Lab 1 – Feasibility Model Phase 1 – Instructions

ECE 298 – S2021

Overview

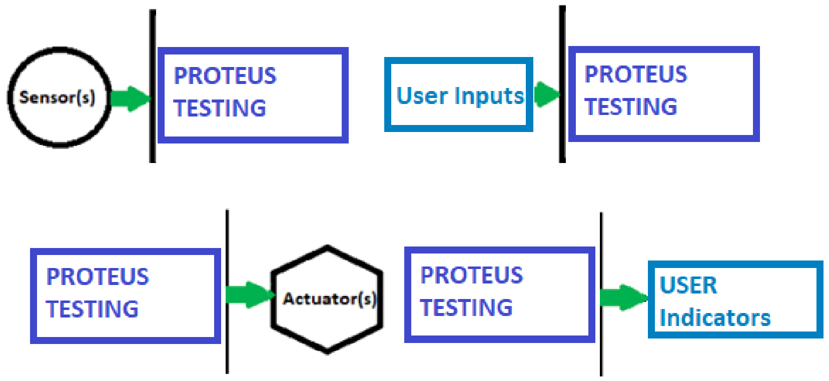

Figure 1 – Sensors, Actuators, and User Inputs and Outputs (Indicators) are stimulated or measured by a Proteus simulation ("Proteus Testing" block) to verify and validate their functionality.

The Feasibility Model is a physical proof-of-concept whose purpose is to demonstrate whether a proposed solution may work and is worth taking to the prototype stage. The feasibility model consists of schematics and simulations with sensors, actuators, and user I/O that your group has selected from provided component libraries for your chosen project, like in Figure 1. Your group will develop the Feasibility Model in Lab 1 and Lab 2.

Begin by choosing a project on the LEARN page.

Create a Lab 1 working directory on the N: drive so you can start exploring and developing your project and document. Download the **ECE298_Lab1.pdsprj** Proteus project file and "ECE 298 S2021 Lab 1 - Template.docx" document from LEARN into this directory. You can zip and submit the contents of this directory for assessment.

The resulting document and schematic should be clear, organized, and professional. The document should use appropriate paragraph and font styles and appropriately formatted tables and figures with explanatory text. The schematics should be neat and tidy. They will be marked using the Lab 1 Rubric on LEARN.

Deliverables

Submit a single .zip file that contains:

- A completed Lab 1 document template (PDF preferred), following the subsequent instructions
- A single Proteus project file (.pdsprj) contains all the presented schematics that simulates successfully

Part 1 – Project Design Requirements

Requirements are the statements of how a product will solve end-user problems. *Requirements* focus on the problem being solved, whereas *specifications* put hard limits on a solution's implementation. In this course we will not over-analyze the boundary between the two, but requirements are usually more abstracted than specifications.

Project Design Requirements fall into three major categories:

1. **Functional requirements** specify the performance of a design. They are related to the design's functions, identified as answers to the question, "What does it do?" For example, one coffee maker's functional requirement may specify the maximum time required to brew a pot of coffee. A DC power supply may specify what kinds of devices it must power or, more specifically, its output voltage. A vehicle alarm system may specify how much noise it makes when activated.

2. **Non-functional requirements** specify characteristics of the design that are not performance-based. These are typically product features desirable to end users. For example, ease of use, ease of manufacturing, and use of recycled materials.

3. **Constraint requirements** place limits on the design and often reflect budget or other project limitations, like PCB size, cost, weight, wiring etc.

The basic form of most of these requirements is the same: A short description that often includes a relationship (“is”, “more than”, “minimum”, etc.) and sometimes specific values. However, some may be qualitative and not quantitative, like “The solution must contain *some* recycled materials”.

State three or more major Project Design Requirements spanning the above categories that your project must meet to successfully solve your problem statement. Identify each one as being Functional, Non-functional, or Constraint (it is not necessary to have every kind of requirement for your project). You may add requirements not directly indicated in your project’s description, but remember that you will need to meet those requirements in your design.

