Introduction

In this activity, students will learn the basics of operating an oscilloscope. This will cover basic, typical controls, but these vary from model to model, so you should consult the manual for your particular model of oscilloscope.

Materials List

1. Arduino Mega
2. USB Programming Cable
3. Programming Laptop
4. Oscilloscope with 2(or more) Probes

Basic Oscilloscope Operations

1. Basic Oscilloscope Parts
   1.1. Before you turn the scope on take a minute to look at the basic parts.
   1.2. The general layout for many scopes is a display screen (LCD or CRT), controls on the right, and 1 or more round BNC connectors for probes along the bottom. These connections can be used simultaneously and are usually referred to as channels.
   1.3. Look over the controls, most of the buttons and knobs will be labelled, but some may be blank, usually that means the screen will show a function for that button.

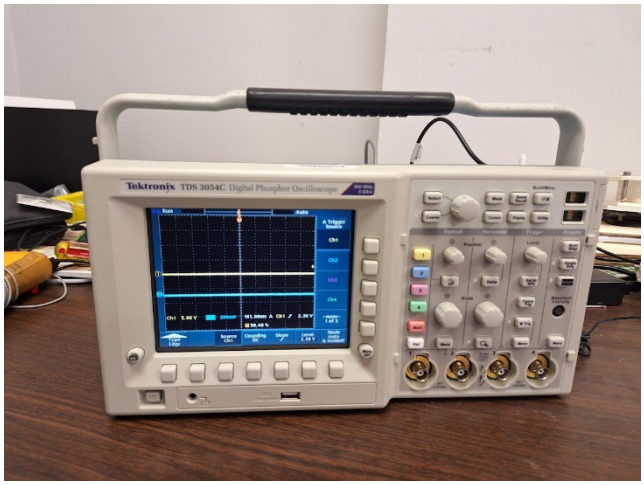

Figure 1: A typical oscilloscope. This scope has 4 channels and a color LCD display. The right portion of the scope is where most of the controls are located. The buttons near the screen do not have permanent labels because their functions change and are given on the display.

   1.4. In general, you should see a set of controls for vertical, horizontal, and trigger settings. Locate these controls on your scope.
   1.5. For this activity, now look at your probes. For this activity, we will use 2 probes so that we can look at serial TX and RX at the same time. The round connector is called a BNC connector; it slides onto the scope connector and locks in place with ~1/4 turn. The opposite end will have two clips, one for the ground and one for the signal. Usually, the ground will be a simple alligator plug, and on some probes, this is removable.

1.6. Looking at the probe, you should see several numbers that give you specifications for the probe. Common(but not always given) values are voltage (maximum voltage), capacitance in Farads (typically pF), max frequency range (MHz), impedance (usually in MOHm), and attenuation (such as 1x, 10x, 5x etc.)

1.7. We are going to use the scope like a multimeter, so we want a probe with a large impedance (like 10 MOhm).

1.8. The attenuation is how much the probe reduces the voltage it sends to the scope. This allows you to measure higher voltages without damaging the scope. For example, a 10x Probe will reduce a 50V signal to 5V. We need to ensure we know the probe attenuation to accurately measure voltages.

Figure 2: Oscilloscope probe, which has 10x attenuation, 10 MOhm impedance, and 10.8 pF capacitance.

1.9. Power scope on and wait for it to complete startup. The screen may give you a prompt like “Hit any button to continue”

2. Channel and Probe Settings

2.1. Connect a probe to the first channel and turn that channel on. It may be on by default, or you may have to hit a channel number button or the on button.

2.2. You want to find the settings menu for the channel.

2.3. The coupling should be set to DC, this means the probe is measuring voltage from its ground connector. AC coupling will center the voltage around 0.

