Lab 2 – Feasibility Model Phase 2 – Instructions

ECE 298 – S2021

Overview

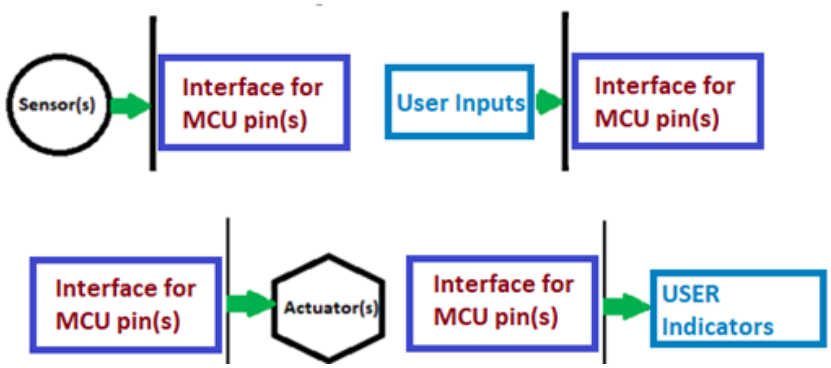

Figure 1 – The different devices and subsystems – Sensors, Actuators, and User I/O – must be interfaced to the MCU. This lab explores that interfacing.

In this lab, you will be documenting and simulating the essential characteristics of the devices you selected for your sensors, actuators, and user I/O. The tables that you complete in this document will be compared against your submitted ECE298_Lab2 project schematic.

Now that you have chosen the devices according to your project's requirements, the next step is to interface those devices to operate with the MCU. This lab consists of developing hardware interfaces that connect your selected devices to MCU pins. The specific MCU pin choices at this point are not critical. You can refer to your System-Level diagram from Lab 1 to recall your initial device interfacing ideas.

Your device interfacing schematics will be done using the Proteus ECE298_Lab2 project schematic sheet. Connectors and schematic ports are included to connect with the project in Lab 3. **These parts and ports must not be altered**. Your interface circuits must be simulated at the MCU interface, meaning that you use generator signals and probes where your circuits would connect electrically to the MCU (at the various PA, PB, PC pins). Note any errors that occur in the Simulation Logs and make corrections. Your simulations must demonstrate your circuit designs cover the sensor/actuator/user I-O range limits when connected to the MCU interface pins. For example, if a sensor outputs an analog voltage between 0.5 to 3.4 V, your simulation must cover this range.

The ECE298_Lab2.pdsprj file on LEARN gives you an easy means to open and load your ECE 298 Lab 2 Proteus project (just double-click after downloading). Save the project file by including your group number in the file name (of the form "ECE298_Lab2_GroupXX.pdsprj", where XX is your group number). In LEARN, download the ECE298_Lab2.pdsprj file into a directory on the N: drive so you can start exploring and developing your project.

Deliverables

Submit a single .zip file that contains:

- A completed Lab 2 document template (PDF preferred), following the subsequent instructions
- A single Proteus project file (.pdsprj) that contains all the necessary devices and subsystems (no MCU yet) that simulates successfully

Background Information

The following may help you understand some of the considerations necessary when designing your interface circuits.

Digital devices have specifications that govern the voltages they generate at their outputs and can expect and tolerate at their inputs. All device pin specification details are available in the Proteus component links so that

the "in-spec" device operation (normal operating conditions) can be verified. Using your devices from Lab 1, add a suitable set of components to operate the devices from within the MCU pin electrical limits.

For ECE 298, we will operate the MCU at 3.3 V only. The MCU pins operate within the voltage limits specified in Section 6.3.16 of the datasheet. The maximum current (source or sink) per pin is limited to ±25 mA. There is a further restriction that the total (aggregate) current supplied across all output pins must be less than 120 mA. At this point, these should be the only details necessary to design your device interfacing circuits for the MCU.

