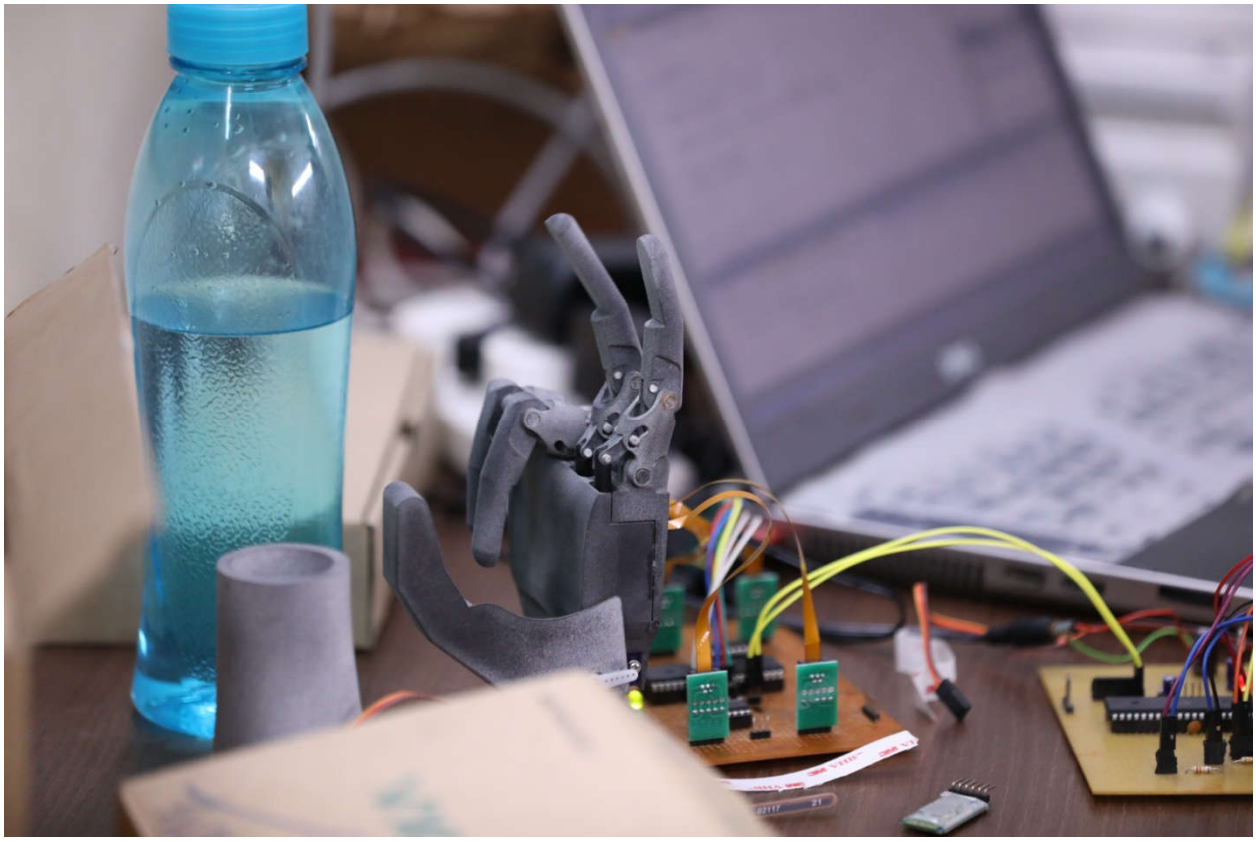
BIONIC HAND

Introduction

Bionic Hand is 3D printed prototype design to enable amputees to perform basic hand functionality. The prototype is prepared using Flex Sensors. The Bionic Hand mimics the hand movements of an individual who wears the Glove. Currently the prototype only has four linear actuators for four fingers providing a single degree of freedom, servo motor for thumb and Flex Sensor Glove which takes data processes it and transmits it to bionic hand. The next version of it will include the Myo Sensors which will eliminate the need of the Glove by attaching the sensors on the muscles of the amputee.



