Shadow hand - a 3 DoF animatronic hand

Ramprasad Rajagopalan
MS in Mechanical Engineering
rrajagopalan1@mines.edu

Bhuvan Tej Kanigiri
MS in Mechanical Engineering
bhuvantejkanigiri@mines.edu

Abstract

From teleoperation to prosthetic hand development, animatronic hand serves as a "simple" implementation to mimic human hand movements. This project deals with design, fabrication and control of 3 finger 3 DOF animatronic "hand" mimicking flex sensor glove worn by human operator. The user wears the glove and the amount of bending in thumb, index and middle finger is captured by the flex sensor attached on each finger. The amount of bending or resistance is sent to Arduino UNO for processing motor commands. The animatronic hand has fingers driven by strings and rubber band. Based on the amount of bending, the Arduino generates corresponding movement in the finger by driving the servo motors attached with the strings.

1. Introduction

1.1. Problem statement

Animatronic hand consists of two systems viz., mechatronic system of the hand and glove worn by the human operator. Only 3 degree of freedom(DoF) is considered out of the 27 DoF present in the human hand. These 3 DoF corresponds to flexion and extension of the middle, index and thumb fingers. The design problem is illustrated in Fig.1. The glove system consists of a design to capture the flexion and extension of each fingers. This can be done by using flex sensors or strain gauges attached to each finger. The flex sensors are placed on the dorsal side of the hand for easier design implementation. The bending data is sent directly to the micro-controller for processing motor commands. The mechatronic system consists of linkages making up the hand driven directly by the motors or by strings pulled by the motors. The linkages should be proportional to the human anatomical structure. The finger linkages are driven by fishing wire tied to each motor. The design should be done in a way to translate effective motion of the motors to maximum flexion movement of each fingers. The microcontroller should apply motor command values by mapping the input flex sensor data to corresponding angle. The required components for building a 3 DoF animatronic hand driven by flex sensor glove is illustrated in the section below.

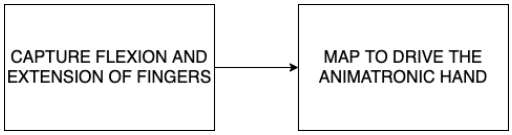


Figure 1. Design problem representation

1.2. System Requirements

Electronics Components

Device name	Model no.	Quantity
Micro Servo Motors	A900	3
Flex sensors(2.2" - 4")	-	3
Arduino	UNO	1
Programmer	AVRISP	1
Resistors	10K, 220	3,3
Jumper cables		

Mechanical Components

Device name	Quantity
3D printed hand with hand base	1
Glove	1
Fishing thread and rubber band	1
Insulation tape	1

Software: Embedded C with Atmel Studio 6 and MATLAB for data visualization.

2. Approach

2.1. Design Implementation

The thumb, index and middle fingers of the right hand are chosen for mimicking the flexion and extension action. These three fingers are commonly used in robotic grippers for object manipulation and pick and place applications.

Glove Design: The implemented glove design is shown in the left labelled image of Fig.3. The flex sensor is at-

Finger	Dimensions (mm)		
	Proximal phalanx	Middle phalanx	Distal phalanx
Index	21	27	48
Middle	24	34	56
Thumb	34	-	28

Table 1. Dimension of each fingers

tached along the length on the dorsal side of the glove at roughly in the middle of the finger with tape. Other effective ways of fixing the flex sensor would be beneficial for better response and prevents unforced error in calibration.

Animatronic hand design: The assembled CAD model and labelled 3D printed hand are shown in Fig.2 and 3 respectively. The 3D models are designed using *Autodesk FUSION 360* software. The hand is designed according to human proportions and measurements as shown in Table.1. The bones in each finger is called as phalanx or phalange. The one closer to the metacarpals or torso of the hand is called as proximal phalanx and the one far from it is called as distal phalanx. The thumb finger is divided into 2 parts

