mbed Lab 4: Can You Hear Me Now?

CS 220

Introduction

For this lab, you'll use your LCD display with an HC-SR04 Ultrasonic Ranging Module (HCSR04) to build a distance measuring tool. Ultrasonic sensors similar to this one are often used for the parking sensors which are offered as options on new cars today. These sensors are easy to spot — four quarter-sized disks embedded within the front or rear bumper of a car. To give you a bit of experience with writing a device driver, you'll write a simple device driver for the HCSR04, working from the specifications given in the device's data sheet. (Yes, hypothetically you could just use the mbed's HCSR04 library, but anyone experimenting with embedded systems should write at least one device driver themselves. Now's the time.)

In addition to the HCSR04, you'll be working with an SN74LVC244A ('244A) buffer integrated circuit in this lab. Why the '244A? The KL25Z uses 3.3V logic and a 3.3V power supply. The HCSR04 uses 5V logic and a 5V power supply. The trigger signal transmitted from the KL25Z to the HCSR04 to start a measurement cycle can be read by the HCSR04 as is. The echo signal, representing the distance measurement, transmitted from the HCSR04 back to the KL25Z must be level-translated from 5V logic levels to 3.3V logic levels to avoid burning-out the input pin on the KL25Z. The '244A is used to accomplish the level translation.

Lab Objectives

1. Write a simple device driver for an HCSR04.
2. Build a complete prototype embedded system for measuring distances.

Lab Parts List

1. KL25Z board and Quick Reference Card
2. USB cable
3. Breadboard, with LCD display already mounted on it (and wired to the KL25Z in Lab 3), and with the HCSR04 and '244A already mounted on it.
4. Wires. More wires.

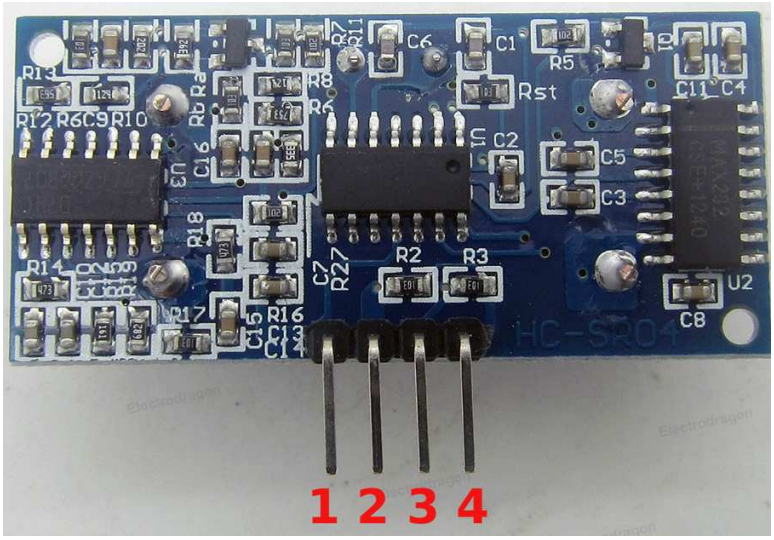
Lab Procedure

Read each of the following steps **completely** before acting on them.

1. You won't need to apply power to the KL25Z until later, so set it and the USB cable aside for now.

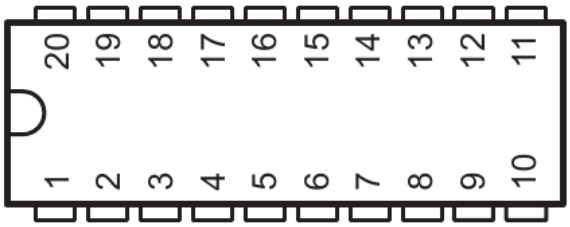
2. Let's start by building the circuitry for the KL25Z and the HCSR04 to communicate with each other. Part of that circuitry uses the '244A to level-translate the signal voltages used by the HCSR04 to those used by the KL25Z. Without this chip, the KL25Z's input pin would be damaged by the voltage output by the HCSR04.

   The following wiring list uses this numbering scheme for the HCSR04's pins:



   Note that this is a rear view of the HCSR04.

   This numbering scheme is used for the '244A's pins:



   Note that the small "U"-shaped indentation is on the left-hand side of the chip and is used to identify pin 1 of the chip. The chip itself is black and located in the center of your breadboard.

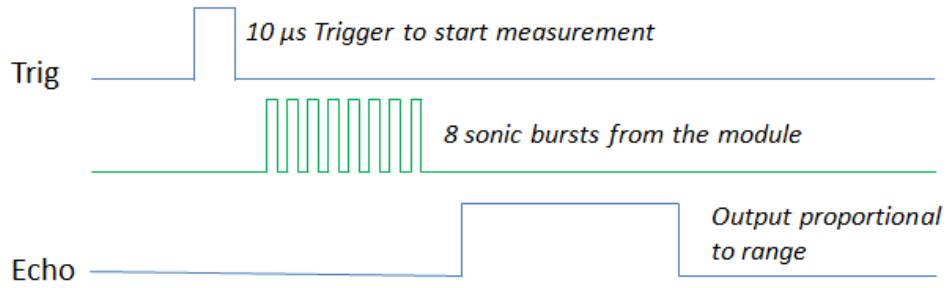
   Use the rows of this wiring list to make your jumper wire connections. "NC," no connection, means do not make a connection to the chip/breadboard bus strip. "Conn," connect, means make a connection to the bus strip.

