Miga Motor Company
1250 Addison Street #208
Berkeley, CA 94702
www.migamotors.com

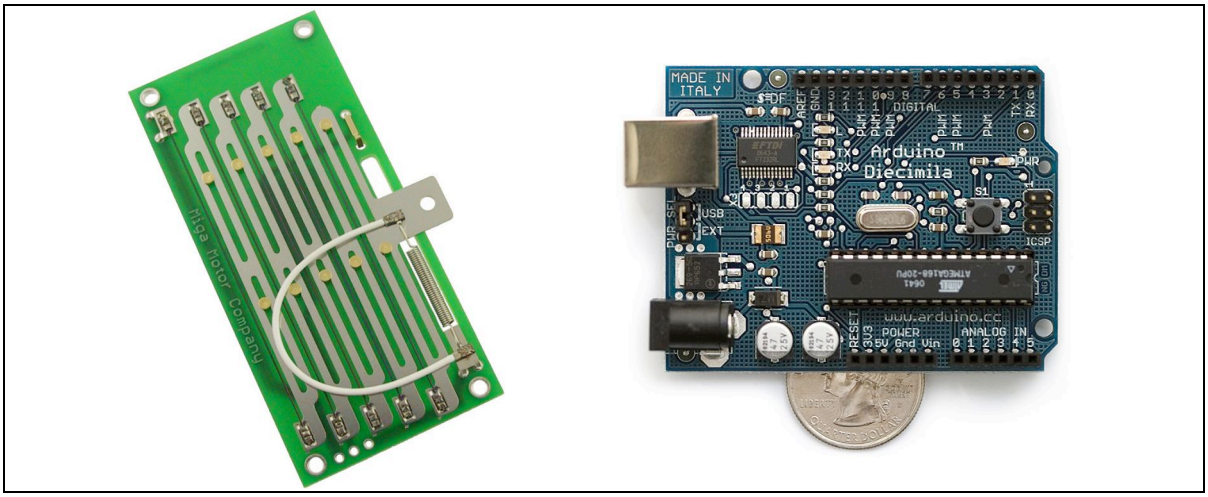
MigaOne-Arduino Tutorial:
Position/Limit Sensing and Cycling with the Arduino Platform

1.0 Introduction

The purpose of this tutorial is to provide a simple introduction to some of the techniques used to control the actuation of the Miga Motor Company MigaOne™ actuator. Using a microcontroller and some basic analog circuitry, position and limit sensing can be used to reliably cycle the MigaOne.

The MigaOne is a linear actuator based on proven shape-memory alloy (SMA) technology. It provides roughly 0.375'' of stroke and 2.5 lbs of force. The actuator can be powered with voltages of 5-30V. For more detailed information, see the MigaOne Application Notes on the "Downloads" page online at: www.migamotors.com.

For sensing and control, an Arduino Diecimila platform (SparkFun SKU#: DEV-00666) is used. The Arduino is a popular open-source electronics platform admired for its easy-to-use hardware and software package along with its affordability (a definite draw for hobbyists and tinkerers). The Diecimila is based on the ATmega168 chip, providing powerful I/O capabilities useful for physical computing. For more details, see the Arduino's homepage at www.arduino.cc or SparkFun at: www.sparkfun.com



2.0 Setup

In order to reliably cycle the MigaOne, the position of the output shaft must be (1) sensed and (2) controlled. Section 2.0 provides an outline for the hardware and software setup needed to achieve repeated actuation.

2.1 Hardware

All of the following circuits can be assembled in a matter of minutes with basic electrical prototyping tools (breadboard, wire, soldering iron, power supply, etc). A basic understanding of electronics and circuit-building techniques is assumed.

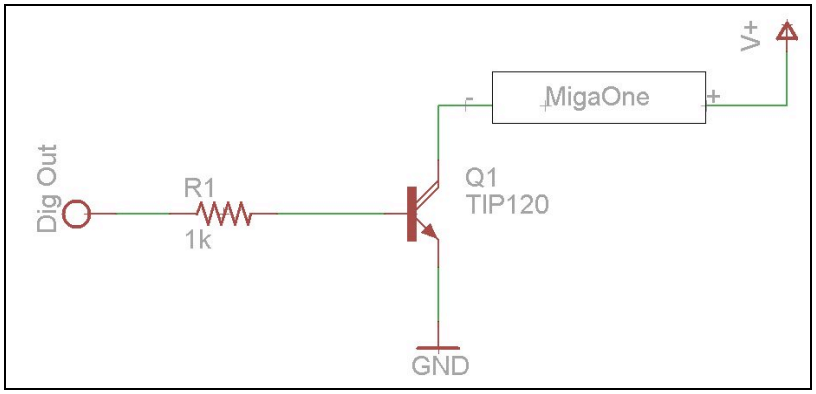
2.1.1 Parts List

(1) Miga Motor Company MigaOne actuator
(2) Arduino Diecimila development platform
(3) Power MOSFET – TIP120 or similar
(4) 10k linear potentiometer (Alps RDC10 series) OR (1) 47kΩ resistor
(5) 1kΩ resistor

