Experiment A8
Digital Sensors
Procedure

Deliverables: checked lab notebook, brief tech memo

Recommended Reading: Section 6.4, Section 14.1, and Ch. 17 of the textbook

Overview

Analog sensors output a single voltage that is related to the measured parameter via some calibration formula. This voltage output can be digitized and recorded by an analog-to-digital (A/D) converter, which then forwards the measured values to some computer or microcontroller where it can be saved in memory.

Digital sensors directly output a digital signal containing the measured parameter. This eliminates the needs for an extra A/D. In many cases, it also eliminates the need for a calibration procedure. These advantages have made digital sensors extremely popular.

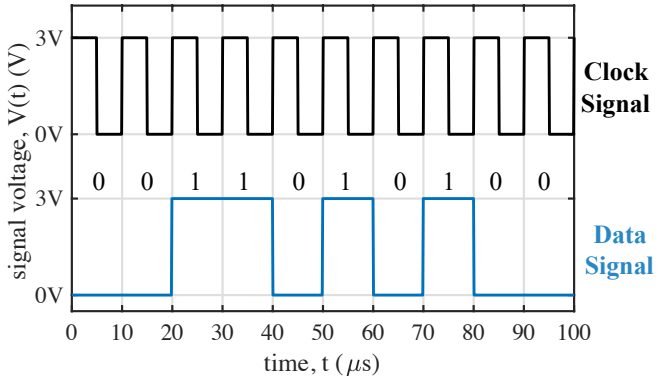

Figure 1 – (Top) The rising edge of a clock signal determines where one bit ends and a new bit begins. (Bottom) The 10-bit binary number 0011010100 is sent as a serial packet.

The measured value from a digital sensor is typically sent as a series of voltage pulses known as a *serial packet* or *bit stream*. A serial packet is plotted above in Fig. 1. In addition to the packet containing the measured data, a clock signal tells the computer or microcontroller where one bit ends and a new bit begins. This clock signal can be internal in the microcontroller (asynchronous), or it can be broadcast on an additional shared wire (synchronous).

In Part I, we will use an *asynchronous* digital sensor that does not have an extra wire for the clock signal.

In Parts II and III, we will use *synchronous* I^2C sensors that have a wire for data (SDA) and a wire for the shared clock signal (SCL). The I^2C communication protocol is so widely used, that the Arduino UNO has dedicated SDA and SCL pins, shown on the right side of Fig. 2.

Part I: Digital Humidity and Temperature Sensor (DHT11)

The DHT11 is an inexpensive digital humidity and temperature sensor that we will be connect to the Arduino. It uses asynchronous data transmission, so there is no connection for a clock signal.

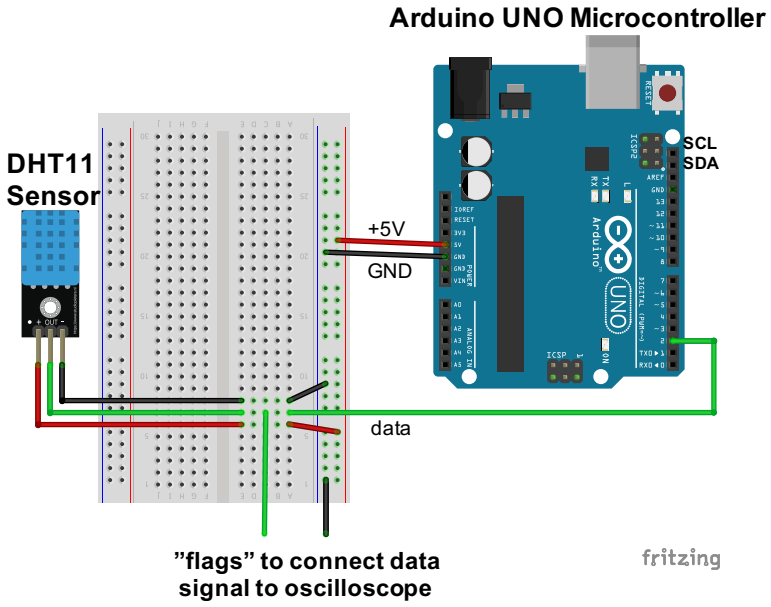

Figure 2 – The DHT11 sensor and Arduino UNO Microcontroller work in tandem with breadboard.

Procedure

Important: Do NOT turn on the breadboard. You will use the Arduino UNO to provide power to the sensor.

