CSE 4403
Project W2011
Fuzzy Logic Controlled Robotic Gripper

By Thomas Young

Table of Contents

List of Figures

APPENDIX A

APPENDIX B

1. Introduction

The goal of this project is to implement a fuzzy logic controlled robotic gripper for the purpose of grasping and moving various objects types. The design of the system consists of various hardware and software components. Among the hardware components are, a microcontroller, a force sensor, a slip sensor, and a robotic gripper. The software components consist of microcontroller code implementing functions such as serial communications, sensor input, fuzzy logic code, and actuator control. The design of the system includes sensors such that the gripper can grasp an object and pick it up without slipping. Since the gripper is very powerful, there is a danger of crushing and destroying the object, hence fuzzy logic is implemented in order to generate sufficient but not excessive force,

2. Components

1. Atmega644p Microcontroller

The microcontroller utilized is an Atmel Atmega44P chosen for its many GPIO (general purpose input/output) pins, 2 serial ports, and ADC (analog to digital) input.

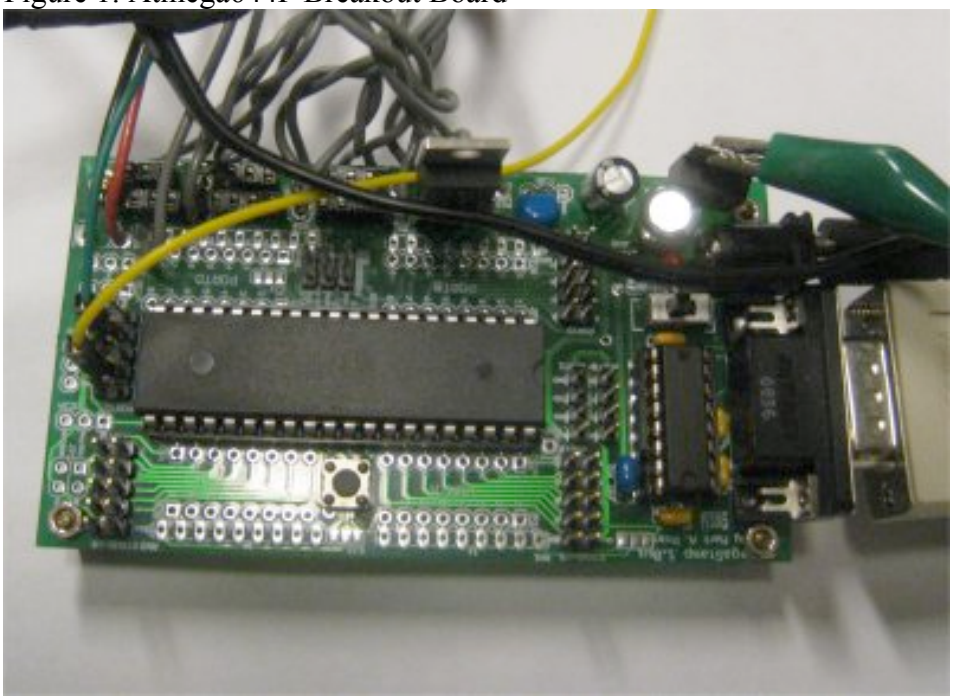
The GPIO pins drive the TLE-5205 H-bridge motor controllers. The TTL data signals control the direction of current flow in the H-bridges which in turn control the direction of motor rotation. The applied motor voltage is +12 and -12V. The voltage and hence speed of the motor is controlled by varying the duration of the TTL pulses rather than implementing PWM (pulse width modulation). PWM is the traditional method of generating analog voltage levels from digital outputs by varying the duty cycle of the pulse train.

The force sensor is connected to an ADC input pin 1 on PORT A of the Atmega644p and generates a 1 byte value representing the voltage. Thus the resolution of the analog signal will be quantized as 5V/255 = 20 mV steps. The ADC samples the analog input voltage at a rate of roughly 125 kHz.

Both serial ports, UART 1 and UART 2 are utilized to communicate with the host computer and the slip sensor respectively.

