MATHEMATICS: THE POWER BEHIND THE DRONE

Frank Wattenberg[1]
Matthew Mogensen
Department of Mathematical Sciences
United States Military Academy
West Point, NY 10996
Frank.Wattenberg@usma.edu
Matthew.Mogensen@usma.edu

The photograph above was taken by an aerial drone controlled by a microprocessor very much like the Arduino. Although you immediately see a drone's motors and rotors, the underlying mathematics – the algorithms and programs that power the drone – are not visible. Without this mathematics a drone would never fly. This paper talks about the underlying mathematics and programming that give Arduinos and drones life. We give several examples showing how these ideas can enliven our current mathematics classes.

[1]The views expressed in this article are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, the Department of the Army, the Department of Defense, or the U.S. Government.

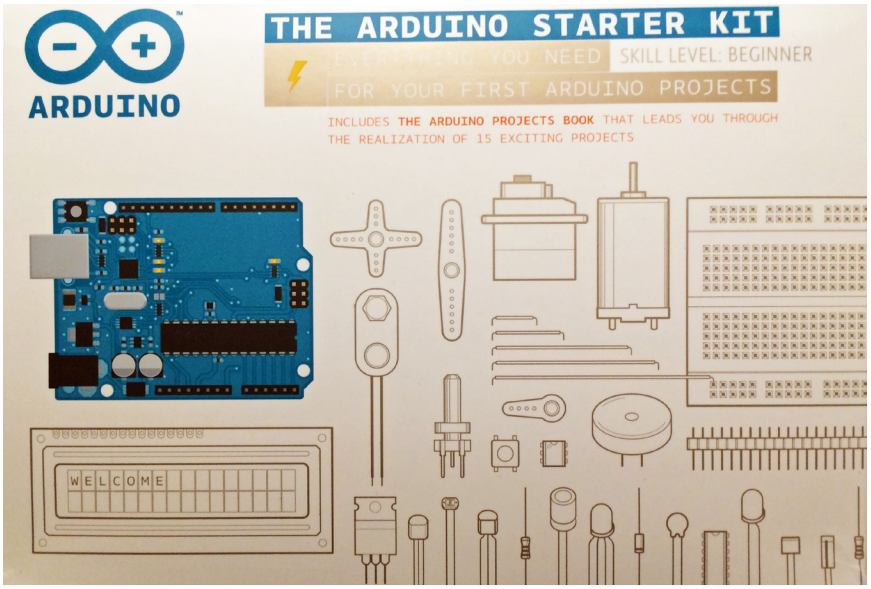

Figure 1: The Arduino starter kit

Figure 1 shows the Arduino Starter kit. This kit is available for under $100.00 and has all the parts and documentation needed to begin the study of computer-controlled devices. For algebra and pre-calculus courses this box is essentially a box of functions and provides an engaging playground in which we can learn about functions. This is a good place to begin a discussion about using robotic devices in mathematics classes.

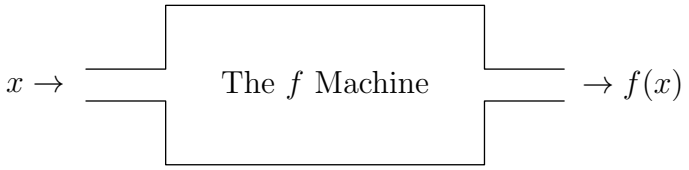

Figure 2: A function – a machine that accepts input and produces output

Project 1: A Simple Function

Figure 2 is a typical figure from a typical mathematics textbook discussion of functions. We often introduce key ideas like **domain** and **co-domain** by looking at algebraically defined functions like:

$$f(x) = \sqrt{1 - x}$$

and noting that this function is undefined when $x < 1$ since you can't take the square root of a negative number. Thus, the domain of this function is the set $\{x \ : \ 1 \leq x\}$, or the interval $[1, +\infty)$ – BORING!!!

