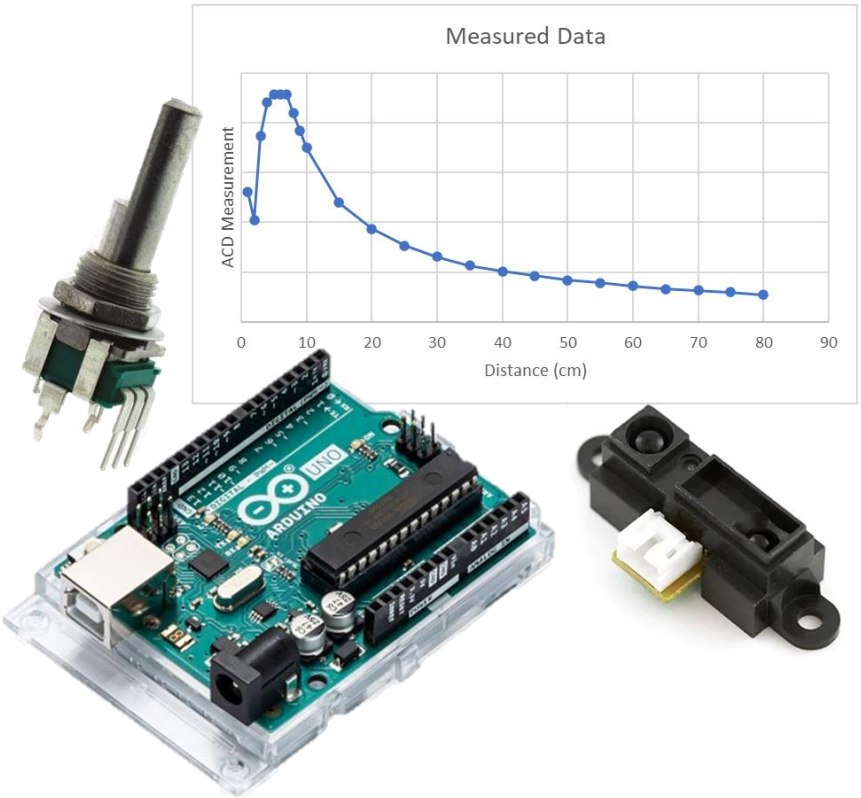
Introduction to Arduino – Analogue Sensors

Ben Taylor
Shuhei Miyashita and Dana Damian



Learning Outcomes

Practical	Be able to interface an analogue sensor to a microcontroller and measure the output voltage of the sensor.
Communicate	Be able to describe the need for sensor calibration for real world applications and describe how this can be done for simple sensors.
Practical	Be able to write and apply code to decode the sensor voltage into meaningful measurement information and apply calibration data to the sensor measurement.

Contact: B.P.Taylor@Sheffield.ac.uk Pre: -%, During -%, Post 100%

Date:

Introduction to Arduino – Analogue Sensors: In-Laboratory Activities

1 Aims and Objectives

The aims of this laboratory session is to provide you with hands on experience of reading analogue sensor voltages into the Arduino device, and to convert these into meaningful measurement variables.

During the laboratory session, you will interface the Arduino device to different sensors and write code to measure the incoming voltages. You will write routines to convert the samples voltage values into meaningful measurement information. You will investigate the need for sensor calibration and apply appropriate compensation to your measurement to minimize measurement inaccuracies.

2 Names of collaborators for the experiment

Use this space to record the member of the group you are conducting the experiment with.

Name of Lab partner(s)	Lab Desk Number

3 Background

Sensors are used in real world applications to measure physical phenomena and convert these into an electrical characteristic, as illustrated in Figure 1.

