High Speed Custom Scooter Throttle Control Prototype

Elliott Mendenhall

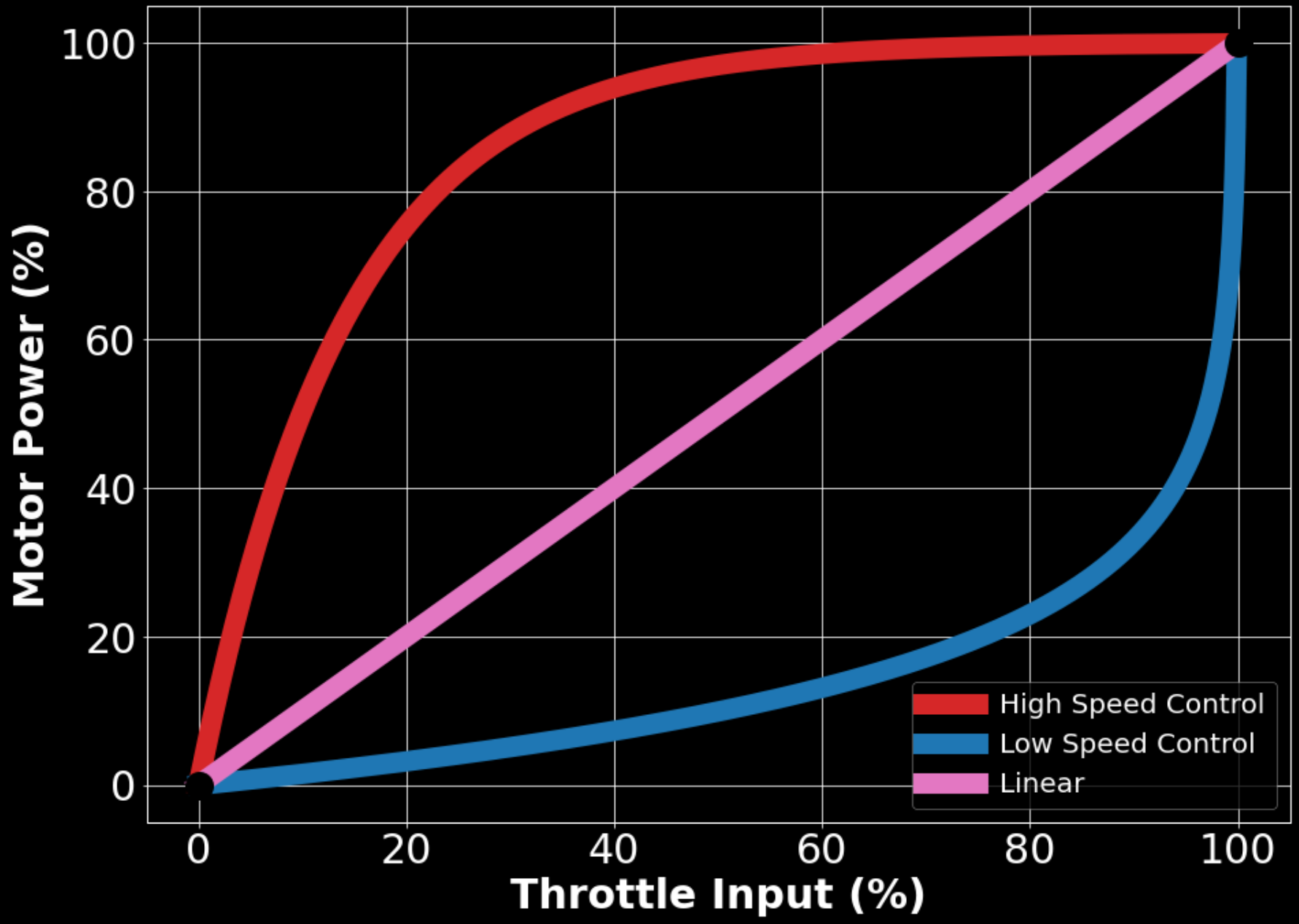
Boston University, eLab

