FIRIA LABS

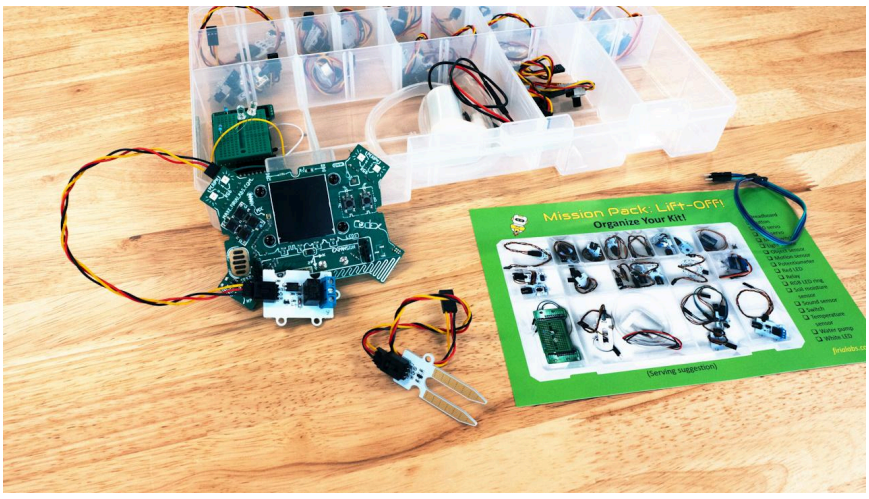
Curriculum Guide

Preview

Mission Pack:

Lift-Off with CodeX

Table of Contents



Lift-Off with CodeX Overview

This mission pack expands Python coding knowledge gained during the Python with CodeX Mission Pack. During this mission pack, students will complete outer-space themed projects in your mission to Lift-Off with code. Adding peripherals to the CodeX allows students to interact with the world in new and exciting ways.

Pre-Mission Assignment

Students may benefit from reviewing the foundations of computational thinking. Discuss algorithms, variables and constants, functions, loops and conditional statements. Go over debugging strategies. Review basics of Python coding, like indenting, use of capitalization, importing libraries, etc.

Mission 1: Welcome to Lift-Off with CodeX

Students will receive the Mission Briefing about their trip to outer space. They explore the different peripherals in their kit, and connect their first peripheral - the red LED light.

Mission 2: Lift-Off!

Students will get the rocket ship off the ground by setting up a power switch, launch button, and countdown sequence.

Mission 3: Conserve Energy

Students will conserve energy on the ship by using motion detection to control when lights come on and how bright they are.

Mission 4: Hatch Lock

Students will use the NeoPixel ring and microswitch to make sure the shuttle's hatch locks are sealed.

Mission 5: Alert System

Students will use the temperature and sound sensors to design an alarm system that will let the crew know if something has gone wrong with the ship.

Mission 6: Life Support

Students will create proper air circulation on the ship by using a 360 Servo to rotate the fans.

Mission 7: Solar Tracking

Students will rotate solar panels to harness enough of the sun's energy to power the ship to Mars.

Mission 8: Prepare Lander

Students will use the object sensor to land safely on the surface of Mars.

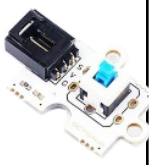
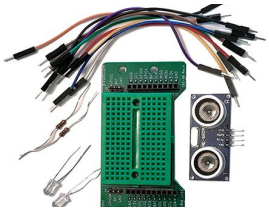
Mission 9: Automatic Garden

Students will build a system to sense soil moisture and automatically water a garden.

Mission 10: Exploring the Surface

Students fire up the Martian Rover to explore the surface, watching out for large boulders that could damage the Rover.

Peripherals Kit Overview

Peripheral	Description	Missions	Peripheral	Description	Missions
	Button A standard momentary push button, used in applications for input.	2		Motion Sensor A PIR sensor, used for motion lights and alarms.	3
	Switch Locks into place when pressed; generally used to apply power or change a setting.	2, 6		Object Sensor It contains an LED that emits IR light and a phototransistor that detects IR light from nearby objects.	8
	Red (and White) LED Light emitting diode. Outputs red (or white) light.	1, 2, 3, 5, 7		8 RGB LED Ring Also called a NeoPixel ring. Pixels can be illuminated in any color individually.	4, 8
	Microswitch A simple form of a button; can be used as a crash sensor or to detect touch.	4, 8		360 (Continuous) Servo This servo rotates 360 degrees and can operate continuously.	6
	Potentiometer Often referred to as a knob; can be physically turned for variable input.	3, 5 +		180 (Positional) Servo This servo rotates forward or backward 180 degrees and can hold its position.	7, 8
	Temperature Sensor Can read raw values of a temperature.	5		3V Relay Can be used to switch larger power and voltages to devices.	9
	Sound Sensor Is sensitive to sound intensity; can be used to detect noises.	5		Soil Moisture Sensor Detects the amount of moisture present in the soil surrounding it.	9
	Light Sensor The photocell measures ambient light; used for solar monitoring & light dimming.	7		Water Pump A small water pump designed for submersible operations.	9
	Divider A circuit that cuts the voltage from analog sensors in half so you can get the full range of the sensor.	3, 5, 7, 9		Mission 10 External Peripherals • Mini Breadboard • Red and Amber LED • 2 100-Ohm Resistors • HC-SR04 Ultrasonic Distance Sensor • 10 Jumper Wires	

Mission 1: Welcome to Lift-Off with CodeX

Overview and Notes: Your Mission: Should you choose to accept it...

We're going to outer space! This first project is all about getting to know the **peripherals** hardware. Before your students finish, they will connect the red LED peripheral to the CodeX and write some code to turn it on.

You may need to show them how to properly connect a peripheral to the CodeX. Each peripheral has a small cable for connecting. At one end of the cable is a latching connector, which fits into a mating receptacle on the sensor. The other end of the cable has a non-latching connector which connects to the CodeX. Both the sensor and the CodeX have "G" and "S" labeled to help you know the direction of the cable.

For the latching connection to each sensor, feel free to leave the cable attached after first use. Should you need to disconnect the latching connector from the sensor, there is a small lever that can be depressed with your thumb or a small screwdriver. For some connectors a slight downward flexing of the connector (in the direction you're pushing the lever) will help it clear the catch.

Remind students to be careful when using the peripherals, that they don't pull on the wires to disconnect, but rather grasp the connector housing when doing so.

Preparation and Materials:

- Create a class on the teacher dashboard.
- Students need a computer / laptop with the Chrome web browser.
- Make sure the students can successfully login to http://make.firialabs.com, create a student account and join the class with the code.
- Each student (or pair) needs a CodeX, peripherals kit and cables (red LED).

Peripherals Used

Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-03 • 2-AP-11 • 2-AP-14 • 2-AP-19 • 2-IC-20	• 3A-CS-01 • 3A-CS-02 • 3A-IC-24	• 3B-CS-02 • 3B-AP-14 • 3B-AP-16 • 3B-IC-27

Mission 1: Welcome to Lift-Off with CodeX	**Time Frame:** 30 - 60 minutes

Learning Targets

- I can safely connect and disconnect the red LED peripheral to my CodeX.
- I can use the peripheral value property to control the peripheral.

Key Concepts

- Cable colors matter – they must match.
  - Black is GROUND
  - Red is VCC Power
  - Yellow is SIGNAL
- One cable end has a latch and is connected to the peripheral.

Assessment Opportunities

- *Check for Understanding* in CodeSpace
- Match peripheral to name/description
- Exit ticket with cable colors
- Submit and/or check the **PeriphIntro** program
- Journal entry on their learning experience

Success Criteria

- ☐ Identify the peripherals in the Lift-Off kit.
- ☐ Successfully connect and disconnect the red LED to the CodeX.
- ☐ Write a program that successfully turns on and off the red LED.

Vocabulary

- **Peripheral**: A device that interacts with the CPU (common peripherals are LED lights, display screen, buttons, mouse, keyboard, and printer)
- **Function:** A named chunk of code you can run anytime just by calling its name; also called a procedure
- **Parameter:** A local variable in a function that receives a value passed into the function when it is called; information the function needs to complete its task
- **Argument:** Passing data to functions (information a function uses to complete its task)
- **Variable:** A name you assign to some data used in code instead of the literal, or actual, values
- **Constant:** A name for a value that doesn't change during program execution

New Python Code

`exp.digital_out(exp.PORT0)`	Used to set up a digital output peripheral (LED)
`led.value`	The property of the LED peripheral used to turn on/off the light

Real World Applications

- Discuss the fact that all electronic devices have circuit boards inside.
- Challenge students to name a few peripherals they use every day, similar to the ones in the kit.
- Encourage students to give examples of how their lives are impacted by technology. See examples below:
  - Bluetooth speaker, headphones
  - VR headsets, webcams, etc.
  - Medical equipment
  - Technology in transportation or entertainment

Extensions

- Light up the CodeX pixel 0 to indicate where the peripheral is connected.
- Use the A and B buttons of the CodeX to turn on/off the red LED.
- Set up and write code for the white LED. Use the CodeX buttons to control the red and white LEDs.