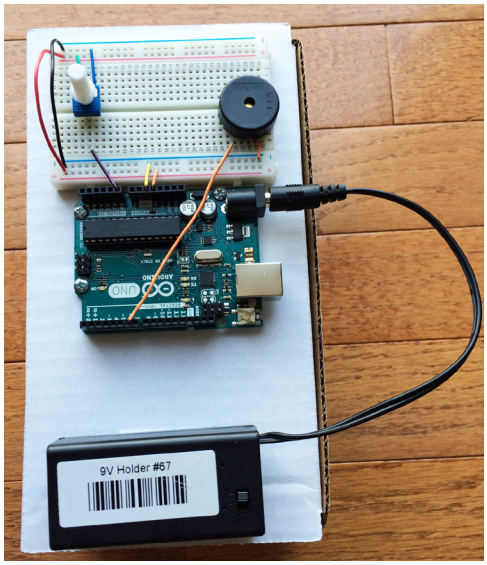

Figure 3: A Real World Function

Figure 3 shows a much more interesting function – a real machine that you can build from the Arduino starter kit in a few minutes. The input to this machine is the setting of the knob on the left at the top. You can try different inputs by turning the knob back-and-forth. You can't see the output because the output is a sound, coming from a piezo – the round black thing on the right at the top. The domain of this function is the set of possible settings for the knob. The domain is a physical notion determined by the construction of the knob. The co-domain is the set of sounds.

Students often have difficulty with the distinction between the co-domain of a function and the image of a function. In fact, these words are sometimes used differently along with the word "range." In this setting we can immediately see how different ideas about the output arise naturally. We need words for these ideas. We use the word "co-domain" for the set of all possible outputs. In this case, we know that the

outputs are sounds. But, this particular machine with this particular domain doesn't produce all possible sounds. You can determine which sounds it actually produces by trying all possible inputs. Turn the knob back-and-forth several times slowly. The set of outputs actually produced from all possible inputs is called the image.

This is also a good setting in which to discuss the other attributes of functions. For example, a function is repeatable – the same input always produces the same output. Students studying functions like $f(x) = 2x + 3$ may not appreciate the meaning or importance of this attribute but in this real world context it is much more meaningful. In fact, we can stress that this is an ideal property of mathematically defined functions that may be hard to achieve in the real world – for example, do the sounds change as the batteries are depleted?

The listing at the bottom of this page shows the entire program for this machine. Like all Arduino programs it includes two functions:

- A function `setup()` – This function is executed once to initialize the device. In this case it sets one of the Arduino pins, digital pin 6, up as an output pin. It will be used to control the piezo.
- A function `loop()` – This function is executed repeatedly while the device is on. It reads the setting of the knob, which is wired to analog pin A0, and sends a signal to the piezo through digital pin 6.

```
void setup()
{
   pinMode(6, OUTPUT);
}

void loop()
{
  int frequency = 440 + analogRead(A0)/2;
  tone(6, frequency);
}
```

