Summary:

Students will familiarize themselves with the Arduino Mega's analog-to-digital converter by building simple circuits on a breadboard, writing programs that introduce the ADC pins and functions, and plotting data in Excel.

Materials:

Each student should have the following materials, equipment, and supplies:

- Computer with Arduino IDE and Excel installed
- USB-AB programming cable
- Arduino Mega microcontroller
- Digital multimeter
- Wires/Jumpers (recommend 22 gauge solid wire) for breadboarding
- Breadboard
- 5k potentiometer (Actual value not important)
- 5k resistor (Actual value not important_
- P-N junction diode (1N457)

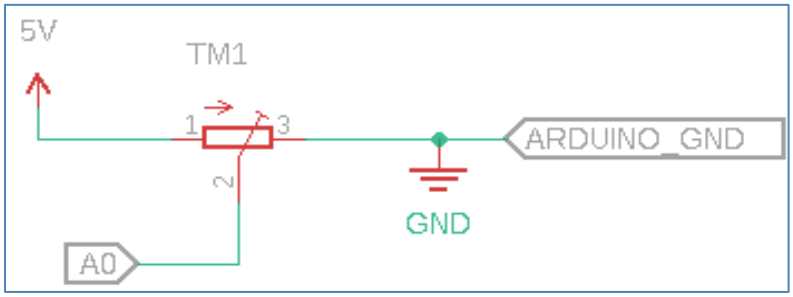
Procedure:

Example 1 – Potentiometer

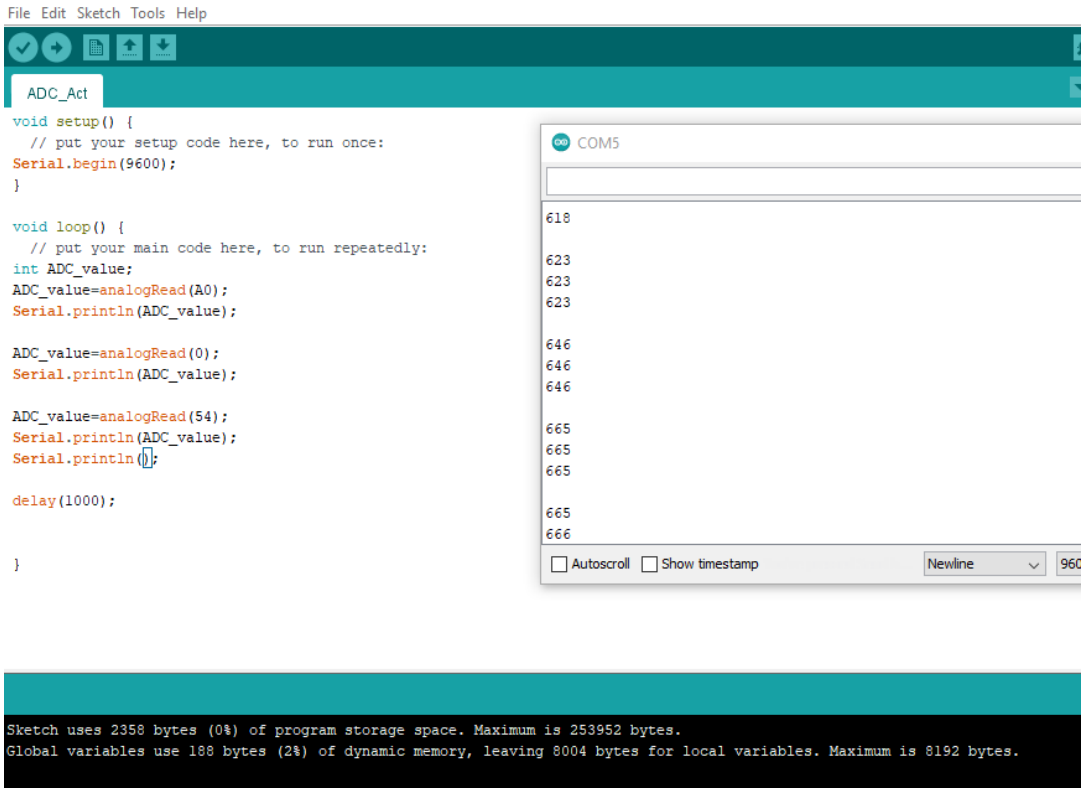
1. Write a simple program that simply reads the voltage at Analog pin 0 writes the ADC value to the serial port.


```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
int ADC_value;
ADC_value=analogRead(A0);
Serial.println(ADC_value);


delay(1000);
}
```

2. Before we hook in anything up let's look at the output without anything connected. You will likely see the value bounce around bit and sort of settle around a random number. This is similar to why we use pullup resistors for the digital pin, when the pin is not connected to actual signal the behavior is unpredictable.
3. Now we want to connect a signal we can vary to the pin
4. Follow the schematic in figure 1 to connect the potentiometer to the microcontroller.



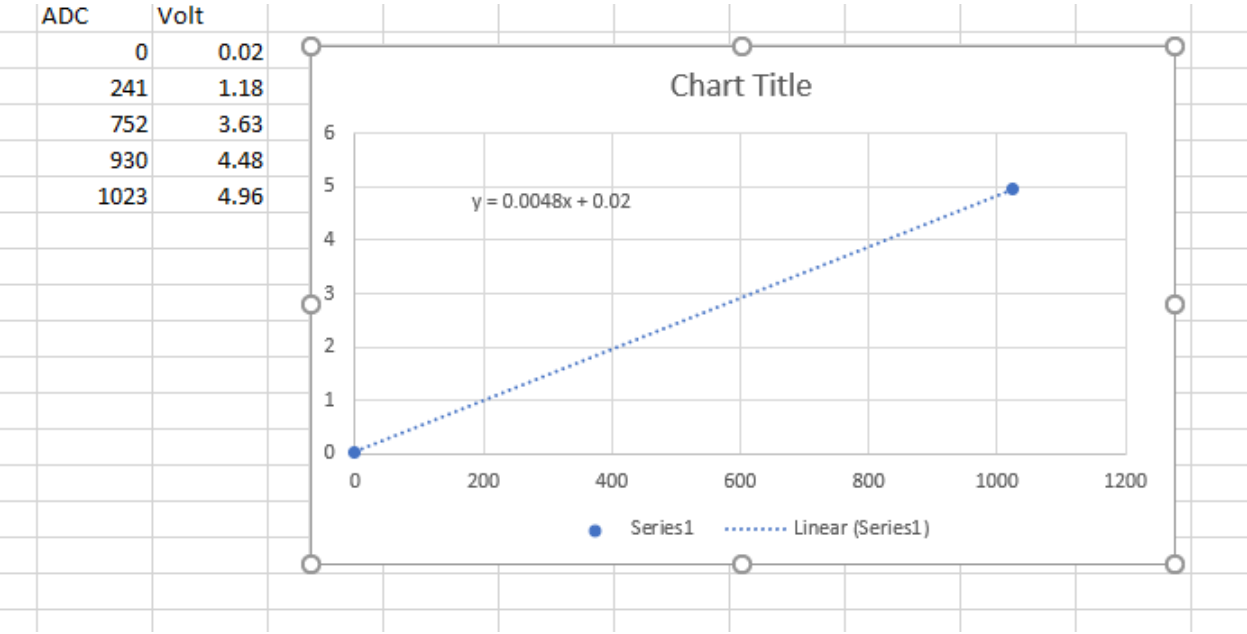
5. Slowly turn the knob or screw of the potentiometer and observe the change in ADC.
6. Next, we should note there are actual several ways to read the same pin.
   In the code below these are all equivalent ways to read pin A0. We can use the constant A0, the actual pin number 54, or if send 0 the Arduino will realize we are wanting A0 since pin 0 is not an analog pin.



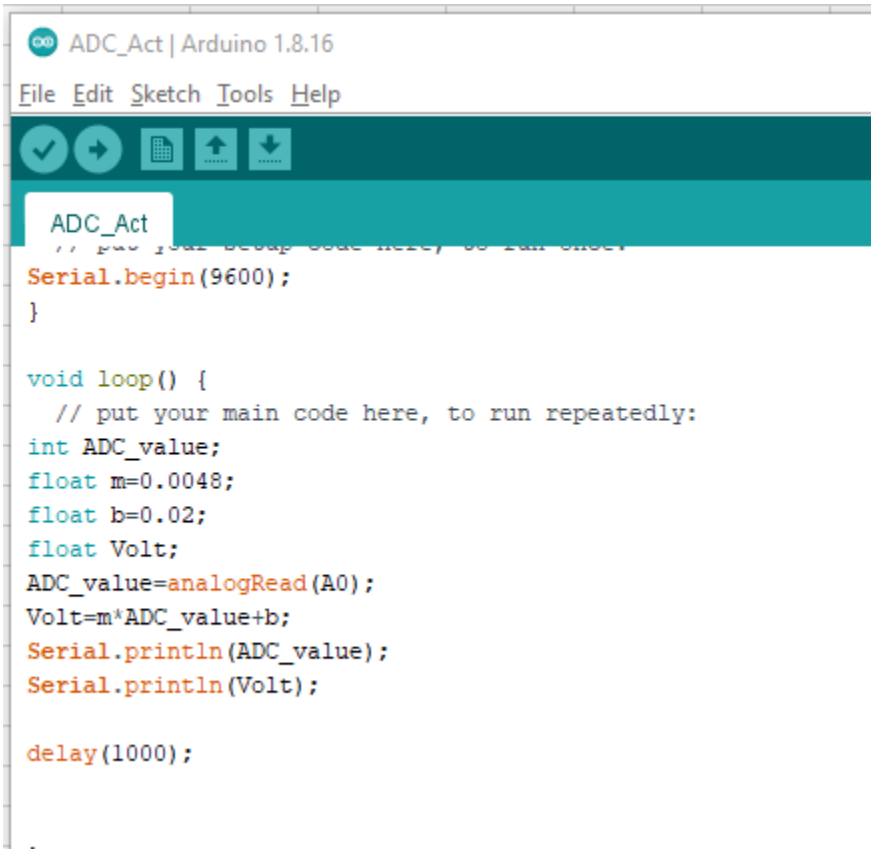
7. Hook up a multimeter to so can measure the voltage at pin A0. Do not forget to connect your negative lead to the Arduino ground.
8. Adjust the potentiometer until the ADC value just reads 0 and record the voltage, then slowly turn the knob and record at least 5 data points between 0 and 1023. It is important to note that the actual ADC range is not exactly 0 to 5V but instead slightly narrower.

ADC	Corresponding voltage
0	
1023	

9. Plot your data in Excel
10. You should observe a linear relationship between the voltage and ADC value.
11. If you fit the line to your data should get the equation for converting form ADC to

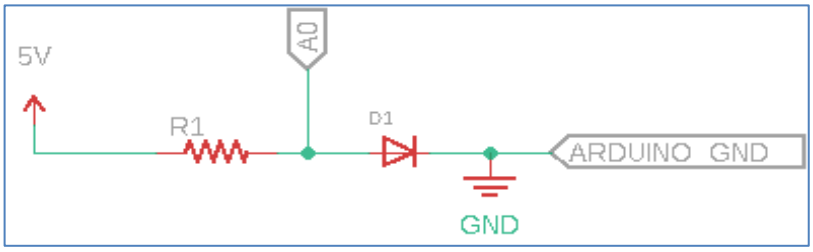


12. Write a function in your code that takes ADC value read and calculated the corresponding voltage using the equation you just determined. Note because we want the voltage, slope and intercept to be decimals we should declare them as floats


ADC_Act | Arduino 1.8.16
File Edit Sketch Tools Help
ADC_Act
Serial.begin(9600);
}
void loop() {
// put your main code here, to run repeatedly:
int ADC_value;
float m=0.0048;
float b=0.02;
float Volt;
ADC_value=analogRead(A0);
Volt=m*ADC_value+b;
Serial.println(ADC_value);
Serial.println(Volt);
delay(1000);

Example 2 – Diode

1. Follow the schematic in figure 5 to connect the diode in series with the resistor to the microcontroller
    a. Be careful not to connect the diode directly to the microcontroller. Without the resistor, you will experience a current surge which can overheat your circuit and can be potentially dangerous to you.



2. Use the same program from example 1 to read the ADC. If connected correctly, the diode should not be hot. Lightly pinch it with your fingers and observe the changes in the serial monitor. You should see the ADC change by approximately 1 to 5 ADC values.
3. This is a silicon P-N junction diode and based on its material properties the voltage should fall near 0.7V.