Voltage Translation Interfacing

Device digital inputs and outputs conform to a standardized set of acceptable values, called a "logic family". See https://www.ti.com/lit/sg/sdyu001ab/sdyu001ab.pdf for an overview from Texas Instruments. Figure 2 shows the basic voltage levels relevant to digital signalling. On the driver side, $\mathbf{V_{OH}}$ is the **output high voltage,** and $\mathbf{V_{OL}}$ is the **output low voltage** expected from a digital output. On the receiver side, $\mathbf{V_{IH}}$, the **input high voltage**, is the minimum acceptable voltage that indicates a digital '1'. Similarly, $\mathbf{V_{IL}}$, the **input low voltage**, is the maximum allowable voltage that indicates a digital '0'.

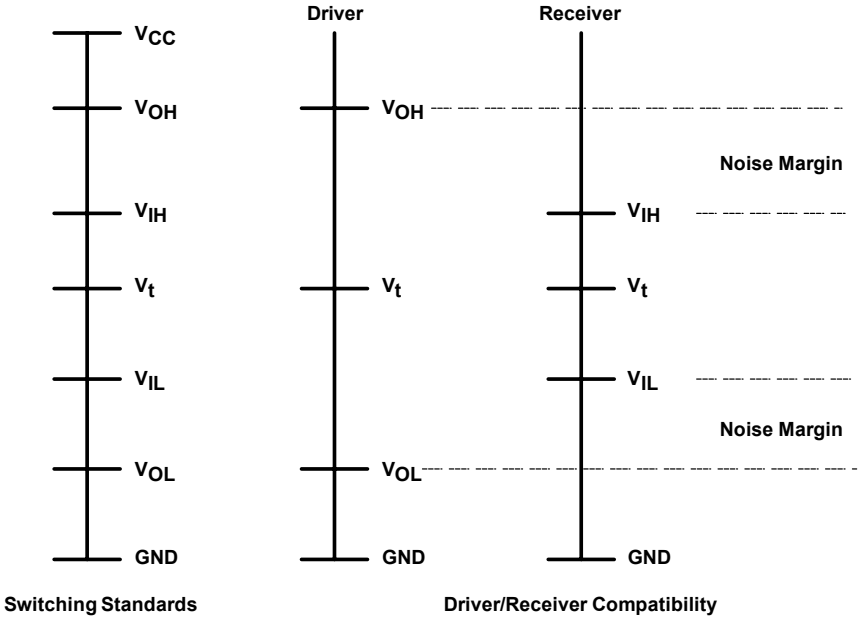

Figure 2 – Switching standards and driver/receiver compatibility (from SCEA021A).

Figure 3 shows the characteristics of Texas Instruments' implementations of different logic families. You will need to create voltage shifting interface circuits to connect separate logic families since the voltages are different.

Figure 3 – Digital I/O voltage characteristics (from SDYU001ab).

Bidirectional Voltage Interfacing

Figure 4 shows a simple circuit to translate digital signals from one level to another. It uses a typical N-channel MOSFET, the BSS138, available in the Proteus libraries. The Zener diode within the symbol is part of the MOSFET device, so it is not something you need to add. The MOSFET drain (D) will connect to the higher voltage signal (e.g., a 5.0 V signal from a device pin), and the source (S) will connect to the lower voltage signal (e.g., a 3.3 V MCU pin). This circuit operates bidirectionally but can only be used for digital signals (not to be used for analog signals).

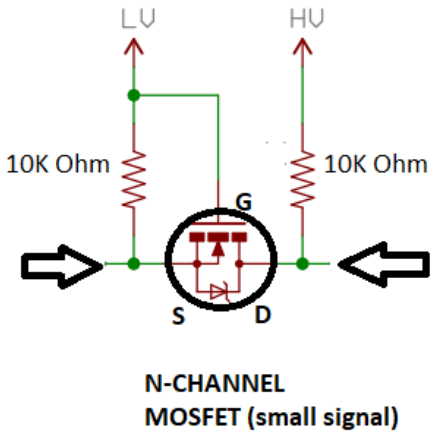

Figure 4 – A simple bidirectional voltage interfacing circuit.

