NORTHERN ARIZONA UNIVERSITY

Department of Mechanical Engineering
Alonso Garcia
ME 476C - 001

To: Dr. Carson Pete

From: *Alonso Garcia*

Date: *02/28/2025*

Re: *HW 3: Self-Learning*

Introduction

This memo discusses what I learned from Arduino basic videos and Arduino application videos for measuring data, and how I applied it to my capstone project. Learning Arduino is very important to my capstone project, because our team will soon make a generator that needs to be tested for voltage, current, torque, and RPM measured from the generator. These tests can be done by getting a bunch of handheld sensors, but this wouldn't be easy to do when our Dynamometer (Dyno) is spinning at high speeds that can be deadly, if the generator breaks off from the Dyno. So, this design will help me improve my skill of software engineering.

Arduino Basics

After getting my certificate in "*Arduino Programming for Kids and Beginners with Tinkercad*" in Appendix A.1, it taught me how to use the basic functions of an Arduino, like using digital output, digital input, analog output, and analog input. With these, it allows data to pass to the Arduino, so it can be programmed to do a certain task. For example, this course wanted me to program with light-emitting diodes (LEDs), where it can turn on or off at any time I want. This is useful for me because learning the basic template of the Arduino is the basis of this whole project. Overall, this can be applied by making a template for any project to allow the code to compile.

Arduino Advanced Techniques

Another course I took was "*Arduino Step by Step Getting Serious*". After getting the certificate, in Appendix A.2, for this course, it taught me many technologies, like sensors, screens, and motors. For this I focused more on screens and sensors, since this upcoming project will be grabbing data and displaying it on a screen. This course used a graphics screen that is 128x64 and showed many methods that the screen has to be able to display data to the screen from the Arduino. This will be a key component for any upcoming projects that a user needs to see the displayed measured data. With this, it is like a software to a user that is seeing this data.

Application

After learning and getting certificates from the previous courses above, my project I wanted to make was creating a software that displays voltage, current, RPM, and torque.

I applied the information into a website called "Tinkercad". This website is a virtual Arduino simulator. With this, I chose an Arduino Uno, since this is the only one available. After choosing my Arduino, I grabbed a power supply, which represents my team's future generator outputting voltage from the Dyno. Next, I chose a 4x16 liquid crystal display (LCD), which will display the measured data. Sending current, torque, and RPM data is not possible within Tinkercad for my situation. However, it can be setup in code, which can be seen in Appendix B.

After setting up the environment for Tinkercad, I can now set up the circuits for the Arduino to read in data and write it back out as an output.