The microcontroller is mounted on a custom designed general purpose breakout board as shown below. All four ports A, B, C, D are broken out to pin headers as well as UART 2. UART 1 is connected to a MAX232 RS232 level shifter and through a DE9 connector to the host computer over a serial cable.

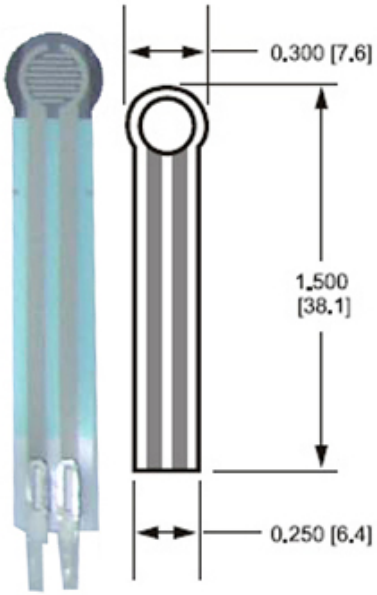
Figure 1: Atmega644P Breakout Board



2. Force Sensor

The Interlink Electronics FSR 400 acts as a potentiometer whose resistance varies with applied force. The maximum force measured by the sensor is 10 N.

Figure 2: Force Sensor



Part No. 400 (0.2" Circle)

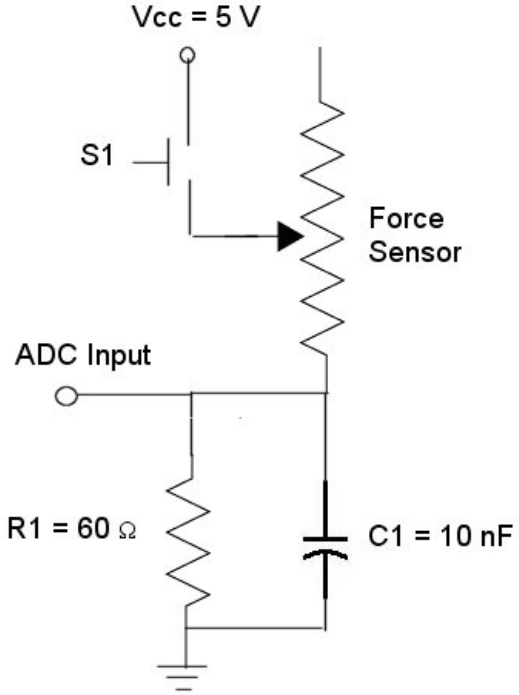
Active Area: 0.2" [5.0] diameter

Nominal Thickness: 0.012" [0.30 mm]

Material Build:

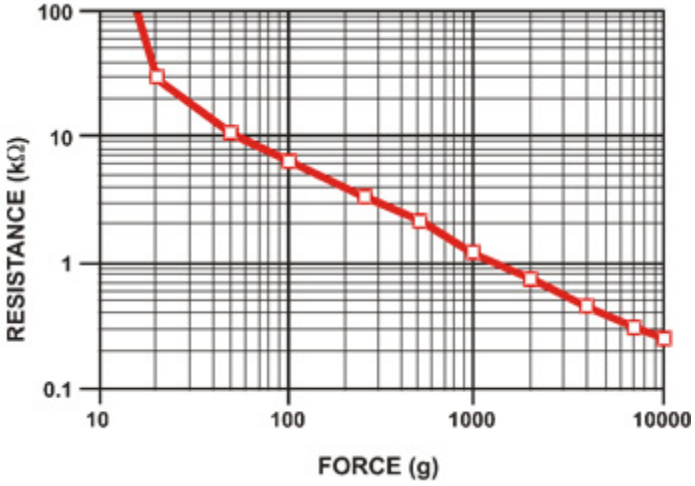
Semiconductive layer
0.004" [0.10] PES
Spacer adhesive
0.002" [0.05] Acrylic
Conductive layer
0.004" [0.10] PES
Rear adhesive
0.002" [0.05] Acrylic
Connector options
a. No connector
b. Solder Tabs (not shown)
c. AMP Female connector

Figure 3: Force Sensor Circuit



With no force applied the sensor behaves as an open circuit and thus S1 is open. Because no current flows the ADC input reads 0 voltage. When a force is applied, the resistance varies from infinite to zero ohms as shown below. With the maximum applied force the resistance is zero and hence the analog voltage read will be the reference voltage Vcc = 5V. Note that the graph is approximately linear.

