MISSION 9: All Systems Go! **Lesson 3 (Objectives 8-12)**	**Time Frame: 45-50 minutes**
Project Goal: Students will use input sensors to monitor physical orientation. **Learning Targets** • I can use system sensors to monitor physical orientation. • I can use physical orientation to detect motion.	**Key Concepts** • The accelerometer detects orientation in three dimensions. The CodeBot can be programmed to act on conditions based on its orientation. • The accelerometer can also be used to detect motion in the CodeBot. Any slight movement can be detected and used for an alarm.
Assessment Opportunities • Mission 9 Lesson 3 Mission Log • Submit completed program ***AccelTest*** • Submit completed program ***GuardBot*** • Mission 9 Obj. 8-12 Review Kahoot!	**Success Criteria** ☐ Print the 3-axis values from the accelerometer ☐ Write basic code that keeps CodeBot pointed up ☐ Improve the basic code for better control using proportional rotation speed ☐ Detect motion on the X-axis ☐ Detect motion on all 3-axes
Teacher Materials in Learning Portal • Mission 9 Lesson 3 Slides • Mission 9 Lesson 3 Mission Log • Mission 9 Lesson 3 Mission Log Answer Key	**Additional Resources** • Mission 9 Obj. 8-12 Review Kahoot! • AccelTest_obj9 sample code (in learning portal) • AccelTest_obj10 sample code (in learning portal) • GuardBot sample code (in learning portal)

Vocabulary

- **Orientation:** The relative position of something.
- **Accelerometer:** A tiny chip that measures the force of acceleration in 3 directions: x, y and z.
- **Navigation:** Knowing where you are and planning and following a route for where you want to be.
- **Oscillate:** Move or swing back and forth at a steady speed.
- **Proportional:** Change at the same rate.
- **Incline:** Sloping upward.

New Python Code

`accel.dump_axes`	Prints the 3-axis values to the console
`x, y, z = accel.read()`	Reads the current axis values and returns a tuple of integers, ranging from -32767 to +32768
`now = accel.read()`	Read the accelerometer and assign all three values to a tuple.
`now[0]`	Access the X value of the accelerometer reading.
`before = now`	Assign the same tuple (now) to a new variable (before).
`dx = now[0] - before[0]`	Calculate the difference between current reading and previous reading.
`if abs(dx) > SENS:` `  alarm()`	If the difference between readings is more than the sensitivity, sound an alarm.

Real World Applications

Many electronic devices use an accelerometer for orientation and navigation. Some examples are:

- Cell phones
- Smart watches
- Game controllers
- Vehicles
- And much more!

Teacher Notes:

- This lesson follows the instructions in CodeSpace fairly closely. It is chunked into smaller bits of information and simplified for clarity.
- The code in this lesson is similar to CodeTrek. It is simplified a little for ease of typing, and it is organized in a way similar to former missions. All goals will be met.
- Objectives 9 and 10 require an inclined surface. It can be anything that is sturdy and supports a CodeBot. A yearbook, shoebox, cardboard box, anything handy will work. A textbook is great for making the incline. I recommend a small surface that can be picked up and tilted for Obj. 10.

Extensions / Cross-Curricular:

- Fall Detector. Use the accelerometer to detect if the ‘bot is falling.
- Bump Bot. Move the ‘bot forward until it detects an impact. Then rotate a random amount and go again.
- **SCIENCE:** Have a lesson on the accelerometer. What are its parts, and how does it work? Research devices that use an accelerometer.
- **SCIENCE:** This lesson uses gravity. Have a lesson on gravity, its measurement, etc.
- **MATH:** This lesson finds the difference between two readings. The variable dx is used for delta x. Learn about Δ (delta) and how it is used in math.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

Preparing for the lesson:

- Look through the slides. Decide what materials you want to use for presenting the lesson. The slides can be converted to Google Slides. They can be projected on a large screen.
- Be familiar with the mission log assignment and the questions they will answer. Prepare the assignment to give through your LMS.
- This lesson requires a sturdy surface on an incline. You can use a large book (like a yearbook), a shoebox, a piece of cardboard, etc. Use something to prop up one end as an incline. The surface doesn’t need to be big; the ‘bot isn’t moving up it, just trying to stay pointed up.
- If you have a word wall, or another form of vocabulary presentation, prepare the new terms.

Lesson Tips and Tricks:

💡 **Teaching tip:**

You can use a variety of discussion strategies to get the most engagement from your students. For example, you can have students write their answers before asking anyone for an answer. You can use one of many think-pair-share methods.

🧑‍🤝‍🧑 **Pre-Mission Warm-up:** -- slide 2

Students can write in their log first and then share, or discuss first and then write in their log. The warm-up question previews the lesson. Students can share their answers, or compare with each other.

- Question: What do you know about gravity?
- Question: What are the three dimensions in our world?

