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 8th-19th Age groups: 7-9, 10-13 9AM - 1PM, 2PM-5PM Monday - Friday

Camp Curriculum/Agenda

Property of **Ansh Verma & Sandesh Shrestha & Patrick Ding**

Lead Curriculum Developer: Ansh Verma

Collaborators: Patrick Ding and Sandesh Shrestha

Irvington High School Robotics

Ansh -

510-512-0863

asapansh@gmail.com

Sandesh -

510-364-2502

sandeshshrestha02@gmail.com

Overview/Goals

TimeLine:

1) 1st and 2nd day circuits
2) 3rd day first 1:30, build bot, 1:30 CAD
    a) Never take apart bot after build it!
3) 4th day 3 hrs @ CAD
4) 1:30 CAD friday 1:30 programming concepts
5) Next mon, 1:30 programming concepts, 1:30 PWM/making both example circuits/programming
    a) Note - second circuit, motor one, do with robot - so assemble first schematic and get this out of the way
        i) If finish early, make do diff stuff with PWM, one could be speed up, other could be slow down
    b) Note, "programming concepts" is essentially intro to programming with following along examples @ same time
6) Schematics 2, 3, and 4 make with testing out
    a) **Instructors need to make obstacles/games that work with these programs**
    b) Explain caviats of the infrared sensor program
        i) Fix the infrared sensor program @ same time
    c) Button sensor
        i) Use the correct one!
    d) If dont get to 4, its ok, try to start atleast
7) Schematics 4, ultrasonic with servo, IR remote
8) IR REMOTE/BLUETOOTH, give them their 3d printed part,have them attach
    a) Bring field from this day - emphasize driving is super important - practice before just go on - test code before and modify to make driving easiest
9) Give them a day to modify their code/circuitry with knowledge they know to create a bot that can....
    a) Explain ur competition
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) Group picture/pack all stuff up as they are taking it home

- circuitry
- Programming
- Build bot and apply concepts to do a few tasks
- Go into specializing in one task camp instructors decide for end of camp game, have them 3d model and print stuff to accomplish this
- End of camp game, awards and certificates for everyone

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

Tell them to bring folder to keep handout/helpsheets

If needed, teach how to use computer first, ie. how to save files, etc

10-13 year olds:

Programming

- 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 (each distinct version as a sep file, rest same as programming file doc)
- Handout (like a Lee or Chung helpsheet), with instructions on how to do things taught in the 3d modeling portion in case they forget and want to do it later

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)
- Take bot home, so this reminder of mech design

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

3D modeling

- Same file organization, tinkercad help sheet - rlly basic

Circuitry

- Same helpsheet as 10-13 (with symbols for circuit components, etc), less advanced info
- Tinkercad build circuits, file organization, draw out before build on tinkercad
    - Dep on computer savyness (none), prolly actually just drawing out, no need to simulate as much more basic info

Notes:

NOTE - FOR ALL LUNCHES, TALK TO THEM/HAVE THEM PLAY BOARD GAMES/WII

NOTE, PEOPLE WHO R DOING 3 DAY CAMP, 1 WEEK CAMP -FULL DAY, AND ANY 7-9 year olds, MODIFY CURRICULUM

-acitivities alongside lessons to re-enforce knowledge fun way

Ask questions alongside lesson to keep kids engaged

-make slides for each day 4 visuals/to help keep all instructors on track to teach the same info, same order

- need to make helpsheets for each day for kids

DAY 1

1. 9-10 (Beginning Introduction)
    - Divide Students into groups of 3, with one teacher per group for more individualized focus on each student.
    - Note- each camp instructor will need to understand all the curriculum in the camp, as they will be teaching EVERYTHING
        - Put into premade groups based on skill level/parent requests
        - 10 min introduction in teams,
            - school, name, one interesting thing about them, others have to remember, ask later (if applicable to group - ie. they don't know eachother before hand)
            - Could play uno or smth before-hand for bonding, but lw awk
        - Take time to gauge familiarity with equipment using in this camp/4 main engineering disciplines teaching this camp
            - Thru small activities or questions - ie. do you know how to save files on computer, maybe like small survey to get general skill level
    - Discuss Summer Camp Project (note: make sure to have different explanations for different age groups as they will be learning subjects to different levels - 7-9, 10-13 groups)
            - For documentation during this camp (necessary in any engineering), please see the overview section
            - CAD (3D Virtual Design) (4)
                - 10-13, fusion (can use worse computers, can access from multiple computers/locations-reasons why use fusion)
                - 7-9, tinker

        - Show them 3d printer, with 3d design you will be 3d printing your own parts!
    - Basic Circuitry (2)
        - See docu under overview
    - intro to arduino (1)
        - 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 (3)
        - See documentation under overview
        - Many programming languages - remind this isn’t only language but is good to know
        - Use this language for arduino, commonly used in robotics
        - **Give them/show them all their equipment, explain what it is going to be used for alongside the explanation of what they are going to learn in this camp**
            - **Ie, open kit, first I want you to take out….. Ask them what they think it is for/explain uses in this camp to help them learn/how it will be used on robot**

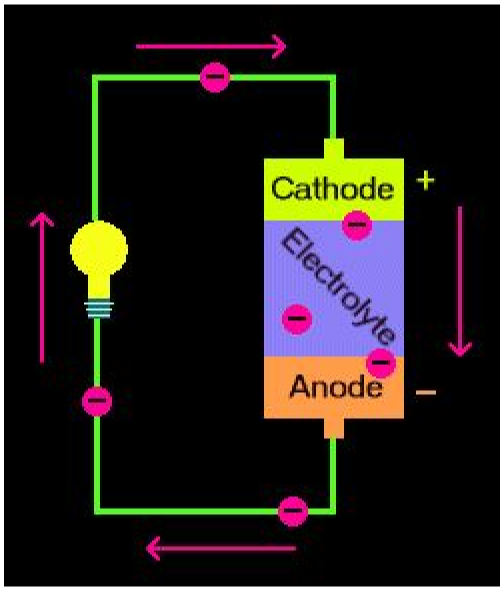
2. Note - since we MIGHT be pressed on time, start building on first day, then once bot is build, teach all 4 focuses of camp through robot!
    - Prolly not tho, cuz can’t really teach general concepts thru the bot trying to build - it’s limited functions (and builds that accompany these functions) - don’t include use of some circuitry concepts ex.

