**Lost In Space**

**Day 1: Getting the board going**

* Uses HERO Raspberry Pi circuit board
* Will be using ARDUINO IDE for coding and communicating with the HERO board
  + Arduino is closest to C++
* PIN = built in part of the board such as LED\_BUILTIN or the blinking light with L next to it
* digitalWrite function gives direction to a certain pin
  + Syntax would be digitalWrite(pin, action) in our case for example 12, HIGH would be turn pin 12 on as HIGH is equivalent to send 5V to that pin

**Day 2: Turning on the lights**

* White board with all the holes in it is called a BREADBOARD and its purpose is to allow one to assemble a circuit board with its various attachments like lights
* Circuit – path or multiple paths in which electricity can flow through in a controlled manner
  + For it to function, a circuit must be completed
* Voltage to electricity is similar to water pressure in the analogy
  + Voltage can be provided by things like batteries
  + The more pressure the more will flow through, however, this might be different depending on the width of what the voltage is flowing through. High pressure flows less in a smaller passage (pipe in the water analogy) so you would actually get more power from lower voltage in this case
* Just like certain pipes have some resistance to the water flowing through, some resistors have certain amount of resistance for the voltage
* Low value resistor is similar to wide pipe and high value resistor is narrow pipe
* To figure out how much is the right amount to flow through a resistor, we calculate Current
* Current = voltage/resistance
* EX – the pin with 5 volts using a 220 ohm resistor will be able to handle is 5/220 - .023 AMPS
  + The amps are what we would refer to is needed to power a specific item like a light bulb
  + So if our 5 volt pin uses the 220 ohm resistor to make .023 amps, this will be sufficient to power our light bulb
* LED – light emitting diode converts electricity to light rather than heating up
* Using well-made code, we can actually use the HERO board to control when, where, and how much electricity to apply depending on our need
* To connect circuits together, we use wires. And when we want to connect circuits with wires permanently, we stick them together with an action called soldering which uses high heat to stick the metal together
* BREADBOARD allows us to just slide the resistors into the holes rather than soldering and each row shares a current such as row 2 a,b,c,d,e and vertical rows + and –
* LEDs only work when electricity flows one way – in through the longer (+) leg and out the shorter leg (-)
* Round side is + and flat side is –
  + The longer leg would connect to the 5V and the short leg would connect to the 0V pin
* Usually longer is + and shorter is (-) and if they are trimmed, the kink is the + side

**Day 3: Controlling the lights**

* DIP Switch – dual inline plastic: switch system used to control electrical circuits
  + When the switch is set to OFF it cuts the circuit, when it is turned to ON it completes the circuit
* GND = Ground
  + Acts as the point where the voltage can return to its source at the end of its circuit journey
* VCC = voltage
* We want a resistor in our circuit with our switch to basically act as a traffic controller and make our voltage unambiguous
  + If there is no resistor, then our circuit would be connected to both 0 and 5 volts which would cause an unknown amount of voltage to flow through and will most likely be more than the LED can handle, and damage it also known as a short circuit
  + With the resistor, this makes the circuit path more explicit to have 0 volts with the switch off and 5 volts with it on
* The switch does not need to be part of the circuit directly. The code we made has the switch as an input and the LED as the output. So the switch talks to the HERO which talks to the LED circuit through the code we made. HERO senses if there is voltage through the switch, if yes then activate the circuit with the LED through the other pin
* Quick Conditional statement recap
  + IF – primary decision-making tool
  + ELSE IF – says to move on to the next condition
  + ELSE – runs if all else fails
  + = is used for assignment of variables
  + == compares two values and comes back as TRUE or FALSE