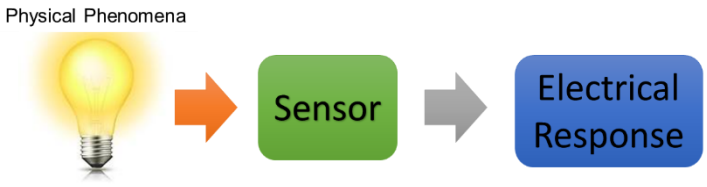


Figure 1. Measuring Physical Phenomena with a Sensor

Most 'raw' sensors do not produce a voltage on their output, but instead the output of the sensor has a modified electrical characteristic, in response to a change in the physical phenomenon being measured. Table 1, illustrates some typical examples of different sensor types used in real world applications, and the type of electrical characteristic the output of each sensor exhibits to a change in the measurement variable.

Phenomena	Sensor Type	Output
Temperature	Thermo couple	Voltage
	Thermistor	Resistive
Light	Photodiode	Conductance
Sound	Microphone	Voltage or capacitive
Force/Pressure	Strain Gauge	Resistive
	Piezoelectric Transducer	Capacitive
Displacement	Potentiometer	Resistive
	LVDT/Resolver	Analogue Phase/Frequency

Table 1. Examples of different sensor types and their output characteristics.

It can be seen, from Table 1, that the 'raw' output of most of the sensors, listed, do not exhibit a voltage output. For those sensors that have a voltage output, the output voltage is not generally of sufficient magnitude and impedance, to be sampled by the system's analogue to digital converter. As a result, signal conditioning is required to interface the vast majority sensors to a microcontroller system.

4 Laboratory Exercises

Before starting the laboratory exercises, you should review the contents of the Mechatronics KIT Information folder, located in the Practicals content area of the Blackboard site for the Mechatronics course.

4.1 Exercise 1: Read a Potentiometer

The aim of this exercise is to use the example code provided to read an analogue input, in this case generated from a potentiometer, to generate an output signal that will change the brightness of an LED, and stream some data to the Serial Monitor.

Procedure:

1. From the Resources folder in the Blackboard folder for this experiment, download the potentiometer test program, POT.ino
2. Build the potentiometer test circuit shown in Figure 2.
    - Connect left hand pin of the potentiometer to 5V and the righthand pin to the ground.
    - The middle pin of the potentiometer is connected to the analogue A0 pin.
    - The anode of the LED is connected to the pin 11, via a 470Ω resistor and the cathode to ground.

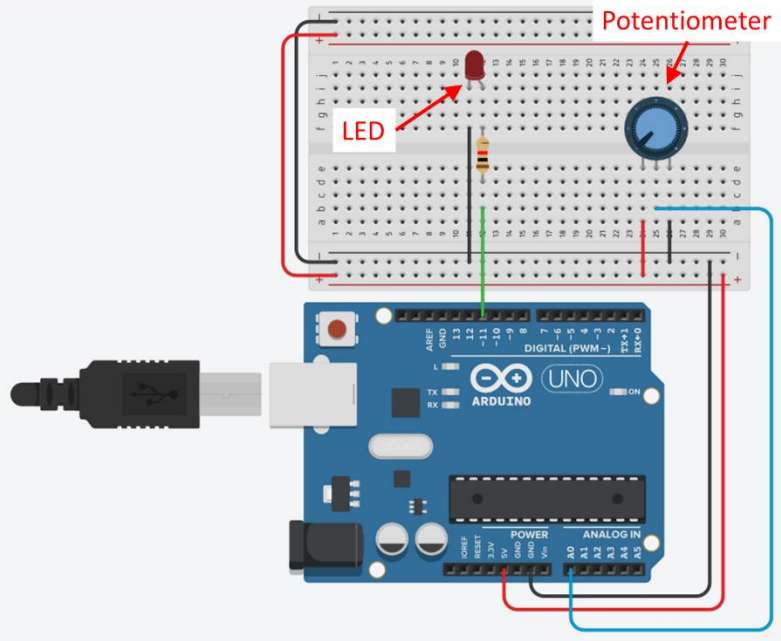


Figure 2. The test circuit used with the potentiometer test program

The ADC inputs to the Arduino have 10 bits of resolution (i.e. 1024 different values). By default they measure from ground (0V) to 5 volts. That means that the 0-5V range of the potentiometer will be proportionally mapped to the 0-1023 digital values range in the microcontroller.