WILL NOT TAKE 1 HR!

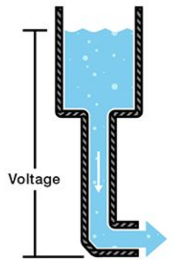
3. 10-12 (Intro. To Circuits)
    - **Props to make more engaging - magnets (attraction - potential energy anode/cathode), tubes (vs wires, display how wire thickness affects current)**
        - **We have a premade, simple circuit with battery, LED, switch to demonstrate to them, make/add on to our circuit as they do**
    - 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 note.
        - If the diameter of the wire in which the electricity is running through decreases, will the current increase or decrease? Why or why not?
    - **NOTE - make helpsheet for this (inc voltage,current, and water pipe analogy)- give to kids**
        - **Inc schematic common component symbols**
    - 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
    - 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
    - What is Current? Voltage? How does a battery "supply these"
        - What is a battery?
            - 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
                - Electrolyte Medium (in between Anode and Cathode, react with atoms in Anode and Cathode to make them form positive and negative charge ions)
                    - Also act as a buffer, electron transfer from the anode to cathode CANNOT happen through here
                - 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
                        - 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!
- Current is the speed of this “flow” of electrons from the anode to cathode
  - Flow of electrons vs time
- Voltage is the measure of the potential energy from the anode to cathode

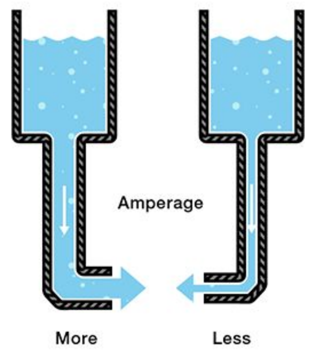


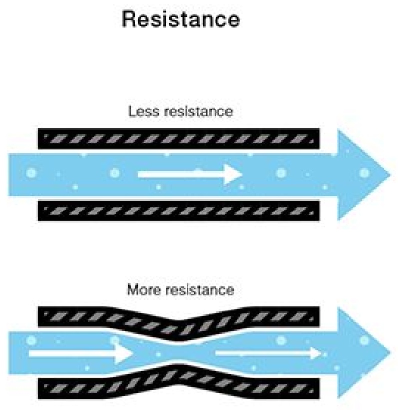
- Water Pipes Analogy

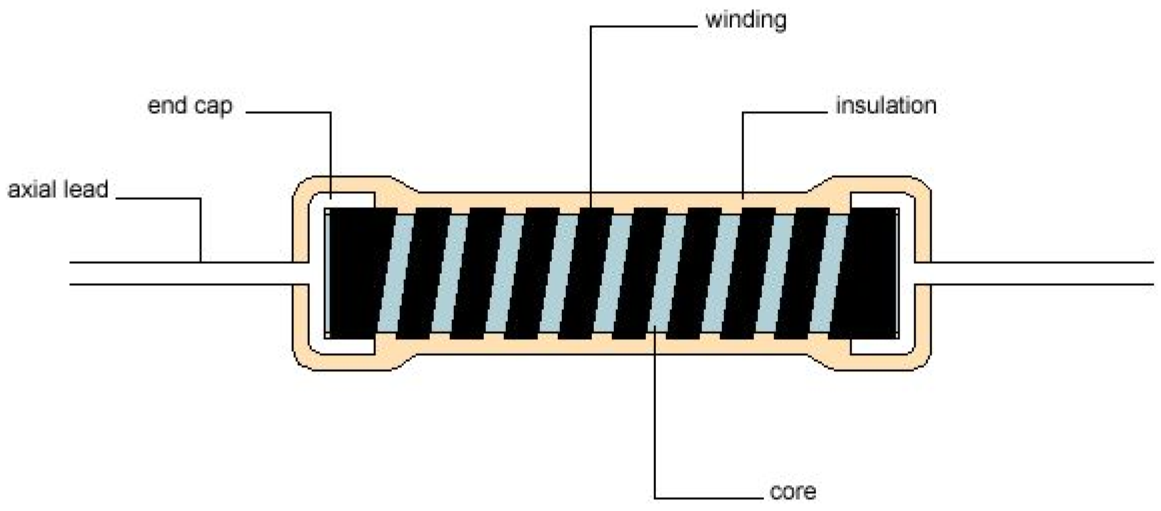


- Reservoir of water can be compared to a battery
  - More buildup of water in reservoir, faster water will flow down pipes due to pressure
- What does this mean in terms of voltage and current? -- question to kids
- **More voltage - more current**
- Speed of water flow can be compared to current

- 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 also, like circuits with “resistors,” basically temporary kinks in circuit that decrease current flowing in circuit