**Day 4: Applying control to multiple LED lights**

* Takeaways
  + Best practice is to set up a constant 5V accessible to all circuits as well as a ground outlet
  + Rather than connecting the LEDs directly to 5V and ground because that would make the light on all the time, we connect it to PINs that are controlled through our code
  + Flow is 5V flowing through the whole board, switch turned on sends flow through switch circuit, code detects is there is flow, if yes then it turns on the LED, if no 5V flow with the switch in OFF, then the code will keep the light off
  + Picture flowing water through canals and different doors opening and closing to let the water through. The code detects and directs the flow and does the action we want if depending if we have let water flow through that specific canal or not
  + Both the wiring of 5V to in though + and out through – has to be complete and then the code needs to match the position of the wiring and the pin to which it corresponds to

**Day 6: Fixing the Battery**

* Photo resistors – similar to solar panels but with light – when they are exposed to light, they allow more current to pass and then they get darker, less current passes with more resistance being applied
* LDR – light-dependent resistor is a type of photo resistor
  + With more light, there is less resistance and the flow of electrons becomes easier and vise versa with less light
  + These devices are made of high-resistance semi-conductor material
  + When light hits the device, the light powers the electrons and allows them to move
* Analog signals are continuous signals such as a level dimmer for a light switch, there are many positions within the range that one can place the switch and the light bulbs reacts in real time with how bright or dim it is depending on where the dimmer switch is positioned
  + Analog are more complicated and prone to degrading faster
  + Pro is that they can be much more precise
* Digital signals can only be in 1 of 2 states, binary, 1 or 0, yes or no, on or off
* A0 pin is specifically an analog pin that can be adjusted for a wide range of inputs, therefore we will be attaching our photo resistor to this
* Access Serial Monitor – what we will be using to send and receive and interpret data to and from our board and to and from the computer
* Function analogRead: allows HERO board to read analog signals
  + In this case we will be measuring voltage range between zero and 5V (1023)
* Read – taking data in
  + Sensors/inputs
* Write – write data out
  + Actuators or outputs – LEDs, motors etc.
* Normal circuit but the blue wire is placed at the high voltage end of the resistor and then connected to our analog pin so that in can act as an input and tell us how many volts are traveling through the high end. It is not connected to the low voltage part because that is ground which will be 0.
* The code today is using the BUILT-IN LED ON THE HERO board, not an external one like what we used the switches for
* LED will flash faster, the more light the resistor senses
* This data for how much light we are sensing will be sent to the serial monitor below our code

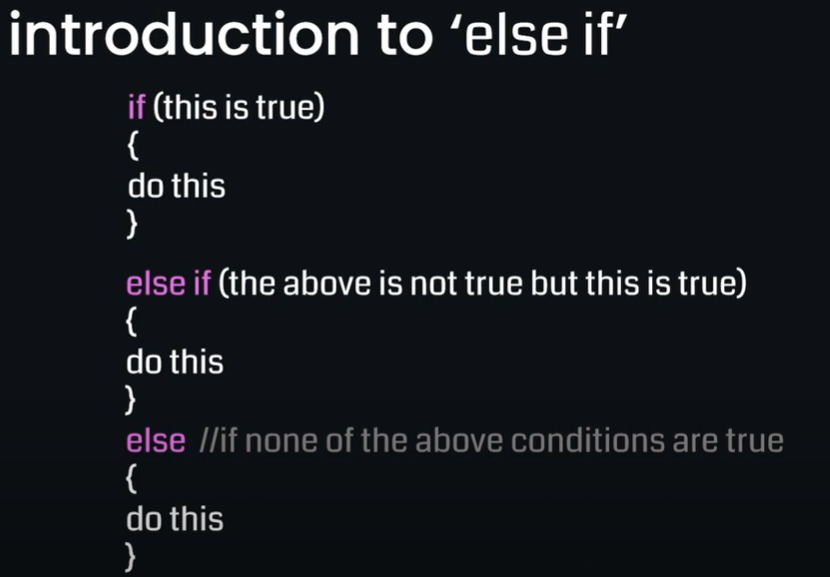
**Day 7: Charging the battery simulator**

* Bit – smallest unit of data that is either 1 or 0 binary
* Integers are usually 16 bits
* Floats are decimal numbers worth about 32 bits
  + Double is 64 bits and can hold more decimal places
* Void – used to define functions that don’t give anything back

