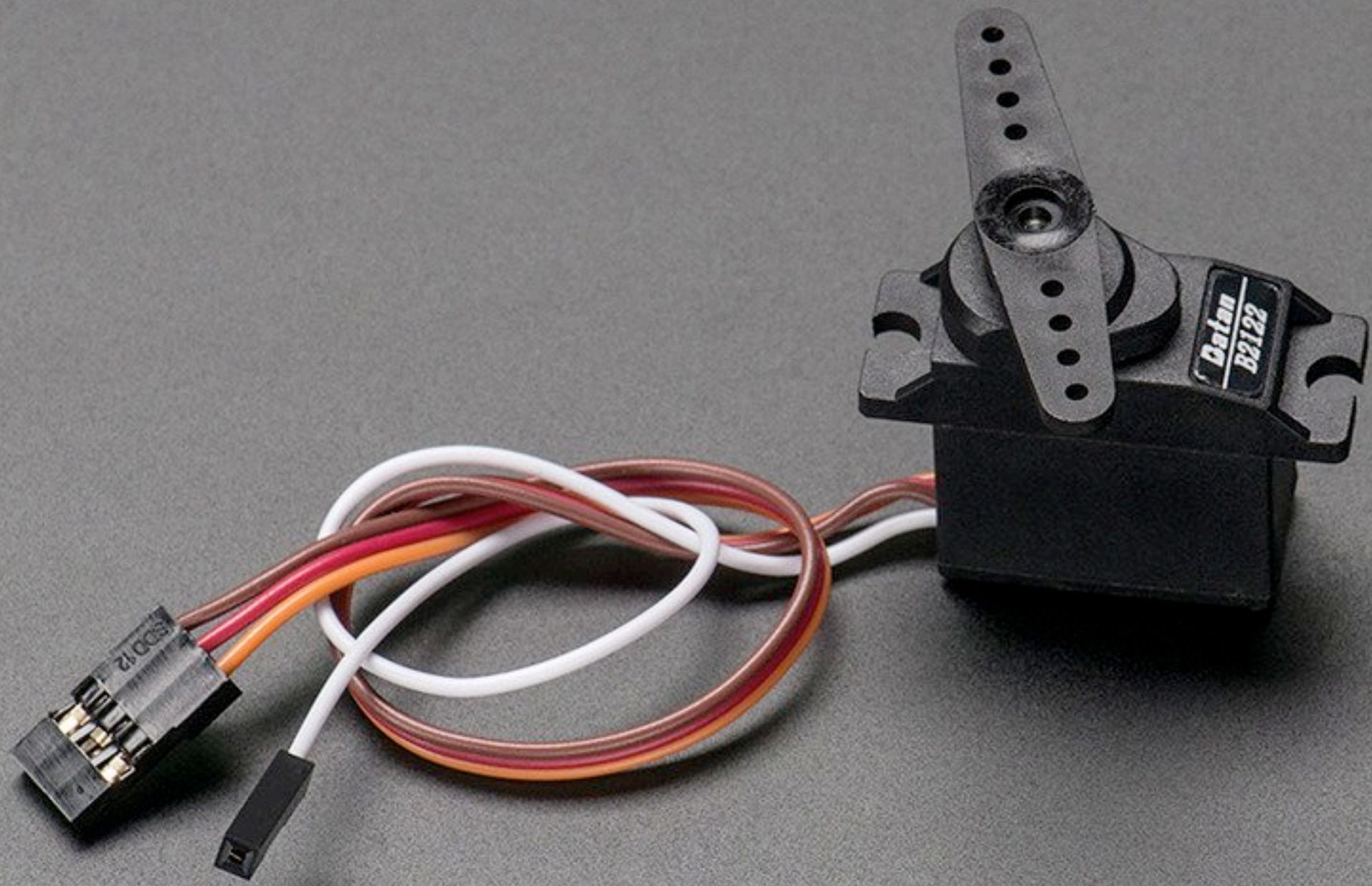
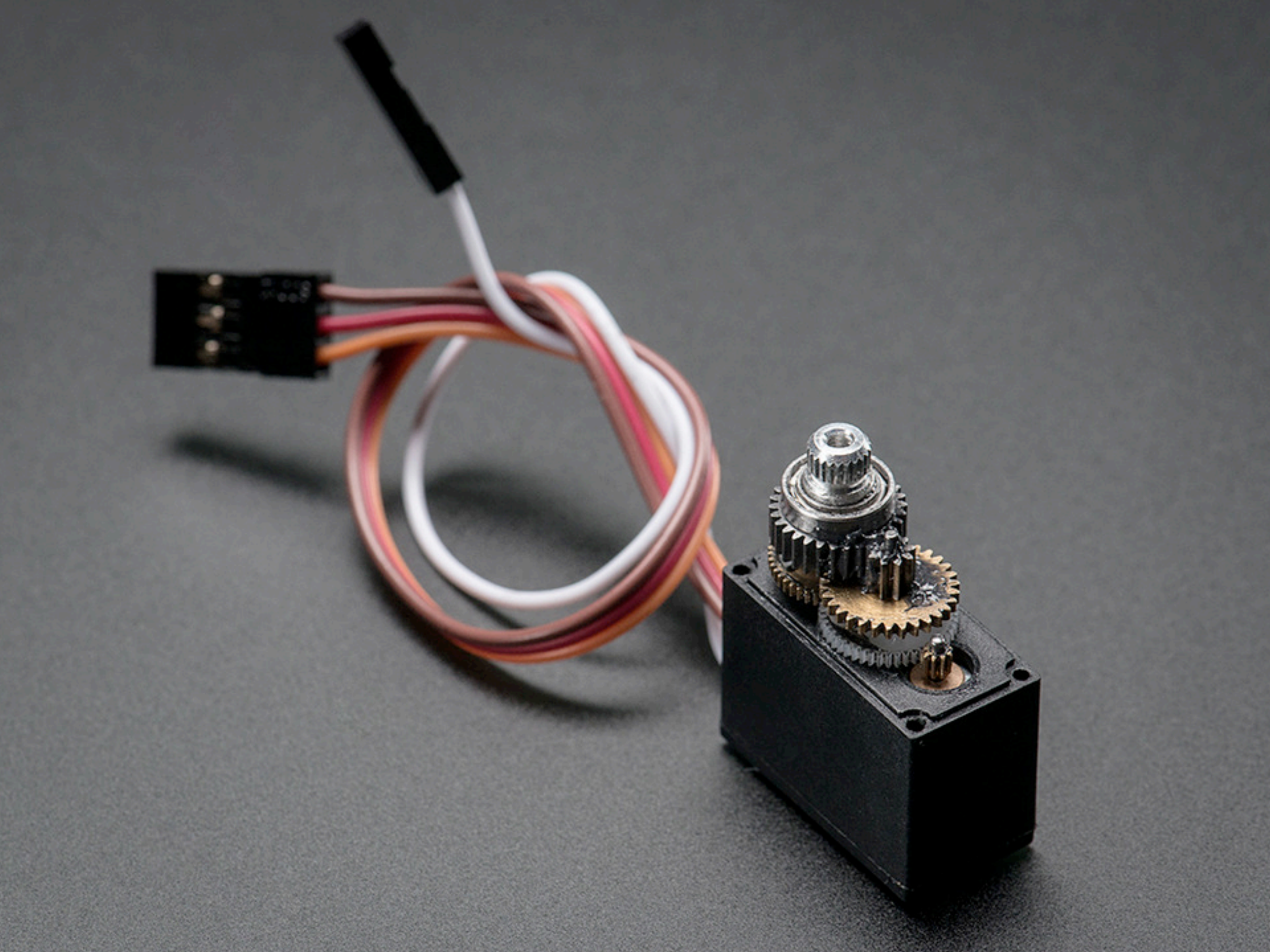
ACTUATORS

SERVOS


Datan
B2122



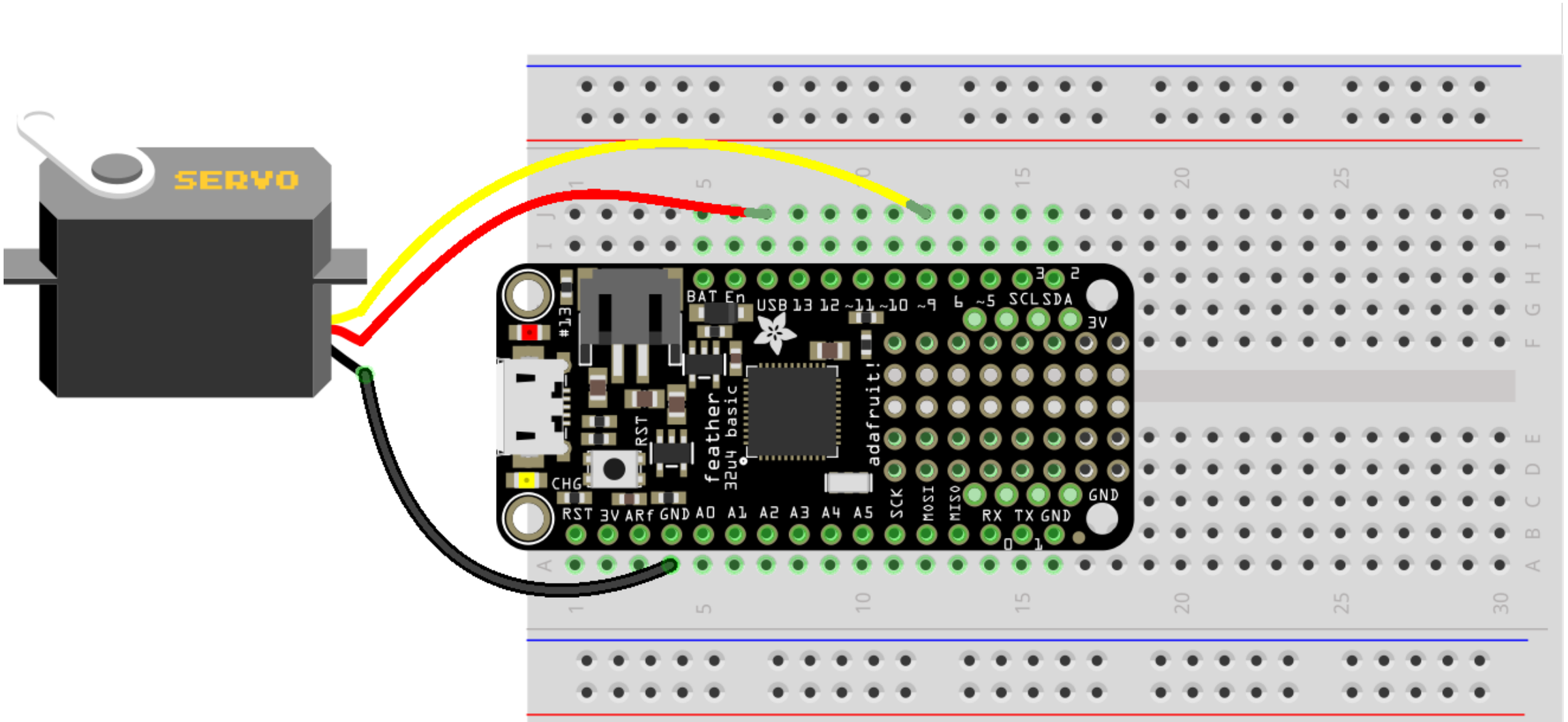
RED => POWER
BLACK (or BROWN) => GROUND
YELLOW (or ORANGE) => INPUT

Input accepts a range from 0 to 180.

