Experiment 2: Experimental Uncertainties and the Human Reaction Timer

No physical measurements can be made to perfect precision so an estimate of uncertainty must be included with any measurement. Also, quantities derived from or calculated from measurements also have a limited precision and uncertainty. We need to figure out how to determine uncertainties in such calculated quantities from our measured uncertainties.

In this experiment we will first discuss sources and types of uncertainties and how to quantify them. We'll then describe how to estimate uncertainties for single measurements and for a set of measured data. For the latter, you will use the Arduino and some circuitry to explore these concepts.

Learning Objectives: In order to earn high marks on this experiment, the student should be able to:

- Articulate the meaning of systematic error.
- Articulate the meaning of random error.
- Correctly express measured values and their absolute uncertainties.
- Perform calculations of percent uncertainty and absolute uncertainty.
- Estimate uncertainty for a single measurement.
- Compute standard deviation (and therefore estimate uncertainty) for a set of data.
- Plot a histogram of experimental data using a spreadsheet.
- Compare and contrast accuracy versus precision.
- Apply the commands `Serial.begin()`, `Serial.print()` and `Serial.println()` in an Arduino sketch.
- Apply the commands `digitalRead()` and `millis()` in an Arduino sketch.
- Explain the operation of a `while{}` loop in Arduino.
- Connect a push-button on a breadboard.

I. Types of Experimental Error

During acquisition of data, two types of experimental errors, systematic and random usually contribute to the error in the measured quantity.

Systematic Errors and Accuracy

Systematic errors are produced by a well-defined cause which affects every measurement in exactly the same way. These errors result in the measurement having poor **accuracy** – the measured values will be consistently too high or consistently too low (see figure 1). Systematic errors may be eliminated once this well-defined cause is isolated. Systematic errors may be of four kinds:

1. *Instrumental.* For example, a poorly calibrated instrument such as a thermometer that reads 101°C in boiling water and 1°C in ice water would give measured temperatures that are consistently too high.
2. *Observational.* For example, parallax in reading a meter scale.
3. *Environmental.* For example, an electrical power "brown out" that causes measured currents to be consistently too low.
4. *Theoretical.* Due to simplifications of the model system or approximations in the equations describing it. An example of this could be ignoring frictional forces in one's calculations even though a frictional force acts during the experiment. The experimental and theoretical results will consistently disagree.

Random Errors:

Even if we eliminate systematic errors, the values obtained in a physical measurement will always lie within a range of values, rather than yield a unique value. These irregular deviations of a measurement are said to be random errors – large random errors are said to produce measurements with poor **precision**. See figure 1. Random errors are positive and negative fluctuations that cause about one-half of the measurements to be too high and about one-half of the measurements to be too low. Sources of random errors cannot always be identified. Random errors, unlike systematic errors, can often be quantified by statistical analysis; therefore, the effects of random errors on the quantity or physical law under investigation can often be determined.