Step-Down Voltage Interfacing

If your signal is one-directional, and the driver outputs a higher voltage than the receiver can handle, then a simple voltage divider will suffice. For example, the circuit in Figure 5 will step down from 5.0 V to 3.3 V.

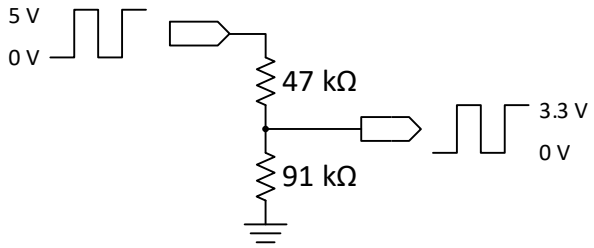

Figure 5 – A voltage divider can scale a signal down to a safe level for a lower-voltage input pin.

Step-Up Voltage Interfacing

If your signal is one-directional, and the driver outputs a lower voltage than the receiver needs, then an N-channel MOSFET with pull-up resistor will replicate the **inverted** signal at a higher voltage. For example, the circuit in Figure 6 will step up from 3.3V to 5.0 V. The 100 kΩ resistor value can be lowered to increase the maximum frequency of the digital signal at the cost of increased static power when the N-FET is conducting. You can add a second stage to un-invert the signal, but if the signal is generated by your MCU code, then just have it output an inverted drive signal. Another alternative is to use a level-shifter integrated circuit, or a comparator with a threshold voltage set to half the input voltage.

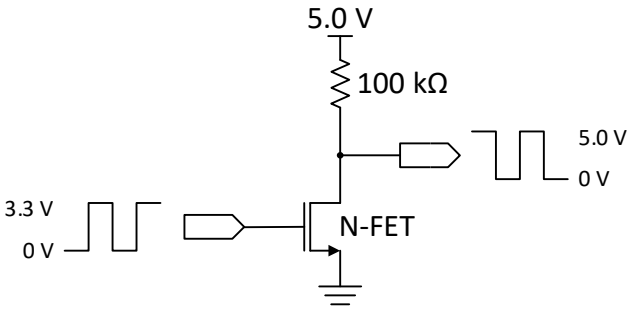

Figure 6 – A circuit to step-up a low-voltage digital signal to a higher voltage.

Analog to Digital Interfacing

Sometimes you need to check if an analog signal is above a certain threshold. One way to do this is to connect it directly to a digital input, and it will cause 0→1 or 1→0 transitions when it exceeds V_{IH} and V_{IL}, respectively. You can use a typical voltage divider circuit with resistors to scale down the voltage range of an input analog signal in such a case. But it is not wise to hold a digital input between '0' and '1' for extended periods because it can cause short circuit current within the IC and possibly damage it. If the analog signal to be converted has slow transitions, then it is better to use a comparator opamp circuit, like the one shown in Figure 7. The 10 kΩ input resistor reduces the current loading on the analog signal source. The comparator circuit creates the sharp, high-speed transitions needed for digital signalling.

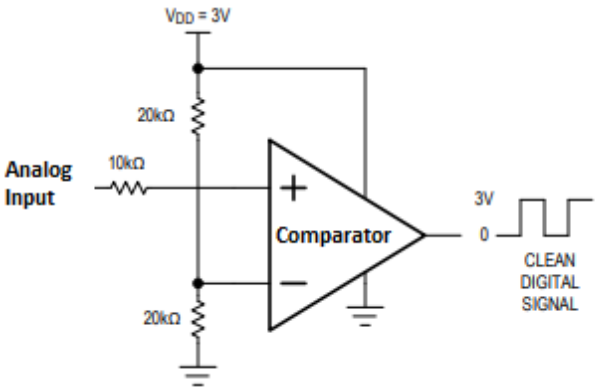

Figure 7 – An opamp comparator circuit.