Figure 4: Force Sensor – Force vs Resistance



3. DC Motor

The DC motor used in the gripper is a Faulhaber DC micromotor with high torque. The precision gear head allows a slower rpm allowing greater precision and control in the positioning of the gripper elements.

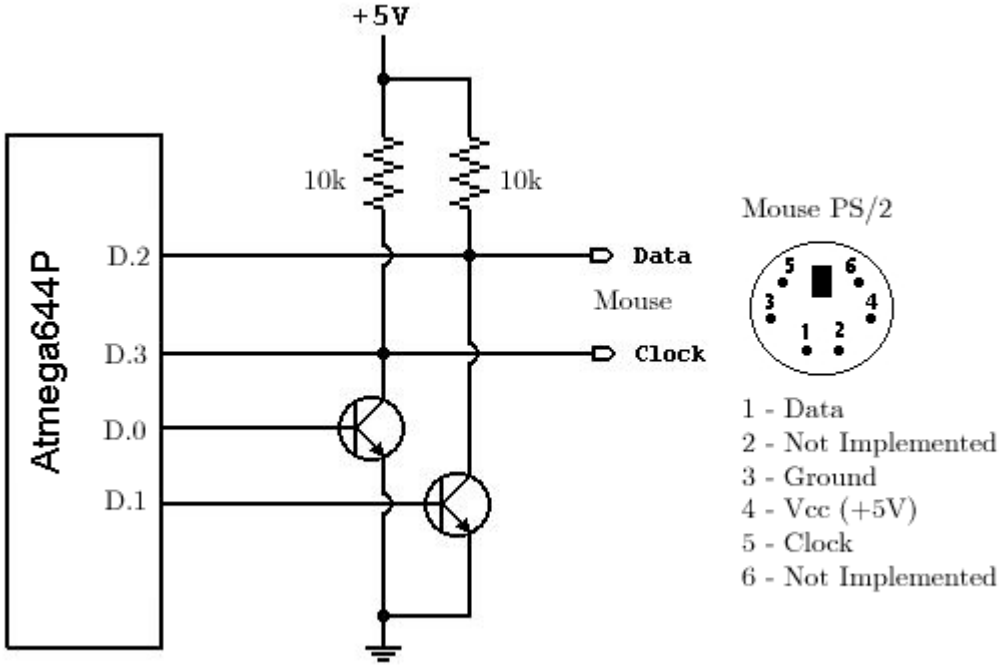
4. Robotic Gripper

The robotic gripper consists of two threaded rods which drive both sides of a hinge. The gripper was originally designed to retrieve soil and rock samples from the desert with two shovel blades. The shovel blades have been removed and replaced with machined pincers one of which bears the pressure sensor, and the other to bear the slip sensor. Unfortunately due to the lack of precision machining capability, the pincers do not align perfectly parallel to each other and hence the force sensor does not read properly. To compensate for this, pieces of Styrofoam have been attached to the surface of the pincers to compensate for this irregularity. With these additions the force sensor is able to read correct values.

5. Slip Sensor

The proposed slip sensor is a modified Logitech LX3 PS/2 optical mouse with a resolution of 1000 dpi. An optical mouse is designed to sense displacements on surfaces and hence should be able to detect an object slipping in the gripper. The interface from the mouse to the microcontroller is shown below. Both the data and the clock lines are connected to the microcontroller through an open collector configuration with pull up resistors. When data D.1 and clock D.2 are pulled high then the mouse is able to send data to the microcontroller.

