UNIVERSITY TRANSPORTATION CENTER FOR
RAILWAY SAFETY
Award Number: 69A3552348340
Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS)
Award Number: 2112650

High School Curriculum

Week Contents

1. **Intro to Python**
2. **Sensor Programming**
3. **Driving the Rover**
4. **Hardware Applications**
5. **Final Challenge**

Day 1

- Ice Breaker (Logic Labyrinth)
- Engineering Design Process
- Algorithm Design Process
- Intro to Variables
- Understanding Logic
- Active Sign Creation

Ice Breaker: Logic Labyrinth (Thinking like a computer)

- Make teams of 4-5 students, each team will try to complete a simple task using the functions provided.
- Computers today can perform very complex tasks such as driving cars autonomously, or give you the answers to your homework 😅

Ice Breaker (Continue)

Allowed commands for your “robot”:

- Move forward (continuously)
- Move Back (continuously)
- Take # steps forward
- Take # steps back
- Turn right # degrees
- Turn left # degrees
- Stop
- Wait # seconds

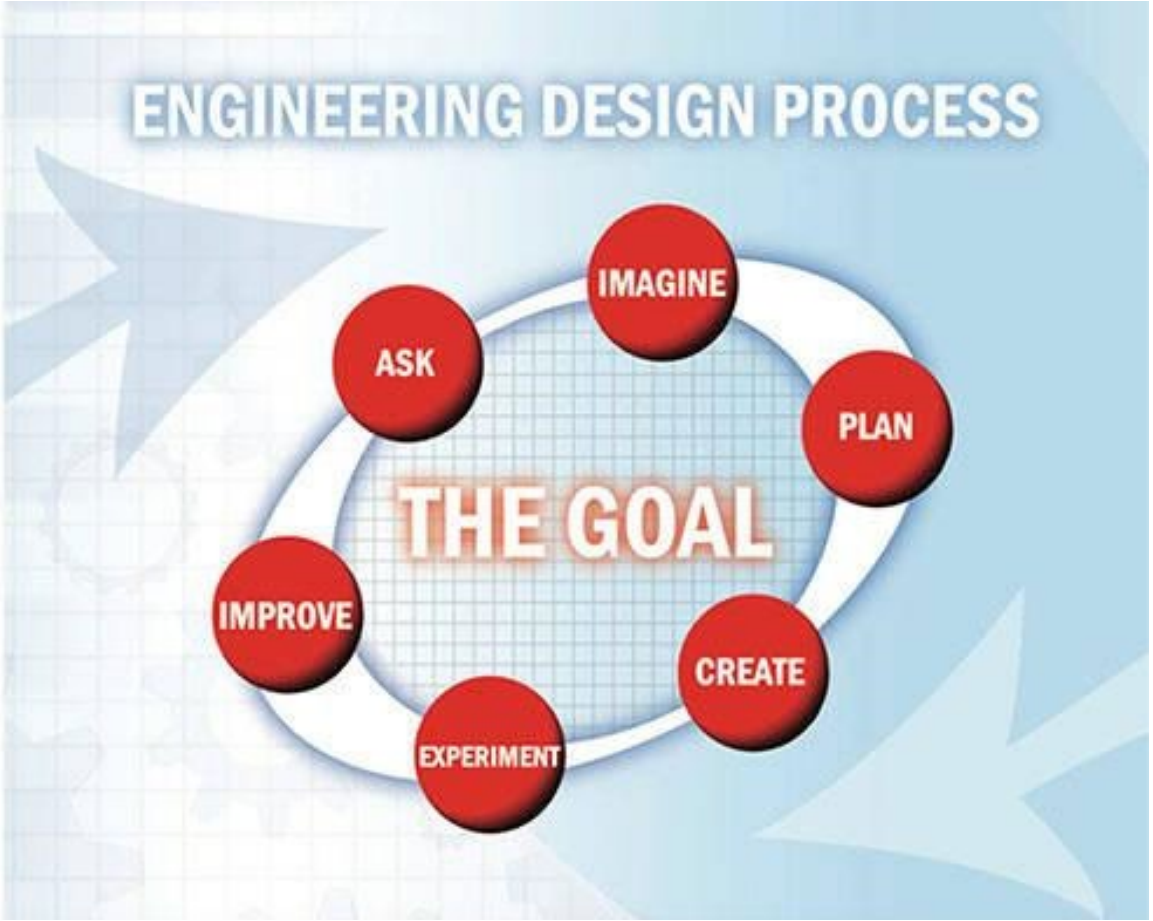
Ice Breaker (Continue)

- In reality computers are very dumb, they can only do simple tasks like adding, subtracting, reading and writing data in memory. That is it! For the computer to complete the complex tasks AI can do, we need a bunch of operations that all use those 4 actions.
- How do we create complex things like moving a robot?
    - Answer: Abstractions/Functions

UNIVERSITY TRANSPORTATION CENTER FOR
RAILWAY SAFETY
Award Number: 69A3552348340
Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS)
Award Number: 2112650

Engineering Design Cycle (Process)

Engineering Design Cycle (Process)



What is Engineering?

- Engineering is the process of creating technology
  - Can you name some examples of technology we use every day?

ASK

- Students identify the problem, requirements that must be met, and constraints that must be considered.

Imagine

- Students brainstorm solutions and research ideas. They also identify what others have done.

Plan

- Students choose two to three of the best ideas from their brainstormed list and sketch possible designs, ultimately choosing a single design to prototype.

Create

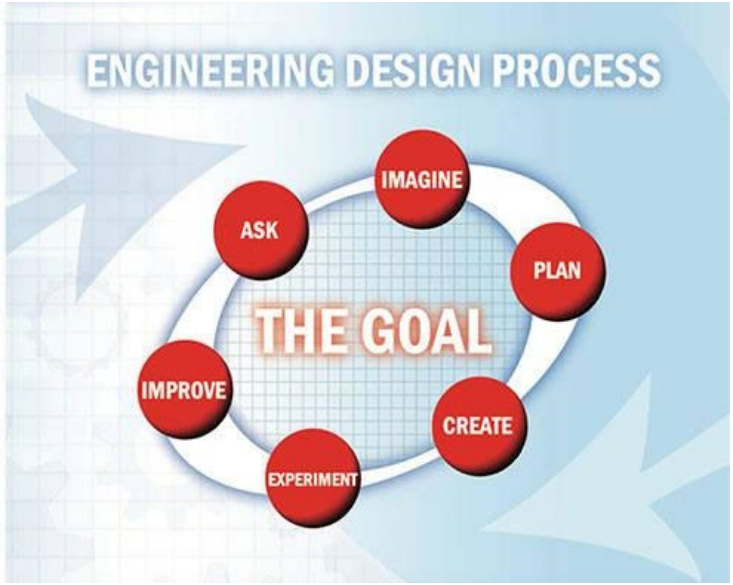
- Students build a working model, or prototype, that aligns with design requirements and that is within design constraints.

Experiment

- Students evaluate the solution through testing; they collect and analyze data; they summarize strengths and weaknesses of their design that were revealed during testing.

Improve

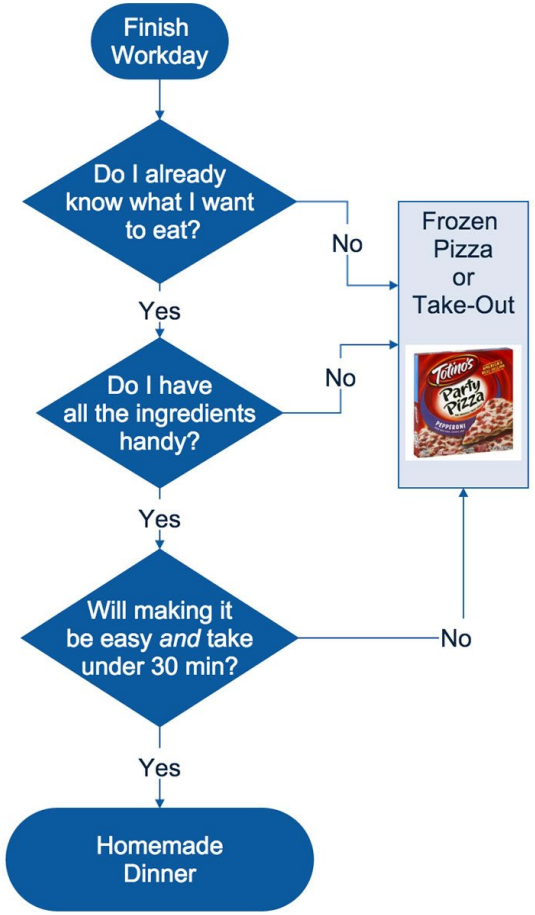
- Based on the results of their tests, students make improvements on their design. They also identify changes they will make and justify their revisions.



