RYERSON UNIVERSITY
FACULTY OF ENGINEERING, ARCHITECTURE AND SCIENCE
DEPARTMENT OF AEROSPACE ENGINEERING

Digital Control for Fatigue Testing Machine Applied to Additively Manufactured Thermoplastics

Andrei Munteanu

AER870 – Aerospace Engineering Thesis - Final Report

Faculty Advisor: Dr. Seyed Hashemi

Date: April 2021

Acknowledgements

The author would like to thank, Dr. Seyed Mohammad Hashemi and Dr. Hamid Ghaemi for their valuable advice and contributions. Furthermore, the author would like to thank Dr. Soran Hassani Fard for his guidance and insight into his post-doctoral research. The author would like to thank Ben Dryden for his contribution towards the initial design of the mechanism for the testing machine. Lastly, the author would like to thank the FRAMES lab, and the Ryerson University Aerospace Department for their continued support during the 2020-2021 COVID-19 pandemic.

Abstract

A system that outputs force and number of applied force cycles with respect to time was designed to make a fatigue analysis machine. The design of electric circuit and the microcontroller Arduino code was developed and tested. The two subsystems comprise of a load cell for force readings and an electric motor for outputting cyclical forces to a test coupon. The subsystems should minimize error and maximize experimental modulation to validate the output data. The testing data of the subsystems of the fatigue apparatus was presented and analyzed to verify the design.

Contents

Nomenclature

Standardized Nomenclature

F – Force

GR – Gear Ratio

I – Current

ISR – Interrupt Service Routine

P - Power

PWM – Pulse-Width-Modulation

R - Resistance

RPM – Rotations Per Minute

T – Torque

t – Time

V – Voltage

W - Weight

Superscript and Subscript Characters

f_F – Force Application Frequency

f_{LC} – Load Cell Reading Frequency

f_M – Motor Shaft Frequency

f_P – Pulse Frequency

t_{LC} – Period of a Load Cell Reading Cycle

t_{PWM} – Period of a PWM Cycle

Table of Contents

Acknowledgements	2
Abstract	3
Contents.....	4
Nomenclature.....	4
List of Figures	7
List of Tables	7
Introduction	8
Theoretical Reviews and Developments	9
Load Cell Electrical Circuit	9
Load Cell Calibration	10
Experimental Apparatus and Procedure	11
Apparatus.....	11
Load Cell Calibration	11
Fatigue Analysis Test	11
Procedure	11
Load Cell Calibration	11
Fatigue Analysis Test	11
Electronic Control System Design	12
Design Objective and Requirements.....	12
Conceptual Design and Alternatives.....	12
Design Details, Analysis and Requirements Compliance	12
Presentation and Interpretation of the Results	14
Calibration Data.....	14
Conclusion.....	17
References	18
Appendices	19
Appendix 1 – Apparatus Photographs.....	19
Load Cell Electric Circuit	19
Load Cell Calibration Apparatus	20
Fatigue Testing Machine.....	21
Appendix 2 – Tabulated Data for Graphs.....	22
Sample Load Cell Testing.....	22

Load Cell Calibration	22
Appendix 3 – Product Specifications.....	24
Microcontroller	24
Load Cell.....	24
Amplifier.....	24
Motor	24
Motor Driver.....	24
Power Supply.....	24
Appendix 5 – Code	25
MATLAB Code.....	25
Arduino Code	26

List of Figures

Figure 1 - Wheatstone Bridge	9
Figure 2 – Amplifier	10
Figure 3 - Arduino Microcontroller.....	10
Figure 4 - System Architecture	13
Figure 5 - Test 2 Load Cell Analogue Readings	14
Figure 6 - Test 5 Load Cell Analogue Readings	15
Figure 7 - Test 6 Load Cell Analogue Readings	15
Figure 8 - Test 8 Load Cell Analogue Readings	16
Figure 9 - Test 9 Load Cell Analogue Readings	16
Figure 10 - Test 14 Load Cell Analogue Readings	16
Figure 11 - Load Cell Electric Circuit Photograph.....	19
Figure 12 - Calibration Apparatus with 20 Pounds Applied Photograph.....	20
Figure 13 - Calibration Apparatus with 40 Pounds Applied Photograph.....	20
Figure 14 - Fatigue Testing Machine Photograph.....	21

List of Tables

Table 1 – Analogue Readings for Calibration Tests.....	22
Table 2 - Test #6.....	22
Table 3 - Test #7.....	22
Table 4 - Test #8.....	22
Table 5 - Test #9.....	23
Table 6 - Test #11.....	23
Table 7 - Test #12.....	23
Table 8 - Test #13.....	23
Table 9 - Test #14.....	23

Introduction

3D printing, a type of additive manufacturing, has been continuously an area of research for implementation in the aerospace industry. One of the most significant constraints preventing manufacturing from implementing this technology has been the varying material and physical properties of 3D printed materials compared to forged and machined dyes for aerospace components subject to fatigue stress such as landing gear parts.