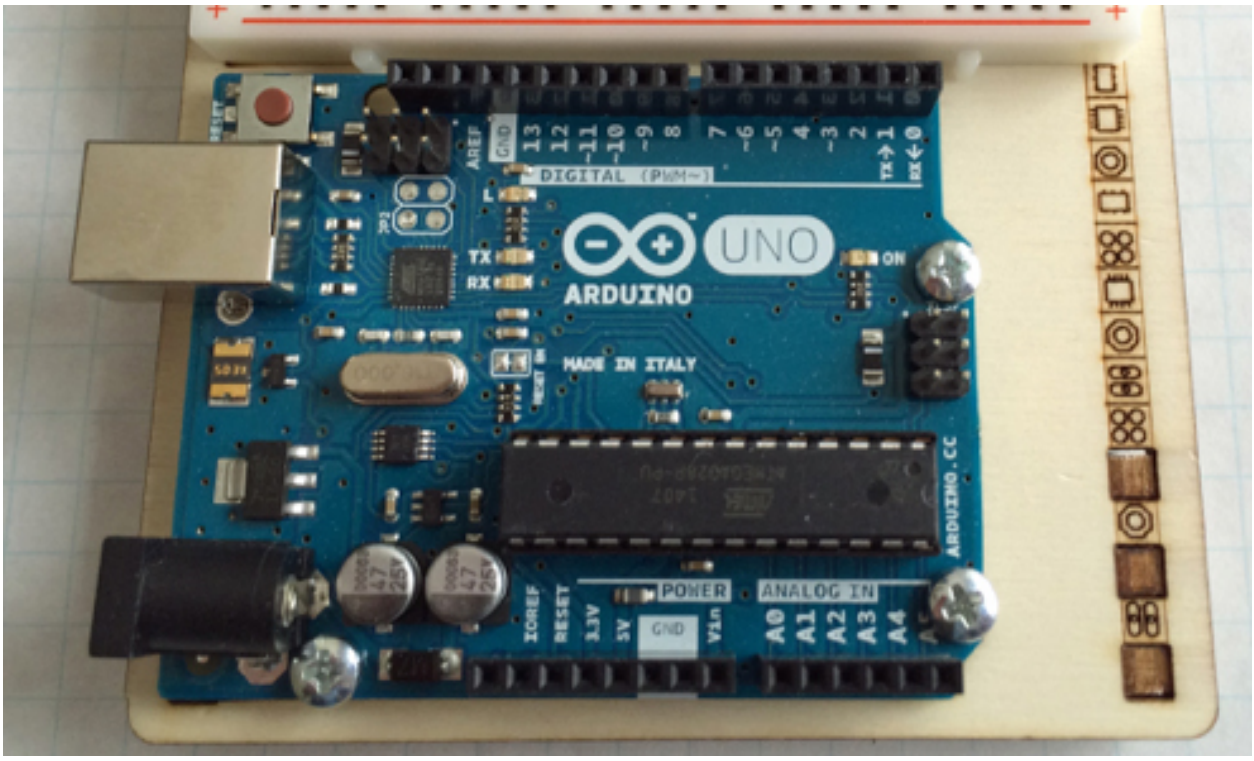

Figure 4: The Arduino Uno

The programs that control an Arduino are called **sketches**. Every sketch has at least these two functions – `setup()` and `loop()`. Figure 4 shows the Arduino Uno board that we use. It has a number of different "ports," sometimes called "pins." Notice in Figure 3 that wires connect some of these ports to components on the breadboard. Wiring the device goes hand-in-hand with programming.

Example 2: Linear Functions and a Simple Digital Rangefinder

This project uses a simple ultrasonic rangefinder – the HC-SR04 – that is available from many suppliers including Amazon.com. We will use it to build a digital rangefinder that is a great project for students studying linear functions. Figure 5 shows the completed project. Notice that we've plugged the HC-SR04 into four sockets on the breadboard. This project requires four or six wires.

- The leftmost (Gnd) pin on the HC-SR04 must be connected to the Arduino

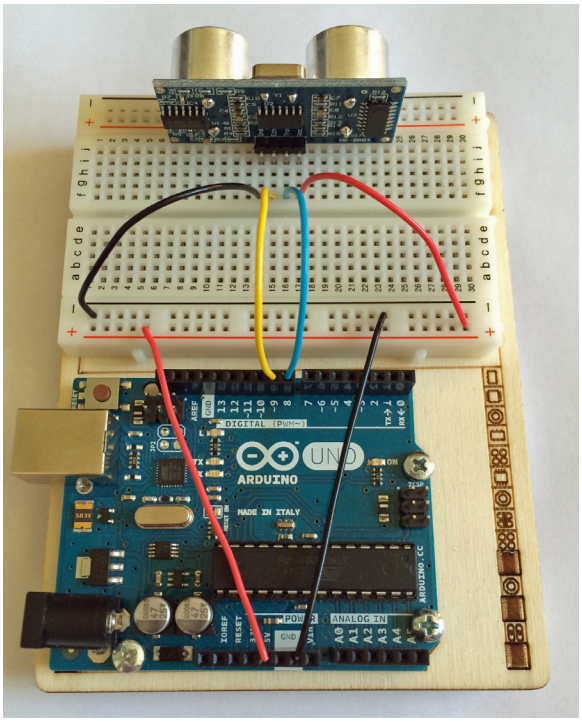

Figure 5: A Simple Rangefinder

Uno ground. In Figure 5 we use two black wires. The first black wire connects the Arduino Uno ground (GND) to the near 25 socket row on the breadboard. This provides a ground connection for this entire row of pins and is useful when you must make several connections to ground. The second black wire goes from one of the sockets in this row to a socket in the same column (and on the same half) on the breadboard into which the HC-SR04 is inserted.

