING PROGRAM (SEE NEXT DAY) --- THAT WOULD REALLY HELP CAMP TIME! - INGORE ALL THE ACTIVITIES AT THE END OF THIS DAY

-
    - **Two programs create after get knowledge (make examples u just talked about p much)**
        - **LED program and motor program - obv dim/brighten for led and inc/dec speed for motor**

- Note, have to do all on breadboard on smartcar as this has already been made
  - Unplug DC motor and do it like that too
- With motor program, transition into how this is the bot we are going to make controls the speed of motors, even turns, etc!
  - ETA for how long it takes

BELOW... HONESTLY IGNORE.... NO TIME -- IF THERE IS A MIRACLE/LOTS OF TIME, THEN DO

Once you finish the above curric....You can go through additional arduino-related activities that transition intro more arduino-focused projects

Note - if you don't finish all these... no biggie, these are supplemental projects, just start the next day with next day's curriculum

- **Activity - when press button, led turns on, otherwise off**
  - **Button on sep circuit, when pressed, completes circuit/pin reads on (digital)**
  - **When this happens, turn on output voltage on another digital pin that turns on LED**
  - **Put a stop function!**

        - **If (on) -LED on, else (only other case is off) - LED is off**
    - **Using a while loop, accomplishing same challenge**
        - **Inefficient, but two while loops, one while LED on, one while LED not on**
    - Challenge: Combine while loop program and for loop to turn on LED with button only 4 times
        - Same while/not while for when button is on and button not on, but for button on while loop….. The for loop for turning on LED 4 times is on there - must be on and off in one run of for loop for it to be visually understanding that turned on 4 times
    - **Serial Functions Cont:**
        - Communication between computer and arduino using Serial Monitor
            - Serial.read(): takes value sent from PC's Serial Monitor
            - Usually given to variable and used in switches
            - Ex: input = Serial.read(); if (input == "Hello") {Do Something}
            - **Can make activity if get here**
    - **Importing Libraries:**
        - Library: additional code files that allow easier coding or implementation of other items like sensors or tools
            - Syntax: #include <name.h>; place at the top of code
            - **If get here, let know that we will be using this in servo program… can do a servo activity with them now if want**
            - **Make servo face forward, left, right in a never ending loop (hint… no need to make never ending loop cuz this is void loop!)**
            -

Nixie tube stuff if get here/u know how to do this:

- Nixie Tube Coding:
    - Nixie Tube: 7 segment digit and decimal point display
    - 8 inputs for individual segment programming
    - Similar coding to LEDs using breadboards and digital ports
    - Activity: Build a circuit and program to display any number 0-9
        - Connect digital ports to each segment and decimal of Nixie tube (use resistors)
        - Label each port for each segment
        - Use DigitalWrite(Pin, HIGH) for on segment, LOW for off
    - Ask about displaying different digits one after another

- Implementing Functions:
  - Instead of having blocks of code for each number, use functions and call them for each digit
  - Syntax: type name(parameters) {code}
    - Types include “void” and “int”
      - Void means returns no value. Aka when completed doesn’t give a value
      - Int means returns a value. Aka when completed, function gives value of 1. Use “return (number);”
    - Functions can use parameters
      - Parameters allow values to be inserted after the function is created.
      - Ex: void Say_Letter(char letter) {Serial.print(letter)}
      - When calling function, can choose what to do specifically: Say_Letter(“a”) will print “a” and Say_Letter(“w”) will print “w”
    - Activity: Use functions to show digits on Nixie Tube easier/faster
      - Ex: void digit_8(void) {for (int i = 0; i < 8; i++){digitalWrite(i, HIGH);}}
  - If time: 4-Nixie Tube
    - Show example code with function from yahboom
    - Explain #define:
      - Used to replace words/characters with other words/char
      - Ex: #define digitalRead dR. Causes dR to be read as digitalRead.
    - Explain char table:
      - Created a table with dimensions 10 by 8
      - Used to simplify calling each segment on or off:
        - {0,0,1,1,1,1,1,1} calls for 0 because first 2 are off (decimal and middle segment) and the rest are on (outside ring)
    - Explain Display function:
      - Use parameters com for choosing nixie tube segment

END OF NIXIE TUBES

DAY 7

All the following red is applicable to all the following days!

Make sure you understand the code and schematics for the following configurations. ENSURE YOU KNOW WHY IT WORKS and not only how it works.

Have the necessary schematics ready for the next day.

With each schematic... dont fully tell them how to do it... its their time to shine... walk them through the logic of what to include, ask them to try to do it themselves... obv since they have only been coding for 2 days this will be very hard... so help them out as needed/if they look stuck... but all the program making/testing should be on their own!

For all schematics - follow the logic of example programs!

Note... u did the below yesterday....

Robot Configurations

- **Motor Wiring Diagram (1st Schematic)**
    - Connecting motors to arduino and using code to start and stop the motors
    - Use pins provided on the circuit board and use appropriate code

- Make sure the robot is suspended in air to prevent the robot from driving off the table
- Give brief explanation of motors
    - They turn when power given to it, this turns the wheels, provides movement
        - Since motors/wheels can turn in diff directions @diff times, how do you think we turn right and left?

Finish of what you did yesterday

Line Tracking Diagram (2nd Schematic)

- Using line trackers on the UNDERSIDE of the robot
    - Remember there are two sets
        - One under the robot in the front
        - Second on the front of the robot
- Explain how a line tracker works
    - Emits light through the CLEAR LED and received light through the BLACK LED
    - Light reflects off the ground
        - Black and other dark colors absorb lots os light and cannot reflect much light back
        - White and lighter colors do not absorb as much light and much is reflected back
    - When little light is reflected back the device outputs HIGH
    - When LOTS of light is reflected back the device output LOW
- Control sensitivity of sensors using potentiometers
    - Potentiometers: control amount of resistance using a “wiper”
        - Changes the distance the electrons must move
        - Turning the potentiometer LEFT will “loosen” it and allow more current to flow making it MORE sensitive
            - Every little change read by light sensors will be transmitted/sent to arduino
        - Turning it RIGHT will “tighten” it and allow LESS current to flow through making it LESS sensitive
    - Ensure both line trackers have the same sensitivity to prevent minor flaws
    - Black tape will be provided to test the program

IR OBSTABLE AVOIDANCE NEXT - PROGRAM IS VERY SIMILAR TO THE ABOVE, so do the same - guide through logic, tell them to try to convert logic into programming/help...and test

IF FINISH W/TIME.... START ULTRASONIC NO SERVO.... SAVE TIME PLZ

DAY 8

Make sure you understand the code and schematics for the following configurations. ENSURE YOU KNOW WHY IT WORKS and not only how it works.

Robot Configurations

- **Ultrasonic with NO Servo**
    - Introduce concept of ultrasonic sensor and its function
        - Ultrasonic: Sensor used to measure distance
            - Consists of Trig and Echo pins
                - Trig is used to output a frequency wave
                - Echo is used to input the SAME frequency wave
            - Since sound is reflected upon most objects, the time it takes to travel can be used to calculate the distance
            - Always use centimeters
                - More exact and precise
        - Show a sample on TinkerCad or breadboard to demonstrate ultrasonic
    - Make sure to test the code along with the arduino to ensure that everything works properly
    - If finished earlier than expected, allow the children to play around with it
- **Ultrasound with Servo**
    - Explain the function of servo

        - Servo
            - Only turn 180 Degrees
            - Used to output a precise value
            - Called an "Actuator"
            - Consists of single pin for signal
                - Signal takes in input from the arduino and outputs using servo
                - Signal pin can control angle the servo is at
    - Both can be used to prevent the use of multiple ultrasonic sensors (ie. reads the distance everywhere in front of it!)
    - Make sure to stop within a safe distance of the object to account for momentum
    - Write a program to look for the least obstructed path between front, left, and right
        - Move accordingly, else move backwards
    - If finished earlier than expected, allow the children to play around with it
- **IR Remote**
    - Assign functions of moving forward, turning left, turning right, and moving backward to buttons on the remote
    - **Explain the code thoroughly, as the code may seem intimidating at first**
    - Explain the application of IR remote, allows for human input without having to physically touching the arduino or any of its components
        - **MUST POINT IR REMOTE DIRECTLY TO ROBOT**
    - Keep the program simple, but only using the 4 buttons for moving
    - If finished earlier than expected, allow the children to play around with it

SPEED UP ON EARLIER PROGRAMMING DAY IF POSSIBLE, FINISH IR TODAY!

DAY 9

Note - set up field for them to practice!

For all coding/understanding competition - help them with ideas of code….driving plans…etc

Tournament Preparation - MAIN FOCUS - Allow enough time for this

- Hand them their 3D printed part
  - Have them attach it to their robot
- Day to modify and make the robot ready for competition
- Explain the logistics of competition
- Competition
  - Explain they will be competing in teams of 2, at end of match both get points scored in match…. At the end… person with most points wins
    - **No alliance stuff cuz then some kids won't pick eachother… feel left out**
  - **Auton highest will win a prize also - variable based on time on comp**
    - **2 attempts!**

- Autonomous Period - maze (circular) - easier for them to program ,and has to be auton...can't premeasure values much - try to get as deep in field as possible! - more distance is more points
    - Is going to be duck tape in the maze (black)
    - Essentially, can us IR light or distance or ultrasonic
- Driver Control phase (IR Remote)
    - **Presplit groups for remotes and give them code to upload to use in this competition... not enough time**

- Test ALL CODE on the field to ensure everything works
    - **As an instructor it is your duty to ensure that everyone is prepared for the competition**
- **PRACTICE PRACTICE PRACTICE**
    - Ensure children understand their robot and how it works

DAY 10

2 auton attempts - 15 kids, 30 min

1:30 driving phase

Set up website to display what ranking kids are at..points scored

- **Tournament Day**
  - Ball Competition
    - Place most balls on one side
    - Robot must be moving (Inspection phase)
    - Avoid walls and balls (Skills Test)
  - Autonomous Period (Maze)
    - Create a maze designed with black tape
    - Use line tracker and ultrasonic to ensure that robot stays on track and does not crash into objects
      - **Dont tell them what to do/how to do the auton tho… leave sensor choice up to them**
  - Trophies for 1st place
    - Rest of the certificates for participation in summer camp
    - Follow VEX tournament style
- Group picture
  - **class photos will be taken/with instructor**
  - **SMILE FOR CAMERA**
- Lunch
- Conclusion
  - Pack up all equipment/get ready to use at home
  - Intro to VEX robotics - if want to continue stuff like this (show online website...competition we are doing this year and stuff)
    - Highlight how they have auton and driving… u did same… same design/building process you guys went through yesterday(day 9) - but at a much higher level
      - If you are interested in doing things like this…. **Def think about joining/coming to IHS robotics once u reach HS!**

- **Leave children on a positive note**