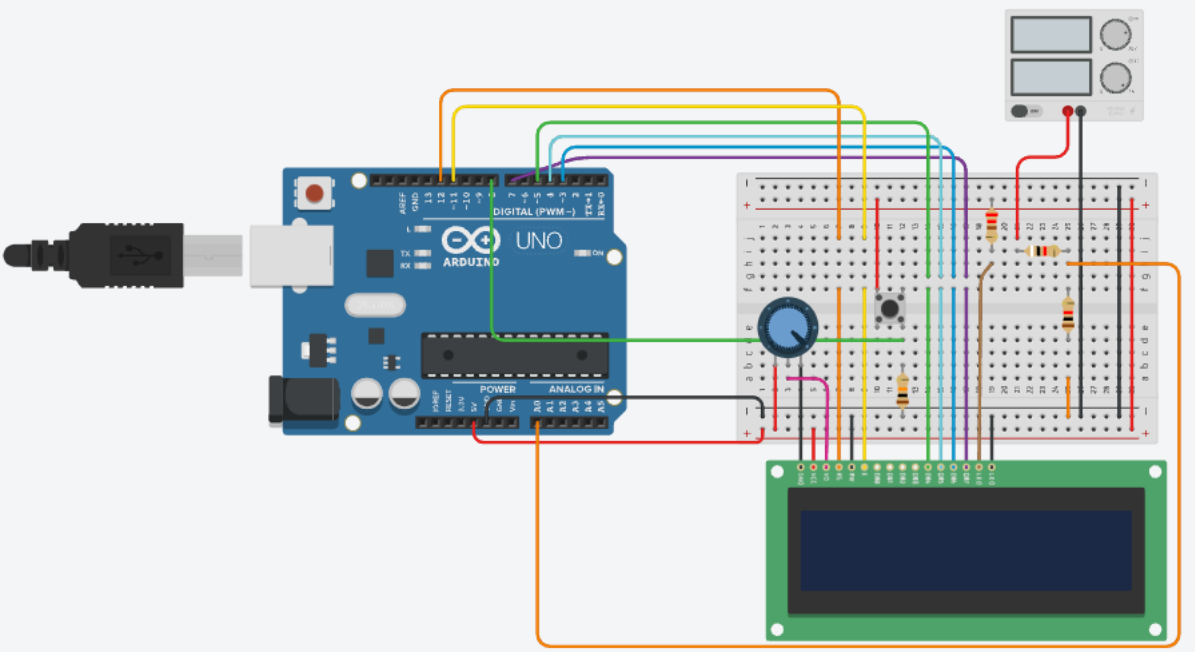

Figure 1 – Schematic of Arduino and circuits

In Figure 1, a schematic of all the parts is connected to the Arduino. The LCD screen is connected by many pins, where it allows altering data to the LCD screen. The power supply will represent the generator providing power when it's applied to the Dyno and changed from AC to DC. The power supply is supplying voltage to a voltage divider, which will be explained later. After all the wires are in their respective spots, it can be initialized to start displaying data.

LCD Screen

```
// LCD screen

  // Pins
  #define RS 12 // Register Select
  #define E 11  // Enable
  #define D4 5  // Decibel 4
  #define D5 4  // Decibel 5
  #define D6 3  // Decibel 6
  #define D7 7  // Decibel 7

  // Screen Size
  #define COLUMNS 16
  #define ROWS 2

  // Units
  #define VOLTAGE_UNIT "V"
  #define CURRENT_UNIT "A"
  #define RPM_UNIT "RPM"
  #define TORQUE_UNIT "N-m"
```

Figure 2 – LCD screen constants

In Figure 2, it is shown that constants are defined for the LCD screen. These are constants to make the code easier to read and allow modularity. For example, if pins need to be changed, you don't have to struggle finding it throughout the code when they're all together. In the three subsections, there is pin constants that define the pins on the Arduino, screen size that defines the size of the screen, and units that defines all the type of units for a specific measurement.

```
// LCD Screen
LiquidCrystal LCD_Screen( RS, E, D4, D5, D6, D7 );

void initializeLCDScreen();
void printMeasurement();
void clearLCDLine(int line);
```

Figure 3 – LCD screen variables and functions

In Figure 3, it shows the variables and functions related to the LCD screen. LCD_Screen is the actual object of the screen, where it can be used to display. InitializeLCDScreen() initializes the screen to properly display. printMeasurement() prints a measurement, like voltage, current, RPM, or torque, depending on the specific display number. Lastly, clearLCDLine() clears a specific line on the LCD screen.

Voltage Measuring

```
// Voltage measuring

 // Pin
 #define VOLTAGE_ANALOG_PIN A0

 // Attribute
 #define REFERENCE_VOLTAGE 5000.0 // mV (ACTUALLY CHECK FOR REFERENCE)
```

Figure 4 – Voltage measuring constants

In Figure 4, it shows the constants for voltage measuring. The first constant is defining the pin number in which the pin will receive the voltage. The reference voltage is referring to how much voltage is being sent to the whole circuit. It is set to 5 V, but this simulator in Tinkercad is assuming it's in an ideal world, that's why we can assume 5 V. So, once this Arduino is assembled, I have to measure the reference voltage to get accurate readings.

```
// Voltage
int multiplier = 10;
float totalVoltage;

void initializeVoltageInput();
void calculateVoltage();
float getVoltageInput( int pinNumber ); // Current also uses this
float digitalToAnalogConverter( int count ); // Current also uses this
```

Figure 5 – Voltage measuring variables and functions

In Figure 5, it shows variables and functions relating to voltage measuring. The variables multiplier and totalVoltage are related since the Arduino can only take 5 V max. The multiplier multiplies the voltage input and sets it into totalVoltage, which is then displayed. initializeVoltageInput() initializes the voltage measuring procedure by allowing the analog pin to read the voltage. The last three functions will be explained in depth as it contains many layers.