The scope of this thesis project was to design an electrical circuit and digital control system for a fatigue stress testing apparatus that can be used to test 3D printed thermoplastics. Furthermore, the apparatus design was tested and analyzed to verify functionality for the purpose of research being conducted for the Ryerson University, Aerospace Department, FRAMES Lab.

This design needs to satisfy the requirements set out by the fatigue analysis research. An S-N diagram needs to be graphed from the outputs of the testing apparatus by varying inputs. These inputs include stress cycle frequency, and geometry for sample testing coupons. Outputs are force and number of cycles with respect to time. The design must minimize error and be tested to estimate functionality and losses.

Theoretical Reviews and Developments

Load Cell Electrical Circuit

The electric circuit of a load cell is a Wheatstone Bridge, and can be seen in *Figure 1*.

Figure 1 - Wheatstone Bridge

The linear relationship, C_i , between a change in resistances, ΔR_i , to mechanical force, F , can be derived:

$$\Delta R_i = C_i F \quad (1)$$

In the middle of the four resistors, a voltage can be measured with respect to time:

$$V_m = V_{mo} + \Delta V \quad (2)$$

, where V_{mo} is the initially measured voltage, ΔV is the change in voltage over a fixed period.

The change in voltage is a function of the initial resistances, R_i , the change in resistances ΔR_i , and the excitation voltage, V_e :

$$\Delta V = f(R_i, \Delta R_i, V_e) \quad (3)$$

The measured voltage is passed through an amplifier that offsets and increases the relatively small changes in that signal.

Figure 2 – Amplifier

$$V_A = P + GV_m \quad (4)$$

, where V_A is the amplified voltage. P is the offset, and G is the gain of the amplifier.

The amplified voltage then gets read by the Arduino Microcontroller as an analogue value, A .

Figure 3 - Arduino Microcontroller

The analogue read value is a function of the amplified voltage and a reference voltage, provided by the microcontroller:

$$A = f(V_A, V_{ref}) \quad (5)$$

, where V_{ref} is the reference voltage.

Load Cell Calibration

Experimentally, two points can be obtained by using a reference weight. The initial analogue reading can be recorded when no weight is applied.

$$A_o = A(F = 0) \quad (6)$$

Furthermore, a second point can be recorded when a known reference weight is applied using a self leveling device with the sensor.

$$A_{ref} = A(F = F_{ref}) \quad (7)$$

The slope of the line connecting the two points a plane can be derived as follows:

$$F = \frac{(A - A_o)}{(A_{ref} - A_o)} F_{ref} \quad (8)$$

, where A_o is the initial analogue reading, and A_{ref} is the analogue reading after the reference weight, F_{ref} , was applied to the load cell. These are all known can be used to derive the relationship between the applied force, F , and the analogue reading, A .

Experimental Apparatus and Procedure

Apparatus

Load Cell Calibration

The load cell calibration apparatus is a self-leveling mechanism used to calibrate the load cell experimentally as described in the *Theoretical Reviews and Developments* section. Photographs of the load cell calibration apparatus can be found in *Appendix 1*. A circuit diagram of the subsystem can be referenced in *Figure 4*. Consecutive input weights of 0, 20, and 40 pounds were applied and then randomly removed and re-added at varying frequencies. The code for the calibration can be found in *Appendix 5*.

Fatigue Analysis Test

The fatigue analysis machine can be broken down into two subsystems. The load cell with its Arduino controller, and the motor that is operated by another controller. The controllers operate between each other through an interrupt that activates when the load cell reads 20% of nominal force. This lets motor know that the test coupon has failed, and it can stop the motor. This is when the test ends. The fatigue analysis apparatus can be found in *Appendix 1*. A full system architecture can be found in *Figure 4*.

Procedure

Load Cell Calibration

1. Structure must be properly fastened and nuts around load should not compress the load cell. Any compression from fasteners adds unnecessary pressure and can cause inconsistent readings.
2. Run the Arduino code using no parameter for `set_scale()`.
3. Place known weight.
4. Divide the resultant analogue reading to the known weight and pass it to `set_scale()`.
5. Adjust the parameter in *step 4* until an accurate reading is made.
6. Speed of weight application can be increased to determine maximum reading frequency so that error is minimized by using this value limit the fatigue apparatus mechanism frequency.