3. Upload the program to the Arduino, then open the serial monitor and observe data being streamed to the terminal.
4. Rotate the knob of the potentiometer and observe the range that the data spans.
5. Observe the rightness of the LED as you rotate the potentiometer.

This example program uses a the map() function to 'map' the ADC measurement values between 0 and 1023, to a range of 0 to 255. This new range is required to write the PWM signal to change the LED brightness – PWM will be covered in the next lab session, so for this example, just assume this is an analogue output from the Arduino.

Look at the code and see how the map() command is used. The Arduino language reference for the map command can be found at:

https://www.arduino.cc/reference/en/language/functions/math/map/

4.2 Exercise 2: Calibrate Potentiometer Angle

The map function can be used to provide a linear calibration between a measured value and a real world parameter. In the case of the potentiometer test circuit, the potentiometer has a rotational range of 300^{O}.

The aim of this exercise is to use the map command to produce a calibrated potentiometer angle, using the ADC measurement as an input, and write the potentiometer angle to the serial monitor.

Procedure:

1. The starting point for this exercise is the Potentiometer test circuit, you have constructed, and the Potentiometer test program from the previous exercise.

2. Add another map() function to the sketch to calibrate the potentiometer angle between -150 and 150 for an ADC measurement input of 0 to 1023.
3. Use the Serial.print() and Serial.println() functions, (in a similar way to that shown on POT.ino), to write the data to the serial monitor.

(See the serial communications functions page from the Arduino language reference pages for a description of the Serial.print() and Serial.println() functons:

https://www.arduino.cc/reference/en/language/functions/communication/serial/)

Note: *This Exercise will need to be demonstrated in the final video assignment. See the Introduction to Arduino assignment brief on the Blackboard site for more details.*

4.3 Exercise 3: Measure the Output of the Sharp IR Distance Sensor

The aim of this exercise is to interface the Sharp IR distance sensor with the Arduino, record the measured voltage as a function of the distance to a target, and observe any constraints for the operation of this sensor.

Procedure:

1. The starting point for this exercise is the Potentiometer test circuit, as used in the previous exercises and shown in Figure 2.
2. This circuit should be modified, by adding the Sharp IR distance sensor, as shown in Figure 3. Where:
    - Pin 1 of the distance sensor is the output voltage and should be connected to pin A1 of the Arduino
    - Pin 2 is the ground connection
    - Pin 3 is connected to the +5V supply from the Arduino

For more details on concerning the pin connections for the Sharp GP2Y0A21YK0F distance measurement sensor, see the data sheet for the sensor on the Blackboard site:

- ACS231 Blackboards Site>>Mechatronics Kit Information>> Component Data Sheets and Technical Documentation