Algorithms and Pseudocode

Algorithms

- An algorithm is a set of instructions for solving a problem or accomplishing a task. One common example of an algorithm is a recipe, which consists of specific instructions for preparing a dish or meal.



Algorithm Example

Pseudocode

- Pseudocode looks almost like real code, but it does not follow any specific syntax for any language, instead it is meant to be written almost in plain English to break down steps in a set of instructions. E.g.

Pseudocode for deciding what to eat:

```
when it is the end of the day
        Then Do I know what to eat?
        if yes
                then Do I have all ingredients?
                If yes
                        then Is it going to be complete in < 30 min?
                        If yes
                                Then Start cooking script
                        else Order take out
        else Order take out
Else continue working
```

What do you know about programming?

What is Python?

What is Python?

Coding:

go to further.pi-top.com

Join: 2025 UTCRS Summer Camps
Code: E4Z4MJDRHM

Further Course: Intro to Variables

1. What is a Variable?
2. Naming Variables
3. Rules for Variables

What is a Variable?

- Variables are declarations or names written close to the top of a script. They are used to store and manipulate pieces of information. Variables allow us to assign values, declare what ports or pins are used, use calculations, etc.

Sample Program with Variables

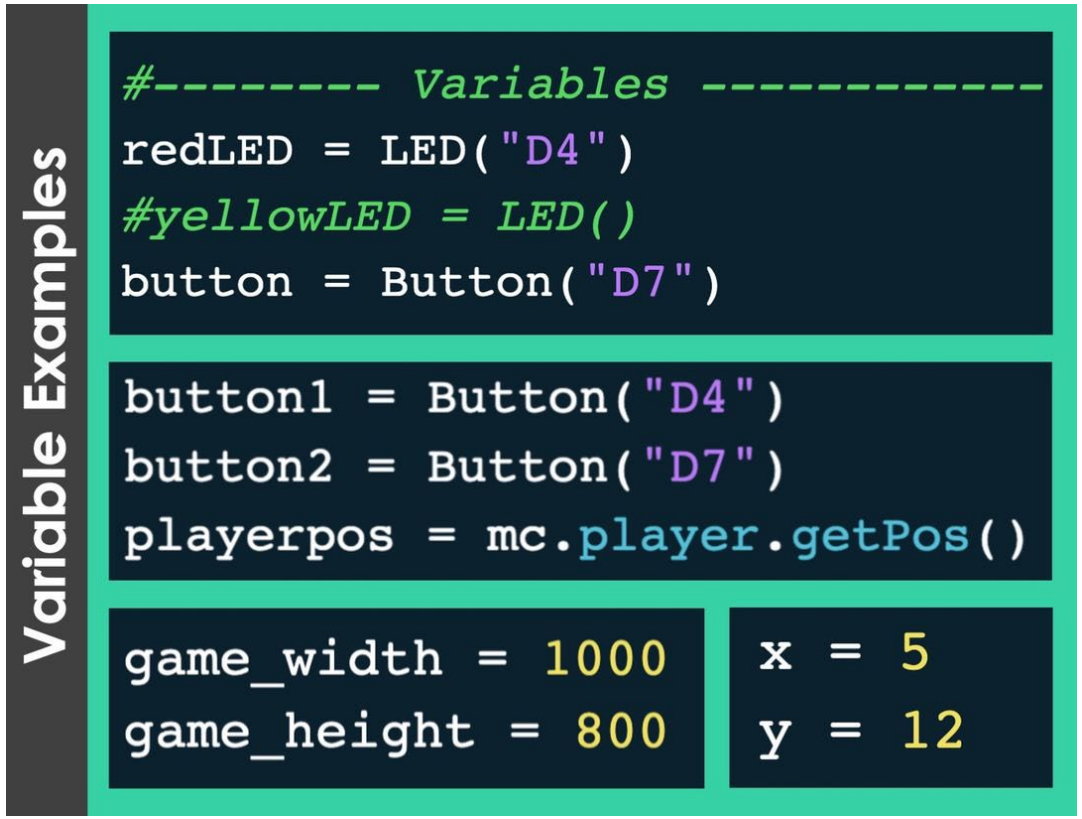
```
1  from signal import pause
2
3  #------ Variables -------
4  yellow_led = LED("D0")
5  red_led = LED("D1")
6  green_led = LED("D2")
7  button = Button("D3")
8
9  #Section 2; defines a 'red light' function
10 def red_light():
11     yellow_led_.on()
12     red_led.off()
13     green_led.off()
14     print("WAIT")
15     sleep(3)
16     yellow_led_.off()
17     red_led.on()
18     print("STOP")
19
20 # Section 3 defines a 'green light' function
21 def green_light():
22     yellow_led_.off()
23     red_led.off()
```

Variables in sample code

Naming Variables

- To create a variable, it must be given a name and a value (or some type of "defining" information).

  For instance, speed = 120.
- The variable is named speed, with an integer value of 120.
- It is important to use a good name that is easy to remember and one that isn't closely related to other variable names. If we use something like walk1 and walk2 it may be hard to remember the difference between the type of walking. But if we used walk_slow and walk_fast it quickly tells us the difference and is easy to remember later in the code.
- See the image on the right for other variable name examples.



Rules for Variables

- Variables can contain words, letters, underscores, and numbers but it is best to use a word or letter to start it off.
- Python code is case sensitive so if capital letters are used, they must always be written the same way throughout the script. For instance, Walk and walk are totally different variable names. Symbols: - / # @ are not to be used when naming variables. Avoid using python commands, like import or print.

```
 8 #------- Variables --------------------
 9 miniscreen = Miniscreen()
10 rocket = Image.open("/usr/lib/python3/dist-
   packages/pitop/miniscreen/images/rocket.gif")
11 select_button = miniscreen.select_button
12 red_led = LED("D3")
```

Further Course: Understanding Logic

- Python Principles: if Statements
- Python Principles: Loops
- Tutorial: Miniscreen

Python Principles: if Statements

- What are if statements?
  - Conditionals, such as if and else, are used for decision making operations and in branching.

```
1 if button.is_pressed:
2     print("pressed")
```

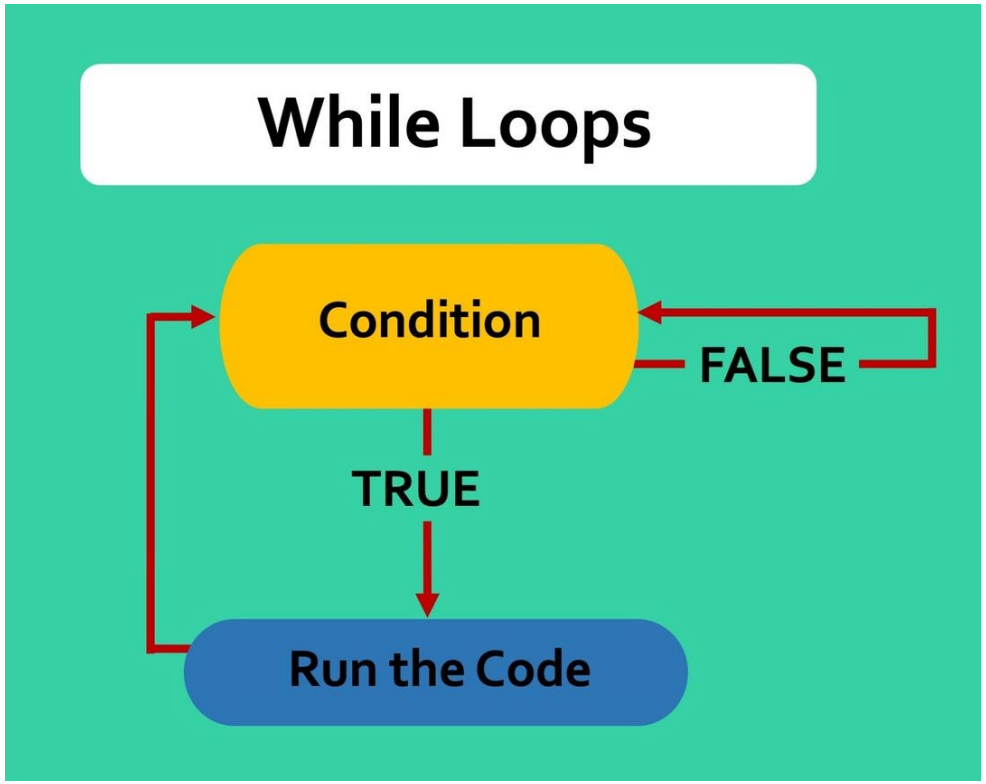
  - If statements are used when programmers want to give the computer a condition to evaluate before activating and to activate any code.