Fatigue Analysis Test

1. Use the MATLAB code, in *Appendix 5*, to derive the required pulses per revolution and PWM settings based on a desired force application frequency or vice-versa.
2. Check motor driver to be set to 400 pulses per revolution, or other if desired, by using the PA Setting switches and instructions on the driver.
3. Adjust the Arduino motor controller code inputs as needed.
4. Complete the electric circuit as described in *Figure 4*, while leaving the power supply disconnected. Note: Arduino motor controller must be supplied by a separate 9V battery.
5. Place 3D printed coupon between clamps and torque bolts such that test coupon is fixed properly to apparatus.
6. Using USB power cord, connect the load cell Arduino controller to computer with Arduino IDE and run the code to output force data with respect to time.
7. Connect power supply to outlet to begin fatigue analysis experiment. The motor stops automatically once 20% nominal force is detected by the load cell over a short period of time, meaning the test coupon has structurally failed and the test is over.

Electronic Control System Design

Design Objective and Requirements

The objective of this design was to output force and number of cycles with respect to time. Error should be minimized through design and testing methods to improve the accuracy of the data. This data would be able to produce S-N curves for the overarching scope of fatigue analysis research.

Conceptual Design and Alternatives

There were two options for digital control design when approaching this problem. The first was a design that used only one microcontroller to control both the motor and load cell readings. The issue with this design was that load cell reading signals interrupted the pulse-width-modulation (PWM).

Design Details, Analysis and Requirements Compliance

Therefore, a two Arduino controller design, whose system architecture can be viewed in *Figure 4*, was adopted to prevent any delays in the individual operations of each subsystem. An interrupt service routine (ISR) was implemented between the two controllers to automate the end of a test when the load sensor reads 20% of the nominal force over a non-instantaneous period relative to the comparative frequencies of applied force cycles to force readings.

An “if” condition that checks for the periodic increase of time based on a desired interval was the approach used in the software design of the load cell controller code. This avoids the use of the `delay()` function so that the ISR can check for the necessary condition for sending a falling signal to the motor controller as defined.

The final design code for each controller can be found in *Appendix 5*.

The fatigue analysis machine system architecture can be found below.

Figure 4 - System Architecture

Presentation and Interpretation of the Results

Calibration Data

Below are graphically presented the time dependent analogue readings. The sample data used for obtaining these graphs can be found in *Appendix 2*.

Figure 5 - Test 2 Load Cell Analogue Readings

In *Figure 5*, some of the earliest tests were recorded. Data was very inconsistent most likely due to failure of hardware. Root causes were deduced to be a malfunctioning component from the supplier, or a soldering failure. The issue was resolved by ordering new parts and re-applying solder to the circuit by following standard soldering techniques for making better quality pin connections.

Another change that was made to improve the readings, was an adjustment of the fasteners around the load cell, to relieve static compression on its surface.

Figure 6 - Test 5 Load Cell Analogue Readings

After making many hardware adjustments, a reading similar to a stepped change in output was observed, as seen in *Figure 6*.

Figure 7 - Test 6 Load Cell Analogue Readings

Furthermore, three steps of weights from 0 to 40 in increments of 20 were used to get a larger variety of output analogue readings with respect to reference inputs. This can be viewed in *Figure 7*.

Figure 8 - Test 8 Load Cell Analogue Readings

The tests were run for longer periods as seen in *Figure 8*.

Figure 9 - Test 9 Load Cell Analogue Readings

Frequency of weight application was increased to determine any physically limiting constraints imposed by the sensing instrument. This is observable in *Figure 9*.

Figure 10 - Test 14 Load Cell Analogue Readings

After applying the calibration factor multiple times to the input code, a consistent output of weight in pounds resulted from experimental procedure. These results can be seen in, *Figure 10*.

Conclusion

In conclusion, the electric circuit and digital control designs were made for the fatigue analysis machine. The load cell was the subsystem that was fully designed, tested, and optimized. The background theory pertaining to Arduino programming of load cell sensors was derived. The data for the load cell calibration was presented and analyzed. Full experimental apparatus definition and procedure for the calibration and fatigue analysis machine were outlined. Testing of the fully integrated fatigue analysis system with a 3D-printed PLA coupon is the only future work that remains to be completed, so that the apparatus can be used for its intended purpose as was described by the objective of this thesis project.