RED => POWER

BLACK (or BROWN) => GROUND

YELLOW (or ORANGE) => INPUT



CONTROL SERVO

sketch_feb25a | Arduino 1.8.10

sketch_feb25a §

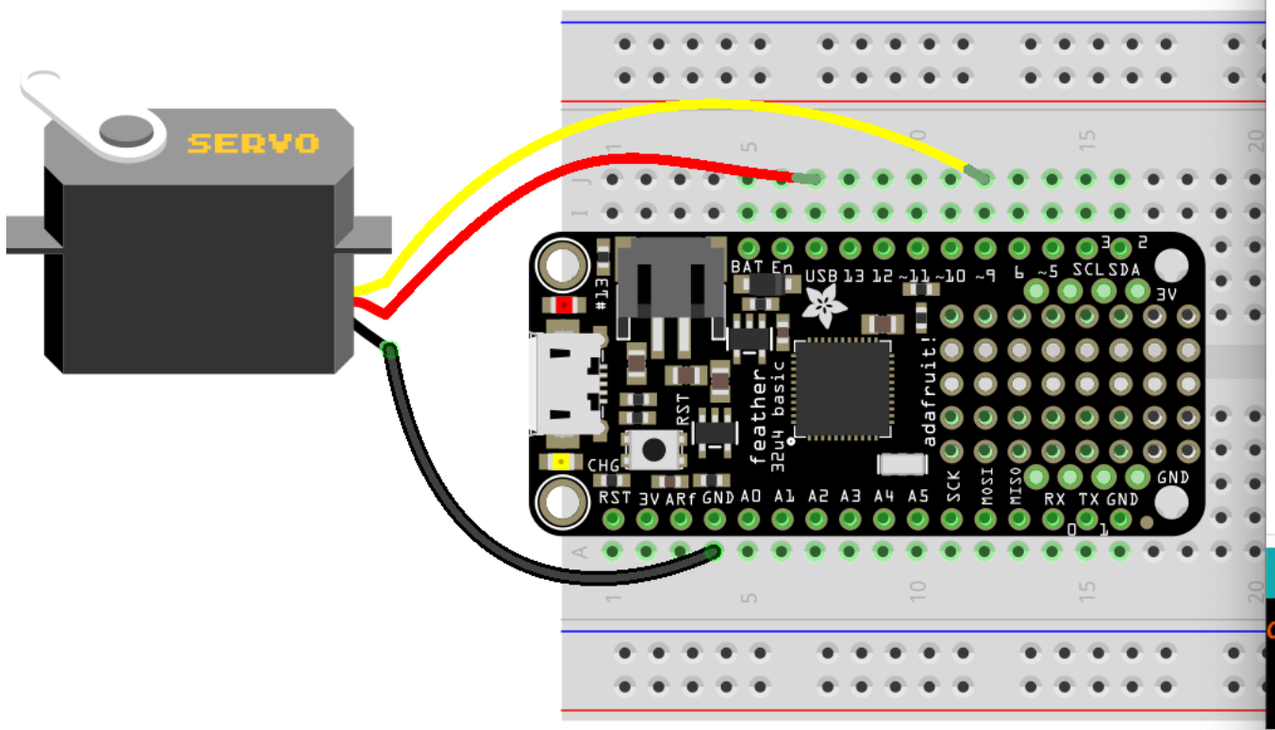
```
#include <Servo.h>

Servo myservo;
const int SERVO_PIN = 9;

void setup() {
  myservo.attach(SERVO_PIN, 430, 2400);
}

void loop() {
  int angle = 0;
  int step = 1;

  for (angle = 0; angle <= 180; angle += step) {
    myservo.write(angle);
    delay(5);
  }

  for (angle = 180; angle >= 0; angle -= step) {
    myservo.write(angle);
    delay(5);
  }
}

```

Done uploading.

```
avrdude done.  Thank you.
```

24 Adafruit Feather 32u4 on /dev/cu.usbmodem14101

CONTROL SERVO



sketch_feb25a | Arduino 1.8.10

sketch_feb25a §

```
#include <Servo.h>

Servo myservo;
const int SERVO_PIN = 9;

void setup() {
  myservo.attach(SERVO_PIN, 430, 2400);
}

void loop() {
  int angle = 0;
  int step = 1;

  for (angle = 0; angle <= 180; angle += step) {
    myservo.write(angle);
    delay(5);
  }

  for (angle = 180; angle >= 0; angle -= step) {
    myservo.write(angle);
    delay(5);
  }
}

```

Done uploading.

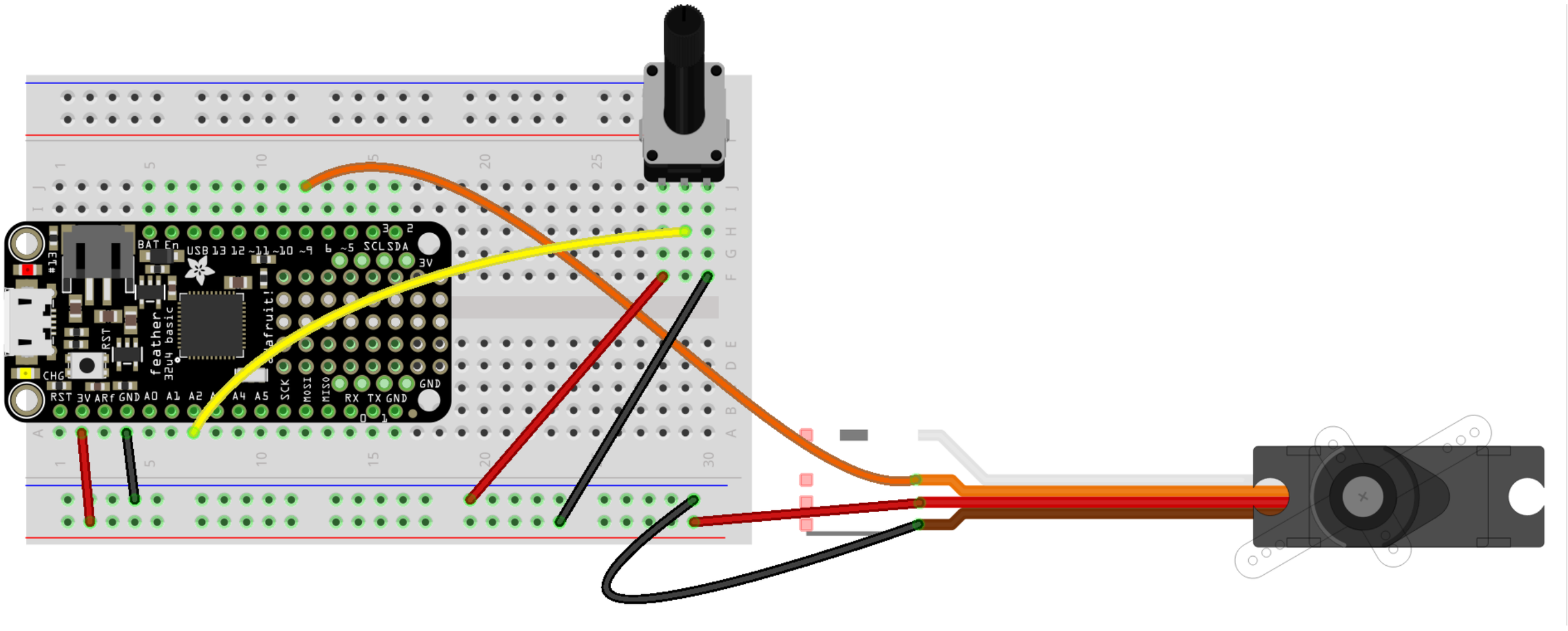
```
avrdude done.  Thank you.
```

24 Adafruit Feather 32

RED => POWER

BLACK (or BROWN) => GROUND

YELLOW (or ORANGE) => INPUT



CONTROL SERVO WITH ANALOG INPUT

sketch_feb25b | Arduino 1.8.10

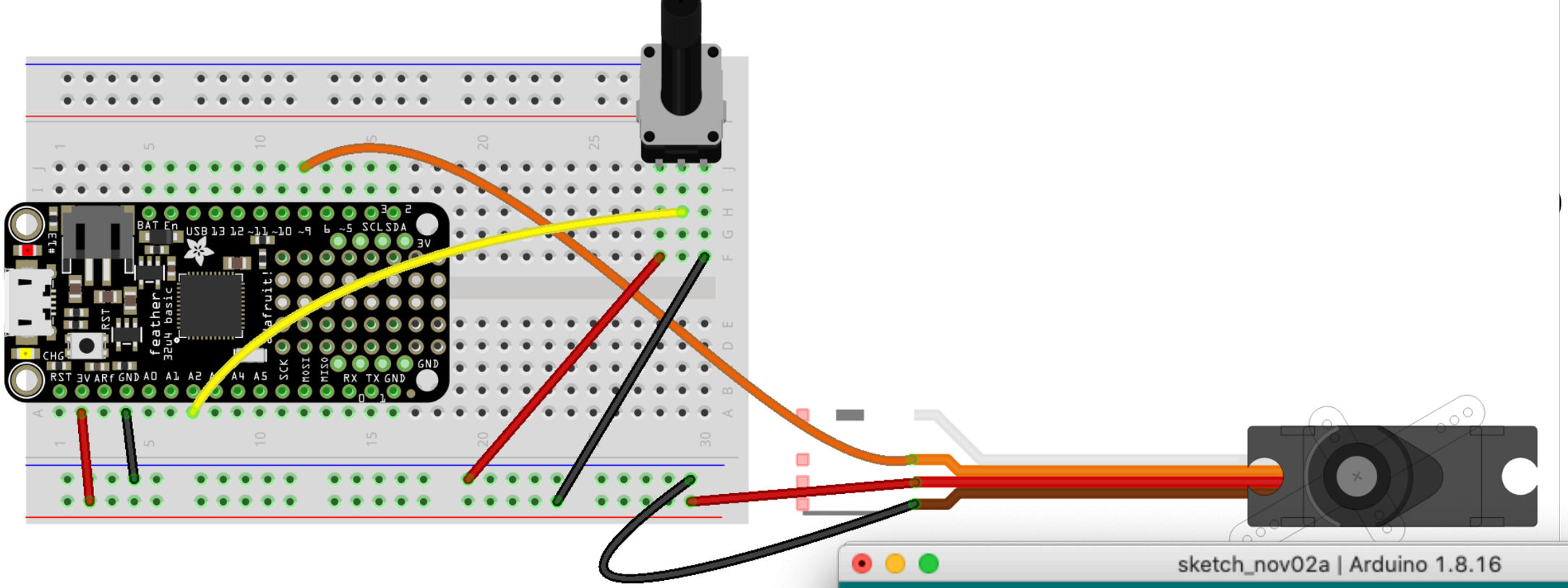
sketch_feb25b

```
#include <Servo.h>

Servo myservo;
const int SERVO_PIN = 9;
const int POT_PIN = A1;

void setup() {
  myservo.attach(SERVO_PIN, 430, 2400);
  pinMode(POT_PIN, INPUT);
}

void loop() {
  int val = analogRead(POT_PIN);
  int angle = map(val, 0, 1024, 0, 180);
  myservo.write(angle);
  delay(15);
}