Part 2 – Project Considerations for I/O

Determine and list the types of sensors, actuators, and user I/O that your project requires. Then, specify their essential parameters (e.g., range of operation for sensors, RPM details for motors, etc.). In Proteus, explore the devices available in the Proteus Libraries and choose ones that you will use for each item in your list. **Note:** Do not use any Raspberry Pi Hats, Arduino Shields, or web-based components in your design. Only use those in the Proteus libraries that have models included (“Show only parts with models” checkbox).

List which internal MCU Peripherals you plan to use in association with the above devices. You can find further details for the MCU in the “STM32F401RE MCU Datasheet.pdf” file on LEARN.

Project Sensors and User Inputs

- List the types of sensors (light, temperature, distance, etc.) and user inputs (pushbuttons, a keypad, etc.) you may need.

- For each sensor and user input, explore the components from the Proteus libraries. Only use those in the Proteus libraries that have models included (“Show only parts with models” checkbox).

- List how you could connect it to an MCU using general component-type terms. For example, “Use a signal converter to transform the sensor current into a voltage range for the MCU,” or “Use an opamp to scale the sensor output voltage to the MCU input pin voltage range.”

Project Actuators and User Outputs (Indicators)

- List the types of actuators and user outputs (indicators) you may require (light, sound, mechanical motion, displays, LED indicators, etc.).

- For each actuator and user output (indicator), explore the components from the Proteus libraries. Only use those in the Proteus libraries that have models included (“Show only parts with models” checkbox).

- List how you could connect it to an MCU using general component-type terms. For example, “Use the MCU’s serial interface to control a motor controller IC,” or “Use a transistor so the MCU can switch the high-current device on and off.”

Project MCU Internal Resources

- List the resources inside the MCU that might be used to implement your project (e.g., ADC, timers, interrupts, GPIO functions, etc.), describing why they are necessary. Review the MCU datasheet on LEARN for more details.
- List parameters that the software running on the MCU might require (coordinates, durations, limits, conversion factors, access codes, etc.), describing why they are necessary.

Part 3 – Device Testing Methodology

You must understand the electrical attributes of the devices you chose to use them properly. For example, operating and signal voltages and currents, signal frequencies, analog vs. digital modes, etc. You will use Proteus to create small, exploratory schematics for your chosen devices, all on one schematic sheet. Use **Proteus Virtual Instrument** models to provide stimuli (e.g., the Pattern Generator) and to observe (scopes, meters, and graphs) the components selected for the sensors, actuators, and user I/O. Your schematic can be screen-captured (⊞ + Shift + S on Windows) and placed inside this part of the Template file.

You will probably need to review the device datasheets to understand how the part works. If Proteus doesn't have the necessary device info, search the web for a datasheet. For example, searching for "LM382 datasheet" will find many links for the LM382 part datasheet (this is just an example). Datasheets can be very long and dense since they have all the electrical, mechanical, thermal, and other design information on the parts they cover. The most important information is usually on the first or second page under "Electrical Characteristics." Note that the "Absolute Maximum Ratings" are the limits beyond which the part will physically break and are not how you should typically use the component.

Implement your component tests in a single Proteus schematic. You will only use the Virtual Instruments to test your devices.

For each sensor, actuator, user input, and user output listed above:

- For each device being tested:
    - Add a subheading with the device name to make the document easier to navigate (use an appropriate Heading style)
    - State how you could control and observe the device's functions (a table may be helpful)
    - Copy/paste the project schematic section for testing it
        - Add text to explain points of interest
        - Describe the signal stimulus used in your simulations
    - Perform the testing simulations, and copy/paste the simulation results (graph, scope outputs)
    - Verification: Describe how these schematics and simulations confirm that the device is behaving as expected
    - Validation: Describe how these device behaviours satisfy your project design requirements (i.e., that the behaviour confirms that you chose an appropriate component to meet your project needs – refer specifically to the requirement(s) in question)

Figure 2 is an example of how a figure in this section may appear. Text would then follow that addresses the required points above.

