Intro to Computer System and Platform Technologies:

Step 1: Understanding the Learning Outcomes

Before starting, ensure you understand the course learning outcomes related to number systems and digital logic. Your project should demonstrate these concepts using the Micro:bit.

Step 2: Forming a Team

Partner with a classmate to form a two-person team. Discuss each other's strengths and how you can collaborate effectively.

Step 3: Brainstorming Project Ideas

Brainstorm ideas for a project that applies number systems or digital logic. For example, you could create:

- A binary counter that increments or decrements with button presses.
- A simple game that uses binary logic to determine winning conditions.
- An encoder/decoder that translates between different number systems.

Step 4: Planning Your Project

Once you have an idea, plan your project. Determine what components you'll need, what each team member will do, and set a timeline for your work.

Step 5: Building the Project

Use the resources available at MakeCode for micro:bit to program your Micro:bit. You may need to learn the platform's specifics and possibly some JavaScript or Python if you're not already familiar.

Step 6: Overcoming Challenges

You may face technical challenges, such as getting the Micro:bit to perform as expected or integrating different components. Use online forums, official documentation, and class resources to troubleshoot issues.

Step 7: Writing the Reflection

For your individual reflection, structure it as follows:

Introduction

- Briefly describe the project and its objectives.

Demonstrated Knowledge

- Discuss the concepts you applied that you were already familiar with, such as number systems or basic programming constructs.

New Learnings

- Reflect on any new skills or knowledge you acquired during the project. This could be learning how to code for hardware if your experience is mostly with software.

Challenges and Solutions

- Describe specific challenges you encountered and how you addressed them. This can include both technical hurdles and teamwork issues.

Areas for Improvement

- Critically assess your project and identify areas where you could improve, such as in planning, execution, or deeper understanding of the concepts.

Conclusion

- Summarize what the project achieved and how it has prepared you for future work.

Step 8: Code Submission

Make sure your code is well-commented and adheres to good coding practices. Submit it through the provided link, ensuring that only one team member does the submission.

Step 9: Submit Reflection

Submit your individual reflection to Turnitin, adhering to the page limit and any formatting guidelines provided by your instructor.

Remember, while working on this project, regular communication with your team member is key. Use version control systems like Git to manage code changes effectively, if permitted. And make sure your project not only works but also clearly demonstrates the learning outcomes related to number systems and digital logic.

Example:

Project Description:

Create a binary counter on the Micro:bit LED display. The Micro:bit has two buttons, `A` and `B`. When button `A` is pressed, the counter increments. When button `B` is pressed, the counter decrements. The counter should wrap around if it reaches the maximum or minimum value it can display.

Components Needed:

- BBC Micro:bit
- USB cable for programming

Programming Environment:

- MakeCode editor (https://makecode.microbit.org/)

Step 1: Initialize Variables

Set up a variable to keep track of the counter's current value and initialize it to zero.

Step 2: Display Function

Create a function to display the current counter value in binary on the LED display. The Micro:bit has a 5x5 LED display, allowing us to display up to five bits, giving us a range from 0 to 31.

Step 3: Button Press Events

Set up event handlers for button `A` and button `B` presses. In these handlers, increment or decrement the counter value, ensuring that it wraps around appropriately.

Step 4: Update Display

After any change in the counter's value, call the display function to update the LED display.

Snippets of Code:

Here's how you might code this in the MakeCode editor using JavaScript:

```
let counter = 0

input.onButtonPressed(Button.A, function () {
    counter += 1
    if (counter > 31) {  // Wrap around if the counter goes above 31
        counter = 0
    }
    displayCounter(counter)
})

input.onButtonPressed(Button.B, function () {
    counter -= 1
    if (counter < 0) {  // Wrap around if the counter goes below 0
        counter = 31
    }
    displayCounter(counter)
})
```

```
function displayCounter(num: number) {
    basic.clearScreen()
    // Convert the number to binary and display it
    for (let i = 0; i < 5; i++) {
        let bit = num & (1 << i)
        if (bit > 0) {
            let x = i % 5
            let y = Math.floor(i / 5)
            led.plot(x, y)
        }
    }
}

// Start by displaying the initial value
displayCounter(counter)
```

How It Works:

- The program initializes a counter variable to 0.
- When button `A` is pressed, the counter is incremented. If it exceeds 31 (the maximum number a 5-bit binary number can represent), it wraps back to 0.
- When button `B` is pressed, the counter is decremented. If it goes below 0, it wraps around to 31.
- The `displayCounter` function converts the counter value into binary and uses the LEDs to represent it visually. It lights up the corresponding LED for each '1' bit in the binary representation.
- We use bitwise operations to extract individual bits from the counter and display them on the LED grid.

Challenges and Learning Points:

- **Challenge**: Implementing a wrap-around for the counter.
    - **Solution**: Conditional statements check if the counter exceeds its maximum or goes below zero and reset it accordingly.
- **Challenge**: Converting a decimal number to binary.
    - **Solution**: Bitwise operations are used to extract and display individual bits.

By building this project, you would reinforce your understanding of binary numbers, bitwise operations, event handling, and the process of translating a concept into a programmable solution on a microcontroller.

Highlights of the Micro:bit Binary Counter Assignment:

1. **Practical Application of Binary Numbers**: This project directly applies the concept of binary numbers to a real-world application, reinforcing theoretical knowledge with practical experience.
2. **Engagement with Physical Computing**: Working with the Micro:bit involves interacting with hardware, which can be more engaging and rewarding than software-only projects.
3. **Event-Driven Programming**: Implementing button press responses is a great way to learn about event-driven programming, a common paradigm in many applications.
4. **Problem-Solving Skills**: You will enhance your problem-solving skills by translating the concept of a binary counter into a programmatic solution.
5. **Iteration and Incrementation Logic**: The project solidifies understanding of loops, conditional statements, and the increment/decrement operations, which are fundamental in programming.

What-to-Avoid in the Micro:bit Binary Counter Assignment:

1. **Overcomplicating the Display Logic**: Keep the binary display logic simple. You don't need complex algorithms when a straightforward bitwise operation will suffice.
2. **Ignoring Edge Cases**: Ensure your counter correctly handles edge cases like wrapping from the maximum to minimum value and vice versa.
3. **Inefficient Coding Practices**: Avoid writing repetitive code. Use functions for operations that are performed more than once, like updating the LED display.
4. **Neglecting the Debounce Mechanism**: Micro:bit buttons might register multiple presses if not handled correctly. Implementing a debounce mechanism could be necessary to avoid this.
5. **Lack of Comments**: Not commenting your code can make it difficult for others (and yourself) to understand the logic behind your implementation.
6. **Disregarding Collaboration**: If working in pairs, ensure both members are contributing equally and understand the project's workings.
7. **Failing to Test Thoroughly**: Test all functionalities, including incrementing, decrementing, and wrapping the counter value, to ensure robustness.

By focusing on these highlights and avoiding common pitfalls, you can create a successful Micro:bit binary counter project that not only meets the educational objectives but also provides a solid foundation for future projects in embedded systems and digital logic.