Figure 5: PS/2 Interface to Microcontroller



The PS/2 clock and data protocol is shown below in the following figures. The PS/2 mouse generates the clock and data signals while the microcontroller board supplies 5V to the mouse.

Figure 6: PS/2 Clock and Data Timing

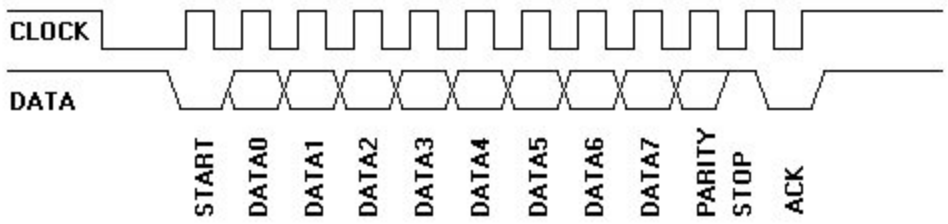
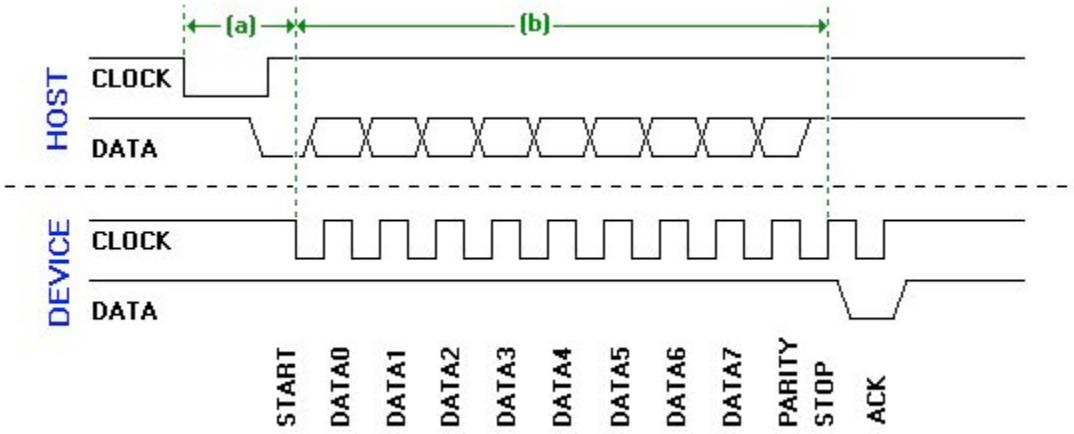


Figure 7: Device to Controller Signal Timing



The communications protocol is as follows

All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11-12 bits. These bits are:

- 1 start bit. This is always 0.
- 8 data bits, least significant bit first.
- 1 parity bit (odd parity).
- 1 stop bit. This is always 1.

The data sent from the mouse is read on the falling edge of the clock. The clock frequency must be in the range of 10 kHz to 16 kHz.

3. Fuzzy Logic Design

The control system input is the reading of the slip sensor that determines whether the gripper is holding the object stationary. If the optical sensor sends data then the object is moving with respect to the sensor position and is hence falling. If no sensor data is sent, then the object is stationary.

The control output for the system consists of the amount of force that the gripper applies to the object that is being grasped. The amount of force is proportional to the position on the axis of movement of the gripper pincers. The control signal consists of a voltage applied to the DC motor for a specific duration corresponding to translation of the pincers along the axis of motion.

The input descriptions for the slip sensor consists of the speed with which the grasped object is moving relative to the sensor position is as follows
Still
Slow
Medium
Fast

The output control description for the gripper consists of the force which is applied to the object as follows
Soft
Medium
Hard

The rules for the fuzzy controller are
1. If the object is still do nothing
2. If the object is moving slowly apply soft force
3. If the object is moving at medium speed apply medium force
4. If the object is moving fast apply hard force

The force measured varies from 0 to 10 N for the force sensor and is presented at the ADC input as voltage range between 0 to 5 V. Thus the range will be

Force output fuzzy variable
Soft : 0 – 2.5 V
Medium: 2 - 3 V
Hard 2.5 – 5 V

The graph of force vs resistance is approximately linear over most of its range.

