DREXEL UNIVERSITY
Electrical and Computer Engineering
College of Engineering

Electric Car Sub-System

ECE 303 - 061: ECE Laboratory – Fall 2018
Instructor: Christopher Peters

Written By: Yonatan Carver
Farhan Muhammad

Date: 12/03/2018

Table of Contents

Lab 1 - Equipment Automation

Objective

The primary focus of this series of experiments is to write MATLAB scripts to automate the measurement equipment in the lab, namely the arbitrary waveform generator (AWG), digital multimeter (DMM), and the oscilloscope. The goal is to estimate the value of a resistor by sweeping the DC voltage out of the AWG and measuring the current, using the DMM, at that voltage, measured using the oscilloscope.

Firstly, the output voltage on the AWG will be changed manually and the data will be recorded from the DMM. In order to capture more data points more efficiently, scripts will be written for the equipment to estimate the value of the resistor. Afterwards, the process will be repeated using MATLAB to automate the voltage sweep and data recording. Finally, the oscilloscope will be tested to automate generating a signal of varying duty cycle and recording the corresponding on-times.

Equipment List

- BNC to BNC cable
- BNC to 2-alligator clip cable
- (2) banana-jack to alligator clip cables
- (1) 2 kΩ resistor
- HP Digital multimeter (DMM)
- HP Waveform generator (AWG)
- HP Oscilloscope

Procedure

The first experiment measures the current for a sweep of DC voltages manually set on the AWG. The voltage is swept from 0 to 10V in 1V increments. Both the voltage and current are recorded in order to cross-check the value of the resistor used.

For the second part of the experiment, the process is automated using MATLAB's USB commands. Both the AWG and DMM are set up using MATLAB to sweep the DC voltage from 0 to 10V in 0.1V increments and record the corresponding readings.

The third part of the experiment connects the AWG to the oscilloscope. Periodic pulse signals of known amplitude and varying duty cycle are automatically generated using MATLAB's USB commands. The duty cycle is swept from 20% to 80% and back down to 20% in 5% increments, and the corresponding on-times are recorded.

Finally, the data from all the experiments are plotted using MATLAB and linear regression is applied to each set of data to compare experimental results with best fit estimates.

Deliverables

1. Plot of Current vs. Applied Voltage for manually swept data set and accompanying best fit line.
2. Plot of Current vs. Applied Voltage for automatically swept data set and accompanying best fit line.
3. Plot of On-Time vs. Duty Cycle for automatically swept data set and accompanying best fit line.