Cross-Curricular

- **LANGUAGE ARTS:** Students write a first-person essay about the impact of technology.
- **SCIENCE:** Students select one peripheral and research how it works or its uses in a science field.
- **MATH:** Many peripherals are digital and only use the values True or False. Review binary numbers.

Mission 2: Lift-Off

Overview and Notes: Let's go to Mars!

This mission is all about getting the rocket ship off the ground. Students will create a power switch for the engines, a countdown sequence for personnel and a launch button for Mission Control. Ideally students will refresh their Python knowledge as they get the mission to Mars underway.

You will want to remind students to be gentle as they connect and disconnect the peripherals from the CodeX. You can work with students in small groups as they practice.

This project gives students plenty of opportunities to engage in the computational thinking skill of abstraction. Encourage students to carefully read about abstraction in Objective 5 and continue to remind them that when they are creating functions, they are practicing abstraction. The concept of algorithms is also in this mission and throughout the mission pack. Review and practice algorithms with your students and encourage them to see algorithms in their daily lives.

Helpful Reminders:

- Always start a new program by creating a new file and naming it appropriately.
- Students are making a project, not just working on random problems. Have them focus on the project-based objectives and avoid rushing through the material too quickly.
- Students should collect all the Tools they find in the instructions so they can reference them later when they need additional help.
- Encourage documentation and testing strategies.
- Read carefully and check the hints. Usually the answer is there right in front of them!
- Students can test their understanding along the way by trying stuff! Have them "color outside the lines."

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- To login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (red LED, white LED, switch, button)

Peripherals Used

Red & white LEDs

Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-03 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14	• 3A-CS-02 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18	• 3B-CS-02 • 3B-AP-10 • 3B-AP-14 • 3B-AP-16

Mission 2: Lift-Off | **Time Frame:** 45-75 minutes

Learning Targets

- I can describe the difference between a button and a switch.
- I can read the value of a button and switch.
- I can use an infinite loop to continuously check the values of the button and switch.
- I can use a button and switch to control an LED.
- I can break out of a loop.
- I can display a countdown sequence.

Key Concepts

- Understand the difference between a button and a switch. The switch locks in place while the button does not.
- The project uses two input peripherals: button and switch. Each controls a different aspect of the lift-off project.
- Use three branches in the infinite loop. One branch will have two conditions.

Assessment Opportunities

- *Check for Understanding* in CodeSpace-(2)
- Practice evaluating conditional statements
- Exit ticket - construct a conditional statement
- Submit and/or check the **LiftOff** program
- Journal entry on their learning experience

Success Criteria

- ☐ Use a switch to turn on/off the red LED.
- ☐ Use a button to launch a countdown sequence.
- ☐ Show a countdown sequence on the CodeX display that includes a flashing white LED.

Vocabulary

- **Abstraction:** the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics
- **Algorithm:** a sequence of steps for completing a task (step by step process)
- **Branching:** Decision points in code; a condition

New Python Code

`exp.digital_in(exp.PORT0)`	Used to set up a digital input peripheral (button, switch)
`button.value`	Returns the button's position: True (not pressed) or False (pressed)
`switch.value`	Returns the switch's position: True (out) or False (in)
`display.fill()`	Fills the CodeX LCD screen with a pre-defined or RGB color

Real World Applications

- Anything that has a lever, button or switch is a real world example. Have students come up with their own examples. Some items are listed below:
  - Elevator buttons and doorbells, keyless entry on an automobile, "start Engine" on an automobile
- The mission will use a function. Discuss abstraction and how it hides details of embedded technology.
- The mission uses an algorithm. Have students develop their own algorithms for:
  - Daily tasks, other computer programs, how they think an electronic device may work, etc.

Extensions

- Display images of a rocket ship taking off during the lift_off phase.
- Add the CodeX pixels as flashing lights during the countdown phase. Use random colors.
- Each astronaut may need to indicate they are ready for lift-off. Use the CodeX buttons as inputs for the astronauts to push in sequence to indicate they are ready.

Cross-Curricular

- **LANGUAGE ARTS:** Have students write about a time they prepared to leave for somewhere. Or discuss the theme of transitioning with a current book assignment.
- **SCIENCE:** Discuss gravity or Newton's Laws and how they relate to the mission to Mars. –OR– include a lesson on space.
- **MATH:** Create a chart of the distance traveled by the ship. –OR– Draw a rocket ship trajectory and then find the equation of the line.

Mission 3: Conserve Energy

Overview and Notes: It's a long way to Mars. Maybe we should plan...

Mars is a hike! In this lesson, students work with three different peripherals to detect motion, simulate a light source, and control the activation/brightness of the lights after motion has been detected.

One of the more challenging concepts for this project is the combined use of digital and analog sensors. It might be helpful to have students discuss the difference between the two types of devices and review the data they return.

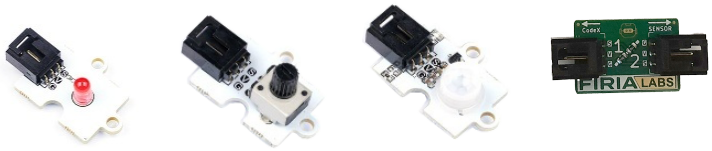
As a cross-curricular tie-in, and as a hook, you could have students talk about a road trip and how they would plan. They can use a map as a geography lesson, calculate distance using the scale of a map, and work on a plan for how much money they will need, how often they will stop, gas mileage, etc. Then have them find the distance to Mars. Have them consider how their plans might change if they knew they were going that far, and how long it will take. If they don't organically bring up conserving energy, pose a question that requires them to consider that particular resource.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- To login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (white LED, potentiometer, motion sensor, divider)
- A light source, like a flashlight

Peripherals Used



White LED

Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-19	• 3A-CS-02 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16

Mission 3: Conserve Energy | **Time Frame:** 45-75 minutes

Learning Targets

- I can use the potentiometer to control the level of brightness on an LED.
- I can use the motion sensor to detect movement.
- I can use a timer to keep the LED on after motion has been detected.

Key Concepts

- Understand the difference between reading analog (potentiometer) and digital (motion sensor) input.
- The LED is first set up as a digital input peripheral, but is changed to analog using PWM–pulse-width modulation.
- PWM requires a duty cycle and frequency.

Assessment Opportunities

- *Check for Understanding* in CodeSpace-(2)
- Identify digital and analog inputs around them
- Exit ticket - explain PWM
- Submit / check the **ConserveEnergy** program
- Journal entry on their learning experience

Success Criteria

- ☐ Control the brightness of an LED with the potentiometer.
- ☐ Keep the LED on when motion is detected using a timing technique.

Vocabulary

- **Analog:** A peripheral with a range of integer values – from 0 (off) to 2^16 - 1 (full power)
- **Digital:** A binary peripheral with two states – True or False
- **ADC:** Analog to digital converter – converts an analog measurement to a finite digital value. For CodeX, which is a 16-bit microcontroller, the digital values range from 0 to 2^16-1 (65,535)
- **Pulse-Width Modulation:** Analog measurement where on/off pulses are sent at a constant rate, determined by the duty cycle and frequency (or analog period)

New Python Code

`exp.pwm_out(exp.PORT0)`	Used to set up a peripheral with PWM; requires a duty cycle and frequency
`exp.analog_in(exp.PORT1)`	Used to set up an analog input peripheral (potentiometer)
`led.duty_cycle`	Determines power to the LED (higher integer is a brighter light)
`sleep_ms()`	Delays program execution in milliseconds
`time.ticks()`	Returns the current clock time (elapsed time since last reboot)
`motion_sensor.value`	Returns the motion sensor value: True (detected) or False (not detected)
`potentiometer.value`	Returns an integer as the potentiometer knob's position

Real World Applications

Motion sensors and built-in timers are used in a variety of ways.

- Lights that come on automatically or shut off automatically. For example, when you go to the grocery store the light comes on when you walk by the refrigerator case.
- Have students discuss other applications of automatic technology, like air conditioning, etc.