```

Done Saving.

```
avrdude done.  Thank you.
```

18 Adafruit Feather 32u4 on /dev/cu.usbmodem14101



sketch_nov02a | Arduino 1.8.16

sketch_nov02a §

```
#include <Servo.h>

Servo myservo;
const int SERVO_PIN = 9;
const int POT_PIN = A2;

void setup() {
  myservo.attach(SERVO_PIN, 430, 2400);
}

void loop() {
  int potVal = analogRead(POT_PIN);
  int pos = map(potVal, 0, 1024, 0, 180);

  myservo.write(pos);
  delay(15);
}

```

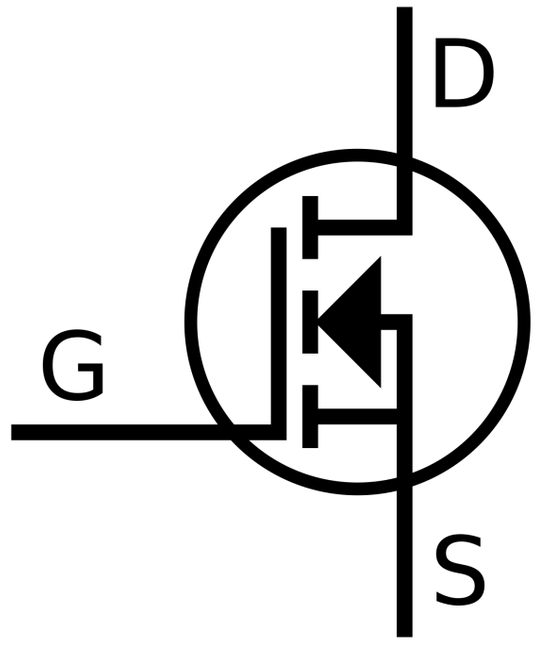
CONTROLLING HIGH POWER LOADS WITH TRANSISTORS

When using electro-mechanical devices, we'll be dealing with currents and voltages that are larger than Arduino's IO pins can handle.

A transistor is an electronic component that can help to amplify small electric currents (about 15mA) that the Arduino can provide.

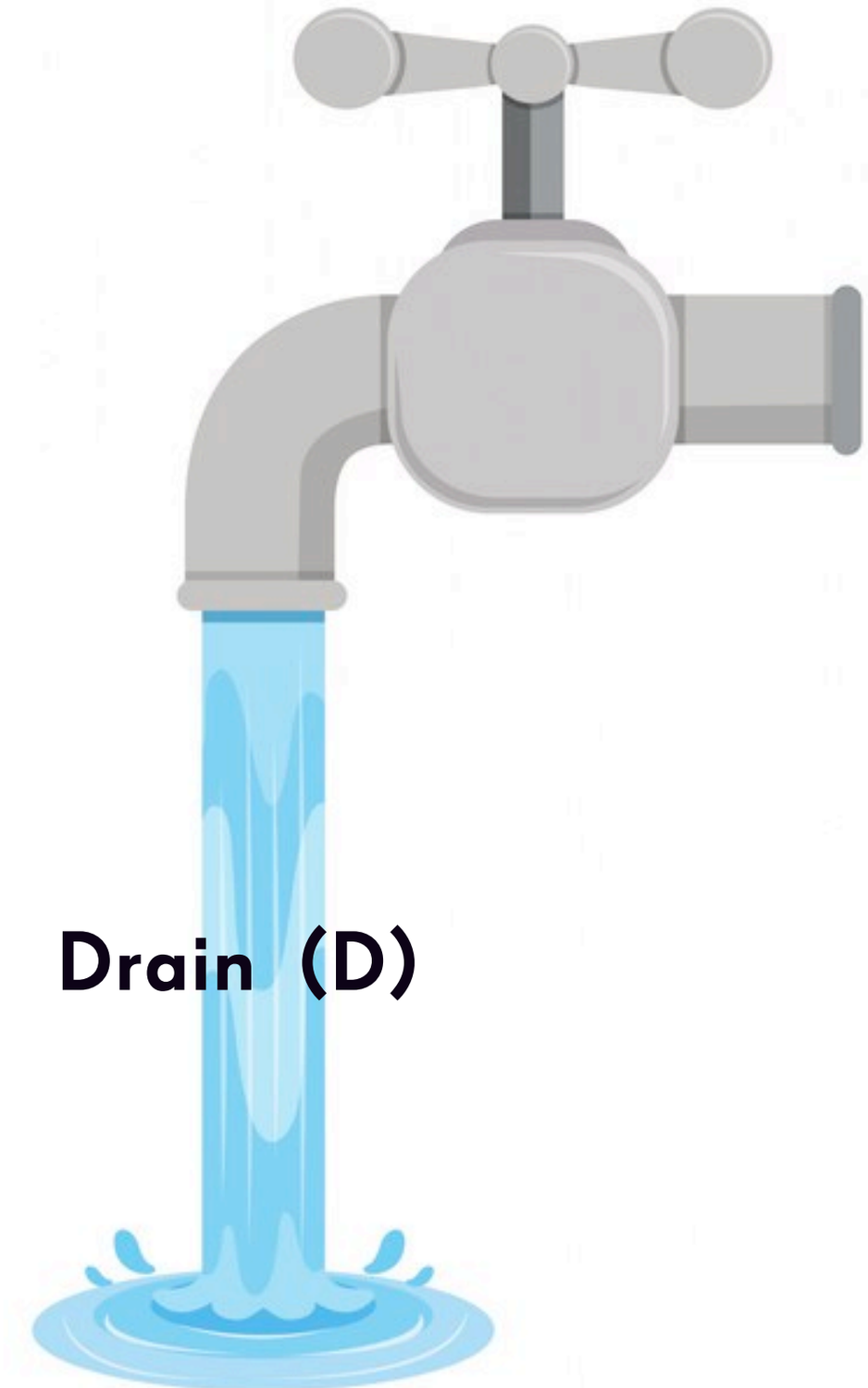
We are going to be using a particular kind of transistor known as a MOSFET. The main advantage to this type of transistor is that it is more efficient, and therefore doesn't heat up as much as other types of transistor.

A MOSFET has three leads:
- gate (G)
- drain (D)
- source (S)



The rule is that small gate voltage controls a larger drain voltage, so we can control high-power devices connected to the drain by switching the gate voltage with the Arduino.

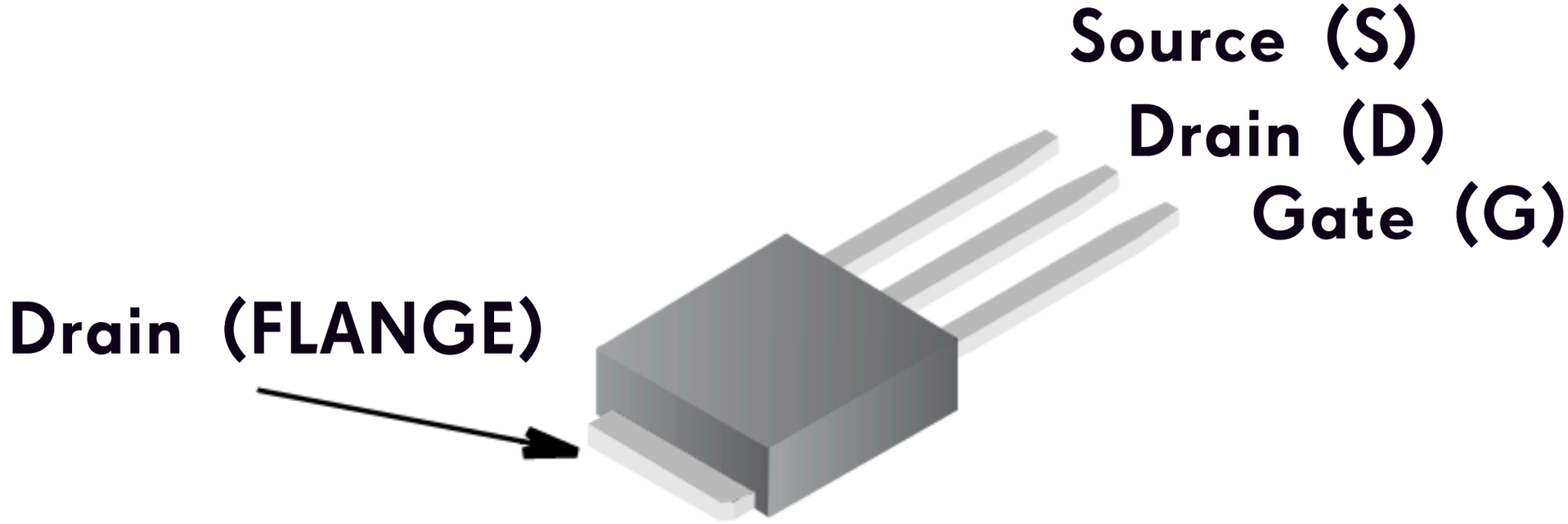
Gate (G)



Source (S)