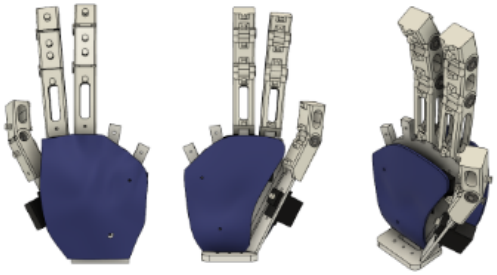


Figure 2. CAD assembly of animatronic hand

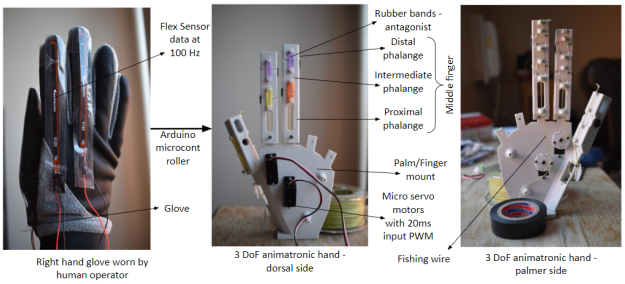


Figure 3. Labelled overview of solution

viz., proximal and distal phalanges while middle and index fingers are divided into 3 parts viz., proximal, intermediate and distal phalanges. The phalanges revolve relative to one another by revolute joint held with the help of plastic bush bearings. All the fingers are designed with the provision to route the fishing wire on the palmer side by providing extruding portions at end of each phalanx. The finger phalanges are also designed with extrusions at each ends on the dorsal side in order to tie the rubber band to the adjacent phalanges. The palm of the hand is designed in such a way that the thumb finger is fixed out-of-plane, to other fingers, in an opposing position as shown in Fig.2. The palm also has provision at the bottom to fix it onto the table.

One end of the fishing wire is tied to tip of the distal phalanx and other end is tied to the servo motor. The placement of the motors is pivotal in defining the range of motion and force exerted on the finger. The spindle of the servo is attached with round hub for compact design and larger force output. The mounting holes for servo motors are provided appropriately in order for it to be inserted in such a way that the spindle faces the palmer side of the hand. The hand design also consist of two form covers for aesthetic appeal and shielding the internal connections.

2.2. PCB Electronics

2.2.1 Circuit schematics

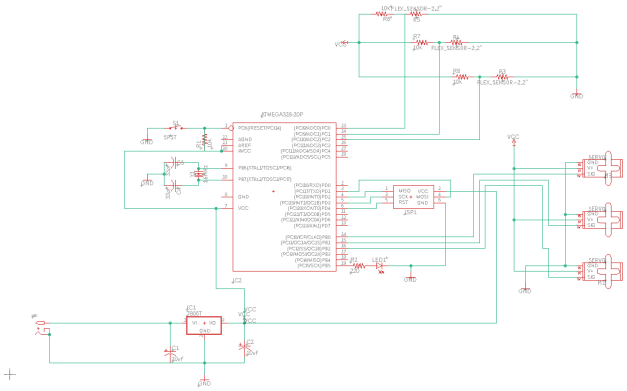


Figure 4. PCB schematics

Power Module: 12 V adaptor is connected to 7806T voltage regulator which serves to output constant voltage to the MCU section. This supply is given to three servos.

MCU Module & Data transfer Module: Used Atmel Studio and AVR ISP programmer to compile the code in the PC and flash the program in the ATMega328P microcontroller.

Pin connections: From ISP to Atmega328P, it has six pins connected to MCU.

Electronics Components

C1: Table.2 is the connection from AVR-ISP programmer to Atmega328P micro controller.

C2: Table.3 is the connection from ATMega328P to flex sensors.

Note: Tapping is done in between the 10k resistor and Flex sensors, both sides are connected to ground and 5V

ISP pin	MCU
MOSI	PD1(TXD)- Pin 3
SPI	PD2(INT0)- Pin 4
Sck	PD3(INT1)- Pin 5
Reset	PD4(XCK/T0) - Pin 6
Vcc	Vin(Any 5v source)
Gnd	Gnd

Table 2. PIN connection between ISP and ATMega328P

MCU	Flex-sensor
PC0(ADC0)	Tapped in b/w **Flex-I** and 10k resistor(r6)
PC1(ADC1)	Tapped in b/w **Flex-II** and 10k resistor(r7)
PC2(ADC2)	Tapped in b/w **Flex-III** and 10k resistor(r8)

Table 3. PIN connections between flex sensor and ATMega328P

MCU	Micro servo motors
PB0	Signal pin (named W) to servo1
PB1	Signal pin (named W) to servo2
PB2	Signal pin (named W) to servo3

Table 4. PIN connections between micro servo motor and ATMega328P

MCU	16MHz Crystal oscillator
PB6(XTAL1)	crystal-oscillator one side
PB7(XTAL2)	crystal-oscillator other side

Table 5. PIN connection between crystal oscillator and ATMega328P

supply.