References

PCB GROUP Inc., Load Cell Handbook, 2014,
https://www.pcb.com/contentstore/MktgContent/LinkedDocuments/Load_Torque/LT-LoadCellHandbook_LowRes.pdf (Jan. 15, 2021).

Appendices

Appendix 1 – Apparatus Photographs

Load Cell Electric Circuit

Below, a photograph of the Load cell circuit being applied to the calibration apparatus is given.

Figure 11 - Load Cell Electric Circuit Photograph

Load Cell Calibration Apparatus

Below, the photographs of the calibration apparatus are given.

Figure 12 - Calibration Apparatus with 20 Pounds Applied Photograph

Figure 13 - Calibration Apparatus with 40 Pounds Applied Photograph

Fatigue Testing Machine

Below a photograph of the Fatigue Testing Machine is shown. Note that the load cell is missing from the central mechanism. By using threaded rods, spacers, and nuts, the load cell can be interfaced between the mechanism, in the central area of the machine, and the testing coupon, which is substituted by the wooden shims in the clamps for reference.

Figure 14 - Fatigue Testing Machine Photograph

Appendix 2 – Tabulated Data for Graphs

Sample Load Cell Testing

Below you have a table with sample analogue readings at different times for load cell experimental calibration. This data was used to derive the analogue reading with respect to time graphs.

Table 1 – Analogue Readings for Calibration Tests

Test	1	2	3	4	5	6	7	8	9	10	11	14
A	202	179	7	103137	287123	220979	30284	31425	32730	50686	45399	0
	202	26978	7	103137	282017	215760	30083	31230	32217	49326	41982	0
		-										
	203	34245	-1	103137	279442	212762	30037	31154	32112	48672	40672	0
	203	32352	7	103137	277997	210356	29827	31085	31941	48186	39725	0
	203	-8743	15	103137	277180	208731	29787	31057	31847	48158	38843	0
	204	179	15	103137	277023	206630	29785	31091	31522	47799	38858	20
		-										
	204	34244	7	103137	276121	205609	29915	31121	31624	47698	38528	20

Load Cell Calibration

For the following tables, average analogue readings and reference weights are presented to obtain the calibration factor. Each table represents a separate test.

Table 2 - Test #6

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	200000	-
1	20	285000	14000
2	40	375000	9375

Table 3 - Test #7

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	2850	-
1	20	12500	625
2	40	212500	5312.5

Table 4 - Test #8

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	3700	-
1	20	121900	6095
2	40	213300	5332.5

Table 5 - Test #9

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	3100	-
1	20	122500	6125
2	40	213500	5337.5

Table 6 - Test #11

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	3600	-
1	20	127000	
2	40	220000	

Table 7 - Test #12

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	0	-
1	20	9750	487.5
2	40	-	-

Table 8 - Test #13

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	0	-
1	20	185	9.25
2	40	-	-

Table 9 - Test #14

Weight Level	Input Weight (lb)	Output Load Cell	Calibration Factor
0	0	0	-
1	20	20	-
2	40	40	-

Appendix 3 – Product Specifications

The following are the fatigue apparatus electronic hardware main subcomponents:

Microcontroller

2 Arduino Uno Microcontrollers

Load Cell

Interface 1000lb load cell

Amplifier

HX711

Motor

Nema23 Stepper Motor with 20:1 step-up gear ratio

28A at 2.5V

Motor Driver

DM556 Stepper Driver

200 kHz maximum input pulse frequency

Power Supply

5.6 A at 115V input to 36V output

Appendix 5 – Code

MATLAB Code

Motor Controller Setting Derivation

```
clear clc
% This code derives the required PWM settings for the Arduino motor
% controller based on a desired force application frequency for the
% apparatus. Can be also used to do the reverse operation, depending on
% personal preference.
```

Motor Settings

Add your inputs here:

```
PulsePerRev = 400; % motor driver pulse per revolution setting
dutyCycle = 0.5; % 50% duty cycle
Period = 500; % milliseconds period/interval of full pulse cycle
t_on = dutyCycle*Period; % period of rising edge
t_off = Period - t_on; % period falling edge
MotorGearRatio = 20; % motor attachment 20:1 step-up torque gear ratio
```

Resultant Motor and Mechanism Speed

Resultant mechanism speeds are calculated here:

```
PulsePerSecond = 1/(Period/1000); % pulse frequency per second
MotorFrequency = PulsePerSecond/PulsePerRev; % motor speed frequency per second
MotorRPM = MotorFrequency*60 % above in RPM
MechFrequency = MotorFrequency/GearRatio; % motor gearbox frequency per second
MechCPM = GearFrequency*60 % above in cycles per minute
```