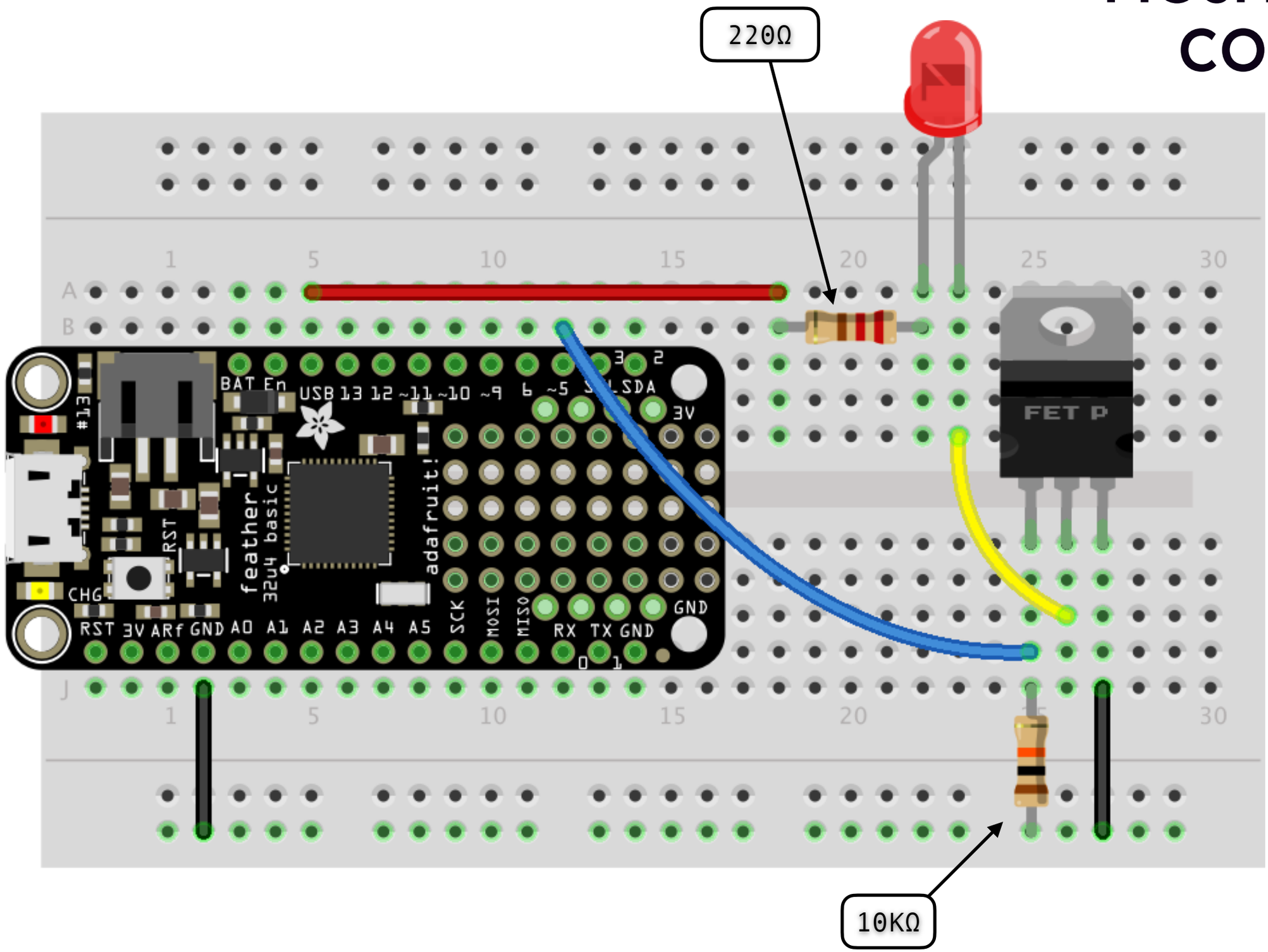
Drain (D)

When you send a HIGH signal to the gate, the MOSFET switches and allows current to flow from the source to the drain.

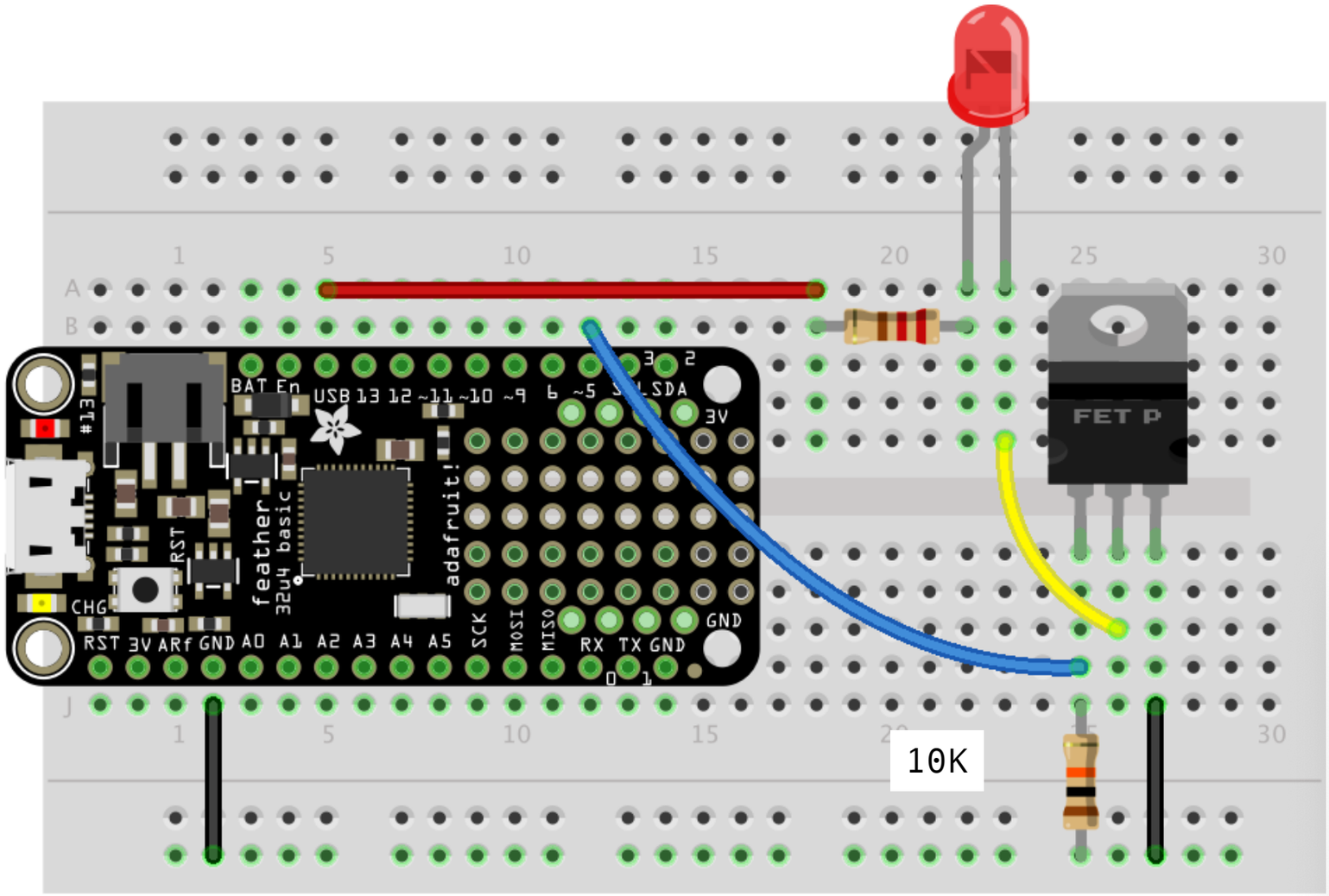


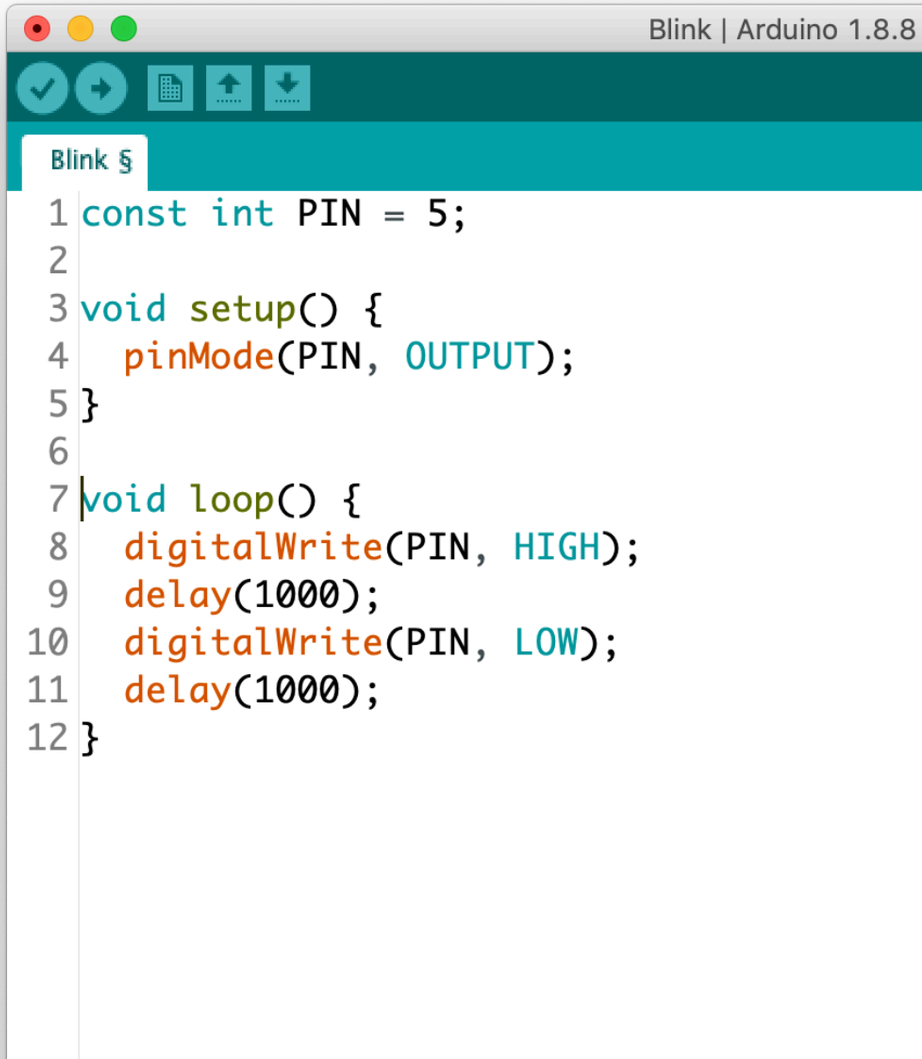
When you send a HIGH signal to the gate, the MOSFET switches and allows current to flow from the source to the drain.