- The next pin on the HC-SR04 is the echo pin. It is connected using a yellow wire to digital pin 9 on the Arduino Uno.
- The next pin on the HC-SR04 is the trigger pin. It is connected using a blue wire to digital pin 8 on Arduino Uno.
- The rightmost pin on the HC-SR04 is the Vcc pin. It is connected using two wires to the 5V pin on the Arduino Uno. The wiring is similar to the ground.

The listing on the next page shows the sketch we use.

```
#define trigger 8                // The rangefinder trigger pin is conected to D8
#define echo 9                   // The rangefinder echo pin is connected to D9

void setup()
{
  pinMode(trigger, OUTPUT);      // The Uno sends signals to the rangefinder trigger pin
  pinMode(echo, INPUT);          // and receives signals from the rangefinder echo pin
  Serial.begin(9600);            // The Uno sends information to the computer
  digitalWrite(trigger, LOW);    // Set the trigger signal to low.
}

void loop()
{
  digitalWrite(trigger, HIGH);   // The Uno sends the rangefinder a pulse whose duration
  delayMicroseconds(10);         // is 10 microseconds.  This causes to rangefinder to
  digitalWrite(trigger, LOW);    // emit a sonar pulse
  Serial.print("Time to pulse return: ");       // Print time for the pulse
  Serial.println(pulseIn(echo, HIGH));          // to return from obstacle.
  delay(1000);                   // Wait one second before next measurement.
}
```

The comments describe how the sketch works. It is designed to return the time required from when the unit is triggered until a sonar pulse returns after being reflected by an object. This is a great project for introducing or exercising linear functions. The time required should be given by a linear function, $y = mx + b$. Be sure to ask your students to interpret the values of the parameters m and b. They can use their answers to print distance measurements in units of their choice.

Example 3: A Simple Digital Level

Figure 6 shows our last example – a good example to use when students are studying vectors. It uses another board that is not included in the Arduino starter kit but is very useful and available for under $15.00 from Adafruit and other sources. This board includes both a magnetometer for measuring the ambient magnetic field and an accelerometer that is often used to measure the ambient gravitational field. Mobile devices, like cell phones and tablets use these capabilities together with GPS to determine both where they are located (GPS) and how they are being held – that is, the direction in which they are facing (magnetometer) and how they are tilted (accelerometer). When the device is still, the output of the accelerometer is the vector pointing toward the Earth's center of mass and whose magnitude is the acceleration due to gravity. When the device is accelerating, the output is the total acceleration vector. The output of the magnetometer is the vector giving the ambient magnetic

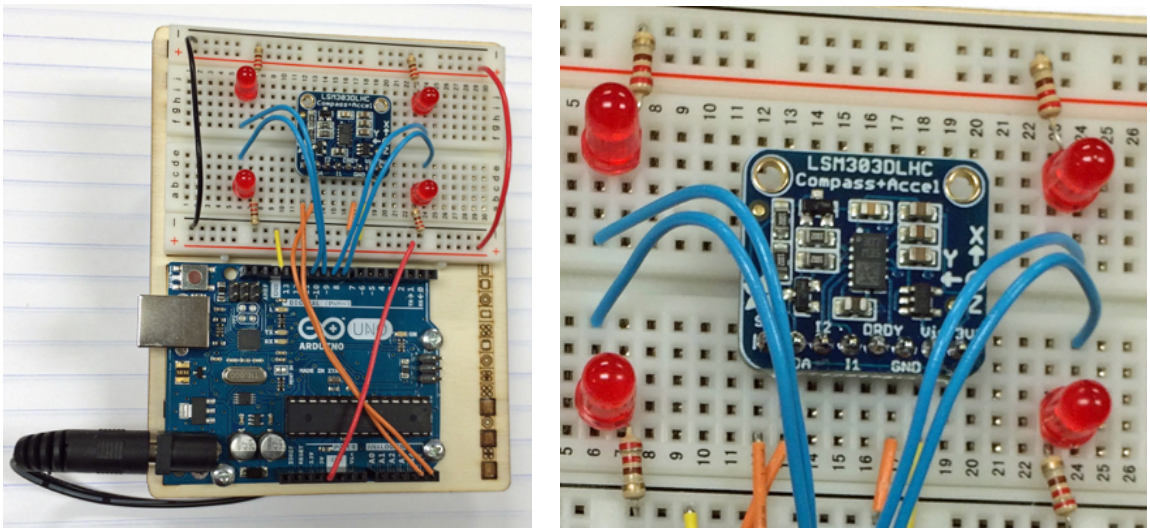

Figure 6: Digital Level Project

force – it points toward compass north.