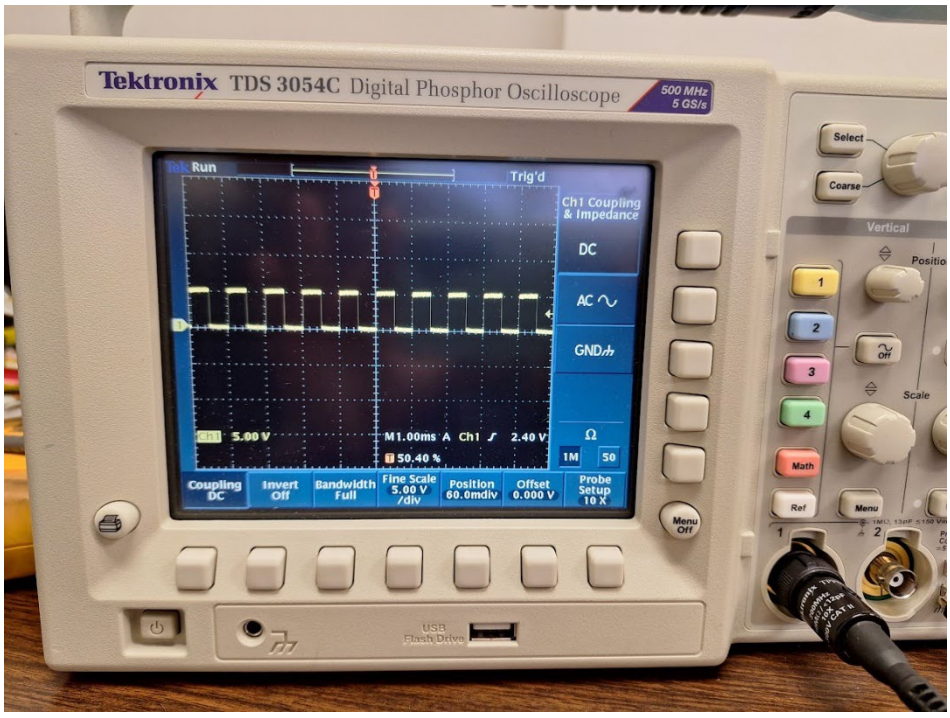

Figure 3: The channel settings. We are concerned about matching the attention (10x) to the probe, having the impedance set to “High” 1MOhm, making sure it is set to DC coupling. There may also be settings for current vs voltage probes (we are using voltage).

2.4. The impedance should be set to a value in the MOhms (50 Ohm settings are for connecting directly to other devices like antennae or signal generators).

2.5. Ensure the attenuation setting matches your probe (10x and 10x or 1x and 1x).

3. Horizontal and Vertical Scales

3.1. Now look at the display and look for the scale markings for the vertical and horizontal scales. Determine what your current settings are for the voltage and time axes.

3.2. Now locate horizontal and vertical control knobs. You should see two knobs for each one for position and one for scale.

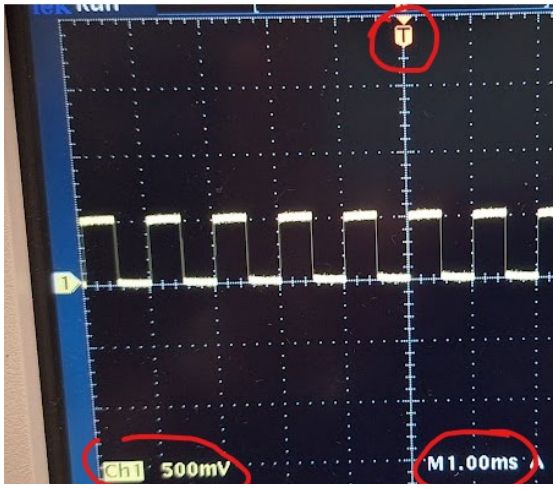

Figure 4: The current horizontal and vertical scales. These show the time and voltage for the large scale marks. In this case, the total display goes from -20V to +20V, and the horizontal axis goes from -5 ms to +5 ms. The red T is the trigger. Notice how the vertical scale has channel and color, because it can be changed for each channel.

Figure 6: Vertical Scale Controls, the left image shows a shared control used by 4 channels, selected via the channel button. The right image has individual controls for each channel.

Figure 7: Horizontal controls. Since the scale is shared there will usually only be a single set of these.

3.3. Use the scale knobs to set the scale for channel 1 to 1V and 1 ms for channel 1.

3.4. Now see if your scope has a "Probe Compensation" connection. This is a spot to clip your probes that has a square wave at a fixed frequency (0-5V at ~1 KHz is common).

3.5. Connect your probe to the signal (don't forget to connect the ground to a ground point).

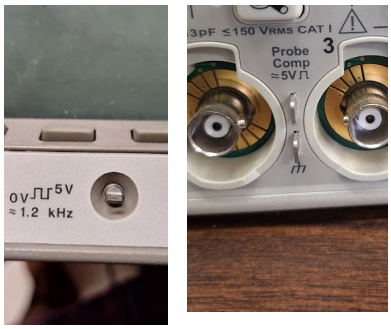

Figure 5: Probe Compensation connection points. The left scope gives the frequency, but the right does not. The lower clip on the right gives a place for connecting the ground clip

3.6. You should now see a square wave displayed on the screen of your scope at the matching frequency and voltage. If you do not see anything, check your vertical scale and position; the measurement may be off the screen or flattened.