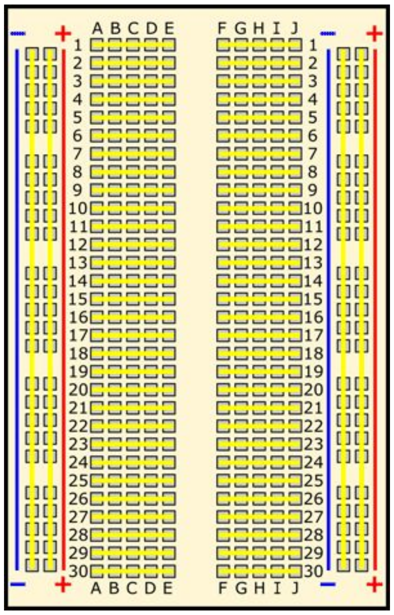




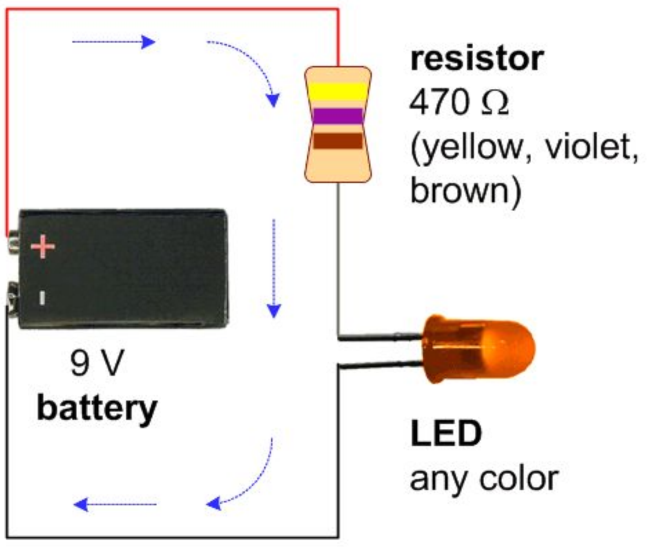


- How do Resistors work? - coils, more coils on resistor, longer the moving electrodes have to travel,
- **Have them take it out and use as a visualization tool**
  - Moving the anode and cathode farther away from eachother, increase in distance effectively decreases potential energy/voltage between the two as less attraction between the two, therefore slower flow/current
  - Also think of it as more kinks in the circuit, slows down current

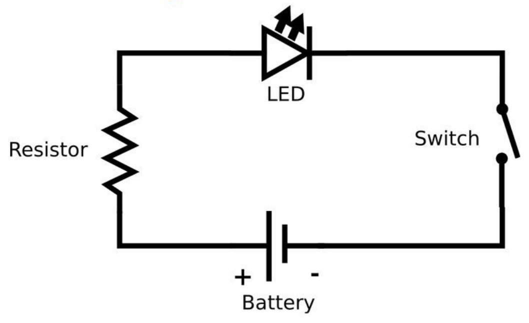
- BreadBoards **(time to apply our knowledge!)**
  - Big problem circuits is keeping track of wires
    - Breakboard link many wires together, provide easy way to make/keep track of circuits
  - Rows are internally connected, all on same circuit
  - Left and right (with +/- signs), vertically connected, not horizontally like other rows
  - If you connect the anode and cathodes of a battery to the +/- collumns, what does this mean? open/closed?
    - Closed bc still no connection between postive and negative cathode/anode
  - Common Practice to put battery wires on “power strip” (+/- rails) to make circuit easier to “read” later



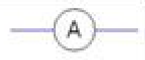
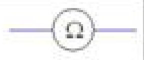
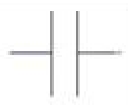
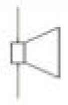
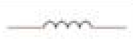
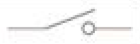
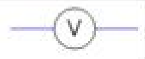
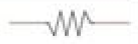
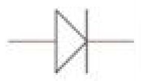
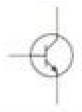
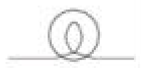
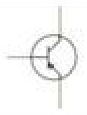
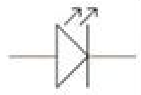
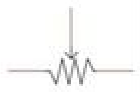
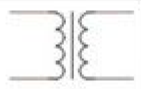
4. Creating Simple LED Circuit
   - Create a circuit with LED/Battery, and then LED/Battery/Resistor, show how resistor affects LED brightness and reinforce how to use a breadboard
   - LEDs have a positive, and negative side
   - Longer leg is positive, shorter leg is negative
   - DO NOT TAKE IT APART UNTIL END OF THE DAY
5. Schematic Diagrams (don’t disassemble the final breadboard model yet!)
   - Used to model circuits, with universal symbols to make it easier for everyone to understand
   - In general, draw schematic first before make circuit, this time is an exception



Circuit Diagram



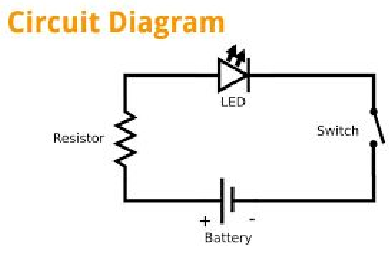
- Given schematic symbols for circuit components given to you in helpsheet, try to draw schematic of circuit you made!
    - Do on back of helpsheet
    - This image has switch, what do you think switch does? Lets add it into our circuit and your schematic
- Common Schematic Symbols (explain each) -- will be using in drawing more schematics/making more circuits later!

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

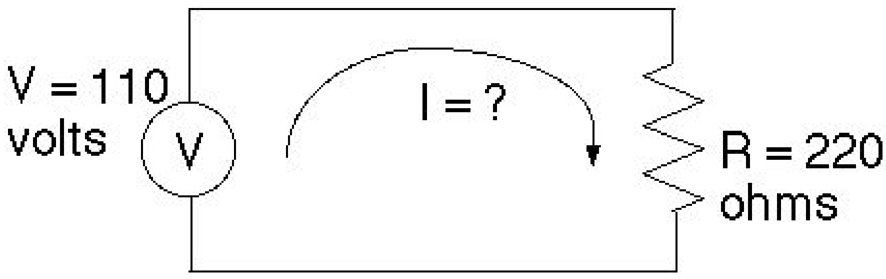
Series Circuits:

- Through Schematic Diagrams first
    - Series circuits

        - all components of circuit are connected directly to the battery
        - If one connection is taken out, circuit is open, doesn't run
        - Our previous circuit/schematic was a series circuit!
        - Take out LED/open or close switch - tuens on/off whole circuit/all components in it!
        - 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
            - Again shows how all components are interconnected, changing part of the circuit changes the state of other components