The last three functions are key components in finding voltage. As explained before, the Arduino can only take 5 V in the analog pins, otherwise it will be damaged. So, this is done by creating a voltage divider, which can be seen in Figure 1 near the right side. The top resistor is 9 kiloohms and the bottom resistor is 1 kiloohm. With this, this will make the input voltage be a tenth of the source voltage due to the equation

$$V_{out} = V_{in}\left(\frac{R_2}{R_1+R_2}\right) \quad (1)$$

The output voltage is the input voltage to the Arduino in this case. After getting the input voltage, the function calculateVoltage() starts the converting process since the Arduino turns the analog voltage to digital "voltage" that ranges from 0 to 1023. This is done by calling getVoltageInput() and once we get the digital "voltage", then digitalToAnalogConverter() is called. This then turns the digital "voltage" into analog voltage, which should be 3 V. After all of these processes are done, the value is then multiplied by the multiplier explained before. In this case, it's 10 since we provided a tenth of the source voltage to the Arduino. Now, we have a total voltage that can be displayed to the LCD screen.

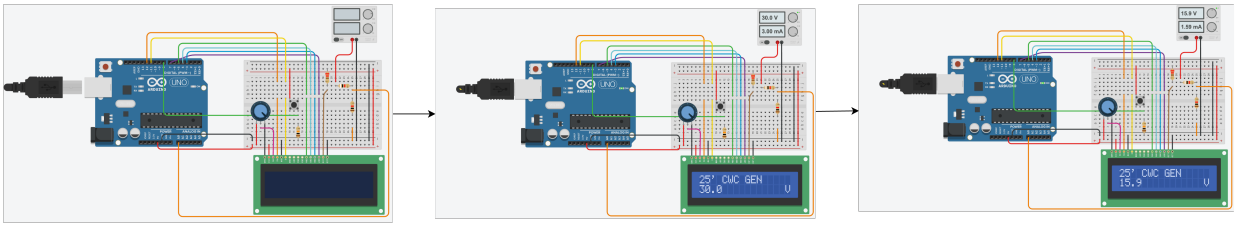

Figure 6 – Sequence of Arduino simulation

In Figure 6, it shows a sequence of what happens when the Arduino is turned on. As you can see in the power supply in the top right, we get the same value in the LCD screen, which means the calculations and reading is accurate and correct.

After viewing the sequence in Figure 6, I now have an implementation of measuring voltage for my capstone project, the generator, to test how much voltage is being sent from the generator.

Conclusion

In conclusion, learning the courses "Arduino Step by Step Getting Serious" and "*Arduino Programming for Kids and Beginners with Tinkercad*", taught me how to use an Arduino's functionalities and come up with my own project that will help with my capstone project's testing. With these skills, I applied it by making a schematic that simulates an Arduino measuring voltage from a power supply, which will be replaced with generator output. The results were that the amount of voltage supplied from the power supply was successfully displayed by using a voltage divider and conversions to get the total voltage. When I have an actual generator in a Dyno, it'll be able to measure the output voltage of the generator correctly. The last step when I apply this is making sure to check the reference voltage. This overall experience taught me a new skill of using an Arduino, which is important because testing our designs will require automation, which then makes everything less time consuming, efficient, and convenient.