**Day 8: Multi-Colored Lights**

* RGB LED – Red Green Blue, Light-emitting diode
  + Allows us to produce any color based on the strength of the color output
* PWM – pulse with modulation acts as an analog output(multiple levels of brightness) using digital inputs(on/off)
  + This technique is what allows us to make so many different colors with only 3 colors because it controls the brightness and time a color is on or off
* Each color can range from 0 – 255 brightness level

**Day 9: Battery Light Indicator**

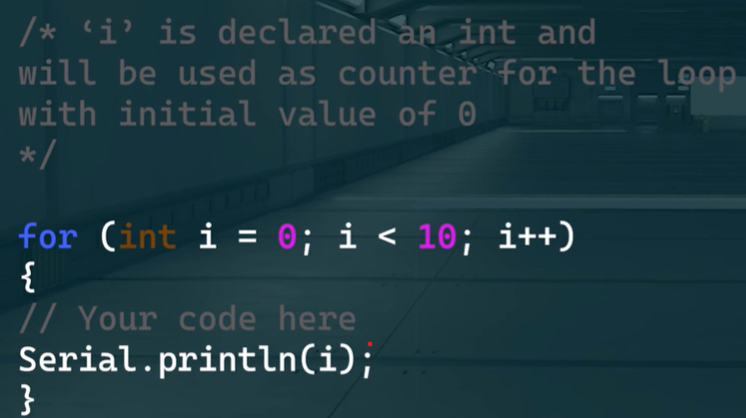
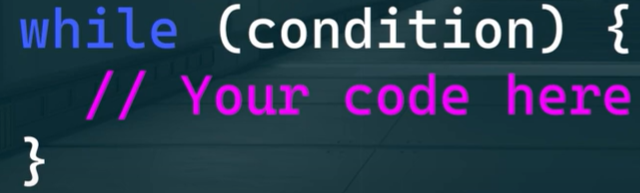
* Combining the photo resistor, we loop through the level of light coming in, measure power level and % and then apply the correct color
* Green for good, yellow for mid, red for needs charging
* Essentially we are going to have the color of the LED show what we see in the monitor instead
* Logical operators: && for AND and || for OR
* Else if – conditional statement that will ONLY run if the statement above it, is not true AND the else if statement is true
* This is different from just ‘else’ which will just always execute if the above is not true
* 

**Day 11: Control Panel**

* 4X4 control panel – basic concept is when each button is pushed, it closes a circuit that is detected and utilized to execute some task or operation.
  + The one we are using now is a simple version of our keypad – usually used for security doors, key pads etc.
* This key pad is known as an input device
* New library we will be using for coding will help us determine which keys are pressed, when they are pressed, and multi-key detection
* Arrays – data type that allows us to store multiple values in one variable
  + Data types must all be the same
* 2D array can be pictured like a grid rather than one line
  + Instead of item 2, it can be item at position 3,4 (row, column)



**Day 12: Controlling the Speakers**

* Passive Buzzer – sound frequency generator, like a mini speaker that we can control
  + Depending on the amount of charge sent to the speaker, the speaker vibrates with from the electricity sent to it and then makes sound
* **THE LOOPS WE HAVE BEEN USING SO FAR HAVE BEEN FUNCTIONS TO RUN OUR CODE, FOR LOOPS AND WHILE LOOPS ARE DIFFERENT**
* For Loop
  + Executes a block of code a certain number of times
  + Basic syntax
  + 
  + ‘i’ represents our counter starting at 0 and runs our intended code until i is no longer less than 10, increasing by +1, the code will then move to the next line
* While Loop
  + Does not specify a certain amount of times code will run
  + As long as a certain condition is TRUE the code will run again and again until that specified condition is no longer TRUE
  + 

