1ST MAKER SPACE
Design.
Build.
Sustain.

Frog Microcontroller Trainer (FMT): Mind the Dog

Length of Time: 45–60 Minutes

Recommended for Developmental

Contents

Overview:

Mind the Dog solves a real household problem: tracking when multiple dogs were last let outside. The program runs on the 1ST Maker Space Frog (Arduino UNO R4-based board) and displays two independent elapsed-time timers on the OLED screen, one per dog. Pressing each button resets the corresponding timer. Cheek LEDs provide a visual heartbeat, confirming the device is running. The central concept is non-blocking timing using millis() rather than delay().

Objectives:

Students will be able to:

Explain what an elapsed-time timer is and how it is implemented using millis() rather than delay().

Describe the difference between polling/non-blocking and blocking (delay-based) approaches to timing in an embedded program.

Identify how a state variable (timerStartTime) encodes the passage of time without a dedicated clock chip.

Explain the role of each function and how they cooperate through a simple main loop.

Modify the program to change display labels, add a third timer, or adjust the heartbeat rate.

Computer Science Teacher Association Standards:

- 3A-AP-16- Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
- 3A-AP-17- Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

Materials:

- Frog Microcontroller Trainer from 1st Maker Space
- USB Power Cord
- Arduino IDE
- Mind_The_Dog.ino sketch loaded onto each Frog, along with the 1ST_Maker_Frog library
- PC or Mac
- Chromebooks if using Arduino Create for Education App

Preparation:

- Confirm the Frog boards, library, and Mind_The_Dog sketch compile and upload successfully.
- Run the program yourself: press SW1 and SW2 to observe timer resets and verify the LED heartbeat.
- Decide which extensions (e.g., third timer, seconds display, NeoPixel color threshold) are appropriate for your students' level.
- Print or distribute the student handouts.

Background Information:

1. Non-blocking timing with millis()

The most important concept in this program is the difference between blocking time measurement (using delay()) and non-blocking time measurement (using millis()). A program built with delay() is frozen while it waits — it cannot read buttons, update the display, or do anything else during that time. Mind the Dog never calls delay() during normal operation; the only delay() calls are the 10 ms loop throttle and the 250 ms debounce hold-off after a button press, both of which are short enough to be imperceptible. Instead, the program records the time at which each timer was started (timerOneStartTime, timerTwoStartTime) as a millis() snapshot. At any point in the loop, the elapsed time is simply the current millis() value minus the snapshot. This approach is sometimes called timestamp-based timing and is the correct pattern for any embedded program that needs to do more than one thing at a time.

2. The Heartbeat Pattern

The cheek LEDs provide a visual heartbeat: one LED is lit at a time, advancing to the next position once per second. This gives users confidence that the device is running even when the display is not changing. The heartbeat uses the same timestamp pattern

as the timers — lastLedUpdateTime stores when the LED last moved, and the LED only advances when enough time has passed. The heartbeat is a useful teaching example because it is a second independent timed event running alongside the display update, both without any delay().

3. Efficient Display Updating

Redrawing an OLED display takes time and causes visible flicker if done every loop iteration. Mind the Dog only redraws when the displayed minute value has actually changed. The variables lastMinutesOne and lastMinutesTwo remember what was shown last. The constant FORCE_SCREEN_UPDATE (set to 99, a value that can never be a valid minute count) is used as a sentinel: when a button is pressed and a timer resets, lastMinutes is set to 99, guaranteeing an immediate redraw on the next loop iteration.

4. Time Scaling

The constants MILLIS_PER_MINUTE and MILLIS_PER_HOUR control the relationship between real time and displayed time. In production these are set to 60000 ms per minute and 3600000 ms per hour. During classroom demonstration, much smaller values allow the timer to run fast enough to see hours and minutes change in seconds. Set MILLIS_PER_MINUTE = 600 and MILLIS_PER_HOUR = 36000 for demo mode This is a good opportunity to discuss the concept of test scaling in engineering: changing a constant to make slow behavior fast enough to test, then restoring it for production.

5. setup()

The program is divided into four named functions in addition to setup() and loop(). Each has a single clear responsibility:

Function	Responsibility
handleInputs()	Polls both buttons. If pressed, resets the corresponding timer start time, sets the forced-redraw sentinel, and sounds a confirmation tone. Applies debounce delay.
updateDisplayIfNeeded()	Calculates current hours and minutes for both timers. Redraws the OLED only if at least one minute value has changed. Draws Timer 1 in the top half and Timer 2 in the bottom half, separated by a horizontal line.
updateLedHeartbeat()	Advances the lit cheek LED to the next position once per HEARTBEAT_INTERVAL_MS milliseconds, providing a visual running indicator.
DrawbitLogo()	Displays the 1ST Maker Space bitmap logo at startup. Called once from setup().

6. loop()

The loop() function calls all three active helper functions in sequence, then pauses for 10 ms. The 10 ms pause is a loop throttle — it prevents the processor from checking buttons and recalculating times thousands of times per second when once per 10 ms is more than sufficient. This is a common pattern in embedded programming.

Students should notice that none of the three helper functions block — each one checks whether it has anything to do and returns immediately if not. This is what allows multiple timed behaviors (two timer displays, one heartbeat LED) to coexist in a single loop without interfering with each other.