Appendix

A.1

udemy

Certificate no: UC-b9e42265-bf89-4fbe-bcc8-71885b61a032
Certificate url: ude.my/UC-b9e42265-bf89-4fbe-bcc8-71885b61a032
Reference Number: 0004

CERTIFICATE OF COMPLETION

Arduino Programming for Kids and Beginners with Tinkercad

Instructors **Eshgin Guluzade**

Alonso Baltazar Garcia

Date **Feb. 28, 2025**
Length **6.5 total hours**

A.2

udemy

Certificate no: UC-ee120c64-519f-4757-ac39-4c362301419c
Certificate url: ude.my/UC-ee120c64-519f-4757-ac39-4c362301419c
Reference Number: 0004

CERTIFICATE OF COMPLETION

Arduino Step by Step Getting Serious

Instructors **Dr. Peter Dalmaris**

Alonso Baltazar Garcia

Date **Feb. 28, 2025**
Length **41 total hours**

B

```
// Libraries
#include <LiquidCrystal.h> // For displaying LCD screen

// Constants

 // LCD screen

  // Pins
  #define RS 12 // Register Select
  #define E 11  // Enable
  #define D4 5  // Decibel 4
  #define D5 4  // Decibel 5
  #define D6 3  // Decibel 6
  #define D7 7  // Decibel 7

  // Screen Size
  #define COLUMNS 16
  #define ROWS 2

                        // Units
                        #define VOLTAGE_UNIT "V"
                        #define CURRENT_UNIT "A"
                        #define RPM_UNIT "RPM"
                        #define TORQUE_UNIT "N-m"


 // IR sensor

  // Pin
  #define IR_PIN 2


 // Voltage measuring

  // Pin
  #define VOLTAGE_ANALOG_PIN A0

  // Attribute
  #define REFERENCE_VOLTAGE 5000.0 // mV (ACTUALLY CHECK FOR REFERENCE)


 // Current Sensor

                        // Pin
                        #define CURRENT_ANALOG_PIN A1

                        // Attribute
                        #define SENSITIVITY 0.185


 // Other
  #define DELAY_TIME 500 // ms
  #define ADC_RESOLUTION 1024
                        #define BUTTON_PIN 8

#define VOLTAGE_PER_COUNT REFERENCE_VOLTAGE / ADC_RESOLUTION

#define DECIMAL_PLACES 3 // Arduino only allows 6 decimal places before it's followed by zeros


// Initialize variables and functions for

 // LCD Screen
 LiquidCrystal LCD_Screen( RS, E, D4, D5, D6, D7 );

 void initializeLCDScreen();
 void printMeasurement();
 void clearLCDLine(int line);


 // Voltage
 int multiplier = 10;
 float totalVoltage;

 void initializeVoltageInput();
 void calculateVoltage();
 float getVoltageInput( int pinNumber ); // Current also uses this
 float digitalToAnalogConverter( int count ); // Current also uses this


 // Current Sensor
 void initializeCurrentInput();
 void calculateCurrent();


 // IR Sensor
 unsigned int rpm = 0;
 int PPR=1; // Pulse per rotation
 volatile unsigned int counter = 0;  // Counter variable for revolutions
 unsigned long previousMillis = millis();

 void initializeRPMInput();
 void calculateRPM();
 void IRinterrupt();


 // Button
 int previousButtonState = 0;
 unsigned long debounceDuration = 50; // ms
 unsigned long previousButtonPressMillis = millis();

 bool buttonWasPressed();
 void initializeButtonInput();
 void checkForButtonPress();

 // Other
 int displayNum = 0;

 void calculateMeasurement();

bool buttonWasPressed() {

 int buttonState;

 if ( millis() - previousButtonPressMillis >= debounceDuration ) {

   buttonState = digitalRead( BUTTON_PIN );

   if ( buttonState != previousButtonState ) {

                        previousButtonPressMillis = millis();
                        previousButtonState = buttonState;

                        return buttonState == HIGH;
   }
 }

 return false;
```

```
}

void checkForButtonPress() {
 if ( buttonWasPressed() ) {
   displayNum++;
   clearLCDLine( 1 );
 }
}

void calculateMeasurement() {
 switch ( displayNum % 3 ) {

   case 0:
    calculateVoltage();
                    break;

   case 1:
    calculateCurrent();
    break;

   case 2:
    calculateRPM();
    break;
 }
}

void setup() {

 initializeLCDScreen();
 initializeButtonInput();
 initializeVoltageInput();
 initializeCurrentInput();
 initializeRPMInput();
}

void loop() {

 checkForButtonPress();

 calculateMeasurement();
}

void initializeLCDScreen() {

 // Initialize LCD screen size
 LCD_Screen.begin( COLUMNS, ROWS );
 LCD_Screen.setCursor( 0, 0 );
 LCD_Screen.print( "25' CWC GEN" );
}

void initializeVoltageInput() {
 pinMode(VOLTAGE_ANALOG_PIN, INPUT);
}

void initializeCurrentInput() {
 pinMode(CURRENT_ANALOG_PIN, INPUT);
}

void initializeButtonInput() {
 pinMode(BUTTON_PIN, INPUT);
}

void initializeRPMInput() {
 pinMode(IR_PIN, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(IR_PIN), IRinterrupt, FALLING);
}

void calculateVoltage() {

 float voltageIn = getVoltageInput( VOLTAGE_ANALOG_PIN );

 totalVoltage = ( voltageIn * multiplier ) / 1000;

 printMeasurement( totalVoltage, VOLTAGE_UNIT );
}

void calculateCurrent() {

 float voltageIn = getVoltageInput( CURRENT_ANALOG_PIN );

 float current = ( voltageIn - 2.5 ) / SENSITIVITY;

 printMeasurement( current, CURRENT_UNIT );
}

void calculateRPM() {

 unsigned long currentMillis = millis();

 printMeasurement( rpm, RPM_UNIT );

 if (currentMillis - previousMillis >= 1000) {
   detachInterrupt(digitalPinToInterrupt(IR_PIN));
   rpm = (counter / PPR) * 60;  // Calculate RPM
   counter = 0;
   attachInterrupt(digitalPinToInterrupt(IR_PIN), IRinterrupt, FALLING);
   previousMillis = currentMillis;
 }
}

float getVoltageInput( int pinNumber ) {

 int count = analogRead( pinNumber );

 return digitalToAnalogConverter( count );

}

float digitalToAnalogConverter( int count ) {

 return count * VOLTAGE_PER_COUNT;

}

void printMeasurement( float value, String unit ) {

 int stringLength = unit.length();
 int cursorPosition = COLUMNS - stringLength;

 LCD_Screen.setCursor( 0, 1 );
 LCD_Screen.print( value, DECIMAL_PLACES );
 LCD_Screen.setCursor( cursorPosition, 1 );
 LCD_Screen.print( unit );

}

void IRinterrupt() {
   counter++;
}

void clearLCDLine(int line)
{
       LCD_Screen.setCursor( 0,line );
       for( int n = 0; n < COLUMNS; n++ )
       {
              LCD_Screen.print(" ");
       }
}
```