The slip sensor measures vertical and horizontal displacement of a surface relative to the sensor position at a user definable sampling rate. When the object is grasped in the gripper it is subject to a number of forces
1. The normal force applied by the gripper orthogonal to the surface of the object at the point of contact
2. The gravitational force causing an acceleration of 9.8 m/s^2 assuming the object is held upright
3. The friction force acting in the direction opposite to the acceleration of the grasped object given by $F_f = \mu F_n$ where μ is the coefficient of friction between the surfaces of the gripper and the grasped object.

Determining the range of speed for the slip sensor is somewhat problematic as there are numerous factors. A sampling rate of 200 samples per second will yield one sample every 5 ms. At 200 ms, an object will have fallen a distance of 20 cm due to acceleration by gravity. Thus a vertical displacement of 20 cm over 200 ms or 40 samples means that the object is in free fall and will correspond to the max speed of fast. Conversely the minimum speed for slow is a displacement of 0 cm over any number of samples. In this case, the displacement is measured over every successive sample to ensure that the object is stationary.

Slip Speed Input Fuzzy Variable [displacement in cm / 40 samples]

Slow:	0 - 4
Medium:	2 -16
Fast:	9 - 20

Figure 8:

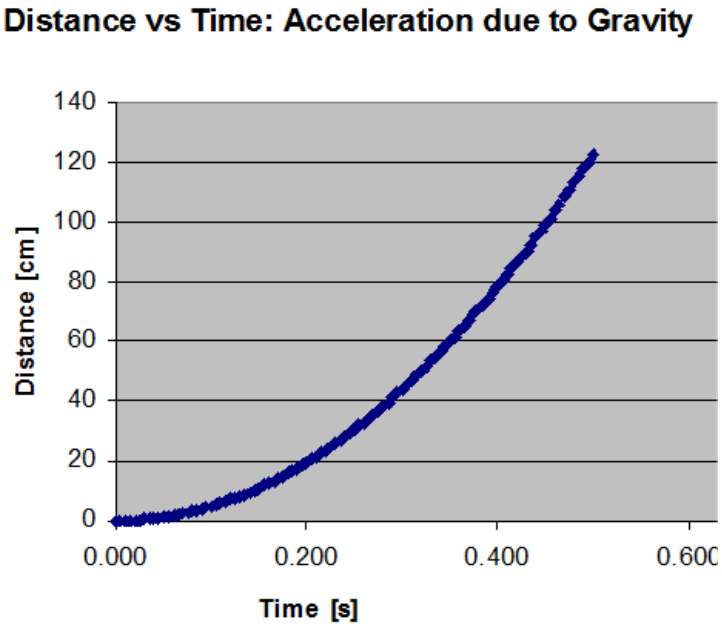
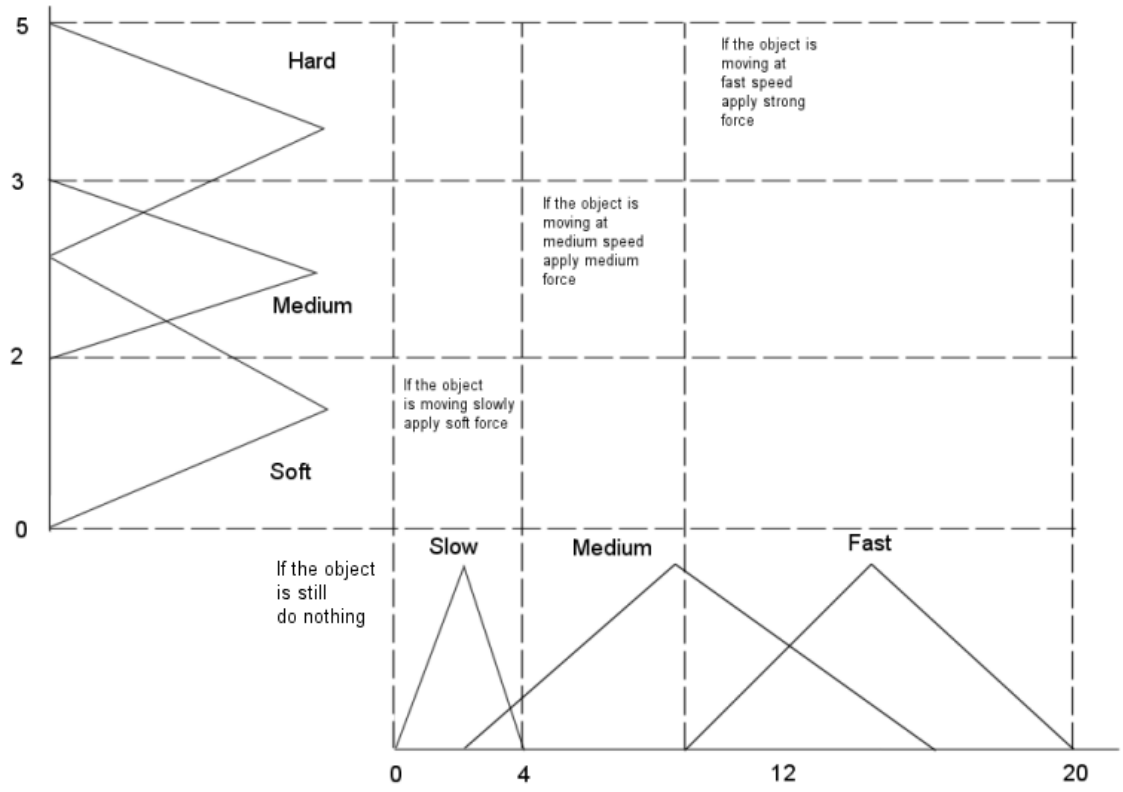