The need for conserving energy is real.

- Lead a discussion or lesson on energy conservation. What are different energy sources? What are the pros and cons of each? Which ones might work in outer space?

Extensions	**Cross-Curricular**
• Use the CodeX to add an indicator in energy-saving mode and energy-wasting mode. Example: sound, pixels, display, etc. • The CodeX also has a built-in light sensor. Use another motion detector to control the other sensor.	Many suggestions for a cross-curricular project are included on the previous page. They include ideas for: • **LANGUAGE ARTS, MATH, GEOGRAPHY** Additional **SCIENCE** ideas: • Distance, rate & time • Conserving energy • Artificial vs natural light • Volts and how the divider works

Mission 4: Hatch Lock

Overview and Notes: Is it drafty in here?

Now that the crew has a way to conserve energy on the way to Mars, they need to make sure they can successfully dock with a supporting craft that was launched ahead of the mission.

With 8 locks that need to be successfully engaged, this project supports mathematical conversations about failure rates along with the percentages and statistics related to them.

Since the locks only work 85% of the time, students will be generating a random number between one and one hundred to mimic what happens in real life. If students are not familiar with percentages or struggle with interpreting data, you might need to have a conversation about how the code for checking the value of the random number models whether or not a hatch is locked.

Helpful reminders:

- Colorblind students might struggle if they inadvertently focus on certain color combinations. The mission can be modified so they use color combinations that work for them.
- The NeoPixel ring can be plugged directly into the CodeX without an additional cable.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- To login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (NeoPixel ring, microswitch, optional switch)

Peripherals Used

Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-19	• 3A-CS-03 • 3A-DA-09 • 3A-AP-13 • 3A-AP-14 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-AP-10 • 3B-AP-12 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17

Mission 4: Hatch Lock | **Time Frame:** 45-75 minutes

Learning Targets
- I can connect the NeoPixel ring to the Codex.
- I can adjust the brightness and colors of the pixels on the NeoPixel ring.
- I can use random values to change the colors of the LEDs on the NeoPixel ring.
- I can return a random failure rate of 15%.

Key Concepts
- The NeoPixel colors are set using tuples - a sequence of values similar to a Python list.
- Variables can be used to give your code memory, like if a button is pressed.
- A True/False failure rate can be calculated by generating a random number and comparing it to a selected range.

Assessment Opportunities
- *Check for Understanding* in CodeSpace-(2)
- Have students develop the calculations for different failure or success rates
- Exit ticket - Show code to generate a random number in a specific range
- Submit / check the **HatchLock** program
- Journal entry on their learning experience

Success Criteria
- ☐ Use the microswitch to control the NeoPixels.
- ☐ Use a loop to set the colors or the NeoPixels.
- ☐ Use a random number generator to simulate the probability of hatch lock failure.

Vocabulary
- **RGB:** Red, Green, Blue; the colors that make up a single pixel on the screen
- **Pixel:** Picture element; tiny dots used to compose images and text on a digital screen
- **Tuple:** An *immutable* sequence of items that you can access with an *index*, or a list with values that don't change. A read-only version of a list.
- **Index:** a common method for referencing the elements in a list, tuple or string using a number

New Python Code

`neopixel.NeoPixel(exp.PORT0, 8)`	Sets up the NeoPixel ring; indicates the port and number of LEDs
`power.enable_periph_vcc(True)`	Turns on extra power to the NeoPixel ring
`randint(low, high)`	Returns a random integer between and including low and high
`(red, green, blue)`	A tuple with three items; used for RGB colors
`np[pixel]`	Accessing a single item [pixel] in a list np
`return`	Returns a value to the statement calling the function

Real World Applications
RGB pixels are used in many digital items
- Video games, digital displays that light up as you complete a task
- NeoPixels can be used in traffic signals, stadium billboards, concert lighting, etc.

Extensions
- Include the disco ball with the hatch lock project.
- Use buttons on the CodeX to control the disco ball speed (speed up, slow down).
- Add a signal when all 8 hatch locks are successful (sound, image, CodeX pixels, etc.).

Cross-Curricular
- **LANGUAGE ARTS:** Have students write about a personal experience where something failed.
- **SCIENCE:** Discuss RGB in the context of light. Have a lesson on the science of pixels.
- **MATH:** Explore rates (like failure rates) and construct the algorithm for calculating them.

Mission 5: Alert System

Overview and Notes: Did you hear something?

Being on another planet can get lonely sometimes. With no neighbors to check on the crew, they'll need a good alarm system. In this project students will consider the technical dangers a crew can encounter on the ship. It might be helpful to have students think about how different types of sensors might be used to track the temperature or detect an explosion.

The complexity of this project might be challenging for students – so schedule several checkpoints as students are working. Some functions have very similar names which can make it difficult to find errors. Students are asked to convert raw analog data from a sensor into degrees Celsius.

The exponential moving averages calculation is also introduced, which is heavily dependent on statistics. If it makes sense for your students, show a graph of an exponential function and talk about how the relationship between two quantities is not linear. You could also discuss moving averages related to fast-moving stock trading markets and give a bit of context to the idea of statistics. And yes, this would be a great time to collaborate with a math teacher!

Helpful reminders:

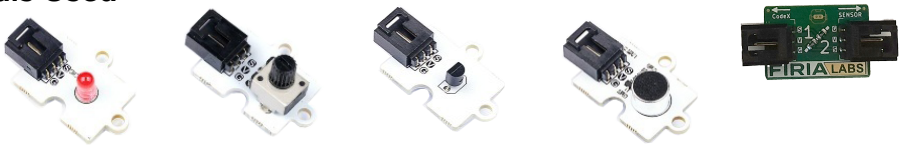
- Depending on the temperature of a student's hand, the sensor might not increase very much. After testing with students you may need to adjust to the proper threshold. Or use something to cool down the temperature sensor before squeezing between fingers.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- Login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (red LED, potentiometer, temperature sensor, sound sensor and divider)
- Optional: something to cool and / or warm the temperature sensor

Peripherals Used



Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-DA-07 • 2-DA-08 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-19	• 3A-CS-02 • 3A-CS-03 • 3A-DA-09 • 3A-DA-12 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-DA-07 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17

Mission 5: Alert System	**Time Frame:** 60-90 minutes

Learning Targets

- I can read data from a sensor and display it on the Console Panel.
- I can convert a raw analog temperature reading to Celsius.
- I can read an analog value from a sound sensor.
- I can calculate the average sound in the room.

Key Concepts

- Raw sensor data must often be converted into a different form before it can be used.
- Sensor behavior can be controlled by setting the analog period.
- A threshold can be used with sensor readings to determine when something is out of normal range.
- Real-world situations can be explored through simulations.

Assessment Opportunities

- *Check for Understanding* in CodeSpace-(2)
- Display temperature readings on the Console Panel. Compare readings with different CodeX.
- Display sound sensor readings on the Console Panel. Compare readings with different CodeX.
- Exit ticket - Explain the difference between how a temperature sensor and sound sensor work
- Submit / check the **AlertSystem** program
- Journal entry on their learning experience

Success Criteria

- ☐ Read data from a sensor and display it on the Console Panel.
- ☐ Convert raw data from the temperature sensor to degrees Celsius.
- ☐ Use data from the sound sensor to calculate an average sound value.
- ☐ Control an alarm system with data from sensors.

Vocabulary

- **Duty Cycle:** The percentage of time power is ON during pulse-width modulation
- **Frequency:** The analog period, or how rapidly the device pulses during pulse-width modulation
- **REPL:** “Read, evaluate, print loop” command line that enables print statement output
- **Simulation:** Code that builds a *model* of something, and lets you play with that model to explore "virtual" situations
- **Threshold:** A specific limit or point that must be met or exceeded in order for something to occur

New Python Code

`LED_ON = 2**16 // 2`	Maximum duty-cycle for an LED using PWM
`LED_OFF = 0`	Minimum duty_cycle for an LED using PWM
`exp.pwm_out(exp.PORT0, frequency=2)`	LED using PWM for blinking light
`degrees_c = raw_temp*0.004577 - 50`	Conversion of raw temp to degrees Celsius
`avg_sound = avg*(1-WEIGHT) + new_val*WEIGHT`	Average sound calculation using exponential moving average
`import soundlib`	Import the sound library to add non-blocking sound functions
`siren = soundmaker.get_tone(“violin”)`	Sets up a variable for the sound
`siren.set_pitch()`	Sets the pitch tone at the given frequency
`siren.play()`	Plays the sound at the set pitch

`siren.glide()`	A non-blocking way to ramp the pitch from the current setting to a new setting over a specified amount of time
`siren.stop()`	Stops playing the tone
`global temp_limit`	Allows for updating a global while being used in a function.
`print("Temp:", degree_c)`	The print() function displays text on the console panel

Real World Applications

Digital and analog sensors are used in a variety of ways:

- Home surveillance detectors and alarm systems
- Refrigeration and cooling in the food industry
- Sensors in cars to detect problems (low tire indicator, overheating, etc.)