MOSFET LED CONTROL

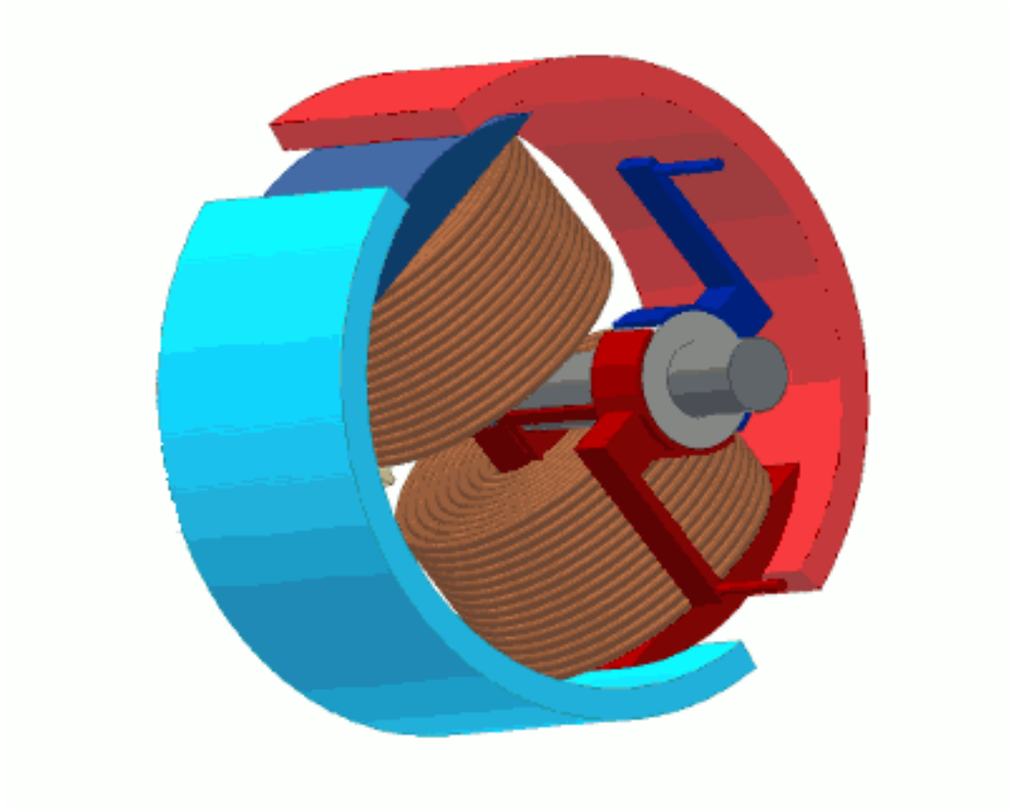


MOSFET LED CONTROL

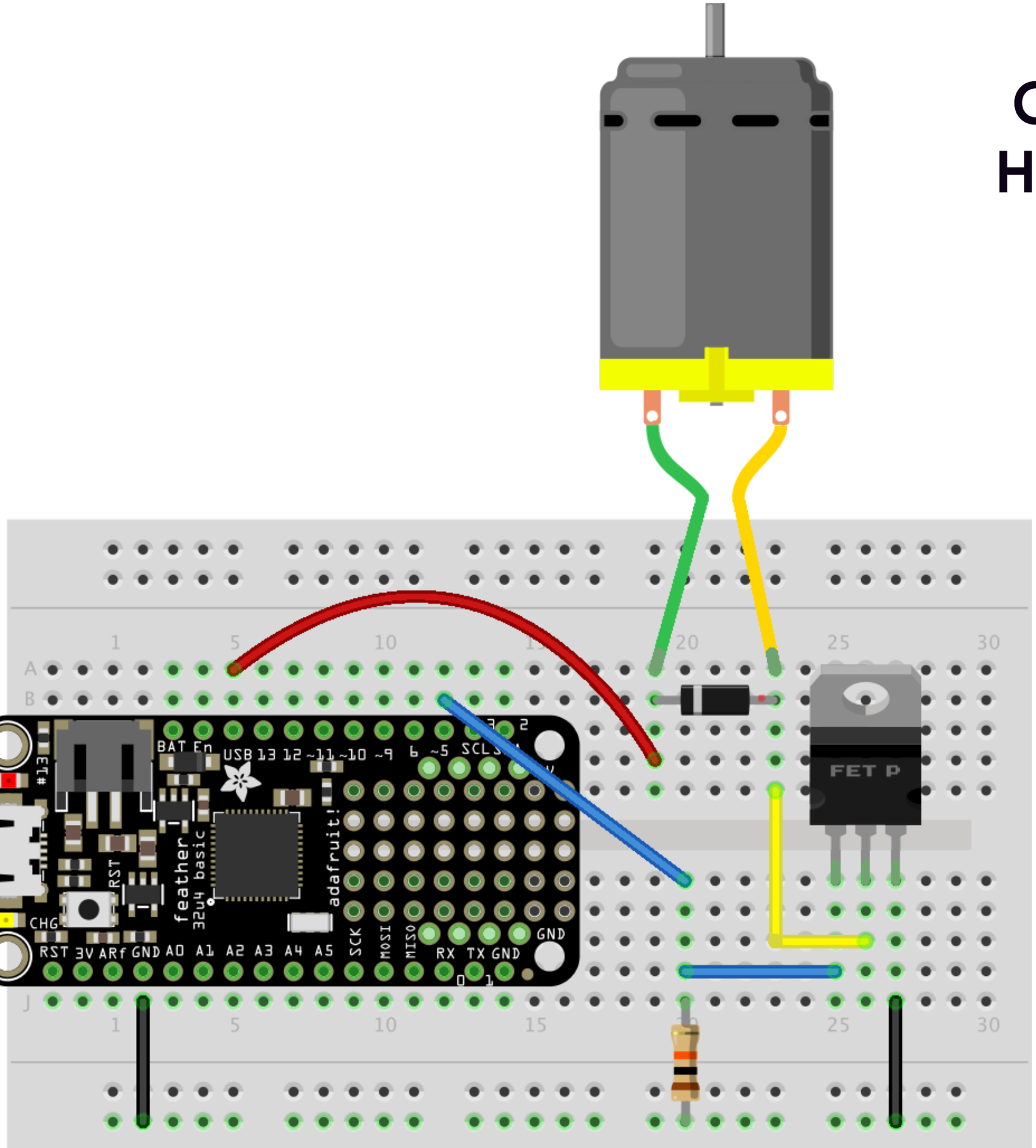




DC MOTOR



CONTROLLING HIGH-CURRENT LOADS



“BLINK” A MOTOR

motor | Arduino 1.8.5

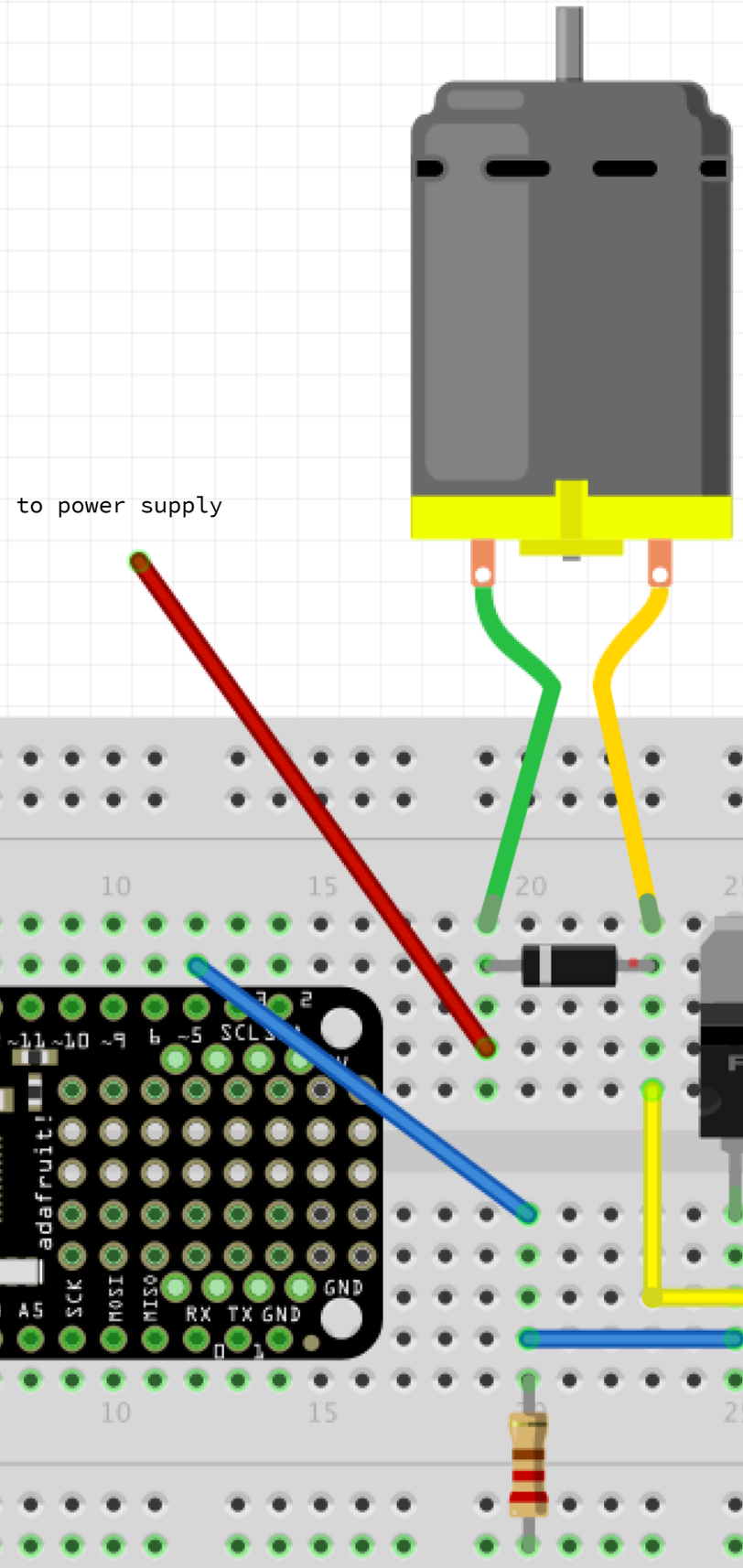
motor

```
1  const int MOTOR_PIN = 5;
2
3  void setup() {
4    pinMode(MOTOR_PIN, OUTPUT);
5  }
6
7  void loop() {
8    digitalWrite(MOTOR_PIN, HIGH);
9    delay(1000);
10   digitalWrite(MOTOR_PIN, LOW);
11   delay(1000);
12 }
```

Done Saving.