Other Kinds of Interfacing

Motors require much higher current than a typical digital output can provide, so they need a **motor driver**. For DC motors, this can be as simple as a transistor, like in Figure 8. But this circuit can only spin the motor in one direction.

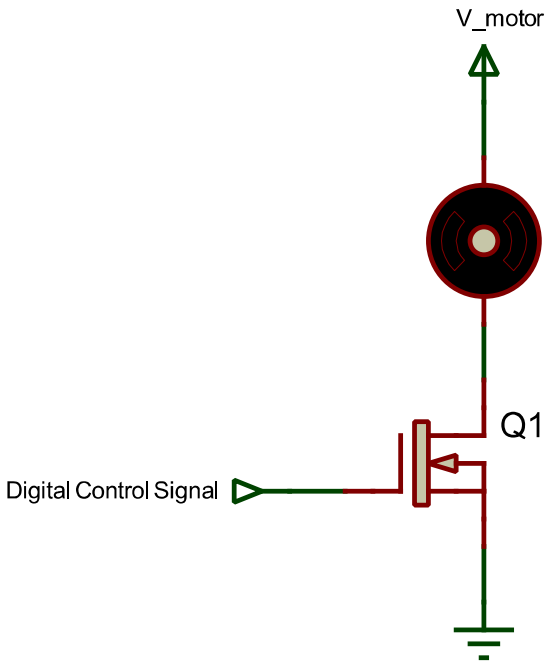

Figure 8 – An N-channel MOSFET driving a DC motor.

ICs can offer more complicated functionality, like the driver IC shown in Figure 9. Read the component datasheet to see how it works internally using H-bridge circuits (transistors connected like the letter H to allow current to flow through the motor in either direction).

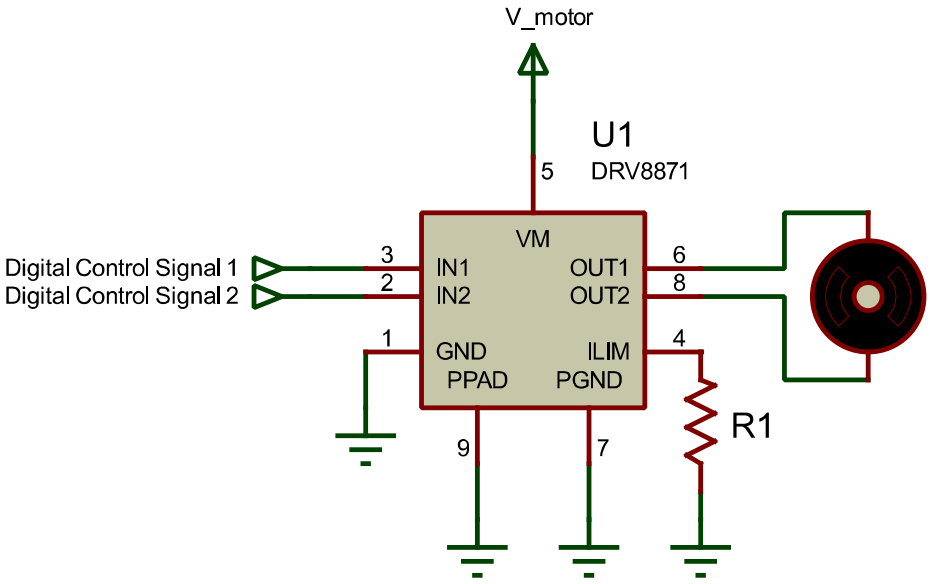

Figure 9 - A DC motor driver IC.

