The attached code is for the setup I use, which I only use for cutting wire so no stripping is done. I just prefer to do that part manually.

I have 2 limit switches installed. 1 to be hit when the cutter is fully opened and the other when the cutter is fully closed. The types of switches I’m using are these:

<https://www.amazon.com/gp/product/B01MQPQGFQ/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1>

My limit switches have 5v , Ground through a 10k resistor, and a sense wire hooked to them. The 10k resistor is necessary because the 5v will be shorted to ground to provide a low signal on A0 pin when switch is triggered. If the resistor is not included this would cause a direct short to ground on the Arduino / power supply with no load (that would be bad). A different resistor value may work, but I just used a 10k ohm and it worked fine.

Example:

Diagram

Description automatically generated

In my Arduino code I assign a variable for each limit switch pin. On the Arduino uno A0 is identified as pin 14, and A1 is identified as pin 15.

In the beginning of the code these lines are for both switch setup:

int intPinA = 0; //will be used for cutter stop switch for cutting

int intPinB = 0; //will be used for cutter stop switch for opening

int openLimit = 14; //default open limit is set to 14 for A0 - will change if A1 is hit instead using defineLimit()

int closeLimit = 15; //default close limit is set to 15 for A1 - will change if A0 is hit instead using defineLimit()

pinMode(14, INPUT); //using Analog 0 as digital I/O for cutter open stop switch

pinMode(15, INPUT); //using Analog 1 as digital I/O for cutter close stop switch

Explanation:

Int intPinA=0; -This line just creates an integer variable named intPinA with a value of 0. The same is true for the intPinB line. This will be used later to determine the switch state. 0 = switch open , 1 = switch closed.

Int openLimit=14; -This line creates an integer variable named openLimit with a value of 14. This is used to identify my openLimit switch hooked to pin 14 on the Arduino.

Further down in the code, the switches are read / used in the functions ‘closeCutter()’ ‘openCutter()’ ‘stripCutter()’.

Example:

digitalWrite(enablePinB,LOW); //enable driveB

digitalWrite(dirPinB,LOW); //will be turning CCW

for(int x = 0; x < 5000; x++) { //putting in stupid high number for steps, as we will utilize switch for stop

intPinB = digitalRead(openLimit); //get A0 state LOW/HIGH

if(intPinB == 1) { //switch has not been hit so run

digitalWrite(stepPinB,HIGH);

delayMicroseconds(500);

digitalWrite(stepPinB,LOW);

delayMicroseconds(500);

}else { //switch has been hit, move back to reset switch

digitalWrite(dirPinB,HIGH); //reverse to CW direction to move off of switch

delay(500);

while(intPinB != 1) { //move until switch goes back to open

intPinB = digitalRead(openLimit); //re-read A0 state for change

digitalWrite(stepPinB,HIGH);

delayMicroseconds(800);

digitalWrite(stepPinB,LOW);

delayMicroseconds(800);

}

break; //exit loop to stop motor from cont to try and move

}

}

digitalWrite(enablePinB,HIGH); //disable driveB

delay(800);

}

Breakdown of each line:

digitalWrite(enablePinB,LOW); -- this simply sets my enable pin to low on the stepper driver. This basically turns the stepper driver ON and should make it so the stepper can be controlled.

digitalWrite(dirPinB,LOW); – Sets the enable pin to low on the stepper driver. Low = turn counter clockwise, High = turn clockwise.

for(int x = 0; x < 5000; x++) { -- This is a loop that will run 5000 times. Used to pulse the stepper 5000 times for movement. Stepper should not actually step 5000 times because it will hit a limit switch and stop (seen here a few lines below)

intPinB = digitalRead(openLimit); --This is where I am actually reading the switch state. As the ‘openLimit’ variable was set to ‘14’ in the beginning of the code, we are setting the intPinB variable to be what state pin 14 is in (1 or 0). If the switch has not been triggered this state should be 1, if the switch has been triggered this state should be 0.

if(intPinB == 1) { --if we read a 1 in the above line, we will continue to step the motor 1 step.

digitalWrite(stepPinB,HIGH);

delayMicroseconds(500);

digitalWrite(stepPinB,LOW);

delayMicroseconds(500);

}else { --if we read a 0 in the above line, we will back the stepper up until the switch reads a 1 indicating the switch is not triggered.

--The below lines are in a loop until the motor turns clockwise and the switch is no longer triggered.

digitalWrite(dirPinB,HIGH); //reverse to CW direction to move off of switch

delay(500);

while(intPinB != 1) { //move until switch goes back to open

intPinB = digitalRead(openLimit); //re-read A0 state for change

digitalWrite(stepPinB,HIGH);

delayMicroseconds(800);

digitalWrite(stepPinB,LOW);

delayMicroseconds(800);

}

break; --now we simply break the operation, stopping the initial for loop where the stepper would run up to 5000 steps.

}