Results

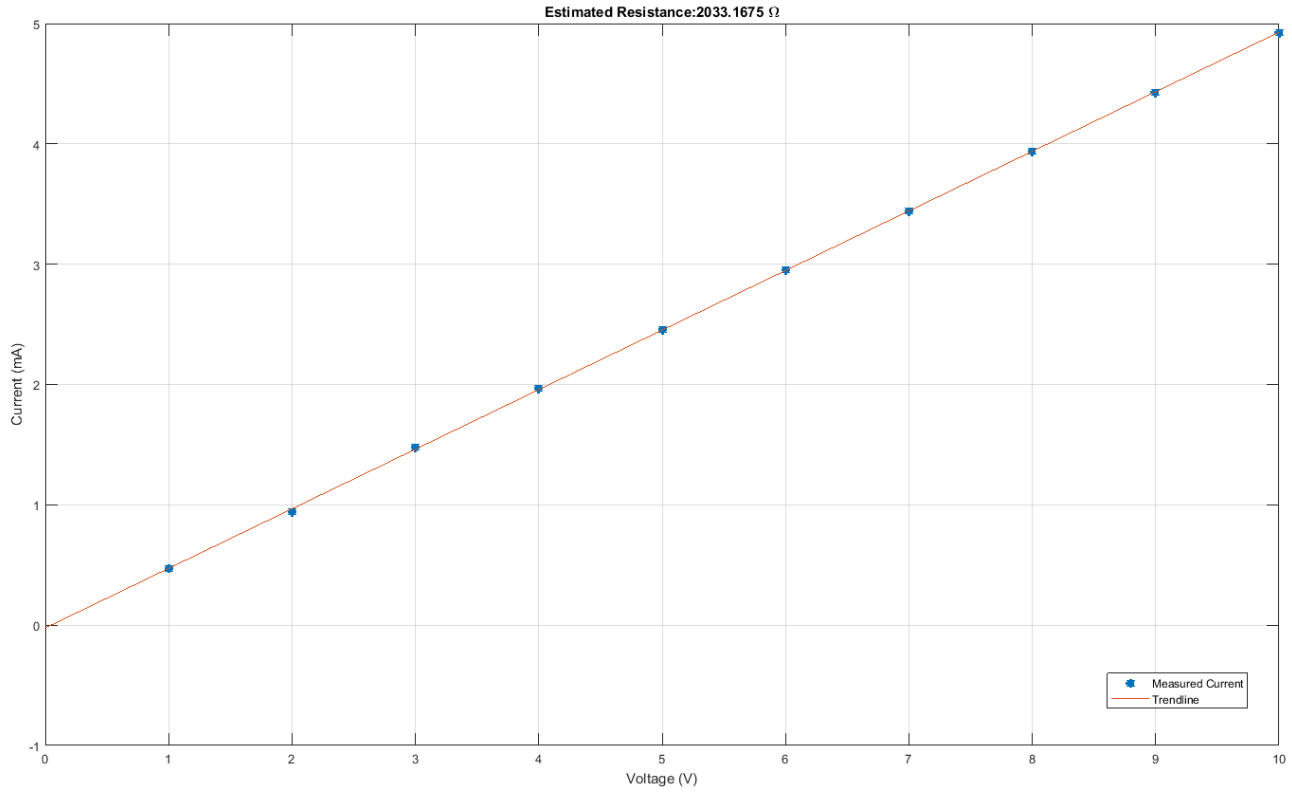

Figure 1: Voltage Vs. Current - Manual Measurement

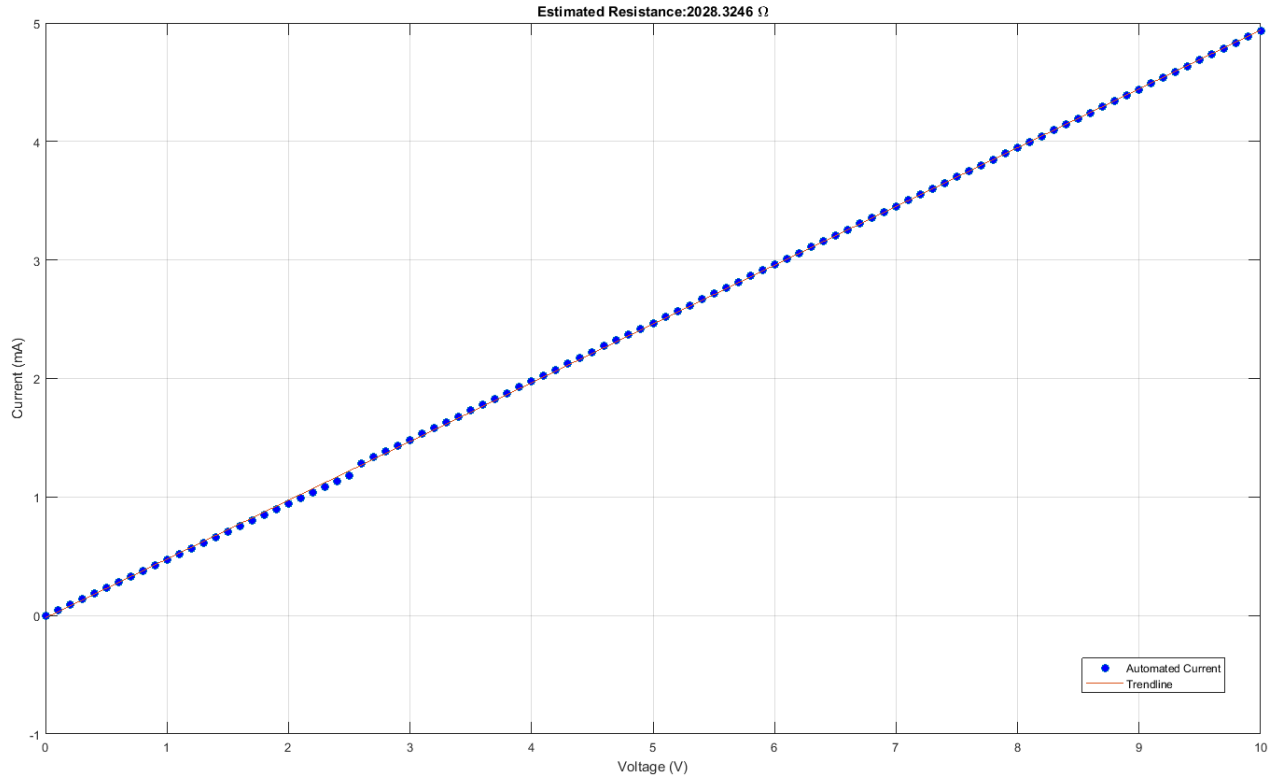

Figure 2: Voltage vs. Current - Automatic Measurement

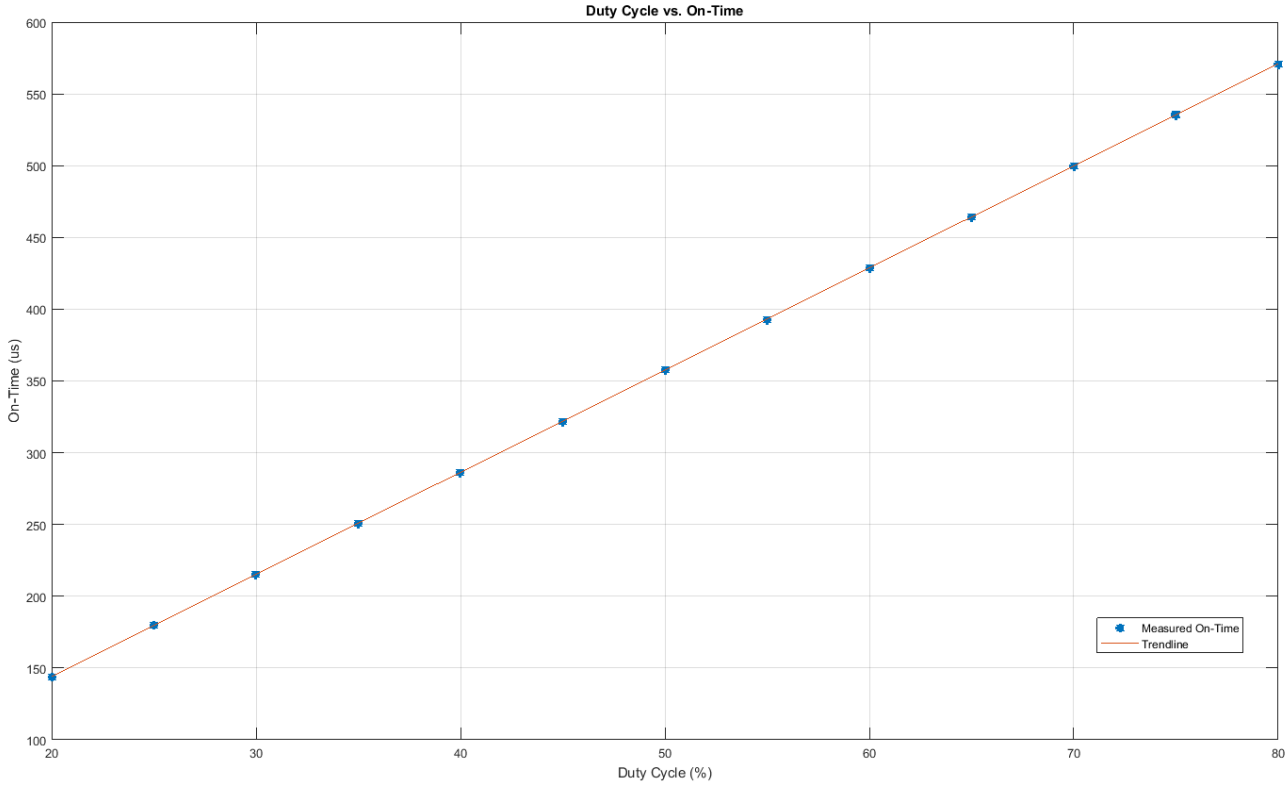

Figure 3: Duty Cycle vs. On Time - Automatic Measurement

Lab 2 – DC Motor Characterization

Objective

The primary focus of this experiment is to write a MATLAB script to automate the sweep of a DC voltage signal from the arbitrary waveform generator (AWG) which is supplied to an Arduino and used to generate a PWM signal of proportional duty cycle, which causes a wheel attached to a DC Motor to rotate. The goal of this experiment is to generate, using automation, the RPM vs. duty cycle characteristic plot of a DC motor using the Arduino as a PWM source.

Equipment List

- HP Oscilloscope
- HP Arbitrary Waveform Generator (AWG)
- Power Supply
- Arduino Mega 2560
- DC Motor with wheel
- H-bridge
- (2) Banana jack-to-alligator clip cables
- (3) BNC to Alligator clip cables
- Optical counter

Procedure

First, the H-bridge is set up and tested to ensure it is functioning correctly by connecting it to the motor and performing a test run using input voltage from the AWG. The optical counter is also connected properly, and its output is connected to the oscilloscope. Next, the Arduino is connected to the system. The voltage from the AWG is connected to an input pin of the Arduino, and one of its output pins is connected to the H-bridge. The AWG output is automated to sweep from 0 VDC to 5 VDC in 0.1 V increments each with a duration of 5s, using MATLAB's USB commands. The resulting signal is sent to the Arduino, which is programmed to adjust an output PWM signal with duty cycle based on the input voltage.

The resulting PWM signal sent to the DC motor causes the motor to accelerate over time until it reaches a 100% duty cycle at 5V. The corresponding RPM of the motor is recorded in MATLAB using two methods. The first is by measuring the pulse frequency of the optical counter via the oscilloscope every 5s. The second is by programming the Arduino to initiate an interrupt that manually counts the number of rising edges from the pulses for a second and then reset the counter. Using serial communication with MATLAB, the pulse count is sent to MATLAB every fifth second and is recorded within the MATLAB code. Once the voltage sweep is finished, the two frequencies are plotted against the corresponding DC voltage inputs in MATLAB for comparison and analysis.

Deliverables

1. MATLAB and Arduino code
2. Plot of RPM (optical sensor pulse frequency) vs. AWG voltage, with a linear fit to the data
3. Plot of RPM (using the counter approach) vs. AWG voltage, with a linear fit to the data
4. Plot of combined graphs for comparison