We use just the acceleration vector to build a digital level. Notice the four red LEDs. When the device is tilted the lower LEDs light up. Usually either two LEDs are lit (if one side is tilted down) or one LED is lit (if one corner is tilted down). When all four LEDs are off the device is close to level. Students can experiment with the term "close to" producing a level that is easy more forgiving and easier to "level" or one that is fussier but is more useful for precise leveling.

Each of the four LEDs is wired to one digital port. They are wired in series with a 220Ω resistor so that they receive the proper current. LEDs are directional components and their cathode (the shorter lead) must be wired to ground while their anode (the longer lead) is wired (through the resistor) to a voltage source – in this case one of the digital pins. The digital pins produce either $0V$ or $5V$. Each of the four blue wires connects one of the four digital ports to the anode of one of the four LEDs. The outputs of the accelerometer and magnetometer are each wired to one of the analog ports.

This device can be used in various ways depending on the needs of your class. It is a simple project and students can build it entirely on their own including the programming. You could give them the devices pre-wired and have them write

the program. They basically have to decide which LEDs to turn on based on the acceleration vector and what tolerance to use. You might also give them everything except the tolerance and have them experiment with different values. The listing below shows the program we use.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Math.h>

Adafruit_LSM303_Mag_Unified mag = Adafruit_LSM303_Mag_Unified(12345);
Adafruit_LSM303_Accel_Unified accel = Adafruit_LSM303_Accel_Unified(54321);

int rightBack = 10;
int rightFront = 11;
int leftBack = 9;
int leftFront = 8;
float epsilon = 0.15;

void setup()
{
   Serial.begin(9600);
   pinMode(rightBack, OUTPUT);
   pinMode(leftBack, OUTPUT);
   pinMode(rightFront, OUTPUT);
   pinMode(leftFront, OUTPUT);
   if(!accel.begin())
   {
      Serial.println("Error initializing accelerometer");
      while(1);
   }
}

void loop()
{
   float ax, ay, az;
   sensors_event_t accelEvent;
   accel.getEvent(&accelEvent);

   ax = accelEvent.acceleration.x;
   ay = accelEvent.acceleration.y;
   az = accelEvent.acceleration.z;
   Serial.print("ax, ay, az = ");
   Serial.print(ax);
   Serial.print(",   ");
```

```
    Serial.print(ay);
    Serial.print(",   ");
    Serial.println(az);
    digitalWrite(rightBack, LOW);
    digitalWrite(rightFront, LOW);
    digitalWrite(leftBack, LOW);
    digitalWrite(leftFront, LOW);
    if ((ax > epsilon) && (ay > epsilon)) digitalWrite(leftFront, HIGH);
    if ((ax > epsilon) && (ay < -epsilon)) digitalWrite(rightFront, HIGH);
    if ((ax < -epsilon) && (ay > epsilon)) digitalWrite(leftBack, HIGH);
    if ((ax < -epsilon) && (ay < -epsilon)) digitalWrite(rightBack, HIGH);
    if ((ax > epsilon) && (abs(ay) < epsilon))
    {
       digitalWrite(leftFront, HIGH);
       digitalWrite(rightFront, HIGH);
    }
    if ((ax < -epsilon) && (abs(ay) < epsilon))
    {
       digitalWrite(leftBack, HIGH);
       digitalWrite(rightBack, HIGH);
    }
    if ((ay > epsilon) && (abs(ax) <= epsilon))
    {
       digitalWrite(leftFront, HIGH);
       digitalWrite(leftBack, HIGH);
    }
    if ((ay < -epsilon) && (abs(ax) <= epsilon))
    {
       digitalWrite(rightFront, HIGH);
       digitalWrite(rightBack, HIGH);
    }
    delay(100);
}
```