C3: Table.4 is the connection between ATMega328P and Micro-servo motors.

Note: The other two pins in each servo motor is connected to the supply. A 5V supply is provided but for an ideal working of servo motor, 6V is required. Due to lack of step-down converter equipment, suply is provided at 5V. As of now, there won't be a considerable bias in servo motor characteristics, this is due to the proper threshold mapping done from the flex to servo accordingly by reducing the window.

C4: Table.5 is the connection between ATMega328P and 16MHz crystal oscillator.

I/O Device Module: Flex-sensor is used take input from the environment, where the user bends his hand (change of resistance: which will be the input signal for the EN pin in micro servo motor) and micro-servos act as a output device which bends the animatronics hand.

2.3. Programming

2.3.1 Motor control

The servo motors can be controlled using PWM signals of 20ms time period in order to make it rotate to the desired angle. The 16-bit `TIMER1` of ATMega328P is utilised for generating the PWM signals with desired time period. Multiple PWM outputs can't be achieved using `TIMER1 OC1A` pin. Therefore, the output voltage value of `PORTB` pins are controlled by `TIMER1` to drive the servo motors. This can be achieved by comparing the `TIMER1` count, `TCNT1`, value with the required motor voltage, `motor_vol`, to clear the pin and compare match `ISR` to set the corresponding pin every 20ms. The value of `motor_vol` is obtained from calibration. The clearing and setting of the pin generates the required PWM to control the motor.

2.3.2 Calibration

Flex sensor: The flex sensor data is read every 10ms and stored in a buffer. The data is initially streamed to MATLAB at 100Hz to record and visualize the signals. The streamed data is free from noise and thus eliminates the need for implementing a filter. The extremities of the data for each finger of a specific human operator are determined for flexion and extension. The values are then recorded for saturation and linear interpolation.

Servo motor: The servo motors has a limited range of motion i.e. from 0 deg to approx. 165 deg. The deadband of the servo motor is identified by setting different values to the `motor_vol` in order to generate different duty cycle. These extremities are also recorded and used in linear interpolation and saturation. After identifying the deadband of servo motor, the round hub is appropriately placed at 0 deg with little tension in the fishing wire. This ensures that the motor pulls the finger right from the start of rotation.

2.3.3 Program flow chart

The extremities obtained from the calibration of the sensor and motor are included in the Embedded C program. The sequence of steps followed in programming the Embedded C program is shown in Fig.5. The initialization of the pin is done by setting `PORTB` pins as output. The `TIMER1` is set in Fast-PWM mode with `ICR1` as TOP and 8 as the prescalar value. The `ICR1` value is chosen as 4999 in a way to generate PWM signals with 20ms time period. `ISR` for `TIMER1` is called every 20ms to set `PORTB` pin value as HIGH in order to generate a non-inverting PWM signal. The `TIMER0` is set in CTC mode with `OCR0A` as TOP. The value of `OCR0A` is set in a way to call an `ISR` on compare match every 10ms to raise `flag_0` every 10ms, where `flag_0` is defined as a volatile variable. The `ADC` is initialized with 1024 as prescalar without initialization the

ADMUX channel. By default, the MUX pins are set to read from pin A0.

A while loop is made to run infinitely by iterating through two different conditions. The first condition checks if the TCNT1 value is greater than the duty cycle value and PORTB pin value NOT set. If the condition gets passed then the PORTB pin value is set as LOW or cleared. The value of PORTB pin is set again when ISR of TIMER1 is called every 20ms. In order to reduce the load on the micro-controller to check this statement all time, an entry IF condition can be placed to check if the TCNT1 value lies within certain threshold of the ICR1 value. This helps in improving the performance of the micro-controller.