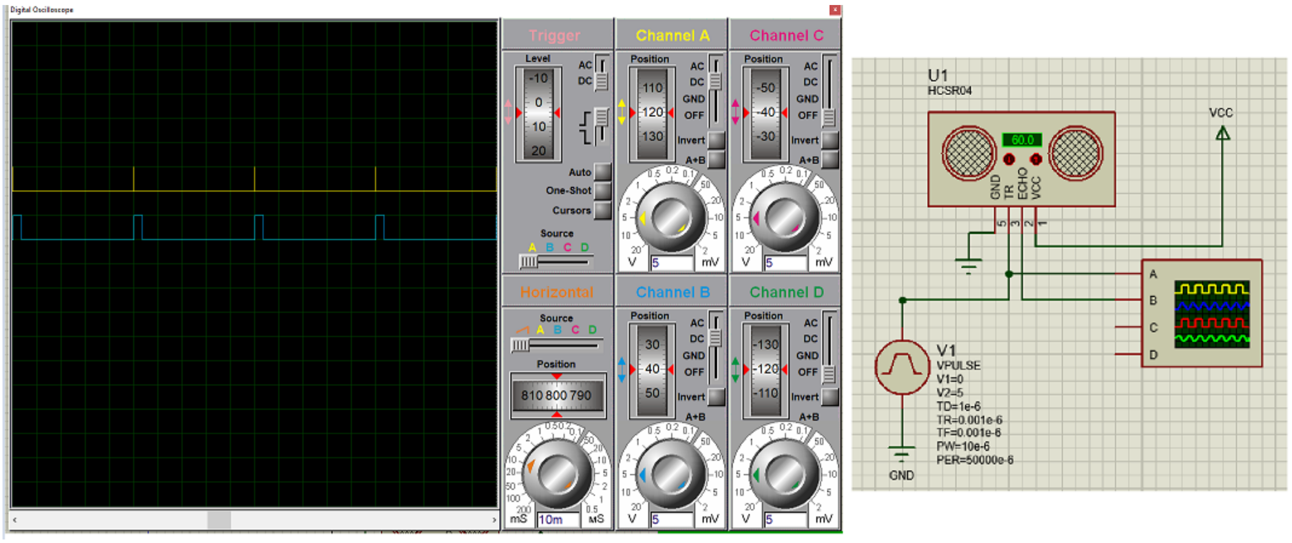

Figure 2 – A schematic and simulation exercising the HCSR04 ultrasound ranging module. The left image shows an oscilloscope that illustrates the applied trigger pulses (yellow) and received echo pulses (blue). The right image shows the test schematic.

Part 4 – System-Level Design

Your ECE 298 project now has a conceptual architecture in place with the essential requirements and devices selected. Organize your project at a high level in the form of a "System-Level Design," which can be a generic block diagram like the one in Figure 3. Note that arrowheads represent the flow of information and control throughout the system. Include your initial ideas of how you can interface your devices to the MCU (see the STM32F401RE Datasheet on LEARN for MCU resources).

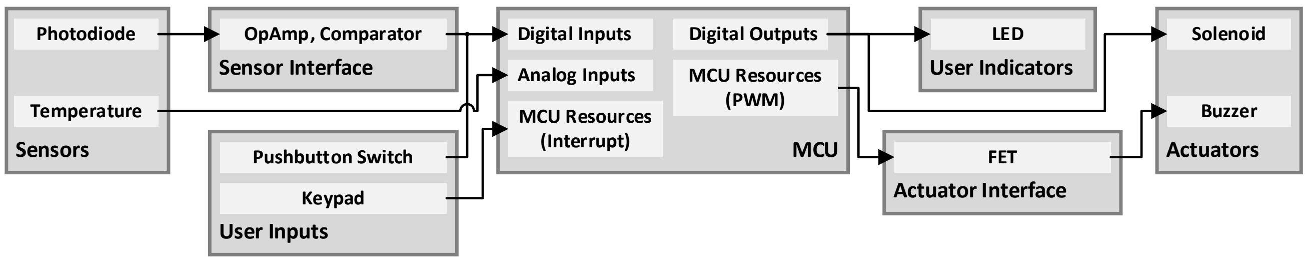

Figure 3 – A system-level block diagram showing a variety of modules and subsystems, along with arrows indicating the direction of information flow.

Create a similar system-level block diagram of your project.