1. Connect the Arduino UNO to the lab computer via the USB cable. You should see a green LED light up on the Arduino.
2. Use red and black jumper wires to connect the +5V and GND pins on the Arduino to the vertical bus lines on the breadboard, as shown in Fig. 2. Use the orange handheld DMM to verify that it is providing +5V of power.
3. Make the following connections via the breadboard, as shown in Fig. 2. Use female-to-male jumper wires make the following connections:
    a. – on DHT11 → GND on Arduino via the breadboard
    b. + on DHT11 → 5V on Arduino via the breadboard
    c. Data on DHT11 → Digital input pin 2 on the Arduino via the breadboard
4. Open the Arduino IDE software.
5. The Arduino will not know how to communicate with the DHT11, unless you add the DHT11 code library to the Arduino IDE:
    a. Download the DHT11 Library from the A8 webpage.

   b. Go to "Sketch" > "Include Library" > "Add ZIP Library...", navigate to the file you just downloaded "DHT.zip", and select it.
   c. If it gives you an error that the library "already exists", don't worry. That simply means the student before you already loaded the library.

6. Download the "DHT11 Example" Arduino code from the A8 webpage, and open it in the Arduino IDE software.
7. In the Arduino IDE software, go to "Tools" > "Port" and select the COM port that says "(Arduino/Genuino Uno)" next to it.
8. Click the check mark at the top of the Arduino program to check the code for errors.
9. Press the arrow button to compile the program and send it to the Arduino.
10. Go to "Tools" > "Serial Monitor" (or press "Ctrl + Shift + M") to view the output from the "Serial.print()" commands at the bottom of the screen. Make sure the baud rate is set to 9600. You should see the measured temperature and relative humidity printed repeatedly.
11. Cup your hand around the sensor for about a minute. You should see the temperature and humidity increase.
12. Use Channel 1 on the oscilloscope to measure the digital data signal from the sensors. Connect the mini-grabbers to the appropriate places in the circuit by connecting the mini-grabbers to "flag" wires in the correct rows of breadboard, as illustrated in Fig. 2.
13. Turn on the oscilloscope (if it's not already on) and enter the following settings:
    a. Press the yellow CH1 button, and set it to "AC coupling".
    b. Turn the Vertical Scale knob to 2.00 V
    c. Turn the Horizontal Scale knob to 200 μs.
    d. Push the Horizontal Position knob inward to center the orange pentagon at the top of the screen.
    e. Press the Trigger Menu button (top right corner) and choose the following:
        i. Type: Edge
        ii. AC coupling
        iii. Rising edge trigger
        iv. Turn the Trigger Level knob (top right corner) to approximately 1 Volt.
    f. Press the "Menu Off" button near the bottom right corner of the screen.
14. If you set up everything correctly, you should see a square "pulse train" appear on the oscilloscope every 3 seconds. This is the serial packet being sent from the sensor to the Arduino.
15. **Demonstrate the working system to the TA.**
16. Disconnect the DHT11 and the Arduino UNO from the breadboard. Press the "default setup" button at the bottom of the oscilloscope.

Part II: Thermocouple with Compensator/Amplifier

Background

A thermocouple is a complicated transducer capable of measuring temperature across a large range. Shown in Fig. 3 below, the thermocouple consists of two wire made from different metals that are soldered or welded together, forming a "junction" at the tip of the probe. The junction at the tip is known as the "hot junction", while the opposite ends of the two wires are known as a "cold junction". This temperature gradient along the length of the two wires creates a voltage gradient, as well—a phenomenon known as the *Seebeck effect*.

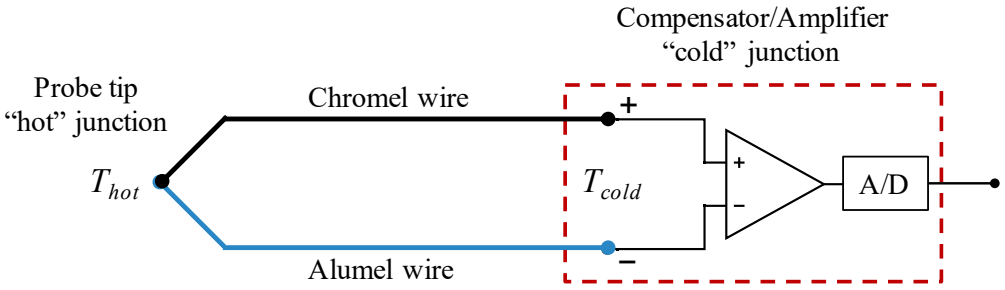

Figure 3 – An illustration of a Type K thermocouple connected to an compensator/amplifier.