The second condition gets passed roughly every 10ms when flag_0 is raised. The current ADC value is read and stored. As it is not possible to read simultaneously from 3 analog pins, a SWITCH conditional clause is placed to map the analog value to corresponding motor voltage as well as switch the MUX pins. The ADC value is saturated within the calibration range before mapping. The MUX pin is set to read sequenctially between analog pins A0, A1, A2 and back again to A0. After the MUX pin value is set, another ADC conversion is started. The motor_vol determines the duty cycle of the PWM signals.

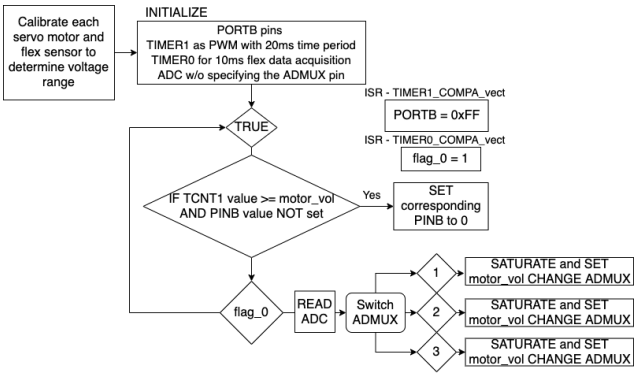


Figure 5. Pseudo code chart

3. Design challenges

The design process has certain critical and challenging steps that needs to be taken into consideration. The animatronic hand should be proportional to human anatomical structure. Unlike the human hand, animatronic hand is not driven by numerous independent linear actuating muscles but driven by single rotating motor. The effective force exerted by the motor depends immensely on the placement height and orientation. Effective pull length of approx. 28mm is delivered by the limited motion of the servo motor. The stiffness of the antagonist rubber band is critical and varies from joint to joint. The effective pull length and stiffness of the rubber band influences the bending angle of each finger. Fig.6 illustrates a finger joint in the actuated condition. The force exerted by the motor acts along the fishing wire. For a given pull length of the wire, the distance from the center of joint to the wire dictates the bending angle of each finger. If the distance increases the bending angle decreases and vice-versa. The effective torque from the motor should overcome the opposing torque of the rubber band. In addition, the opposing torque exerted by the rubber band should be sufficient enough to cause extension of the fingers.

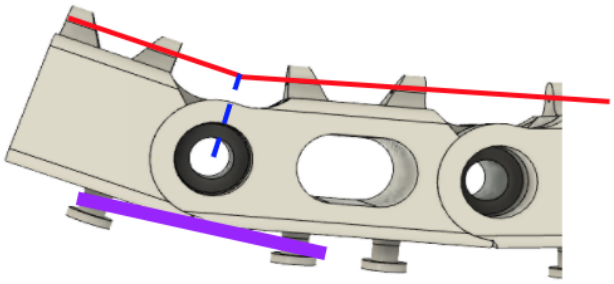


Figure 6. Side view of finger: Red line - fishing wire, Blue dashed line - distance from joint center and fishing wire and purple line - stretched rubber band

The servo motors with nylon gearing is sensitive to external loads. The gears and potentiometer can be easily damaged if the motor is manually turned. The flex sensor calibration can vary from person to person. Friction in the joints is undesirable and unavoidable in 3D printed parts. By overcoming the above mentioned challenges and considerations, a better design of the animatronic hand can be obtained.

4. Future Work and Lessons learnt

The 3 DoF animatronic hand is successfully controlled by tethered flex sensor glove. The next steps is to implement wireless data transfer in order to provide unconstrained interaction to the human operator. This promotes the scope for teleoperation applications. Better motors translate to better performance of the animatronic hand. Custom PCB with power distribution should be ordered and tested.

This project demonstrates the implementation of embedded programming and working with circular buffers. Some of the design challenges can be approached in a more systematic way. Kinematic study should be performed to optimize the geometry. An interactive approach to calibrate the sensor automatically before each run would be beneficial when used for multiple human operators.

5. References

[1] CoffeeCup Software, Inc. “The Animatronic Robotic Hand.” Animatronic Robotic Hand, URL.