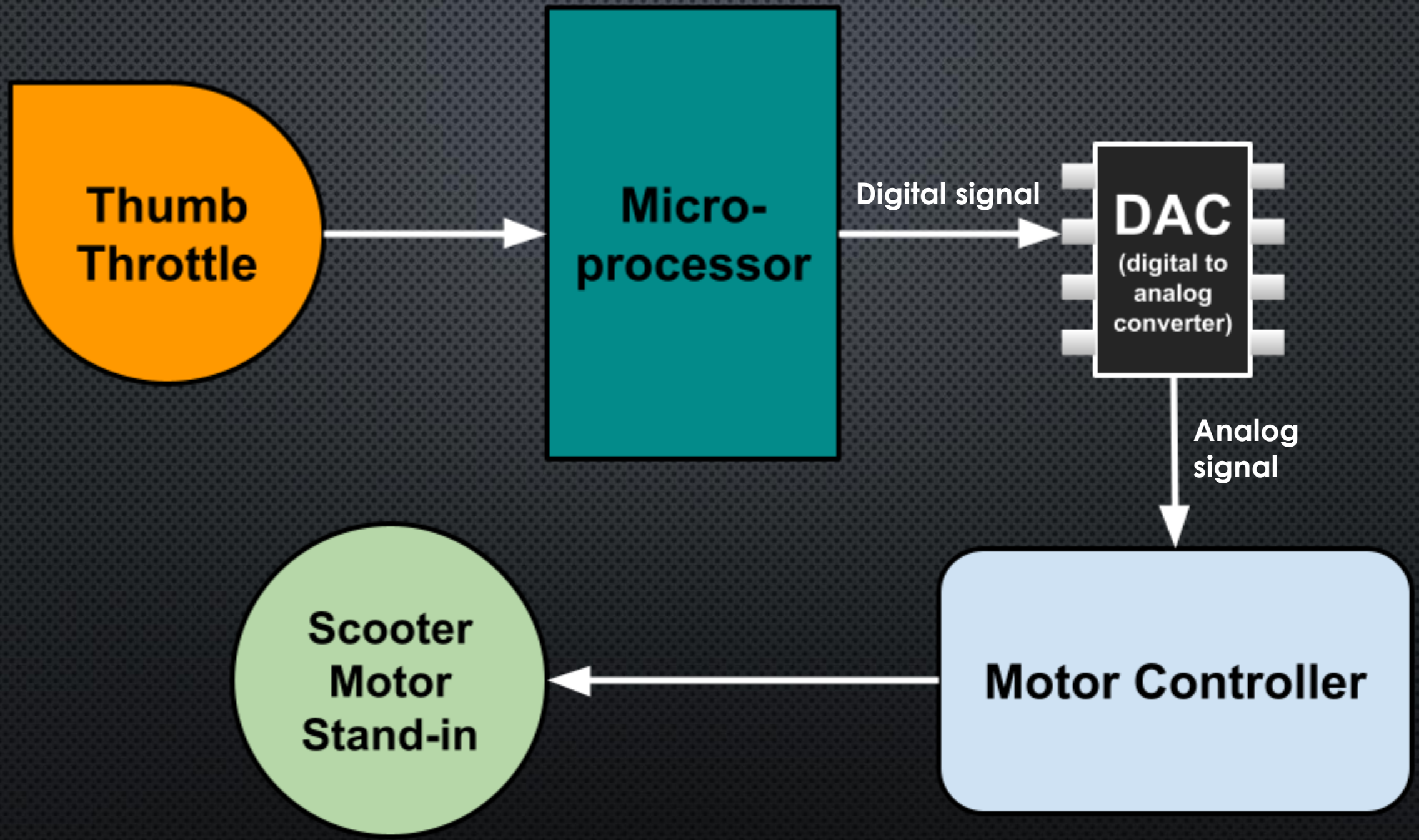
April 19, 2023

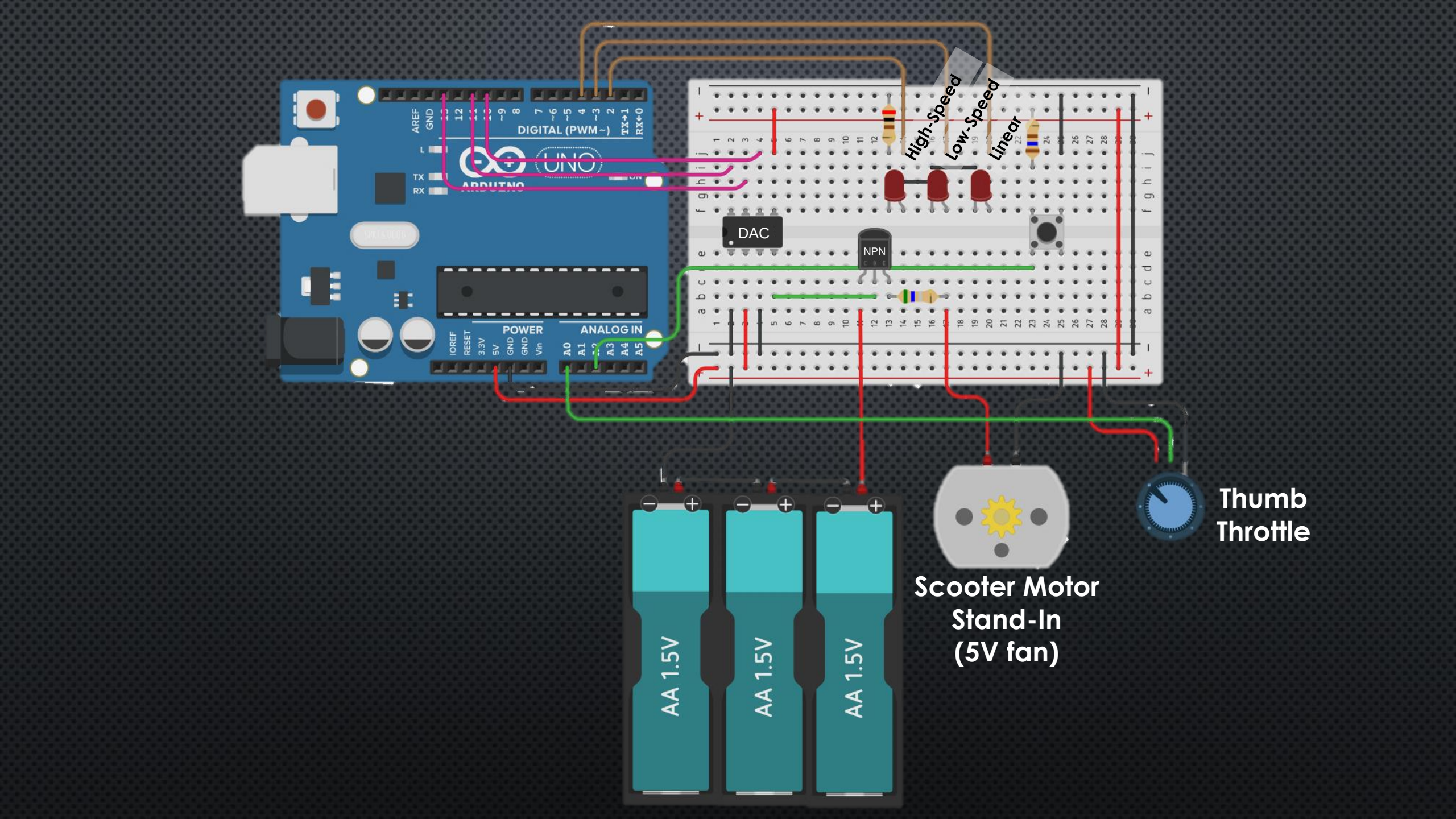
BOSTON UNIVERSITY

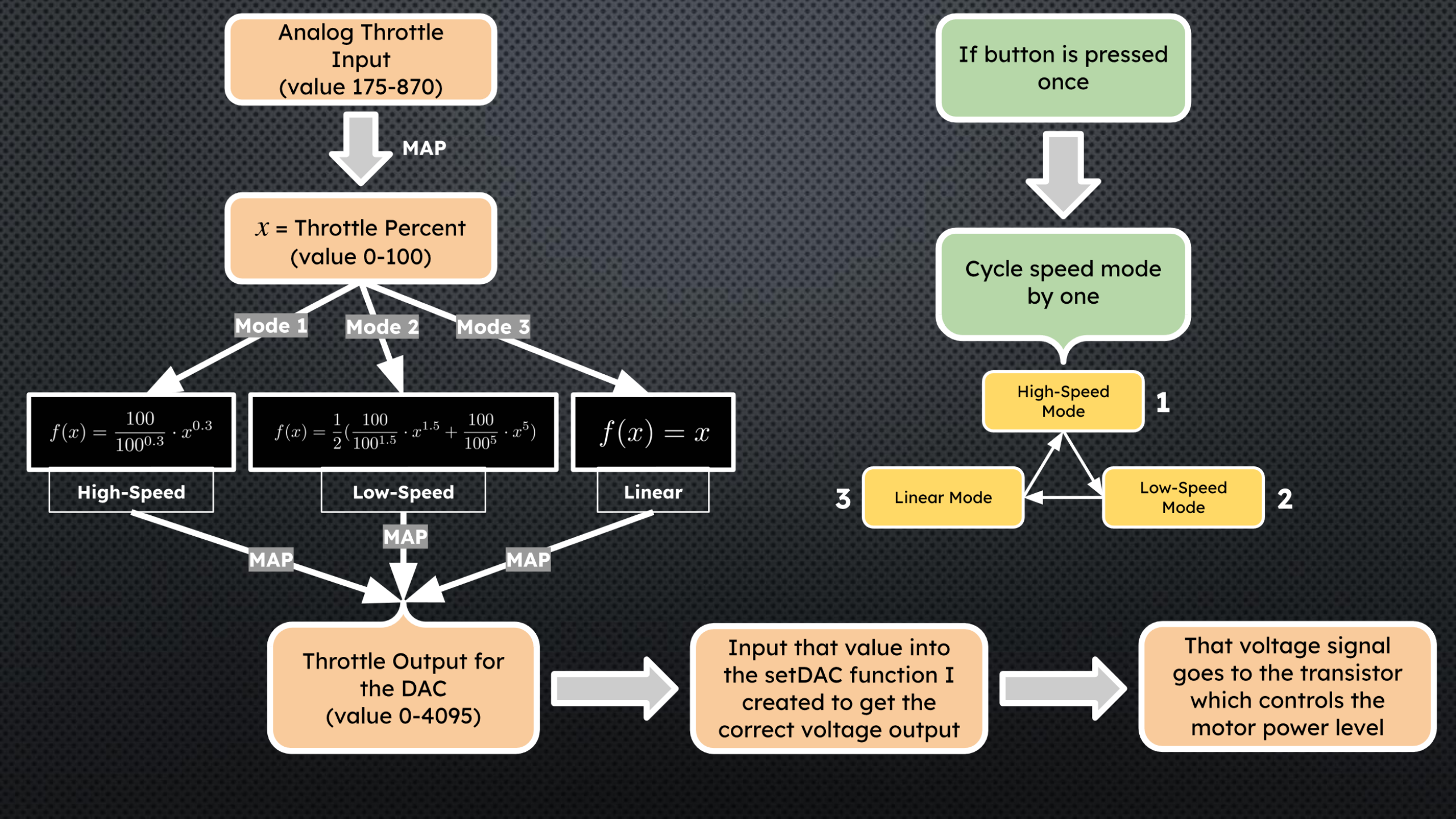

WOLF WARRIOR

Example Throttle Response Curves




Thumb Throttle
Micro-processor
Digital signal
DAC
(digital to analog converter)
Analog signal
Motor Controller
Scooter Motor Stand-in


High-Speed
Low-Speed
Linear
DAC
NPN
DIGITAL (PWM~)
POWER
ANALOG IN
AA 1.5V
AA 1.5V
AA 1.5V
Scooter Motor
Stand-In
(5V fan)
Thumb
Throttle