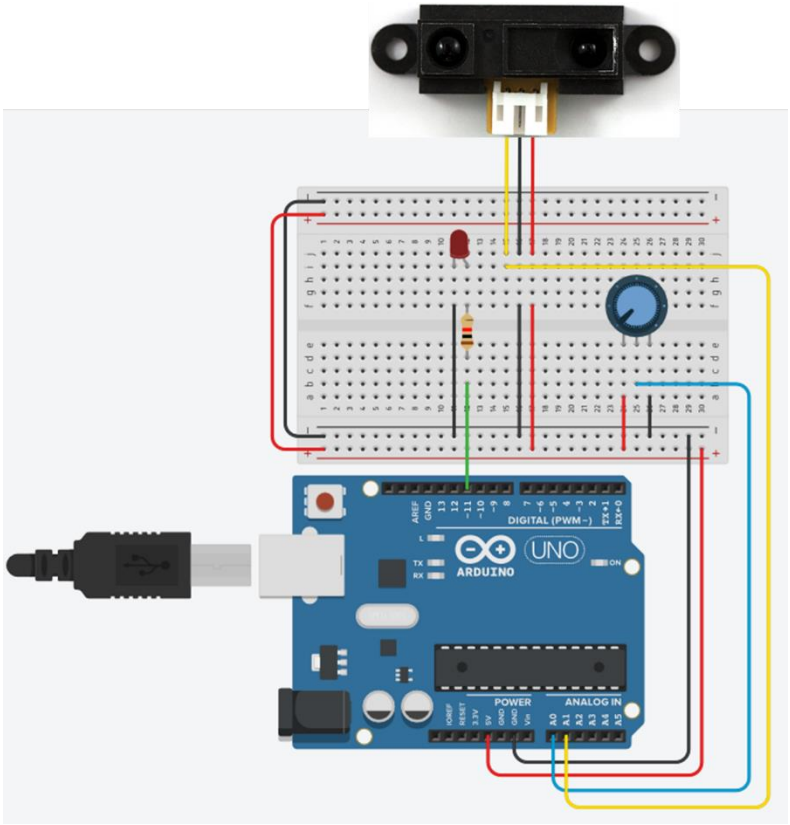


Figure 3. The Sharp IR distance sensor added to the potentiometer test circuit

Note: The LED and Potentiometer should be left in the circuit, because you will need them for later exercises.

3. Write an Arduino sketch to read the analogue measurement value from the IR distance sensor and display the reading on the serial monitor. (It is acceptable to re-download the POT.ino sketch and use this as a template for this exercise.)

Once you have the measured value of the distance sensor being streamed to the serial monitor, you will be required to plot the distance sensor measurement value against distance measured, (with a ruler/scale). This characteristic will be used in Exercise 5 to calibrate the distance sensor to produce a real world distance, (in centimetres), from the distance sensor ADC measurement.

During the development of this exercise, I briefly documented the setup I used to achieve this and have placed this document in the Blackboard for this laboratory. The “Calibrating the IR sensor” document contains several pointers for physical setup, software implementation and expected characteristic.

4. Perform an experiment to characterise the Sharp IR sensor. The results from this experiment should be a graph of the ADC measurement value from the Sharp IR distance sensor, against the target distance. (See the Experimental Results section of the “Calibrating the IR sensor” document for a suggestion for distance point spacing).

You should plot your results in a computer package, such as Excel or MATLAB, with the x-axis as the distance measured, and the y-axis as the ACS measurement value. You must record the data values, because you will need them for Exercise 5.

(**Note:** on your plotted graph, you should include a title and axes labels for both the x-axis and the y-axis)

5. Do not disassemble your experimental setup, because you will need it for Exercise 5.

Note: *This Exercise will need to be demonstrated in the final video assignment. See the Introduction to Arduino assignment brief on the Blackboard site for more details.*

4.4 Exercise 4: 1-D Lookup Table Example

The characteristic that you have recorded in the previous exercise is non-linear, and consequently, cannot be suitably approximated using the map() function, because this function will only be used on linear characteristics.

A look-up table, LUT, is a function that can be used map an input variable through a non-linear characteristic, (generally based on experimental data), to find a corresponding output, as illustrated in Figure 4. The look-up table uses a search algorithm to find the most appropriate output for a given input.

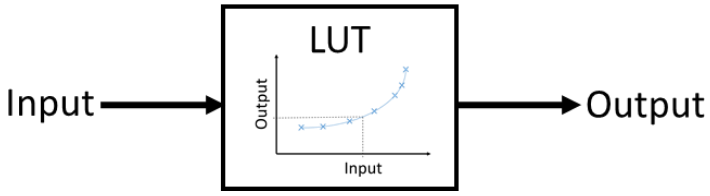


Figure 4. Block Diagram representation of a look-up table, LUT.

During this exercise, you will see how a simple look-up table function can be used to map the input from a potentiometer, through a non-linear function, (made-up from experimental data), to generate the desired output value.

