DEREK REPSCH

MECHATRONICS DESIGN ENGINEER

derekrepsch@gmail.com

206.351.9222 | linkedin.com/in/drepsch

September 18, 2020

Hello Kenworth Engineering!

I am *very* excited to see Kenworth is hiring for Product Development & Interiors! I've been in touch with Kim Shoemake since being acquainted last winter and have been eagerly awaiting hiring activities to resume – this role is everything I had hoped for. Having completed a post-bac BSME at the University of Washington this June (following a BFA from the School of the Art Institute of Chicago), I am eager to apply a broad skillset, strong interpersonal skills, client-facing attitude, and fearsome work ethic. My time at UW provided me with substantial experience both in automotive and human-centered product design - that experience along with the creative sensibility of an industrial designer makes this role sound a perfect fit. Having significant CAD / CAE / FEA chops (including advanced techniques for organic 3D printed structures), a knack for circuits, programming, prototyping, years of DIY-ing, and actual hands-on mechanical work, I have no doubt that I'd be able to hit the ground running as an agile and dynamic member of your team.

A bit on recent work: My final quarters at UW were marked by two major projects and an internship, the culmination of which demonstrated technical proficiency, strong leadership, clear ambition and a natural ability to learn quickly with little to no guidance. The first project was a Boeing-sponsored / UW Formula Motorsports Capstone where I lead a team in constructing a method for manufacturing high-performance, 3D printed titanium parts in-house at UW. The second, a microcontroller based musical instrument which used a stepper motor for sound production, coded and built from scratch (worth noting: this was part of a Human Centered Design & Engineering class that I was accepted in to by petition). Outside the classroom, while interning with Divergent Technologies / Czinger Vehicles (3D printed automotive / automated manufacturing), I completed several structures / mechanical projects including a major redesign of a suspension subsystem, working under minimal supervision with total project ownership, requiring regular contribution to meetings, all in a high-pressure startup environment.

Please see the following pages for details on these projects and some praise from my manager at Divergent. Additional references are available upon request, including my Boeing mentors. Thank you for your time and consideration – feel free to contact me by phone or email with any questions.

Sincerely,

Derek Repsch

Figures from final report showing topology optimization, design, and verification

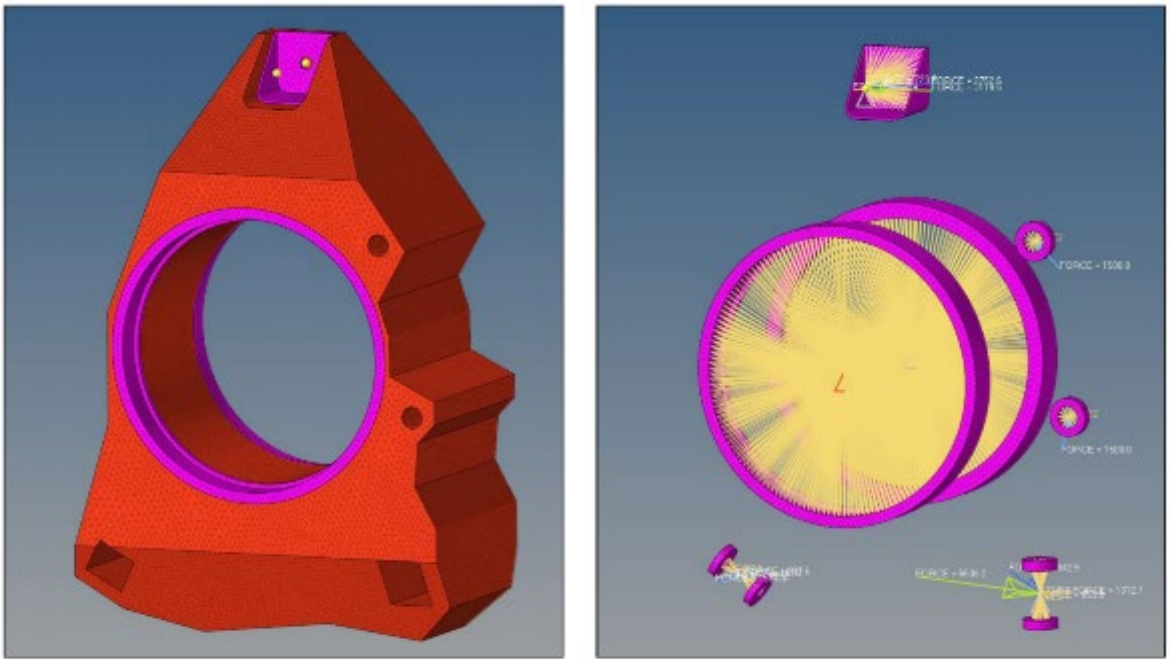


Figure 4.16: Final design volume (red), non-design space (magenta) and RBEs (yellow)