Have students discuss how sensors are used in their lives to help people.

Extensions

- Change the raw temperature to Fahrenheit.
- Use buttons on the CodeX to use either Celsius or Fahrenheit temperatures.
- Find images that go with the different alerts and display them on the CodeX screen when the alarm is triggered.
- Like a traffic light, add a "warning" period before the alarm is triggered, that alerts the crew to a potential danger before the actual threshold is reached.
- Check for the sound reading two different ways. Below threshold could mean power failure and no electronics are working. Above threshold could mean an explosion. Give a different warning for each situation.

Cross-Curricular

- **LANGUAGE ARTS:** Have students write a poem about a topic from the lesson, or about their coding experience.
- **SCIENCE:** Explore sound waves and sound in space.
- **MATH:** Many applications from this lesson
  - Make a chart of the sensor readings (temperature and sound).
  - Practice converting temperatures – Fahrenheit to Celsius, Celsius to Fahrenheit.
  - Review the calculations for changing volts to celsius. Graph some sample data and then write the equation.
  - The lesson uses EMA for the sound average. Compare and contrast other ways to find a weighted average.
- **ART:** Students can draw (or use mixed-media) the interior of their own spacecraft. Discuss the use of color and/or perspective in their artwork.

Mission 6: Life Support

Overview and Notes: Astronauts need air too!

It doesn't matter how safe the crew is if they can't breathe. Ensuring consistent air quality is key to the success of this mission. Students will be using a servo to make sure the air circulates throughout the spacecraft. This project is a fantastic opportunity to integrate life sciences with computer science. Students can discuss the impact of poor air quality on human health.

Percent of Max	Speed of Motor	Rotation Direction
21	100%	Clockwise
42	50%	Clockwise
62	0%	Stopped
80	50%	Counterclockwise
100	100%	Counterclockwise

You may need to discuss the duty cycles of the servo to help students understand the servo's analog control signals. You can use this duty cycles table as a guide.

You can learn more about servos and duty cycles at: PCB Cadence

Helpful reminders:

- The wires are slightly different on the servo. Brown corresponds to black; while orange corresponds to yellow.
- Students only need one of the plastic pieces included with the servo. You may want to remove the others as well as the screws before distributing the peripherals to students.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- Login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (360 servo, one servo horn, switch)

Peripherals Used

Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-DA-07 • 2-DA-08 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-17 • 2-AP-19	• 3A-CS-01 • 3A-CS-03 • 3A-DA-11 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-DA-07 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 6: Life Support	**Time Frame:** 30-60 minutes

Learning Targets

- I can set the analog period on the 360 servo.
- I can make the servo spin clockwise and counter-clockwise.
- I can start and stop the servo using a switch.

Key Concepts

- The duty cycle determines the speed and direction of the 360 servo.
- The program can use “states” or phases to determine what happens.
- Use nested if statements to transition from one state to the next.

Assessment Opportunities

- *Check for Understanding* in CodeSpace
- Make a chart of percent, speed and direction of a servo, based on observation using code.
- Exit ticket - Draw a diagram of the finite-state machine for this mission.
- Submit / check the **LifeSupport** program
- Journal entry on their learning experience

Success Criteria

- ☐ Connect a servo with servo horn to the CodeX.
- ☐ Use a function call to control the servo’s speed and direction.
- ☐ Control the servo with a switch.

Vocabulary

- **Servo:** A DC motor with a controller circuit, internal feedback mechanism and gearbox. The 360 continuous servo goes in both directions at different speeds. The 180 positional servo turns in either direction to a specific angle and holds that position.
- **Finite-State Machine:** The status of a system with transitions. With this system, your program can only be in one of a known set of “states” at any given time. Usually “state” is based on variables in your code.
- **State:** A phase of a program. Keeping track of states helps you understand and manage your code. Each state might have its own set of conditions it is tracking.
- **Transition:** Moving between states; the program transitions from one state to another when certain conditions are met.

New Python Code

`servo = exp.pwm_out(exp.PORT0, frequency=20)`	Set up a 360 servo using PWM with a frequency of 20
`//  example: CYCLE * percent//100`	Division that returns only the integer (no decimal, no rounding)

`if state == "maintenance":`
`  if switch.value == POWER_ON:`
`  state = "active"`
`  fan.duty_cycle = set_servo(FORWARD)`
	Nested if statements. The first if statement is checked. If true, it will check the second if statement. If false, the block of code is skipped.
	*This example also shows **transitioning** to a different **state**.*

Real World Applications

Servos can be used in a variety of applications:

- Motors on rover wheels for terrain exploration.
- The life support fans in a hospital.
- Hydraulic pump operation.
- Controlling the cabin pressure on an airplane.

Have students discuss where and how servos might be used in their lives.

Extensions

The mission ends with a working life support system, but you can make it more sophisticated. Some extra features you can add to the system are:

- Have multiple speeds for the fan and cycle through the speeds, like a ceiling fan.
- Add a temperature sensor and increase the speed of the fan when it is warm.
- Use the computer's clock and set a timer for the fan to stay on.
- Use the motion detector to turn on the fan.
- Use the potentiometer to make the fan turn at variable speed and direction.
- Use the CodeX NeoPixels or display screen to show additional information.

A more advanced challenge:

- Use the accelerometer to simulate the effects of movement on the air circulation system. Tilt the CodeX to represent changes in the spaceship's orientation, which could affect airflow and require adjustments.

Cross-Curricular

- **LANGUAGE ARTS:** Have students read a technical report on servos, and then summarize the information in a paragraph, or share verbally with each other.
- **SCIENCE:** How much air does a person or animal need? Have a lesson about oxygen levels and survival.
- **SCIENCE:** How much power will the fan need to operate? How can the ship power the fan? Have a lesson about energy and conservation.
- **MATH:** Using the program code, have students experiment with different percent values and chart the speed and direction for each data point. For more advanced math, plot the data points and write the equation for the servo.

Mission 7: Solar Tracking

Overview and Notes: Everybody loves the sunshine!

Throughout the Lift Off curriculum, students are asked to write code that helps the crew monitor and conserve different types of resources. This project is no different. For the Solar Tracking project the crew needs a way to generate energy from the sun. Learners will use a light sensor to determine when to "rotate" the solar panels to follow the path of the sun.

Percent of Max	Angle of Rotation	Direction of Rotation
25	90 Degrees	Clockwise
50	45 Degrees	Clockwise
75	0 Degrees	Centered
100	45 Degrees	Counterclockwise
125	90 Degrees	Counterclockwise

Like the Life Support project, there is a natural link between life sciences and computer science. Have students consider the impact of technology on ideas around clean energy as well as fossil fuels. There are cross curricular ties to mathematics. Students use percentages to solve the problems. You may also want to point out that the duty cycles for this servo's analog control signals are different (see the table to the left).

Helpful reminders:

- The wires are slightly different on the servo. Brown corresponds to black; while orange corresponds to yellow.
- Students only need one of the plastic pieces included with the servo.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- Login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (180 servo, one servo horn, light sensor, white LED, divider)

Peripherals Used

white LED

Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-DA-08 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-DA-11 • 3A-DA-12 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-DA-07 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 7: Solar Tracking	**Time Frame:** 60-90 minutes

Learning Targets

- I can set the analog period on the 180 servo.
- I can change the position of the 180 servo.
- I can read data from the light sensor.
- I can use states and a state variable to control the position of the 180 servo.

Key Concepts

- There is no “off” position for the 180 servo.
- A variable representing the state allows the program to remember information about events.
- The console panel can be used to monitor analog data.

Assessment Opportunities

- *Check for Understanding* in CodeSpace-(2)
- Make a chart of percent, angle and direction of a servo, based on observation using code.
- Exit ticket - Explain “bouncing” and how it affects digital input readings.
- Submit / check the **SolarTracking** program
- Journal entry on their learning experience

Success Criteria

- ☐ Connect a servo with servo horn to the CodeX.
- ☐ Use a function call to control the servo’s position.
- ☐ Read and display data from the light sensor.
- ☐ Use a variable for states to control the position of the 180 servo.