3.7. Instead of square corners, you may see rounded edges or spikes on the pulses. That is caused by the capacitance of the probe. Some probes may have a small screw that allows you to adjust the capacitance to correct that, but it should not affect our activity.

3.8. Use the position and scale knobs to change the displayed signal. Notice you can move the vertical position so the center is not always 0V, but there will be a 0V indicator on the display. The 0 on the time axis is determined by the "trigger," which we will now discuss.

4. Time and Trigger Settings

4.1. Unlike voltage, there is no 0 point on the time scale. For a constantly repeating signal like the square wave, we could plot from an arbitrary point in the wave, and it would look the same. But what about a signal that only lasts 1 ms sent every second? In such a case, we want the scope to plot a much smaller time around the start of the signal. That is what the trigger menu lets you do.

4.2. Find the trigger controls and the trigger menu. The trigger level knob raises and lowers the voltage the trigger occurs. The trigger menu contains the other settings for the trigger.

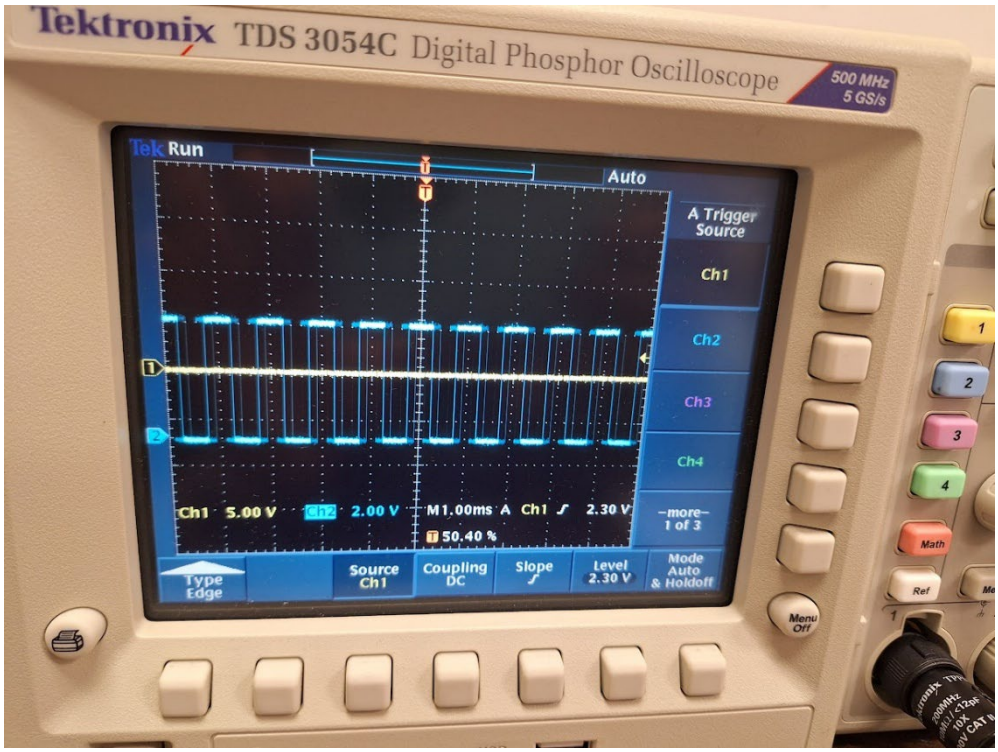

Figure 8: A typical trigger menu. The trigger is currently set to trigger on Channel 1 as it rises(see the slope setting) above 2.30V(level). No signal is currently connected to Channel 1. Because of this, no trigger is occurring causing the rolling signal shown on channel 2 as the scope attempts to display a signal.

4.3. Open the trigger menu and look at the settings. The source is the channel that will cause the trigger; the scope may also have other options, such as External (usually a separate connector on the back) or AC (trigger off 60 Hz AC power).

4.4. The slope will either be a rising or a falling edge. Does the trigger happen on an increase or a decrease?

4.5. Mode and holdoff settings control how long the signal will be displayed before a second trigger occurs and overwrites the signal.