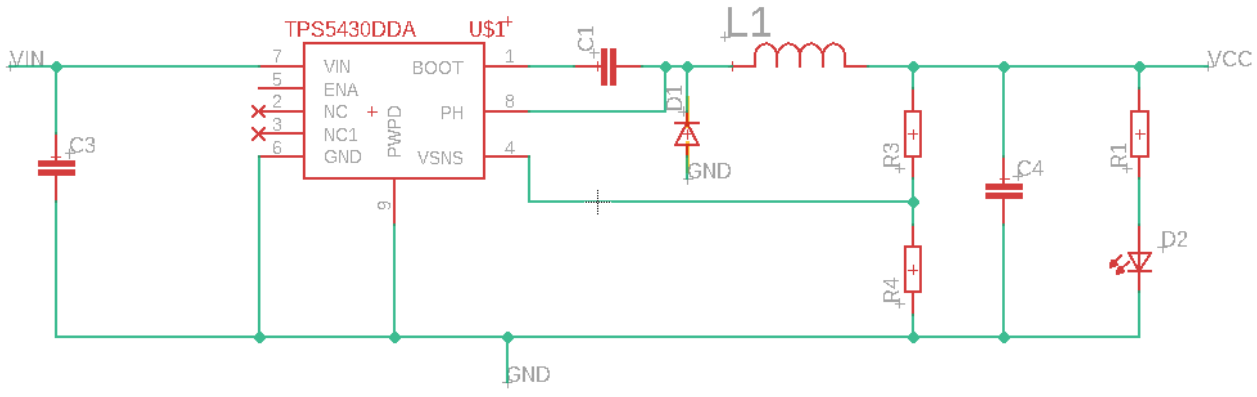
Circuit

There are two circuits to be discussed here:

1. Circuit controlling the bionic hand
2. Circuit on the Flex Glove

Circuit on the Flex Glove

The circuit on Flex Glove has Flex Sensors attached to it. The circuit consists of a Buck Converter Power circuit which converts the input DC voltage in range 5.5V - 30V to 3.9V enough to drive the microcontroller. ATMega328PB is used as microcontroller. ATmega328PB is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture which has operating voltage range of 1.8V to 5.5V. The circuit also contains the Reset circuit. ADC pins 1 to 5 are connected to the Flex Sensors which take the data from the Flex Sensors. Mapping functions are defined which are linear mathematical functions which converts the values from Flex sensors into a 0-1023 range number which is transmitted to the Bionic Hand which defines the position of the finger then. The mode of communication is UART operating at a Baud Rate of 9600. In this case Bluetooth module is used which eliminates the wires and provides a Wireless connection. The Bluetooth module acts as Master. This microcontroller uses only ADC interrupt.



Circuit controlling the Bionic Hand

The circuit controlling the Bionic hand consists of ATMega16 microcontroller. The Circuit drives four Linear actuators and a Servo Motor based on the data it receives from the Flex Glove. The circuit has two L293D motor driver IC which controls the four motors. The motors are connected on PORT B (Index and Middle Motors) and PORT C (Ring and Little Motors). The fingers motion is governed by two parameters: 1) Received Value 2) Feedback Value. The feedback of the fingers are the numbers which defines the position of the finger at an instant. Apart from that the circuit has Reset circuit. The communication protocol is UART operating at a Baud Rate of 9600. Again the Bluetooth module is used which acts as Slave. The operation of microcontroller works on three interrupts – UART (takes data sent from the Flex Glove), ADC (Monitors the position of fingers) and TIMER0(which compares the data received and the current position of the finger and moves the finger accordingly). To reduce the sensitivity of the bionic hand an error margin Is defined which avoids the unnecessary vibrations and noise sent from the flex sensors to bionic hand. The numbers are decided after careful observations and experiments done.