The temperature induced gradient along the bottom wire is

$$\Delta V_1 = -S_1 \cdot (T_{hot} - T_{cold}), \quad (1)$$

where S_1 is the Seebeck coefficient of the bottom metal wire. Similarly, the temperature induced gradient along the top wire is

$$\Delta V_2 = -S_2 \cdot (T_{hot} - T_{cold}), \quad (2)$$

where S_2 is the Seebeck coefficient of the top metal wire. Applying Kirchhoff's voltage law to the circuit, we find that the output voltage at the cold junction is

$$V_{out} = (S_2 - S_1) \cdot (T_{hot} - T_{cold}). \quad (3)$$

In this lab, we will use a Type K thermocouple, with wires made of *Alumel*, an alloy of nickel and aluminum with S_1 = -17.3 µV/°C, and *Chromel*, an alloy of nickel and chromium with S_2 = 21.7 µV/°C. The probe tip temperature is given by T_{hot}. Solving for T_{hot} and plugging in values for S_1 and S_2, we find that

$$T_{hot} = T_{cold} + \frac{V_{out}}{40\mu V/°C}. \quad (4)$$

To determine the temperature at the probe tip, a simple RTD or thermistor is used to measure the cold junction temperature T_{cold}, and the voltage difference at the cold junction V_{out} is measured as an analog output from the circuit. This work is all done on a single, dedicated IC chip known as a compensator/amplifier. In this lab, we will use the MCP9600 chip, which measures T_{cold} and V_{out}, computes the temperature T_{hot}, and outputs T_{hot} and T_{cold} as a serial digital packet.

Procedure

We will now use the SparkFun Qwiic Thermocouple Amplifier – MCP9600 breakout board and a Type K thermocouple to measure temperature. The MCP9600 will compute the temperatures and send them to the Arduino UNO microcontroller as I^2C serial data. The I^2C protocol is a common form of *synchronous* serial communication.

All I^2C sensors will have at least four electrical connections:

- Ground (GND)
- DC power source between 3 to 5V (VCC)
- Shared clock signal (SCL)
- Serial data (SDA)

Important: You will use the **3.3V** power source on the Arduino to power the MCP9600. You do NOT need to use the breadboard for this part of the lab.

1. Go to SparkFun.com and search for the MCP9600 Qwiic Thermocouple Amplifier, part number 16294. When you have located the part webpage, scroll down and click the "Get started …" link.
2. On the "Hookup Guide" page, scroll down and click the link that says "Download the SparkFun MCP9600 Arduino Library (ZIP)".
3. Open the Arduino IDE software, and add the MCP9600 code library to Arduino IDE. Similar to Part I, go to "Sketch" > "Include Library" > "Add ZIP Library...". Then navigate to the ZIP file you just downloaded and select it.
4. The library you downloaded will teach the Arduino how to communicate with the MCP9600, and it includes some basic examples codes. (Arduino calls its codes "sketches".) Open one of the example sketches. Go to "File" > "Examples" > "SparkFun MCP9600 Thermocouple Library" > "Example1_BasicReadings".
5. Make the following connections between the Arduino and MCP9600 breakout board using female-to-male jumper wires:
    a. GND on MCP9600 → GND on Arduino
    b. 3V3 on MCP9600 → 3.3V on Arduino
    c. SCL on MCP9600 → SCL on Arduino
    d. SDA on MCP9600 → SDA on Arduino
6. Sketch the MCP9600, Arduino UNO, and electrical connections between the two devices in your lab notebook. Label the four connections.
7. In the Arduino IDE software, go to "Tools" > "Port" and select the COM port that says "(Arduino/Genuino Uno)" next to it.
8. Click the check mark at the top of the Arduino program to check the code for errors.
9. Press the arrow button to compile the program and send it to the Arduino.
10. Go to "Tools" > "Serial Monitor" (or press "Ctrl + Shift + M") to view the output from the "Serial.print()" commands, and set the baud rate to 115200.

11. Press the reset button on the Arduino UNO, and you should see the measured temperatures printed. Note that MCP9600 measures *both* the thermocouple probe temperature (hot junction) and the ambient temperature (cold junction).
12. Hold the thermocouple probe tip in your hand. Record the value in your lab notebook. Does the printed temperature seem reasonable?
13. Test the accuracy of the thermocouple and MCP9600:
    a. Look at the mercury thermometer on the counter and record the ambient air temperature in the room in your lab notebook.
    b. Record ambient temperature from the MCP9600 in your lab notebook. Compare it to the ambient temperature that you measured with the mercury thermometer.
    c. Use the thermocouple probe to measure the temperature of ice water. Record the value in your lab notebook.
14. **Demonstrate the working system to the TA.**
15. Disconnect the MCP9600 from the Arduino UNO. Leave the thermocouple plugged into the MCP9600.

