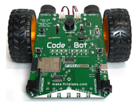


MISSION 9: All Systems Go!
Lesson 2 (Objectives 5-7)

Time Frame: 45-50 minutes

Project Goal: Students will use system sensors to monitor system temperature and display unacceptable temperatures with a UI.

Learning Targets
- I can use system sensors to read system temperature in either Celsius or Fahrenheit.
- I can find the average of several temperatures.
- I can create a UI that uses LEDs to show an alert if the average temperature is too high or too low.

Key Concepts
- The ‘bot can measure its own system temperature.
- Just like the system data for power, the temperature data returned by the system sensors can be used to show alerts to avoid problems. The CodeBot’s LEDs can be used to show alerts.
- The system sensors can measure temperature in either Celsius or Fahrenheit.
- The CPU in the CodeBot CB2 (older model) is more exposed than the CB3 and can more easily be manipulated. At the end of the slide deck, specific instructions are given to test the code with the CB2 that doesn’t work well with the CB3. If you have a newer ‘bot, just skip those two slides.

Assessment Opportunities
- Mission 9 Lesson 2 Mission Log
- Submit completed program ***TemperatureCheck***
- Mission 9 Obj. 5-7 Review Kahoot!

Success Criteria
- ☐ Read system temperature using system sensors
- ☐ Define an empty list for temperature reading
- ☐ Get a sum of all temperatures in the list
- ☐ Return the average of temperatures in a list
- ☐ Define constants for baseline and deadband
- ☐ Turn on/off LEDs as an alert when the temperature is too high or too low

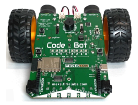
Teacher Materials in Learning Portal
- Mission 9 Lesson 2 Slides
- Mission 9 Lesson 2 Mission Log
- Mission 9 Lesson 2 Mission Log Answer Key

Additional Resources
- Mission 9 Obj. 5-7 Review Kahoot!
- TemperatureCheck_obj5 sample code
- TemperatureCheck_obj6 sample code
- TemperatureCheck_obj7 sample code
- TemperatureCheck_final sample code

Vocabulary
- **Milliseconds:** A millisecond is a thousandth of a second.
- **Ambient:** Surroundings.
- **Append:** Adding a new item to the end of a list.
- **Traverse:** Accessing each item in a list in order.
- **Baseline data:** Starting point used for comparison; original data.
- **Deadband:** In a control system, the range or band of input values where the output doesn’t change.

New Python Code

`bot_temp = system.temp_C()`	Measure temperature in Celsius.
`bot_temp = system.temp_F()`	Measure temperature in Fahrenheit.
`sleep_ms(200)`	Delay program execution for 200 milliseconds.



`samples = []`	Initialize an empty list.
`samples.append(bot_temp)`	Append (add) a new item to a list.

`while i < count:`
`    sum = sum + nlist[i]`
`    i = i + 1`
	Traverse a list using a loop to sum all the items.

`return sum / count`	Return the average without assigning the value to a variable.
`samples.clear()`	Clear a list of all items.

`BASELINE = 25.5`
`DEADBAND = 3.0`
	Define a constant.

`for i in range(5):`	A quick way to loop a block of code five times.

`if t > BASELINE + DEADBAND:`
`    leds.user(0b11111111)`
	Compare temperature for a value that is too high, above the baseline + deadband.

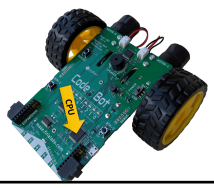
`if t < BASELINE - DEADBAND:`
`    leds.ls(0b11111)`
	Compare temperature for a value that is too low, below the baseline - deadband.

Real World Applications
You already use this kind of code daily! Many electronic devices use a temperature-control system, or thermostat to avoid overheating or overcooling. Some examples are:

- Smart thermostats
- Central heating and air conditioners
- Portable heating and/or cooling units
- Ovens, refrigerators, coffee makers, etc.
- Water heaters, vehicles, medical equipment
- And much more!