Python Principles: Loops

- What are Loops?
  - Loops are used when programmers want to execute repetitive code when a program runs. A loop will run the same set of code over and over again when written in a program.
  - Think of this as a place where you would put the ongoing actions that a game character would need to have. If they should take more than one step when a button is pressed, then it is most convenient when in a loop.

Python Principles: Loops

- While Loops (Infinite Loops)
  - What happens when a programmer isn't sure how many times to run the code? Or if they want to make it run repeatedly until they press stop? In this type of situation, a while loop is used. While loops are loops that will run if and when a condition is met. It is a type of Boolean expression - meaning the condition can only be true or false.




```
1 while True:
2     x += 1
3     print(x)
4     sleep(0.5)
```

Python Principles: Loops

- for loops
  - When the programmer wants a section of code to happen a set number of times, the for loop is used. The amount of times is known as an iteration.
  - For loops iterate over a given sequence, or run the same block of code over and over until the sequence ends. They basically will run a body of code for a specified amount of time or only as long as the code that is written needs to run. Complete exercise 6 below to practice this concept.

```
15 #Rocket Function ----------------------
16 def rocket_launch():
17     for text in ["5", "4", "3", "2", "1"]:
18         miniscreen.display_multiline_text(text)
19         red_led.on()
20         sleep(0.5)
21         red_led.off()
22         sleep(0.5)
23
```

Python Principles: Loops

- Nested Loops
  - Python allows us to use a loop inside of another loop. Yes, there may be times that for loops need to be written inside of other loops. While loops or infinite loops can also contain for loops, depending on what the programmer wants to happen. This is easy to do, but always make sure that the lines are indented properly.

```
1 for x in range(5):
2     for y in range(2):
3         print(x,y)
```

Tutorial: Mini Screen

- The Pitop comes included with a Mini screen we can use to display messages or get information the robot is processing.
- In this tutorial you will learn how to use it to display simple messages, and construct a counting game.

Further Course: Programming Hardware

- Traffic Lights
- Python Principles: Functions (Optional)

Traffic Lights

- Belisha Beacons
  - marks pedestrian crossings, such as zebra crossings, in the United Kingdom and other countries

Traffic Lights

- Traffic lights normally consist of three signals, transmitting meaningful information to road users through colors and symbols, including arrows and bicycles.

Traffic Lights

- What other traffic lights do you know?
- Can you recreate it?