    - Series Circuit Characteristics:
    - LED's add 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
    - This relationship can be seen with **ohm's law, V = IR - fundamental law**
        - 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
        - Dont need to focus to much on this
    - Example problem using ohms law/ adding up all resistances in series circuit
        - 10,12,14 ohms respectively for three resistors (ohms is the unit for resistance)



- **Lunch Break - 12-12:30 -- Bring Switch, play Smash, or other activities**
- 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
- If have time, start parallel circuits
  - Therefore, keep all help-sheets handy and all instructors should be ready to start teaching this

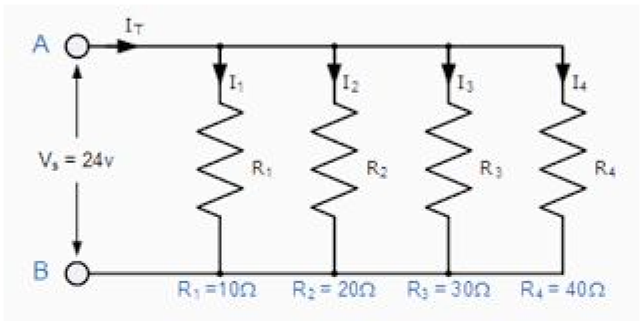
END OF DAY 1

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, leave space to later add resistors, etc in each component (the visual)**
  - **NOTE - with description of parallel circuit (resistors to less resistance and more current), use multimeters on your, and their circuits so they can see what's happening! - engaging activity**
- Transition from yesterday's activity, ie. if didnt finish that yet, then finish it, and start this! - say smth like oh this is a parallel circuit woohooooO! Lets talk about its characteristics/why it allows u to separately control all three LED's
- Parrallel 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 get at once
- Each LED/part of component not connected to eachother, one "column" in a parallel circuit is not connected to other columns

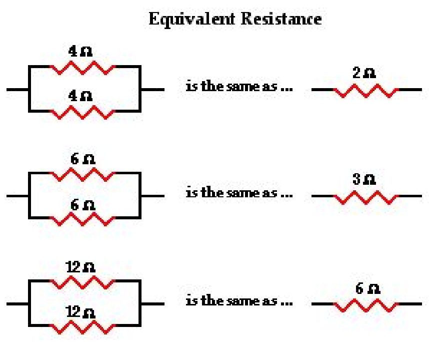


  - **Give visual example with a 3 component parallel circuit, put switches in each circuit to show that if turn off 1 component LED, doesn't effect the rest**
    - **Activity ended with yesterday, but show again for re-enforcement**
  - Emphasize negative to positive flow for electricity, map it with finger or smth
  - Switch causes circuit between negative to positive to break in component, why LED turns off
- Each component can be treated like a series circuit
- Build a parallel circuit!
  - 3 components, each with led - leave3 space to add resistors
  - Make schematic before build, utilize component symbols - add resistors on schematic too later
- Through experimentation, ppl figured out that as a whole parallel circuit, an increase the resistors, increase the current also! Whatttt?!?
  - Adding resistors to each component essentially increases the amount if "wire"/distance electrons need to travel to get from cathode to anode in each component of parallel, this means electrons will be less "crowded in each component" as it goes from the main branch to side branches. This lack of crowded-ness in each side branch allows electrons to flow at a higher rate from the main branches to side branches to the cathode, which intern increases the current.
    -
    - FalseIncrease in distance does decrease the current, but the space in each component to allow more electrons to fill each component before the negative-negative charges repel takes priority over this increase in distance, intern increasing the current
    - **Show them this by having them add resistors into each component, measuring new current in circuit**

- **TEACH STUDENTS HOW TO USE MULTIMETERS FOR ABOVE (10ISH MIN)**
    -
- More resistors === less resistance in a parallel
    - experimental conclusion
    - Formula is

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

    - Try out formula to see that more resistors == less overall resistance
        - Simple calculation, remember level of students teaching
    - Do calculations to the right,with doing calculations, emphasize the conceptual side of it --- why more resistors decreases overall resistance
        - **Have a worksheet or something - have them make a bunch different circuits, with multimeters meaure resistance and calculate it!**
            - **Or maybe just a few calculations**



    - more resistors is less resistance, ohms law still applies!
        - More resistors, Less resistance, more current with V = IR
        - Note, before curent goes into the components and when it comes out, the current is the same, therefore 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 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!**

- 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!

10:30-10:45 -- BREAK

10:45-12, BUILD SMARTCAR

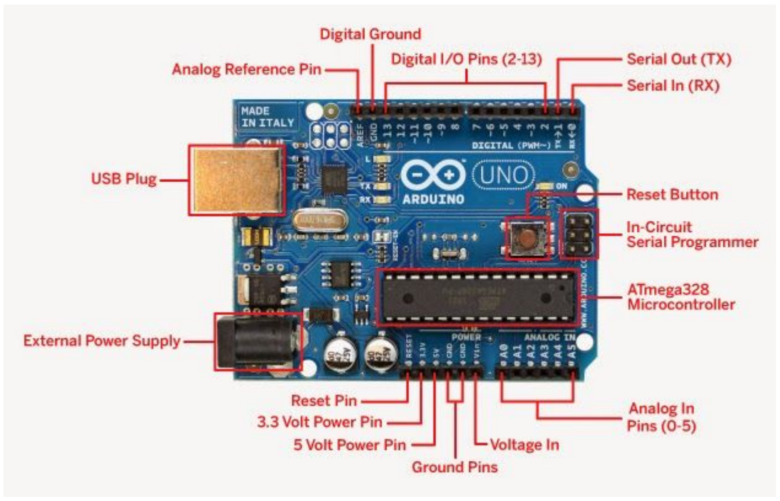
- Purpose is to fit in CADDing earlier so we have enough time to print all their part