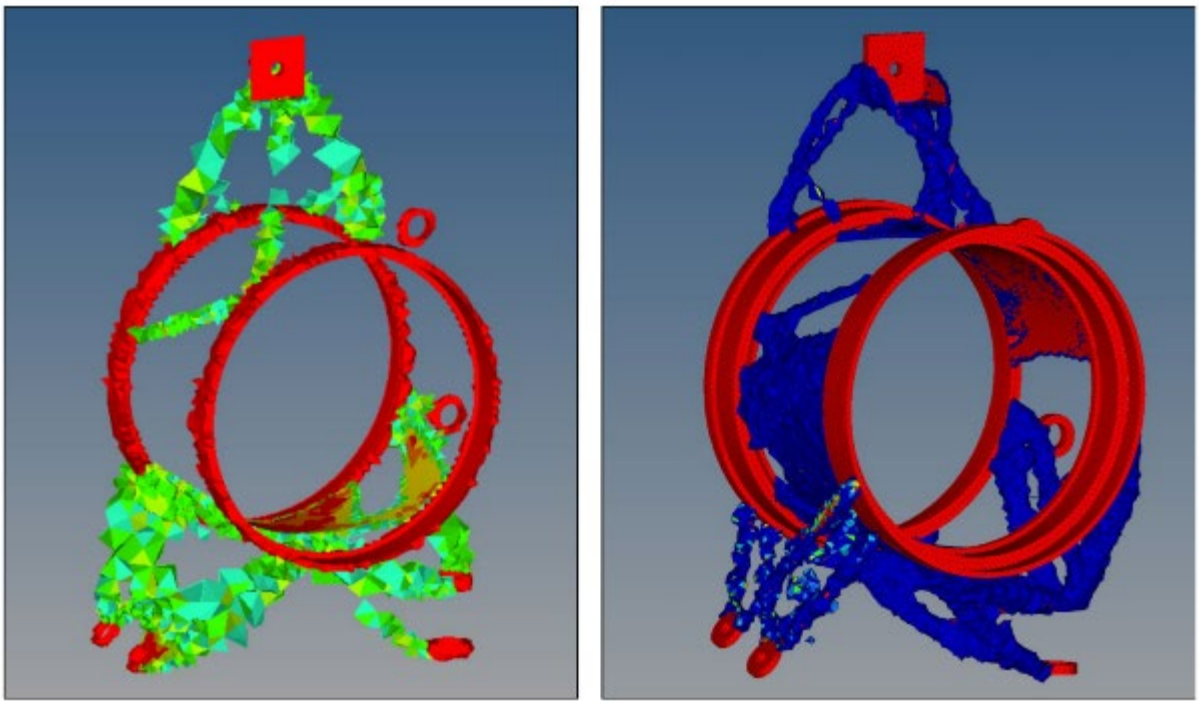


Figure 4.19: Results comparing mesh size of 4mm (left) and 2mm (right)