Motor controllers are ICs that do more than switch current through the motors. Some can be connected to the MCU with a digital serial bus, like I2C or SPI, and can use closed-loop feedback to maintain a requested motor

RPM. Those devices are not contained in Proteus, but you should understand how to use one by the end of this course. Proteus also has stepper motors and a stepper motor controller.

Instructions

Develop the devices for your project in more detail. List the details for **each device or subsystem** by completing a table like in the template. Copy and complete a table for each type of sensor, actuator, or user I/O. Delete rows that do not apply to your device (i.e., delete the analog rows for a digital device, and vice versa). Delete rows that do not apply to your device (i.e., delete the analog rows for a digital device, and vice versa). Add any clarifying information you feel is necessary. Similar devices can be grouped. For example, one table for switches, one table for LEDs, etc.

Summary

Table instructions.

Item	Description
Purpose	Describe why this device or subsystem is needed for your project.
Device physical domain and range	Describe the physical parameters to be sensed or actuated. For example, light, motion, distance, sound, temperature, etc. Describe the maximum, minimum, and nominal operating points, where applicable. For example, a motor's minimum and maximum could be 0 to 800 rpm. A distance sensor may work from 5 cm to 100 cm. Some device ranges may not be known at this time, so it is okay to note their absence. For example, "The LED's maximum brightness will depend on the specific LED chosen and is not known at this time." "The buzzer's maximum volume is currently unknown."
Device type chosen	Describe the kind of device chosen to perform the sensing, actuating, or user input/output. Here are some examples: A distance sensor could be ultrasonic, infrared, LIDAR, and other technologies. A motor could be brushed DC, brushless DC, stepper, servo, or AC, among others. A switch could be on/off, momentary, rotary, etc. A display could be character or graphical, LED, OLED, LCD, etc.
Proteus Library component name	If the device or subsystem corresponds to one or more specific Proteus components, put the part names here. Otherwise, this can be left blank, and does not need to include basic components like resistors, capacitors, etc.
Device input / output properties	Describe the device or subsystem input and output signal types. Broadly, these can be analog or digital. Analog signals usually represent information as time-varying voltages, currents, resistances, capacitances, or frequencies. Digital signals usually communicate information on serial or parallel data busses, or encode the information in digital pulse timing, like a time-varying pulse width, duty-cycle, or frequency.

Item	Description
	Also note whether digital outputs are push-pull (outputs a strong '0' and strong '1'), tri-state ('0', '1', or disconnected – called 'High Z' or just 'Z'), or open-drain / open collector (strong '0', but does not pull up to '1' – used to allow many devices to signal on a bus that has a pull-up resistor). Outputs can also be open drain / open collector, but not for the purpose of bus signalling. The output acts as a switch to ground, powering high-current devices by enabling or interrupting the current like a switch.
Device input / output range	Describe the expected range of the device inputs and outputs. Here are some analog examples: An analog sensor voltage output might range from 0.5 V to 2.5 V. An analog actuator, like a DC motor, may accept 0 V to 12 V. An LED may accept 0 mA through 20 mA max. A buzzer may have a nominal frequency of 3 kHz at 5.0 V (meaning a 3 kHz sine or square wave that goes from 0 V to 5 V). Here are some digital examples: An IC may communicate via an I2C serial bus, or perhaps a one-direction 8-bit parallel bus with Chip Select and Clock. Such busses would have a maximum frequency or data rate. An ultrasonic sensor may require a digital active-high trigger signal and output a variable pulse width echo signal. The trigger signal would have required pulse characteristics, like maximum pulse repetition frequency and minimum/maximum pulse width. The echo signal would also have expected characteristics, like the delay between trigger and echo, and minimum and maximum pulse widths. Motor driver ICs may accept a maximum PWM frequency. State the working voltage of the digital I/O (e.g., 3.3 V or 5.0 V). Each digital output will have a minimum output high voltage, $V_{OH,min}$, and maximum output low voltage, $V_{OL,max}$. Each digital input will have an maximum input high voltage, $V_{IH,max}$, and minimum input low voltage, $V_{IL,min}$, respectively. Note them here or note if the values are unavailable.
MCU connectivity details	Describe how you wish to connect to the MCU. For example, an ultrasound module would use an MCU digital output for the trigger signal, and an MCU digital input for the echo signal. A light intensity sensor may be connected to an MCU analog input. Note that there are no native MCU analog outputs, but a PWM signal can be low-pass filtered to produce a stable analog output voltage.
Device/MCU interfacing details	Describe any interfacing required to connect your device or subsystem to the MCU. There are four possibilities:

Item	Description
	• **Analog to Analog (A/A)**. For example, scaling down voltage using a voltage divider, or scaling up voltage using an opamp. • **Analog to Digital (A/D)**. This does not include using the MCU ADC (analog to digital converter), because the signal is provided to the MCU in analog form. This interfacing corresponds to using an external ADC IC or using a comparator circuit to provide a digital output when an analog signal has exceeded a given threshold voltage. • **Digital to Analog (D/A)**. Sometimes the MCU needs to control devices that require an analog input voltage, so a digital value from the MCU needs to generate a corresponding analog voltage. This is usually done with a DAC (digital to analog converter), and many MCUs contain DAC modules. The MCU in this course does not, so you may elect to use a DAC IC to perform this interfacing. Alternatively, a low-pass filtered PWM signal can be low-pass filtered into an analog voltage proportional to the duty-cycle. The inverse case of a digital device driving an MCU analog input is uncommon (purposefully converting a digital signal into a scaled analog voltage for the MCU ADC). • **Digital to Digital (D/D)**. This corresponds to the unidirectional or bidirectional voltage scaling necessary to connect devices with different logic working voltages.

Schematics and Simulations

In this section Copy/Paste snips of your schematics and simulations relevant to the above devices or subsystems that enable its connection to the MCU pins. The MCU pins do not need to be specified. Describe the component signal stimulus used in your simulation. Describe the observed behaviour of the simulations and how the component will meet the requirements of your project. Add text to explain points of interest.

Examples

Here are a few examples using devices that probably are not in your project. Accordingly, the Purpose is with reference to a fictitious project.

Linear Position Sensor

Item	Description
Purpose	A Linear Position Sensor is required to measure the position of the slider on the track.
Device physical domain and range	The chosen linear position sensor (TSP-L-0012-103-1%-RH) measures linear position from 0 to 12.5 mm.
Device type chosen	The chosen sensor is a resistive membrane type.
Proteus Library component name	There is no Proteus component directly available, so ECE298_GEN_POTENTIOMETER will be used in its place.
Device input / output properties	The passive two-terminal device resistance varies linearly with position.
Device operating maxima and minima	The resistance varies from 0 Ω to 10 kΩ
MCU connectivity details	The MCU will use an Analog Input pin to detect the voltage generated across the device.
Device/MCU interfacing details	A voltage divider circuit will convert the sensor's resistance into a voltage that the MCU's ADC can measure.

Buzzer

Item	Description
Purpose	The buzzer will warn users when the water level is too low.
Device physical domain and range	The buzzer creates sound at 500 Hz. The volume level is not known as this is a generic Proteus component.
Device type chosen	The Proteus component type is not specified, but the chosen buzzer will be a piezoelectric type.
Proteus Library component name	BUZZER
Device input / output properties	The device has two terminals on which an analog voltage is applied.
Device operating maxima and minima	The buzzer accepts a 0 to 12 $V_{pk\text{-}pk}$ square wave at 500 Hz.
MCU connectivity details	An MCU digital output will generate the 500 Hz square wave.
Device/MCU interfacing details	An interface circuit will translate the 0 to 3.3 V square wave from the MCU into a 0 to 12 V square wave. Since the buzzer load resistance is 12 Ω, this will require 1 A of current, so an appropriately sized transistor will generate the 12 V pulse.