10:45 - 12 - INTRO TO ARDUINO

- **Take out arduino/use as a visual the whole time**
- **Provide help sheet with all info go over**
- What is an arduino? It is a microcontroller!
    - What is a microcontroller? Essentially the "brains" of the machine
    - How does a "microcontroller" control a machine/use your code?
    - Through transistors in the micro-controllers --- like neurons! In our brains!
    - What are neurons?
        - Building blocks of our brains, form links to send signals from one part of the brain to another, and many signals fired at the same time produce "thought"
        - Think of it like this - when a neuron fires a signal, it is "on", otherwise "off"

- Transistors are in the arduino's we will be using tomorrow, micrcontroller neurons, these on/off switches that controt of current of electricity flow in arduino - like sending a signal in the brain
- Each block of code is executed by arduino through these electric signals by transistors -- sends these signals to the ports of the arduino (show the ports), which then control circuits, other machines, communicate with other machines, and communicate withr computers! Transistors have given machines "brains" essentially, to understand/communicate information in binary

- 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**
        - **With everything ask hella questions to remind them of previous content, ex, what is voltage again? Ie. what is the arduino really measuring?**
        - **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!**
- 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)/7-12 volts (depend on pin u pick!)
            - 20 volts is max
            - Any more, can FRY arduino - dont want that -
                - Transistors cant handle current (all electrons approaching n channels at once), breaks
    - Reset pin - resets pin in your arduino?
        - 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 yesterday, when add battery to arduino circuit, ground and positive must line up to battery positive and ground
            - 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 for any component to be in a closed circuit for it to function in the circuit (what arduino is doing - functioning in circuit to read its data (current/voltage) or manipulate it)
                - This allows for the arduino to have a reference to (0 Voltage) when measuring the voltage at other places in the circuit through analog pins, 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 occured, it would just be reading randomn numbers (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, ground voltage is 0 now!

- Analog reference pin - sets reference voltage for arduino to refer to when seeing voltage of other analog pins
    - Gets voltage from ground pin if plugged in, otherwise plug this pin into circuit!
    - **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**
- Analog ports 0-5
    - 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
    - Arduino understands digital levels/numbers 4 code
    - Ex of a similar conversion - microphone on phone
        - takes in waves of sound (talking produces waves) - measures certain aspects of the waves sound produces (ex. Time between the start of each wave), compares it to a preinstalled reference fro the measurement, then converts digital levels/numbers for phone to understand and interpret
- Digital Input/Output Pins - pins 0-13
    - Difference this and analog - digital is only 1/0, while analog is 0-1024
    - 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
        - **Talk abt this l8ter (tmr)**
    - Each pin can intake 40 mA, but recommended is 20 mA
        - More than this, fry arduino! - make sure good before connect to arduino
    - 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 rules/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**
            - **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 FOR 10 MINUTES, KIDS FS TIRED

PROLLY 11:20Ish

FINISH EARLY? - START WITH DOWNLOADING ARDUINO IDE/EXPLAINING HOW TO USE IT, OR START LUNCH EARLY, THEN DO THIS AFTER!

- **NOTE, WITH WHOLE EXPLAINING IDE, HAVE A BASIC, BUT COMPLEX ENOUGH PROGRAM TO DESCRIBE ALL OPEN, USE IT TO HELP DESCRIBE THE IDE**
    - **Maybe one of the example programs**
    - **Have them open the example program on their computer also, and have them follow along with the examples you are giving**
- 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 to download website and download it! Even though online version, so there is the same curriculum for everyone
    - For people who are not using their own computer, have them send stuff to their email once done
- Explain how IDE works
        - Top "void" thing is to **define** all stuff using in program - viod setup
        - What does that mean?
        - In a program, we give names to numbers, and a general pair of instructions that is used multiple times in the code, and is therefore redundant to write out repeatedly, also for where put certain ports equal to variables, so can call upon in the code
            - All this is written in the setup area, and used in the following void
        - Void loop
            - Where use premade sets of instructions/declarations to write instructions for arduino to follow, what the arduino executes
- Get general consensus of programming knowledge
- 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
    - Open up Arduino IDE (downloaded)
    - Ask students about buttons they see
        - Main workspace w/ main code
        - Black Console output for code uploading and error msgs (at bottom of IDE)
        - Tab w/ code name - each tab is a different program file
        - Verify & Upload Btns, New,Open,Save buttons
        - Top Dropdowns:
            - Classic File tab
                - Open new files/old files
                - 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
                - 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

        - 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 or smth
    - Paste keyword is good to paste code, but really easy keyboard solution for this too
        - Show kids, control shift 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 numbers, words (say strings later), etc) - for example, say x =5, can use x in place of 5 for the whole program
        - 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) - dont tell them now, then these are necessary for the program to identify which blocks of code are in the structure, which are not

- Sketch w/ verify repetition & 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!
    - 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
    - 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
- Tools w/ library manager, serial monitor (Show serial monitor shortcut btn), and platform options
    - Manage libraries, again since we dont need to worry 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 as this only has one port, sometimes it doesn't ask for it
    - Serial Monitor and Serial Plotter
        - **Use one of the examples and the serial monitor when run program - have arduino connected/use at the same 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
  - 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
  - Will plot the data points gotten against time, allow you to see if you are getting the correct data you intended to get to allow program to work, adjust program accordingly!
    - **Give an example of when need to use this**
  - Serial monitor - just data values
  - Serial plotter - graph
  - Everything else ignore for not, not really relevant

LUNCH BREAK - 12:10-12:30, lunch break!