Code

For Flex Glove

The code for the Flex Glove is

```
/*
 * 5transmit_328p.c
 *
 * Created: 6/13/2019 4:08:23 PM
 * Author : PratikAhuja
 */

#define F_CPU 8000000

#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <avr/interrupt.h>
#include <util/delay.h>


void timer_0(int);
void adc_init(void);
void adc_start(char);
```

```
void uart_init(void);
void uart_tx(char);
char uart_rx(void);
void uart_string(char*);
int maap2(int no);   //index
int maap3(int no);   //middle
int maap4(int no);   //ring
int maap5(int no);   //little
int mapth(int no);   //thumb


char a[6],b[5],waste[10],wasteth[10];
int x,gett=0;
uint8_t feedback;
int max_configure=0,min_configure=1000,previousth=0;


int main(){
	char uartData;
	//DDRA=0x01;
	uart_init();
	adc_init();
	//uart_string("Home Automation\r\n");
	sei();
	while (1);
}

void uart_init(void)
{
	UBRR0L=51;		//Baud Rate=9600
	UCSR0B=0x18;
	UCSR0C=0x86;
}


void uart_tx(char data)
{
	UDR0=data;
	while((UCSR0A & (1<<TXC0))==0);
	UCSR0A|=1<<TXC0;
}

char uart_rx(void)
{
	while((UCSR0A & (1<<RXC0))==0);
	return UDR0;
}

void uart_string(char *p)
{

	while(*p!='\0'){
		uart_tx(*p);
		p++;
	}
}

int mapth(int no)
```

```
{
	int temp;
	if(no < 500)  temp = 110;
	else if(no>500 && no<800) temp = 130;
	else if(no>800) temp=140;
	return temp;
}

int maap2(int no){
	int temp;
	//temp = -3.33*(no-700); //originally used
	temp = -2.84*(no-711);
	return temp;
}

int maap3(int no){
	int temp;

	//temp = -3.33*(no-700); //originally used
	temp = -3.41*(no-750);      //for middle
	return temp;
}

int maap4(int no){
	int temp;

	//temp = -3.33*(no-700); //originally used
	temp = -3*(no-638);  //for ring
	return temp;
}

int maap5(int no){
	int temp;

	//temp = -3.33*(no-700); //originally used
	temp = 4.5*(no-802); //for little
	return temp;
}

void adc_init(void){
	ADMUX |= (1<<REFS0) ;

	ADCSRA |= (1<<ADEN) |  (1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2);

	ADCSRA |= (1<<ADIE);
	ADCSRA |= (1<<ADSC);
}

void adc_start(char channel)
{
	char temp;
	temp= channel;
	temp|=0xc0;
	ADMUX = temp;
	//ADCSRA |= (1<<ADSC);
}

void timer_0(int t)
```

```
{

	for(int i=0;i<t;i++)
	{
		TCCR1B|=(1<<CS12)|(1<<CS10);	//prescalar 1024
		OCR1A=6000;
		TCNT1=0;
		while((TIFR0 & (1<<OCF1A)) == 0);// wait till the timer overflow flag is
SET
		TCNT1 = 0;
		TIFR0 |= (1<<OCF1A) ; //clear timer1 overflow flag
	}
}


ISR(ADC_vect)
{
	int dat,mapvalue,mapvalueth;
	dat=ADC;
	//mapvalue=maap2(dat);

	ADCSRA |= (1<<ADIF);
	//itoa(dat,a,10);
	//itoa(mapvalue,waste,10);


	if(feedback!=0 && feedback!=6)
	{
		//uart_tx(feedback+48);
		if(feedback==1)
		{
			mapvalue=maap2(dat);
			itoa(mapvalue,waste,10);
			uart_string("*a");
			uart_string(waste);
			uart_string("|");
			//uart_string("M:");
			//uart_string(waste);	uart_string("\r\n");
			//	_delay_ms(100);

		}

		else if(feedback==2)
		{
			mapvalue=maap3(dat);
			itoa(mapvalue,waste,10);
			uart_string("*b");
			uart_string(waste);
			uart_string("|");
			//uart_string("M:");
			//uart_string(waste);	uart_string("\r\n");
			//	_delay_ms(100);
		}

		else if(feedback==3)
		{
			mapvalue=maap4(dat);
			itoa(mapvalue,waste,10);
```

```
                        uart_string("*c");
                        uart_string(waste);
                        uart_string("|");
                        //uart_string("M:");
                        //uart_string(waste);       uart_string("\r\n");
                        //     _delay_ms(100);
                }

                else if(feedback==4)
                {
                        mapvalue=maap5(dat);
                        itoa(mapvalue,waste,10);
                        uart_string("*d");
                        uart_string(waste);
                        uart_string("|");
                        //uart_string("M:");
                        //uart_string(waste);       uart_string("\r\n");
                        //     _delay_ms(100);
                }

                else if(feedback==5)
                {
                        mapvalueth=mapth(dat);
                        if(previousth != mapvalueth){
                                itoa(mapvalueth,wasteth,10);
                                uart_string("*e");
                                uart_string(wasteth);
                                uart_string("|");
                        }
                        previousth = mapvalueth;
                }
        }

        feedback++;
        _delay_ms(10);
        adc_start(feedback);

        if(feedback>5)
        {
                feedback=0;
        }

        ADCSRA |= (1<<ADSC);
}
```