Analog Throttle Input (value 175-870)
MAP
x = Throttle Percent (value 0-100)
Mode 1
Mode 2
Mode 3
$f(x) = \frac{100}{100^{0.3}} \cdot x^{0.3}$
$f(x) = \frac{1}{2}(\frac{100}{100^{1.5}} \cdot x^{1.5} + \frac{100}{100^{5}} \cdot x^{5})$
$f(x) = x$
High-Speed
Low-Speed
Linear
MAP
MAP
MAP
Throttle Output for the DAC (value 0-4095)
Input that value into the setDAC function I created to get the correct voltage output
That voltage signal goes to the transistor which controls the motor power level
If button is pressed once
Cycle speed mode by one
High-Speed Mode
1
Low-Speed Mode
2
Linear Mode
3

PARTS

- Arduino Uno
- 3 Pin Thumb Throttle
- 5V DC Motor with fan blades
- 2N3904 NPN Transistor
- MCP4921 DAC
- External 5V power supply (3 AA battery pack)
- Pushbutton
- LED’s

DATA SHEETS

- http://physics.bu.edu/adlab_and_elab/wp-content/uploads/2022/05/MCP4921.pdf
- https://www.sparkfun.com/datasheets/Components/2N3904.pdf
- https://components101.com/sites/default/files/component_datasheet/Toy%20DC%20motor%20Datasheet.pdf

Final_Code | Arduino IDE 2.1.0

File Edit Sketch Tools Help

Arduino Uno

Final_Code.ino

```ino
#include <SPI.h>  //import the SPI library

int dac_nCS = 10; // pin number for nCS
int dac_SCK = 13; // pin number for SCK
int dac_SDI = 11; // pin number for SDI

void setup(){
  Serial.begin(9600);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(13, OUTPUT);

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);

  pinMode(A0, INPUT);
  pinMode(A2, INPUT);

  SPI.begin();                    // initialize the library
  SPI.setBitOrder(MSBFIRST);      // tell the library to use the most significant bit (MSB) first
  pinMode(dac_nCS, OUTPUT); // define nCS as an output
  digitalWrite(dac_nCS, 1);
  pinMode(dac_SCK, OUTPUT); // define SCK as an output
  digitalWrite(dac_SCK, 0);
  pinMode(dac_SDI, OUTPUT); // define SCK as an output
  digitalWrite(dac_SDI, 0);
}


int throttle_output;      // defines the "throttle_output" variable, which will be the binary value (in base 10 form) given to the DAC to output our desire

int mode = 1;             // defines the "mode" variable, which acts as the switch between high-speed, low-speed, and linear speed modes
int change = 0;           // defines the "change" variable, which guarantees that one button press gives one mode change, regardless of the button press le

float slope = 3;          // defines the "slope" variable, which determines the steepness of the high-speed throttle response curve
float low_slope = 1.5;    // defines the "low_slope" variable, which determines the steepness of the low-speed throttle response curve
float low_slope2 = 5;     // defines the "low_slope2" variable, which also helps determine the steepness of the low-speed throttle response curve

void loop(){

  Serial.print("0%:");  Serial.print(0);           // Serial.prints which show the upper and lower boundary of the serial plotter at 0% and 100% for easie
  Serial.print(" , ");
  Serial.print("100%:");  Serial.print(100);
  Serial.print(" , ");


  int original_config_bits = 28672;             // decimal value that equals "0111 0000 0000 0000" the config bits for the DAC
  int transistor_correction = 420;              // voltage correction for the transistor to make the DAC output an initial 0.6 volts for the initial satura
  int motor_activation_correction = 650;        // voltage correction for the activation voltage of the DC motor
  int config_bits = original_config_bits + transistor_correction + motor_activation_correction;
  // automatically adds the voltage corrections to the DAC.  So even when there is not throttle input, the DAC still outputs a set 0.6V + 0.8V for the trar
  int map_val = 4095 - transistor_correction - motor_activation_correction;   // new upper map value for the throttle_output variable



  int throttle_val = analogRead(A0);                                // reads the analog input from the thumb throttle and sets it to the int "throttle_val
  int throttle_percent = map(throttle_val, 175, 870, 0, 100);       // maps the analog input from the thumb throttle to a percentage value (0-100) and set

  int button = digitalRead(A2);
  if (button == 0 && change == 0){                                  // button counter (single click results in one increase of the mode variable regardles
    if (mode == 3){
      mode = 0;
    }
    mode++;
    Serial.println(mode);
```

Output Serial Monitor ×

Not connected. Select a board and a port to connect automatically.