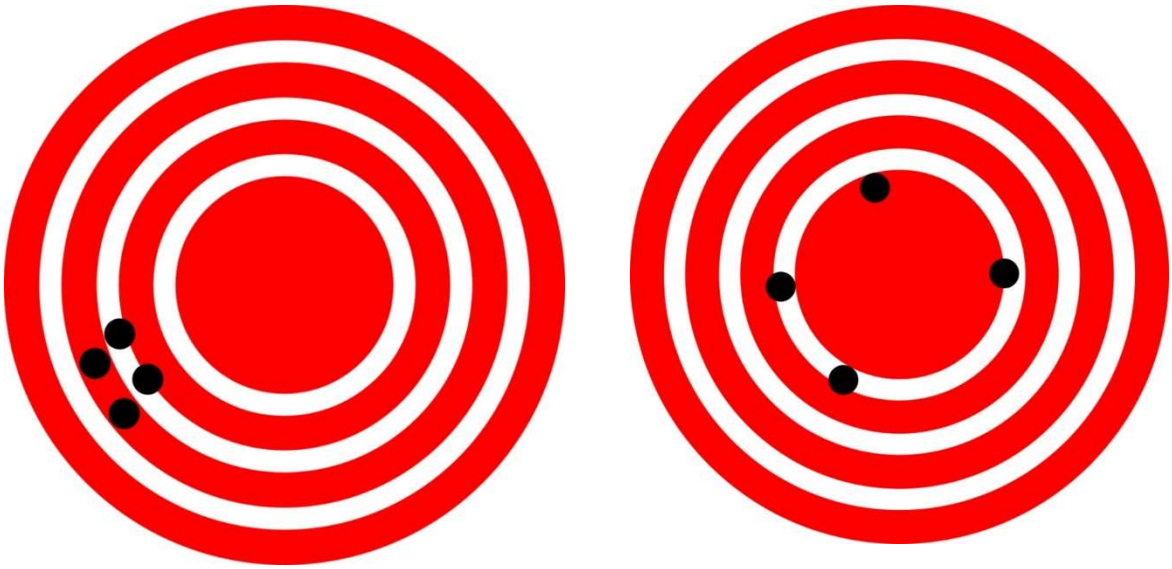


Figure 1: Illustration of significant systematic error and poor accuracy (left bullseye) versus larger random error and lower precision (right bullseye). A worthy goal for the experimenter would be to maintain the high precision of the measurements made on the left bullseye while eliminating the systematic error resulting in the low accuracy (which caused the target to be missed!).

II. Expressing Uncertainty:

The most common way to express a measured value and its associated uncertainty is to use *absolute uncertainty*. If we use the symbol δx to represent the absolute uncertainty of the measured quantity x, we'd write $x \pm \delta x$.

Example 1: A length measurement has an uncertainty δx = 6cm in a length measurement of x= 200cm. We would express this as 200 ± 6cm or 2.00 ± 0.06m. It is important to notice that x and δx have the same number of digits after the decimal point[1].

Example 2: In some situations, you may hear people refer to their results in terms of the *percent uncertainty*. In example 1 above, this would amount to dividing the absolute uncertainty by the measured value: Percent Uncertainty = $\frac{\delta x}{x} = \frac{6cm}{200cm} = 0.03 = 3\%$. So the experimenter in this case could boast that they "performed a 3% experiment"[2].

[1] In other words, we'd *never* write this as 200±6.0cm or 200.00±6cm.
[2] Some "old timers" may even say that their experiment was "good to 3 parts in 100".

Example 3: I measure my body mass on a scale with a precision of about 2%. If the scale reads m = 109.0kg, how should I report my weight? In this case we would determine the absolute uncertainty by multiplying δm = (109.0kg)(0.02) = 2.2kg. So we'd have to say my mass was 109.0 ± 2.2 kg. (Note again the matched number of digits to the right of the decimal point!)

III. Estimating Uncertainty from a *Single Measurement*

In many situations you'll need to estimate the uncertainty of a measurement you make only one time. In this case, you'll have to take clues from the instrument you are using to make the measurement as well as from the quantity you are measuring.

Figure 2 shows and example of this. We are using a centimeter marked ruler to measure the width of a Raspberry Pi computer board. In the image on the left, we can see that the ruler's finest markings are 1mm wide. We can generally estimate down to one-half of this smallest division which, in this example would amount to an uncertainty of ±0.5mm. The image on the right would tell us that the width of the board is 55.5±0.5mm. (Remember there are 10mm in each cm and again, note that the number of digits to the right of the decimal points is the same for the measurement and its estimated uncertainty!)

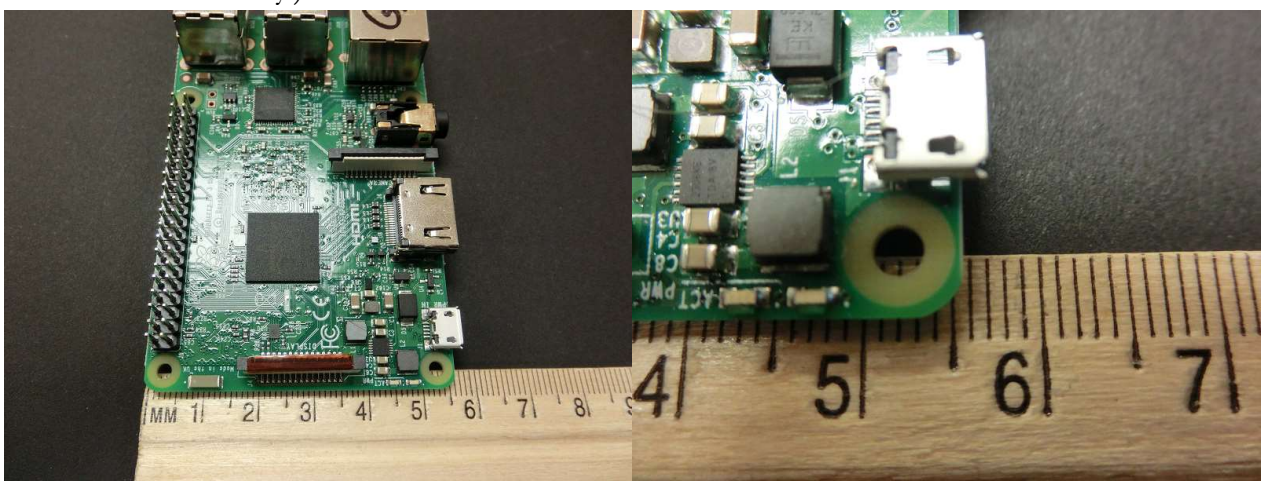


Figure 2: Measuring the width of a Raspberry Pi board using a 1mm ruled ruler. The image on the right indicates a width of 55.5±0.5mm. The uncertainty in this example is estimated by dividing the smallest ruler division in half.