Figure 8: Fuzzy Control Graph



4. Future Work

Because of noisy ADC data it may be necessary to Kalman Filter the sensor input. However, finding the right value of capacitance may be sufficient to filter out high frequency component and thus smooth the input data.

Implement PS/2 protocol in microcontroller code.

Mount the optical mouse sensor on other side of the gripper.

Test and calibrate the system.

APPENDIX A

Microcontroller Code

```
FOR BOARD 2  #define F_CPU 14745600UL

Set Fuse to external 8+ MHz
To check AVR
> sudo make status
To program AVR with external full swing clock and no jtag
> sudo make fuse-extclkfs
> sudo make fuse-nojtag
============================================================================*/

#define F_CPU 14745600UL
#define BAUD 115200
#define BUFSIZE 4
#define DELAY 50

#include <inttypes.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>
#include <util/parity.h>
#include <avr/wdt.h>

void up(void);
void down(void);
void stopY(void);
void left(void);
void right(void);
void stopX(void);
void in(void);
void out(void);
void stopZ(void);
void yawR(void);
void yawL(void);
void stopYAW(void);
void delay();
void laserOn();
void laserOff();
void printBuffer(void);
void serialWrite(int c);
int sendString(char *string);
int i, bufferFull = 0, bufferPointer = 0, d = 1; ADCVAL= 0;
static volatile uint8_t SBUFBUF, uartBuffer[BUFSIZE];

static void initSerial(void) {
        UBRR0H = UBRRH_VALUE;
        UBRR0L = UBRRL_VALUE;
        UCSR0B = _BV(TXEN0) | _BV(RXEN0) | _BV(RXCIE0);
        UCSR0C = _BV(USBS0) | (3 << UCSZ00);
        if bit_is_set(PINA, PA6) {
                UCSR0C &= ~_BV(UPM01);     // no parity bit
        } else {
                UCSR0C |= _BV(UPM01);                   // even parity
        }
#if USE_2X
        UCSR0A |= _BV(U2X0);
#else
        UCSR0A &= ~_BV(U2X0);
#endif
        //UART1
        UBRR1H = UBRRH_VALUE;
        UBRR1L = UBRRL_VALUE;
```

```
	UCSR1B = _BV(TXEN1) | _BV(RXEN1) | _BV(RXCIE1);
	UCSR1C = _BV(USBS1) | (3 << UCSZ10);
	if bit_is_set(PINA, PA6) {
		UCSR1C &= ~_BV(UPM11);	// no parity bit
	} else {
		UCSR1C |= _BV(UPM11);			// even parity
	}

#if USE_2X
	UCSR1A |= _BV(U2X1);
#else
	UCSR1A &= ~_BV(U2X1);
#endif
}

int sendString(char *string)
{
	uint8_t counter = 0;
	// Copy string contents into transmit buffer
	for(counter = 0; counter < 254 && string[counter] != '\0'; counter++)
	{
		UDR0 = string[counter];
		while ( !( UCSR0A & (1<<UDRE0)) );
	}
	return (int)counter;
}

void delay()
{
	for(i = 0; i < d; i++)
	{
		_delay_ms(1);
	}
}

void serialWrite(int c)
{
  while ( !(UCSR0A & (1 << UDRE0)) )
    ;
  UDR0 = c;
}

/* YAW MOTORS */
void stopYAW()
{
	PORTD |= _BV(PD6);
	PORTB &= ~_BV(PB7);
}

void yawR()
{
	PORTB |= _BV(PB7);
	PORTD &= ~_BV(PD6);
}

void yawL()
{
	PORTB &= ~_BV(PB7);
	PORTD &= ~_BV(PD6);
}

/* X DIRECTION L/R MOTORS */
void stopX()
{
	PORTB |= _BV(PB3);
	PORTB &= ~_BV(PB2);
}

void right()
{
```

```
		PORTB |= _BV(PB2);
		PORTB &= ~_BV(PB3);
}

void left()
{
		PORTB &= ~_BV(PB2);
		PORTB &= ~_BV(PB3);
}
/* Y DIRECTION UP/DOWN MOTORS */

void stopY()
{
		PORTB |= _BV(PB4);
		PORTB &= ~_BV(PB5);
}

void up(){
		PORTB |= _BV(PB4);
		PORTB &= ~_BV(PB5);
}



void down()
{
		PORTB &= ~_BV(PB4);