Final_Code | Arduino IDE 2.1.0

File Edit Sketch Tools Help

Arduino Uno

Final_Code.ino

```ino
    float high_speed_control_percent = ((100 / (pow(100 , 1/slope)) * (pow(throttle_percent , 1/slope))));        // equation to map the raw throttle perce
    int throttle_output = map(high_speed_control_percent, 0, 100, 0, map_val);                                    // mapping from the high_speed_control_pe

    digitalWrite(2, HIGH);      // Light up the "High-Speed" LED
    digitalWrite(3, LOW);       // Turn off the "Low-Speed" LED
    digitalWrite(4, LOW);       // Turn off the "Linear" LED

    Serial.print("Throttle_Input_Percent:"); Serial.print(throttle_percent);                 // print out the Throttle_Input_Percent and Motor_Speed_Outpu
    Serial.print("% , ");
    Serial.print("Motor_Speed_Output_Percent:"); Serial.print(high_speed_control_percent);
    Serial.print("%");
    Serial.println(" --- High Speed Control ");

    setDac(config_bits + throttle_output);        // set the DAC to our desired output using the throttle_output variable

  }


  if (mode == 2){                                                 // low speed control if statement
    float low_speed_control_percent = (((100 / (pow(100 , low_slope))) * (pow(throttle_percent , low_slope))) + (100 / (pow(100 , low_slope2))) * (pow(thro
    int throttle_output = map(low_speed_control_percent, 0, 100, 0, map_val);

    digitalWrite(3, HIGH);      // Light up the "Low-Speed" LED
    digitalWrite(2, LOW);       // Turn off the "High-Speed" LED
    digitalWrite(4, LOW);       // Turn off the "Linear" LED

    Serial.print("Throttle_Input_Percent:"); Serial.print(throttle_percent);                  // print out the Throttle_Input_Percent and Motor_Speed_Out
    Serial.print("% , ");
    Serial.print("Motor_Speed_Output_Percent:"); Serial.print(low_speed_control_percent);
    Serial.print("%");
    Serial.println(" --- Low Speed Control ");

    setDac(config_bits + throttle_output);        // set the DAC to our desired output using the throttle_output variable
  }


  if (mode == 3){                                                 // linear speed control if statement
    float linear_speed_control_percent = throttle_percent;
    int throttle_output = map(linear_speed_control_percent, 0, 100, 0, map_val);

    digitalWrite(4, HIGH);      // Light up the "Linear" LED
    digitalWrite(2, LOW);       // Turn off the "High-Speed" LED
    digitalWrite(3, LOW);       // Turn off the "Low-Speed" LED

    Serial.print("Throttle_Input_Percent:"); Serial.print(throttle_percent);                  // print out the Throttle_Input_Percent and Motor_Speed_(
    Serial.print("% , ");
    Serial.print("Motor_Speed_Output_Percent:"); Serial.print(linear_speed_control_percent);
    Serial.print("% , ");
    Serial.println(" --- Linear Speed Control ");

    setDac(config_bits + throttle_output);        // set the DAC to our desired output using the throttle_output variable
  }


  delay(5);
}

void setDac(word ori){            // Create a new word (16 bit variable) called ori, which is the new input of the setDac function.  Take this input and sp
                                  // These new words are the inputs into the SPI.transfer commands which encode the bits to go to the DAC
  digitalWrite(dac_nCS, 0); // set nCS to 0 (LOW)
  word a = ori >> 8;
  word b = (ori << 8) >> 8;
  SPI.transfer(a);
  SPI.transfer(b);
  digitalWrite(dac_nCS, 1); // set nCS to 1 (HIGH)
}
```

Output Serial Monitor ×

Not connected. Select a board and a port to connect automatically.