Before moving on, consider the measurement being made in figure 3. In this case, the length of the string is a bit tougher to see because of the frayed end. It's not likely I'd want to bet the farm on knowing the length of the string to a precision of ±0.5mm!!! Because of the fraying, we'd probably want to be less confident in the string's length. A good estimate of this uncertainty might be ±1mm. So we might conclude that the string is 72±1mm long (or 7.2 ± 0.1cm).

Figure 3: The string has a frayed end so our precision needs to reflect that fact. The string is about 7.2 ± 0.1cm long.

IV. Estimating Uncertainty from a Set of Measurements

In many situations, we will measure a quantity *N* times so we end up with a data set. In this case, we can use some basic statistical measures to estimate the uncertainty of our measurements. Consider the example data (with *N* = 5 data points) shown in Table 1 below. To get the best measure of the elapsed time, we'd simply find the average (or *mean*) of the measured times. We'd write this as $\overline{\Delta t}$ and we'd have

$$\overline{\Delta t} = \frac{4.92 + 4.89 + 5.26 + 4.90 + 5.10}{5} = 5.01s$$

Table 1: Elapsed Time Measurements	
Trial	Δt (s)
1	4.92
2	4.89
3	5.26
4	4.90
5	5.10

In order to estimate the uncertainty in these measurements, we'd use the standard deviation which is computed according to

$$\delta x = \sqrt{\frac{\sum(x_i - \overline{x})^2}{N}}$$

where the x_i are the measured values. In the example provided this would amount to:

$$\delta x = \sqrt{\frac{(4.92 - 5.01)^2 + (4.89 - 5.01)^2 + (5.26 - 5.01)^2 + (4.90 - 5.01)^2 + (5.10 - 5.01)^2}{5}}$$

yielding

$$\delta x = 0.14s$$

We would then conclude that the elapsed time was 5.01±0.14 s.

Exercise: Determine the mean and standard deviation of the following measured values: 3.0, 3.2, 3.1, 2.8, 2.9. *Answer:* 3.0 ± 0.1. (My calculator actually reads δx= 0.1414, but the number of digits to the right of the decimal has to agree with the mean to it has to get rounded off a bit.)

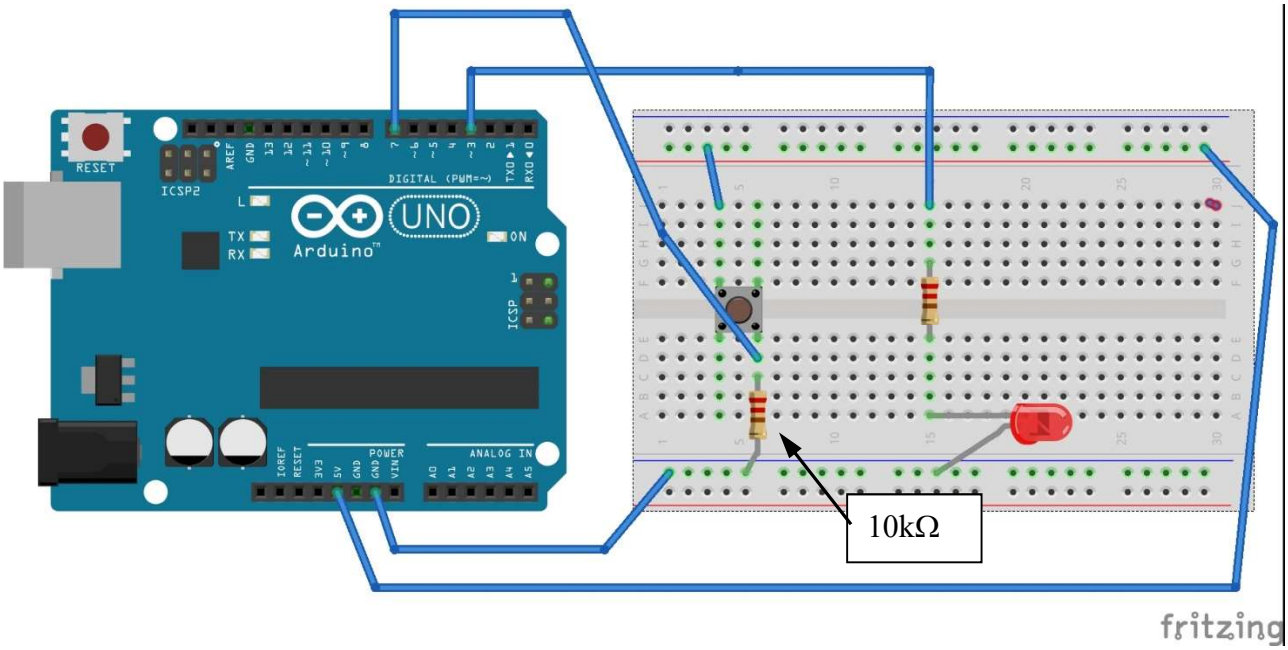
V. Using Arduino to Measure Human Reaction Time