[2]“Learn the Fundamentals of Mechatronics!” Microcontroller - Program Transfer Part 1- A Beginners Guide to the Atmel AVR Atmega32, URL.

6. Appendix

Video Demonstration link: Click here.
Eagle CAD: attached
Embedded Code: attached

PSEUDO CODE

Demonstrating embedded programming for reading multiple flex sensor data and controlling multiple servo motors

PROGRAM: Reading analog values from flex sensors and controlling the servo motors

DESCRIPTION: Flexion and extension of a finger is captured by change in voltage across the flex sensors. The analog voltage is read and mapped to corresponding control voltage for each servo motors. The PWM signal is controlled with TIMER1 at 50Hz and data acquisition is done with TIMER0 at 100Hz.

FUNCTION: pin_init

INPUT: None

RETURNS: None

INITIALIZE data direction register of PORTB, DDRB as OUTPUT

FUNCTION: timer_init

INPUT: None

RETURNS: None

INITIALIZE TCNT1 and TCNT0 as 0, SET TIMER1 in fastPWM mode with ICR1 as 4999 as TOP and 8 as prescaler value to generate non-inverting PWM signals at 50Hz, SET TIMER0 in CTC mode with OCR0A as 156 as TOP and 1024 as prescaler value to raise flag every 10ms and enable interrupts for both timers.

FUNCTION: ISR

INPUT: TIMER1_COMPA_vect

RETURNS: None

SET PORTB as 0xFF (HIGH)

FUNCTION: ISR

INPUT: TIMER0_COMPA_vect

RETURNS: None

flag_0 as 1

FUNCTION: adc_init

INPUT: None

RETURNS: None

INITIALIZE ADMUX and SET REFS0, ENABLE ADC in ADCSRA register with 128 as prescaler value using ADPSx pins

FUNCTION: adc_read
INPUT: None
RETURNS: ADC

RETURN ADC

FUNCTION: uart_init
INPUT: None
RETURNS: None

INITIALIZE the baudrate registers UBRR0 and transmission and receiver pins in UCSR0B and SET the data bits to be 8-bit long in UCSR0C register