Arduino Code

Load Cell Calibration Code

```
/* Calibration */

#include <HX711.h> // Library

// HX711 Module connections
const int LoadCell_CLK_Pin = 4; // CLK of HX711 connected to pin 4 of Arduino
const int LoadCell_DOUT_Pin = 5; // DOUT of HX711 connected to pin 5 of Arduino

HX711 scale; // Initiation of library

void setup() {
  Serial.begin(57600);
  scale.begin(LoadCell_DOUT_Pin, LoadCell_CLK_Pin);
  scale.set_scale();
  scale.tare();
}

void loop() {
  long current_weight = scale.get_units(10); // get average of x scale readings
  long val = current_weight; // Use this line to add calibration factor
  Serial.println(val); // Print the scale factor to use
  delay(500);
}
```

Load Cell Controller Final Code

```
/* Load Cell Final Code */

#include <HX711.h> // Library

// HX711 Module connections
const int LoadCell_CLK_Pin = 4; // CLK of HX711 connected to pin 4 of Arduino
const int LoadCell_DOUT_Pin = 5; // DOUT of HX711 connected to pin 5 of Arduino
const int ISR_Out_Pin = 2; // Pin connection for ISR

HX711 scale; // Initiation of library

// Define Input Variables
int Pulse_state = LOW; // Initial state of pulse
long previousMillis = 0; // Will store the last time the pulse rised or fell
long Interval = 500; // Period between readings
long val = 0; // Output Force
long newval = 1; // It.
```

```

int cal = 1; // Calibration Factor

void setup()
{
  Serial.begin(57600);
  scale.begin(LoadCell_DOUT_Pin, LoadCell_CLK_Pin);
  scale.set_scale(cal);
  scale.tare();
  digitalWrite(ISR_Out_Pin, HIGH)
}

void loop()
{
  if(newval == 0.2*val)
    {digitalWrite(ISR_Out_Pin, LOW)}

  unsigned long currentMillis = millis(); // Reads current time since start of code

  if(currentMillis - previousMillis > Interval)
  {
    previousMillis = currentMillis; // Saves last time the pulse edge changed

    long current_weight = scale.get_units(10); // get average of 10 scale readings
    val = current_weight; // Use this line to add calibration factor
    Serial.println(millis()/1000); // Print the time since start in seconds
    Serial.println(val); // Print the scale factor to use
  }
  newval = val;
}

```

Motor Controller Final Code

```

/* Stepper Motor Test */

// Define Driver Module Pin Connections
int ISRpin = 2;
int Driver_DIR_Pin = 6; // Direction of Driver Connected to Pin 6 of Arduino
int Driver_PUL_Pin = 7; // Pulse of Driver Connected to Pin 7 of Arduino

// Define Input Variables for Motor
int Pulse_Width = 1000; // Pulse Width in Milliseconds (Rising Edge Active)
int Period_Off = 0; // Period of time that the Falling Edge is Active in Milliseconds

//Interrupt Variables
volatile int LoadFlag = 1;

void setup()

```

```

{
  // Motor Driver Pin Modes
  pinMode(Driver_DIR_Pin, OUTPUT);
  pinMode(Driver_PUL_Pin, OUTPUT);
  pinMode(ISRpin, INPUT_PULLUP); // Needs 10k pullup resistor to +5V
  attachInterrupt(digitalPinToInterrupt(ISRpin), ISR_Load, FALLING); //Initate interrupt: ISR_Load
  function to run when signal falls from high to low
}

void loop()
{
  if(LoadFlag == 1)
  {
    digitalWrite(Driver_DIR_Pin, LOW); // LOW = CW / HIGH = CCW Rotation of Motor
    digitalWrite(Driver_PUL_Pin, HIGH); // 4-5V Rising Pulse Edge Active
    delayMicroseconds(Pulse_Width); // Pulse Width Delay
    digitalWrite(Driver_PUL_Pin, LOW); // 0-0.5V Falling Pulse Edge Active
    delayMicroseconds(Period_Off); // Pulse Width Delay
    delay(0);
  }
  elseif(LoadFlag == 0)
  {
    digitalWrite(Driver_PUL_Pin, LOW); // 0-0.5V Falling Pulse Edge Active
  }
}

void ISR_Load()
{
  LoadFlag = 0;
}

```