Results

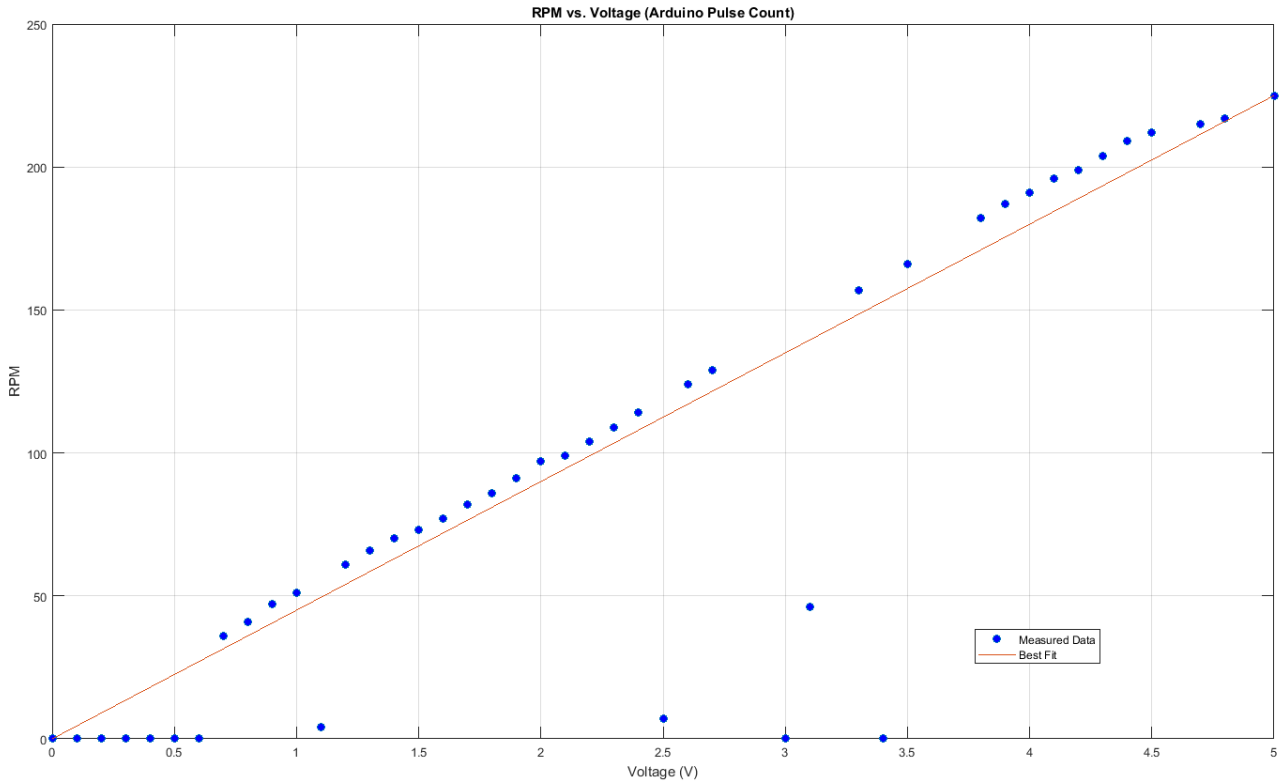

Figure 4: RPM vs. Voltage

Lab 3 – Battery Characterization

Objective

The first goal of this experiment was to write a MATLAB script to read the instantaneous voltage and current from a power source and record the results every second. The second goal of the experiment was to implement a fan motor that would begin spinning once the wheel motor (prepared in the previous experiment) hit 50% of its maximum RPM and stops again when the wheel falls back below 50%. The final goal of this experiment is to generate, using MATLAB, several graphs, including a graph of Battery Voltage vs. Time, Battery Current vs. Time, Battery Instantaneous Power vs. Time, and Battery Cumulative Energy Delivered vs. Time.

Equipment List

- HP Oscilloscope
- HP Digital Multimeter
- Fan with fan blades
- Arduino Mega 2560
- DC Motor with wheel
- H-bridge
- (4) Banana jack-to-alligator clip cables
- (2) 9V batteries
- Optical counter

Procedure

First, the H-bridge is set up and tested using a 9V battery as a power source (instead of the 12V power supply). The H-bridge is then further configured so that the IN1 pin acts as a PWM source. Leads are then connected to the battery from the multimeter, as well as the oscilloscope in order to measure current and voltage respectively. A MATLAB script is then written that will take the data every second until the voltage in the battery drops to 4.8V. The time taken, voltage and current are all recorded and used to produce the graphs.

Beyond measuring the battery as a power source, a fan motor needs to be implemented into the overall design. Code is added to the main Arduino that says if the RPM read in from the wheel motor, reaches 50% of a maximum threshold (set arbitrarily), a digital pin should be written high to send a signal to start the fan motor. In order to minimize the noise coming from the power supply in the circuit a separate bread board should be used to help isolate the fan circuit, capacitors should also be added across the terminals of the fan to filter out low and high frequency noise. These steps help mitigate noise and allow for better readings of the RPM from the optical sensor.

