Mission 6: Navigate

Time Frame: 120-180 minutes

Mission Goal: Students will program CodeAIR to fly autonomously using sensor data.

Learning Targets

- I can explore positioning systems with the flow sensor for x, y tracking.
- I can observe and analyze flow sensor accuracy by flying CodeAIR in a square.
- I can conduct a battery check to ensure safe and sustained flight.
- I can customize selectable operations to control code behavior during runtime.
- I can experiment with flight parameters including height, distance and velocity.

Key Concepts

- The flow sensor is an optical device that can discern patterns on the ground and report movement in two directions.
- The sensor tracks motion by summing up the changes in x and y over time.
- You can use an f-string to determine how information is printed.
- Data from multiple sensors is needed for accurate flight control.
- CodeAIR's battery is charging whenever it is plugged in.
- The 8 LEDs can use binary patterns to display integers between 0 and 255.
- Handle exceptions in code using a *try* block.

Assessment Opportunities

- Quiz after Objective 2
- Quiz after Objective 4
- Quiz after Objective 6
- Complete the program *FlowTracker*
- Complete the program *SquareUp*
- Complete the program *SquareTurns*
- Complete the program *BattTest*
- Complete the program *utility.py*
- Complete the program *RouteSelect*
- Mission 6 Assignment
- Mission 6 Review questions

Success Criteria

- ☐ Use flow sensor data to track position
- ☐ Fly CodeAIR in a square
- ☐ Perform a battery check
- ☐ Create a user interface using binary patterns
- ☐ Handle exceptions using a *try* block
- ☐ *RouteSelect* works correctly and runs without errors or bugs
- ☐ Complete Mission 6 Assignment

Standards

CSTA Standards Grades 9-10	CSTA Standards Grades 11-12	AI4K12 Standards Grades 9-12
3A-CS-01	3B-CS-02	1-A-ii
3A-CS-03	3B-DA-06	1-B-i
3A-DA-09	3B-DA-07	1-C-ii
3A-AP-13	3B-AP-10	
3A-AP-14	3B-AP-14	
3A-AP-16	3B-AP-15	
3A-AP-17	3B-AP-16	
3A-AP-18	3B-AP-21	
3A-AP-21	3B-AP-22	
3A-IC-26	3B-AP-23	

Student Materials

- Laptop/computer with Chrome browser
- CodeAIR drone and USB cable
- Metric tap measure or ruler
- CodeAIR Mission 6 Assignment
- Optional: Mission 6 Flight Data (spreadsheet)
- CodeAIR Flying Guide

Teacher Resources

- CodeAIR Mission 6 Assignment Answers
- CodeAIR Mission 6 Review Questions
- CodeAIR Flying Guide

Vocabulary

Positioning systems	A system for determining the position of an object in space.
Flow sensor	A sensor used to track horizontal movement across a surface; essential for stable hover and precise navigation.
Delta	A change in position, symbol from the Greek letter Δ and used in math and science to represent change.
String	A data type that is a sequence of characters all strung together. Can be numbers, letters, spaces, whatever!
Format string	A template for printing a string using replacement fields that are designated with {curly braces}. This allows actual arguments to be inserted into the template.
MotionCommander Interface	An interface between Python code and the flight controller that provides a high-level flight control interface and uses onboard sensors to maintain stability.
Sensor fusion	When data from multiple sensors is combined. For example, combining altitude data from the laser ranger with the data from the flow sensor.
Under load	When a battery is powering peripherals, like CodeAIR’s motors, it is under load.
Binary numbers	How computers deal with digits. Two states (on and off) are represented with 1 and 0.
Bit	A single binary digit (1 or 0)
Byte	An 8-bit number
Selectable operations UI	A user interface that uses one button to scroll through a menu and another button to confirm the current selection, and then start the action.
Exceptions	Errors that might happen during your program execution.
External positioning system	A positioning system that uses something outside the drone to determine location, such as GPS or a fixed-location beacon.

New Python Code

`dx, dy = get_data(FLOW)`	Read data from the flow sensor; returns the change in x direction and change in y direction
`print(x, y)`	A simple print statement that converts data to strings and displays them on the console
`print(“Flow Sensor Output”)`	Print a string text on the console
`print(f”x={x}, y={y}”)`	F-string with replacement fields in curly braces
`abs(x)`	Returns the absolute value of x
`vbatt = power.battery_voltage(10)`	Read battery voltage, average 10 samples
`amps = power.charger_current()`	Read charging current when connected with USB
`usb_connected = power.is_usb()`	Returns True if currently powered by USB

`value = 0b1001`	Set the value to 9 using binary
`leds.set_mask(255, 50)`	Set BYTE LEDs to 255 (on) with brightness = 50
`leds.set_mask(0b10101010, 50)`	Set BYTE LEDs using binary
`If __name__ == '__main__':`	Detects when this program is being run as the “main program” instead of an import
`try:`	A block of code that executes when no error occurs, or until an error occurs.
`except:`	A block of code that lets your program respond to an error without crashing.

Teacher Notes

- The last objective requires students to create a chart and log the results of several test flights. The assignment document has a chart they can use. Alternatively, students can use a spreadsheet (one is provided) and use the spreadsheet for more extensive data analysis.
- This mission involves several programs, and each one can be used for experimentation. Don’t rush, and give students plenty of time to try their code with different values and in different environments. You could easily spend a class period on almost every single objective.
- Objective #4 has some important instructions after the goal is met. Students need to add their battery check to *safety.py* and then run the code to reinstall it on CodeAIR. Don’t let students skip this part!
- The test flights during the Objectives will need plenty of unobstructed room. Clear out as much furniture as you can, or use hallways. Also check the flooring. Use flooring with a pattern for best results.
- Extensions and cross-curricular projects are included to enhance the concepts in the mission. You can use the extensions to extend students’ programming experience. A remix is planned after Mission 7.

Extensions

- Fly the drone in a different shape: triangle, hexagon, etc.
- Create a more elaborate battery check that shows the charging amp when plugged in or the battery charge when disconnected.
- Have students do a code review. Pick any of the six programs to review.
- Write algorithms for the functions, or explain what each line of code does. Discuss parameters and arguments.

Cross-Curricular

- **MATH:** The code uses built-in math functions like abs() and min(). Discuss other built-in math functions that could be used in code. Practice using them in practical problems.
- **MATH:** Objective #5 introduces binary numbers. Practice converting binary and decimal numbers. Explore other number systems besides decimal and binary.
- **MATH:** Several objectives involve testing the drone’s flight capabilities. The last objective uses a chart to document the tests. Do an in-depth analysis of the data. Create charts for other objectives and analyze the data.
- **LANGUAGE ARTS:** Make a list of troubleshooting strategies used to identify and fix errors.