4.6. Raise the trigger voltage above 5V and notice what happens to the displayed signal. Lower below 0V and do the same. Restore the trigger normal value.

4.7. Try changing the trigger to rising and falling edges. See what part of the square wave lines up with the trigger.

4.8. Disconnect Channel 1 from the square wave and connect Channel 2 to the square wave, leaving the trigger on Channel 1. What happens?

4.9. Now switch the trigger to channel 2.

Measure Serial Signals

1. Now you are ready to look at serial signals with Arduino.
2. Connect Channel 1 of the scope to pin 1 (TX) and Channel 2 to pin 0 (RX).
3. Now write a simple program to output a single character to the Serial port at 9600 Baud every 1 second. **Make sure you use print() and not println().**
4. Now set your scope so both channel 1 and 2 voltage scales to 5V, and the time scale to 1 ms.
5. Set your trigger to a falling edge on Channel 1, and you should see something like the image to the right. **Remember, 9600 baud means 9600 bits per second. So ~10,000 bits per second. A single serial byte is 1 Start bit, 8 data bits, and 1 Stop bit (10**

Activity0308.ino

```
1  void setup() {
2    // put your setup code here, to run once:
3    Serial.begin(9600);
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8    Serial.print("a");
9    delay(1000);
10 }
11
```

Figure 9: A simple program to output a single character to the serial port.

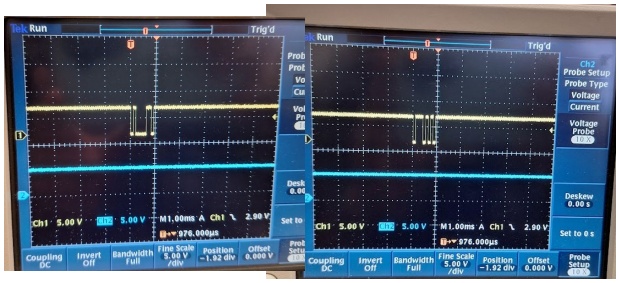

Figure 10: Two different transmitted characters, the image on the left shows an "a" and the right is "W"

bits). So, we should expect the signal to be 1 ms long.

6. Try changing the transmitted character and see what happens to the data. You may need to zoom in on the time scale to see the bits changing clearly.
7. Now change the baud rate to 4800 and 19200 and look at the signal. What happens?
8. Change the baud rate back to 9600. Change the **print()** to **println().** Remember that the println adds a newline and carriage return character to the end of the transmission. You should see 3 bytes transmitted, though the gap between bytes will be very small.
9. Try reducing the delay in the loop so you can see multiple transmissions on the scope. Try a delay of 10 (you may need to zoom out).
10. Reset your time and voltage scales back to 5V and 1ms.
11. Now change your trigger settings to trigger off of channel 2.
12. Open the serial monitor and try typing characters into the text box, and you should see similar pulses on the receive line. **Note that the transmission is not sent until you hit enter.**
13. Experiment with different messages and baud rates in the serial monitor.
14. Now, modify the program so that it checks to see if it has received any data via Serial. It should then read the data and print it back out the serial port. It should repeat this until no more data is available.

