```python
import serial
import time
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np
from datetime import datetime
import glob
import sys

# --- Serial setup (auto-detect) ---
def find_serial_port():
    # Search for likely serial device patterns
    candidates = glob.glob('/dev/tty.usbserial-*') + glob.glob('/dev/tty.usbmodem*')
    if not candidates:
        print("⚠  No serial devices found matching /dev/tty.usbserial-* or /dev/tty.usbmodem*")
        print("   Run 'ls /dev/tty.*' to check your device name manually.")
        sys.exit(1)
    elif len(candidates) > 1:
        print("Multiple serial devices found, using the first one:")
    print(f"→ Using port: {candidates[0]}")
    return candidates[0]

try:
    port = find_serial_port()
    ser = serial.Serial(port, 9600, timeout=1)
    time.sleep(2)
except Exception as e:
    print(f"Failed to open serial port: {e}")
    ser = None  # Fallback for testing

# --- Serial setup (manual) ---
# ser = serial.Serial('/dev/tty.usbserial-14330', 9600, timeout=1)  # Mac
# ser = serial.Serial('COM3', 9600, timeout=1)  # Windows
time.sleep(2)

# For testing without Arduino:
# ser = None

# --- Data storage ---
all_times = []
all_thicknesses = []
recent_times = []
recent_thicknesses = []

# --- Log file setup ---
log_filename = "data_log.csv"
with open(log_filename, "a") as f:
    f.write(f"\n# Log started: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")
    f.write("Time(ms),Thickness(mm)\n")

# --- Plot setup ---
fig, ax = plt.subplots()
(line,) = ax.plot([], [], lw=2)
ax.set_title("Live Thickness Readings")
ax.set_xlabel("Time (ms)")
ax.set_ylabel("Thickness (mm)")

# Horizontal reference lines
ax.axhline(1.65, color="gray", linestyle="--", linewidth=1)
ax.axhline(1.85, color="gray", linestyle="--", linewidth=1)
ax.axhline(1.75, color="red", linestyle="-", linewidth=1.5)

text_stats = ax.text(
    0.02,
    0.95,
    "",
    transform=ax.transAxes,
    fontsize=11,
    verticalalignment="top",
    family="monospace",
)
```

```python
# --- Update function ---
def update(frame):
    global all_times, all_thicknesses, recent_times, recent_thicknesses

    # Read line from serial (mocked here)
    if ser:
        raw_line = ser.readline().decode("utf-8").strip()
    else:
        # Simulated test data
        current_time = len(all_times) * 100
        simulated_thickness = 1.75 + 0.05 * np.sin(len(all_times) / 10)
        raw_line = f"{current_time},{simulated_thickness:.3f}"
        time.sleep(0.1)

    if not raw_line:
        return line, text_stats

    try:
        t_str, thick_str = raw_line.split(",")
        t, thick = float(t_str), float(thick_str)

        # Append to full and recent lists
        all_times.append(t)
        all_thicknesses.append(thick)
        recent_times.append(t)
        recent_thicknesses.append(thick)
        if len(recent_times) > 200:
            recent_times = recent_times[-200:]
            recent_thicknesses = recent_thicknesses[-200:]

        # --- Stats ---
        mean_total = np.mean(all_thicknesses)
        std_total = np.std(all_thicknesses)
        mean_recent = np.mean(recent_thicknesses)
        std_recent = np.std(recent_thicknesses)

        # --- Update plot ---
        line.set_data(recent_times, recent_thicknesses)
        ax.relim()
        ax.autoscale_view()

        # --- Update stats text ---
        text_stats.set_text(
            f"Total Mean:   {mean_total:6.3f} mm\n"
            f"Total Std:    {std_total:6.3f} mm\n"
            f"Last 200 Mean:{mean_recent:6.3f} mm\n"
            f"Last 200 Std: {std_recent:6.3f} mm"
        )

        # --- Log to file ---
        with open(log_filename, "a") as f:
            f.write(raw_line + "\n")
            print(raw_line)

    except ValueError:
        pass  # skip malformed lines

    return line, text_stats

# --- Initialize plot ---
def init():
    line.set_data([], [])
    text_stats.set_text("")
    return line, text_stats

# --- Animation ---
ani = animation.FuncAnimation(fig, update, init_func=init, interval=100, blit=False, cache_frame_data=False)
plt.tight_layout()
plt.show()

# --- Cleanup ---
if ser:
    ser.close()
```