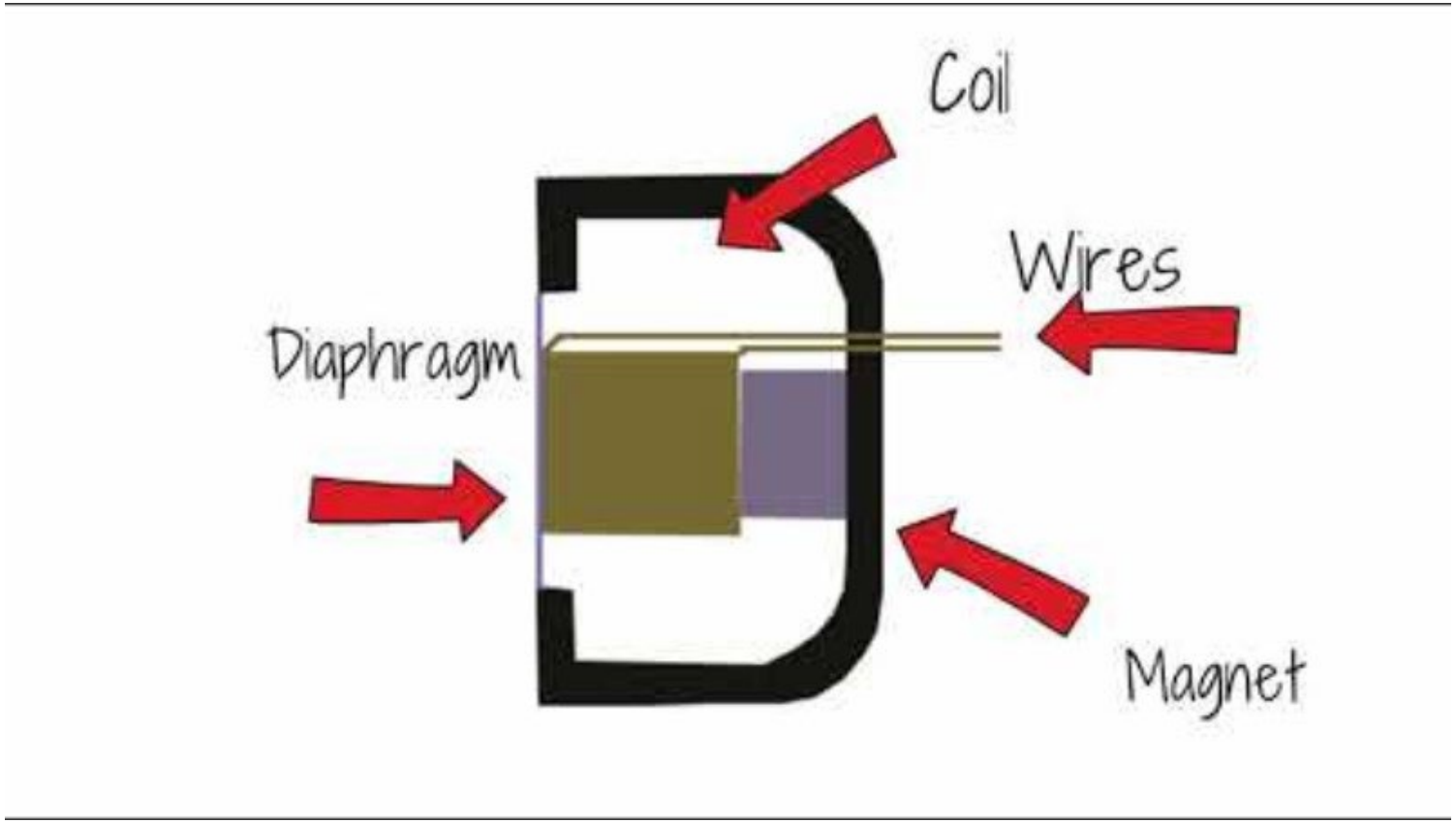
Day 2

1. Understanding the Sensors
2. Active Sign Creation
3. Rover Build and Navigation

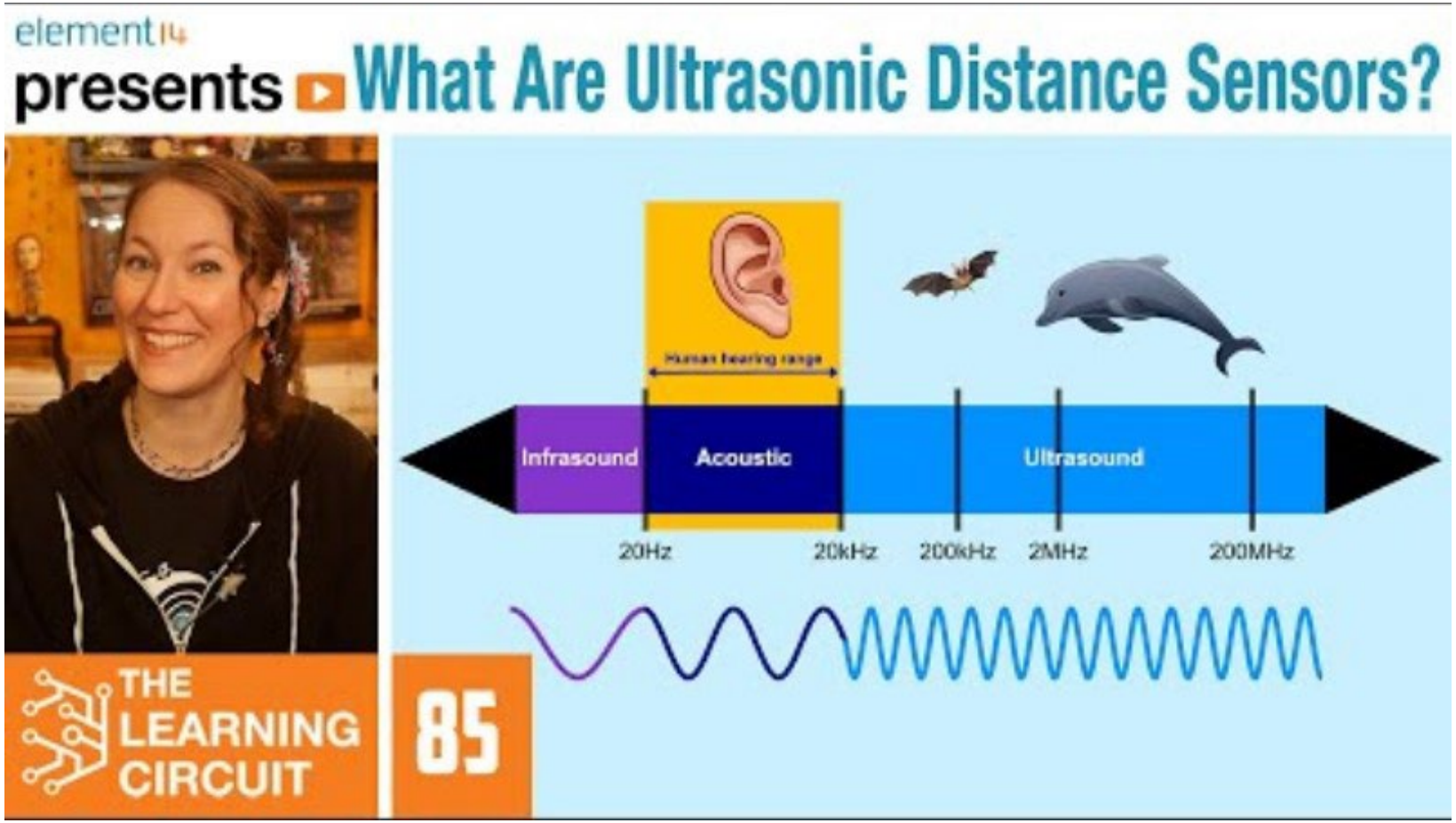
Further Course: Understanding the Sensors

- Tutorial: Sound Sensor
- Tutorial: Ultrasonic Sensor
- Tutorial: Light Sensor

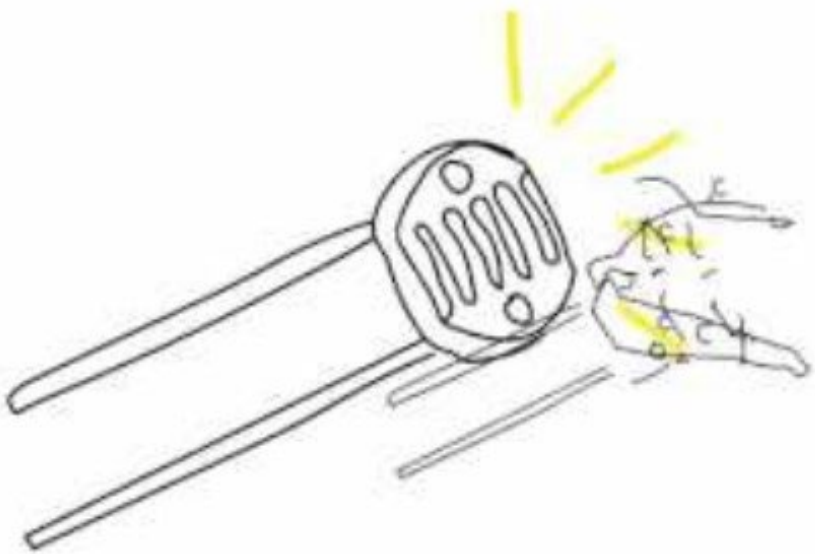
Tutorial: Sound Sensor



Tutorial: Ultrasonic Sensor



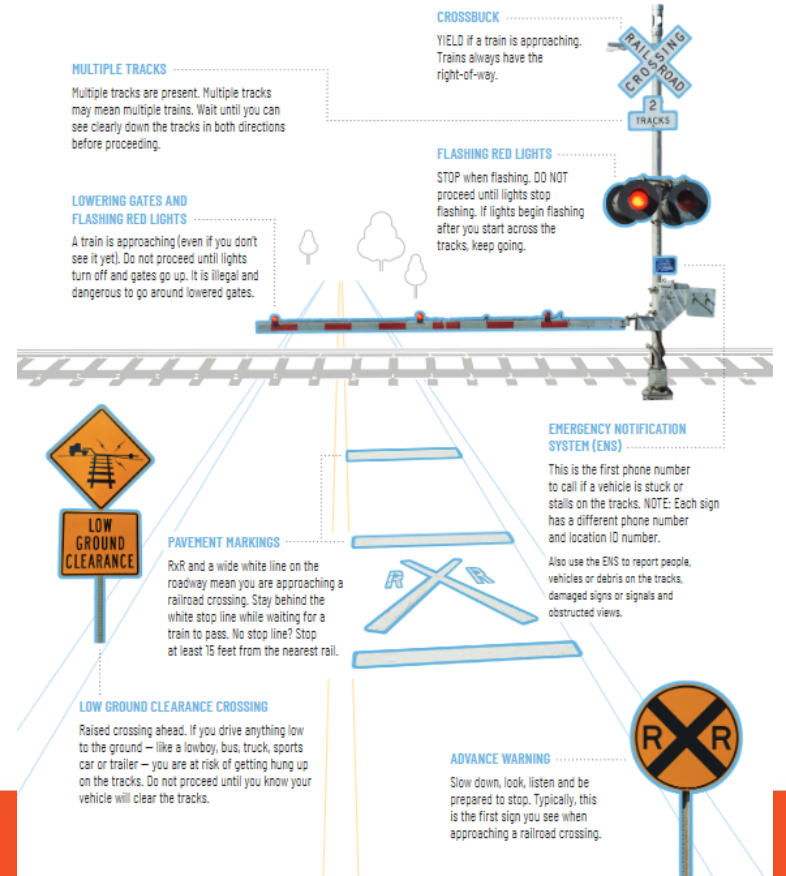
Tutorial: Light Sensor



what happens within a semiconductor material to regulate the flow of electricity.

Further Challenge: Active Sign Creation

Why are railway crossing signs important?