For Bionic Hand

The code for the Bionic Hand is

```
/*
 * Created: 6/1/2019 2:24:41 PM
 * Author : PratikA
 */


#define F_CPU 8000000
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char dataBuffer[10];
char data[10];
int indexrecieve,middlerecieve,ringrecieve,littlerecieve,thumbrecieve=130;
int indexFeedBack,middleFeedBack,ringFeedBack,littleFeedBack;
uint8_t loop=0;
int i=0,y=0,error=50,j=0;


void indexMotor(int indexrecieve, int indexFeedBack);
void middleMotor(int middlerecieve, int middleFeedBack);
void ringMotor(int ringrecieve, int ringFeedBack);
void littleMotor(int littlerecieve, int littleFeedBack);
void recieve(char *x);
void adcInit(void);
void adcStart(uint8_t channel);
void uart_init();
char uart_recieve();
void uart_transmit(char x);
void uart_string(char *x);
void recieve(char *data);
void timerInit();
void WDT_ON();
void WDT_off(void);
void servoInit();
void servo(int x);
void servomove(int thumbrecieve);


int main(void)
{
       uart_init();
       servoInit();
       servo(155);
       adcInit();
       timerInit();
       adcStart(0);
       uart_string("Hello World");
       sei();
    while (1);
```

```
}


void WDT_ON()
{
	/*
	Watchdog timer enables with typical timeout period 1s
	second.
	*/
	WDTCR = (1<<WDE)|(1<<WDP2);
}

void WDT_off(void)
{
	/* Write logical one to WDTOE and WDE */
	WDTCR = (1<<WDTOE) | (1<<WDE);
	/* Turn off WDT */
	WDTCR = 0x00;
}


void adcInit(void)
{
	ADMUX=0x00;
	ADCSRA=0x8F;  //ADC Enable, ADIE Enable, Presacller 128
}

void adcStart(uint8_t channel){
	uint8_t temp;
	temp=channel;
	temp|=0xC0;
	ADMUX=temp;
	ADCSRA|=1<<ADSC;
}

void recieve(char *x){
	//uart_string("recieve");
	if(x[0]=='a'){
		uart_string("a");
		memmove(data,data+1,strlen(data));
		uart_string(data);
		indexrecieve=atoi(data);
	}
	else if(x[0]=='b'){
		uart_string("b");
		memmove(data,data+1,strlen(data));
		uart_string(data);
		middlerecieve=atoi(data);
	}
	else if(x[0]=='c'){
		uart_string("c");
		memmove(data,data+1,strlen(data));
		uart_string(data);
		ringrecieve=atoi(data);
	}
	else if(x[0]=='d'){
		uart_string("d");
		memmove(data,data+1,strlen(data));
```

```
                uart_string(data);
                littlerecieve=atoi(data);
        }

        else if(x[0]=='e'){
                uart_string("e");
                memmove(data,data+1,strlen(data));
                uart_string(data);
                thumbrecieve=atoi(data);
        }
}

void timerInit(void)
{
        TCCR0=0x0D; //CTC mode , 1/1024
        OCR0=20;  //aprox 2.5 ms at 8 MHz
        TIMSK=1<<OCIE0;
}

void uart_init(void){
        UBRRL=51;   //Baud rate 9600 at 8 MHz Crystal
        UCSRB=0x98;  //0X98 //Enable Rx and Tx  and Rx Interrupt enable
        UCSRC=0x86;  //8 bit, 1 stop, bit parity disable
}


void uart_transmit(char x){
        UDR=x;  //usart i/o data register
        while((UCSRA & (1<<TXC))==0);
        UCSRA|=1<<TXC;
}


void uart_string(char *x){
        int z=0;
        while(x[z] != 0){
                uart_transmit(x[z]);
                z++;
        }
}


char uart_recieve(void){
        while((UCSRA & (1<<RXC))==0);
        return UDR;
}

ISR(USART_RXC_vect){
        //WDT_ON();
        dataBuffer[i]=UDR;

        if (dataBuffer[i]=='*')
        {
                //WDT_off();
                //WDT_ON();
                i=0;
                y=1;
        }
```

```
        else if(y){
                if (dataBuffer[i]=='|')
                {
                        dataBuffer[i]='\0';
                        //WDT_off();
                        //WDT_ON();
                        strcpy(data,dataBuffer);
                        //uart_string("InsideUART_ISR");
                        recieve(data);
                        i=0;
                        y=0;
                }
                else{
                        i++;
                        if (i>8)
                        {
                                i=0;
                                y=0;
                                //WDT_off();
                                //WDT_ON();
                        }
                }
        }
}

ISR(ADC_vect){
        int adcValue;
        char waste[10];
        adcValue=ADC;
        switch(loop){
                case 0:
                indexFeedBack=adcValue;                  //indexfeedback is the location of
index finger in Bionic Hand
                //WDT_off();
                //WDT_ON();
                //itoa(indexFeedBack,waste,10);    uart_string("I");    uart_string(waste);
uart_string("\n");
                break;

                case 1:
                middleFeedBack=adcValue;
                //WDT_off();
                //WDT_ON();
                //itoa(middleFeedBack,waste,10);   uart_string("m");    uart_string(waste);
uart_string("\n");
                break;

                case 2:
                ringFeedBack=adcValue;
                //WDT_off();
                //WDT_ON();
                //itoa(ringFeedBack,waste,10);     uart_string("R");    uart_string(waste);
uart_string("\n");
                break;

                case 3:
                littleFeedBack=adcValue;
                //WDT_off();
```

```
                //WDT_ON();
                //itoa(littleFeedBack,waste,10);   uart_string("L");    uart_string(waste);
uart_string("\n");
                break;
        }
        loop++;
        if(loop>3)    loop=0;
        adcStart(loop);
}

ISR(TIMER0_COMP_vect){

        int target1, position1;
        int target2, position2;
        int target3, position3;
        int target4, position4;

                target1=indexrecieve;
                position1=indexFeedBack;
                indexMotor(target1,position1);


                target2=middlerecieve;
                position2=middleFeedBack;
                middleMotor(target2,position2);


                target3=ringrecieve;
                position3=ringFeedBack;
                ringMotor(target3,position3);


                target4=littlerecieve;
                position4=littleFeedBack;
                littleMotor(target4,position4);

                //if(j==100){
                        servomove(thumbrecieve);
                //      j=-1;
                //}
                //j++;

}



void indexMotor(int indexrecieve, int indexFeedBack){
        if(indexrecieve>=indexFeedBack-error && indexrecieve<=indexFeedBack+error){
                //stop indexmotor
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<0);
                PORTB&=~(1<<1);
        }

        else if(indexrecieve>indexFeedBack+error){
                //index finger up
                WDT_off();
```

```
                WDT_ON();
                PORTB&=~(1<<0);
                PORTB|=(1<<1);
        }
        else if(indexrecieve<indexFeedBack-error){
                //index finger down
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<1);
                PORTB|=(1<<0);
        }
        else{
                //stop indexmotor
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<0);
                PORTB&=~(1<<1);
        }
}

void middleMotor(int middlerecieve, int middleFeedBack){

        if(middlerecieve>=middleFeedBack-error && middlerecieve<=middleFeedBack+error){
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<3);
                PORTB&=~(1<<2);
        }

        else if(middlerecieve>middleFeedBack+error){
                //index finger up
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<2);
                PORTB|=(1<<3);
        }
        else if(middlerecieve<middleFeedBack-error){
                //index finger down
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<3);
                PORTB|=(1<<2);
        }
        else{
                //stop indexmotor
                WDT_off();
                WDT_ON();
                PORTB&=~(1<<3);
                PORTB&=~(1<<2);
        }
}

void ringMotor(int ringrecieve, int ringFeedBack){

        if(ringrecieve>=ringFeedBack-60 && ringrecieve<=ringFeedBack+60){
                //stop indexmotor

                WDT_off();
```