Part III: Light Detection and Ranging (LIDAR)

Background

LIDAR sensors are commonly used in most new automobiles to detect nearby objects. Feedback from these sensors can alert the driver that someone is in their blind spot, or they can be used to completely automate the driving process (i.e. "self-driving cars").

Illustrated on the left side of Fig. 4, a LIDAR sensor sends out a pulse of light from an infrared laser. The pulse reflects off a nearby object and returns to a detector adjacent to the laser. The time it takes for the reflected pulse to return Δt is used to calculate the distance $r = c\Delta t/2$, where $c = 2.99\times10^8$ m/s is the speed of light. Sophisticated LIDAR cameras send out thousands of laser pulses in multiple direction to create a 3D image of surrounding objects or terrain.

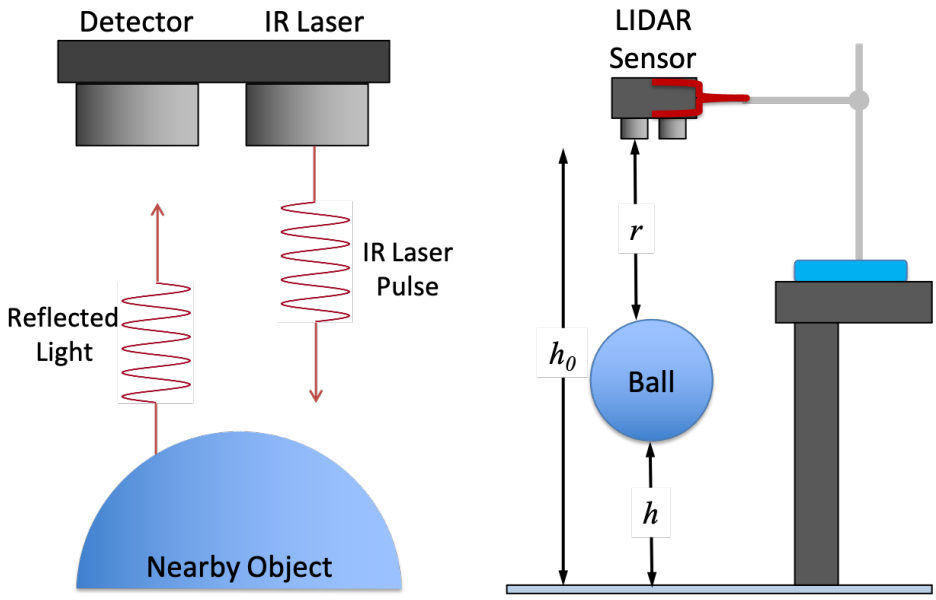

Figure 4 – (Left) A LIDAR sensor works by measuring the time it takes for a reflected laser pulse to return to a detector. (Right) A LIDAR sensor is used to measure the height h vs. time t of a bouncing ball.

In this lab, you will use a LIDAR sensor to measure to measure the height h vs. time t of a ball as it bounces beneath the sensor. Shown on the right side of Fig. 4, the LIDAR sensor directly measures the distance r between the sensor and the top of the ball. To compute the height of the ball from the floor, you must subtract r and the diameter of the ball from the height of the sensor h_0.

The elasticity of a ball is quantified by its *coefficient of restitution* (COR), which is defined as the ratio of the velocity just after it leaves the ground divided by the velocity just before the ball hits the ground $C_R = v_f / v_i$. A perfectly elastic ball will have $C_R = 1$, while a completely inelastic ball will have $C_R = 0$. The ball used in this lab will have a COR somewhere between zero and one, and it will be your job to measure it.

Procedure

We will now use a Garmin LIDAR-Lite v3 sensor to measure the height as a function of time for a bouncing ball. This sensor uses the same I^2C output as the MCP9600, so you will have similar connections with the Arduino: GND, VCC, SCL, and SDA.