Although you might think you can respond instantaneously to some sort of external stimulus (like a buzzer or visual cue of some sort), in reality there is neurological delay inherent in our physiology. This delay is known as our reaction time and depends on many factors. But can we *measure* that reaction time? Of course we can…we're **physics** and we got some **Arduino**…

The game is played by programming the Arduino to turn on an LED with a random amount of delay (so cool kids can't "game" the system!). When it does, the player mashes the pushbutton as quickly as possible…the delay between when the LED came on and when the button gets mashed is then reported as the reaction time. In order to make these measurements we'll have to first learn how to make the Arduino sense a button push (a ***digital input***!) and then recall how to connect up an LED which is what will trigger the whole thing. Here we go…

Part A: Using the Arduino's Digital *INPUT* (or Sensing Button Pushes)

1. The figure shows a pushbutton connected to the Arduino on the left half of the breadboard. It's got a resistor connecting it to ground and wire connecting its "upper left corner" to +5V. The lower right corner is also connected to `pin 7` on the Arduino.
2. There is also an LED (and 220Ω limiting resistor) connected in just the same way you used in our first "blink" lab last week.



3. ***Carefully*** connect the circuitry. You may want to have your instructor check that it's OK before moving ahead.

4. Now it's time to work some Arduino magic. Notice that there are some old favorites here (like `pinMode`, `void setup` and `void loop`). But there are some new things too…

```
void setup()
{
   pinMode(7, INPUT);
   pinMode(3, OUTPUT);
   Serial.begin(9600);
   Serial.println("#goPhysics!");
   delay(5000);
}
void loop()
{
    if (digitalRead(7) == HIGH)
    {
        Serial.println("Somebody pushed the button!!!!");
    }
    else
    {
        Serial.println("The button isn't pushed");
    }
}
```

5. To see the output go to the "`Tools`" menu and click on "`Serial Monitor`". This should show you a message from the Arduino.

Part B: Adding in the LED

1. After getting the serial monitor and pushbutton working, add some code so that the LED goes on when the button is pushed and off when it's not…this should involve exactly two lines of code…

Part C: Measuring Your Reaction Time(s)

It's now time to take some measurements of your reaction time. The circuit is (cleverly) already set up! All we need to do is code the thing and then have some fun with it. The idea is to have the Arduino turn on the LED after a random wait/delay and then wait for you to push the button while keeping track of time.

1. Open a new Arduino sketch and enter the code on the next page. Repeat your reaction time measurement at least 20 times. Record the data for each event in your lab notebook.
2. What is your average reaction time? What is the uncertainty (standard deviation) in this measurement?

```
unsigned long waitTime;
unsigned long startTime;
unsigned long reactTime;

void setup()
{
   Serial.begin(9600);
   pinMode(3, OUTPUT);
   pinMode(7, INPUT);
   Serial.println("Starting game...");
   delay(5000);
}

void loop()
{
   Serial.println("Get ready...");
   waitTime = random(1000, 5000);       //Wait a random amount of time 1 – 5s
   delay(waitTime);

   digitalWrite(ledPin, HIGH);          //Turn LED on
   startTime = millis();                //Start timing; units are ms

   while(digitalRead(buttonPin) == LOW)  //This stalls for as long as the
   {                                         //button isn't yet pushed
   }

   reactTime = millis() - startTime;      //Now get the elapsed time
   digitalWrite(ledPin, LOW);             //Turn LED off

   Serial.println(reactTime, 1);
   Serial.println();
   delay(10000);                         //Wait 10 seconds and do it again!
}
```

3. Above and Beyond: Does the reaction time depend on the *brightness* of the LED you use? What about its *color*? Are you faster or slower with your non-dominant hand? Can you make some measurements to answer one of both of these questions?[3]

[3] Or think of some *other* question you may want to investigate!