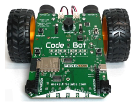
Teacher Notes:

- 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.
- Three extensions are given for the lesson. The lesson may not take an entire class period. I recommend doing the first extension together. Detailed instructions are given for the extension.
- The older CodeBot model (CB2) is different from the new model (CB3). Its CPU is more



Extensions / Cross-Curricular:

- Fix the check_baseline() function to turn off LEDs when the condition is no longer met.
- Add beeps to the alert system to indicate too high or too low temperatures.
- Add another function that reads the system temperature in Fahrenheit, and use button presses to determine which reading is executed.
- **SCIENCE:** Have a lesson on temperature, what affects temperature, and how temperature affects devices.
- **MATH:** This lesson uses averages. Review the formula for calculating an average and practice with your own data.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.



accessible. At the end of the lesson, additional instructions are given for testing if you have the CB2 model. Otherwise, skip the slides.	

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.
- Students may want to use a calculator for the Obj. 5 mission log assignment.
- If you have a word wall, or another form of vocabulary presentation, prepare the new terms.
- If you have the older CB2 model, have hot and cold options that can heat/cool the CodeBot CPU. Disposable water bottles work well and won't damage CodeBot (see slides 40-41).

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 temperature?

Mission 9 Lesson 2 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 second goal. Students answer one question in their mission log.

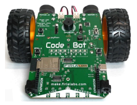
Teaching tip: Objective #5 -- slides 5-7
System temperature is introduced. Students learn about two functions they can use to read the temperature.

The sleep_ms() function is introduced and compared to the sleep() function.

Teaching tip: Objective #5 Activity -- slides 8-11
Students write a short program and observe the results on the console panel. The temperature readings will scroll quickly on the console. Students only need to run the program for a few seconds. Then they stop the code and record the five last readings.

Students calculate the average of the last five readings. If students don't know how to do this, you will need to teach it to them. They can use a calculator. In the next objective, the computer will find the average, so this is a good exercise for understanding what the computer is doing, and comparing the speed of the calculation.

Teaching tip: Objective #6 -- slide 12
This slide discusses the need for a moving average function.



💡 **Teaching tip: Objective #6 Algorithm** -- slides 13-17
Algorithm for the moving average algorithm. It is broken down step-by-step, and the code is shown for each step. Students can refer back to the algorithm when needed.

The algorithm uses a list, which was introduced in Mission 6. Review lists as needed.

💡 **Teaching tip: Objective #6 Activity** -- slides 18-22
Students implement the moving average algorithm. They test the code as written, and record results in the mission log. Then they change the length of the list and test again. They test for a total of four different list lengths.

💡 **Teaching tip: Objective #7** -- slides 23-25
Students apply the system controls for temperature by creating a thermostat (real-world application).

💡 **Teaching tip: Objective #7 Algorithm** -- slide 26
Algorithm for the thermostat, a user interface for the system temperature.

💡 **Teaching tip: Objective #7 Activity** -- slides 27-30
Students implement the algorithm for the user interface by defining two constants and a new function.

NOTE: The first extension is optional but highly recommended. Detailed instructions and sample code are included.

💡 **Teaching tip: Extension (mild)** -- slides 31-39
At this point the program is complete and all the mission goals have been met. However, students haven't been able to really test the user interface. This extension guides them through adapting the code to try all ranges of temperatures. They will discover that the check_baseline() function has a bug – the LEDs don't turn off. The slides continue with the bug fix so they have a nicely working final program. The sample code for this extension can be found in the learning portal.

💡 **Teaching tip: Extension (mild) for CB2 only** -- slides 40-41
If you have the older model CodeBot CB2, students can test the system temperature in real time with hot and cold objects. I recommend using empty water bottles. Fill one with some water (not full) and heat in a microwave. Fill another with cold water or freeze it into ice. Place the objects (one at a time) close to or just above the CPU and watch the temperatures change in real time.

💡 **Teaching tip: Extensions** – slides 42-43
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** – slide 44
This slides sums up the mission by applying it to the real world.

Optional: 🗝️ Mission 9 Obj 5-7 Kahoot! Review. A review Kahoot! Is available for these four objectives.

🧑‍🤝‍🧑 **Post-Mission Reflection:**

The post-mission reflection asks students to think about how a temperature-controlled system can be used in other CodeBot programs. Answers can vary widely.

You can use a cross-curricular activity for a post-mission activity.
End by collecting the Mission 9 Lesson 2 Log.

SUCCESS CRITERIA:

❏ Read system temperature using system sensors
❏ Define an empty list for temperature reading
❏ Get a sum of all temperatures in the list
❏ Return the average of temperatures in a list
❏ Define constants for baseline and deadband
❏ Turn on/off LEDs as an alert when the temperature is too high or too low