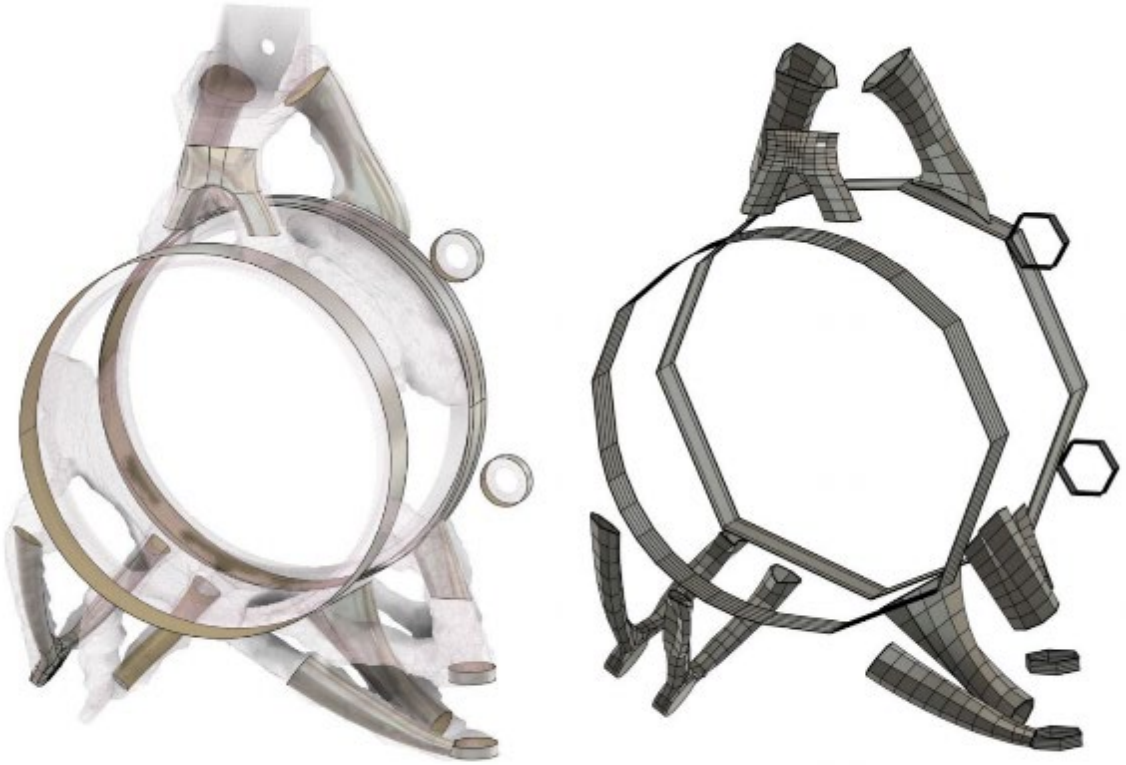


Figure 4.20: *Direct modeling in Fusion 360*

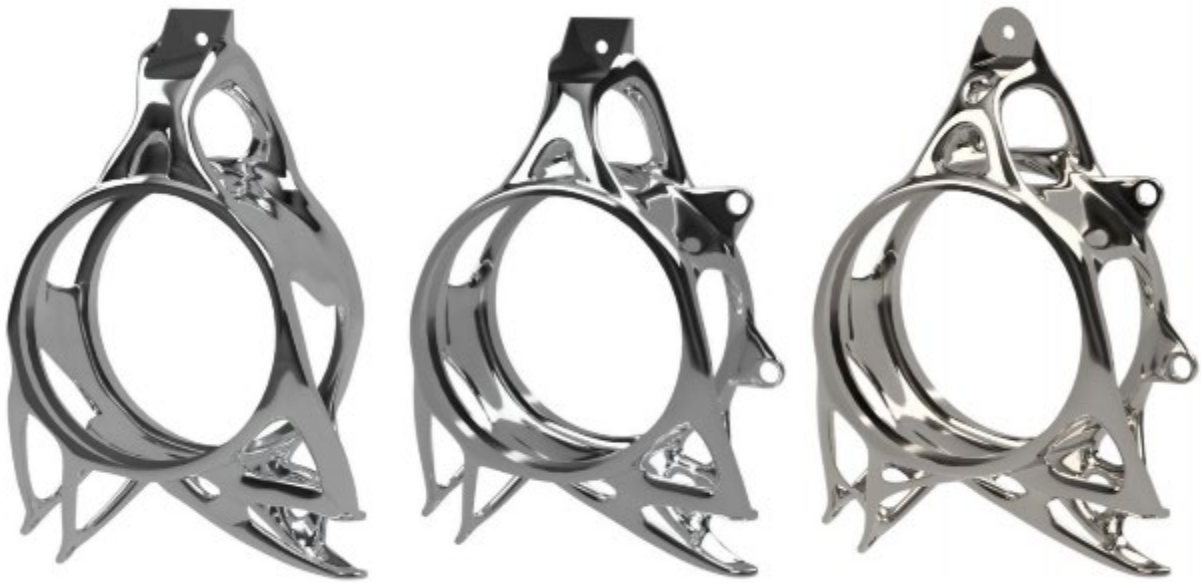


Figure 4.21: *Early, intermediate and final design (left to right)*

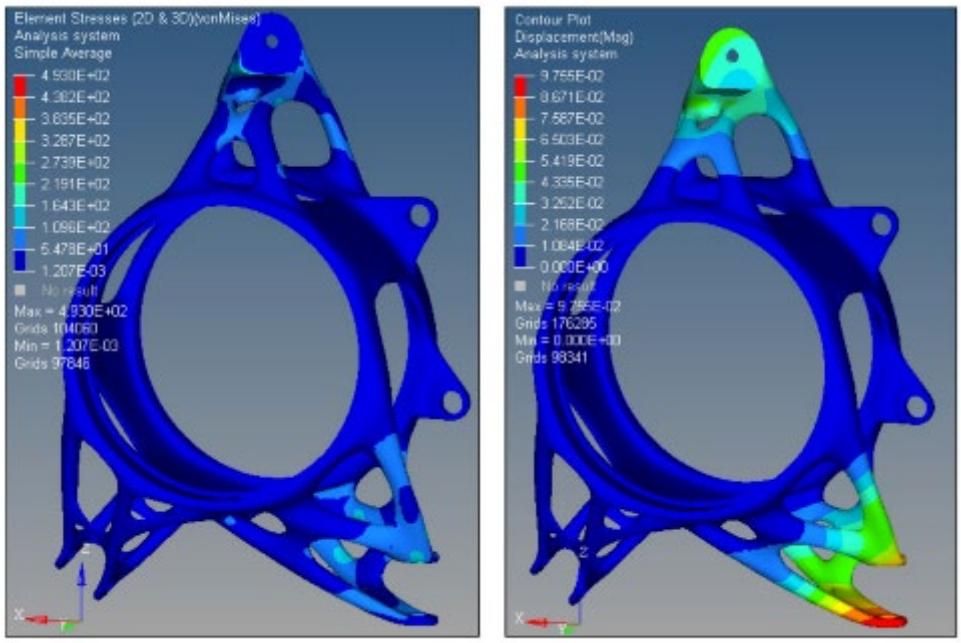


Figure 4.22: *Von Mises Stress in MPa (left) and displacement in mm (right) for Acceleration load case*

