peter's tinkering

Sunday, June 21, 2015

Arduino 4-20 mA output

Need an Arduino to output a standard 4 to 20 milliamp signal for a control loop? Here's how!

*Note: Looking for how to **input** a 4-20 milliamp signal to an Arduino? This link here is what you're looking for.*

At work, we had a surplus Fisher D4 control valve with an i2P transducer that we weren't sure was functional. Of course, we could have had one of our instrument techs test it, but where's the fun in that? After work I whipped up a circuit powered by an Arduino that would do the trick.

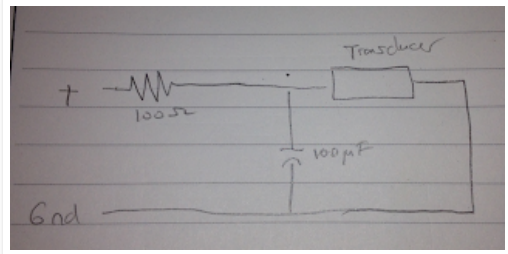
According to the transducer's documentation, the equivalent circuit is a 40 ohm resistance with a 4 volt drop, and the current supply should be able to supply at least 30 milliamps and have a maximum voltage of 30V. Doing the math,

$$V = IR$$

$$V = 0.03\ A * 40\ ohms = 1.2\ V$$

Adding the 4 volt constant drop, we'll need at least 5.2 volts available across the transducer. Since we'll have some additional resistive loss in the circuit, we'll need to use a transistor to switch the Arduino's 5V output to a higher voltage.

Now how do we get the Arduino to apply a specific voltage across the transducer to give it the desired current? Some of the Arduino digital pins can do pulse width modulation that applies 5V to the circuit for a time and then zero volts for a time, but we want a smooth voltage output to give a smooth current, not bang-bang control. We can use an RC circuit to transform the on-off signal into a smooth output. This site here has a nice writeup of how to use an RC circuit with an Arduino to output a given voltage. It links to a RC-PWM simulator here that you can use to estimate the values of a resistor and capacitor needed for a reasonably smooth output. I had a 100 microfarad capacitor on hand and paired with a 100 ohm resistor, it looked good enough.



So now we need some way of making sure we know how much current we're putting through the circuit. If we can measure the current in the circuit, we can adjust the PWM output to the circuit until we get the desired current. We can accomplish this by putting a resistor of a known resistance in the circuit and measure the voltage drop across it. The Arduino can read a 0-5V voltage with its analog input pins. The bigger the resistor we use, the larger the voltage drop we can read and the better the resolution we'll get on measuring the current. At the maximum of 20 mA we want it to read, the biggest resistor we can use is:

About Me

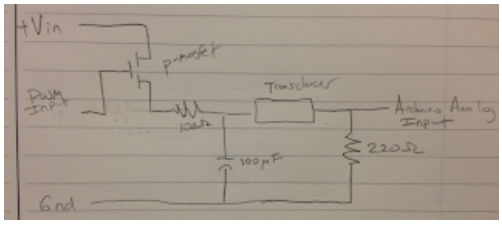
Peter

View my complete profile

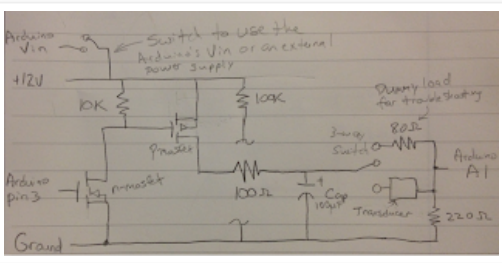
$$R = V/I$$

$$R = 5\ V/\ 0.02\ A = 250\ ohms$$

220 ohm was the closest I had on hand. You'll want to measure the actual resistance to use in the code - the resistor I had was actually 217 ohms, so that's what I used in the code. This resistor does need to be the last element in the circuit above ground so that the voltage you measure at the high end is the true voltage drop across the resistor. Measuring the current like this does mean that we'll need to switch the circuit with a transistor on the high side. A p-mosfet like this guy will be appropriate.



Since the source voltage will be quite a bit higher than 5V, we'll need an n-mosfet to apply the full source voltage to the gate when switching. It in turn can be switched by one of the Arduino's PWM pins. Here's my full circuit, including a dummy load for troubleshooting and some resistors to keep the mosfets happy (I'll be honest, I can't remember why I included the 100K one).