Vocabulary

- **Photoresistor:** A sensor that changes its resistance when light shines on it. A high intensity of light causes less resistance, and less light causes more resistance.
- **Bouncing:** When a digital input registers multiple times instead of once, like a button press.

New Python Code

`servo.duty_cycle = 0`	Stop a 180 servo
`state = 'morning'`	Define and initialize a variable for the state. Also, single quotes can be used for strings (see hint in Objective 4).

Real World Applications

Positional servos can be used in a variety of applications:

- Solar fields or solar powered devices
- Controlling robotic arms
- Positional devices, such as satellites, antennae, etc.

Have students discuss where and how positional servos might be used in their lives.

Extensions

- Use the LED with PWM and the potentiometer readings to change the light from dim to bright
- Use a timer or switch to turn on/off the LED
- Use a servo to spin the LED around the light sensor
- Add a temperature sensor and rotate the panels when the temperature is past a certain threshold value (too hot or too cold)
- Use the CodeX display screen to show additional information.
- Light up CodeX pixels to indicate the position of the panels.
- Use CodeX buttons to manually control the servo position.

Cross-Curricular

- **LANGUAGE ARTS:** Have students write a summary of their project, using technical terms.
- **SCIENCE:** Have a lesson on solar power.
- **SCIENCE:** What does a “day” look like in space? Have a lesson about light in space.
- **MATH:** Have a lesson about percents.
- **MATH:** Have a lesson about angles.

Mission 8: Prepare Lander

Overview and Notes: Another small step for mankind...

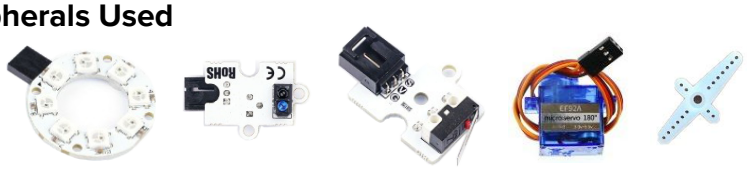
Getting safely to Mar is the easy part. Landing safely, now that's where the fun starts. A complete ground sensing system will be constructed using several peripherals. The Object Sensor will be used to detect the nearing surface of Mars. The NeoPixel will again be used to alert the astronauts of their impending landing. The Microswitch will be used to indicate surface contact and lastly, the 180 Servo will be used to extend and retract the landing gear. The concept of phases/states will be again used in the program.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- Login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (NeoPixel ring, object sensor, microswitch, 180 servo, one servo horn)
- A small Phillips screwdriver to adjust the object sensor
- A small ruler (with millimeters) for adjusting the object sensor

Peripherals Used



Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-DA-11 • 3A-DA-12 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 8: Prepare Lander	**Time Frame:** 45-90 minutes

Learning Targets

- I can connect and adjust the object sensor.
- I can change the position of the 180 servo.
- I can use the object sensor to trigger an event.
- I can use the microswitch to trigger an event.
- I can use states and a state variable to control the NeoPixel ring and 180 servo.

Key Concepts

- A variable that represents the state, or phase, allows the program to remember information about events.
- Different input peripherals can be used in the same program to change the state and trigger events.

Assessment Opportunities

- *Check for Understanding* in CodeSpace.
- Explain the steps or how the set_lighting() function works.
- Exit ticket - Draw a diagram of the states of the program.
- Submit / check the **PrepareLander** program.
- Journal entry on their learning experience.

Success Criteria

- ☐ Connect an object sensor to the CodeX and adjust it to the manufacturer specifications.
- ☐ Use the object sensor to trigger an event.
- ☐ Use the microswitch to trigger an event.
- ☐ Use a variable for states to control the NeoPixel ring and 180 servo.

Vocabulary

- **Pull:** A property that can be changed when setting up an input peripheral that determines the default value of a pin when nothing is connected. The pull can be set to “up” to move a weak pull toward 3 volts.

New Python Code

`def set_lighting(rgb_color):`
`  for pixel in range(8):`
`  np[pixel] = rgb_color`
	Set all pixels in the NeoPixel ring to one color. This was first used in Mission 4.

`sensor = exp.digital_in(exp.PORT1, pull=digitalio.Pull.UP)`	Change the pull property to “up” so the weak signal is pulled toward 3 volts, or “high”.

Real World Applications

The object sensor can be used for a variety of applications, including robotics autonomous cars, etc:

- Line or object detection
- Impact avoidance

Have students discuss where and how an object sensor might be used in common items, appliances, etc.

Extensions

- Use a CodeX button as an abort button.
- Display the current state on the display screen.
- Add a fourth state (before init) that requires some crew action.
- Use the CodeX buttons as a check-in for each crew member, indicating they are ready for landing.
- Add an alert sound when landing.
- Use the CodeX buttons as communication devices – a button press would indicate a phase to mission control.
- Once landed, use the CodeX’s ambient light sensor to adjust solar panels.

Cross-Curricular

- **LANGUAGE ARTS:** Have students compare and contrast this mission to previous missions.
- **SCIENCE:** Several missions now have used the finite-state machine model. There are many applications of this in science as well, such as the life cycle of something. Discuss how science has finite states.
- **SCIENCE:** Have a lesson on how the object sensor works.

Mission 9: Automatic Gardner

Overview and Notes: A good garden needs watering

The crew will be unable to bring all the food necessary to survive on the surface of Mars. They will have to be able to grow their own food once they get there. They will want this process to be automated as much as possible. We will construct a system to sense soil moisture levels and then automatically water the garden.

Discussion or research questions: Where will the crew get the water and seeds? Will things grow on Mars like they do on Earth?

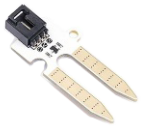
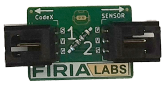
Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- Login to http://make.firialabs.com, with their student accounts
- A CodeX, peripherals kit and cables (relay, water pump, soil moisture sensor, divider)
- A small screwdriver for relay connections
- Two containers, water and soil

Peripherals Used



Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-DA-09 • 2-AP-10 • 2-AP-11 • 2-AP-13 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-DA-11 • 3A-DA-12 • 3A-AP-13 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-17 • 3B-AP-21

Mission 9: Automatic Gardner | **Time Frame:** 30-60 minutes

Learning Targets

- I can connect the relay to the CodeX.
- I can connect the water pump to the relay and CodeX.
- I can connect the soil moisture sensor to a divider and then the CodeX.
- I can prime the water pump.
- I can use the soil moisture sensor to trigger an event.

Key Concepts

- A relay is required to isolate circuits and use the water pump.
- Like other analog sensors, the soil moisture sensor can be used to control a peripheral.
- A button on the CodeX can be used to trigger an event, like priming the pump.

Assessment Opportunities

- *Check for Understanding* in CodeSpace.
- Draw a diagram of the relay/water pump setup.
- Exit ticket - Explain the meaning of the values the soil moisture sensor returns.
- Submit / check the **AutomaticGarden** program.
- Journal entry on their learning experience.

Success Criteria

- ☐ Connect the water pump and turn it on and off.
- ☐ Print the soil moisture reading to REPL.
- ☐ Use the soil moisture sensor to control the water pump.
- ☐ Use a CodeX button to prime the pump.

Vocabulary

- **NC / NO:** Terminals on the relay used for connecting peripherals. NO = normally open; this terminal is the most common one used.
- **Priming:** The process of removing air from pump lines.
- **Conductivity:** The ability of a material to conduct electricity. In this mission, it is water. More water in the soil means more conductivity.

New Python Code

`buttons.was_pressed(BTN_A)`	Returns True if the button was pressed since the last check; otherwise it returns False.
`relay.value`	The property of the peripheral used to turn on/off the relay (True or False)

Real World Applications

- The soil moisture sensor can be used in real-world applications, like farming and gardening.
- The relay is used in many real-world applications. Discuss with your students or have them research where they can find relays in their everyday lives.

Extensions

- Add NeoPixels and LEDs to indicate the status of the soil moisture. For example, “green” could indicate optimal moisture, “yellow” that watering is needed soon, and “red” for immediate watering.
- Show the process of watering with a sequence of LED lights to simulate the water flow.
- Use the LCD screen to display real-time soil moisture levels.
- Have settings for different plants, and use a CodeX button to determine which settings to use.

