# ME 102B Final Report - ADAMM

**By: Ethan Lee, Elianna Peng, Max Ries, Rishi Wadgaonkar**

**Opportunity:**

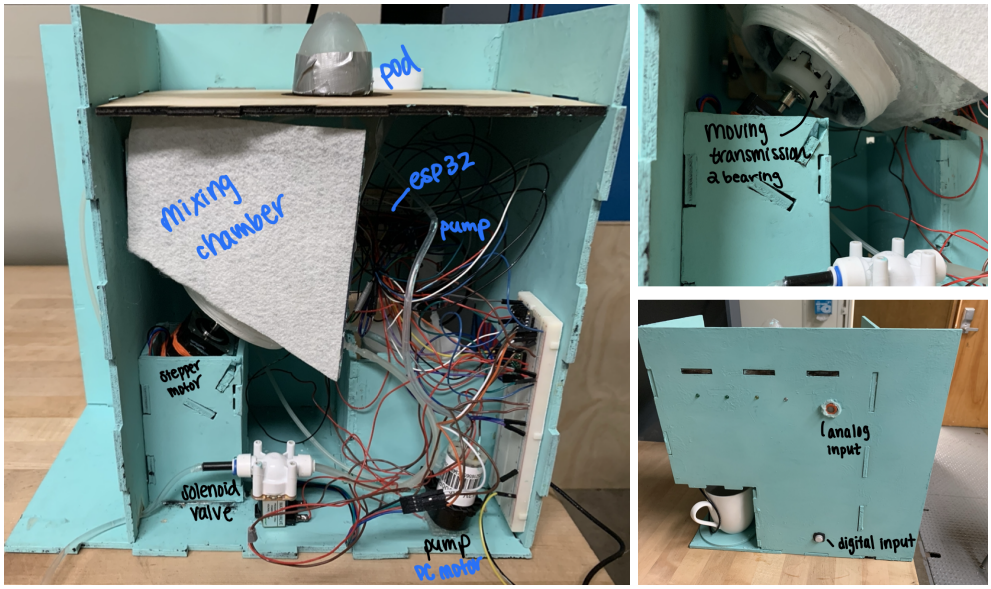
With the automatic drink and mixing machine (ADAMM), we pursued the opportunity to create a more convenient, consistent and safe way to make and consume beverages. There is no longer the need to worry about learning how to make your own drinks, ensure that they have a standardized alcohol content or worry about consumers over-serving themselves. We see this product being commercially viable in not only household kitchens/bars, but also nightlife settings to aid often overloaded bartenders.

**High Level Strategy:**

On a high level, the strategy behind the ADAMM is quite simple. The user first blows into a breathalyzer that reads their blood alcohol content and if the reading is below a certain threshold for BAC they are able to insert the pod, close the lid and begin the machine with the press of a button. Alcohol is then pumped out a reservoir and through the inserted pod which contains mixers/syrups for the drink. These two liquids then funnel down into a mixing chamber where a blender is running and is finally dispensed/pumped into a cup for the user to enjoy. Afterwards, the machine runs a cleaning cycle and pumps water through the chamber then dispenses this dirty water into a disposable alternate chamber, access to which is controlled with a solenoid valve. Then, the machine restarts. At each stage a LED lights up to display which stage the machine is at. 1st to signal the button is ready to be pushed, 2nd and 3rd to represent blending and pumping, 4th to represent the cleaning cycle.

Initially, we hoped to be able to create a whole drink within just a few minutes. However, we ran into efficiency limitations with the pumps that we chose which could only pump a certain amount of liquid at once. We wanted a flow rate of 0.5 ounces/second to dispense the drink in 1 minute, but the pump was only capable of 0.05 ounces/second. Additionally, after realizing the solenoid valves needed a minimum pressure to run effectively we wanted to switch it out with another pump, however due to budget constraints, we were forced to use the valve with a slow flow rate.

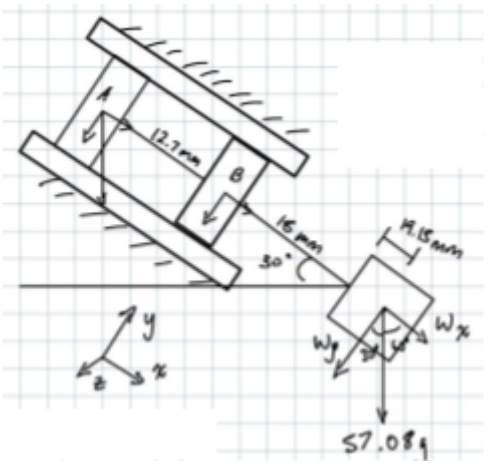
**Photos:**



**Function-Critical Decisions:**

One function-critical decision we made was the use of 2 DC pumps instead of 1 DC pump and 2 solenoid valves. We realized that the valve needed more pressure to run and switched to pumps to increase the flow rate. Additionally, we decided to use a stepper motor to run the blender for more power instead of a DC motor.

Motor Force Calculations




$W_y = 57.08 \sin(30)$
$W_x = 57.08 \cos(30)$
$\Sigma M_B = 0 = [(-W_y * g)\, j * (37.15\ mm)i] + [(-A_y * g)\, j * (-12.7\ mm)i]$
$\therefore 57.08 \sin(30)(37.15) = -12.17A_y$
$\therefore A_y = -87.121\ g\ (j)$
$\therefore A_y = -0.85N\ (j)$ ***radial load on bearing A ***

$\Sigma F_x = W_x = 49.4327g = (0.485N)\ i$ (axial load)
$\Sigma F_z = 0$
$\Sigma F_y = 0 = (-W_y\ g)j + (A_y g)j + B_y g\ (-j)$
$\therefore -57.08\sin(30)\ g - (0.85N) - B_y g$
$\therefore B_y = 57.08 \sin(30)\ g + 0.85N$
$\therefore B_y = 1.13N\ (j)$ ***radial load on bearing B ***

From Motor Specification Sheet:

| | |
|---|---|
| Max axial load = 60N | $\lvert A_y \rvert = 0.85N < 0.66kN$ & 220N |
| Max radial load = 220N | $\lvert B_y \rvert = 1.13N < 0.66kN$ &220N |

From Bearing Specification Sheet:

| | |
|---|---|
| Max radial load = 0.66kN | $\lvert F_x \rvert = 0.485N < 60N$ |

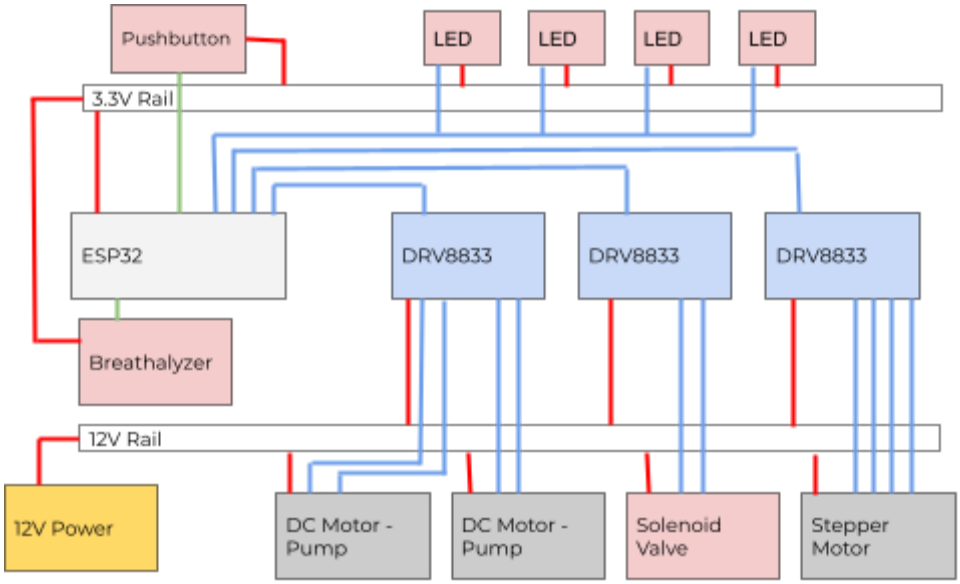
$\therefore$ Axial and radial loads can be handled within specification!

Pump Flow Rate

- $10\ \frac{ml}{min} = 0.056\ \frac{oz}{s} \rightarrow 17.714\ \frac{s}{oz}$
- $Time\ to\ fill\ chamber\ with\ 6oz\ per\ pod = (6oz) * (17.744\ \frac{s}{oz}) = 106\ sec$
- $Time\ to\ fill\ chamber\ with\ 6oz\ per\ pod = (4oz) * (17.744\ \frac{s}{oz}) = 71\ sec$
- $Time\ to\ dispense\ 10\ oz\ to\ user = (10\ oz) * (17.744\ \frac{s}{oz}) = 178\ sec$
- $Total\ time\ from\ start\ to\ finish = 178\ sec + 106\ sec = 284\ sec = 4\ min\ \&\ 44\ sec$

**Circuit Diagram:**

The key difference between the previous circuit design and our final design was the replacement of one solenoid with a pump due to minimum pressure issues not allowing fluid through the solenoid. Everything else remains the same.

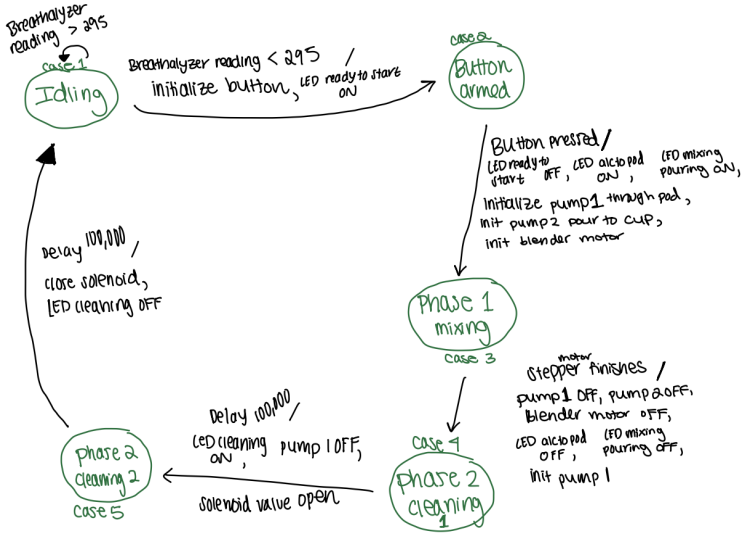



**Legend:**
DRV8833 Full H Bridge.
Red Wires: Power
Green Wires: Input Signal
Blue Wires: Output Signal
Everything grounded, but wires not shown to reduce clutter.

**ESP32 Pins Used:**
GND: Connected to ground rail
3.3V: Supplies 3.3V rail
21, 17, 16, 19: LED pins
26: Button pin
33: Breathalyzer pin
4: Solenoid output for cleaning cycle
23, 22, 14, 32: Stepper motor pins

**ESP32 Protection:**
12V power rail is not connected to the ESP32. H Bridges are only connected to 12V power rail.

**State Transition Diagram:**




**Reflection**

Looking back on our project, we see how splitting up into hardware and software teams was useful in completing the project, additionally having a makerpass was paramount.

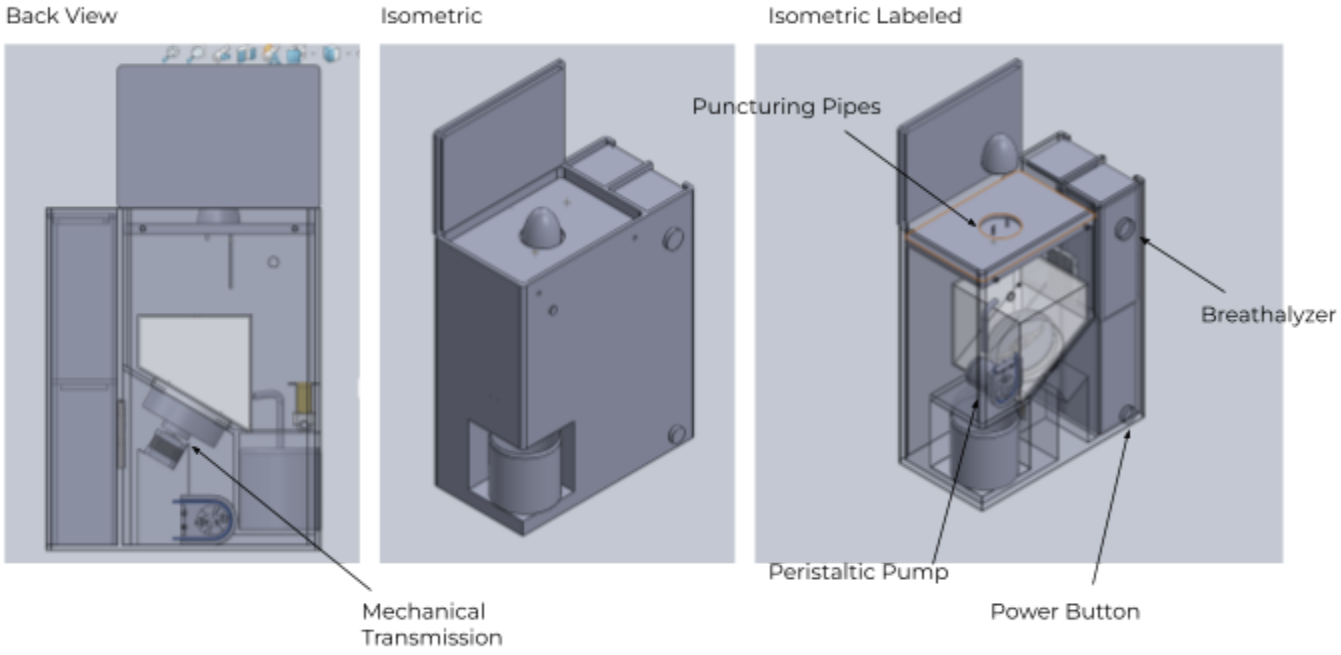
Some things to improve upon was ensuring everything was making everything waterproof. We did not anticipate the amount of leaking that occurred and had some minor leaks. Also, with the electrical components, it is important to ensure all the wires are firmly inserted and secured into the breadboards. When drilling acrylic or resin keep in mind that they are highly brittle materials, so drill slowly. Finally, make sure to have set screws for motor shafts and couplings for smoother and reliable rotation.

APPENDICES

BOM

| Item | Quantity | Supplier | Part Number | Price Total |
|---|---|---|---|---|
| Silicone Tubing for pump | 2 | AdaFruit | | $ 7.00 |
| Nema 17 Stepper Motor | 1 | McMaster | | $ - |
| Plywood 12x24X1/4inch | 6 | Jacobs | | $ 8.35 |
| Peristaltic Pump | 2 | AdaFruit | | $ 23.96 |
| Metal straw | 1 | | | $ - |
| Solenoid Valves | 2 | AdaFruit | | $ 13.90 |
| H bridge for stepper | 1 | AdaFruit | | $ 5.95 |
| H bridge for solenoids/pu | 2 | AdaFruit | | $ 17.90 |
| Pushbutton | 2 | AdaFruit | | $ 3.90 |
| Mixer Housing | 1 | NutriBullet | | $ 19.99 |
| Wood Screws, 1/8inch | 100 | McMaster | | $ 18.05 |
| Breathalyzer | 2 | Amazon | | $ 7.49 |
| Form Labs Material | 300ml x 2 | Jacobs | Formlabs - Engineeri | $ 126.00 |
| LEDs | 100 | Amazon | | $ 4.99 |
| Spray Paint | 3 | Home Depot | | $ 6.00 |
| Adaptors/Spare Tubes | | Jacobs | | $ - |
| Hinges | | | | |
| | | | | |
| | | | | |
| | | | Total | $ 252.49 |

## CAD




## Code Screenshots:

```
//Includes the arduinoFFT library as well as the Stepper library.
#include <Arduino.h>
#include <Stepper.h>

//Define pins LEDs are connected to
#define ReadyToStart 21
#define AlcToPod 17
#define MixingPouring 16
#define Cleaning 19

#define BTN 26

#define BreathPin 33

#define Pump1_1 15

#define SolenoidALC 13
#define SolenoidWATER 4

//Number of steps for stepper motor
#define STEPS 200

int DRINK_MAKING_STEP = 1;

Stepper stepper(STEPS, 23, 22, 14, 32);

int val = 0; // value of breathlyzer reading

void setup() {
  Serial.begin(115200); //Baud rate for the Serial Monitor

  pinMode(Pump1_1, OUTPUT);

  pinMode(SolenoidALC, OUTPUT);
  pinMode(SolenoidWATER, OUTPUT);
```

```
  pinMode(BTN, INPUT_PULLUP);  // configures the specified pin to behave either as an input


 //whatever speed for max RPM while staying within torque limits
 stepper.setSpeed(250);

  //LEDs are outputs according to step in process
  pinMode(ReadyToStart, OUTPUT);
  pinMode(AlcToPod, OUTPUT);
  pinMode(MixingPouring,OUTPUT);
  pinMode(Cleaning, OUTPUT);
}

void loop() {
switch(DRINK_MAKING_STEP){
  //State awaits event of breathalyzer reading, provides service of turning on LED to signi
  case 1:
    {
     val = readAlcohol(); //calls helper function to read alcohol level (event checker)
     Serial.println(val);
     if (val < 1000 && val > 400) {

        led_READYTOSTART_on(); //turns on LED signifying ready to start (service)
        Serial.println("YOU ARE SOBER");
        Serial.println("READY TO START");
        delay(1000);
        DRINK_MAKING_STEP = 2;
    }
    break;
    }
  //State awaits event of button being pressed, provides service of turning off LED 1 and b
  case 2:
     {
     int buttonVal = digitalRead(BTN);
     if (buttonVal == HIGH) { //(event checker)
        Serial.println("Button pressed");
        led_READYTOSTART_off(); //(service)
        DRINK_MAKING_STEP = 3;
     }
     break;
     delay(500);
     }

  //State contains the service of alcohol mixing with alcohol pump turned on, motor running
  //opening to users drink
  case 3:
    {
    led_ALCTOPOD_on(); //(service)
    led_MIXINGPOURING_on(); //(service)

    pumpOn(); //(service)
    Serial.println("Alcohol pumping to pod");
    openSolenoidALC(); //(service)
    Serial.println("Mixing and pouring out into cup");

    //CHANGE TO MATCH TIME FOR PUMP TO RUN (75)
    // change 400
    stepper.step(STEPS*10); //(event checker AND service) note this value determines how lo
                            //halts the code process until motor finishes steps, so it func
                            //pump and solenoid valve run concurrently.

    //The below services occur after the stepper motor finishes tunning through specified n
    pumpOff(); //(service)
    Serial.println("Alcohol pump is off");
    closeSolenoidALC(); //(service)
    Serial.println("Alcohol solenoid closed");

    led_ALCTOPOD_off(); //(service)
    led_MIXINGPOURING_off(); //(service)
    Serial.println("Drink has been dispensed");
    DRINK_MAKING_STEP = 4;
```

```
    break;
    }

  //State awaits event of solenoid valve closing, alcohol pump turning off, and motor to fi
  //State contains service of water pump turning on, motor running to clean tank, and solen
  //dirty water.

case 4:
  {
    led_CLEANING_on(); //(service)
    pumpOn(); //(service)
    Serial.println("Clean water pumping to pod");
    delay(10000);
    //change 100,000

    pumpOff(); //(service)
    DRINK_MAKING_STEP = 5;
    break;
    }

 case 5:
  {
    Serial.println("Cleaning started");
    delay(100);

    openSolenoidWATER(); //(service)
    Serial.println("Dirty water pouring out to dirty water chamber");
    delay(10000);
    //change 100,000

    closeSolenoidWATER(); //(service)
    Serial.println("Dirty water solenoid closed");

    delay(100);

    led_CLEANING_off(); //(service)
    DRINK_MAKING_STEP = 1;
    break;
    }
}
}


//EVENT CHECKERS (apart from stepper motor function which works as a timer event checker)

int readAlcohol() {
  int val = 0;
  int val1;
  int val2;
  int val3;

  val1 = analogRead(BreathPin);
  delay(10);
  val2 = analogRead(BreathPin);
  delay(10);
  val3 = analogRead(BreathPin);

  val = (val1+val2+val3)/3;
  return val;
 }


//SERVICES

void led_READYTOSTART_on() {
  digitalWrite(ReadyToStart, HIGH);
}

void led_READYTOSTART_off() {
  digitalWrite(ReadyToStart, LOW);
}
```

```
}

void led_ALCTOPOD_on() {
  digitalWrite(AlcToPod, HIGH);
}

void led_ALCTOPOD_off() {
  digitalWrite(AlcToPod, LOW);
}

void led_MIXINGPOURING_on() {
  digitalWrite(MixingPouring, HIGH);
}

void led_MIXINGPOURING_off() {
  digitalWrite(MixingPouring, LOW);
}

void led_CLEANING_on() {
  digitalWrite(Cleaning, HIGH);
}

void led_CLEANING_off() {
  digitalWrite(Cleaning, LOW);
}

void pumpOn() {
  digitalWrite(Pump1_1, HIGH);
}

void pumpOff() {
  digitalWrite(Pump1_1, LOW);
}

void openSolenoidALC() {
  digitalWrite(SolenoidALC, HIGH);
}

void closeSolenoidALC() {
  digitalWrite(SolenoidALC, LOW);
}

void openSolenoidWATER() {
  digitalWrite(SolenoidWATER, HIGH);
}

void closeSolenoidWATER() {
  digitalWrite(SolenoidWATER, LOW);
}
```