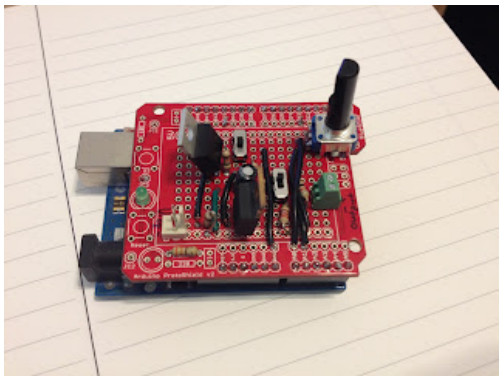
With this full circuit and the transistor fully open, the resistance across the circuit is 100 + 40 (transducer itself) + 220 = 360 ohms. At a full 20 mA, the voltage drop across the resistive elements here will be:

$$V = 0.02 * 360 = 7.2\ volts$$

Add in the constant 4 volt drop across the transducer, and we'll need a power supply that outputs at least 11.2 volts in order to power the circuit through its full 4 to 20 mA range. A 12V power supply should work swimmingly.

I controlled the current-supply circuit with pin 3, and an indicator LED with pin 11. I read the reference resistor's voltage with analog pin 1. I included a potentiometer to adjust the current set point.

In order to adjust the PWM output to generate the desired current, I used the Arduino PID library. It's probably overkill, but it does the trick. It also easily compensates for changes in the transducer load resistance or power supply input. After getting it to work on a breadboard, I soldered it all up:



We tested it out on a valve, and sure enough it worked! Move the potentiometer and the stem moved up and down as the transducer received the signal and applied the proper pressure to the valve's topworks.

There's probably far more elegant ways to accomplish the same thing, but this did do the trick for me.

Here's the code:

```
//This sketch works to output a 4 to 20 milliamp signal by measuring the resistan
#include <PID_v1.h> //PID library
#define TRANSISTOR 3 //n-Mosfet digital pin
#define LED 11 //LED digital pin
#define POTPIN 0  // Analog input pin for the potentiometer
#define REFRES 1 //referece resistor analog input pin

float val = 0;    // variable to store the value coming from the potentiometer

char buffer[11];

int Rref = 217; //resistance of reference resistor

float Vref = 0; //measured voltage across reference resistor

float MeasCurrent = 0;

int counter = 0;
double Setpoint, Input, Output;

//Specify the links and initial tuning parameters for the PID loop. I didn't use
double Kp=1, Ki=30, Kd=.02;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //initialize the PID

void setup(){
  pinMode(TRANSISTOR, OUTPUT);
  pinMode(LED, OUTPUT);
  Serial.begin(9600);

  //Arbitrarily start the PID at 12 mA
  Setpoint = 12;

  //turn the PID on
  myPID.SetMode(AUTOMATIC);

  //Header for debugger
  Serial.println("Time (ms), Setpoint (mA),PID output,Measured voltage across ref
}
```

```
void loop(){

 //take ten measurements of the reference resistor and potentiometer and average
  counter = 0;
  val = 0;
  Vref = 0;
  while(counter < 10){
   counter = counter + 1;
   val = val + analogRead(POTPIN);  // read the value from the potentiometer
   Vref = Vref + analogRead(REFRES); // read the value from the reference resisto
   delay(1);
  }
  val = val/10;
  Vref = Vref/10;

  //Convert the measured potentiometer setting to a 4 to 20 mA set point for the
  Setpoint = mapfloat(val, 0.0, 1023.0, 4.0, 20.0);

  //Calculate the current in milliamps from Ohm's law. The 0-1024 analog reading
  MeasCurrent = Vref/1024.0/Rref*1000.0*5.0;

  //Feed the PID loop with the measured current.
  Input = MeasCurrent;
  //print out everything for the debugger. Time, PID setpoint, PID output, measur
  Serial.print(millis());
  Serial.print(",");
  Serial.print(Setpoint);
  Serial.print(",");
  Serial.print(Output);
  Serial.print(",");
  Serial.print(Vref*5/1024.0);
  Serial.print(",");
  Serial.println(MeasCurrent);

  //Run the PID
  myPID.Compute();

  //Take the PID output and apply it to the transistor pin
  analogWrite(TRANSISTOR, Output);

  // Adjust the brightness of the LED to the setpoint with PWM
  analogWrite(LED, mapfloat(Setpoint, 4.0, 20.0, 0, 255));

  //wait 20 milliseconds before looping again
   delay(20);

}

//The normal map function uses longs, we need floats - from http://forum.arduino.
float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
 return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Posted by Peter at 8:57 AM

3 comments:

Unknown October 3, 2016 at 3:13 PM

Thanks for posting this.

I noticed you have a model number for the P-channel MOSFET but not the N-channel. What do you recommend?

Reply

Replies

Peter October 5, 2016 at 5:39 PM

I used a FQP27P06, this one here - https://www.sparkfun.com/products/10349

It's wildly overkill since it's only controlling a few milliamps of current, but it does the trick for ninety five cents.

Peter October 6, 2016 at 3:29 PM

Whoops, you asked about the N-channel. That's an IRF510. Again, overkill, but it works fine.

Reply

To leave a comment, click the button below to sign in with Blogger.

SIGN IN WITH BLOGGER

Newer Post Home

Subscribe to: Post Comments (Atom)