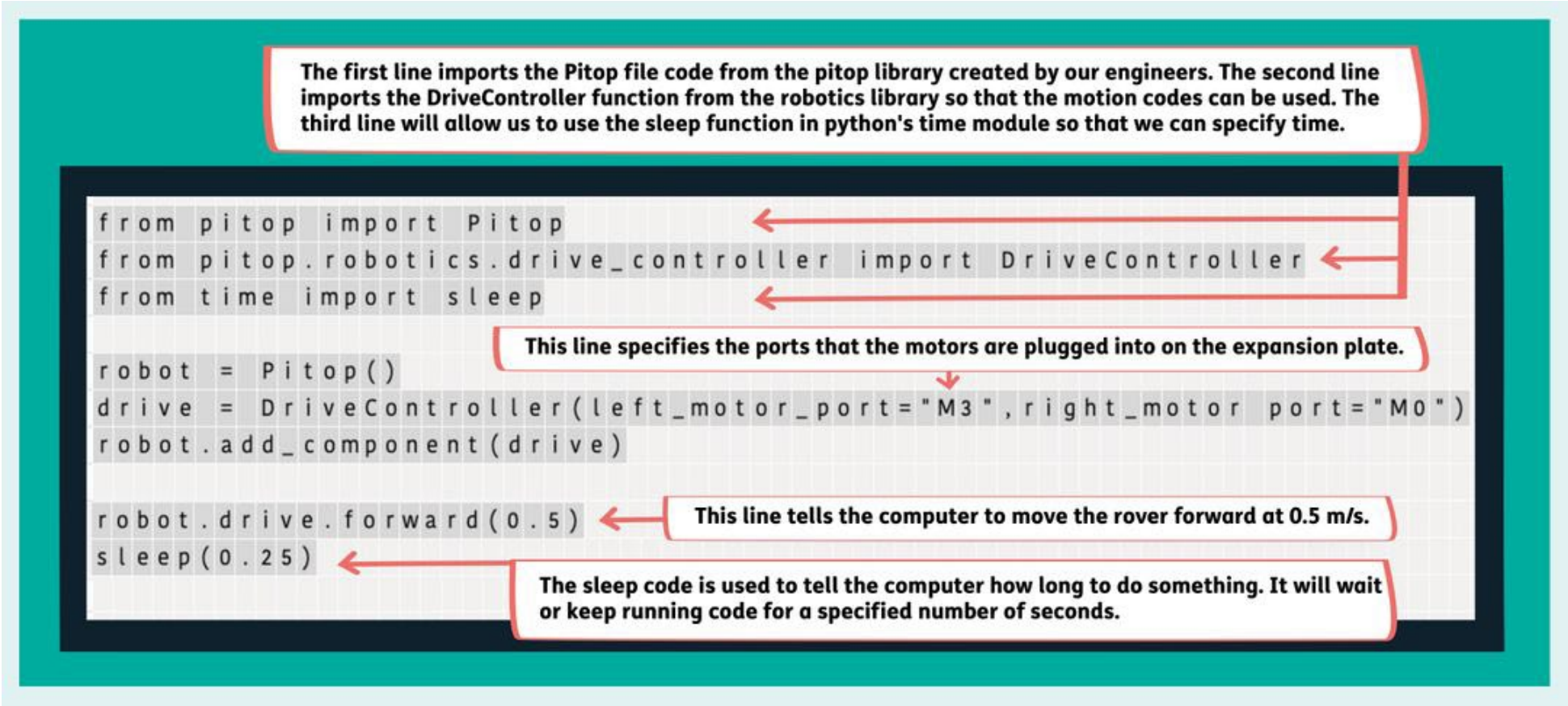


Further Course: Rover Build and Navigation

- Robotics 101 [1]: The Build
- Robotics 101 [2]: Navigation
- Reverse Parking Sensor

Robotics 101 [1]: The Build

Robotics 101 [2]: Navigation

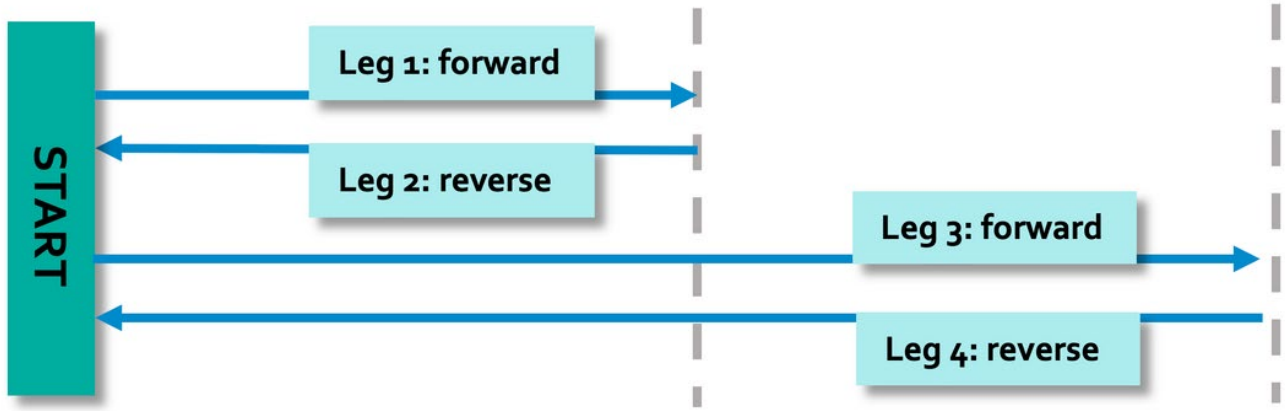


Navigation Challenges

- Change 1forward.py so the robot travels double the distance
- Change 1forward.py again to testing different distances, how do you move exactly 1 feet?
- Change 2reverse.py so the robot moves forward first then reverses to its original position.

Navigation Challenges

- Change 2reverse.py again so the robot moves forward 2 feet then reverses to its original position, then again but this time 4 feet. All in 1 program execution.



Reverse Parking Sensor

- Create your own reverse parking sensor using sensor_control.py.
  - This code incorporates a while True loop. The beginning of the loop is the same as it was in a previous section, but now there is a conditional after the statements to read the distance from ultrasonic sensor and print the reading.
  - This conditional does the following: if something is within 30 cm of the sensor, it turns on the red LEDs and the buzzer.
  - If neither of these conditions are true (else), it turns off all the components except for the green LED.

```
11 while True:
12     # this line reads the distance from the sensor
13     distance = ultrasonic_sensor.distance
14     # this line displays the distance on the computer screen
15     print(distance)
16     sleep(0.1)
17     # this section turns the LEDs and buzzer on or off depending on the distance
18     if distance < 30:
19         red_led.on()
20         yellow_led.off()
21         green_led.off()
22         buzzer.on()
23
24     else:
25         red_led.off()
26         yellow_led.off()
27         green_led.on()
28         buzzer.off()
```

Navigation Challenges

Day 3

1. Rover Speed and Acceleration
2. Rover Race
3. Rover Vision

Further Challenge: Rover Speed and Acceleration

1. You will use main.py to run your robot on 2 experiments the first will run the robot for 5 seconds at a constant speed, and the second will slowly accelerate from 0 to 100% speed in 5 seconds, after the experiment you will record the distance for each, and get the speed of your robot based on distance travelled.

$$speed = \frac{distance}{time}$$

Further Challenge: Rover Speed and Acceleration

1. Revolutions per Minute (RPM) gives use the number of times the motor will spin in 1 minute, but it does not tell us the speed of a robot exactly. For that we need to know the diameter of the wheel we are turning, so 1 motor spin will equal the circumference of the wheel.

$$Speed = \left(\frac{RPM}{60}\right) * \pi * diameter$$

Further Challenge: Rover Speed and Acceleration

1. What is the top speed of the Rover?
2. How did the actual graphs compare to your hypotheses? Were there any surprises?
3. What does a straight line in the distance-time graph for constant velocity indicate about the motion?
4. How does the shape of the velocity-time graph for acceleration differ from that of constant velocity?
5. What factors could affect the accuracy of the data collected in these experiments?
6. The rover weighs 3.18 kilograms and Perseverance, the Mars rover, weighs about 45 kilograms. How do you think mass affects speed?
7. NASA's Mars Rover drives with a top speed of 0.16 kilometers per hour (0.1 miles per hour). Calculate the speed of your rover and compare it to that of the Perseverance rover.

Tutorial: USB Keyboard

- There are 2 ways we can interact with the Pitop[4] using a keyboard
  - Polling (direction_arrows.py)
  - Event Handling (control_LED_buzzer.py)

Polling

- Polling means we wait for a loop to check things one by one, add too many things and the robot will appear to become unresponsive, but in reality, the robot was probably checking if up is pressed when you pressed a down in your keyboard and the input will look like it was dropped. Try pressing the key longer, now it responds!

```
15 while True:
16     if up.is_pressed:
17         print ("Up is pressed")
18         sleep(.5)
19     if down.is_pressed:
20         print ("Down is pressed")
21         sleep(.5)
22     if left.is_pressed:
23         print ("Left is pressed")
24         sleep(.5)
25     if right.is_pressed:
26         print ("Right is pressed")
27         sleep(.5)
```

Event Handling

- Unlike polling, events are handled on threads, what are those!? Think of them as little processes that will just listen to keys being press. With events you can immediately react to a key being pressed instead of relying on a loop being in the right position at the right time.

```
17 def red_on():
18     red_led.on()
19     print ("RED ON")
20
21 def red_off():
22     red_led.off()
23     print ("RED OFF")
24
25 def yellow_on():
26     yellow_led.on()
27     print ("YELLOW ON")
28
29 def yellow_off():
30     yellow_led.off()
31     print ("YELLOW OFF")
```

```
50 r.when_pressed = red_on
51 r.when_released = red_off
52 y.when_pressed = yellow_on
53 y.when_released = yellow_off
```

Tutorial: USB Keyboard

- How responsive is the robot using a loop? Did it get slower the more keys you added?
- How did your robot design affect your performance in the race?

Rover Race

- Can you create your own controller to drive on a race?
- Will you use polling or event handling?
- Will you want to go full speed (1) or less than that?