10 Adafruit Feather 32u4 on /dev/cu.usbmodem148

CONTROLLING HIGH-CURRENT LOADS

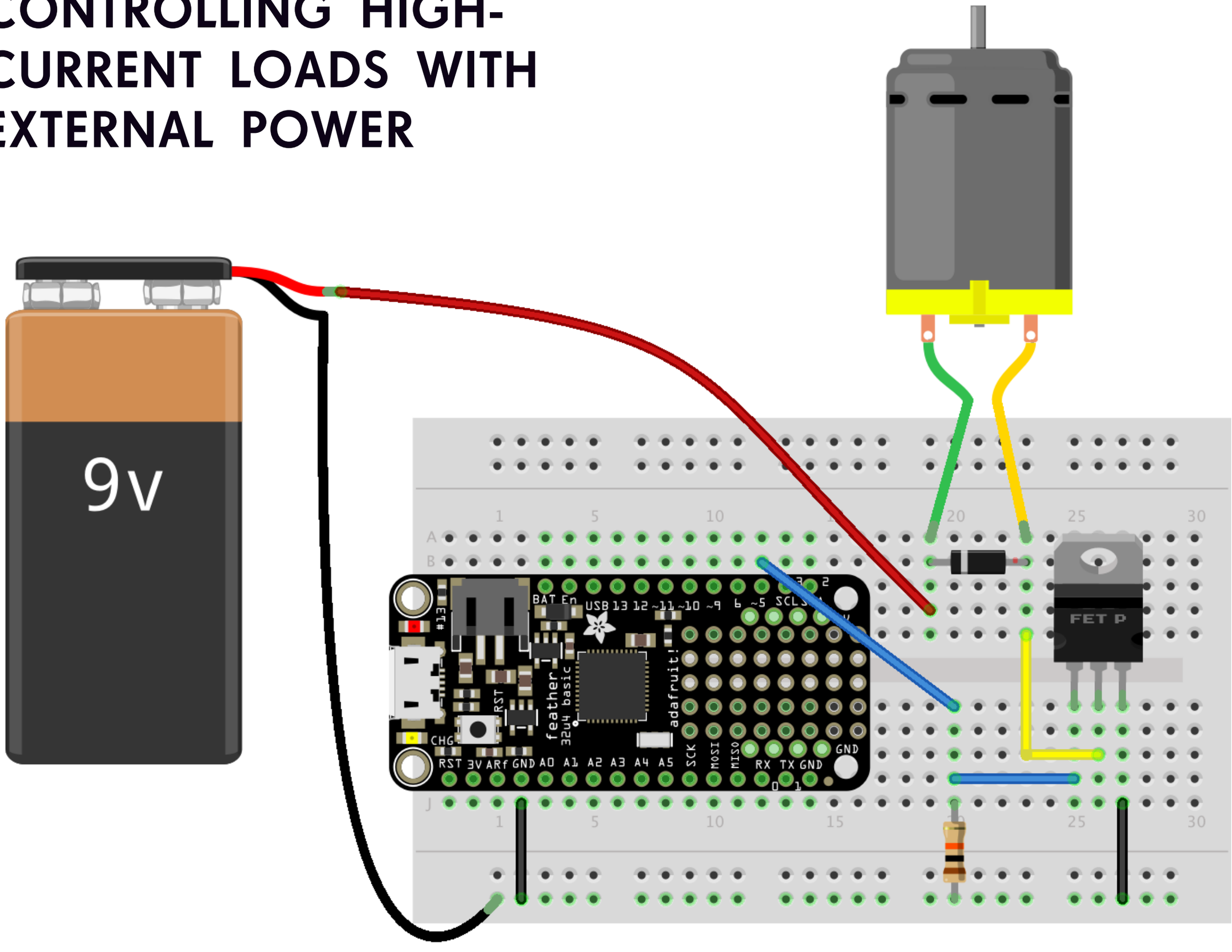


motor | Arduino 1

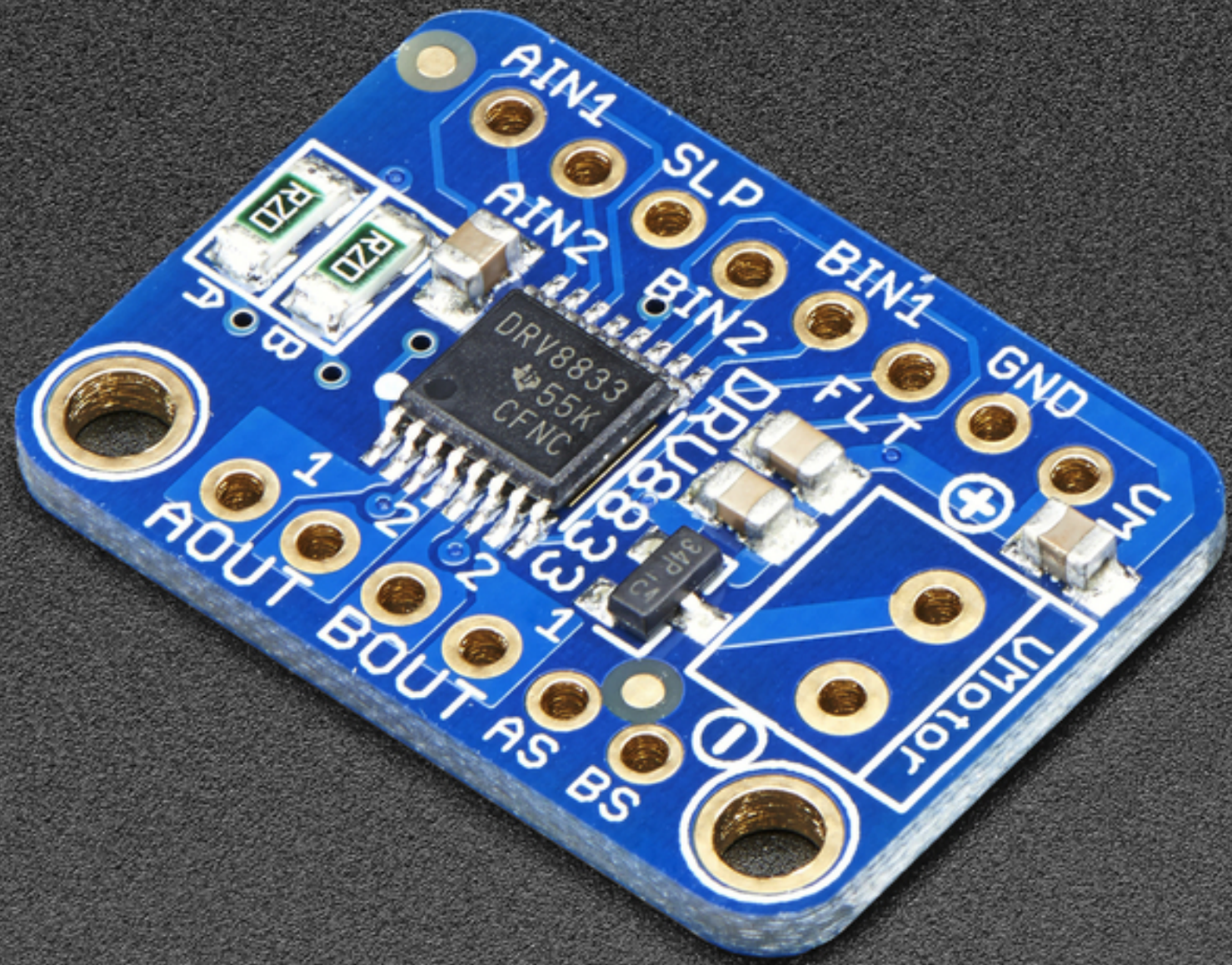
motor

```
1  const int MOTOR_PIN = 5;
2
3  void setup() {
4    pinMode(MOTOR_PIN, OUTPUT);
5  }
6
7  void loop() {
8    digitalWrite(MOTOR_PIN, HIGH);
9    delay(1000);
10   digitalWrite(MOTOR_PIN, LOW);
11   delay(1000);
12 }
```