Deliverables

1. MATLAB and Arduino code
2. A fan motor that starts when the wheel spins at 50% of a pre-determined max RPM and stops when the wheel is operating lower than 50% of that max RPM
3. Plot of battery voltage vs. Time (demonstrating the battery being discharged)
4. Plot of battery current vs. Time (demonstrating the battery being discharged)
5. Plot of battery instantaneous power vs. Time (demonstrating the battery being discharged)
6. Plot of battery cumulative energy delivered vs. Time (demonstrating the battery being discharged)

Results

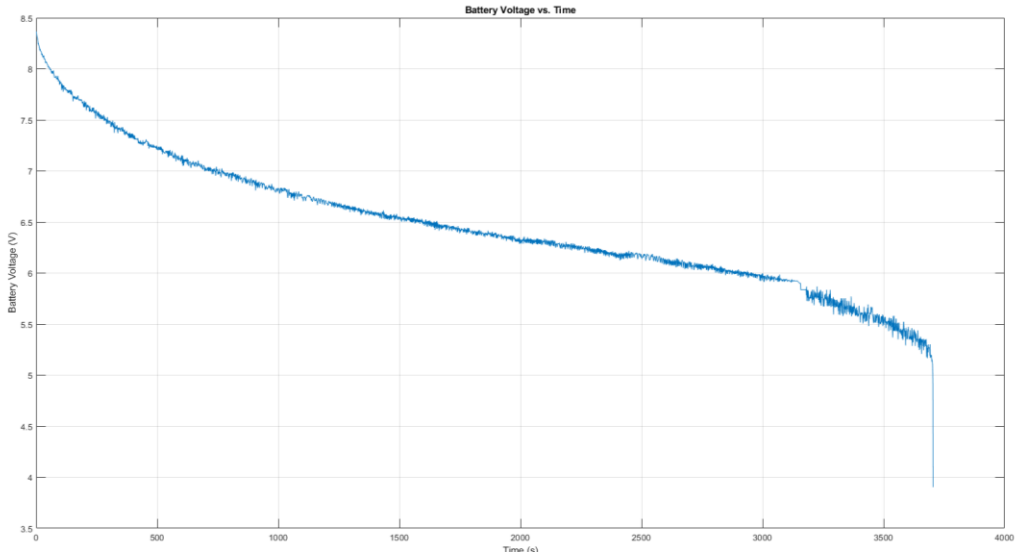

Figure 5: Voltage vs. Time Graph

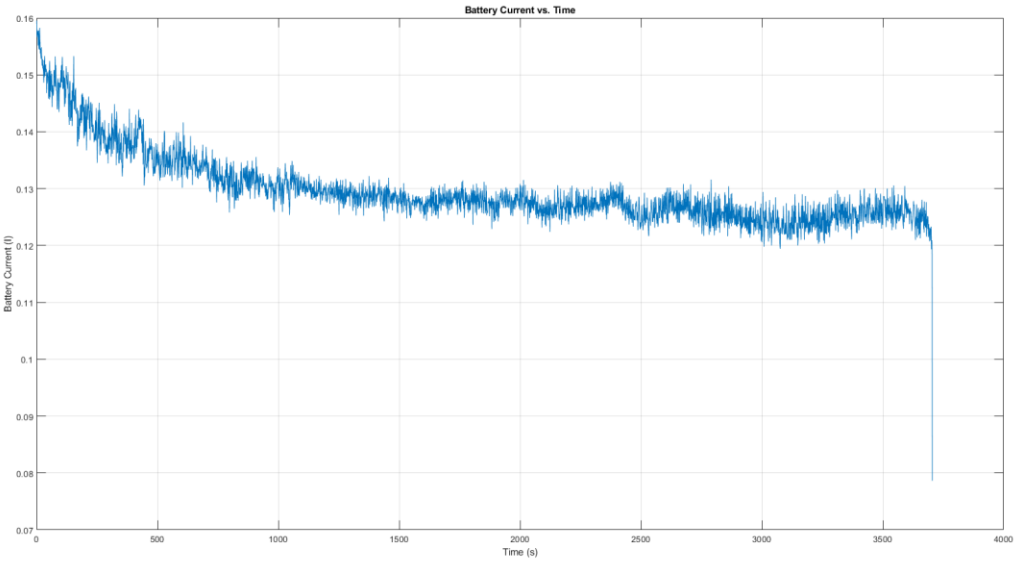

Figure 6: Current vs. Time Graph

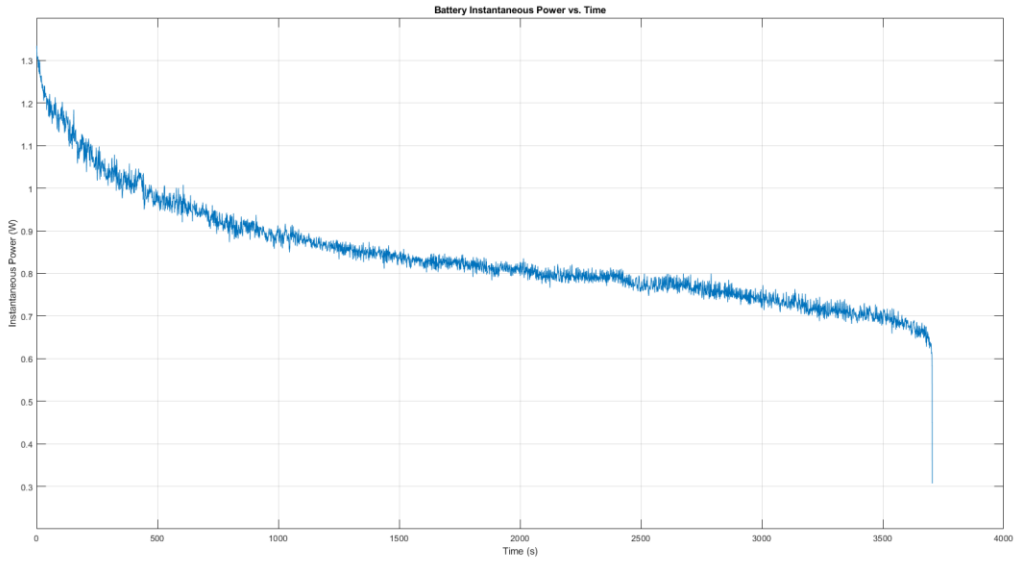

Figure 7: Instantaneous Power vs. Time Graph

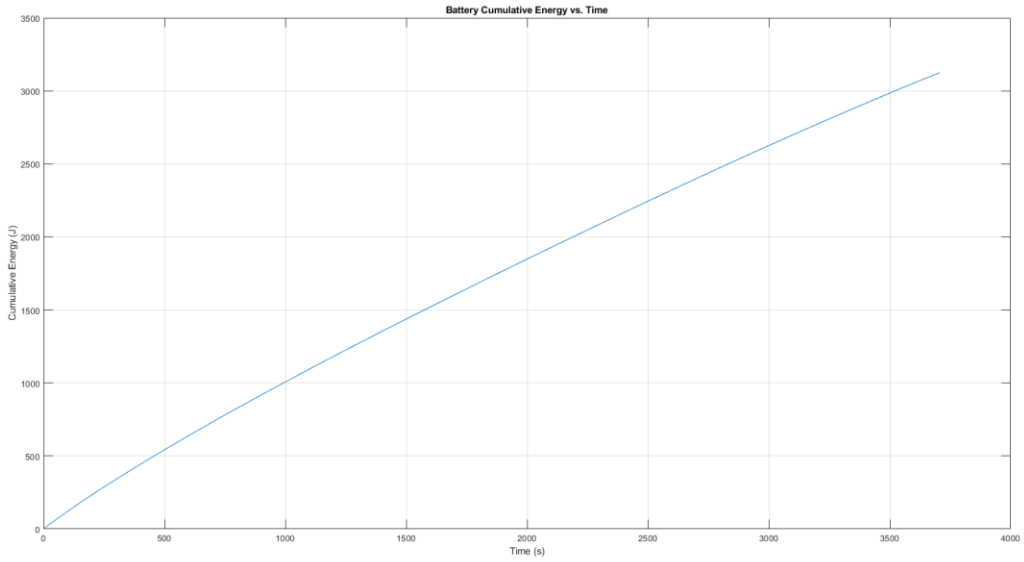

Figure 8: Cumulative Energy vs. Time Graph

Lab 4 – Data Acquisition and Emergency Shutdown Systems

Objective

The main goal of this lab is to devise a data acquisition system, using a secondary Arduino board, which collects RPM counts from the main controller, monitors the performance of the main motor as well as some characteristics of the battery, and initiates an emergency shutdown protocol based on certain triggers when normal functionality threshold values for the overall system are exceeded which triggers the main controller to use a relay to forcefully shutdown the system.

Equipment List

- Arduino Mega 2560
- Water Level Sensor
- DHT11 Temperature & Humidity Module
- R, B, W LEDs
- Passive Buzzer
- Relay Switch

Procedure

First, the circuitry for the sensors is built and connected to the DAQ. This consists of the water level and temperature sensors which are connected to two analog input pins of the Arduino board, as well as the three LEDs and passive buzzer which are connected to digital output pins. Optimal threshold values for the two sensors are determined experimentally and used to program the Arduino to trigger the alarm whenever these values are exceeded. Each of the three states of the