Cross-Curricular

- **LANGUAGE ARTS:** Have students write a lab report or technical paper about this mission.
- **MATH:** Experiment with different times for the pump to turn on and measure the water output. Create a graph with the data. As an **algebra** extension, write the equation for the data.
- **SCIENCE:** Have a lesson on conductivity. Use the soil moisture sensor in different substances.
- **BIOLOGY:** Have a lesson on plants and what they need to grow. How much sunlight? How much water? What are the alternatives?

Mission 10: Exploring the Surface

Overview and Notes: A new journey begins

After landing on Mars, the crew has a new journey ahead. One of their tasks is to explore the surface. But if the CodeX, representing the Rover, is damaged, the peripherals you have been using so far may not work. This project will use the breadboard and electronic components to detect an object, calculate the distance to the object, and develop a warning system for the Rover.

The project will require a breadboard connected to the CodeX for power. Then additional components will be added. First the ultrasonic sensor to detect an object and return important data. Then use the data to calculate the distance to the object and use LEDs as a warning system so the Rover can avoid harmful objects.

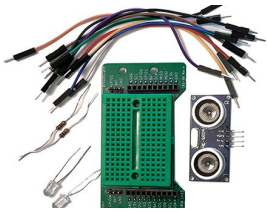
This project has many points of discussion and research: Warning systems, sonar detection, autonomous vehicles, and much more. This technology is used in so many real-world applications.

Preparation and Materials:

Each student / pair needs:

- A computer / laptop with the Chrome web browser
- Login to http://make.firialabs.com with their student accounts
- A CodeX, peripherals kit and jumper wires (external peripherals: breadboard, 2 LEDs, 2 resistors, and the ultrasonic sensor)

Peripherals Used



Standards addressed in the mission:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-DA-08 • 2-DA-09 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-DA-11 • 3A-DA-12 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-AP-21 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-AP-10 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 10: Exploring the Surface	**Time Frame:** 60-90 minutes
Learning Targets • I can connect a breadboard to CodeX. • I can connect external peripherals to the breadboard. • I can use sonar to detect an object. • I can convert sensor time data to a distance in centimeters. • I can create a warning system for the Rover.	**Key Concepts** • The types of components that can connect to the CodeX are greatly expanded with a breadboard. • A sonar can detect an object and how far away it is. • Use a well-known formula for distance to calculate the distance to an object. • The CodeX timer can be used in several types of applications.
Assessment Opportunities • *Check for Understanding* in CodeSpace-(2). • Write instructions, or draw a diagram, for connecting a sensor on a breadboard. • Explain how sonar works. • Exit ticket - Write pseudocode for setting up an alarm system using an if/elif/else statement. • Submit / check the **ExploreSurface** program. • Journal entry on their learning experience.	**Success Criteria** ☐ Connect the breadboard to the CodeX ☐ Use jumper wires for the sensor's power, ground and input/output. ☐ Interpret data from the sonar sensor by calculating the distance to an object. ☐ Create two simple circuits using an LED and resistor. ☐ Create a warning system for the Rover based on the distance to a harmful object. ☐ Add a function to "power down" the Rover.

Vocabulary

- **Breadboard:** A plastic board with tiny holes for inserting electronic components to build a circuit.
- **Sonar:** Short for "sound navigation and ranging"; it is a method of detecting and locating objects by using reflected sound waves.
- **Ultrasonic Sensor:** A peripheral that uses sonar to detect an object and the distance to the object.
- **Terminal Strip:** A column of tiny holes on a breadboard that are electrically connected together.
- **Jumper Wires:** Wires with connector pins at each end; used for connecting items on a breadboard.
- **Resistors:** Electronic components that limit the amount of current that passes through them. They are used with other peripherals to keep them from being damaged.

New Python Code

`import pulseio`	Import the pulse in/out library for the ultrasonic sensor's echo pin to receive a sound wave (pulse)
`pulseio.PulseIn(exp.GPIO0)`	Used to set up an input peripheral that receives a pulse
`trigger.value`	Set it True to turn on, and False to turn off, the sonar's digital trigger
`echo.clear()`	Clear the echo so it is ready to receive a newly transmitted signal
`echo[0]`	The return value of the echo, which is the transmission and receiving time in microseconds
`return -1`	Can be used to break a loop and return a value not typically given by a peripheral. It would be used when the loop condition may not be met, like timing out.
`display.show(pics.HAPPY)`	Displays a pre-defined bitmap image on the CodeX LCD

Real World Applications

Ultrasonic sensors are used in many real-world applications. Have students research smart objects that have ultrasonic sensors as embedded technology.

- The sensors are used in cars to support drivers with their driving tasks, like parking and nearby obstacle detection.
- Sonar is used in ships and other water transports to measure depth or detect objects on the sea bed.
- Other applications include: anti-collision detection, people detection, box sorting, bottle counting on drink filling machines, and much more.

Warning systems are very common in real-world applications. Have students discuss warning systems in their everyday lives. What data do they use for the warning?

- Examples include: warning systems in cars; low-battery warning; home alarm systems; etc.

Extensions

- Add more features to the “power down” function, like a beep or song.
- Use additional CodeX features to enhance the warning system, such as the NeoPixels.
- Add other sensors to the CodeX, like the temperature sensor or sound sensor. Add their data to the warning system (too hot, too cold, explosion, etc.)
- Add the 360 servo to the CodeX to simulate the wheels. Have the servo go full speed when no object is detected, but slower speeds for the warning and alert. As an additional challenge, program a sequence of backing up and going in a different direction when alerted.
- Once landed, use the CodeX’s ambient light sensor to adjust solar panels.

Cross-Curricular

- **LANGUAGE ARTS:** Have students write a short story about a Rover or autonomous robot.
- **MATH:** Use the formula D=RxT to make predictions. Select a distance and predict the time for a signal to get there and back. Then use the sonar sensor to check your predictions.
- **MATH:** Use the project as a way to introduce D=RxT. Then use the formula for different rates, like the speed of a car or train. Compare the results and make a graph.
- **PHYSICS:** Study the science of the ultrasonic sensor and sonar.
- **SCIENCE:** What is the speed of sound in different mediums, like water, an electric circuit, through a pillow, etc.
- **SCIENCE:** Think about breaking distances and friction. At what distance must the Rover start to break in order to avoid a collision? Then select an alert distance that will keep the Rover safe based on your calculations.
- **BIOLOGY:** Explore how bats, and other animals, use sonar for navigation.
- **SOCIAL STUDIES:** Research when sonar was invented and all the ways people have used it. What did people use for navigation before sonar?

Appendix A: Required Resources

Computer Resources

Each student will need:

- A computer with the Chrome web browser.
- Chromebooks work great – just make sure they are up to date.
- Windows 10 or Windows 11 will work with no additional drivers needed.
- A current Mac OS will also work with no additional drivers needed.
- A USB port is used to connect and program the CodeX. The CodeX comes with a USB to USB-C cable. If your laptop or computer has any other configuration, you will need a cable that has USB-C on one end.

Software Resources

- The interactive textbook and text editor is web-based. Make sure the website is not blocked.
- An email is required for signing in and saving work. It can be a gmail account, but any email will work.
- A per device license is needed to access the curriculum.

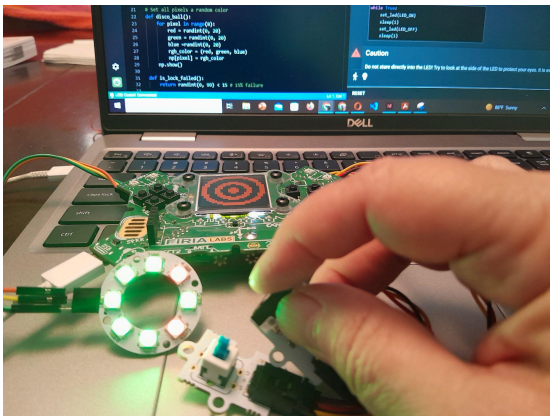
Physical Resources

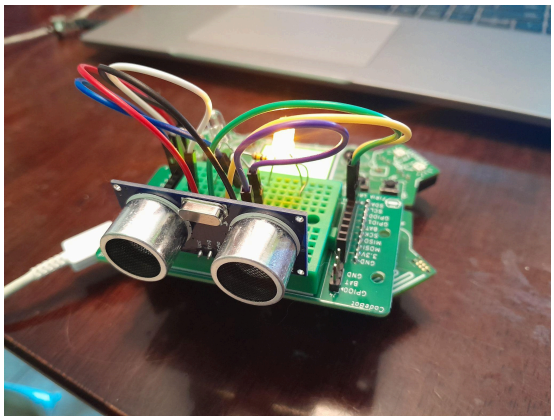
The missions can be completed by individual students or student pairs utilizing pair programming. It is possible to share a kit with more than one student or student pair, but that is not recommended. Each student or student pair will still need a CodeX and license for the curriculum.