CONTROLLING HIGH-CURRENT LOADS WITH EXTERNAL POWER



DRV8833 MOTOR DRIVER

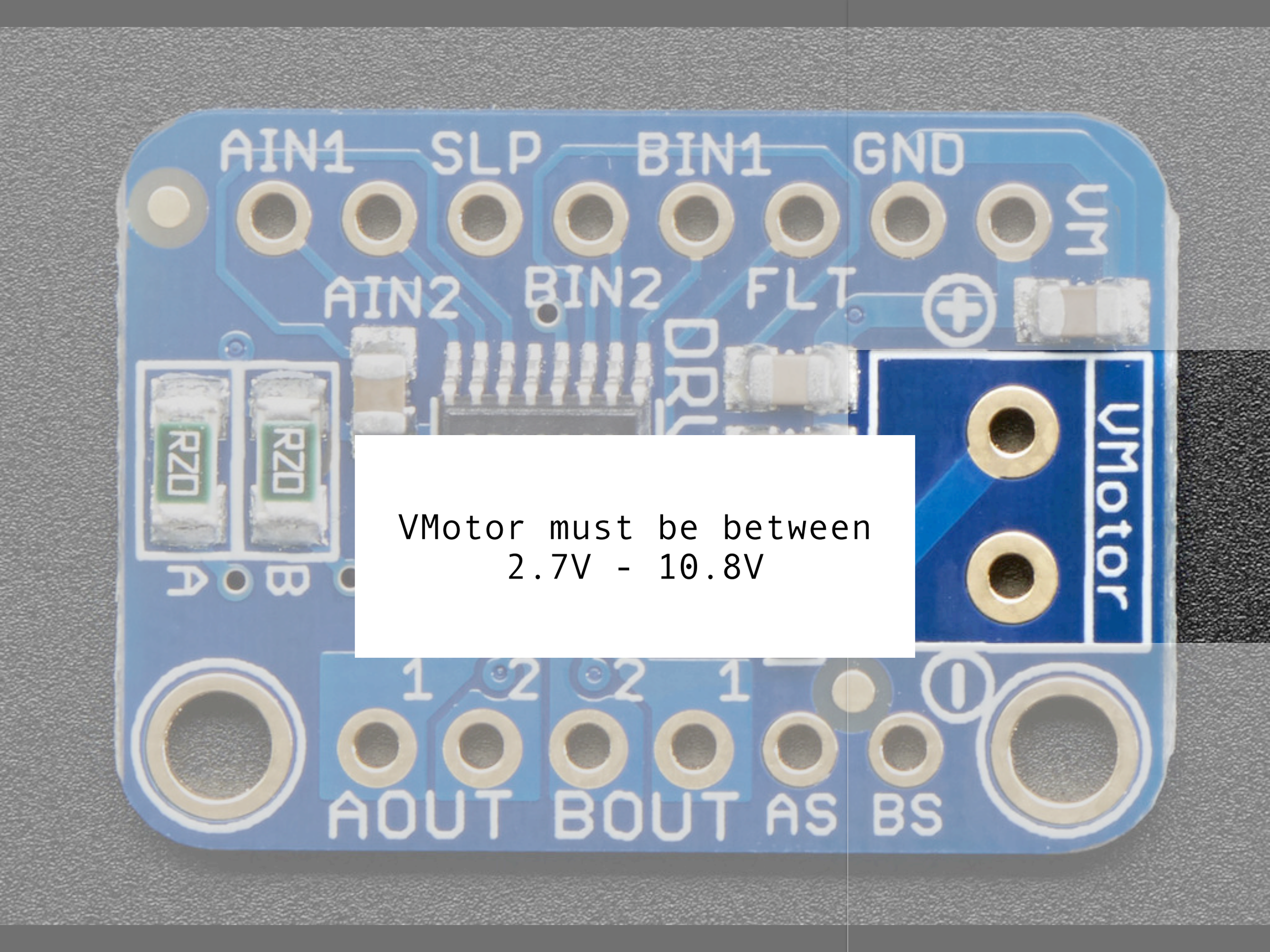

AIN1
AIN2
SLP
BIN1
BIN2
GND
VM
AOUT
BOUT
AS
BS
VMotor
DRV8833
55K
CFNC
RZD
RZD

AIN1 SLP BIN1 GND VM
AIN2 BIN2 FLT
DRV8833
55K
CFNC
R20 R20
A B
1 2 2 1
AOUT BOUT AS BS

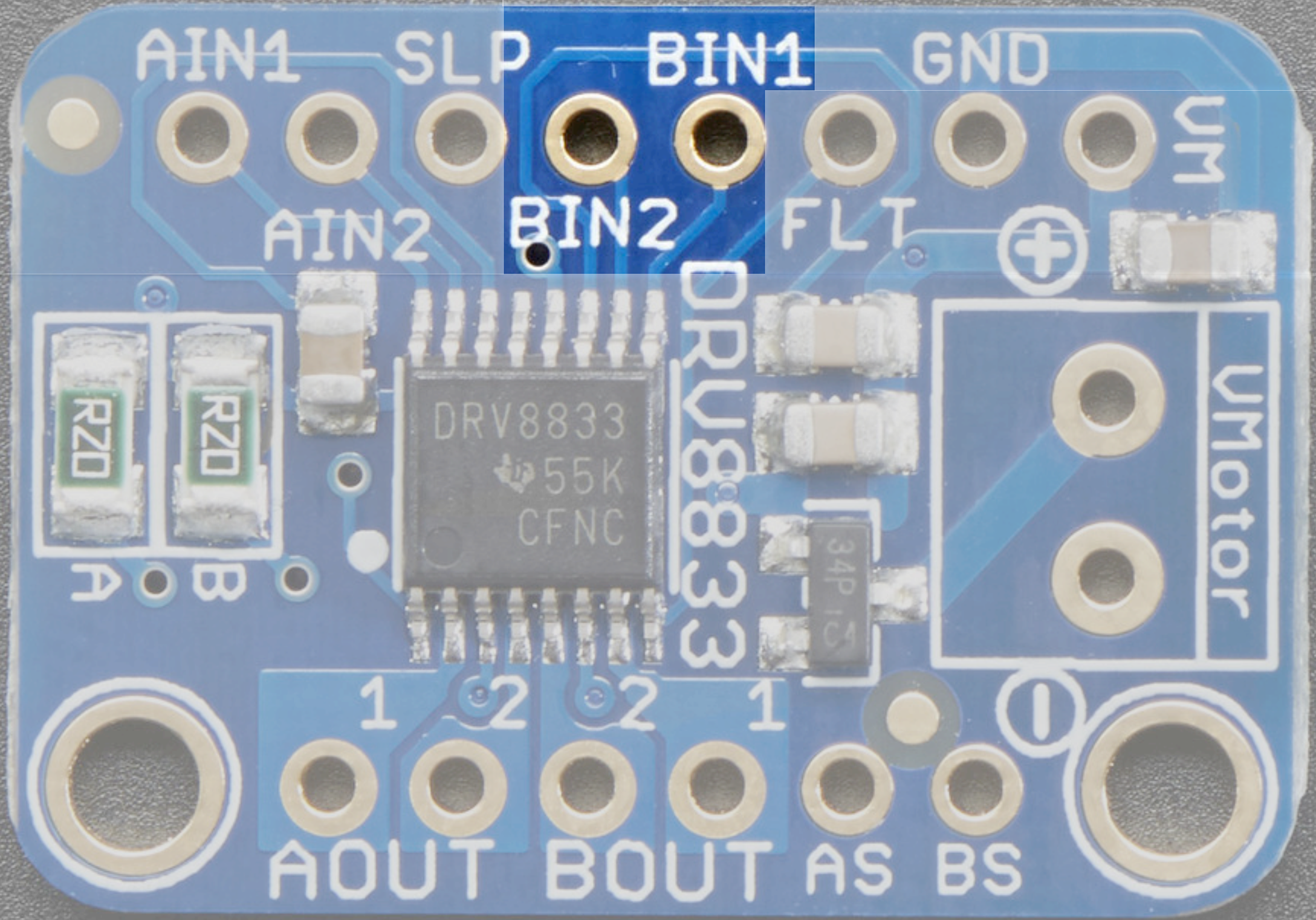
AIN1 SLP BIN1 GND VM
AIN2 BIN2 FLT
DRV8833
DRV8833
55K
CFNC
R20 R20
A B
34P
1 2 2 1
AOUT BOUT AS BS
VMMotor

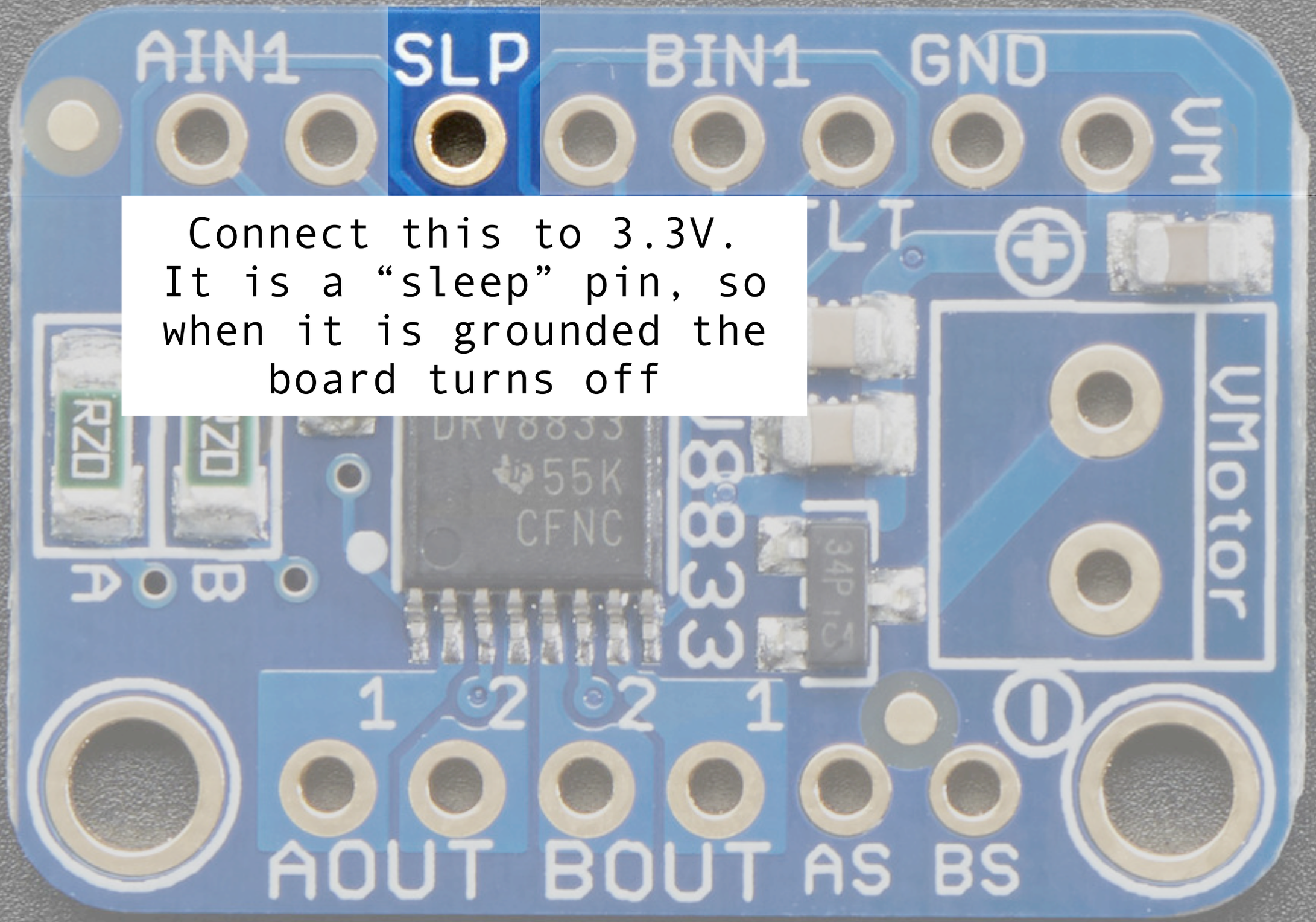
AIN1 SLP BIN1 GND VM
AIN2 BIN2 FLT
DRV8833
DRV8833
55K
CFNC
R20 R20
A B
1 2 2 1
AOUT BOUT AS BS
VMotor

AIN1 SLP BIN1 GND VM
AIN2 BIN2 FLT
DRV8833
55K
CFNC
DRV8833
34P
R20 R20
A B
1 2 2 1
AOUT BOUT AS BS
VMMotor


AIN1
SLP
BIN1
GND
VM
AIN2
BIN2
FLT
DRV
VMotor must be between
2.7V - 10.8V
A
B
1
2
2
1
AOUT
BOUT
AS
BS

