FSAE Car Clutch Automation System

A Technical Report submitted to the Department of Mechanical and Aerospace Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Arthur Browne
Spring, 2022

Technical Project Team Members
Ethan Caldas
James Easter
Blake Garrett

Signature ____ Date __________
Arthur Browne

Approved ____ Date __________
Michael Momot, Department of Mechanical and Aerospace Engineering

Introduction

The objective of this project was to design a device that would automate the operation of the clutch on the VA Motorsports FSAE Car. A clutch is the component separating the engine from the driveline and what allows for the engine to run fast enough to not stall while the wheels are stopped or being brought up to speed to match the engine. This is accomplished through friction. A friction clutch, as found in any manual car, uses a friction plate sandwiched between a flywheel and a pressure plate. The flywheel is driven by the engine and the pressure plate is connected through the driveline to the wheels. The flywheel and pressure plate are both made of a steel alloy and are pulled together to sandwich the pressure plate by several large springs. The friction plate is made of a composite material designed to have a high coefficient of friction with the flywheel and pressure plate and to wear a minimal amount. In order to release the clutch and allow it to "slip", a throw-out bearing is used to force the pressure plate away from the flywheel which counters the force caused by the springs. This is what allows a clutch to slip, where the flywheel and pressure plate move at different speeds.

There are three positions that are important in a clutch. The first is the fully engaged position; this is where the full force of the springs holds the pressure plate against the friction plate and even the full load of the engine will not cause the flywheel or pressure plate to slip. In this state, the clutch acts as a rigid connection between the two sides. The second important position is the "bite point". The bite point is the point at which the throwout bearing has countered enough of the spring force from the pressure plate for the engine to transmit some power to the wheels without being bogged down and stalling or for the engine to overpower the friction between the tires and ground and cause the tires to slip. This is the most important of the three positions and as it is a very precise position, is the hardest to dial in. At the bite point, the two sides of the clutch may rotate at different speeds, but friction is transmitting torque from the engine side to the wheel side. The last position is the fully disengaged position in which the pressure plate has fully released force on the friction plate and the clutch is able to slip with no resistance. The wheels and engine act entirely independently of one another in this position (Nice, 2021).

This project is focused on the clutch used in the FSAE car, which employs the principles described above. It is the stock clutch used in a 2016 Yamaha R6 motorcycle. The clutch is operated using a cable to pull the throw-out bearing in stock form. The clutch was automated with the hope of reducing race-start inconsistency due to differences in driver experience. This is a large problem in FSAE competitions as multiple drivers are required and testing time in the cars is limited. Arguably, the most complicated part of driving the car is learning how to use the clutch competently, because it requires precise movement of the hand while pulling a heavy lever. This is further complicated by the fact the bite point position can change due to stretching in the cable. The system needs to be able to pull with roughly 120 lbs of force to precise positions and do so quickly. Solving this problem is of the utmost importance to give VA Motorsports an advantage over other teams during races. Our mission is to build a subsystem that gives Virginia Motorsports a competitive edge over other teams by improving consistency in off-the-line acceleration between drivers while continuing to design and build components with driver safety in mind.

Background

Research

In researching potential solutions to automating clutch actuation, several existing solutions were found. The current state of the art is found in Formula 1 racing. The clutch in a

Formula 1 car is operated through hydraulics which are in turn driven electronically. The driver has two paddles to be used at the start of the race. The position of the bite point is stored by the control unit and releasing one paddle takes the clutch from the disengaged (or in) position to the bite point. The second paddle then acts as an analogue input from the driver, who lets the clutch out to the fully engaged position. After the start of the race, operation of the clutch is entirely computer controlled with no driver input. This solution was not feasible in the context of this project as it is highly dependent on a high level of driver experience, the exact problem this project intends to address (Mitchell, 2013).

Next, the University of Michigan FSAE team solved the problem by directly mounting an electric motor to the clutch actuator. This is the simplest approach at clutch automation, however it was not feasible for us because it is extremely costly. Another downside to this approach was weight. The motor needed to be able to produce a large amount of torque in order to activate the clutch so it was very heavy. Adding to the weight problem was that the motor required a lot of power so an auxiliary battery was required to power the motor. Overall, the design added close to 50lbs of new components to a car that only weighs about 400lbs. This was an unsatisfactory design in that it accomplished the goal very inefficiently and added a lot of weight in a competition where a lightweight car is extremely advantageous (Chiu et al., 2015).

The third solution found was for launch control for road cars. This uses an electronic system to limit the engine speed to gain traction, rather than adjusting the clutch. This system also requires in depth tuning and information from many sensors. In order for the system to function properly information such as wheel speed, crankshaft position, throttle position, ignition timing, engine torque, engine acceleration, and tire temperature need to be known. These are normal sensors on a production car but not sensors that were incorporated into the design of the current FSAE car and are expensive as well as add weight and complexity. This solution requires a lot more knowledge of electronic launch control systems than our team had time to research, so this option was also not feasible to implement (Braga, 2021).

Constraints

The design needs to conform to several constraints. A major constraint is that of driver safety; a student will be driving the car, so in the case of a system failure, the driver should be safe. The system must also fit into the already very tight space of the FSAE car. The system must not impede the driver or any existing systems within the car. The whole system also needed to cost under $666, the budget allocated for this project. The system should be easily repairable and should it need to be removed from the car, reattaching the original cable operated clutch should be quick and easy. Troubleshooting the system must also be relatively simple. The user operated components of the system need to be easily accessible to the driver. The system should be reliable, as it works in many different driving conditions (wet, dry, incline, decline). The system needed to be completed prior to the end of term, December 7th.

Specifications

The only specification truly required to operate the clutch is the system needs to pull with a force of about 120lb for a distance of about a quarter of an inch. This was measured by pulling the clutch cable with a force gauge and measuring the distance travelled. This distance was measured directly at the throw-out bearing lever in order to eliminate any stretching of the cable or linkages in the clutch system. This is a relatively simple requirement before considering that the position needs to be precisely adjustable and actuation needs to occur in about the time it takes for an experienced driver to operate the clutch: less than 1-2 seconds.

Design Process

Concept Selection

Six designs were initially considered when concept screening the system: an electric motor with a gear set mounted directly to the throw-out bearing actuator lever (1), an electric motor with gears attached to a cable (2), two pneumatic linear actuators each responsible for one stage of the motion (3), a linear actuator attached to a hydraulic system (4), a hydraulic system driven by a motor through a rack and pinion (5), and an electric linear actuator connected to a cable (6). A number of important variables to consider were listed and the design options were judged as to whether they offered an improvement or deficit compared to the reference of the existing hand operated cable clutch. Through this concept screening the options were narrowed down to four designs. The table of rankings for concept screening can be found in appendix A. Next the four chosen design options were put through a scoring process.

Design	**Weight**	**Reference**	**1**	**4**	**5**	**6**
Speed	10	2	3	4	3	3
Accuracy	10	4	3	4	5	4
Ease of Use (set up)	7	4	2	3	3	3
Ease of repair	7	4	3	4	3	4
quick disconnect	1	3	4	4	4	5
Reliability	10	4	5	5	4	5
Replaceability	7	5	2	3	2	3
Repeatability (drift)	10	1	5	5	5	5
Cost	10	4	2	2	2	1
Weight	1	3	2	2	2	2
Failsafe	1	1	5	5	5	5
Physical Override	1	1	1	4	4	1
City Driving	10	3	5	5	5	5
Looks Good	10	2	5	5	5	5

Works with electronics	5	1	5	5	5	5
Score out of 100:		60.8	73.2	82	77.2	77.6
Rank			4	1	3	2

Table I. Concept Scoring.

Each of the considered attributes was given a weight to represent its relative importance and design options were given scores 1-5 which were multiplied by the weights and summed to provide a final score which was normalized to a percentage of a perfect score. Through this concept scoring process, design 4 was chosen: a hydraulic system driven by an electric linear actuator. A linear actuator would provide sufficient force and speed while remaining relatively cheap and the use of hydraulics allowed for the inclusion of an emergency physical override to the system, greatly increasing the safety in the case of a system failure.

Decision Making

Once the design concept was selected, a more detailed design process started. First the components were selected in the system based on the requirements. The initial components known to be needed for the system consisted of: a slave cylinder, a master cylinder, a manual override, a linear actuator, input devices like switches and potentiometers, and a microcontroller.

The slave cylinder was selected first and was based on calculation from the force and distance constraints in order to determine the best bore size and throw for the cylinder. Next, the master cylinder was selected and was matched to the slave cylinder so that the bore was the same size as the slave in order to have no hydraulic advantage. This was to preserve the force and distance profile for the linear actuator.

The linear actuator was selected based on force and speed requirements. Data sheets for linear actuators provide many statistics which needed to be considered. The most important to consider in this context are maximum load, speed, and total throw. Early on, it was decided that the linear actuator should gain its mechanical advantage through a lever arm rather than through a difference in hydraulic cylinder bore. This is because changing the length of the lever arm would be much simpler than changing out the hydraulic cylinder. For this reason, we were able to normalize the available actuators by assuming that the lever arm would translate the maximum output force into the 120 lbs needed to actuate the clutch. This lever arm lowered the force needed, while increasing the distance from ¼ inch of movement by the same ratio. This new distance was divided by each actuator’s speed under load to provide an estimate of the time necessary to fully actuate the clutch. Additional information about the actuators was considered such as price, ingress protection (resistance to water and dust), and whether built-in position feedback was available.

The final component selected was the manual override. It was decided that the simplest most cost-effective way of accomplishing this was to use a “drift brake” for a car to go in-line between the master and slave cylinder and act as an “emergency lever” in case the system were to fail and the clutch needed to be disengaged. Such a drift brake was originally designed to go in-line with a car’s brakes so that the driver could manually lock up the wheels to induce a slide. When not in use, the drift brake acts simply as if it were just a piece of the hydraulic line; when the lever is used by the driver it cuts off input (in our case from the actuator) and pressurizes the output hydraulic line itself, putting in the clutch.

The linear actuator used a simple mechanical lever to actuate the master cylinder that was designed so multiple mechanical advantages could be used and varied easily to fit our constraints on speed and force. All of the parts were then put into a Computer-Aided Design (CAD) model of the FSAE car and mounts were designed to hold the components and allow them to interact with each other. Overall decisions were made as a team using each member's experiences in engineering. Some members' experience with hydraulic systems, others' experience designing and machining mounts, and yet other members' experiences designing automated systems with internal feedback and external user input all proved extremely useful.

Standards

The standards this project must comply with are the rules governing the Formula SAE competition. There are relatively few rules governing the clutch use. Rule T.1.4 states:

All vehicle controls, including the shifter, must be operated from inside the cockpit without any part of the driver, including hands, arms or elbows, being outside of:

a. The Side Impact Structure defined in F.6.4 / F.7.6

b. Two longitudinal vertical planes parallel to the centerline of the chassis touching the uppermost member of the Side Impact Structure.

In our application, this refers to the drift brake hand clutch and the buttons controlling the linear actuator movement. The buttons are mounted on the back of the steering wheel, and the drift brake is mounted in the cockpit to the side of the driver. This mounting keeps it safely within the crash structure while allowing easy access from the driver. The only other rule governing our system is rule T.5.2.1, which states:

Exposed high speed final drivetrain equipment such as Continuously Variable Transmissions (CVTs), sprockets, gears, pulleys, torque converters, clutches, belt drives, clutch drives and electric motors, must be fitted with scatter shields intended to contain drivetrain parts in case of failure.

This rule refers to the actuator on the clutch and the linear actuator assembly. This does not refer to the hydraulic lines or the drift brake. This was solved by mounting the linear actuator and the clutch actuator behind the firewall. However, it could also be solved by making a scatter shield for the linear actuator itself. If given the time, a box would have been made for the linear actuator to protect it from oil and fuel spills as well as water. This box could have been made to the specifications required to function as a scatter shield as well.

Risk Analysis

The biggest risk of our system is if the system were to fail while driving. This would cause the engine to stall once the car slows down enough and leave the driver without any option to start the car again. As such, the code has been significantly edited and tested to ensure reliability, and why the manual override (emergency clutch) was added to the system. If the system fails and is unable to put the clutch in, the driver will still be able to do so. Unfortunately, if the system were to fail while the actuator is already holding in the clutch, the driver would have no way to override this leaving the engine running but the car unable to move. This, however, was deemed to be a safer failure mode and less likely to occur, as the clutch is out much more often than it is in. It will also usually only be put all the way in if the driver intends to come to a stop, or is starting the engine; should the failure be mechanical or hydraulic rather than electronic, the clutch would default to its natural position of out.

<u>Solution</u>

Final Design

Our final design utilized a linear actuator with a hydraulic system to operate the clutch. The linear actuator is controlled by an Arduino Uno R3, which takes user input from two switches and a potentiometer. The wiring diagram can be found in appendix B.The switches are mounted on the back of the steering wheel with paddles for easy operation. When the driver wishes to put in the clutch, both switches are held down. The actuator then moves to a predetermined position, depressing the piston rod of the hydraulic master cylinder through the lever arm. This pushes hydraulic fluid through the lines, through the emergency manual clutch, and into a slave cylinder, which pulls the clutch actuator to disengage the clutch plates. In normal operating conditions, the emergency clutch does not impact the hydraulic flow, allowing the system to operate as normal. However, if the handle on the drift brake is pulled, it can fully disengage the clutch without assistance from the linear actuator. A diagram of the hydraulic system layout can be found in appendix C. The driver may then release the right switch, commanding the linear actuator to move to the bite point. When the left switch is also released, the actuator moves all the way to remove pressure from the master cylinder, leaving the clutch engaged and transmitting torque.

The position of the bite point, as well as the speed that the actuator moves between locations, can be changed easily. The position of the bite point can be changed by the driver on the fly from inside the car using a potentiometer mounted to the dashboard. The speed the system activates can also be changed in the code running on the arduino. This is useful because although as-fast-as-possible is preferable for putting in the clutch and moving to the bite point, it can be helpful to slow down the movement from the bite point to fully out so that the engine does not stall nor do the tires slip. This is why, as described above, the second stage is manually controlled by the driver in a Formula 1 car. This allows for a large amount of potential tuning. Depending on the conditions and other circumstances – driver, engine rpm, etc. – different bite points and engagement speeds will be ideal, so designing a system with tunable, easy to change characteristics was important. This system also has a flexible mounting system where only the slave cylinder has to be mounted on the engine; the hydraulic lines allow the drift brake and the master cylinder to be mounted almost anywhere on the car. This is a major benefit over a system with a motor directly mounted to the clutch actuator. Those systems require a direct mechanical connection between the motor and the clutch actuator, which severely limits mounting options. This can cause less than ideal mounting locations of electronics very close to high heat sources such as the engine and exhaust. Furthermore, since the clutch actuator is located near the bottom of the engine, those systems have strict maximum height requirements. The engine is mounted as low as possible to ensure the lowest center of gravity for maximum cornering ability.

The use of a lever arm with the master cylinder additionally allowed us to adjust the force acting on the linear actuator. Multiple holes were drilled in the lever arm in order to allow us to easily change the mechanical advantage if the system needed to be faster or exert less force on the linear actuator. Although we were not able to mount the system on the car due to time constraints, the system was mounted on a Yamaha R6 motorcycle, a bike with the same engine as the formula SAE car, for testing purposes. Using this motorcycle as a test bed, we were able to confirm that the system works as intended, reliably fully depressing the clutch over multiple days of testing with no indication of wear. While initially concerned about the possibility that the linear actuator would move too slowly to effectively improve acceleration times, during the test setup we found that we needed to slow down the speed of actuation to prevent the clutch from engaging too quickly. Photos of the system mounted to the test motorcycle can be found in appendix D and the code used in appendices E and G.

Public Health and Safety

The emergency clutch lever was included mainly to ensure that the driver would always have a manual and direct connection to the clutch. In the event of a malfunction in our system, the driver has the ability to use the emergency clutch as a hand clutch and be able to stop and start the car. This is mainly with the goal of public safety, as the driver always has the ability to safely control the clutch and prevent an uncontrollable car.

GSCE Factors

Global, social, cultural, and environmental (GSCE) factors were not a large consideration given the narrow focus of this project. However, there is a fuel efficiency aspect of the Formula SAE competition, so improving fuel efficiency is an important part of the competition. This was one of the purposes of keeping the weight of our system to a minimum, as lighter cars are more fuel efficient.

Cost Analysis

The cost of our device in the configuration tested was $642. A commercial application of this device would cost even less. This design could be adapted easily for consumer applications where a master cylinder and an application lever like the drift brake are already required. If those components are subtracted, the cost drops to $480. The Formula SAE car is already designed as a consumer track car, and the application of this system, in order to make the car easier to drive, would align with the purpose of a formula race car built for ordinary drivers with no racing experience.

Conclusion

Our design worked as we expected, but the speed the linear actuator moved at was the variable we were most concerned about prior to testing. This turned out to not be an issue. We did learn that the linear actuator will attempt to force its way to a coded position even if it reaches a hard stop. This issue was easily resolved with a code to detect the mechanical limits of the system, but provided a small, unexpected challenge to our use of the linear actuator. However, our short time frame forced us to limit our testing to the test motorcycle. The design performed well on the test motorcycle and it was able to be safely ridden. This leaves potential work to be done to install the system in the car, as well as time to test different bite point and speed settings in order to determine which will produce the fastest acceleration times. The design also leaves the possibility for a more advanced launch algorithm. The car currently has wheel speed sensors installed, so those sensors could be used to detect wheel slip into an algorithm that automatically adjusts the clutch to achieve an even faster launch than our two stage system.

References

Braga, B. (2021, July 20). What is launch control, and how does it work? J.D. Power. Retrieved December 15, 2021, from https://www.jdpower.com/cars/shopping-guides/what-is-launch-control-and-how-does-it-work

Chiu, C., Karkos, P., Haldar, S., &amp; Qiu, C. (2015, December 14). Team 30: Electronic actuation of a motorcycle. University of Michigan Library. Retrieved December 15, 2021, from https://deepblue.lib.umich.edu/bitstream/handle/2027.42/117348/ME450-F15-Project30-FinalReport.pdf;sequence

Mitchell, S. (2013, September 17). Inside F1 cars: Does a Formula 1 car have a clutch, and how does it work? Bleacher Report. Retrieved December 15, 2021, from https://bleacherreport.com/articles/1776973-inside-f1-cars-the-technology-application-and-importance-of-a-formula-1-clutch#

Nice, K., Bryant, C. W., &amp; Hall-Geisler, K. (2021, July 30). How clutches work. HowStuffWorks. Retrieved December 15, 2021, from https://auto.howstuffworks.com/clutch.htm

<u>**Appendix**</u>

Appendix A: Concept Screening

Design Options: an electric motor with a gear set mounted directly to the throwout bearing actuator lever (1), an electric motor with gears attached to a cable (2), two pneumatic linear actuators each responsible for one stage of the motion (3), a linear actuator attached to a hydraulic system (4), a hydraulic system driven by a motor through a rack and pinion (5), and an electric linear actuator connected to a cable (6).

Design	**1**	**2**	**3**	**4**	**5**	**6**	**Reference**
Speed	+	+	+	+	+	+	0
Accuracy	0	0	+	+	+	+	0
Ease of Use (set up)	0	0	-	+	+	+	0
Ease of repair	+	+	-	+	+	+	0
quick disconnect	+	0	-	+	+	+	0
Reliability	-	0	-	0	0	-	0
Replaceability	-	-	-	-	-	-	0
Repeatability (drift)	+	0	+	+	+	+	0
Cost	-	-	-	-	-	-	0
Weight	-	-	-	-	-	-	0
Failsafe	+	+	0	+	+	+	0
Physical Override	-	-	-	0	0	-	0
City Driving	+	+	-	+	+	+	0
Works with electronics	+	+	+	+	+	+	0
Plus	7	5	4	9	9	9	
Sames	2	5	1	2	2	0	
Minus	5	4	10	3	3	5	
Net Score	**2**	**1**	**-6**	**6**	**6**	**4**	
Rank	**4**	**5**	**6**	**1**	**1**	**3**	
Continue?	Yes	No	No	Yes	Yes	Yes	

Table II. Concept screening

Appendix B: Wiring Diagram

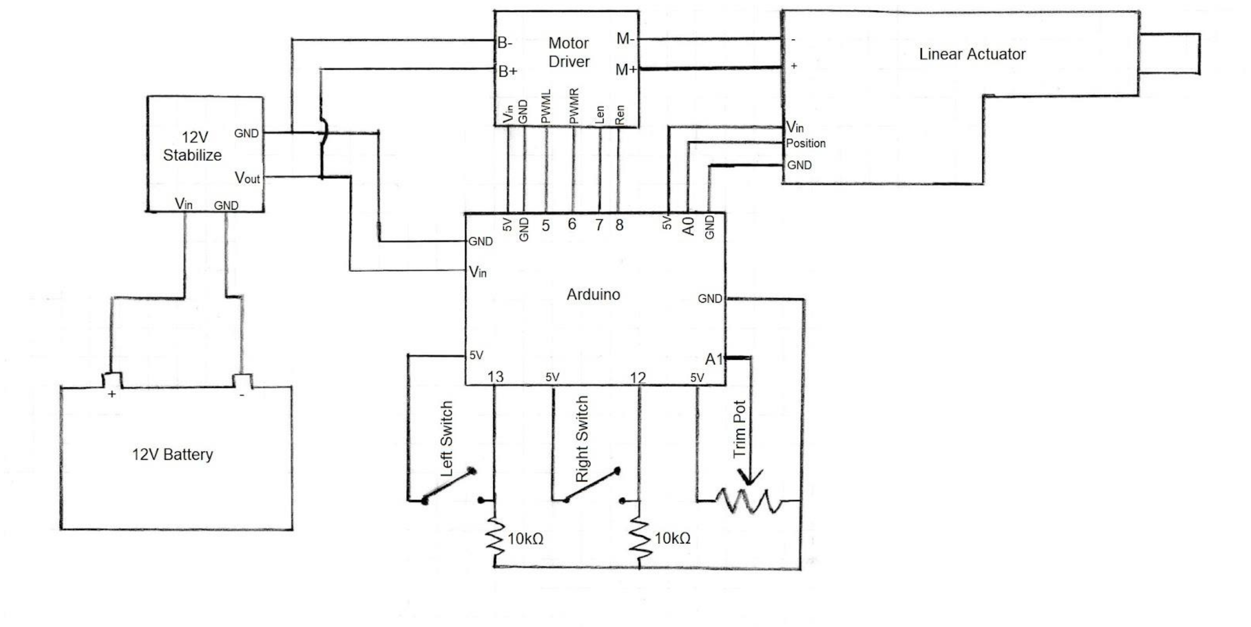


Figure I. Wiring Diagram

*The two switches were wired with pull-up rather than pull-down resistors in the final system

*Some arduino pins were changed to ease soldering

Appendix C: Hydraulic System Diagram

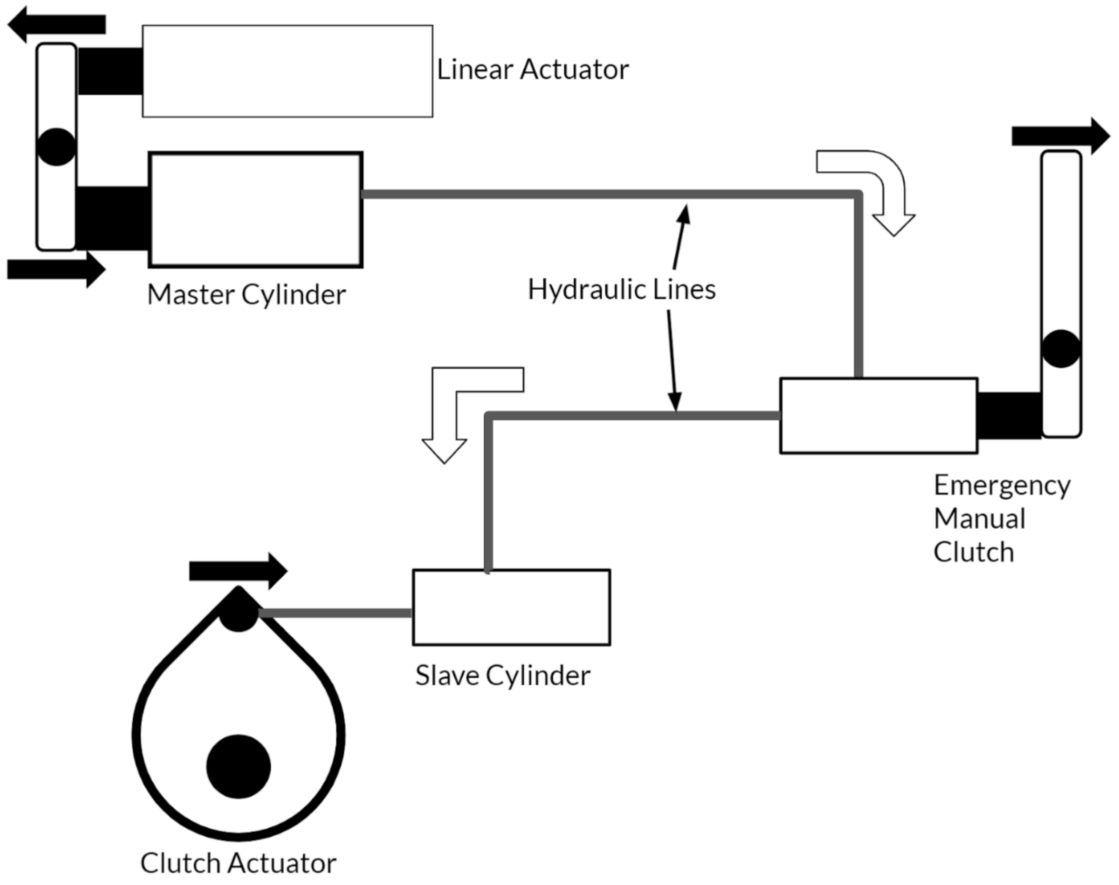


Figure II. Hydraulic system

Movement marked with black arrows

Flow of hydraulic fluid marked with white arrows

Appendix D: Photos of system components

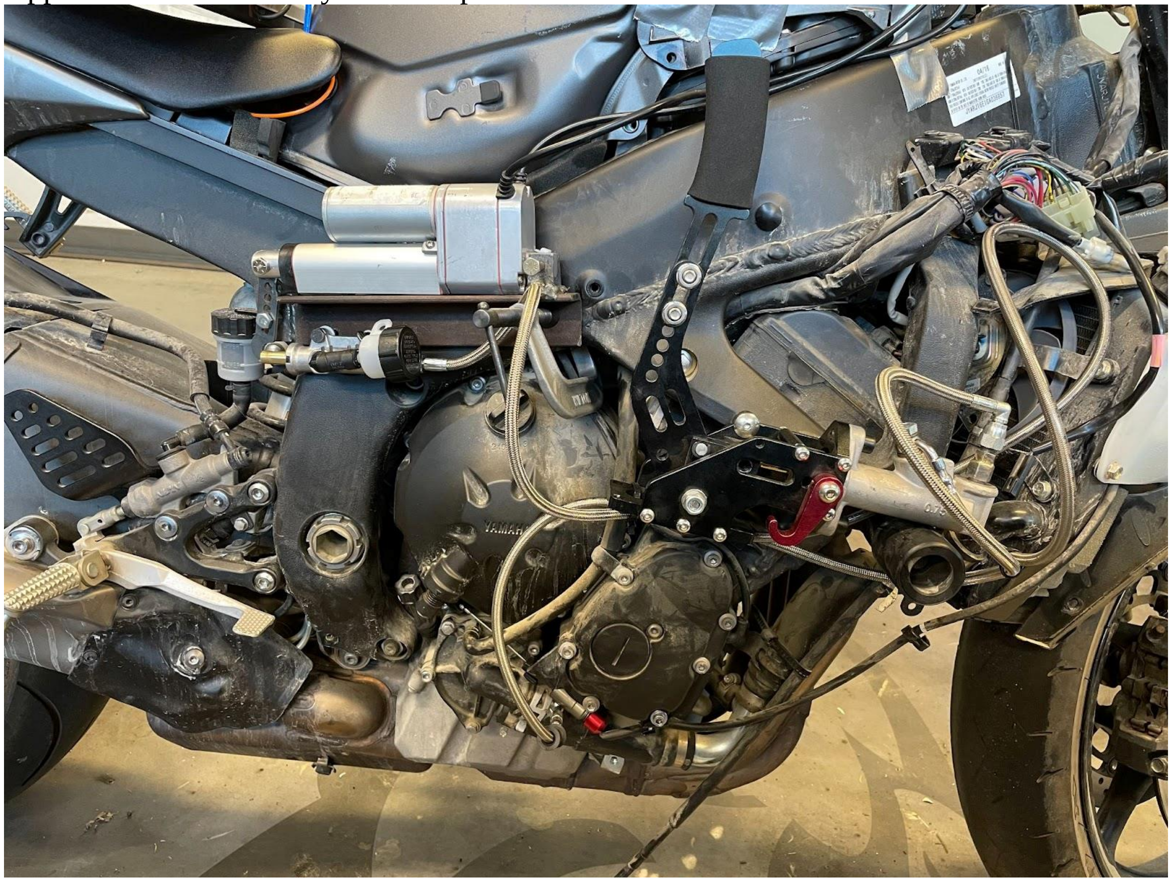


Figure III. System mounted to test motorcycle

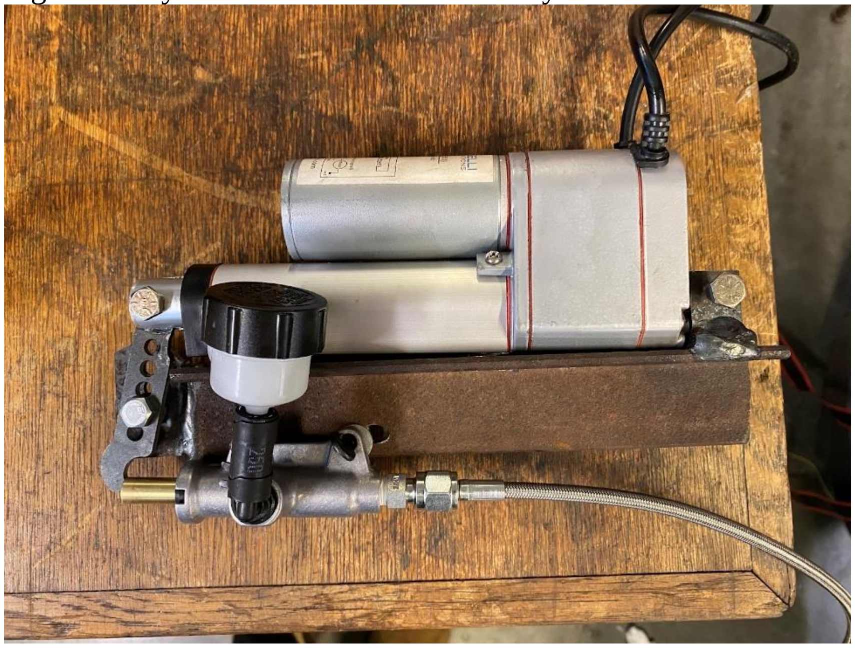


Figure IV. Linear actuator and master cylinder

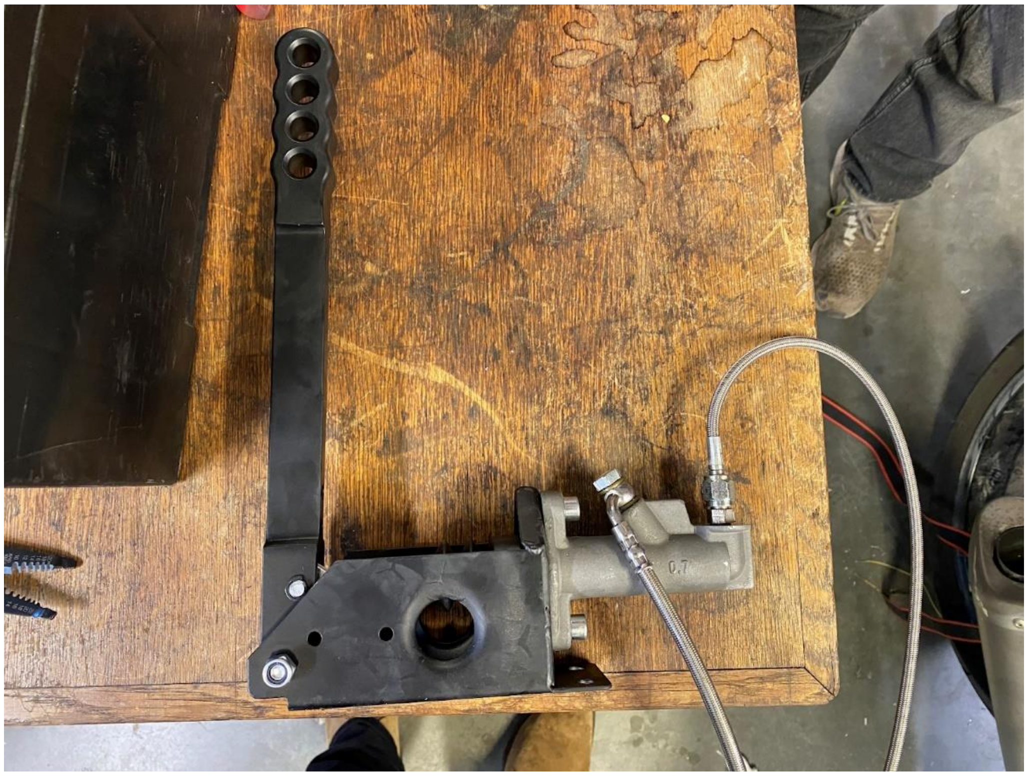
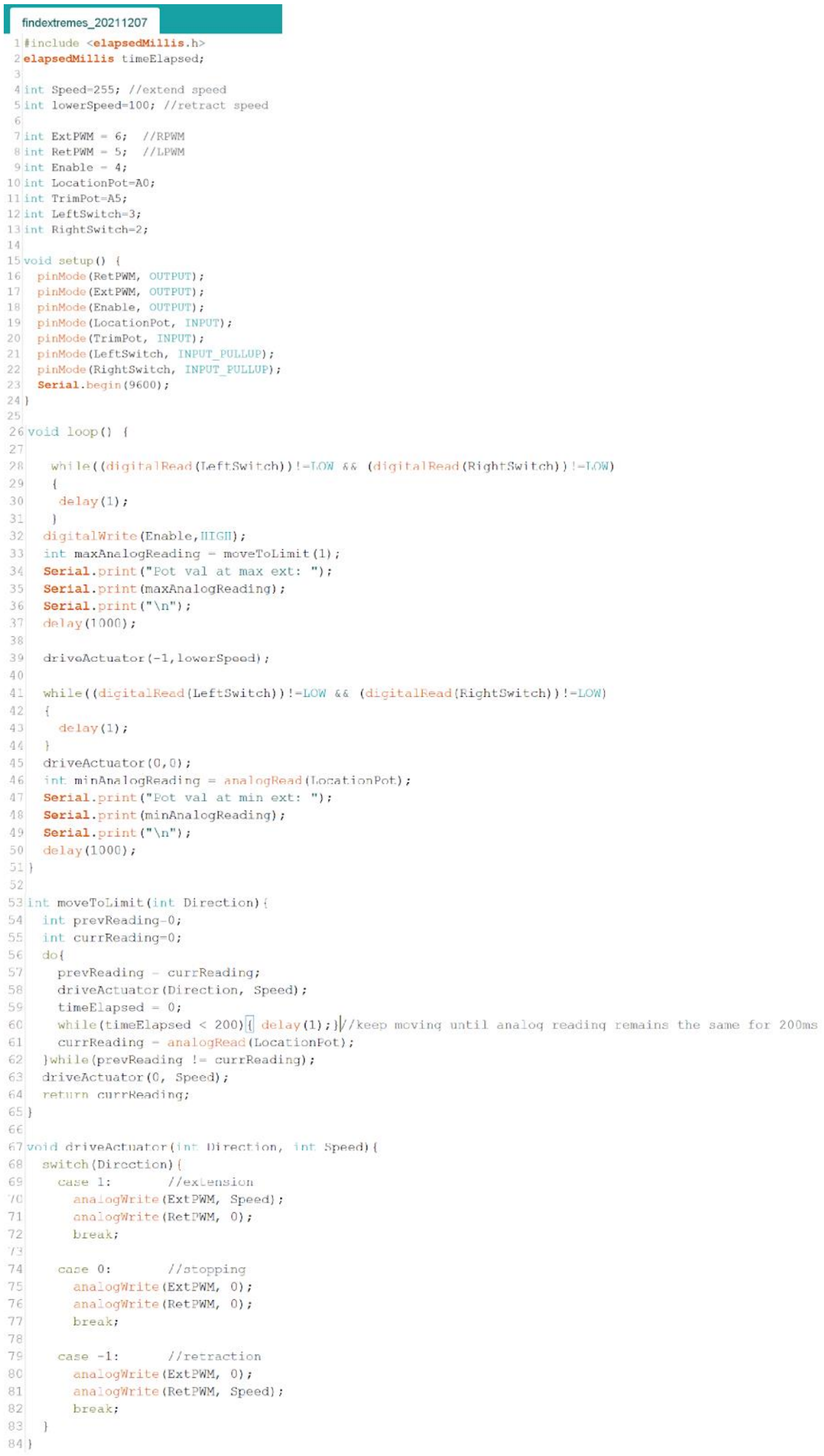


Figure V. Emergency manual clutch

Appendix E: FindExtremes Code for testing stall points of the actuator

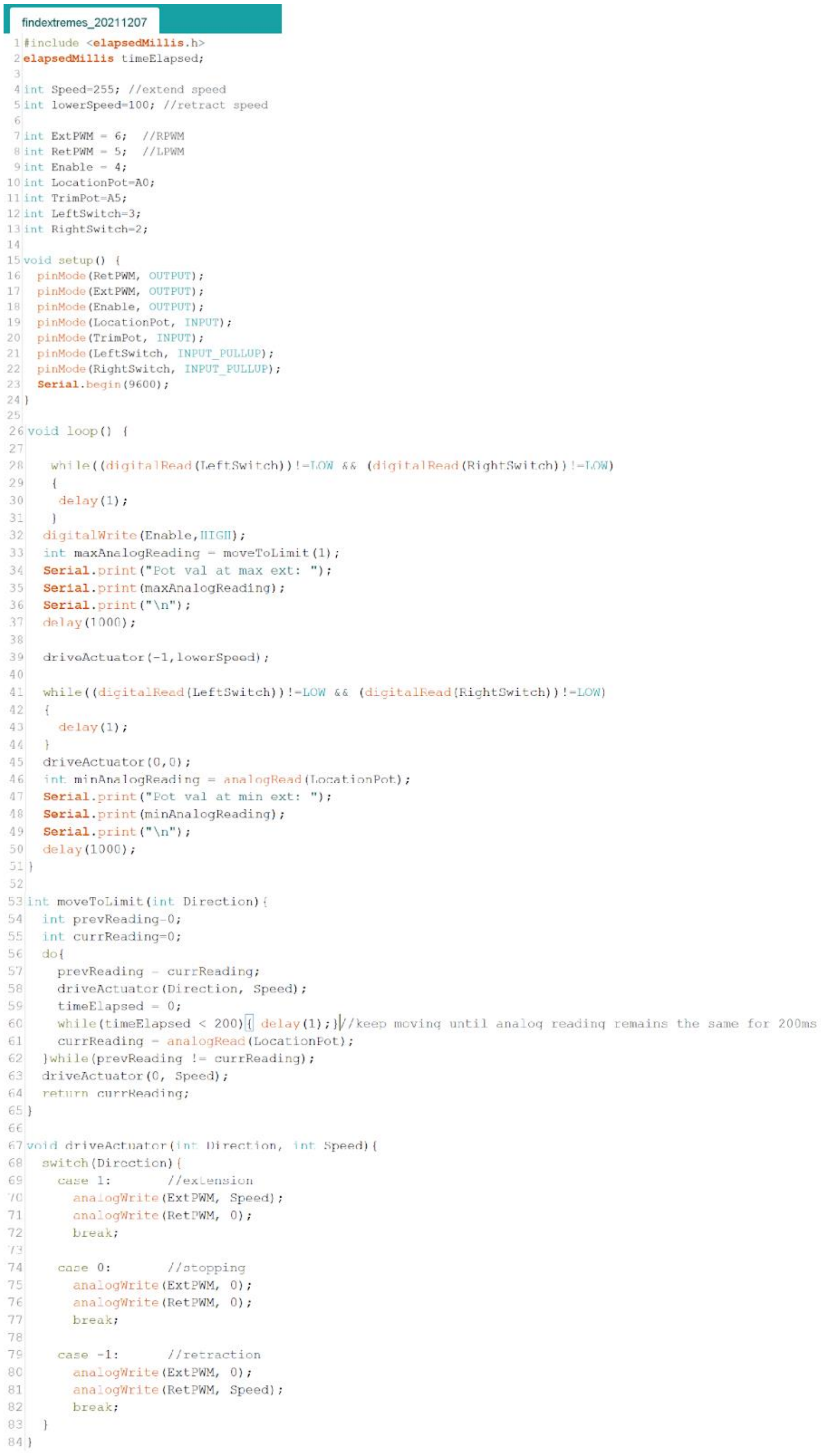


findextremes_20211207

```
#include <elapsedMillis.h>
elapsedMillis timeElapsed;

int Speed=255; //extend speed
int lowerSpeed=100; //retract speed

int ExtPWM = 6;  //RPWM
int RetPWM = 5;  //LPWM
int Enable = 4;
int LocationPot=A0;
int TrimPot=A5;
int LeftSwitch=3;
int RightSwitch=2;

void setup() {
  pinMode(RetPWM, OUTPUT);
  pinMode(ExtPWM, OUTPUT);
  pinMode(Enable, OUTPUT);
  pinMode(LocationPot, INPUT);
  pinMode(TrimPot, INPUT);
  pinMode(LeftSwitch, INPUT_PULLUP);
  pinMode(RightSwitch, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {

    while((digitalRead(LeftSwitch))!=LOW && (digitalRead(RightSwitch))!=LOW)
    {
     delay(1);
    }
  digitalWrite(Enable,HIGH);
  int maxAnalogReading = moveToLimit(1);
  Serial.print("Pot val at max ext: ");
  Serial.print(maxAnalogReading);
  Serial.print("\n");
  delay(1000);

  driveActuator(-1,lowerSpeed);

  while((digitalRead(LeftSwitch))!=LOW && (digitalRead(RightSwitch))!=LOW)
  {
    delay(1);
  }
  driveActuator(0,0);
  int minAnalogReading = analogRead(LocationPot);
  Serial.print("Pot val at min ext: ");
  Serial.print(minAnalogReading);
  Serial.print("\n");
  delay(1000);
}

int moveToLimit(int Direction){
  int prevReading=0;
  int currReading=0;
  do{
    prevReading = currReading;
    driveActuator(Direction, Speed);
    timeElapsed = 0;
    while(timeElapsed < 200){ delay(1);}//keep moving until analog reading remains the same for 200ms
    currReading = analogRead(LocationPot);
  }while(prevReading != currReading);
  driveActuator(0, Speed);
  return currReading;
}

void driveActuator(int Direction, int Speed){
  switch(Direction){
    case 1:       //extension
      analogWrite(ExtPWM, Speed);
      analogWrite(RetPWM, 0);
      break;

    case 0:       //stopping
      analogWrite(ExtPWM, 0);
      analogWrite(RetPWM, 0);
      break;

    case -1:      //retraction
      analogWrite(ExtPWM, 0);
      analogWrite(RetPWM, Speed);
      break;
  }
}
```

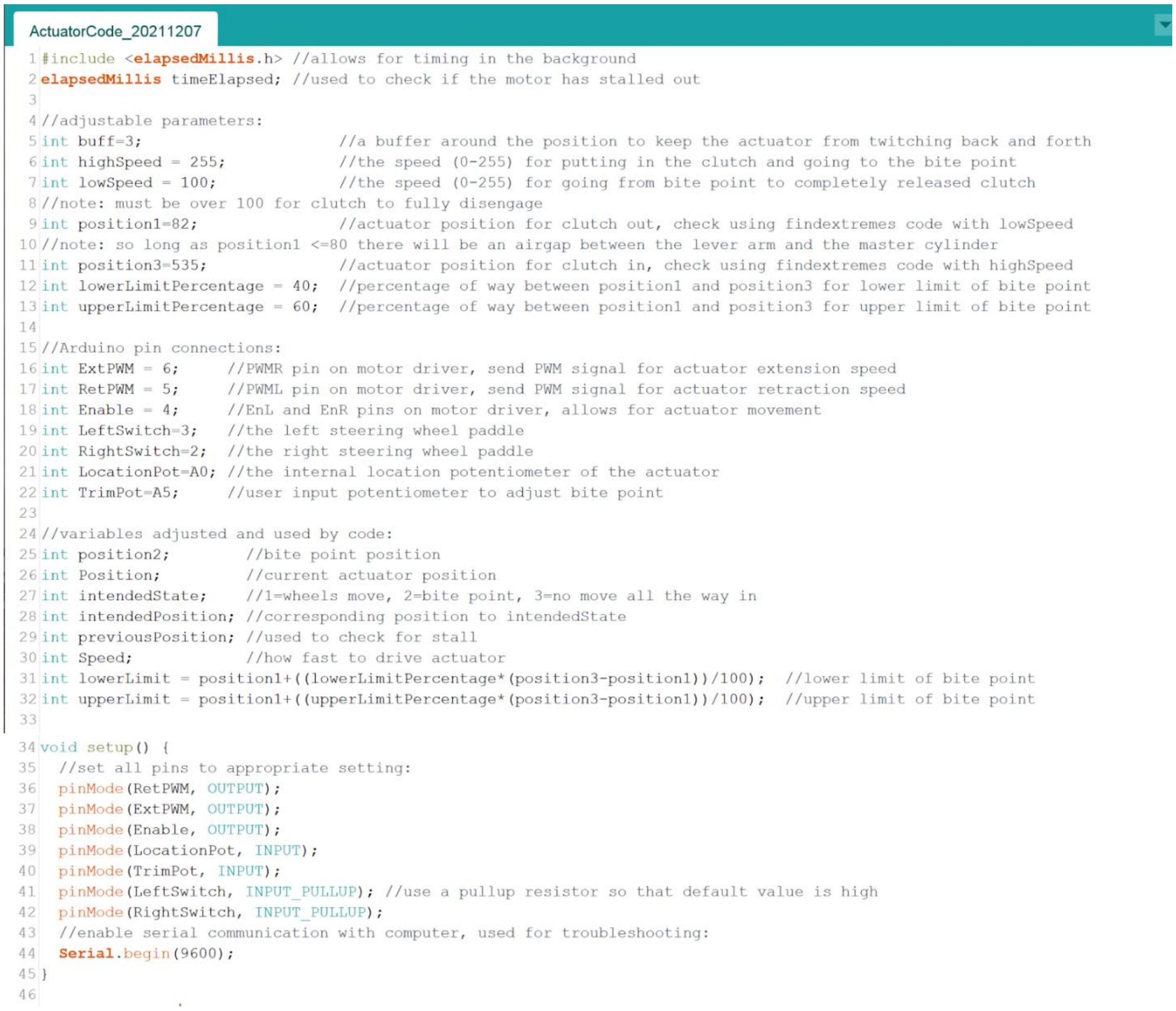
Continued on next page

Appendix E Continued


findextremes_20211207

```
#include <elapsedMillis.h>
elapsedMillis timeElapsed;

int Speed=255; //extend speed
int lowerSpeed=100; //retract speed

int ExtPWM = 6;  //RPWM
int RetPWM = 5;  //LPWM
int Enable = 4;
int LocationPot=A0;
int TrimPot=A5;
int LeftSwitch=3;
int RightSwitch=2;

void setup() {
  pinMode(RetPWM, OUTPUT);
  pinMode(ExtPWM, OUTPUT);
  pinMode(Enable, OUTPUT);
  pinMode(LocationPot, INPUT);
  pinMode(TrimPot, INPUT);
  pinMode(LeftSwitch, INPUT_PULLUP);
  pinMode(RightSwitch, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {

   while((digitalRead(LeftSwitch))!=LOW && (digitalRead(RightSwitch))!=LOW)
   {
    delay(1);
   }
  digitalWrite(Enable,HIGH);
  int maxAnalogReading = moveToLimit(1);
  Serial.print("Pot val at max ext: ");
  Serial.print(maxAnalogReading);
  Serial.print("\n");
  delay(1000);

  driveActuator(-1,lowerSpeed);

  while((digitalRead(LeftSwitch))!=LOW && (digitalRead(RightSwitch))!=LOW)
  {
    delay(1);
  }
  driveActuator(0,0);
  int minAnalogReading = analogRead(LocationPot);
  Serial.print("Pot val at min ext: ");
  Serial.print(minAnalogReading);
  Serial.print("\n");
  delay(1000);
}

int moveToLimit(int Direction){
  int prevReading=0;
  int currReading=0;
  do{
    prevReading = currReading;
    driveActuator(Direction, Speed);
    timeElapsed = 0;
    while(timeElapsed < 200){ delay(1);}//keep moving until analog reading remains the same for 200ms
    currReading = analogRead(LocationPot);
  }while(prevReading != currReading);
  driveActuator(0, Speed);
  return currReading;
}

void driveActuator(int Direction, int Speed){
  switch(Direction){
    case 1:       //extension
      analogWrite(ExtPWM, Speed);
      analogWrite(RetPWM, 0);
      break;

    case 0:       //stopping
      analogWrite(ExtPWM, 0);
      analogWrite(RetPWM, 0);
      break;

    case -1:      //retraction
      analogWrite(ExtPWM, 0);
      analogWrite(RetPWM, Speed);
      break;
  }
}
```

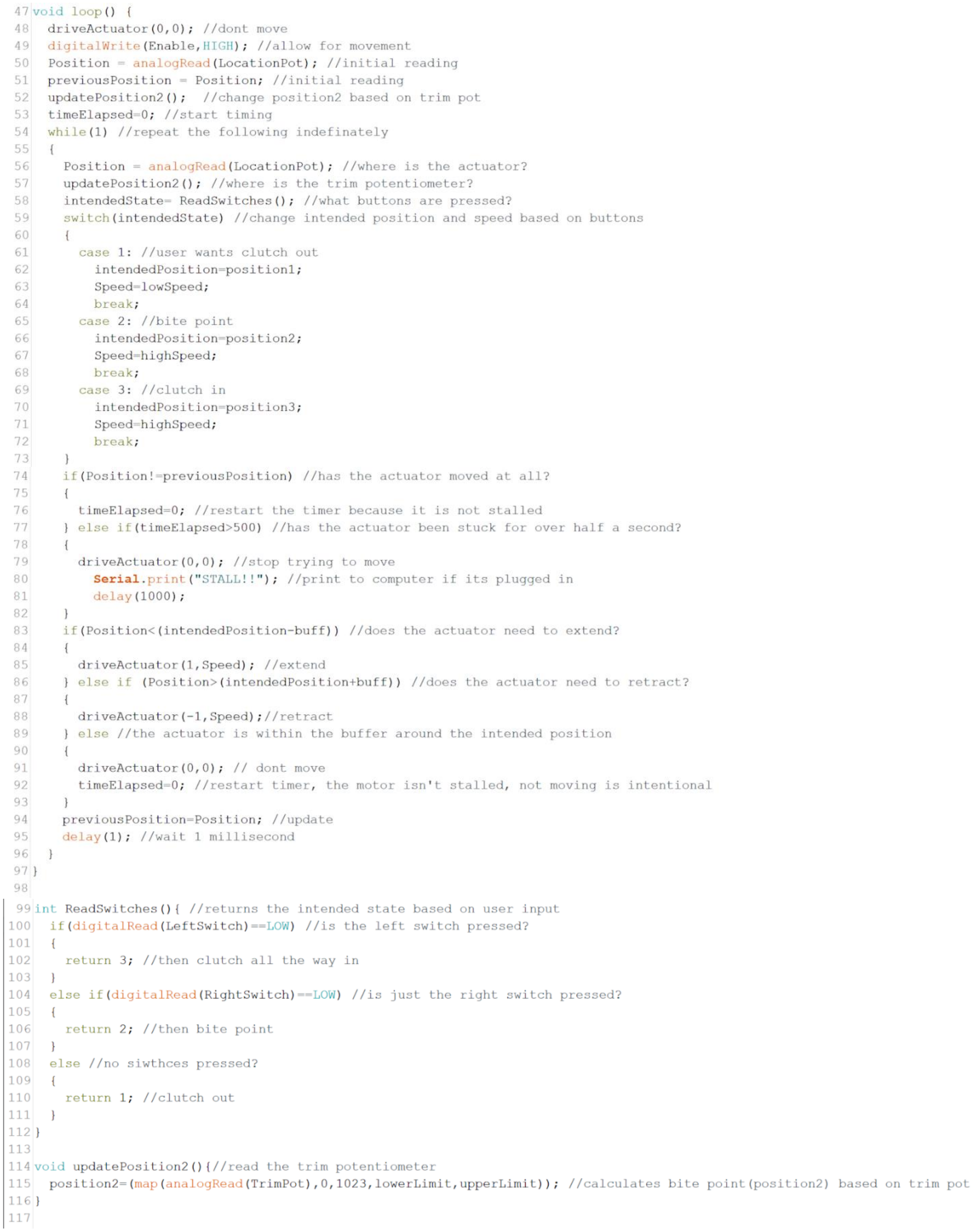
Appendix F: ActuatorCode, the live running code for the system


ActuatorCode_20211207

```
#include <elapsedMillis.h> //allows for timing in the background
elapsedMillis timeElapsed; //used to check if the motor has stalled out

//adjustable parameters:
int buff=3;                     //a buffer around the position to keep the actuator from twitching back and forth
int highSpeed = 255;            //the speed (0-255) for putting in the clutch and going to the bite point
int lowSpeed = 100;             //the speed (0-255) for going from bite point to completely released clutch
//note: must be over 100 for clutch to fully disengage
int position1=82;               //actuator position for clutch out, check using findextremes code with lowSpeed
//note: so long as position1 <=80 there will be an airgap between the lever arm and the master cylinder
int position3=535;              //actuator position for clutch in, check using findextremes code with highSpeed
int lowerLimitPercentage = 40;  //percentage of way between position1 and position3 for lower limit of bite point
int upperLimitPercentage = 60;  //percentage of way between position1 and position3 for upper limit of bite point

//Arduino pin connections:
int ExtPWM = 6;     //PWMR pin on motor driver, send PWM signal for actuator extension speed
int RetPWM = 5;     //PWML pin on motor driver, send PWM signal for actuator retraction speed
int Enable = 4;     //EnL and EnR pins on motor driver, allows for actuator movement
int LeftSwitch=3;   //the left steering wheel paddle
int RightSwitch=2;  //the right steering wheel paddle
int LocationPot=A0; //the internal location potentiometer of the actuator
int TrimPot=A5;     //user input potentiometer to adjust bite point

//variables adjusted and used by code:
int position2;        //bite point position
int Position;         //current actuator position
int intendedState;    //1=wheels move, 2=bite point, 3=no move all the way in
int intendedPosition; //corresponding position to intendedState
int previousPosition; //used to check for stall
int Speed;            //how fast to drive actuator
int lowerLimit = position1+((lowerLimitPercentage*(position3-position1))/100);  //lower limit of bite point
int upperLimit = position1+((upperLimitPercentage*(position3-position1))/100);  //upper limit of bite point

void setup() {
  //set all pins to appropriate setting:
  pinMode(RetPWM, OUTPUT);
  pinMode(ExtPWM, OUTPUT);
  pinMode(Enable, OUTPUT);
  pinMode(LocationPot, INPUT);
  pinMode(TrimPot, INPUT);
  pinMode(LeftSwitch, INPUT_PULLUP); //use a pullup resistor so that default value is high
  pinMode(RightSwitch, INPUT_PULLUP);
  //enable serial communication with computer, used for troubleshooting:
  Serial.begin(9600);
}

```

Continued on next page

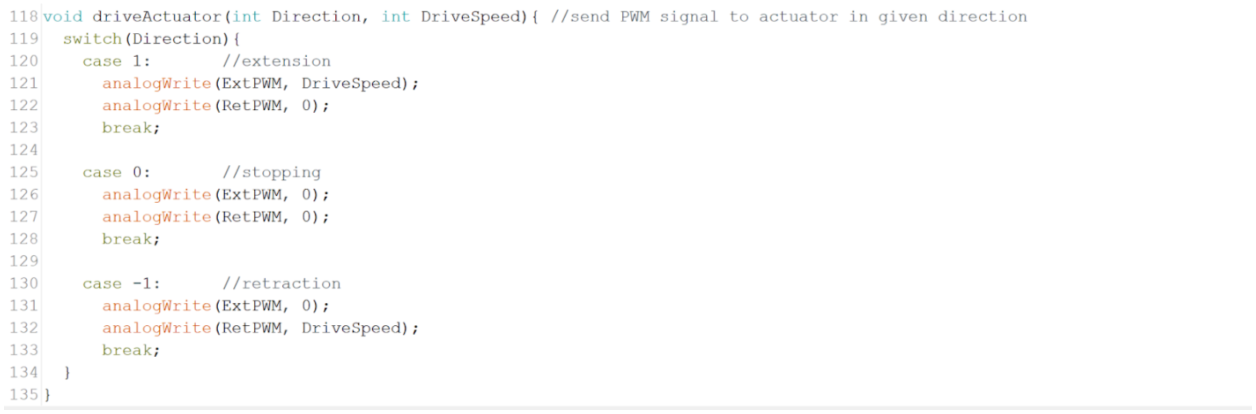
Appendix F Continued



```
void loop() {
  driveActuator(0,0); //dont move
  digitalWrite(Enable,HIGH); //allow for movement
  Position = analogRead(LocationPot); //initial reading
  previousPosition = Position; //initial reading
  updatePosition2();  //change position2 based on trim pot
  timeElapsed=0; //start timing
  while(1) //repeat the following indefinately
  {
    Position = analogRead(LocationPot); //where is the actuator?
    updatePosition2(); //where is the trim potentiometer?
    intendedState= ReadSwitches(); //what buttons are pressed?
    switch(intendedState) //change intended position and speed based on buttons
    {
      case 1: //user wants clutch out
        intendedPosition=position1;
        Speed=lowSpeed;
        break;
      case 2: //bite point
        intendedPosition=position2;
        Speed=highSpeed;
        break;
      case 3: //clutch in
        intendedPosition=position3;
        Speed=highSpeed;
        break;
    }
    if(Position!=previousPosition) //has the actuator moved at all?
    {
      timeElapsed=0; //restart the timer because it is not stalled
    } else if(timeElapsed>500) //has the actuator been stuck for over half a second?
    {
      driveActuator(0,0); //stop trying to move
        Serial.print("STALL!!"); //print to computer if its plugged in
        delay(1000);
    }
    if(Position<(intendedPosition-buff)) //does the actuator need to extend?
    {
      driveActuator(1,Speed); //extend
    } else if (Position>(intendedPosition+buff)) //does the actuator need to retract?
    {
      driveActuator(-1,Speed);//retract
    } else //the actuator is within the buffer around the intended position
    {
      driveActuator(0,0); // dont move
      timeElapsed=0; //restart timer, the motor isn't stalled, not moving is intentional
    }
    previousPosition=Position; //update
    delay(1); //wait 1 millisecond
  }
}

int ReadSwitches(){ //returns the intended state based on user input
  if(digitalRead(LeftSwitch)==LOW) //is the left switch pressed?
  {
    return 3; //then clutch all the way in
  }
  else if(digitalRead(RightSwitch)==LOW) //is just the right switch pressed?
  {
    return 2; //then bite point
  }
  else //no siwthces pressed?
  {
    return 1; //clutch out
  }
}

void updatePosition2(){//read the trim potentiometer
  position2=(map(analogRead(TrimPot),0,1023,lowerLimit,upperLimit)); //calculates bite point(position2) based on trim pot
}

```

Continued on next page

Appendix F Continued



```
void driveActuator(int Direction, int DriveSpeed){ //send PWM signal to actuator in given direction
  switch(Direction){
    case 1:       //extension
      analogWrite(ExtPWM, DriveSpeed);
      analogWrite(RetPWM, 0);
      break;

    case 0:       //stopping
      analogWrite(ExtPWM, 0);
      analogWrite(RetPWM, 0);
      break;

    case -1:      //retraction
      analogWrite(ExtPWM, 0);
      analogWrite(RetPWM, DriveSpeed);
      break;
  }
}
```