Further Course: Rover Vision

- Robotics 101 [7]: Rover Vision
- (Optional) Robotics 101 [8]: Path Finder

Robotics 101 [7]: Rover Vision

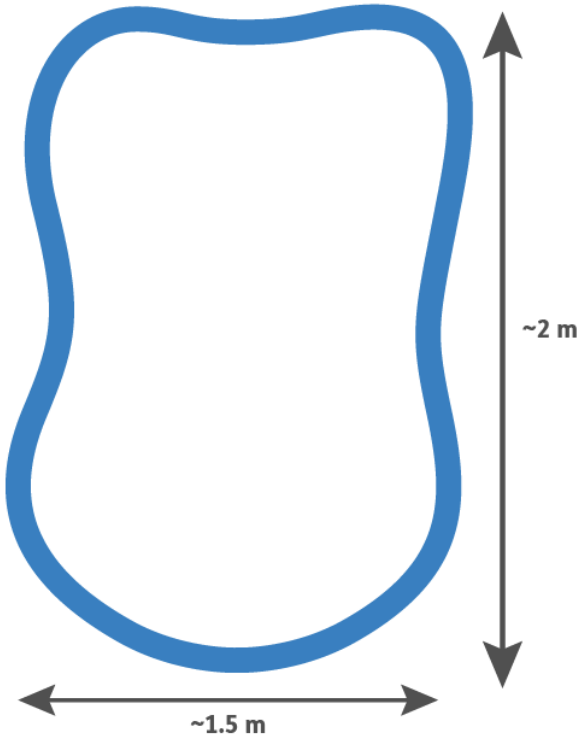
- Ensure you have the standard Alex Build and a USB camera for this lesson
- (Optional) Run the zero.py and follow the instructions on Further to calibrate the servos
- Explore the servos!

Robotics 101 [7]: Rover Vision

- What does the robot see?
  - Run camera5.py
  - Verify the camera has the correct orientation
- What can we do with the camera feed?
  - The camera component has a cool event handler called “on_frame” that runs on every frame it receives.

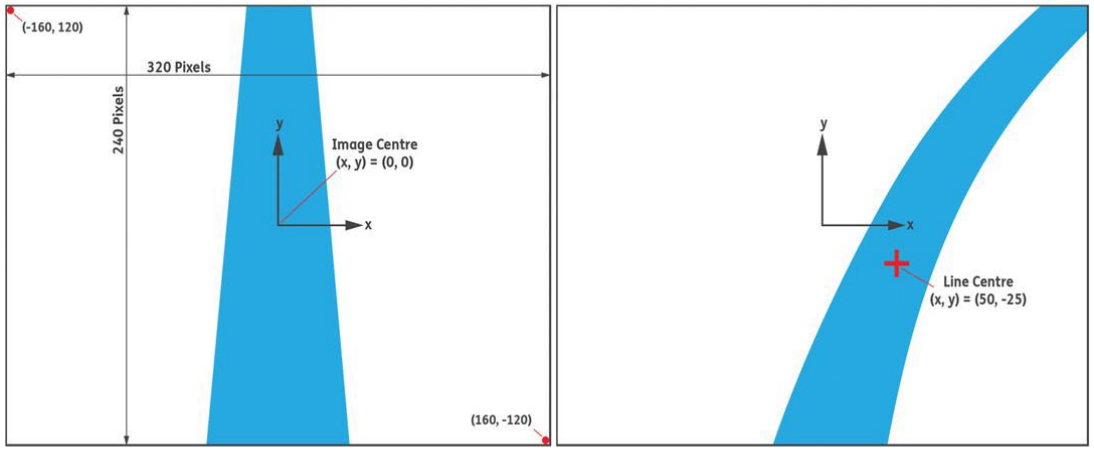
Robotics 101 [7]: Rover Vision

- If we want to build a line follower rover, we'll need a line to follow - grab the blue tape that was included in your Robotics Kit. Set up a line track on your floor based on the following:
- Keep the track simple with gentle curves, we will make a more advanced one later!
- You can make it whatever size you want, but approximately 1.5m x 2m should be big enough.



Robotics 101 [7]: Rover Vision

- How does the robot follow a line with only a camera?
  - The Pitop has a built-in color detector and can find the blue tape edges, with this we can find the middle section of the blue edges with some math.
  - The objective is to keep the center of the camera in the center of the tape.



Robotics 101 [7]: Rover Vision

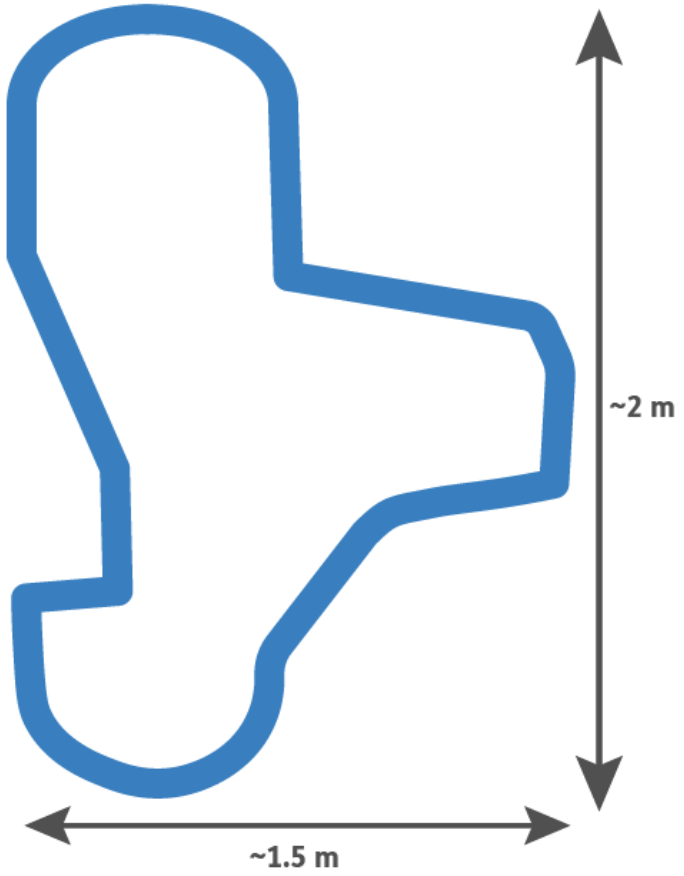
- Using 7line.py complete the code in 8LRM.py and build a line following robot!

Robotics 101 [7]: Rover Vision

- What is the fastest forward speed value you achieved?
- List and describe two real life examples where a line detection process is used. If you plan to use a search site to help you answer the question, be sure to reference the source(s).
- If the processes from this lesson were to be applied in real-world applications, what potential issues could arise for the user?
- What is the benefit of using functions in your code?

Robotics 101 [8]: Path Finder

- Build a better version of the line following robot from the previous version.
- Try to complete a track on the right



Day 4

- Collision Simulation
- Forklift Assembly Challenge

Further Challenge: Collision Simulation

- You are locomotive engineer driving at top speed when suddenly a truck tries to go around the railway crossing gates and gets stuck, you have no time to break and crash with the truck
- What happens? You've probably can guess the car is completely destroyed
- How far does the train drag the car along?

Further Challenge: Collision Simulation

- Consider a robot and a box involved in a collision. We want to calculate the kinetic energy before and after the collision and the work done by the robot on the box.
- What are the 3 laws of motion?
- What is Kinetic Energy?
- What is Force?
- What is Work?
- What is Power?

Newton's Laws of Motion

- 1st an object at rest will remain at rest until an external force is applied.
- 2nd force is equal to an object's mass times its acceleration. F = m*a
- 3rd every action has an equal an opposite reaction.

Kinetic Energy

- Energy of Motion
- Imagine the scenario of the robot rolling across the floor toward the cardboard box. This robot isn't just sitting still; it's moving—maybe slowly, maybe quickly. When it crashes into the box, something happens: the box might slide across the floor, tip over, or even get crushed, depending on how fast and heavy the robot is.
- The robot has something called kinetic energy. Kinetic energy is the energy an object possesses because of its motion. Whenever something is moving, it carries energy with it, and this energy can be transferred to other objects when they interact.

$$\text{Kinetic Energy (KE)} = \frac{1}{2} \times mass(m) \times velocity(v)^2$$

Understand Kinetic Energy

- Kinetic Energy depends on the object's mass (weight in kilograms(Kg)) and its current velocity (speed in a direction).
- Take a moment and reflect what would be the difference of a toy car crashing into a wall at 10 m/s (22.3 mph), versus a train crashing into a wall at 10 m/s. Which one would cause more damage? A toy car weight approx. 0.04 Kilograms (0.08 lb), and a train weights approx. 15,000 tons or 15000000 Kg

$$\text{Kinetic Energy (KE)} = \frac{1}{2} \times mass(m) \times velocity(v)^2$$

How do I calculate the forces in a car crash?