DAQ (normal operation, low coolant level, and high battery temperature) are represented by the white, blue, and red LEDs respectively. While the system is under normal condition, no alarm is triggered and only the white LED is on. If at any point, the water level or temperature goes beyond the threshold then the white LED turns off and the corresponding LED turns on (possibly both). This also triggers the buzzer to turn on and emit an audible alarm.

Serial communication is used to communicate between the two Arduino boards as the main controller sends the DAQ the RPM count every second. This RPM count, along with water level and temperature readings are sent from the DAQ to the monitor, using serial communication again, to display the readings. The two Arduino boards are also connected through a digital pin. Whenever the alarm is triggered, the DAQ sends a digital high signal to the main controller which triggers it to open the relay and force the motor to perform the emergency shutdown.

Deliverables

1. Arduino code
2. Fully integrated and functional DAQ that –
    a. Implements a temperature sensor
    b. Implements a water level sensor
    c. Implements three different colored LEDs (white for normal operation, blue for low coolant, red for high temperature)
    d. Sends an audible alarm if a high temperature or low coolant level exists
    e. Receives RPM count from main controller
    f. Communicates with the main controller to shut down the motor upon sensor alarm
    g. Uses a relay to perform the emergency shutdown protocol
    h. Displays sensor information and RPM count

Results

There are no results to report and plot in this lab, however the fully integrated system was tested under several conditions and proper functionality was demonstrated. Through experimental observation and sensor threshold calibration, the water level sensor was determined to activate the alarm whenever its reading decreased below an analog input value of 280. And the temperature sensor was determined to activate the alarm at readings above 30℃.