💻 Mission 9 Lesson 3 Activities:

The Chrome browser works best, but other browsers also support CodeSpace. Each student will complete a Mission Log. Students could work in pairs through the lesson, or they can work individually.

💡 **Teaching tip: Mission Introduction** -- slides 3-4
This mission is divided up into three lessons. This lesson completes the last two goals. Students answer one question in their mission log.

💡 **Teaching tip: Objective #8** -- slides 5-7
The accelerometer is introduced.

💡 **Teaching tip: Objective #8** -- slides 8-10
Discussion on gravity and its measurement, and how this relates to the accelerometer. Look over the slides and decide if you need to add anything.

💡 **Teaching tip: Objective #8 Activity** -- slides 11-14
Students write a simple program that prints the 3-axis values to the console. Students try CodeBot in different orientations and record the readings in the mission log.

💡 **Teaching tip: Objective #9** -- slides 15-18
Students learn about how CodeBot is oriented and how to use the accelerometer readings to point the ‘bot up.

💡 **Teaching tip: Objective #9 Activity** -- slides 19-24
Students modify the program. They use CodeBot’s orientation to keep the nose pointed up. Students need a sturdy surface on an incline. It doesn’t have to be a very steep incline or a very big surface. Just big enough for CodeBot to fit on it nicely. The bot will do a lot of wiggling and end up at the bottom, but no forward movement is made. Students answer one question in the mission log.

NOTE: Students add the safety feature from Mission 4 (wait for a button press before enabling the motors). Review as needed.

💡 **Teaching tip: Objective #10** -- slides 25-27
Students learn how to improve their program with proportional rotation speed. This section has a bit of math. If the math looks too difficult, you can skip the slides and just do the program, or even skip the objective and go on to the next one. Decide what will work best for your students.

💡 **Teaching tip: Objective #10 Activity** -- slides 28-31
Students modify the program to include proportional rotation speed. The speed needs to be around 50 or higher to see any significant change. If the surface is small enough, students can pick it up and tilt it in several directions. The ‘bot should always work to maintain its nose pointed up. They answer one question in the mission log.

💡 **Teaching tip: Quiz** -- slide 32
The quiz has four questions about reading the accelerometer. Questions are listed below.

💡 **Teaching tip: Objective #11** -- slide 33
Students are given an application for using the accelerometer: the guard bot.

💡 **Teaching tip: Objective #11 Algorithm** -- slides 34-38
The algorithm for the program is given. The program involves a couple of new concepts, so each step is discussed on the slides, and the exact programming code is given.

💡 **Teaching tip: Objective #11 Activity** -- slides 39-42
Students start a new program for their guard bot. They will define a function for sounding the alarm. They can make the alarm anyway they want. Sample code is provided, but they don’t have to program it the same way.

💡 **Teaching tip: Objective #12** -- slide 43
The program is expanded to include all three dimensions. This is a very short objective.

💡 **Teaching tip: Objective #12 Activity** -- slides 44-48
Students improve their program to include all three dimensions. They also add parameters to the alarm() function so they can display the differences in each.

💡 **Teaching tip: Extensions** – slides 49-50
Two extension ideas are given. One is listed as medium and the other is spicy. They are optional. The solutions for the extensions are not available.

💡 **Teaching tip: Post-Mission** – slides 51-52
These slides sum up the mission by applying it to the real world.

Optional: 🗝️ Mission 9 Obj 8-12 Kahoot! Review. A review Kahoot! Is available for these four objectives.

👫 **Post-Mission Reflection:**

The post-mission reflection asks students two reflection questions. Answers can vary widely.

You can use a cross-curricular activity for a post-mission activity.
End by collecting the Mission 9 Lesson 3 Log.

SUCCESS CRITERIA:

- ❑ Print the 3-axis values from the accelerometer
- ❑ Write basic code that keeps CodeBot pointed up
- ❑ Improve the basic code for better control using proportional rotation speed
- ❑ Detect motion on the X-axis
- ❑ Detect motion on all 3-axes

QUIZ after Objective 10 (see next page)

Checkpoint

Approximately what value would the **Y-axis** have if you pointed CodeBot toward the sky?

(Proximity sensors pointed up)

- [ ] -1.0
- [ ] 0
- [ ] +16,384
- [x] -16,383

The **X-axis** value would be *zero* if your 'bot was facing straight up.

- In the code above, what would the speed `rot_spd` be in that case?

- [ ] -SPEED
- [x] 0
- [ ] +SPEED
- [ ] 50

What would the value of `y` be after the following assignment statement?

```
x, y, z = (30, 20, 40)
```

- [ ] (30, 20)
- [ ] (30, 20, 40)
- [ ] 40
- [x] 20
- [ ] 30

The `accel.read()` function returns a tuple with **3** integers (x, y, z).

If you wrote: `vals = accel.read()`, which *axis* would `vals[1]` refer to?

- [x] Y-axis
- [ ] Z-axis
- [ ] X-axis