Basic electrical prototyping tools (as suggested above)

2.1.2 Driver Circuit

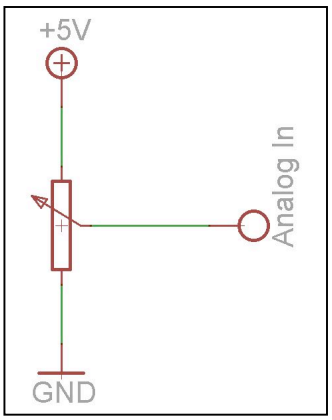
The driver circuit uses a power MOSFET and a current limiting resistor to switch the power to the MigaOne with a logic signal from the Arduino controller. The circuit schematic is shown below:



When the digital signal is high, current is allowed to flow to ground through the MOSFET, actuating the MigaOne. When the digital signal is low, no current flows and the MigaOne is effectively "off". This setup basically allows a high-current circuit to be switched with a low current, logic-level signal provided by the Arduino.

2.1.3 Position Sensing

The position of the output shaft is sensed with a linear potentiometer. The potentiometer acts as a dynamic voltage divider, seen in the following schematic:

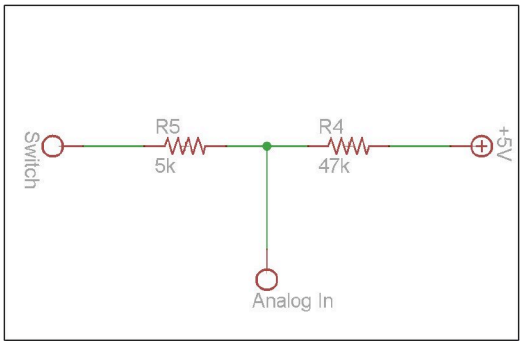


The Arduino supplies +5V. As the potentiometer slider translates during actuation, the output voltage changes due to changing resistance. This output voltage is monitored

with one of the Arduino's ADC inputs, returning a value between 0 and 1023 depending on the observed voltage level. With the output shaft securely coupled to the linear potentiometer, reliable position feedback can be achieved.

2.1.4 Limit Sensing

Since position feedback during the entire stroke is not necessary for actuator cycling, an alternative scheme can be used to sense when the MigaOne has reached the end of a full stroke. This method uses the limit switch built into the MigaOne. The following circuit is used:



This is a simple "pull-up resistor" scheme. Before the actuator has reached the end of stroke, the switch node is floating and the analog input is "pulled-up" to +5V. When the output shaft comes in contact with the limit switch, the switch node is pulled to roughly one-sixth of the voltage supplied to the actuator*. This change in voltage can be read by the analog input, which converts the analog signal to a value between 0 and 1023 (0 for 0V, 1023 for +5V). A 5k resistor is included to limit the current. It is recommended that the voltage supplied to the actuator be no greater than +18V if this limit-sensing scheme is used, due to increased actuator speed and more inconsistent limit sensing.

**Note: This value comes from the fact that the output shaft is not exactly at ground, due to resistance in the jumper wire and PCB copper tracing. Ideally, the output shaft would be at ground and a simpler digital input (HIGH or LOW sensing) could be used to detect the end of stroke.*

2.2 Programming the Arduino

After the basic circuits have been assembled, the Arduino platform can be programmed to cycle the MigaOne actuator repeatedly and reliably. The microprocessor on the Arduino is programmed using its own language and development environment. Even if the user has very little or no programming experience, it is very easy to write and understand programs for the Arduino. See www.arduino.cc for full documentation.