AIN1
AIN2
SLP
BIN1
GND
BIN2
FLT
VM
DRV8833
55K
CFNC
DRV8833
34P
RZD
RZD
A
B
1
2
2
1
AOUT
BOUT
AS
BS
Motor

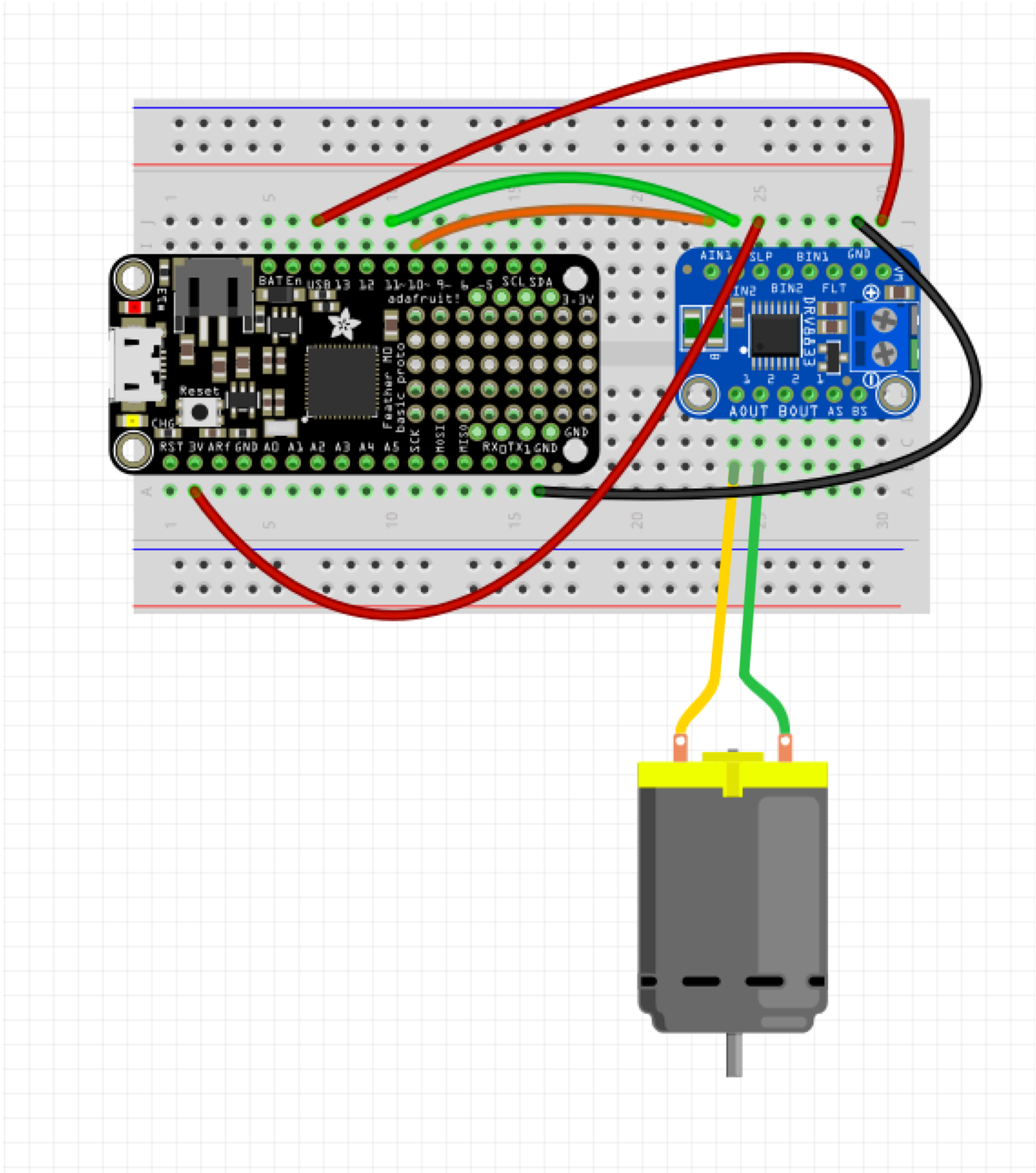

AIN1
SLP
BIN1
GND
AIN2
BIN2
FLT
VM
DRV8833
55K
CFNC
DRV8833
34P
R20
R20
A
B
1
2
2
1
VMotor
AOUT
BOUT
AS
BS



Connect this to 3.3V. It is a “sleep” pin, so when it is grounded the board turns off

IN1	IN2	RESULT
0	0	Coast
1	0	Forward
0	1	Reverse
1	1	Brake

CONTROLLING A MOTOR WITH THE DRV8833



CONTROLLING A MOTOR WITH THE DRV8833

DRV8871 | Arduino 1.8.5

DRV8871 §

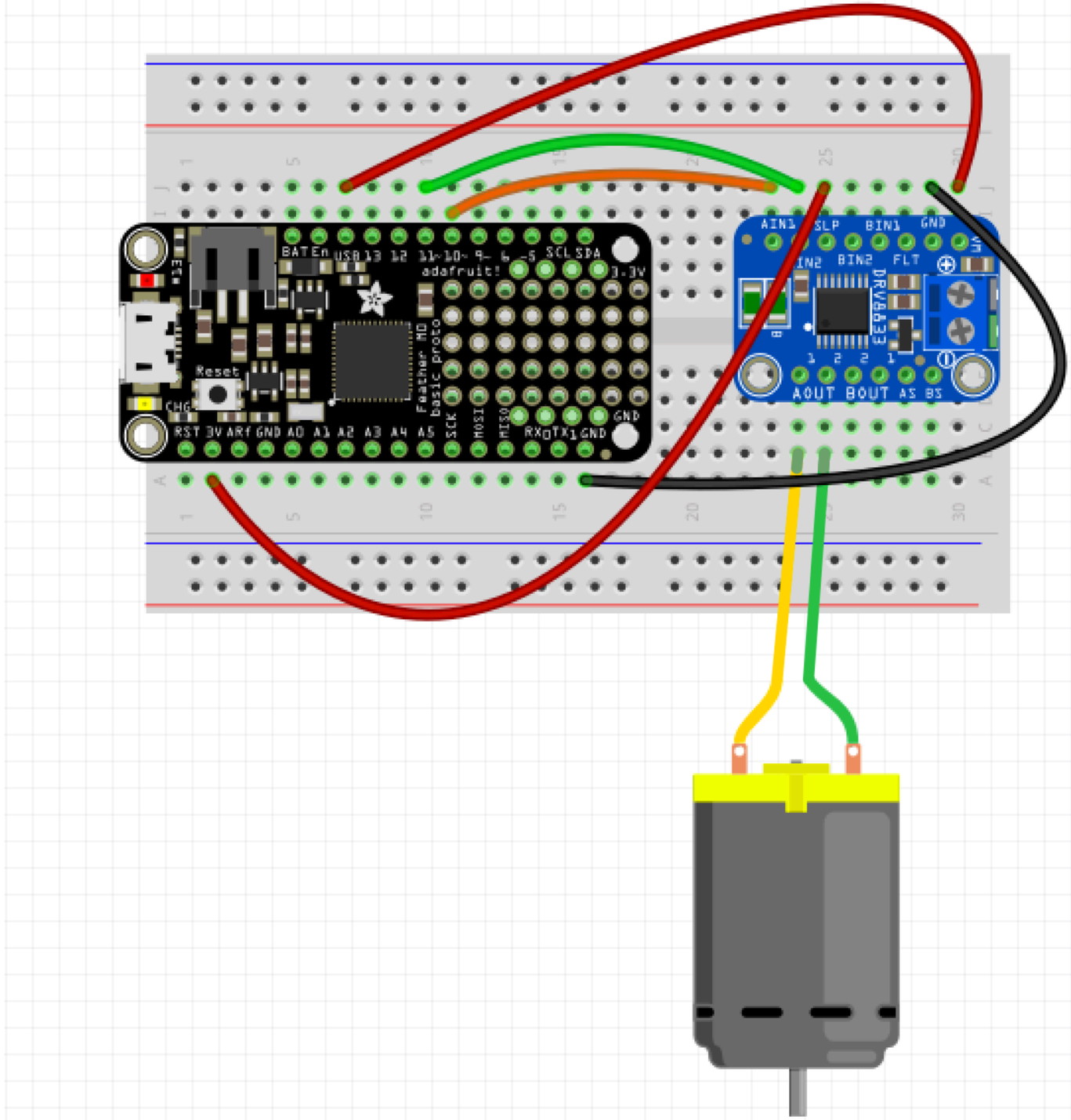
```
const int MOTOR_FORWARD_PIN = 9;
const int MOTOR_REVERSE_PIN = 10;

void setup() {
  pinMode(MOTOR_FORWARD_PIN, OUTPUT);
  pinMode(MOTOR_REVERSE_PIN, OUTPUT);
}

void loop() {
  // forward
  digitalWrite(MOTOR_FORWARD_PIN, HIGH);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
  delay(1000);

  // coast
  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
  delay(1000);

  // reverse
  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, HIGH);
  delay(1000);

  // coast
  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
  delay(1000);
}
```

Auto Format finished.

```
Global variables use 149 bytes of dynamic memory.
```

24 Adafruit Feather 32u4 on /dev/cu.usbmodem319

CONTROLLING A MOTOR WITH THE DRV8833



DRV8833 | Arduino 1.8.10

DRV8833

```
const int MOTOR_FORWARD_PIN = 11;
const int MOTOR_REVERSE_PIN = 10;

void setup() {
  pinMode(MOTOR_FORWARD_PIN, OUTPUT);
  pinMode(MOTOR_REVERSE_PIN, OUTPUT);
}

void loop() {
  digitalWrite(MOTOR_FORWARD_PIN, HIGH);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
  delay(1000);

  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
  delay(1000);

  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, HIGH);
  delay(1000);

  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
  delay(1000);
}
```

Done Saving.

CONTROLLING A MOTOR WITH THE DRV8833 (with functions)

DRV8871 | Arduino 1.8.10

DRV8871 §

```
const int MOTOR_FORWARD_PIN = 9;
const int MOTOR_REVERSE_PIN = 10;

void setup() {
  pinMode(MOTOR_FORWARD_PIN, OUTPUT);
  pinMode(MOTOR_REVERSE_PIN, OUTPUT);
}

void loop() {
  forward();
  delay(1000);

  coast();
  delay(1000);

  reverse();
  delay(1000);

  coast();
  delay(1000);
}

void forward() {
  digitalWrite(MOTOR_FORWARD_PIN, HIGH);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
}

void reverse() {
  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, HIGH);
}

void coast() {
  digitalWrite(MOTOR_FORWARD_PIN, LOW);
  digitalWrite(MOTOR_REVERSE_PIN, LOW);
}

void brake() {
  digitalWrite(MOTOR_FORWARD_PIN, HIGH);
  digitalWrite(MOTOR_REVERSE_PIN, HIGH);
}
```

41 Adafruit Feather 32u4 on /dev/cu.usbmodem14101