Key Coding Ideas Highlighted

•Timestamp-based timing: Using millis() snapshots to measure elapsed time without blocking.
•Input handling: Reading digital pins with INPUT_PULLUP and interpreting PRESSED as LOW.
•Display efficiency: Only redrawing the OLED when the minute value changes; using a sentinel to force immediate updates after resets.
•Heartbeat LED: An independent timed behavior running alongside the display, both without delay().
•Debouncing and efficiency: Using a short delay after each button press to prevent multiple triggers.

Lesson Elements	
Introduction to Mind the Dog 5-10 minutes	Ask: “If you needed to track two separate timers with no phone and no clock, how would you do it?” Take a few answers. Then show the running Frog — two timers counting, LED cycling. Ask students what they think it does before explaining. Explain the difference between delay() (blocking) and millis() (non-blocking). Use a simple analogy: a stopwatch records a start time and lets you check elapsed time at any moment without freezing. A delay() call is like putting the whole program to sleep.
Access 1MS Code – 15 minutes	Visit the 1st Maker Space website and access the library of code developed specifically for the FMT. Here, you will find a section with various code (called projects) written for each major component of the trainer. The teacher will locate the sketch by clicking on Mind the Dog. Here, one can find the sketch to share with students. Students can simply copy the code by selecting the blue copy button.

Project Code:

```
////////////////////////////////////////////////
// 1.03 - Analog Write

byte LED1 = 13;

void setup() {
  pinMode(LED1, OUTPUT);
  analogWrite(LED1, 127);
}

void loop() {

}
////////////////////////////////////////////////
```

Have students (individually or in pairs) run Mind the Dog on their Frog. Ask them to:
• Press SW1 (BUTTON_ONE) and observe the screen and NeoPixel eyes.
• Press SW2 (BUTTON_TWO) and observe the changes.
• Try pressing buttons quickly vs. slowly and notice whether the display flickers or repeats.
Students record their observations.

Explain the Program–
15-25 minutes

Bring the class together and show the Mind_The_Dog.ino code on a projector.

Walk through the program structure:
• Constants and tuneable values (MILLIS_PER_MINUTE, HEARTBEAT_INTERVAL_MS, etc.).

Constant	Production Value	Purpose
MILLIS_PER_MINUTE	60000	Milliseconds in one displayed minute. Change to a small value (e.g. 600) for accelerated demo.
MILLIS_PER_HOUR	3600000	Milliseconds in one displayed hour. Must equal MILLIS_PER_MINUTE × 60 for consistent display.
HOURS_ROLLOVER_LIMIT	100	Maximum hours displayed before wrapping back to zero.
HEARTBEAT_INTERVAL_MS	1000	How often the active cheek LED advances, in milliseconds.
DEBOUNCE_DELAY_MS	250	Hold-off after a button press to prevent multiple triggers from one physical press.
FORCE_SCREEN_UPDATE	99	Sentinel value written to lastMinutes after a reset to force an immediate display redraw.

• Timer start-time variables and how elapsed time is calculated.
• Helper functions: handleInputs(), updateDisplayIfNeeded(), and updateLedHeartbeat().

Function	Responsibility
handleInputs()	Polls both buttons. If pressed, resets the corresponding timer start time, sets the forced-redraw sentinel, and sounds a confirmation tone. Applies debounce delay.
updateDisplayIfNeeded()	Calculates current hours and minutes for both timers. Redraws the OLED only if at least one minute value has changed. Draws Timer 1 in the top half and Timer 2 in the bottom half, separated by a horizontal line.
updateLedHeartbeat()	Advances the lit cheek LED to the next position once per HEARTBEAT_INTERVAL_MS milliseconds, providing a visual running indicator.
DrawbitLogo()	Displays the 1ST Maker Space bitmap logo at startup. Called once from setup().

• setup(): what runs once.
• loop(): what repeats and how button presses change the state.

OLED Display: The 128×64 pixel display is divided into two equal halves by a horizontal line at y=31. Each half shows one timer:

Region	Pixels	Content
Top label	y = 0	"DOG ONE ELAPSED:" in text size 1 (small).
Top time	y = 12	Hours and minutes in text size 2 (large), zero-padded: e.g. "02h 07m".
Divider	y = 31	Full-width horizontal line separating the two timer regions.
Bottom label	y = 35	"DOG TWO ELAPSED:" in text size 1 (small).
Bottom time	y = 47	Hours and minutes in text size 2 (large), zero-padded: e.g. "00h 45m".

Encourage students to connect each part of the code to what they saw the Frog do.

Extension – 20-30 Minutes

Students choose one or more challenges to extend the program. Examples include:
• Change the display to show seconds in addition to minutes.
• Replace the cheek LED heartbeat with the NeoPixel eyes, changing eye color based on elapsed time (e.g., green under 2h, yellow 2–4h, red over 4h).
• Add an alarm: play a tone on the piezo if either timer exceeds a configurable threshold (e.g., 4 hours).
Students should test their changes and be prepared to explain what they modified and what effect it had.

Career Exploration:

- A good resource is the IDOE Career Explorer database.

● Another good resource for career information is the Bureau of Labor Statistics
Practical Application: Programs such as Mind the Dog are practical embedded systems applications. The same elapsed-timer pattern is used in everything from game engines to industrial controllers. Students can adapt it to track any two-event problem by simply changing the labels and threshold constants.

Additional Resources:

- 1st Maker Space Frog Microcontroller Library
- 1st Maker Space Learn Arduino with the Frog Microcontroller

Performance Assessment/Check for Understanding :

- Was the student able to successfully load Mind_The_Dog.ino to the Frog?
- Are students connecting button presses with changes in the program's state?

Additional Feedback: