

ECE 480 – Senior Design
Prototyping A Scaling Program and Circuit
November 13 2014
Name: Husain Aleid
Facilitator: Dr. Jian Ren

Abstract

Weighing scale is being used heavily in the industry nowadays for many purposes as it tends to be beneficial in many ways. In this note, a tutorial will be provided explaining how to create a circuit that will read the weight of an item in grams and convert it to voltage, and then how to program an Arduino Uno that will take the voltage as an input and convert it back to grams and store it.

Objective

This tutorial focuses on the strategy of building a scale circuit as well as a scale program that will train beginners in this field to create their first prototype, and then improve it once they have mastered skills needed to build such prototype after successfully completing this tutorial.

Introduction

In this prototype, a scale must be connected to the scale circuit in order to convert the grams from the scale to voltage in the circuit, and then the circuit using an amplifier will output the corresponding voltage to the Arduino, which will convert the voltage to grams and store it in the database.

In order to build this prototype, the following components are needed:

- 1) A Scale
- 2) A Broad board
- 3) 1 Instrumentation Amplifier
- 4) 1 47-uF Capacitor
- 5) 2 10-ohms Resistors
- 6) Arduino Uno Board

Please refer to References for the recommended type of scale amp, and Arduino.

The first 5 components will be used to create the circuit that will output the voltage to the Arduino which will store the weight in the database. Furthermore, a background in C language is strongly recommended to build the program for the Arduino however, this tutorial will go step by step to create this program.

Building the Hardware

Preparing the Scale

After obtaining the specific scale that is recommended in the references section, dismantling it is required to connect it to the circuit. Once the parts have been taken off, four wires will be clearly visible. These wires are needed to output the weight to the circuit. The scale should look similar to figure 1 after dismantling.

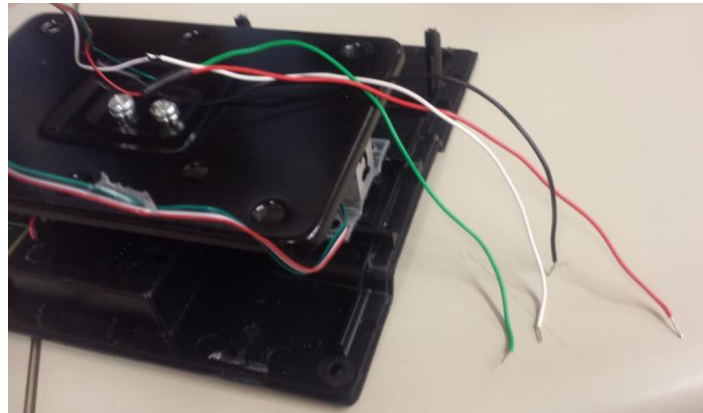


Figure 1: Dismantled Scale

Building the Circuit

First, the amplifier must be connected to the circuit, and then a 47-uF capacitor must be connected between ground and pin 11 to prevent any potential damage to the Arduino in case there was a voltage surge. Figure 2 shows how to connect the wires with the amplifier and build the circuit.

For the load cell:

E+ = Red wire
E- = Black wire
S+ = Green wire
S- = White wire

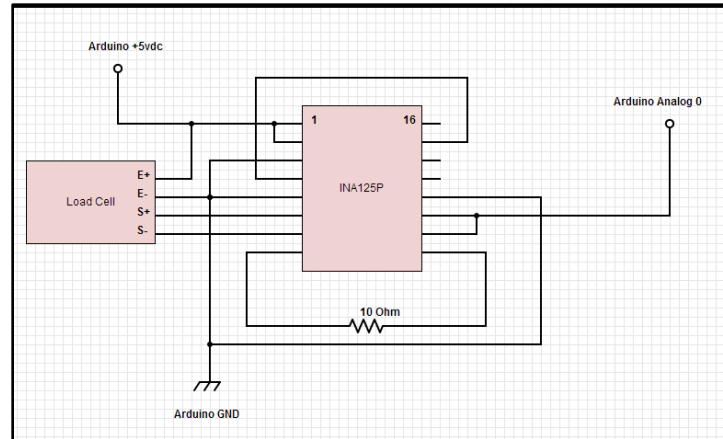


Figure 2: Circuit Schematic

The maximum voltage that could be applied to the Arduino is 5V, that is any higher voltage will fry it however, the scale used in this tutorial can output a weight up to 29 kilograms which corresponds to 5 volts in the Arduino, and thus there is no danger.

Testing the Hardware

To test if the circuit is connected correctly, an item could be placed on the scale, and the output voltage at pin 11 could be monitored using the following formula:

$$V = \frac{w}{5.8} , \quad \text{where } w = \text{weight in kg, } V = \text{voltage}$$

For example; an item that has a weight of 3.1 kg should output a voltage of .5344.

Connecting the Hardware with Software

The output of the amplifier can be connected to one of the input pins on the Arduino which is A1 in the case of this tutorial.

Arduino 1.5.8 software must be downloaded first to the computer, please refer to references for the website.

The computer should identify the Arduino once it has been connected to it, and it will link it to the Arduino 1.5.8 software, thus no further action is required for this matter.

Building the Software

Setting up the Program

First, frequency of readings have to be set up, in this example, we will have a reading every 200ms.

Then, the following variables must be declared:

- 1) time: starting time which is 0.
- 2) timeBetweenReadings: how often the software reads the weight
- 3) analogValue: it is the analog reading the Arduino takes for each item
- 4) analogValueAverage: the average analog readings for an item
- 5) i: counter
- 6) tare_weight and tare_valid: these variables will make the Arduino reads zero every time it restarts even if there were items on the scale. (*note: these variables are not important but could be used for other purposes*)

Figure 3 shows the code for this step.

```
float analogValueAverage = 0;

// How often do we do readings?
long time = 0; //
int timeBetweenReadings = 50; // We want a reading every 200 ms;

float tare_weight = 0.0;
int tare_valid = 0;

int i = 0;

void setup() {
  Serial.begin(9600);
  tare_weight = 0;
  tare_valid = 0;
}
```

Figure 3: Setup

Body of the Code

Since this program runs for indefinite time, a void loop must be used. The program must know which input pin is being used, thus the variable “analogValue” has to be set to read input pin A1. After that, the program will need to smooth the readings, that is the Arduino will keep getting almost the analog reading every 200 milliseconds. Figure 4 illustrates the code.

```
void loop() {

    int analogValue = analogRead(A0); // Reading input pin A0
    analogValueAverage = 0.999*analogValueAverage + 0.001*analogValue ; //Making our values smooth "little variations for one item"
```

Figure 4: Setting the Arduino to read from pin A0

Next step is to create two functions “analogToLoad1” and “mapfloat1”. “mapfloat1” will take the analog value reading from the Arduino and convert it back to grams and store it to variable “load”. Finally, “analogToLoad1” will call “mapfloat1” and return “load1”. Figure 5 summarize the code.

```
float analogToLoad1(float analogval1){

    // using a custom map-function, because the standard arduino map function only uses int
    float load1 = mapfloat1(analogval1);
    return load1;
}

float mapfloat1(float x1)
{
    if(!tare_valid1){
        return 0;
    }

    if(x1 > .95*tare_value1 && x1 < 1.05*tare_value1){
        return 0;
    }

    else{

        if (x1 >1)
            return 40.9*(x1) ;

        // ADD A HEAVY PLATFORM TRY TO MEASURE SMALL CHANGES IN THE MUCH-MORE-LINEAR REGION
    }
}
```

Figure 5: Two functions to convert weight into grams and return variable

Note that x1 is the same as “analogValue” we had in the previous codes. Also, note that 40.9 is the constant that is multiplied by x1 to get the correct value of a load in grams, since the scale is linear, 40.9 is held true for all values of x1.

Printing Results

The program outputs analog value reading as well as the load in grams, as a result, serial.print -which prints what is written inside it- and serial.println -which return value of a variable to serial.print- commands are needed. Figure 6 shows the code.

```
// Is it time to print?
if(millis() > time + timeBetweenReadings){

    float load1 = analogToLoad(analogValueAverage);

    i++;
    if(i >= 10){
        i = 0;
        Serial.print("analogValue: ");
        Serial.println(analogValueAverage);
        Serial.print("                load1: ");
        Serial.println(load1,5);
    }
}
```

Figure 6: Printing Load and analog value

Once finished writing the code, it must be verified by clicking on the check mark button  found at the top left of the toolbar, then the program will start compiling and check if there are any errors in the code. Once the verification is done, the code needs to be uploaded to the Arduino by clicking on  which is to the right of the check mark button. Finally we need to show the results on the screen which done by clicking on serial monitor button  which can be found at the top left of the toolbar.

Results

Now that the code is working, if an item is placed on the scale, the analog value reading and load in grams should appear on the serial monitor window. Figure 7 shows an example of the results.

```
analogValue: 59.30  
          load1: 2847.09130  
analogValue1: 64.80  
          load0: 0.00000  
FINAL WEIGHT: 2847.09130
```

Figure 7: Serial monitor window with load

In this example, an item that weights 2850 grams was placed on the scale, and the Arduinos reads that as 64.80, but the program calibrates it and return it back to grams, the result was 2847 grams which very accurate to the real weight.

Note that load0 is the output of another scale, so more than one scale could be used to measure weights, simply you will have create another circuit for second scale, and then follow the same procedure to add in the code what is needed to output the second scale. If an item is placed across both scales, then there will a reading from each scale, and the final weight is the sum of these two scales, in this case the second scale has nothing on it so the final weight is the same as load1.

Conclusion

Weighing scales can come be very beneficial in many industries ranging from small to bigger. For example; quality assurance engineers have to measure maximum force that could be applied to an airplane wheel, and that is helpful to determine the range of forces that a wheel can handle upon impact with the runway.

Resources

"Arduino Your Home & Environment." : *This Weeks Projects: Load Cells, Breadboard Power Supply, Temp / Humidity and More* N.p., 6 Apr. 2013. Web. 14 Nov. 2014.
<<http://arduinotronics.blogspot.com/2013/04/this-weeks-projects-load-cells.html>>.

References

- Instrumentation Amplifier used in this tutorial : INA125p
<http://www.ti.com/product/INA125/samplebuy?keyMatch=ina125p&tisearch=Search-EN>
- Scale used in this tutorial: Accuteck postal scale
http://www.amazon.com/Accuteck-Shipping-Display-Batteries-Adapter/dp/B005BTWMPQ/ref=sr_1_1?ie=UTF8&qid=1415943367&sr=8-1&keywords=accuteck+postal+scale
- Arduino used in this tutorial: Arduino Yun
http://www.amazon.com/Arduino-A000008-Y%C3%9AN/dp/B00F6YJK3S/ref=sr_1_1?ie=UTF8&qid=1415943865&sr=8-1&keywords=arduino+yun
- Downloading Arduino 1.5.8:
<http://arduino.cc/en/main/software>