FUNCTION: uart_transmit
INPUT: data
RETURNS: None

UDR0 as data

FUNCTION: sensor_middle_to_motor
INPUT: sense_pos
RETURNS: motor_vol

CALCULATE mapping using linear interpolation to convert sense_pos to motor_vol

FUNCTION: sensor_index_to_motor
INPUT: sense_pos
RETURNS: motor_vol

CALCULATE mapping using linear interpolation to convert sense_pos to motor_vol

FUNCTION: sensor_thumb_to_motor
INPUT: sense_pos
RETURNS: motor_vol

CALCULATE mapping using linear interpolation to convert sense_pos to motor_vol

FUNCTION: main
INPUT: None
RETURNS: None

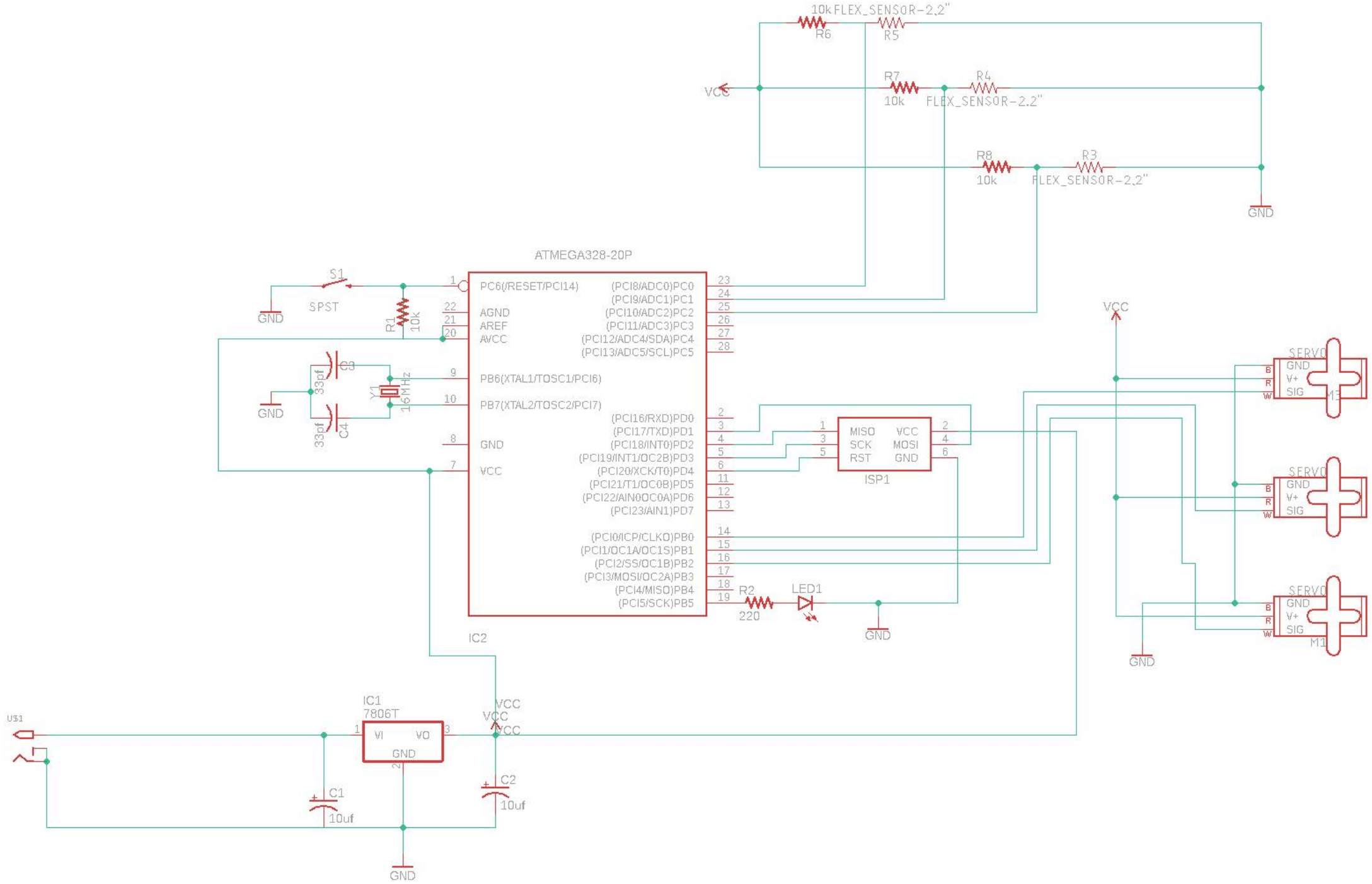
```
INITIALIZE flexthumb, flexmiddle, flexindex variables, pin_init, uart_init, timer_init, adc_init,
float buffer to store the flex sensor data and start an ADC conversion

    FOR ALL TIME
        IF TCNT1>100 AND TCNT1<800 // removing overload on micro-controller
            IF TCNT1 >= middle_count AND bit_is_set(PORTB, PINB1)
                PORTB &= ~(1<<PINB1)

            IF TCNT1 >= index_count AND bit_is_set(PORTB, PINB2)
                PORTB &= ~(1<<PINB2)

        IF flag_0
            IF !(ADCSRA & (1<<ADSC)) // ADC conversion is done
                SET adcValue as adc_read()

                SWITCH ADMUX
                    CASE 0x40
                        SET flexValue as adcValue
                        SATURATE flexValue within calibration limit
                        SET middle_count as sensor_middle_to_motor(flexValue)
                        SET ADMUX as 0x41

                    CASE 0x41
                        SET flexValue as adcValue
                        SATURATE flexValue within calibration limit
                        SET middle_count as sensor_index_to_motor(flexValue)
                        SET ADMUX as 0x42

                    CASE 0x42
                        SET flexValue as adcValue
                        SATURATE flexValue within calibration limit
                        SET middle_count as sensor_thumb_to_motor(flexValue)
                        SET ADMUX as 0x40

                Start another ADC conversion
            SET flag_0 as 0 // reset for TIMER0

    RETURN 0
```