		PORTB &= ~_BV(PB5);
}

/* Z DIRECTION EXTENSION IN/OUT MOTORS */
void stopZ()
{
		PORTD |= _BV(PD7);
		PORTB &= ~_BV(PB6);
}

void in()
{
		PORTB |= _BV(PB6);
		PORTD &= ~_BV(PD7);
}

void out()
{
		PORTB &= ~_BV(PB6);
		PORTD &= ~_BV(PD7);
}

/* TURN ON/OFF LASERS */
void laserOn()
{
		PORTB &= ~_BV(PB2);
		PORTB &= ~_BV(PB3);
}

void laserOff()
{
		PORTB |= _BV(PB2);
		PORTB &= ~_BV(PB3);
}

ISR(USART0_RX_vect)
{
		int byte = UDR0;
		bufferFull = byte;
		//serialWrite(byte);
		if(byte == '1')
		{
			serialWrite(d);
```

```
                d = DELAY/(2);
                //serialWrite(d);
}
if(byte == '2')
{
                serialWrite(d);

                d = DELAY/(5);
                //serialWrite(d);
}

if(byte == '3')
{
                serialWrite(d);
                d = DELAY/(10);;
                //serialWrite(d);
}

if(byte == '4')
{
                serialWrite(d);
                d = DELAY/(25);
                //serialWrite(d);
}

if(byte == '5')
{
                serialWrite(d);
                d = DELAY/(50);
}
if(byte == '0')
{
                //serialWrite(d);
                d = DELAY;
}
if(byte == 'k' && bit_is_clear(PINA, PA2))  //W Up
{
                up();
                delay();
                stopY();
}
else if(byte == 'l' && bit_is_clear(PINA, PA3))             //S Down
{
                down();
                delay();
                stopY();
}
else if(byte == 0x2C && bit_is_clear(PINA, PA0))  // New < Old A Left
{
                left();
                delay();
                stopX();
}
else if(byte == 0x2E && bit_is_clear(PINA, PA1)) // New > Old D Right
{
                right();
                delay();
                stopX();
}
else if(byte == 'q' && bit_is_clear(PINA, PA4)) // New q  Old I
{
                laserOn();
}
else if(byte == 'e' && bit_is_clear(PINA, PA5))  // New e Old O
{
                laserOff();
}
else if(byte == 'm' && bit_is_clear(PINA, PA2)) // K
{
                yawL();
```

```
                delay();
                stopYAW();
        }
        else if(byte == 'n' && bit_is_clear(PINA, PA2)) // L
        {
                yawR();
                delay();
                stopYAW();
        }
        else
        {

        }
}

ISR(USART1_RX_vect)
{
        UDR0 = UDR1;
}

int main(void)
{
        if bit_is_set(MCUSR, WDRF) {
                MCUSR &= ~_BV(WDRF);
                WDTCSR |= _BV(WDCE) | _BV(WDE);
                WDTCSR = 0;
}
        DDRA = 0x00;
        DDRB = 0xFF;
        DDRC = 0xFF;
        DDRD = 0xFA;
        PORTA = 0;
        PORTC = 0;
        PORTD = 0;

        initSerial();
        ADMUX = _BV(REFS0) | _BV(REFS1) | _BV(ADLAR) | _BV(MUX0);
        ADCSRA = _BV(ADEN) | _BV(ADATE) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0) | _BV(ADSC);

        sei();
        stopX();
        stopY();
        stopZ();
        stopYAW();

        delay();
        delay();
        delay();
        sendString("ADC v2  ONLINE");

        d = 50;
        for(;;)  // Loop Forever
        {
                ADCVAL = ADCL;
                ADCVAL = ADCH;
                for(i = 0; i < ADCVAL/5; i++)
                {
                        sendString("*");
                }
  }
        while(1)
        {

        }
}
```