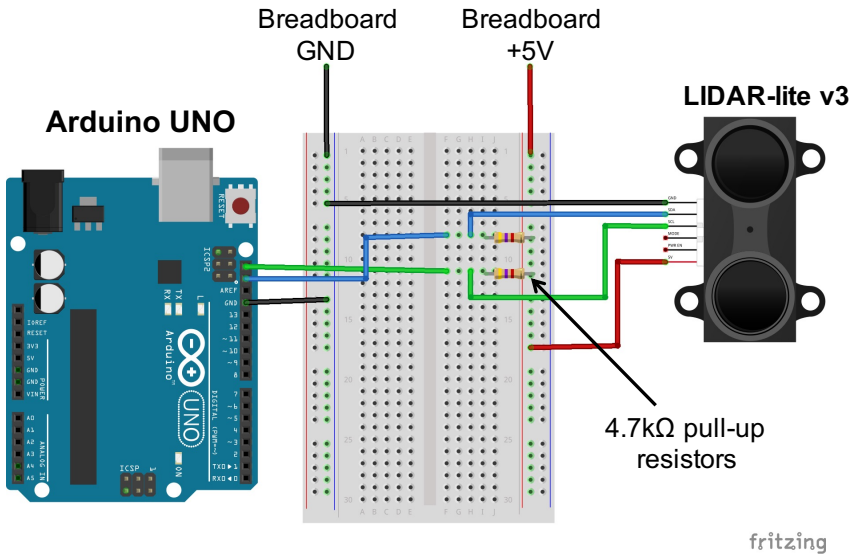

Figure 5 – The LIDAR sensor is connected to the Arduino via a breadboard.

Safety First: Do NOT look directly into the LIDAR sensor. It contains an infrared laser. You will not be able to see the IR laser spot, but it will still burn your retinas!

1. Mount the LIDAR sensor in the 3-finger beaker clamp so it is pointing downward and hanging over the edge of the lab bench, as illustrated in Fig. 4. Take a moment to examine the 3-finger beaker clamp. Focus your engineering awareness on the clamping mechanism. Try to think of the most stable way to clamp the sensor.
2. Go to SparkFun.com and search for the LIDAR-lite V3, part number 14032. When you have located the part webpage, scroll down and click the "Get started …" link.
3. On the "Hookup Guide" page, scroll down and click the link that says "Download the Garmin LIDAR-lite V3 Arduino Library". Save the ZIP file to the desktop of the computer.
4. Open the Arduino IDE software, and add the LIDAR-lite code library to Arduino IDE. Similar to Part II, go to "Sketch" > "Include Library" > "Add ZIP Library...". Then navigate to the ZIP file you just downloaded and select it.
5. Go to the A8 webpage on the course website, download the "LIDAR Data Acquisition" code, and open it in the Arduino IDE software.
6. Open the LIDAR data acquisition code you download from the A8 webpage in the Arduino IDE software. Press the check mark to make sure it compiles correctly.

IMPORTANT: The Arduino cannot provide enough current to power the laser, so you will need to use the 5V power supply on the breadboard. Also, it is recommended that you include 4.7 kΩ "pull-up resistors", as illustrated in Fig. 5.

7. Connect the LIDAR sensor to the Arduino UNO via the breadboard, as shown in Fig. 5. (You may need to re-adjust the beaker clamp, so the wires can reach the breadboard.) Make the following connections using jumper wires:
    a. BLACK on LIDAR → GND on breadboard
    b. RED on LIDAR → 5V on breadboard
    c. GND on Arduino → GND on breadboard
    d. GREEN on LIDAR → SCL on Arduino (via 4.7 kΩ pull-up on breadboard)
    e. BLUE on LIDAR → SDA on Arduino (via 4.7 kΩ pull-up on breadboard)

CAUTION! Do NOT look directly into the LIDAR sensor. It contains an infrared laser. You will not be able to see the IR laser spot, but it will still burn your retinas!

8. Sketch the LIDAR-lite, Arduino UNO, and electrical connections between the two devices in your lab notebook. Label the four connections.
9. In the Arduino IDE software, go to "Tools" > "Port" and select the COM port that says "(Arduino/Genuino Uno)" next to it.
10. Click the arrow button in the IDE software to upload the LIDAR data acquisition code to the Arduino UNO.
11. Go to "Tools" > "Serial Monitor" (or press "Ctrl + Shift + M") to view the output from the "Serial.print()" commands, and set the baud rate to 9600.
12. Press the reset button on the Arduino UNO, and you should see the time t (units of ms) and measured distances r (units of cm) printed in the serial monitor. The code will record data from the LIDAR sensor for a duration of $T_{max} = 8$ seconds at a sampling frequency $f_S \approx 75$ Hz. It will repeat the data acquisition every time you press the reset button on the Arduino UNO.
13. Use the tape measure and your hand to verify the system is measuring the correct distance in centimeters.
14. Sketch the experimental setup illustrated on the right side of Fig. 4 in your lab notebook.
15. Measure the distance h_0 from the LIDAR sensor to the floor. Record the value in your lab notebook.
16. Measure the diameter D of the ball. Record the value in your lab notebook.
17. You will now measure the distance vs. time for a bouncing ball. Locate the basketball or kickball. Hold it directly under the sensor, and have your lab partner press the reset button on the Arduino. When data starts to appear in the serial monitor, drop the ball so it bounces *several times directly underneath the sensor*.