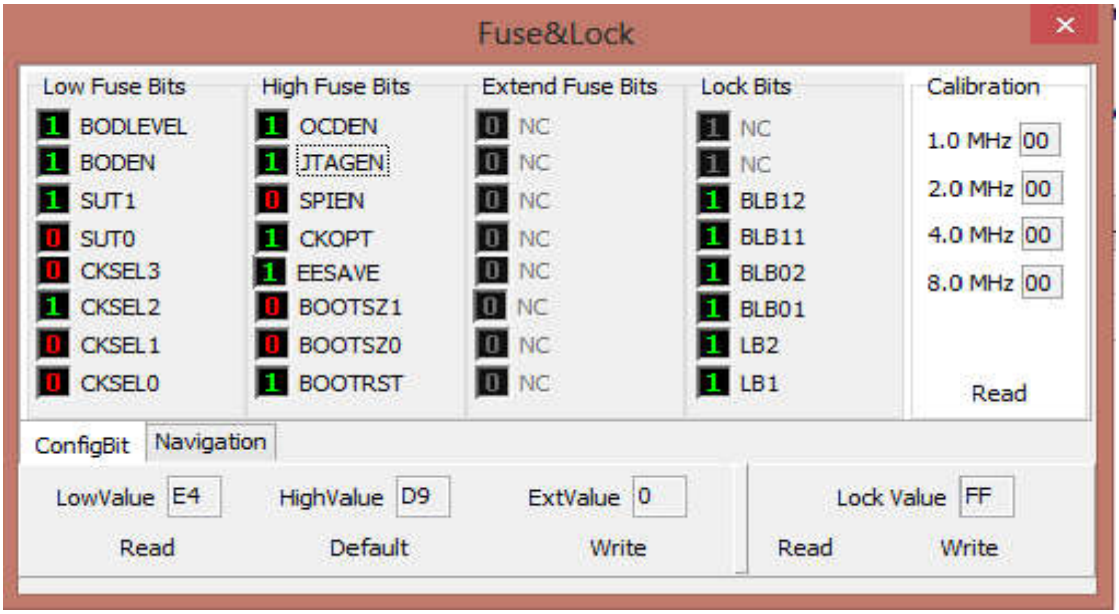
```
                WDT_ON();
                PORTC&=~(1<<1);
                PORTC&=~(1<<0);
        }

        else if(ringrecieve>ringFeedBack+60){
                //index finger up
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<0);
                PORTC|=(1<<1);
        }
        else if(ringrecieve<ringFeedBack-60){
                //index finger down
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<1);
                PORTC|=(1<<0);
        }
        else{
                //stop indexmotor
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<1);
                PORTC&=~(1<<0);
        }
}

void littleMotor(int littlerecieve, int littleFeedBack){

        if(littlerecieve>=littleFeedBack-25 && littlerecieve<=littleFeedBack+25){
                //stop indexmotor
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<2);
                PORTC&=~(1<<3);
        }

        else if(littlerecieve>littleFeedBack+25){
                //index finger up
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<2);
                PORTC|=(1<<3);
        }
        else if(littlerecieve<littleFeedBack-25){
                //index finger down
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<3);
                PORTC|=(1<<2);
        }
        else{
                //stop indexmotor
                WDT_off();
                WDT_ON();
                PORTC&=~(1<<2);
                PORTC&=~(1<<3);
```

```
        }
}

void servoInit()
{
        DDRD|=1<<PD5; // OC1A as output
        TCCR1A=0x82;  //Non inverting FAST PWM mode 14
        TCCR1B=0x1A;  //Prescaler 8   Timer frequency 1 us
        ICR1=19999; //50 Hz PWM Frequency
        //uart_string("Servo Initialised\n");
}

void servo(int x){
        int i;

        i=400 + (x*10.3);
        OCR1A=i;
}


void servomove(int thumbrecieve){
        WDT_off();
        WDT_ON();
        servo(thumbrecieve);

}
```