Lab 5 – Data Display

Objective

The main goal of this lab is to create a graphical user interface (GUI) to display certain data sent from both the DAQ Arduino and the Main Controller Arduino. The RPM of the wheel, battery level, normal operation LED, high temperature LED, low coolant LED are among the pieces of information being displayed. In addition to displaying data, the GUI provides a way for the user to start and stop the system as well as input numerical values for the maximum load (the threshold at which the cooling fan will turn on) and calibration factor for the load cell.

Equipment List

- Arduino Mega 2560
- USB Cable
- MATLAB

Procedure

The graphical user interface (Figure 9) will be used as the main interface between both the data acquisition Arduino and the main controller Arduino. In order to receive data from both Arduinos, one must use serial communication. This includes being able to read and write from the serial port that the Arduino is connected to on the computer. The GUI parses over the string of data being sent from the Arduinos every second and updates (in real-time) the values for RPM, battery life, distance traveled, e-stop status, normal operation indicator, high temperature indicator, and low coolant indicator.

The user must also input a value for the maximum RPM. This value is used as the threshold at which the cooling fan will turn on – i.e. if the RPM of the wheel exceeds this value, the fan will turn on. The user must also input a value for the calibration factor. This value is used to adjust the calibration setting on the load cell. In order to send these two values to the Arduinos, the user must either press the “Send Data” button or stop and start the system.

In addition to displaying and sending data, the GUI has three buttons – Start, Stop, and Reset. In typical automation systems, if there is a fault or the system goes into e-stop, the user must “acknowledge” this issue by hitting the reset button. As such, the user must press the reset button prior to pressing the start button to run the system.

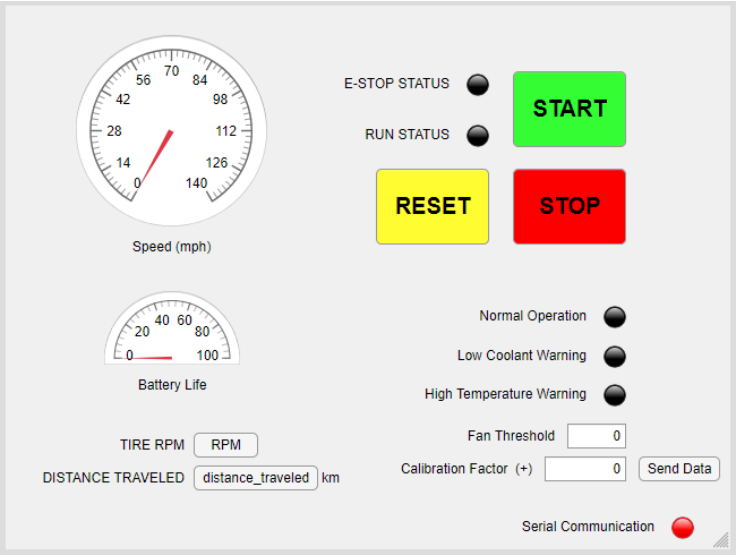

Figure 9: Graphical User Interface Layout

Deliverables

1. Graphical User Interface

Lab 6 – Load Cell and Calibration

Objective

The main goal of this lab is to add a component to simulate the accelerator foot pedal of the system by using a load cell. The reading from the load cell will then be amplified and the resulting signal will be sent to the main controller which will then be mapped and used to vary the PWM signal that is sent to the main motor of the wheel. First, the load cell will be calibrated using an Arduino test script. The resulting calibration factor will then be stored in the main Arduino's internal EEPROM and will be loaded and initialized upon startup for the rest of the project.

Equipment List

- Arduino Mega 2560
- Analog Wheatstone Bridge Load Cell
- HX711 Breakout Module

Procedure

First, the circuit for the load cell is set up and the load cell, amplifier, and Arduino are connected as shown below in Figure 10. Next, an Arduino script is written to measure the weight applied on the load cell and calibrate the readings by using known weights and adjusting the calibration factor to return the desired values.

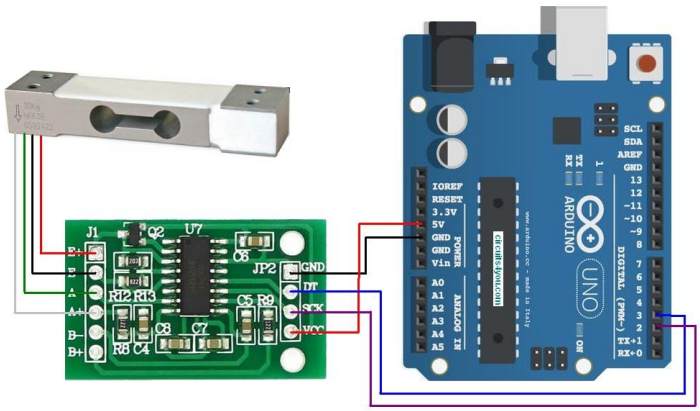

Figure 10: Circuit for Load Cell

Once the load cell is properly calibrated, it is added to the code for the main controller where the input from the load cell is constrained within a desired range of values. The resulting values are then mapped to output the PWM signal that will drive the wheel of the system. The functionality to set the desired maximum value of the load cell is added to the preexisting MATLAB GUI. This is done by adding a field in the GUI to reassign the calibration factor, which is sent from the GUI to the DAQ, and then to the main controller.