- The Lift-Off! Kit comes with all the peripherals needed for the projects.
- The Kit comes with all the cables needed for the projects.
- Each Kit needs a CodeX
- Materials needed but not included:
    - A phillips screwdriver (Mission 8)
    - A ruler that measures millimeters (Mission 8)
    - A small screwdriver (Mission 9)
    - Two large plastic cups (Mission 9)
    - Water and soil (Mission 9)
    - (optional) 4 AA batteries for the CodeX

Notes

- When the CodeX is plugged into a computer, it will appear as a USB mass storage device, similar to a flash drive. This is not required for normal classroom use. So don't worry if your school has a policy preventing flash drives. You just close the pop-up window and continue.
- Occasionally Firia Labs will provide a software update that requires updating the core software on the CodeX. At those times you will need the flash drive feature to update the software, so you will need to use a computer with USB drive access. Often a teacher's computer is used to update all the CodeX.





Appendix B: Our Approach

Physical Computing and CodeSpace: a web-based professional-learning platform

Hardware brings code to life! Our versatile physical computing devices and peripherals get students excited about code. Our CodeSpace learning environment enables them to step up to computer science with real-world text-based Python coding. We include ready-to-teach standards-aligned curriculum with hands-on projects that motivate students.

While there are some great online coding educational programs, we think our approach helps reach a broader range of students. Our approach:

- Gets students focused "off-screen," programming with physical hardware that connects and interacts independently of their computers.
- Teaches a real, professional programming language. Even younger students appreciate that you can make real money with these exact skills. If they can read, and they can type, they can code in text-based Python.
- Gives students the tools to create *anything* they can imagine. Beyond projects and curriculum, we give students a full-fledged software development environment. These are professional-strength tools for writing code. Instead of a game-playing environment, students can "win with code" through engaging hands-on projects and their own creativity.

Project Based Motivation

Students may wonder why they are learning to code. We all find that knowledge tastes so much better when you're hungry for it! Our goal is to **motivate** students with tangible, challenging and practical **projects**...that just so happen to require coding to build. We want students to think about how they might code a given project using what they already know. Only then do we teach *just enough* coding concepts to help them get the job done. This approach gives reason and meaning to each concept, as well as relevant problem context, which helps them retain it.

Type it In

Students are often tempted to just copy and paste from lesson examples. Prior to our extensive testing of the curriculum on groups of 4th through 12th grade students, we were concerned that the typing burden might be a problem. But we were willing to risk it.

- Typing in the code forces focus, dramatically improving retention.
- Keyboarding proficiency is key to expressiveness in using a programming language.
- Mistakes in structure, grammar, punctuation, capitalization, etc. are priceless learning opportunities.

Students learn an incredible amount from their mistakes. Our goal is to provide awesome safety-nets for them, guiding them to iterate quickly through successive failed attempts to arrive at a working solution. Extensive classroom observation has convinced us that the typing burden is not a problem. Students dive right in, and they don't have to be speed typists to make great progress in coding.

Exploration and Creativity

One of the great things about coding is the expressiveness it affords. Coding is a craft that takes time to master, but with only a few basic tools you can start crafting some pretty amazing things! Before they even complete the first project, some of your students will probably be experimenting "off-script" with some ideas of their own. That's a good thing! In every lesson we list some ideas for re-mixing each project's concepts. Remember that students are learning programming skills they can use to build *any* application – from controlling a rocketship to choreographing dance moves. Nurture creativity, explore, and instill the joy of coding!

Appendix C: Teacher Resources

If you and your students are still fairly new to text-based coding, don't worry! Like other physical devices and their curriculum, we've designed the Lift-Off! With CodeX Kit and this curriculum guide to gently guide you from absolute beginner to a very comfortable level of proficiency. Remember this – Don't Panic 🙂

We understand that tackling a subject like Computer Coding can be pretty intimidating. Fear not, we've built some amazing tools to help you! As you begin this journey, know that the team at Firia Labs is here to help, too. If you run into any problems, just let us know and we'll get you back on track.

Classroom Preparation

Writing code can be like literary writing. Like developing writing skills requires individual practice, learning to code requires students to compose and test their work individually. They need to make their own mistakes and struggle through correcting them.

There is also a place for pair programming and collaboration in the coding classroom. Such practices foster knowledge sharing, collective code ownership and code review "on the go". It also gives students a chance to communicate about what they are learning and reflect on their practices. It builds confidence and keeps students focused on the taks. Pair programming can result in better quality work with less errors, and keeps teams "in the flow".

You may need to think about a balance between independent work and pair programming to give your students the best opportunities to succeed and truly engage in and enjoy programming.

Daily Routine

We recommend students work for at least 30 minutes each programming session. Adjust accordingly to your day. Because of the time it takes to set up equipment, log in to computers, and then collect equipment at the end of the learning period, it may take more time than you anticipate. Each lesson has a suggested time frame. This range accounts for completing the basics to continuing with cross-curricular lessons or extensions. Some missions may go even longer, depending on the time you have to spend in coding, the length of time for each mission, the abilities of your students, etc.

This mission pack has a lot of flexibility built-in. You should complete each mission in order, but the amount of time spent on each mission is up to you. A pacing calendar isn't provided, given the flexibility and options for the mission pack.

We recommend that students complete the Python with CodeX mission pack in advance, but it is not required.For pacing considerations, the mission pack can be:

- A once-a-week activity for an elective class or after school club
- A drop-in unit in a required or elective course
- Extended to a 9-week or 18-week course

Remixing and Extensions

Naturally students will progress at different speeds. The material is set up for independent study. You can allow students to work ahead at their own pace, or slow down as needed.

As an alternative, you can keep the class together and have "high flyers" work on extensions to the missions. Several suggestions are given for each mission. This gives students a chance to review their learning and add to their program in ways that interest them. Many students will want to experiment with what they've learned, and we offer suggestions along the way to spur this creative tinkering. Remixes are also an excellent opportunity for students to synthesize their learning and create their own projects. We highly recommend including extensions and/or remixes into your pacing calendar.

Managing a Class

Our CodeSpace learning platform makes it easy for you to create a class for your students to join, and enables you to monitor their progress.

For help and step-by-step instructions, visit: https://learn.firialabs.com/curricula/code-space

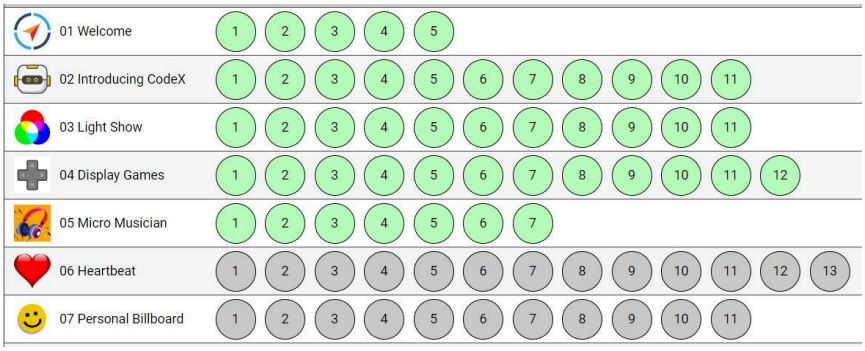
If you need assistance for anything, please send an email to: support@firialabs.com

Here are the basics of the CodeSpace Teacher Dashboard

- Log in to CodeSpace and from HELP, select CLASS DASHBOARD
- Once you are in the dashboard, click + in the green bar, top right corner, to add a class.
- Assign each class a name, and allow members to join with a join code.
- You can assign Google Classroom as your LMS.
- After the class is created, you can edit the class, get a join code, disable joining, etc.
- You can delete a student using the “remove” function.
- Students go to CodeSpace and click the SELECT CLASS button.
- They can click the JOIN CLASS button and enter their join code for your class.
- The class will be activated and they are ready to start working!
- In the dashboard, you can see student progress, as a whole class and individually.