To calculate the impact force in a car crash, follow these simple steps:

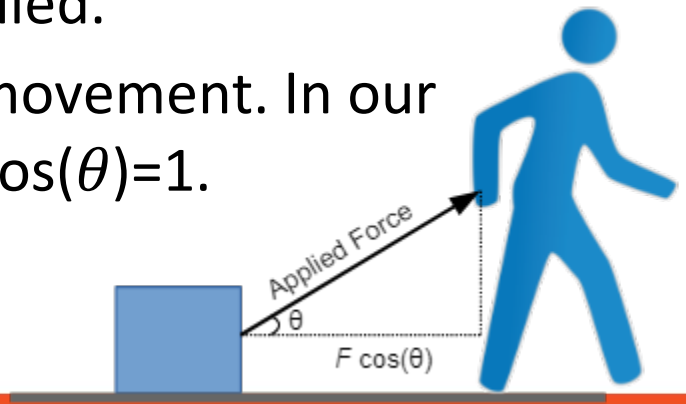
- Measure the velocity at the moment of the impact, **v**.
- Measure the mass of the subject of the collision, **m**.
- The stopping distance d in the formula: $\mathbf{F = mv^2/2d}$; or
- If you want to measure the g-forces, divide the result by **mg**, where $\mathbf{g = 9.81\ m/s^2}$.

Work

- In everyday language, "work" can mean many things, but in physics, it has a very specific definition:

$$\text{Work (W)} = \text{Force (F)} \times \text{Distance (d)} \times \cos(\theta)$$

- Force (F): The push or pull acting on an object. Equal to the product of mass and acceleration of the object.
- Distance (d): How far the object moves while the force is applied.
- Theta (θ): The angle between the force and the direction of movement. In our case, the force and movement are in the same direction, so $\cos(\theta)=1$.



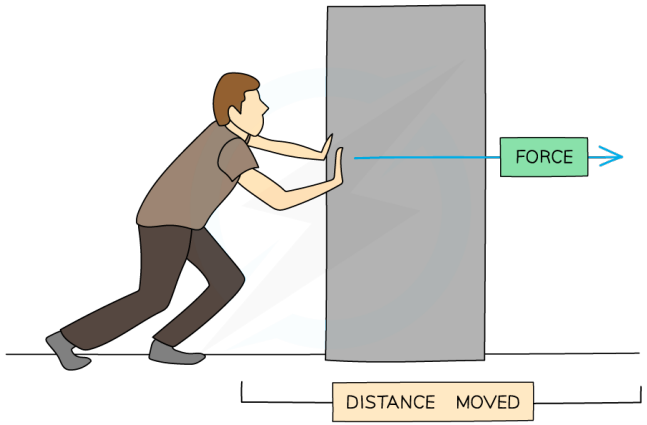
Work (Simplified)

- Since the force and movement are in the same direction:

$$\text{Work (W)} = \text{Force (F)} \times \text{Distance (d)}$$

- Work can also be calculated with the change in Kinetic Energy

$$\text{Work (W)} = KE_{final} - KE_{inital}$$



Power

- In physics, power is defined as the rate at which work is done or energy is transferred.
- The standard unit of power is the watt (W)
- The Pi-top has a 38.85 Watt-hour Lithium Battery

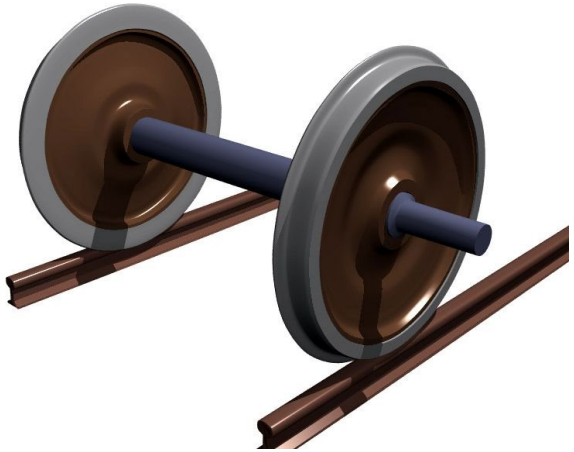
Power (P) = Force * Velocity

Run your Experiment

- Run the experiment with your own Alex robot.
- The approximate weight of the standard Alex build is 3.18 kilograms.

Further Challenge: Forklift Assembly

- In this challenge you will use your knowledge on Encoder and Servo motors to design and build a forklift to move a train wheel-axle assembly to predestined locations.



What is a Wheel-Axle Assembly?

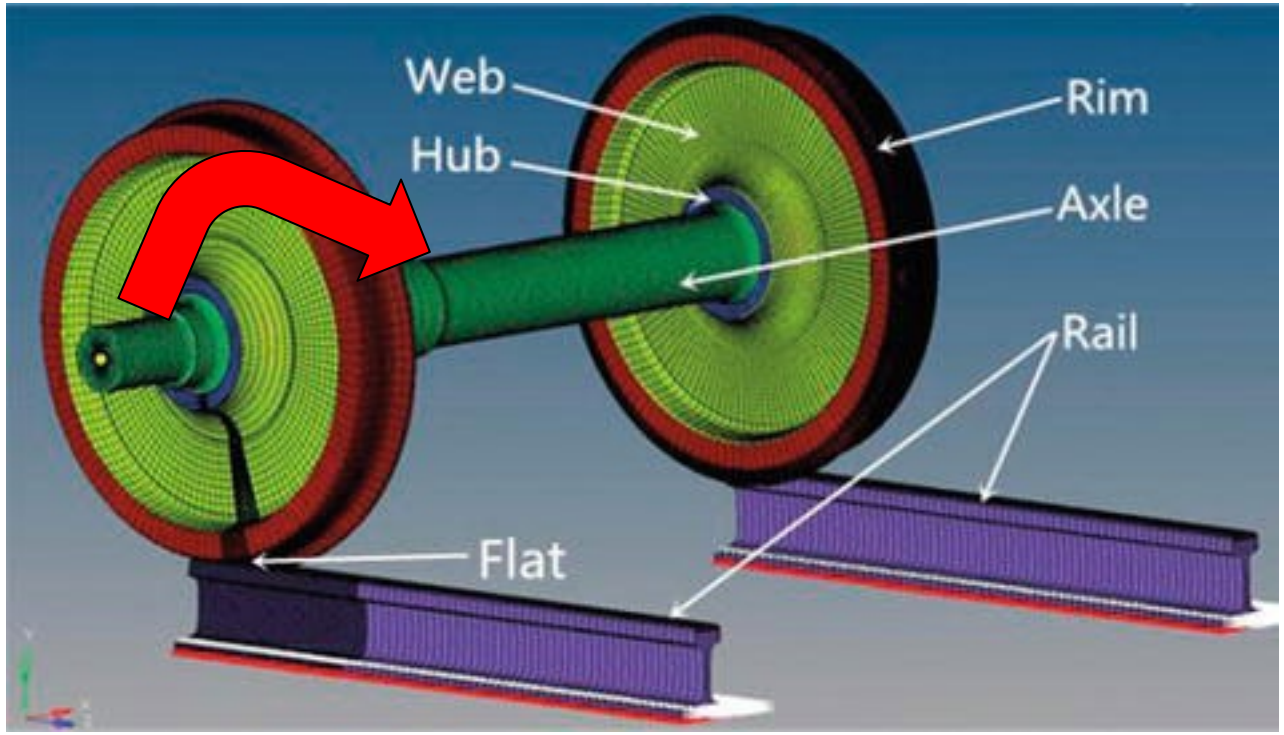
The axle connects both wheels where they move together while traveling at the same speed.

Approximately weighs 400 pounds (depending on size & material)

Wheel-Axle Assembly

The axle connects both wheels where they move together while traveling at the same speed.

Approximately weighs 400 pounds (depending on size & material)



How Are They Damaged?

Most of the time is due to **sudden braking(stopping)**.

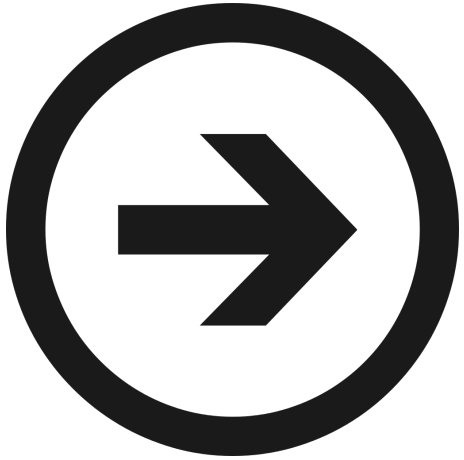
The **more weight & load the freight train carries, the more dangerous** it can become.