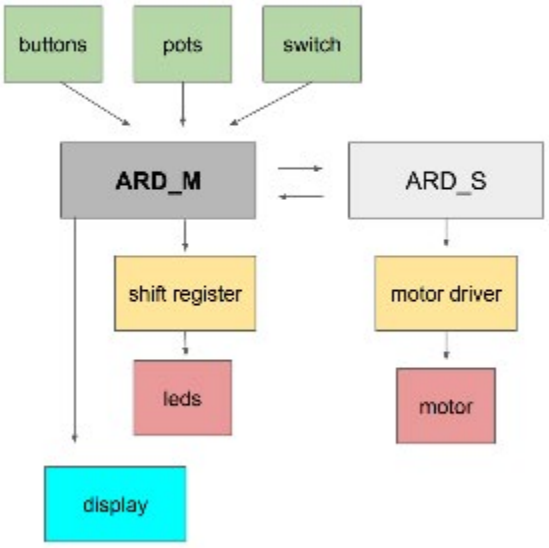
Table 4.5.1: *Comparison of weight and stiffness between T30 and T1000*

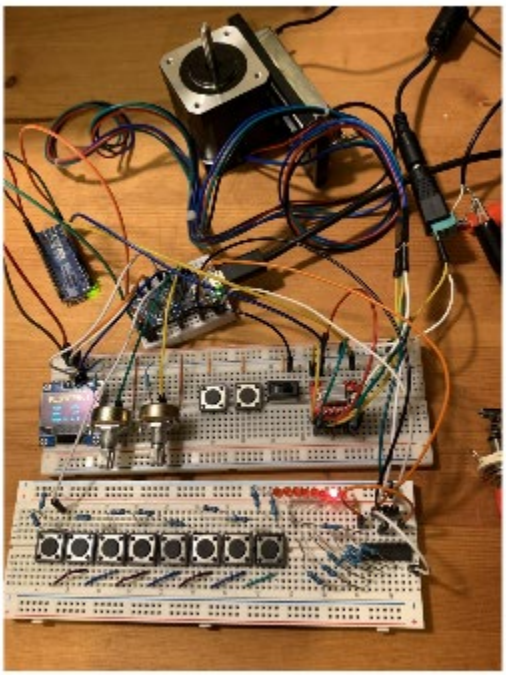
Model	Material	Weight (g)	Stiffness (kN/mm)
T30	Al 7075	643.37	28.312
T1000	Ti 6Al-4V	488.64	94.385
		- 24.05%	**+ 333%**

Figure 4.23: *Renderings of T1000 concept (left) and T30 design (right)*

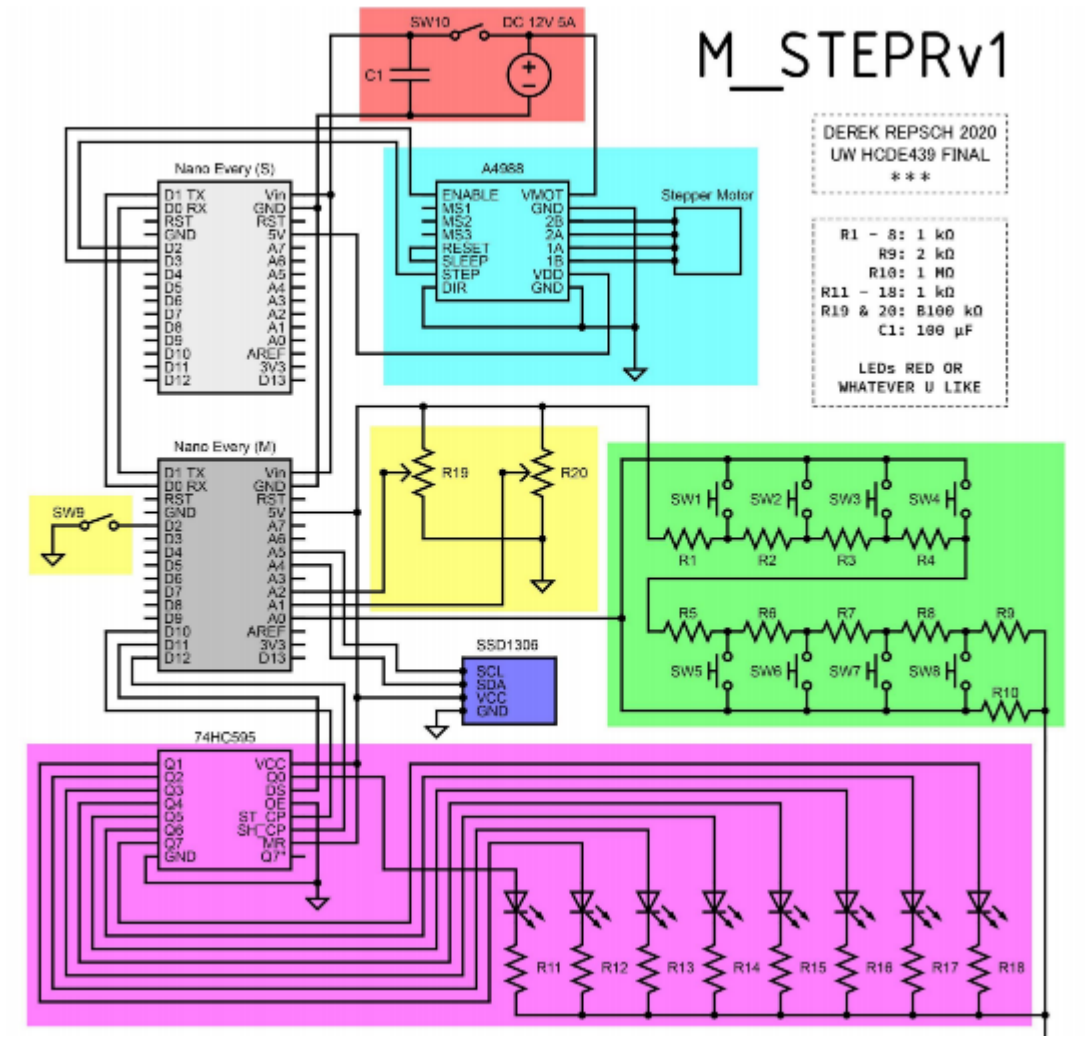
Select slides from final presentation of device. Video of operation at https://youtu.be/ZBlBnlzaSMY

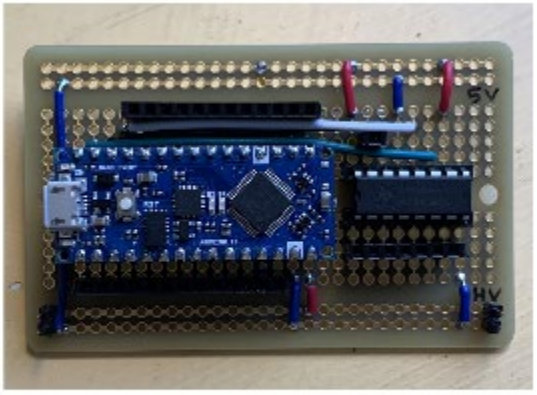
M STEPRv1



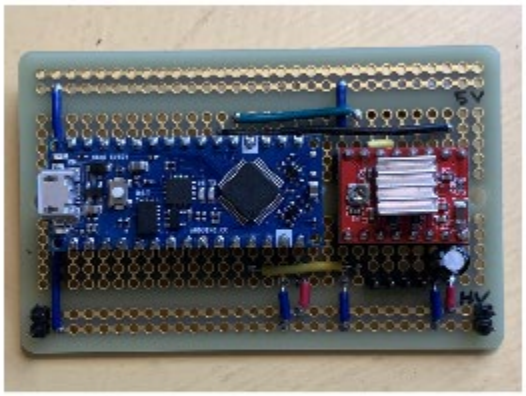


ARCHITECTURE





master



slave

ARD BOARDS

```
#include "notePeriods.h"
#include <Wire.h>
#include <Adafruit_SSD1306.h>
// #include <Adafruit_GFX.h>
Adafruit_SSD1306 display(128,
64, &Wire, -1);
#define OLED_ADDR   0x3C


#define playSwitchPin 2
#define latchPin 10
#define dataPin 11
#define clockPin 12
const int tunePotPin = A0;
const int timePotPin = A1;
const int buttonsPin = A2;

const int numberReadings = 5;
int
timePotReadings[numberReadings];
int timeReadIndex = 0;
int timePotTotal = 0;
int timePotAvg = 0;

int
tunePotReadings[numberReadings];
int tuneReadIndex = 0;
int tunePotTotal = 0;
int tunePotAvg = 0;

byte noteArray[9];
char *nameArray[] = {"C", "C#",
"D", "D#", "E", "F", "F#", "G",
"G#", "A", "A#", "B"};

int periodArray[9];
int durationArray[9];
byte muteArray[9] = {0, 1, 1, 1,
1, 1, 1, 1, 1};

char *stepCharArray[9];
byte stepRegisterArray[9];

int buttonValue;
byte buttonPress = 1;
byte lastButtonPress;

int tunePotVal;
byte tunePotTracked;
int timePotVal;

byte updateMenu = 1;

unsigned long prevMicrosPeriod =
0;
unsigned long prevMillisStep =
0;
int BPM = 120;
unsigned int beatDuration = 500;

byte stepNumber = 1;
byte playState;
byte mlEnable;

byte shiftBit;

void setup() {
  for (byte i = 1; i < 9; i ++)
{
    noteArray[i] = i + 55;
  }
  //Serial.begin(115200);
  Serial1.begin(115200);
  while (!Serial1) {}

  pinMode(playSwitchPin,
INPUT_PULLUP);
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);

  shiftBit = 8;
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin,
MSBFIRST, shiftBit);
  digitalWrite(latchPin, HIGH);


display.begin(SSD1306_SWITCHCAPV
CC, OLED_ADDR);
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 20);
  display.println("\W MODE439");
  display.setTextSize(1);\
  display.setCursor(25, 45);
  display.println("DERBPSZ
2020");
  display.display();
  delay(3000);

  display.clearDisplay();
  display.setCursor(35, 20);
  display.print("Loading.");
  display.display();
  delay(1250);
  display.clearDisplay();
  display.setCursor(35, 20);
  display.print("Loading..");
  display.display();
  delay(1250);
  display.clearDisplay();
  display.setCursor(35, 20);
  display.print("Loading...");
  display.display();
  delay(1250);

  display.clearDisplay();
  delay(1000);
  display.setCursor(10, 20);
  display.println("Ok Alexa,");
  display.display();
  delay(1000);
  display.setCursor(10, 35);
  display.println("turn it up to
11.");
  display.display();
  delay(3000);
  display.clearDisplay();
  display.display();
  delay(750);
}

void loop() {
  playState =
digitalRead(playSwitchPin);
  readPots();
  readButtons();

  stepMenu();


  if (playState == 1) {
    stepSeq();
  }

  if (playState == 0) {
    liveKey();
  }
  // motorSyn(stepPin1);
  //  motorSend();

//////////////FUNCS
void readPots() {
  tunePotTotal = tunePotTotal -
tunePotReadings[tuneReadIndex];
  tunePotReadings[tuneReadIndex]
= analogRead(tunePotPin);
  tunePotTotal = tunePotTotal +
tunePotReadings[tuneReadIndex];
  tuneReadIndex = tuneReadIndex
+ 1;
  if (tuneReadIndex >=
numberReadings) {
    tuneReadIndex = 0;
  }
  tunePotAvg = tunePotTotal /
numberReadings;

  tunePotVal =
constrain(map(tunePotAvg, 0,
1023, 23, 151), 23, 151);
  if (tunePotTracked == 1 &&
tunePotVal !=
noteArray[buttonPress]) {
    noteArray[buttonPress] =
tunePotVal;
  }
  else if (tunePotTracked == 0
&& abs(noteArray[buttonPress] -
tunePotVal) < 1) {
    tunePotTracked = 1;
    noteArray[buttonPress] =
tunePotVal;
  }

  timePotTotal = timePotTotal -
timePotReadings[timeReadIndex];
  timePotReadings[timeReadIndex]
= analogRead(timePotPin);
  timePotTotal = timePotTotal +
timePotReadings[timeReadIndex];
  timeReadIndex = timeReadIndex
+ 1;
  if (timeReadIndex >=
numberReadings) {
    timeReadIndex = 0;
  }
  timePotAvg = timePotTotal /
numberReadings;

  BPM =
constrain(map(timePotAvg, 0,
1020, 10, 330), 10, 330);
  beatDuration = 60000 / BPM /
2;

  // periodArray[buttonPress] =
notePeriods[noteArray[buttonPres
s]];
  stepCharArray[buttonPress] =
nameArray[noteArray[buttonPress]
% 12];
  stepRegisterArray[buttonPress]
= noteArray[buttonPress] / 12 -
1;
}

void readButtons() {
  buttonValue =
analogRead(buttonsPin);
  if (buttonValue > 950) {
    buttonPress = 1;
    tunePotTracked = 0;
  }
  else if (buttonValue > 850) {
    buttonPress = 2;
    tunePotTracked = 0;
  }
  else if (buttonValue > 750) {
    buttonPress = 3;
    tunePotTracked = 0;
  }
  else if (buttonValue > 600) {
    buttonPress = 4;
    tunePotTracked = 0;
  }
  else if (buttonValue > 500) {
    buttonPress = 5;
    tunePotTracked = 0;
  }
  else if (buttonValue > 400) {
    buttonPress = 6;
    tunePotTracked = 0;
  }
  else if (buttonValue > 300) {
    buttonPress = 7;
    tunePotTracked = 0;
  }
  else if (buttonValue > 200) {
    buttonPress = 8;
    tunePotTracked = 0;
  }
}

void stepMenu() {

  stepCharArray[buttonPress] =
nameArray[noteArray[buttonPress]
% 12];
```

```
  stepRegisterArray[buttonPress]
= noteArray[buttonPress] / 12 -
1;

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(10, 0);
  display.println("M_STEPRv1");

  //-----------------------------
-----

  display.setTextSize(1);

  display.setCursor(20, 25);
  display.print("MODE:");
  if (playState == 1) {
    display.setCursor(70, 25);
    display.print("SEQ");
  }
  else {
    display.setCursor(70, 25);
    display.print("KEY");
  }

  display.setCursor(20, 40);
  display.print("BPM: ");
  display.setCursor(70, 40);
  display.print(BPM);

  display.setCursor(20, 55);
  display.print("STEP ");
  display.print(buttonPress);
  display.println(":");
  display.setCursor(70, 55);

display.print(stepCharArray[butt
onPress]);
  display.setCursor(85, 55);

display.println(stepRegisterArra
y[buttonPress]);
  display.display();
}

void stepSeq() {
//  digitalWrite(mlEnablePin,
0);
  if (millis() - prevMillisStep
>= beatDuration) {
    prevMillisStep = millis();
    stepNumber++;
    if (stepNumber > 8) {
      stepNumber = 1;
    }
    ledShift(stepNumber - 1);

Serial1.println(noteArray[stepNu
mber]);
    //
Serial.println(stepNumber);
  }
}


void liveKey() {
  if (buttonPress !=
lastButtonPress) {
    ledShift(buttonPress - 1);
    lastButtonPress =
buttonPress;
  }

  if (buttonValue < 200) {
    Serial1.println(0);
  }
  else {

Serial1.println(noteArray[button
Press]);
  }
}

void ledShift(byte bitNumber) {
  shiftBit = 0;

  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin,
MSBFIRST, shiftBit);
  digitalWrite(latchPin, HIGH);

  bitSet(shiftBit, bitNumber);
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin,
MSBFIRST, shiftBit);
  digitalWrite(latchPin, HIGH);
}

//void motorSyn(byte stepPin) {
//  if ((micros() -
prevMicrosPeriod >=
periodArray[stepNumber])) {
//    prevMicrosPeriod +=
periodArray[stepNumber];
//    digitalWrite(stepPin,
HIGH);
//    //
delayMicroseconds(periodArray[st
epNumber] / 2); // PULSE WIDTH
//    digitalWrite(stepPin,
LOW);
//  }
//}

//void motorSend() {
//
Serial1.println(noteArray[stepNu
mber]);
//}
//////////////SLAVE
#include "notePeriods.h"

#define stepPin1 2
#define mlEnablePin 3

const byte numChars = 32;
char receivedChars[numChars];
// an array to store the
received data
boolean newData = false;
int dataNumber = 0;

unsigned long prevMicrosPeriod =
0;
int currentNote = 0;

void setup() {
  Serial1.begin(115200);
  Serial1.println("<Arduino is
ready>");
  pinMode(stepPin1, OUTPUT);
  pinMode(mlEnablePin, OUTPUT);
}


void loop() {

  // ALL THIS IF FOR TESTING
  //  digitalWrite(mlEnablePin,
0);
  //  if ((micros() -
prevMicrosPeriod >= 700)) {
  //    prevMicrosPeriod += 700;
  //    digitalWrite(stepPin1,
HIGH);
  //
//delayMicroseconds(notePeriods[
currentNote] / 2); // PULSE
WIDTH
  //    digitalWrite(stepPin1,
LOW);
  //  }


  recvWithEndMarker();
  showNewNumber();

  if (notePeriods[currentNote]
== 0) {
    digitalWrite(mlEnablePin,
1);
  }
  else {
    digitalWrite(mlEnablePin,
0);
  }

  if ((micros() -
prevMicrosPeriod >=
notePeriods[currentNote])) {
    prevMicrosPeriod +=
notePeriods[currentNote];
    digitalWrite(stepPin1,
HIGH);

//delayMicroseconds(notePeriods[
currentNote] / 2); // PULSE
WIDTH
  digitalWrite(stepPin1, LOW);
  }
}

void recvWithEndMarker() {
  static byte ndx = 0;
  char endMarker = '\n';
  char rc;

  if (Serial1.available() > 0) {
    rc = Serial1.read();

    if (rc != endMarker) {
      receivedChars[ndx] = rc;
      ndx++;
      if (ndx >= numChars) {
        ndx = numChars - 1;
      }
    }
    else {
      receivedChars[ndx] = '\0';
// terminate the string
      ndx = 0;
      newData = true;
    }
  }
}

void showNewNumber() {
  if (newData == true) {
    currentNote =
atoi(receivedChars);   // new
for this version
    newData = false;
    //
Serial.println(currentNote);
    //
Serial.println(notePeriods[curre
ntNote]);
  }
}
```

Divergent Technologies / Czinger Vehicles

Actual work under NDA – images below exemplify structures and mechanisms developed – www.czinger.com

Ewan Baldry
Chief Engineer at Czinger Vehicles
July 18, 2020, Ewan managed Derek directly

Derek did an internship with us at Divergent. I would strongly recommend him to anyone looking for a bright, hard working and inquisitive engineer.

Derek has a very good base of engineering knowledge, however on top of that, he is the kind of person who can be set a task and left to get on with it. He uses his knowledge and intuition to get the job done. If he is unsure of anything he will respectfully ask for input and advice but is able to get his head around problems quickly and develops solutions that are thoroughly thought through and executable.

Double thumbs up from me !!! See less