ECE480 Design Team 3

Using Arduino Microcontrollers to Sense DC Motor Speed and Position

Tom Manner

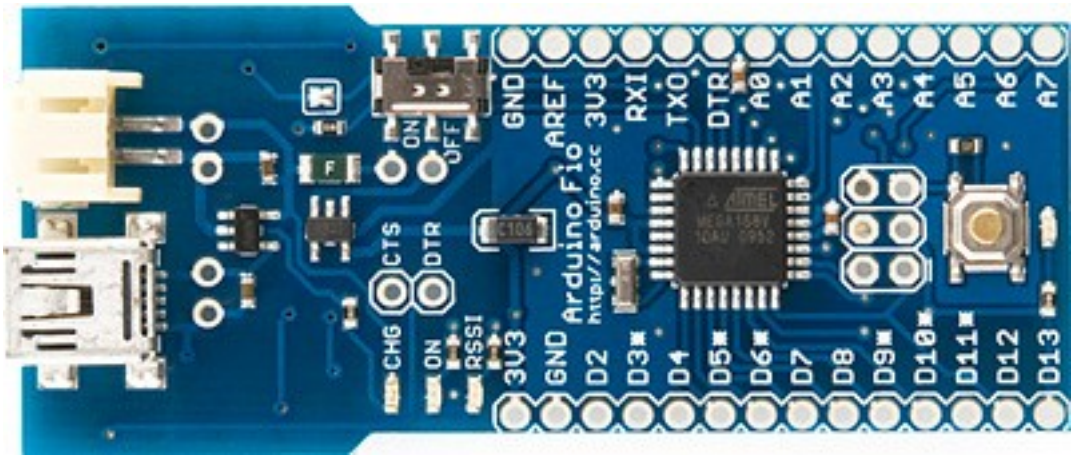
April 4, 2011

Table of Contents

1 Introduction

Brushless DC motors are a very attractive option for use in systems that require small high torque motors, but they do not provide any feedback. To solve this problem a microcontroller with at least one interrupt pin can be configured to receive feedback from an optical sensor attached to the motor. The Arduino development platform uses ATmega168 and ATmega328 processors, this note will focus on the ATmega328. The ATmega328 has an 8MHz core clock speed, 32KB of flash memory on board and in most Arduino products it is configured to have 8 10-bit analog inputs, and 14 digital input output pins, two of which can be attached to interrupts.

Figure 1 – Photo of the Arduino Fio, which is being used to interpret motor feedback. Pins labeled A_ are analog inputs, D_ are digital input output pins, the ones marked with an asterisk support pulse width modulated output. 3V3 outputs a constant 3.3Volts, GND the relative ground and AREF can be used to set an external reference voltage for the analog inputs.



2 Rotation Sensors

A photo interrupter can be used to detect rotation by placing a gear with spokes on the drive shaft of the motor so that each spoke will interrupt the beam. A photo interrupter in its most basic form is an led placed near a phototransistor so that when an object comes in between the led and the transistor the source to drain voltage drops. The photo interrupter used here is the Sharp Microelectronics GP1A73AJ000F. It has three pins, Vcc, Vout, and Gnd, with Vcc connected to the system's voltage source and Gnd connected to ground, Vout is near Gnd when the beam is uninterrupted, and near Vcc when the beam is interrupted. This results in a square wave output with a frequency that is proportional to the speed of the motor and the number of spokes on the gear. By routing this signal into one of the interrupt pins on the Arduino, the number of rotations can be calculated and stored in a persistent variable. The following is a simple code example that sets up a pin for interrupts and updates a variable.

```
//Simple interrupts
void setup()
{
	//attachInterrupt(<interruptPin>,<interruptServiceRoutine>,<trigger>);
	// interruptPin is 0 or 1, for digital pin 2 or 3, interruptServiceRoutine is the function to run
	//	trigger is what will cause the ISR to run, can be LOW, RISING, FALLING or CHANGE

	attachInterrupt(0,updatePosition,FALLING);

	//Setup pin 13 as an output, to flash the built in led

	pinMode(13,OUTPUT);
}
volatile int position =  0;
volatile boolean direction = false;

//The main program loop that runs while the board is active
//	This loop flashes the pin 13 led at 1Hz, each delay is 500ms
void loop()
{
	digitalWrite(13,HIGH);
	delay(500);
	digitalWrite(13,LOW);
	delay(500);
}
void updatePosition()
{
	if(direction = true)	//moving in the positive direction, increment position
		{position++}
	else			//moving in the negative direction, decrement position
		{position--;}
}
```