Train Derailments

What caused the train derailment in Ohio? Could it have been prevented?

What can we do?

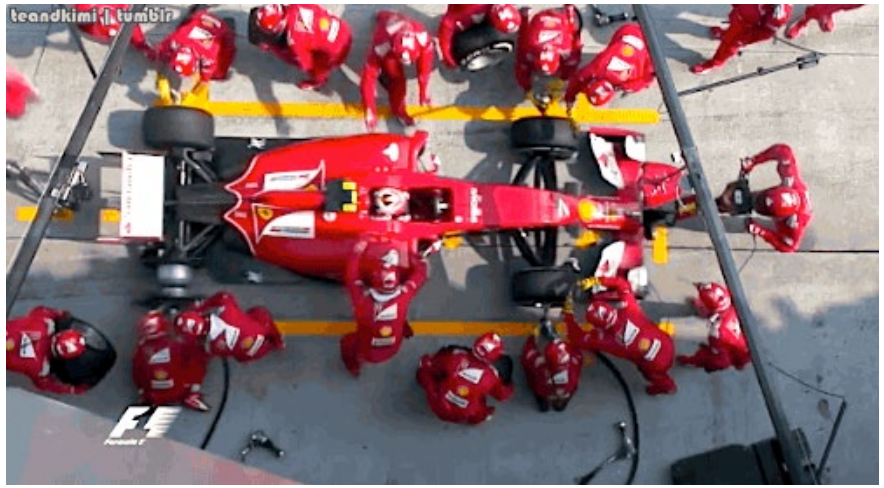
Severe braking over time can cause **fractures, scrapings & deterioration of the Wheel-Axle Assembly**...which can then cause **accidents**.



This needs **maintenance** very frequently. This is where Rail Flaw Detection can be used to **avoid** such accidents and failures.

Also...

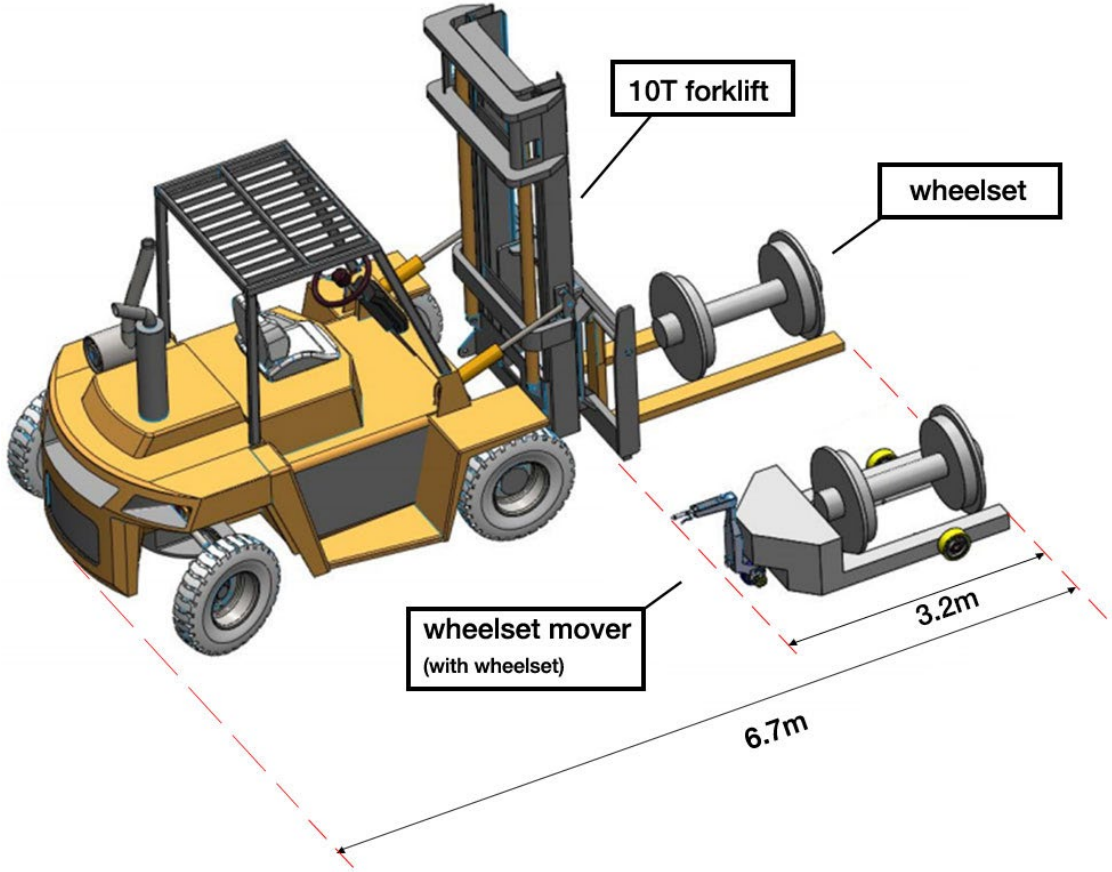
Just like we change the wheels of our cars very often (for every 50,000 miles traveled).....



....Freight trains need an exchange of Wheel-Axle Assemblies for every 500,000 miles as well.

How Are They Moved?

Wheel-Axle Assemblies are moved by **forklifts** to make the process more **efficient(easy)**. Helps place Wheel-Axle Assemblies at a desirable spot as well.



Build and Program a Forklift

Day 5

- Final Competition

Final Competition

Instructions

- In this final competition the goal is to run a track as fast as possible without getting off track, or crashing into any obstacles, all while driving the robot through its camera feed only.

Rules before competition starts

- You can test any section of the course as much as you want.
- Once the official competition begins you have 3 tries showing the final solution.

Rules during the competition (Done in less than 3 minutes = 50 points, -10 points any extra minute)

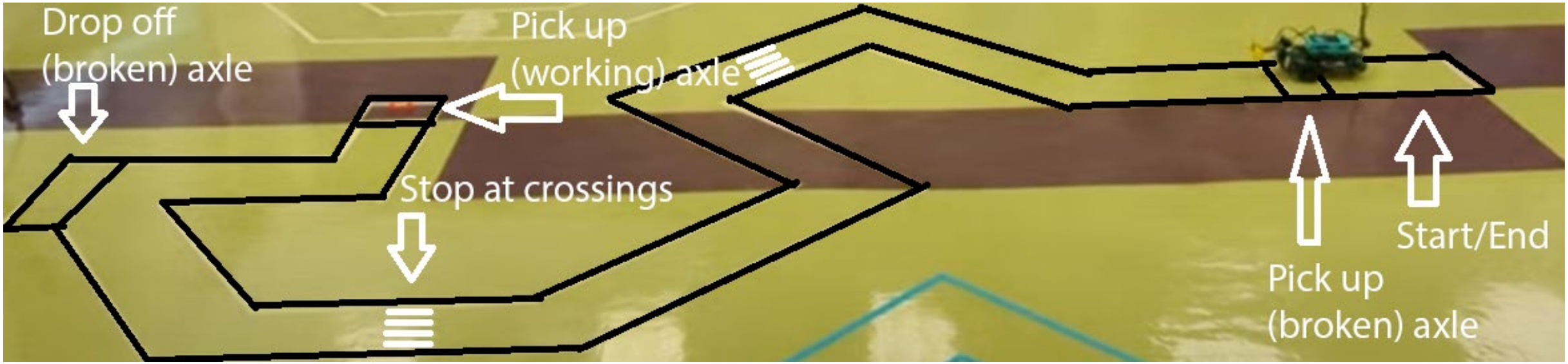
- You must pick up the broken wheel-axle assembly at the start of the track. (Forklift needed)
- Carry the wheel-axle assembly all the way to its drop off location.
- Pick-up the working wheel-axle assembly and bring it back to the beginning of the course.

Bonuses (+10 each extra):

- Stop at pedestrian crossings only if there are pedestrians crossing (Ultrasonic sensor needed)
- Use the light sensor and LEDs to detect low lighting and turn on your headlights automatically.
- Use the buzzer and LEDs to beep and blink your back lights when going in reverse.
- Turn on the back LEDs when the robot is not moving
- Use the Camera to drive using the Line Follower algorithm

Exiting the course or dropping the axle will have a 5 point penalty for each instance.

Final Competition



Final Competition Rules

- Start by following the blue line with the line follower, stop the AI with esc when it reaches the end.
- Manually pick up the axle and bring it back to the start using the line follower and drop at the specified location
- Repeat for lines green and yellow respectively.
- Fastest team, or team that gets the farthest wins.