Note – 1. The TIMER0 is a 2.5 millisecond TIMER Interrupt at 8MHz internal Clock Frequency.

2. The Watch Dog Timer Is used here that generates a system reset if the main program neglects to periodically service it. It is often used to automatically reset an embedded device that hangs because of a software or hardware fault.

3. The Fuse Bit settings for the Bionic Hand working on ATMega16 are



4. The servo can only be at two positions as per the code attached above. The mapping function can be tuned for the desired functionality.

5. The error or margin can be changed or tuned as per the application.

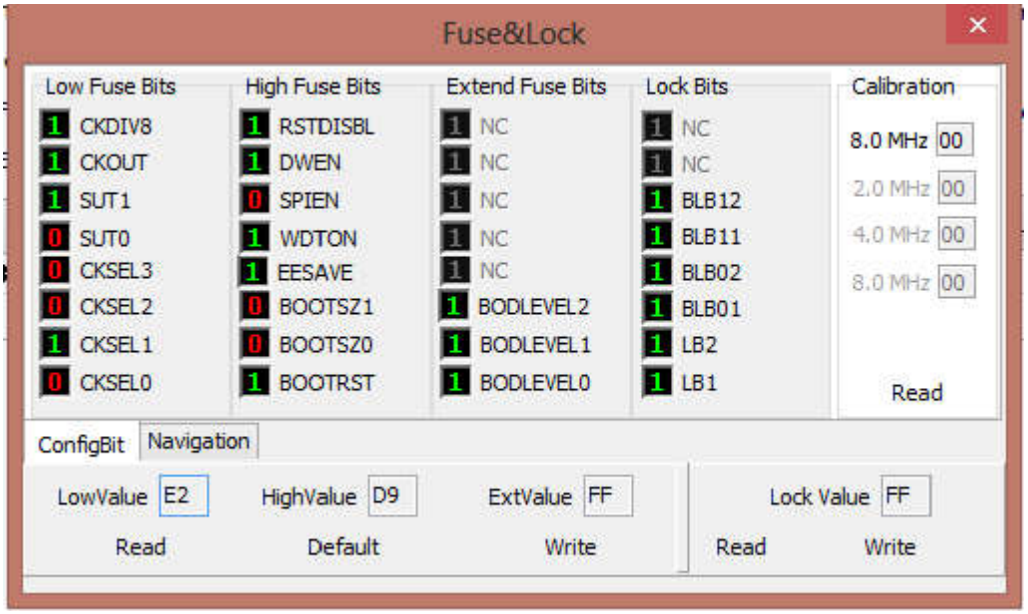
6. The data is encoded as :-

*a800| or *b213| or *c123| or *d1123| or *e400|

Here '*' marks the starting character, '|' marks the end of the string.