Finally, the EEPROM is added to the main controller such that it stores the user desired maximum RPM and calibration factor in specific addresses. The EEPROM functionality will include two Arduino functions, one that will read in the stored values and assign them accordingly upon startup and the another that will update the EEPROM addressed upon receipt of new values from the GUI. The EEPROM is set up using the EEPROMex library which allows a more user-friendly method of storing large values without concern for maximum space of each register in the EEPROM.

Deliverables

1. Arduino code
2. Integration of load cell and amplifier
3. Calibration system for reading weights in grams, settable from the GUI
4. Sensitivity parameter that maps load cell weight to motor speed by adjusting PWM signal
5. Functions that store and retrieve calibration and sensitivity parameters from EEPROM

Results

The base calibration factor acquired from experimenting with the given weights resulted in a value of -6266660. This factor was applied and tested using several different weights and the results were accurate to the nearest gram. The reading obtained from the calibrated load cell was constrained within a range of 0 to 100, where 100 is the initial maximum allowed weight and is subject to modification if desired. This range is then mapped to a range of 0 to 255 which represents the PWM signal that will be sent as output to the wheel motor to vary the motor RPM. The functionality of the EEPROM was also tested under several conditions which showed to successfully store, load, and update the calibration factor and maximum RPM values. A safety check was added to set the variables to the base values in the event that the EEPROM does not load properly.

Lab 7 – Finishing the Testbed

Objective

The main goal of this lab is two add two additional functionalities to the existing fully operational test bed. The list of options includes:

- RFID card reader-for security purposes
- LCD screen (with button to toggle display of different data)
- Remote operation with IR transmitter and sensor for speed control and system start/stop
- 4-Digit, 7-segment display for RPM display
- Accelerometer used to shut down system in case of excessive tilt or vibration

For this project, the 7-segment display and Accelerometer were chosen to be implemented.

Equipment List

- Existing Test Bed
- 4-Digit, 7-Segment Display
- GY-521 Module

Procedure

The 7-segment display is set up and tested first. It is connected to the DAQ Arduino using the pinout diagram shown below in Figure 11, and is programmed to update and display, in real time, the current RPM that is received by the DAQ from the main controller via serial communication. Each digit of the 7-segment display needs to be programmed and updated individually, so a function to breakdown the incoming RPM value into its separate digits and display them in the corresponding position of the 7-segment is written.

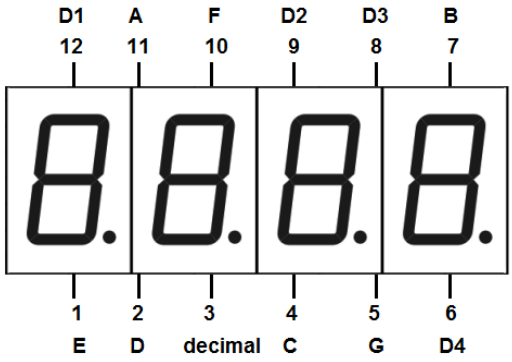

Figure 11: 4-Digit, 7-Segment Display Pinout

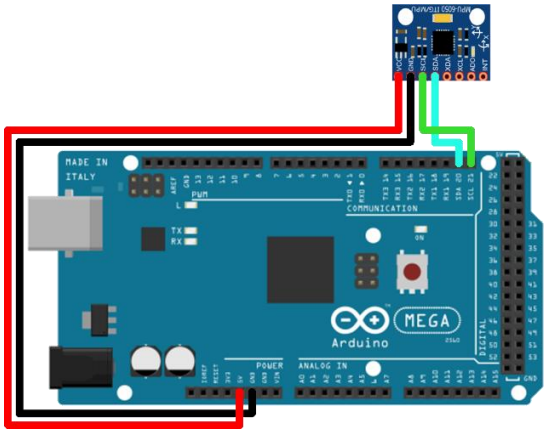

Figure 12: GY-521 Module Pinout

Next, the accelerometer is set up and tested using the GY-521 module. The breakout board for the module is connected to the main controller using the diagram shown below in Figure 12. This module requires the built-in “Wire” library in Arduino to initiate I2C communication. The module measures acceleration and tilt in the XYZ planes individually and sends the data through the communication channel to the Arduino where it is read and programmed to initiate the emergency shutdown protocol if any of the planes registers an excessive acceleration or tilt that crosses the set threshold values.

Deliverables

1. Arduino code
2. Full integration and functional system

Results

The 7-segment display updates each digit separately and so it was programmed to update from left to right. Due to circuit switching limitations, a small delay needed to be added in between the update of each digit in order to prevent short circuits. This delay was determined to be 4ms through repeated experiments.

The GY-521 module showed to have a high sensitivity to small changes in movement of the system. Small vibrations and tilts would result in large changes in the readings, so the thresholds for the movements were set to be relatively high; -8000 to 8000 for the X and Y planes, and 3000 to 19000 for the Z plane. These values were based on the initial, non-zero values that were being returned by the component even when it was stationary.