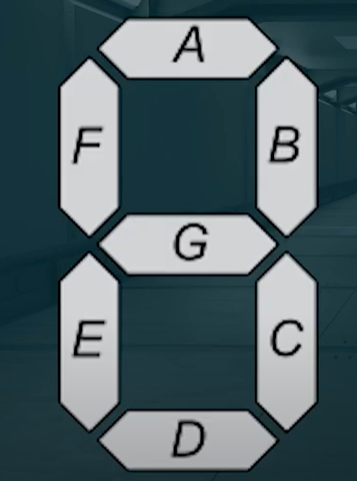
**Day 13: Setting a Pin**

* Bool UNLOCK function returns TRUE if the statement is true and FALSE is it is not
  + BOOL just means TRUE or FALSE
* Press # to enter system and \* to change password

**Day 14: Advanced Security Panel**

* Make the RGB glow green when unlocked or a correct password has been entered
* Make it glow Red when it is the wrong password and have the buzzer emit a harsh sound
* Make a neutral sound when entering the pin

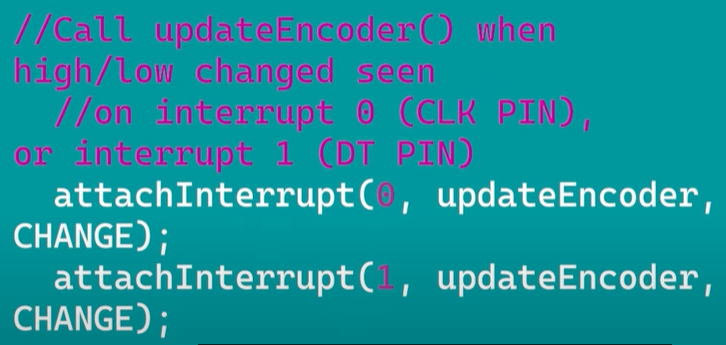
**Day 16: New Display: Create a clock, decrypt the code, limit rate of ascent**

* Creating the clock
  + Digital clocks are usually made up of 7 segment displays make up of 7 LED segments A-G
  + 
  + Display wiring
    - GND = ground
    - VCC = voltage coordinator or our power
    - CLK = clock that converts binary to light up what is needed for 0-9
    - DIO = digital input/output – what reads the binary code to know what to display
  + Set Segments function – like the puppet master for our 7 segment display and tells which segment, A-G, to turn on or leave off through an input of binary 1=ON, 0=OFF
  + For example – light up or 1 for A,B,C,E,F,G to display the letter A
  + 

**Day 17: Decrypting**

* Encryption – used to hide information
* Key – what you would need to solve the encryption such as Caeser cypher
  + Key would be shifting one letter over such as A becomes B, B becomes C etc.
  + This can be shifting 1-25 placements over

**Day 18: Ascending Sensor Using Rotary Encoder**

* Rotary encoder – sends direction of rotation and the amount of turn plus on or off
  + Every click is counted and every turn is noted
* Absolute encoder – unique value for every position
* Relative encoder – turns endlessly and tells us how far and where it has turn RELATIVE TO ITS STARTING POSITION but does not give a specific location **this is what we are using**
* Two key factors we want to know using this device – direction of rotation and rate of rotation
* CLK high is clockwise
* DT high is counter clockwise
* **Interrupt feature** - allows us to interpret the normal flow of the code to determine direction
  + This feature allows us to interrupt our normal code flow to pay attention and react accordingly to a specified important happening such as the change in direction
  + Practical analogy is like setting an alarm clock at 7am instead of checking the clock every hour
* 

**Day 19: Adding to our ascent**

* No additional concepts – adding a buzzer to our ascent reader

**Day 21: Monochrome LED Display**

* Organic LED Screen – uses many LED lights so we don’t need to use one LED per pin circuit
* GND – ground
* VCC – power
* SDA – serial data
  + Sends information using I2C protocall
* SCL – serial clock
  + Keeps the data from going out of sync like a data transmitting traffic light
* Protocols – a set of rules of procedures for transmitting data so that they are all “speaking the same language”
* Two common protocols are:
  + I2C – inter-integrated circuit (pronounced i-squared c)
  + SPI – serial peripheral interface
  + Both are used for communication between devices
* drawStr function – allows us to display words or symbols on our device

**Day 22: Display Panel Extended Details – using Icons, pictures, and graphics**

* Do While Loop – a while loop that is guaranteed to execute at least once
* Switch function – takes in specified and multiple input information and based on the set criteria, will execute a specific function
  + Case statements will act as scenarios. If Case or situation 1 do x, switch if case 2 then do y
* Bitwise Shift - <<or>>
  + Means shifting the binary bits over by one position
  + Ex 0101 >> 1 -> 0010 which is the equivalent to dividing by 2 in this case
* Bitwise AND
  + Checks two bits from two different numbers at the same position
  + If they are both 1, the result is 1, otherwise 0

**Day 23: Launch System**

* Macros
  + Begins with #define, this is a way of creating a shortcut or alias
  + We set a keyword equal to a specific value or string of code rather than typing it out
  + Similar to setting a variable or constant but instead of just a value, it is a shortcut
  + Useful when we have to type a certain chunk over and over

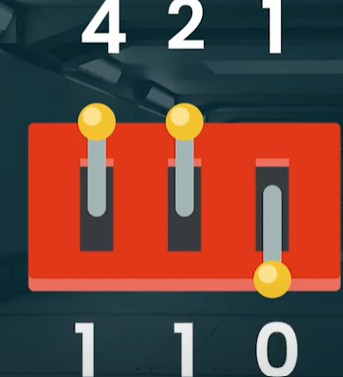
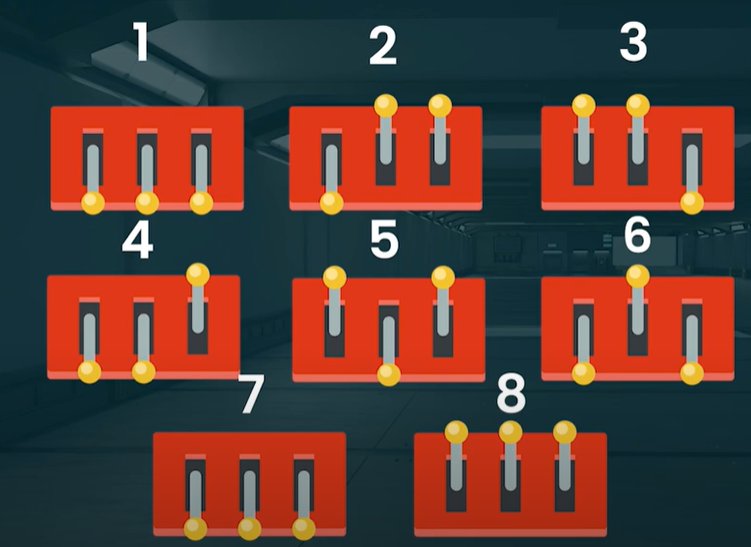
**Day 24: Launch System 2**

* Combining OLED display, 7 segment display, buzzer, and dip switch into our launch system

**Day 26: Fun with Bitmaps**

* Our OLED display measures 128 x 64 pixels (or individual small LED lights) that are capable of being turn on or off to display our specified image
* A bitmap maps out our physical OLED display into code and dictates which LEDs should light up and which ones should stay turned off
* These are specified as hexadecimals instead of binary
  + 0x00 = OFF
  + 0xFF = ON (bull brightness)
  + Anything else is somewhere in between full brightness and off

**Day 27: Autopilot**

* Counting in binary
  + OFF = 0
  + ON = 1
  + Our 3 dipswitches represent 000 (when all off) or 111 (when all on) in binary
* Our 3 switches represent 1 2 4 when counting
* 
* This translates to 4+2+0 = 6
* Different on/off combinations represent different numbers
* All possible cases:
* 

**Day 28: Landing Gear**

* Incorporating the 4x4 keypad to for additional checks for our landing gear
* Need to create logic that determines the state of our dipswitches and if all correct, proceed to the next step

**Day 29: Conclusion**

* Adding rotary encoder and logic to tell us our speed and distance from the landing bay in order to touch down safely