'a' is for index finger, 'b' is for middle finger, 'c' is for ring finger, 'd' is for little finger, 'e' is for thumb finger. The number is calculated from the mapping functions based on the resistance value received from the Flex Sensors.

7. The fuse settings for the the flex Code on ATMega328PB is



Bluetooth Module Configuration

The Bluetooth module used was HC-05. The HC-05 module unlike HC-06 module can be made both master and slave using the AT Command mode. It has got one LED, Which shows its state. If it is blinking that means it is not connected. If it is staying in glowing condition that means it is connected.

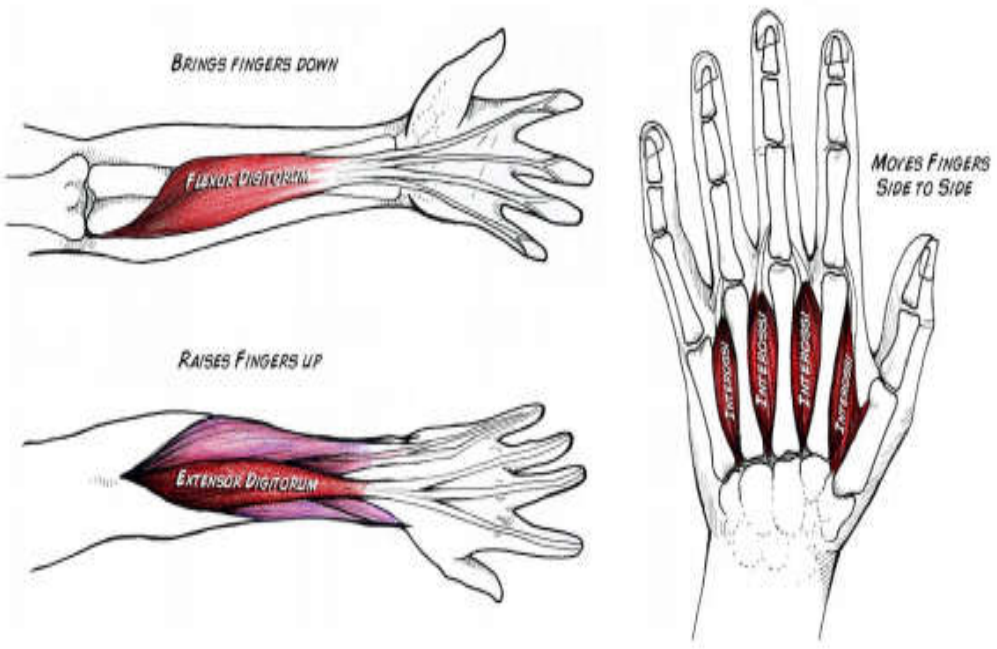
To understand HC05 follow the link - https://roboindia.com/tutorials/arduino-hc-05-at-mode/

The AT Mode Command list explained - https://www.teachmemicro.com/hc-05-bluetooth-command-list/

Using the above AT Commands note down the ADDR of the slave Bluetooth module. Now using the AT Commands make the other HC-05 as master and put the slave's address.

EMG(Myo) Sensors

The next version of the Bionic Hand will include the EMG Sensors which will eliminate the Flex Sensors. Natural fingers and thumb have no muscles. All the muscles that control the fingers are located in the palm and forearm. These muscles connect to the finger via tendons. The muscles that help to control the hand start at elbow or forearm running down to cross the wrist and hand. Few of these muscles position and steady the wrist and hands while some others help to allow the thumb and fingers to grab things and perform as a motor function. The main muscles that move the finger up and down are flexor digitorum and extensor digitorum and the interossi muscle in the hand allow fingers to move side to side.



An EMG signal is the graphical representation of the electrical activity of muscles. Myoelectric refers to the electrical properties of muscles. When somebody thinks about flexing a muscle, their brain sends a neuromuscular or electrical signal to the muscles. At this moment, the muscles start using motor units, or bundles of muscle fibers which generate forces behind the muscles. The amplitude of an EMG signal ranges from 0 to 10 mV and dominant energy is limited to the 0 to 500 Hz frequency range.