References

[1] A. Cartwright, "TUTORIAL: How to Measure / Read Voltages Into Arduino - (Part 3/3 Voltage Dividers - Analogue)," YouTube, https://www.youtube.com/watch?v=ZN5L6vdmi9s&list=PL6KWru1nEuZMvh1rJ8nKZDfa1iGZXZyrd&index=3 (accessed Feb. 28, 2025).

[2] A. Cartwright, "TUTORIAL: How to Measure / Read Voltages Into Arduino - (Part 2/3 Voltage Dividers)," YouTube, https://www.youtube.com/watch?v=hixEGmf1y5c&list=PL6KWru1nEuZMvh1rJ8nKZDfa1iGZXZyrd (accessed Feb. 28, 2025).

[3] A. Cartwright, "TUTORIAL: How to Measure / Read Voltages Into Arduino - (Part 1/3 Voltages Less than 5v)," YouTube, https://www.youtube.com/watch?v=lec7kPv3VS8&list=PL6KWru1nEuZMvh1rJ8nKZDfa1iGZXZyrd&index=1 (accessed Feb. 28, 2025).

[4] P. Dalmaris, "Arduino step by step getting serious | Udemy," Udemy, https://www.udemy.com/course/arduino-sbs-getting-serious/ (accessed Feb. 28, 2025).

[5] E. Guluzade, "Arduino programming for kids and beginners with Tinkercad | Udemy," Udemy, https://www.udemy.com/course/arduino-programming-for-kids-and-beginners-with-tinkercad/ (accessed Feb. 28, 2025).