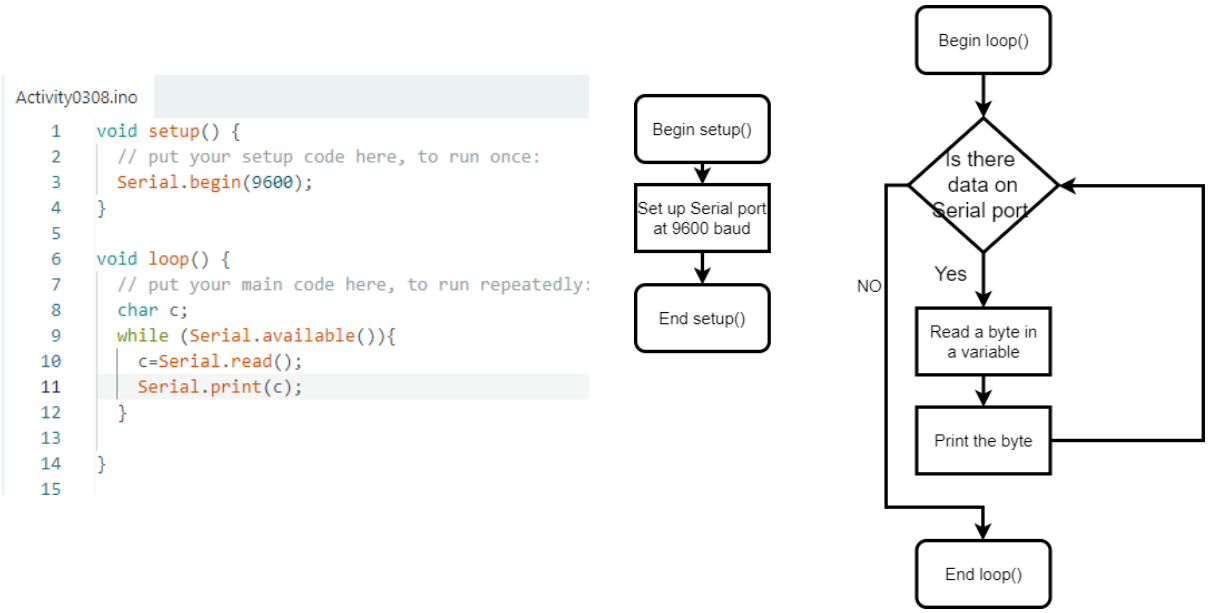

Figure 11: Sample code and flowchart of a simple program to listen for data on the serial port and then echo it back,

15. Upload your code and try sending messages using the Serial Monitor. You should see the Arduino repeat back the character you send. You should also see both messages on the scope as shown in the figure below, with the retransmission slightly delayed.

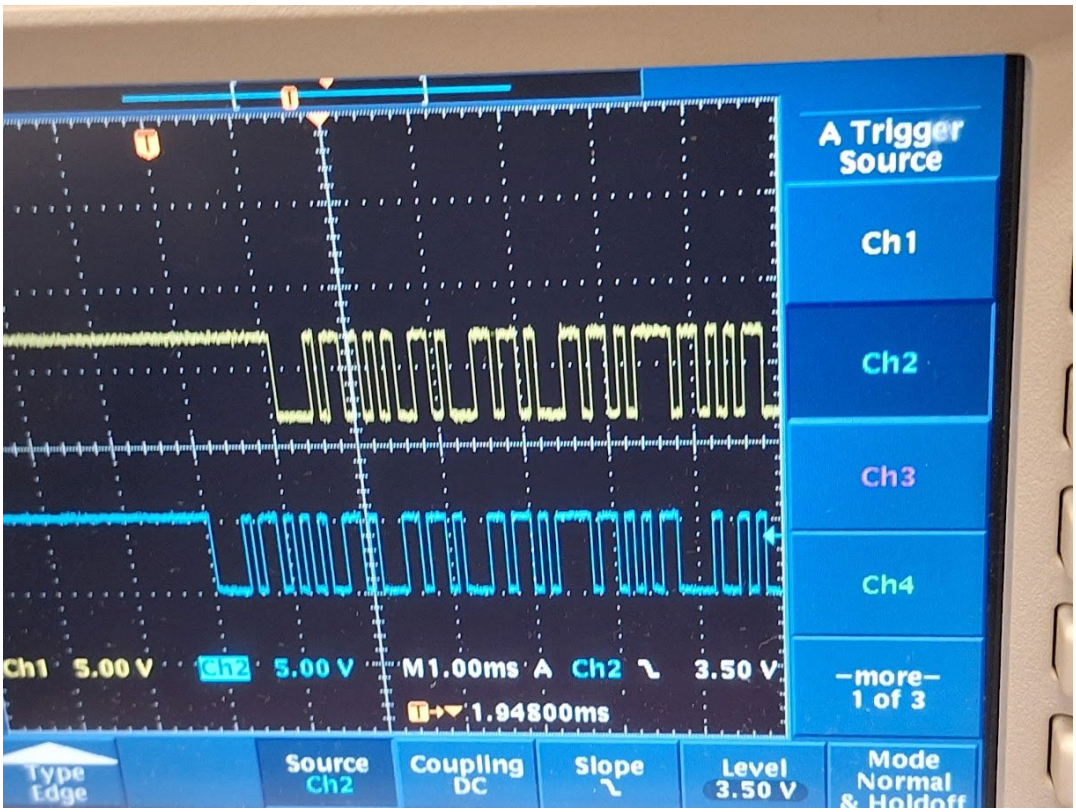

Figure 12: The original bytes on the RX line are shown in blue, and the retransmission out the TX line is in yellow. Notice how the retransmission starts ~1ms (about 1 byte) after the RX.