INTERLINK ELECTRONICS
Sensor Technologies

FSR 402 Data Sheet

FSR 400 Series Round Force Sensing Resistor

Features and Benefits

- Actuation Force as low as 0.1N and sensitivity range to 10N.
- Easily customizable to a wide range of sizes
- Highly Repeatable Force Reading; As low as 2% of initial reading with repeatable actuation system
- Cost effective
- Ultra thin; 0.45mm
- Robust; up to 10M actuations
- Simple and easy to integrate

Description

Interlink Electronics FSR™ 400 series is part of the single zone Force Sensing Resistor™ family. Force Sensing Resistors, or FSRs, are robust polymer thick film (PTF) devices that exhibit a decrease in resistance with increase in force applied to the surface of the sensor. This force sensitivity is optimized for use in human touch control of electronic devices such as automotive electronics, medical systems, and in industrial and robotics applications.

The standard 402 sensor is a round sensor 18.28 mm in diameter. Custom sensors can be manufactured in sizes ranging from 5mm to over 600mm. Female connector and short tail versions can also be ordered.

Industry Segments

- Game controllers
- Musical instruments
- Medical device controls
- Remote controls
- Navigation Electronics
- Industrial HMI
- Automotive Panels
- Consumer Electronics

Figure 1 - Force Curve

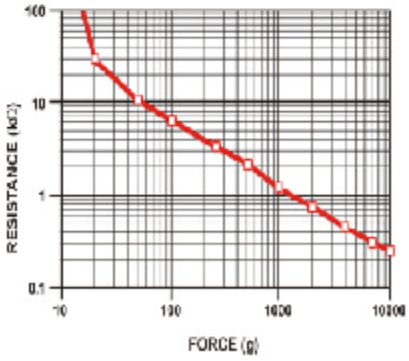
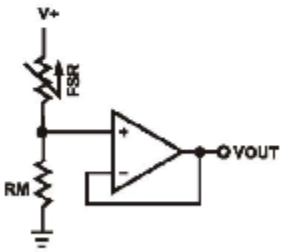


Figure 2 - Schematic



Interlink Electronics - Sensor Technologies