Multiple Choice Questions

1. Which of the following best describe the difference between a potentiometer and a rheostat?
    a. A potentiometer is a variable resistor, while a rheostat is a variable voltage divider.
    b. A potentiometer is a variable voltage divider, while a rheostat is a variable resistor.
    c. A potentiometer is a rheostat.
    d. A potentiometer is a fixed resistor, while a rheostat is a variable resistor.

    Correct answer: b) A potentiometer is a variable voltage divider, while a rheostat is a variable resistor.
    Explanation: (source-slides) A potentiometer is a variable voltage divider, while a rheostat is a variable resistor.

2. Why is a flyback diode needed when driving an inductive load like a DC Motor?
    a. To control the direction of a DC motor
    b. To reduce electromagnetic interference
    c. To suppress voltage spikes when the current stops flowing
    d. To convert AC to DC

    Correct answer: c) To suppress voltage spikes when the current stops flowing.
    Explanation: (source-slides) When current stops flowing in an inductive load, the collapsing magnetic field can cause a voltage spike. A flyback diode suppresses this to protect the transistor.

3. In 1991, Mark Weiser envisioned that in the 21st century, computers would weave themselves into the fabric of everyday life until they become indistinguishable from it. However, the proliferation of computers and digital devices has led to the scarcity of a particular resource, which was suggested by a Carnegie Mellon University faculty member back in 1969, even before Weiser's vision. What is this scarce resource?
    a. Silicon
    b. Attention
    c. Privacy
    d. Physical space

    Correct answer: b) Attention.
    Explanation: (source-slides) Simon talks at Johns Hopkins & CIOS Conf. in Tokyo, Fall 1969 “In an information-rich world, the scarce resource is [...] attention”

4. Based on our learnings from the Sensors in Home readings, when deploying sensing systems in homes, which of the following is likely to be a significant challenge?
    a. Ensuring the privacy and security of the collected sensor data
    b. Accurately detecting overlapping / simultaneous activities from multiple occupants
    c. Generalisability of the systems working in arbitrary homes
    d. All of the above

    Correct answer: d) All of the above.
    Explanation : (source-slides) A significant difficulty here is overlapping / simultaneous activities - This generally can be a hard problem. Other Issues - How to handle arbitrary homes, Privacy not addressed

5. When multiple devices attempt to communicate simultaneously on a single shared wire, it can lead to signal corruption and potential hardware damage due to short circuits. Which of the following can help prevent short circuits in this scenario?
    a. Implementing a coordination protocol to ensure that only one device transmits at a time
    b. Using an "open collector" partial hardware solution with a pull-up resistor
    c. both a & b
    d. None of the above

    Correct answer: c) both a & b
    Explanation: (source-slides) Message obviously gets messed up or we have a short. Hard to avoid this by coordination protocol alone, typically use a partial hardware solution: “Open Collector” output [..] pull-up resistor

2 Long-Answer Questions

Question 1)
You are working on an activity recognition system to detect six different workout activities (pushups, situps, squats, lunges, burpees, jumping jacks) using data from sensors embedded in a workout mat. You train a ML model on a large dataset collected from 50 different users. The model achieves 99.5% accuracy on the training set. But when you test it on data from 10 new users, the accuracy drops to 75%. Getting additional data from new users is not feasible due to the high cost and effort involved in data collection.

1. What problem do you think the model is experiencing? Name the common term for this issue and explain why it occurs and how it is typically identified.
2. Given the constraint of limited test data, how would you evaluate an improved version of the model to get a realistic estimate of its performance on new users? Name and describe the process you would use and explain why it helps assess the model's generalization ability.

Answer 1)
The model is likely experiencing overfitting - it has learned to fit the training data very closely but fails to generalize well to new, unseen data. Overfitting occurs when a model becomes overly complex and starts to learn noise and irrelevant patterns in the training set rather than learning the underlying patterns that generalize to new data. Signs of overfitting include extremely high performance on the training set but a significant drop in performance on a separate test set, as observed here.

Given the constraint of limited test data, I would use cross-validation to assess the model's generalization performance. In cross-validation, the available data is partitioned into k subsets or "folds". The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, using each fold as the validation set once. The validation results are averaged across the k folds to get a more robust estimate of the model's performance. The key is tuning the model to maximize cross-validation performance, not just training set fit.

Question 2) This summer, you are building an interactive gadget. The prototype exhibits some inconsistent behavior. In the midst of this madness, you frantically try to recall the sage advice of Prof. Scott Hudson. In your answer:

1. Explain the key debugging principles you would follow and your reasons behind them.
2. Identify the debugging tools you would use and describe how you will use them.

Answer 2) To debug the smart home device, I would follow these key principles:

Start with the basics:
- Check component orientations, wiring connections, and look for shorts. Verify these visually and with a multimeter before powering on.
- Ensure reliable power and ground connections using a multimeter. Inconsistent behavior could be caused by insufficient current. Add an "I'm alive" LED that flashes early in the code to confirm successful startup.

Check, check, check, test, test, test:
- Hardware: Thoroughly inspect soldering under magnification after each addition/change. If a problem is found and fixed, recheck other components for potential damage.
- Software : Write simple, isolated functions that verify functionality step-by-step. Only move on after confirming the current step is bug-free to avoid compounding issues.

Proceed from known working elements:
- Maintain a list of verified working components and code snippets and functions.
- When stuck, go back and reconfirm things that were thought to be working, even if that means starting from the basic "blink" program.

Perform experiments to isolate causes:
- Create hypotheses about whether issues are software or hardware related, or which specific parts are problematic. Design tests to confirm or reject these hypotheses.
- Use test frameworks to independently verify component functionality when possible.
- Swap in known good components to test, being mindful not to damage multiple parts.

I would employ the following **debugging tools**:

1. Multimeter and Logic probe - for continuity checks, to verify power reach, measure current load
2. Oscilloscope - Observe and measure voltage levels over time. Adjust scaling and triggering to capture relevant details.
3. LED debugging - Strategically place LEDs to indicate if a code snippet was executed or a condition was met.