SHOULD TAKE REST OF THE DAY TO FINISH ALL ABOVE CURRICULUM WITH INTERACTIVE LEARNING - HAVING THEM USE AN EXAMPLE PROGRAM TO LEARN ALL STUFF ABOUT ARDUINO IDE, TRY TO FLUSH OUT ALL INFO/EXPLANATIONS TO MAKE TIME MOVE SLOWLY

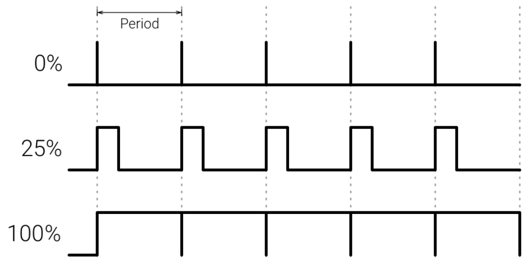
IF NOT….. EAT INTO DAY 3 BEGINNING

Day 4:

- **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)**
    - **Ik curriculum for this day seems/is really disorganized rn, but when make helpsheet it will make more sense**
      - **Hella pictures of duty cycles to reinforce concepts**
  - **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)`
      - 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
  - **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**

Start with assembling kit (Ansh mades notes on faster ways to do it when doing it himself - add when sandesh comes back and gives ansh the kit)

Note - each instructor should HAVE pliers to break off the sides of the bot when assembling

- **1:30ish tho, so till 11 including a break!**

12-12:30 Lunch

- AFTER LUNCH, CONTINUE WORKING ON PROJECT!

SANDESH'S STUFF BELOW, KIND OF LIKE THE EXAMPLE PROGRAMS WE WILL HAVE SET FOR THEM DURING TEACHING PROGRAMMING DAY

- Activity - integrating C syntax they know and breadboards - example how to light up LEDS
    - Knowledge needed: breadboarding, wiring, coding w/ ports
    - Materials: LED, resistors, wires, breadboard, arduino, psu, coding PC
    - Tutorial (make it general to see if children can figure parts out:
        - Place LED on breadboard with resistor. Lead to connection to Arduino
        - Set pins and turn on LED in program for set amount of time
        - Turn off LED and program completes
        - Challenge: Create LED pattern that loops (be creative)
    - Debug tips: use Serial.print() in between actions to act as checkpoints
- Code naming:
    - Briefly describe naming code files for versions and purposes
- Sensors:
    - Sensor's role in programming: allow robot to sense surrounding (touch, light, distance, etc...)
        - Button: detects touch, outputs on or off
        - Ultrasonic: detects distance, outputs range of values
        - Photoresistor: detects light, outputs range of values

DAY 5

- If/else switch:
    - Switch to determine which path to take
    - Syntax: if (Logic) { } elif (Logic) { } else { }
        - Example if button is pressed then turn on LED, else keep LED off.
        - Elif for replacing if, if, if, if etc...

        - Button Logic: When button pressed, digitalread(pin) == HIGH
        - Advanced button logic Rising & Falling edge (change from LOW to HIGH and ViceVersa)
- Activity - Adding Button to LED program
    - Knowledge: Same + Button Logic
    - Materials: Same + Button
    - Tutorial:
        - Add button to breadboard and wire it to digital pin
        - Add setup pin in code
        - Setup if statement in loop to set LED to high when button pressed
        - Challenge: convert button press to a latch (button turns on and holds until button pressed again)
- While loop:
    - Continuously loop through code until condition is no longer met
    - Syntax: while (condition) {code}
        - Condition usually logic
            - Ex: DigitalRead(5) == HIGH, repeats code until pin input isn't high; True, always runs similar to void loop() {}
            - Usage includes "irobot" movement: continuously move until hits wall, then turn and repeat. Would look like:
                - Void loop() {
                - While (Button not pressed) {move forward}
                - Turn 45 degrees }
    - Activity: Use button and turn on LED while button is pressed
- 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
- Ultrasonic sensors:
    - Logic: Outputs distance as either conversion from distance or as library module number output
    - Activity: Add Ultrasonic sensor to the While loop program for when distance is close
- For loop:
    - Loop code until counter met
    - Syntax: for (initialization; condition; increment) {code}
        - Initialize counting variable: int i = 0;
        - Condition needed to keep looping: i < 5; loops until i > 5
        - Increment every loop: i++ or i = i + 1; adds 1 to i every loop
            - Ex: for (int i = 0; i < 5; i++) {
            - digitalWrite(LED) = HIGH;

            - delay(3);
            - digitalWrite(LED) = LOW;
            - delay(3); }
        - Will turn on and off LED 5 times
        - Why not repeat code 5 times?
            - This is very simple use case where it is possible, other times this is most efficient for coding or code inside for loop changes condition of loop
    - Activity: Use for loop to make any phrase print to Serial Monitor 8 times
    - Activity: Use for loop to turn on line of LEDs consecutively
        - Breadboard with resistors and >4 LEDs
        - Connect all to digital ports
        - Config pinsetup (Could use for loop to set all as OUTPUT)
        - Turn 1 on then off and continue to next
    - Challenge: Combine while loop program and for loop to turn on LED with button only 4 times
- 9-11
- 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}
- 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
- 11-12
-
- 12-1

DAY 5

- Whole day, go through the rest of arduino examples, go through ⅗ other smartcar things
- Coding the bot:
    - Movement:
        - To move the bot, must control the 2 motors on Left and RIght of bot
        - Basic movement logic: both forward = forward, both backward = backward, left forward right backward = spin right, left backward right forward = spin left
        - Coding:
            - To configure each motor as pwm to the Arduino, each motor requires two output pins, therefore 4 output pins total in setup
            - Ex: LMotorForward & LMotorBackward
                - To activate forward movement, use:
                - digitalWrite(LMotorForward, HIGH)
                - digitalWrite(LMotorBackward, LOW)
                - analogWrite(LMotorForward, 200)
                - analogWrite(LMotorBackward, 0)
                - 200 is speed from 0-255. Repeat for Right Motor. Use delay() to change duration
            - Activity: Move the bot forward and spin
            - Use functions to create forward, backward, spin left/right, and brake mvmt with time duration as parameter
    - Buzzer:
        - Simple component used to make sound
        - Use as a digital output with digitalWrite(buzzer,HIGH)
        - Activity: Use button to play sound whenever pressed
            - Use if statement in void loop to check if value of button is HIGH
            - When pressed set buzzer to HIGH
    - Line Tracking:
        - Using 2 sensors, detects white and black
        - Continually updating which sensors “on” and “off” to locate line
        - Line logic:
            - With robot aligned on line, neither sensor activated
            - Continue forward until either is activated
            - If left activated: turn left until right activated
            - If right activated: turn right until left activated

        - Use curve turns not spin turns
        - Code with while and if elif
- Ultrasonic Sensor & LCD
    - In order to use LCD to display, need to import LiquidCrystal.h library
    - Need to create lcd with pins to setup display. Use LiquidCrystal lcd(7 pins); Pin order: RS pin, EN pin, 4 data pins, Ground pin
        - RS pin used to choose mode of control (instruction/character mode)
        - EN pin is like a on/off switch to allow data to be input to display
        - 4 data pins for data
        - Groud pin to complete circuit
    - Variable Declaration "trick": To declare multiple variables of same type most efficiently, use commas
        - Ex: const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
    - In setup function, use lcd.begin(dimensions) to start using the LCD. We using 16x2, meaning 16 characters across by 2 lines up/down
    - To display test starting from top left, use lcd.print("hello world")
        - Activity: use lcd.print("") for any text and number
    - Other functions:
        - lcd.clear() to remove all text
        - lcd.home() for returning cursor to top left
        - lcd.setCursor() to move cursor to wherever
            - Ex: lcd.setCursor(6,2) moves cursor to 6th character on 2nd row
        - lcd.cursor() to show cursor blinking, lcd.noCursor() to show no cursor
        - lcd.blink() similar for block cursor blinking at 500ms
        - lcd.display() to turn on display, lcd.noDisplay() to turn off
        - lcd.scrollDisplayLeft() & lcd.scrollDisplayRight() moves text
        - lcd.autoscroll() to scroll text up to 40 characters, lcd.noAutoscroll() to turn off
        - lcd.rightToLeft() switches "hello" to "olleh"
        - lcd.createChar() to create custom characters in 5 px wide by 8 px high
            - Use binary matrix to choose which px on or off
            - Syntax: byte customChar[8] = {0b00110,0b01001, 0b01001,0b00110,0b00000,0b00000,0b00000,0b00000}; Creates degree symbol
            - 8 "0b00000" signifies each row of 5 on or off
            - Use lcd.createChar(0, customChar) to create character called 0 using customChar Matrix

            - Use lcd.write((uint8_t)0) to display character
        - Activity: Create custom chars and make something special. Use imagination
    - Ultrasonic sensor used with Echo pulse and Trig pulse to measure distance
        - Define Echo pin as input
        - Define Trig pin as output
    - To use ultrasonic sensor, Trig must pulse and Echo will measure intensity
        - Use digitalWrite with Trig. Set low for 2ms, high for 10ms, and low again.
        - Measure with pulseIn(Echo, HIGH) and set value to float Fdistance. Divide Fdistance to get distance in cm
    - Activity: Create function using logic shown to measure distance
    - Activity: Create program using function and LCD to display updating distance of objects
- Activity: Use Ultrasonic sensor to move robot like iRobot (move until obstacle, rotate randomly, continue)
- Servos:
    - Continuous vs nonCont Servos: Cont spins 360+ degrees like motors, nonCont spins 180deg only
    - Used for precise rotation positioning
    - Controlled through PWM
        - To control, create ServoPulse Function to make positioning easier and faster:
        - Function with angle integer parameter
        - Convert angle to pulsewidth by *11 and +500
        - digitalWrite(servo, HIGH)
        - Use delayMicroseconds(pulsewidth) to move servo to desired angle
        - Stop servo with dWrite(servo,LOW)
        - End function with delay
    - Activity: Control Servo to measure distances of N, NW, W, SW…
- IR Remote:
    - IR Remote uses infrared light and sensor to send signal to the arduino. Same use in TV remotes
    - Usually sends code in HEX code
    - Must use IRremote.h library
    - Library functions:
        - IRrecv irrecv(pin) creates receiver & initializes
        - irrecv.enableIRIn() starts receiver in setup func

- irrecv.decode() decodes parameter into hex code

DAY 6

- CAD day - basics of how to use software - with an added project
- File organization-projects and stuff
- Begin part file practice
    - Sketches - start on xy for consistency
        - Basic geometry, pattern tool
        - Dimensioning
        - Construction lines
        - Constraints
        - Editing sketches - remember to right click and edit sketch, not make a new sketch
    - Controls
        - F4 for rotate
        - Middle Mouse to pan
    - Extrusions
        - Need a polygon
    - Rotations
        - Need a polygon and axis
    - Sweeps
        - Need a polygon and path
    - Lofts
        - Needs 2 polygons
    - Filets and chamfers
        - Needs an edge
    - Construction planes
        - Various types
- Assemblies
    - Just join things lol

DAY 7

- CAD the chassis given to you/put it together, try to utilize functions you learnt yesterday
- Help kids cad
- ⅕ other smartcar program

DAY 8

- Advanced CAD procedures teach all! With a project
  - Use pltw train challenge
  - https://forums.autodesk.com/autodesk/attachments/autodesk/78/340387/1/project2_3_1aminiature_train.pdf
- Teach them how to use sensors to stop robot in front of an object (LAST SMARTCARE PROGRAM)

DAY 9

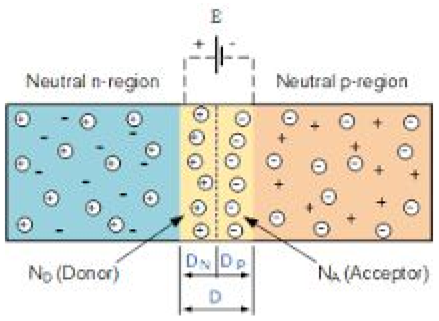
- BACKUP DAY, because stuff will def go behind schedule
    - Once camp has started, gauge how fast moving/breaks kids need, and accordingly adjust curriculum/use this day to finish all in time
    - I'd run out of time/too fast, add projects!
- Create an autonomous period (program), and customize driver control to get ready for comp tmr! practice!