The process of collecting myoelectric input ranges from very complex, sophisticated systems that can accurately determine the user's exact intention for muscle movement. However, much simpler options are available for systems that only require the detection of any sort of muscle flex. These systems, such as the MyoWare Muscle Sensor, use a signal amplifier and electrodes to output a raw signal which can then be read by a microcontroller.

MyoWare Muscle Sensor - https://www.tanotis.com/products/genuine-sparkfun-myoware-muscle-sensor?gclid=EAIaIQobChMIoem6j-Sd4wIV1g0rCh0_RA24EAYYAyABEgKMHPD_BwE

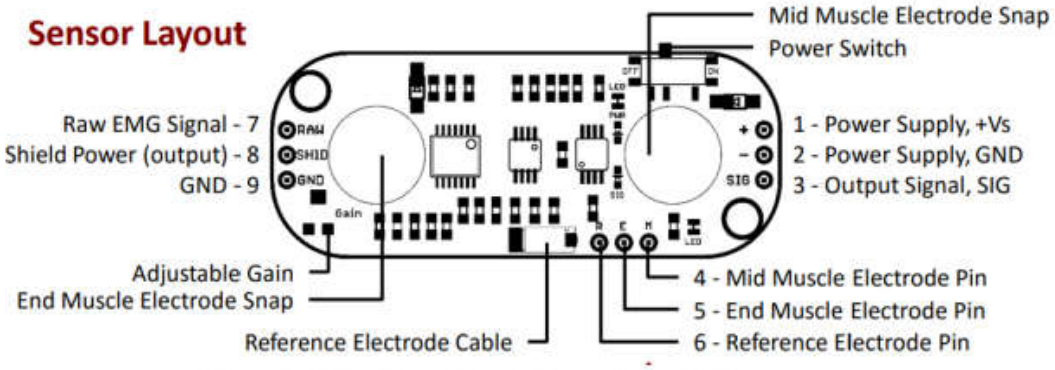


Figure 24: MyoWare Muscle Sensor layout [22]

Datasheet and Setting up of Myoware Sensor - https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MyowareUserManualAT-04-001.pdf

Code for Arduino - https://github.com/AdvancerTechnologies/MyoWare_MuscleSensor

We can use the RAW EMG SIGNAL at PIN 7 or SIG PIN at PIN 3.

Input Sensors are used sometimes for amplification - http://www.liberatingtech.com/products/electronics/Input_Sensors.asp

Few other links to follow for Myoware Sensor guide are:

1. http://www.theorycircuit.com/myoware-muscle-sensor-interfacing-arduino/
2. https://medium.com/physiatry/using-myoware-a-low-cost-surface-electromyography-sensor-for-developing-rehabilitation-devices-1d04a16f5396
3. https://medium.com/@leex5202/an-unofficial-introductory-tutorial-to-myoware-muscle-sensor-development-kit-e2169948e63
4. https://github.com/aurobots/Arduino-EMG-sensor

Data from EMG Sensors can be used in two ways – 1) Finding the ADC value by repetitive experiments for different positions of hand. This method will not be very accurate and can be used if number of positions of hand are limited.

2) Preprocessing the amplified Raw EMG from the sensor using EMG Signal processing Algorithms.

EMG Signal processing Algorithms

Moving average Filter – (Pg 34)
https://www.theseus.fi/bitstream/handle/10024/147196/thesis_susan.pdf?sequence=1&isAllowed=y

With SVM based Classsification Model -
http://portal.amelica.org/ameli/jatsRepo/30/3026002/html/index.html

Embedded EMG Gesture Recognition –

https://www.researchgate.net/publication/283326092_A_Versatile_Embedded_Platform_for_EMG_Acquisition_and_Gesture_Recognition

Advanced Signal Processing using HD Computing -
https://arxiv.org/ftp/arxiv/papers/1901/1901.00234.pdf

MATLAB Libraries for EMG support - https://in.mathworks.com/matlabcentral/fileexchange/71514-emg-feature-extraction-toolbox

Python Library for EMG support - https://github.com/cancui/EMG-Signal-Processing-Library

References –

https://www.theseus.fi/bitstream/handle/10024/147196/thesis_susan.pdf?sequence=1&isAllowed=y

https://digitalcommons.wpi.edu/cgi/viewcontent.cgi?article=3505&context=mqp-all

https://interestingengineering.com/13-prosthetic-arms-and-legs-and-more-that-appear-to-have-come-from-the-future