Class dashboard

Individual progress



Appendix D: Vocabulary by Mission

Mission 1 – Welcome to Lift-Off! Peripherals	
Peripheral	A device that interacts with the CPU (common peripherals are LED lights, display screen, buttons, mouse, keyboard, and printer)
Function	A named chunk of code you can run anytime just by calling its name; also called a procedure
Parameter	A local variable in a function that receives a value passed into the function when it is called; information the function needs to complete its task
Argument	Passing data to functions (information a function uses to complete its task)
Variable	A name you assign to some data used in code instead of the literal, or actual, values
Constant	A name for a value that doesn't change during program execution
Mission 2 – Lift Off	
Abstraction	The process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics
Algorithm	A sequence of steps for completing a task (step by step process)
Branching	Decision points in code; a condition
Mission 3 – Conserve Energy	
Analog	A peripheral with a range of integer values – from 0 (off) to 2^16 - 1 (full power)
Digital	A binary peripheral with two states – True or False
ADC	Analog to digital converter – converts an analog measurement to a finite digital value. For CodeX, which is a 16-bit microcontroller, the digital values range from 0 to 2^16-1 (65,535)
Pulse-Width Modulation	Analog measurement where on/off pulses are sent at a constant rate, determined by the duty cycle and frequency (or analog period)
Mission 4 – Hatch Lock	
RGB	Red, Green, Blue; the colors that make up a single pixel on the screen
Pixel	Picture element; tiny dots used to compose images and text on a digital screen
Tuple	An *immutable* sequence of items that you can access with an *index*, or a list with values that don't change. A read-only version of a list.
Index	A common method for referencing the elements in a list, tuple or string using a number
Mission 5 - Alert System	
Duty Cycle	The percentage of time power is ON during pulse-width modulation.
Frequency	The analog period, or how rapidly the device pulses during pulse-width modulation.
REPL	"Read, evaluate, print loop" command line that enables print statement output.

Simulation	Code that builds a *model* of something; lets you play with that model to explore "virtual" situations.
Threshold	A specific limit or point that must be met or exceeded in order for something to occur.
Mission 6 - Life Support	
Servo	A DC motor with a controller circuit, internal feedback mechanism and gearbox. The 360 continuous servo goes in both directions at different speeds. The 180 positional servo turns in either direction to a specific angle and holds that position.
Finite-State Machine	The status of a system with transitions. With this system, your program can only be in one of a known set of “states” at any given time. Usually “state” is based on variables in your code.
State	A phase of a program. Keeping track of states helps you understand and manage your code. Each state might have its own set of conditions it is tracking.
Transition	Moving between states; the program transitions from one state to another when certain conditions are met.
Mission 7 – Solar Tracking	
Photoresistor	A sensor that changes its resistance when light shines on it. A high intensity of light causes less resistance, and less light causes more resistance.
Bouncing	When a digital input registers multiple times instead of once, like a button press.
Mission 8 – Prepare Lander	
Pull	A property that can be changed when setting up an input peripheral that determines the default value of a pin when nothing is connected. The pull can be set to “up” to move a weak pull toward 3 volts.
Mission 9 – Automatic Gardner	
NC / NO	Terminals on the relay used for connecting peripherals. NO = normally open; this terminal is the most common one used.
Priming	The process of removing air from pump lines.
Conductivity	The ability of a material to conduct electricity. In this mission, it is water. More water in the soil means more conductivity.
Mission 10 – Exploring the Surface	
Breadboard	A plastic board with tiny holes for inserting electronic components to build a circuit.
Sonar	Short for “sound navigation and ranging”; it is a method of detecting and locating objects by using reflected sound waves.
Ultrasonic Sensor	A peripheral that uses sonar to detect an object and the distance to the object.
Terminal Strip	A column of tiny holes on a breadboard that are electrically connected together.
Jumper Wires	Wires with connector pins at each end; used for connecting items on a breadboard.
Resistors	Electronic components that limit the amount of current that passes through them. They are used with other peripherals to keep them from being damaged.

Appendix E: Python Code by Mission

Mission 1 – Welcome to Lift-Off! Peripherals	
`exp.digital_out(exp.PORT0)`	Used to set up a digital output peripheral (LED)
`led.value`	The property of the LED peripheral used to turn on/off the light
Mission 2 – Lift Off	
`exp.digital_in(exp.PORT0)`	Used to set up a digital input peripheral (button, switch)
`button.value`	Returns the button's position: True (not pressed) or False (pressed)
`switch.value`	Returns the switch's position: True (out) or False (in)
`display.fill()`	Fills the CodeX LCD screen with a pre-defined or RGB color
Mission 3 – Conserve Energy	
`exp.pwm_out(exp.PORT0)`	Used to set up a peripheral with PWM; requires a duty cycle and frequency
`exp.analog_in(exp.PORT1)`	Used to set up an analog input peripheral (potentiometer)
`led.duty_cycle`	Determines power to the LED (higher integer is a brighter light)
`sleep_ms()`	Delays program execution in milliseconds
`time.ticks()`	Returns the current clock time (elapsed time since last reboot)
`motion_sensor.value`	Returns the motion sensor value: True (detected) or False (not detected)
`potentiometer.value`	Returns an integer as the potentiometer knob's position
Mission 4 – Hatch Lock	
`neopixel.NeoPixel(exp.PORT0, 8)`	Sets up the NeoPixel ring; indicates the port and number of LEDs
`power.enable_periph_vcc(True)`	Turns on extra power to the NeoPixel ring
`randint(low, high)`	Returns a random integer between and including low and high
`(red, green, blue)`	A tuple with three items; used for RGB colors
`np[pixel]`	Accessing a single item [pixel] in a list np
`return`	Returns a value to the statement calling the function
Mission 5 - Alert System	
`LED_ON = 2**16 // 2`	Maximum duty-cycle for an LED using PWM
`LED_OFF = 0`	Minimum duty_cycle for an LED using PWM

`led = exp.pwm_out(exp.PORT0, frequency=2)`	LED using PWM for blinking light
`degrees_c = raw_temp*0.004577 - 50`	Conversion of raw temp to degrees Celsius
`avg_sound = avg*(1-WEIGHT) + new_val*WEIGHT`	Average sound calculation using exponential moving average
`import soundlib`	Import the sound library to add non-blocking sound functions
`siren = soundmaker.get_tone("violin")`	Sets up a variable for the sound
`siren.set_pitch()`	Sets the pitch tone at the given frequency
`siren.play()`	Plays the sound at the set pitch
`siren.glide()`	A non-blocking way to ramp the pitch from the current setting to a new setting over a specified amount of time
`siren.stop()`	Stops playing the tone
`global temp_limit`	Allows for updating a global while being used in a function.
`print("Temp:", degree_c)`	The print() function displays text on the console panel
Mission 6 - Life Support	
`servo = exp.pwm_out(exp.PORT0, frequency=20)`	Set up a 360 servo using PWM
`//` `example: CYCLE * percent//100`	Division that returns only the integer and no decimal (no rounding)

`if state == "maintenance":`
`  if switch.value == POWER_ON:`
`  state = "active"`
`  fan.duty_cycle = set_servo(FORWARD)`
	Nested if statements. The first if statement is checked. If true, it will check the second if statement. If false, the block of code is skipped.
	
	This example also shows **transitioning** to a different **state**.

Mission 7 – Solar Tracking	
`servo.duty_cycle = 0`	Stop a 180 servo
`state = 'morning'`	Define and initialize a variable for the state. Also, single quotes can be used for strings (see hint in Objective 4).
Mission 8 – Prepare Lander	

`def set_lighting(rgb_color):`
`  for pixel in range(8):`
`  np[pixel] = rgb_color`
	Set all pixels in the NeoPixel ring to one color. This was first used in Mission 4.

`sensor = exp.digital_in(exp.PORT1, pull=digitalio.Pull.UP)`	Change the pull property to "up" so the weak signal is pulled toward 3 volts, or "high".

Mission 9 – Automatic Gardner	
`buttons.was_pressed(BTN_A)`	Returns True if the button was pressed since the last check; otherwise it returns False
`relay.value`	The property used to turn on/off the relay (True or False)
Mission 10 – Exploring the Surface	
`import pulseio`	Import the pulse in/out library for the ultrasonic sensor's echo pin to receive a sound wave (pulse)
`pulseio.PulseIn(exp.GPIO0)`	Used to set up an input peripheral that receives a pulse
`trigger.value`	Set it True to turn on, and False to turn off, the sonar's digital trigger
`echo.clear()`	Clear the echo so it is ready to receive a newly transmitted signal
`echo[0]`	The return value of the echo, which is the transmission and receiving time in microseconds
`return -1`	Can be used to break a loop and return a value not typically given by a peripheral. It would be used when the loop condition may not be met, like timing out.
`display.show(pics.HAPPY)`	Displays a pre-defined bitmap image on the CodeX LCD