Pro-tip: For consistent vertical bounces, hold the ball so the black air hole for pumping up the ball is pointed upward. Lightly hold the ball by the sides with flat hands. Release it by quickly moving both hands horizontally outward in a synchronized fashion.

18. The ball must bounce several times directly under the sensor. If it bounces off to the side, you must click "clear output" on the serial monitor and try again.
19. When you are confident you have good data, save the data.
    a. Highlight all of the data in the serial monitor up to time $t = 0$. (Make sure you are only selecting data from the most recent drop.)
    b. Press "Ctrl + c" to copy the data.
    c. Paste the data in a blank .txt file using notepad.
    d. Save the data as a .txt file.
20. Import the data into Matlab or Excel and make a quick plot of distance vs. time to show to the TA. **The data should look like a series of several consecutive parabolas.** If it does not look like this, you must repeat the measurement.
21. Email the data to yourself and your lab partner, or use a USB flash drive to transfer it to you laptops.
22. Return the lab bench to its initial state:
    a. Disconnect the LIDAR and Arduino UNO from the breadboard. Disassemble the circuit on the breadboard.
    b. Delete the LIDAR code library from the download folder.
    c. Return the ball to the cart at the center of the lab.

Data Analysis and Deliverables – Using the format and style outlined on HW1, write a brief tech memo containing the following items.

1. A table comparing temperatures measured with the thermocouple and MCP9600 to the known standards (mercury thermometer and known temperature of ice water).
2. A plot of the ball height h vs. time t of the bouncing ball. (The height $h(t)$ is defined in Figure 4. It must be computed from the measured value of h_0, D, and $r(t)$.)
3. Crop out a single parabola worth of height vs. time data for the bouncing ball, and apply a quadratic curve fit.
    a. Plot the measured data as individual points.
    b. Plot the quadratic curve fit as a smooth, continuous line.
    c. Use the fitting parameters to extrapolate the acceleration due to gravity g. Report the extrapolated value of g in the caption of the plot.

Talking Points – Please discuss the following in your lab report.

- Which experiment yielded a better value of g? This one or Galileo's inclined plane?
- Include the theoretical equation for the height vs. time of a single bounce of the ball.

Appendix A

Alternative Procedure for Collecting LIDAR Data

Procedure

1. Click on the "Tools" menu in the Arduino IDE software. On the drop down menu, go to "Port", and select the one that says COM# "(Arduino/Genuino Uno)", where # will be a number corresponding to the COM port being used for the Arduino. **Write down the number of the COM port corresponding to the Arduino in your lab notebook.**
2. Open the PuTTY software from the desktop. Select "Session" on the left and check "Serial" as the connection type. Make sure the baud rate is 9600, if it is already not set to the value. Then, enter the COM# you wrote down earlier.
3. Select the "Logging" tab right below session on the left side of the PuTTY app. In the option below "Session logging" select "Printable output". Click "Browse" to choose the file name and destination where the data will be saved. Give it an appropriate name and save it on the computer as a .txt file.
4. Click the "Open" button, and a black terminal window should appear. You should see numbers appear in the terminal corresponding to Serial.print() statements from the Arduino code.

Appendix B

Equipment

- Arduino UNO Microcontroller
- 12” male/male jumper wires
- 12” male/female jumper wires
- 6ft USB cable
- Breadboard
- BNC cable with mini-grabber adapters
- DHT11 Digital Humidity and Temperature sensor (blue with 3 pins)
- MCP9600 thermocouple amplifier (SparkFun Part # 16294)
- 1/16” Type K thermocouple
- Garmin LIDAR-lite V3 with 50cm cable (SparkFun Part # 14032)
- Lab stand w/ 3-finger beaker clamp
- Measuring taper
- Kickball or basketball
- Extech MN35 DMM
- Hg Thermometer
- Ice Water