The document “1-D Lookup Table”, located in the blackboard folder for this exercise, contains a brief description of the map() function detailing its limitations, and the description of a simple 1-D lookup table implementation, with example code.

During this exercise, you should read the 1-D lookup table document, download the code, run the Arduino sketch, and observe the operation of the lookup table implementation.

Procedure:

1. Read the document: “1-D Lookup Table” located in the blackboard folder for this experiment
2. Download the 1-D_Lookup_Example.ino Arduino sketch, from the Blackboard folder for this experiment
3. Ensure the circuit is wired as shown in either Figure 2 or Figure 3 of this worksheet. (Note: Figure 2 is the same circuit as shown in the “1-D Lookup Table” document.)
4. Run the Arduino sketch and observe the output on the serial monitor and the behaviour of the LED.

4.5 Exercise 5: Use the 1-D Lookup table to calibrate the IR sensor

The aim of this exercise is to use the 1-D lookup table implementation to map the non-linear measurement characteristic of the IR sensor into a real-world distance measurement. During this exercise, you will be required to incorporate the 1-D lookup table implementation into the IR sensor measurement code you have written for Exercise 3. The measurement

characteristic you have recorded for the IR sensor will be used to form the xElements and yElements lookup table arrays, required for the implementation.

Procedure:

1. Before proceeding, review the “Implementation of the Simple 1-D Look-Up Table” section of the “1-D Lookup Table” document, and ensure you can identify the code elements from the example code, referred to in the lookup table document.
2. Save a copy of your code from Exercise 3 with an appropriate name for this exercise.
3. Copy the global namespace elements from the 1-D_Lookup_Example sketch into your Exercise 5 code.
4. Change the xElements and yElements arrays to match the data you have recorded at the end of Exercise 3.

The ADC measurement values should form the xElements array, and the distance values should form the yElements Array. Remember that you will need to ensure that xElement values must increase with increasing index, and that and changed you make to the ordering of the xElements array must be reflected in the ordering of the yElements array.

5. Copy across the required lookup table elements from the from the setup() section of the 1-D_Lookup_Example sketch into your Exercise 5 sketch.
6. Copy the two lookup table functions, located after the loop() function, in the 1-D_Lookup_Example sketch, to the end of your Exercise 5 sketch
7. Modify the loop() function of your Exercise 5 code to implement the lookup table function on the measured value from the distance sensor.
8. Modify your code to display the analogue reading from the distance sensor and the measured distance, in an appropriately formatted message, onto the serial monitor terminal.

(An example of an appropriately formatted message can be found in the 1-D_Lookup_Example sketch – remember this will need to be modified for your application.)

9. Using the experimental setup from Exercise 3, validate your calibration of your system, comparing the measured data your have streamed to the serial monitor with the experimental distance scale.

Note: *This Exercise will need to be demonstrated in the final video assignment. See the Introduction to Arduino assignment brief on the Blackboard site for more details.*

5 Acknowledgements

This document was written by Ben Taylor, with contributions from a number of authors: Dana Damian, Shuhei Miyashita, and Kilian Marie.

Introduction to Arduino:
Post-Laboratory Assignment

There is a video assignment for the three Introduction to Arduino laboratory exercises. The exercises you are required to demonstrate in the video are:

- **Digital I/O - Exercise 3: LED Pattern**
- **Digital I/O - Exercise 4: Switch Debounce**
- **Analogue Sensors - Exercise 2: Calibrate Potentiometer Angle**
- **Analogue Sensors - Exercise 3: Measure the Output of the Sharp IR Distance Sensor**
- **Analogue Sensors - Exercise 5: Use the 1-D Lookup table to calibrate the IR sensor**
- **PWM Control of Actuators - Exercise 2: Controlling a Standard Servomotor**
- **PWM Control of Actuators - Exercise 3: Controlling a DC motor, using a driver board**

See the Introduction to Arduino assignment brief on the Blackboard site for more details.