HCSR04 Pin	'244A Pin	KL25Z Pin	+ Bus Strip	– Bus Strip
2	2	NC	NC	NC
NC	18	PTB8	NC	NC
3	NC	PTB9	NC	NC
4	NC	P5V_USB	NC	NC
1	NC	NC	NC	Conn
NC	20	NC	Conn	NC
NC	1 and 10	NC	NC	Conn

   With this wiring list, the KL25Z's pin `PTB9` drives the HCSR04's trigger input, and pin `PTB8` receives the echo signal from the HCSR04. These trigger and echo signals will be explained momentarily.

3. If you're not yet confident about your wiring ability, have me check your work.

4. Now, let's talk about how the HCSR04 works. It has an input named `Trigger` and an output named `Echo`. Ordinarily, the `Trigger` input, which, in our case, will be controlled by the KL25Z, is low (logic 0). The `Echo` output is ordinarily low, also. This output will be read (sampled) by the KL25Z. To command the HCSR04 to begin a measurement cycle, we bring the `Trigger` input high (logic 1) for 10 microseconds and then bring it low again. (This low-high-low sequence is known as a "pulse.") In response to the `Trigger` pulse, the HCSR04 sends out a sequence of ultrasonic sound wave pulses and brings its `Echo` output high. The ultrasonic pulses bounce off hard objects and return to the HCSR04. Once the HCSR04 receives the echoed ultrasonic pulse, it brings its `Echo` output low. Thus, the width of the `echo` pulse corresponds to the amount of time that it takes the ultrasonic pulses to travel from the HCSR04 to an object and then back to the HCSR04. This waveform diagram illustrates the idea:



5. Let's say that we have the width of the `echo` pulse in units of microseconds. The speed of sound in air is 1,125 ft./sec. Determine the equation that converts the pulse width to the distance in inches from the HCSR04 to an object:

(If you don't want to derive this equation yourself, you'll find the derivation at the end of the lab.)

6. Now a few final, but important, details.

   The HCSR04 "gives up" and resets itself if it doesn't receive the echo of the ultrasonic pulses it emitted within 30 milliseconds. It can take the HCSR04 up to 60 milliseconds to reset itself. These details have consequences for the driver that you're about to write:

   - If the measured `Echo` pulse width is more than 30 milliseconds, then no measurement has occurred.
   - To ensure that the HCSR04 isn't in the middle of resetting itself, the driver must ensure that the `Trigger` signal is low for at least 60 milliseconds before beginning a measurement cycle.

7. Now, go to the mbed development site. Clone your LCD/temperature sensor project and name it "Rangefinder" or some name of your own choosing. (Right-click on the project name in the Program Workspace window pane and choose the Clone option.)

   Remove the temperature sensor code from your cloned project.

8. Now it's time to think about designing your device driver function. Here's the basic flow in pseudo-code form:

```
float measureDistance()
{
   create a timer;  // see the mbed's Timer interface
   reset the timer;
   make sure the HCSR04 is idle;
   send a trigger pulse to the HCSR04
   sample the echo signal from the HCSR04 until it goes high;
   start the timer;
   sample the echo signal from the HCSR04 until it goes low;
   stop the timer;
   read the timer value in microseconds;
   if the timer value exceeds the HCSR04's maximum range delay
      return -1.0;
   else
      return the distance in inches;
}
```

   You should read the documentation for the `Timer` interface, available here.

   Documentation for the `wait` function is here, where you'll find an example of `wait_us()`, which can be used to wait for a given number of microseconds.

9. Implement your device driver. You'll need to use the mbed's `DigitalOut` (trigger; `PTB9`) and `DigitalIn` (echo; `PTB8`) interfaces in the driver.

10. Write a main program that display the measured distance in inches once per second, to two digits after the decimal point, on the LCD display.

11. Connect the KL25Z to your host computer. Download and test your design.

12. Measure some distances!!!

13. You're now finished with the LCD display, the HCSR04, and the '244A. Carefully remove all the wires from the breadboard, placing them back into the clear plastic bag.

Derivation of the equation relating pulse width to distance from the HCSR04 to the target.

The speed of sound in air is 1,125 ft./sec. We convert this to units of inches per microsecond by multiplying by 12 and dividing by 10^6. (One second is 10^6 microseconds.) So, the speed of sound in air is 0.0135 in./microsecond. The sonic bursts travel from the module to the target device, and then back to the module. Therefore, our equation is

$$2D = 0.0135T$$

which simplifies to

$$D = 0.00675T$$

where the distance, D, is in inches and the time, T, is in microseconds.