2.2.1 What you need

PC or laptop with a USB port
USB cable (A-type to B-type connectors)
Installed Arduino development environment (see website)
Arduino board with installed USB drivers

2.2.2 Cycling with Position Sensing

The following code sample is designed for use with the linear pot:

```
//--------------MigaOne Cycling-------------------------------
//Cycles the MigaOne, using analog position readings from pot
//Supply desired voltage (suggested 12V)
//Outputs number of cycles, time to reach end of stroke

//Initial variable declarations
int motorPin = 8;         //motor control
int potPin = 2;           //analog input (wiper)
int val = 0;              //serial reading
int potBuffer = 10;       //pot reading buffer
int reps = 0;             //cycles completed
long startTime = 0;       //starting time value
long time = 0;            //current time
int dist = 150;           //desired travel distance (calibrated)
int initPot = 0;          //initial pot value
int thresh = 0;           //pot value threshold

void setup() {
  //Pin setup
  pinMode(motorPin,OUTPUT);
  pinMode(potPin,INPUT);

  //Begin serial communication
  Serial.begin(9600);
}

void loop() {

  //Initialization
  initPot = analogRead(potPin);
  thresh = initPot + dist;
  startTime = millis();

  //Power actuator
  digitalWrite(motorPin,HIGH);

  //Wait until threshold is reached
  while (val < thresh) {
    val = analogRead(potPin);
    time = millis() - startTime;
  }

  //Stop acuation
  digitalWrite(motorPin,LOW);

  //Wait until actuator reaches initial position (+ buffer)
  while (val > (initPot + potBuffer)) {
    val = analogRead(potPin);
  }

  //Increment cycles
  reps = reps + 1;

  //Print reps completed and actuation time
  Serial.print(reps);Serial.print(" ");Serial.println(time);

}
```

The code does the following:

1) Determine the initial position of the output shaft and calculate how far the output shaft should travel based on the desired stroke
2) Actuate the MigaOne
3) Wait until limit is reached and record actuation time
4) Cut power
5) Wait until the output shaft returns to initial position (plus desired buffer)
6) Record cycle number
7) Print cycles completed and actuation time in serial console
8) Repeat

The '*dist'* variable will need to be calibrated for each setup, and will depend on the particular potentiometer used in the experiment. Keep in mind that the analog-to-digital converter (ADC) on the Arduino will provide a value between 0 and 1023 for a range of 0 to 5V, or the full travel of the linear potentiometer.

One way to calibrate the desired travel distance is to start small (say, at 50 or 100) and slowly increase the value until the correct stroke is reached. Be sure to select a value that lies within the possible stroke of the actuator, or the actuator might never reach an end point and overheat!

The '*dist'* variable may also be calibrated by experimenting with the linear potentiometer alone. Use the following code snippet in a separate Arduino sketch (or comment out main code) to get an idea of how the ADC readings vary with slider position:

```
//Pot calibration
//Move slider and watch value in serial console
val = analogRead(potPin);
Serial.println(val);
delay(250);
```

Also be sure that the ADC value *increases* in the direction of actuation. The code above can be used as a check.

2.2.3 Cycling with Limit Sensing

The following code sample is designed for use with the limit-sensing scheme:

```
//--------MigaOne Cycling--------
//Cycles MigaOne using limit sensing
//Supply desired voltage (+12V suggested, <18V)
//Outputs number of cycles completed

//Initial variable declarations
int motorPin = 8;     //motor control pin
int adcPin = 0;       //analog in pin
int val = 0;          //analog in reading
int limitVal = 900;   //limit threshold (calibrated)
int reps = 0;         //number of cycles completed

void setup() {
//Pin setup
pinMode(motorPin,OUTPUT);
pinMode(adcPin,INPUT);

  //Begin serial communication
Serial.begin(9600);
}

void loop() {

  //Read analog pin
 val = analogRead(adcPin);

//Test to see if limit has been reached
  if (val < limitVal) {
    digitalWrite(motorPin,LOW);
    reps = reps + 1;
    Serial.println(reps);
    delay(10000);
  }

  else {
    digitalWrite(motorPin,HIGH);
  }

}
```

The above code does the following:

1) Read the analog input pin
2) If limit hasn't been reached → actuate MigaOne
3) If limit has been reached → cut power, increment and print cycle number, delay 10 seconds to let output shaft return to starting position
4) Repeat

As a reminder, this scheme should not be used with voltages above 18V.

3.0 Additional Experiments
Using the MigaOne along with the Arduino, the possibilities for experimentation are endless. The following list presents some experiment ideas and useful control techniques:

- Actuate the MigaOne and hold at a desired position
- Use PWM control to modulate actuation speed
- Do stroke profile characterization, plotting output shaft position versus elapsed time
- Control the velocity profile of the MigaOne: actuating at high speed to the midpoint, then slowing down to the end of stroke, for example
- Life or cycle testing
- Control multiple MigaOne actuators at once

As always, be sure to follow the operation guidelines presented in the MigaOne Application Notes, available at www.migamotors.com/Downloads.html.

4.0 Summary
This tutorial has provided a basic overview for controlling the MigaOne linear actuator with the Arduino development platform. More specifically, methods for controlling and sensing the position of the output shaft for safe and reliable cycling have been shown.

Happy Actuating!