DAY 10

- Ball Competition! Whoever can put all the most balls on side wins! Trophies 1st place, rest trophies for participating in summer camp and certificates, maybe individual awards for each kid, individual awards for best at learning, more enthusiastic , etc
- Group picture and lunch, bye bye!
- Mock vex comp? Interviews, other awards, all that stuff?

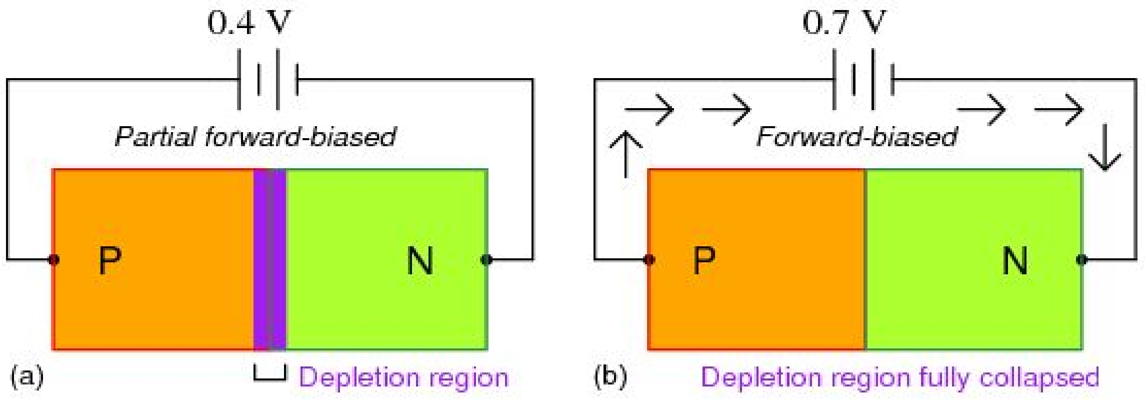
EXTRA INFO:

(FROM DAY 2 - REMOVED BC TIME AND TOO FAR ABOUT THEIR HEADS)

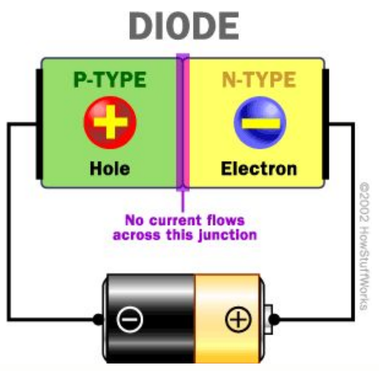


9-10:30 - Introduction to semiconductors/transistors

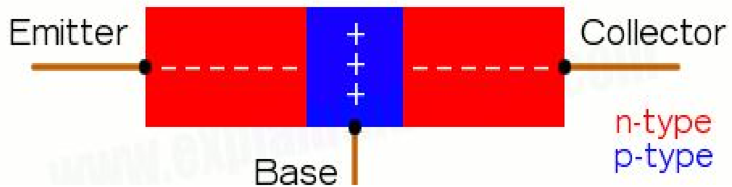
- LEDS are semiconductors!
- Make LED circuit, turn it on and reference to it as explain concept
  - employ p and n junctions (that we just used in our circuit!)
    - Semiconductor elements -- look at periodic table, silicon
    - N-type
    - N Type --- extra electrons
    - P type -- electron holes (this is what positive is)



- inbetween p/n type is a junction, where electrons can flow from n to p due to holes that need to be filled up, as flow across, electrons give out free energy, this is light energy
- Electricity flows only one way, if battery anode/cathode were reversed, electrons would not flow through depletion region, and nothing would light up
- Current from anode flows to p type, n type electrons flow to cathode

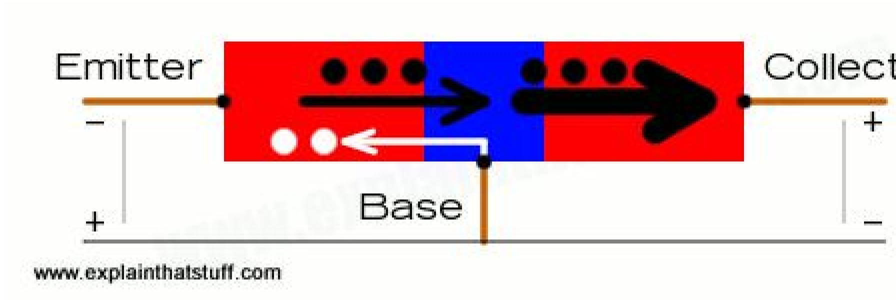


- Transistors
  - Neurons are the building blocks of our brain, essentially yes/no



switches, yes or no to electricity flowing through the brain

- Transistors are the same, like the diode, but with 3 layers
- EX - NPN transistors (n type, in the middle p type, other end n type)
- Can amplify current, and act as an on/off switch to current at the same time
- Emitter,Collector, base are wires attached to transistor
- If add small positive voltage base, negative voltage emitter, positive collector



  - Regularly, p holes make the p type a barrier, no electrons can flow through
  - When small positive charge added, more holes created in p type, negative in emitter causes more free electrons in n type, the huge difference causes LARGE flow of electrons from n to p type, and the positive hole in collector makes flow continue all the way though - AMPLIFIES current
  - If no positive voltage on base, holes never become abundant enough, no current flow, acts like a switch

○

-