Direction in the above code is being tracked based on a variable, which would be generated by the Arduino when it sent a signal to a motor to move it. To make the system more robust a second photo interrupter can be added such that the spokes will interrupt both at once. When one is interrupted, the ISR will run, it can test a digital input pin connected to the second sensor. Its value at the time of the interrupt will be dependent on the direction the motor is spinning. The following is code that has been updated to detect the motion based on sensor input. This code will detect motor direction without having to infer anything based on the current output configuration. This means even external forces acting on the motor shaft will be properly interpreted by the controller.

```
//Direction Detection
void setup()
{
	//attachInterrupt(<interruptPin>,<interruptServiceRoutine>,<trigger>);
	// interruptPin is 0 or 1, for digital pin 2 or 3, interruptServiceRoutine is the function to run
	//	trigger is what will cause the ISR to run, can be LOW, RISING, FALLING or CHANGE

	attachInterrupt(0,updatePosition,FALLING);

	//Setup pin 13 as an output, to flash the built in led
	pinMode(13,OUTPUT);

	//Setup pin 13 as an input, to read the second sensor for input
	pinMode(4,INPUT);
}
volatile int position =  0;
volatile boolean direction = false;

//The main program loop that runs while the board is active
//	This loop flashes the pin 13 led at 1Hz, each delay is 500ms
void loop()
{
	digitalWrite(13,HIGH);
	delay(500);
	digitalWrite(13,LOW);
	delay(500);
}
void updatePosition()
{
	if(digitalRead(4) == HIGH)  //moving in the positive direction, increment position
		{position++}
	else                        //moving in the negative direction, decrement position
		{position--;}
}
```

3 Speed Detection

In order to detect the speed of the motor and compensate for it time must be taken into account. If the time is recorded at each pulse from the photo interrupter and compared to the time of the previous pulse, the motor rpm can be calculated.

```
//Direction and Speed Detection
void setup()
{       //attachInterrupt(<interruptPin>,<interruptServiceRoutine>,<trigger>);
        // interruptPin is 0 or 1, for digital pin 2 or 3, interruptServiceRoutine is the function to run
        //    trigger is what will cause the ISR to run, can be LOW, RISING, FALLING or CHANGE

        attachInterrupt(0,updatePosition,FALLING);

        //Setup pin 13 as an output, to flash the built in led
        pinMode(13,OUTPUT);

        //Setup pin 13 as an input, to read the second sensor for input
        pinMode(4,INPUT);
}
volatile int position =  0;
volatile boolean direction = false;
volatile unsigned long time = 0;
volatile unsigned int dTime;
volatile int rpm;
const int spokes = 2;

//The main program loop that runs while the board is active
//    This loop flashes the pin 13 led at 1Hz, each delay is 500ms
void loop()
{
        digitalWrite(13,HIGH);
        delay(500);
        digitalWrite(13,LOW);
        delay(500);
}
void updatePosition()
{
        if(digitalRead(4) == HIGH)   //moving in the positive direction, increment position
                {position++}
        else                                 //moving in the negative direction, decrement position
                {position--;}
        //millis() returns the time in milliseconds since the board was turned on
        dTime = millis()-time;
        time = time + dTime;
        //use the time calculation and the number of spokes to calculate the rpm of the motor
        rpm = 1/spokes * 1/dTime * 1000 * 60
}
```

4 Conclusion

Configuration of an Arduino microcontroller to determine motor speed and direction is a fairly simple process. By using two photo interrupters on a motor and one interrupt line on the controller both values can be found reliably.

Figure 2 – The internal circuitry for the Sharp Microelectronics photo interrupter. With a pull up resistor between Vout and Vcc the circuit operates as described above.