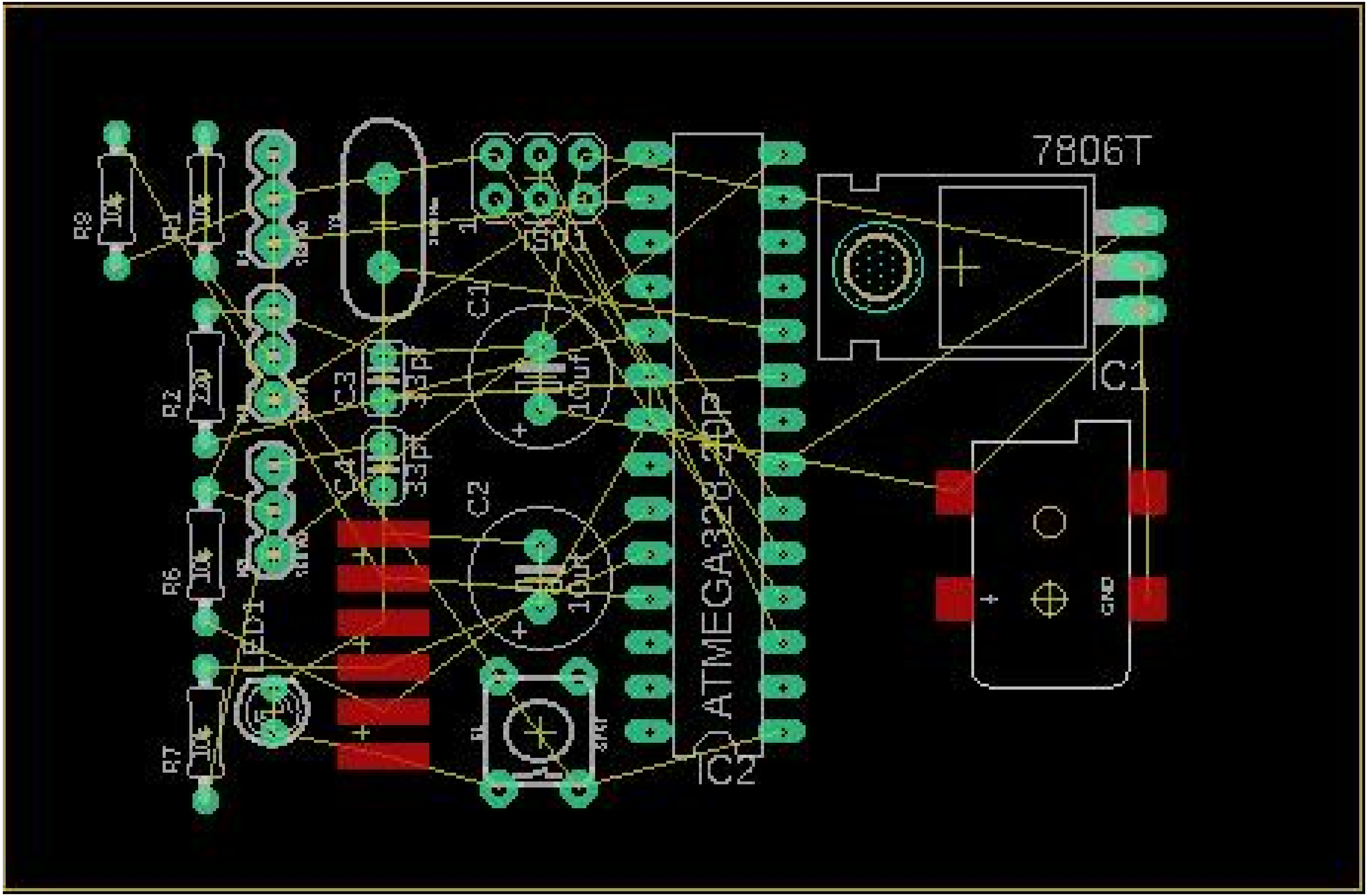
```c
/*
 * animatronics.c
 *
 * Created: 4/24/2020 11:34:49 PM
 *  Author: Ramprasad Rajagopalan and Bhuvan Tej Kanigiri
 */

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <stdio.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "Ring_Buffer.h"

#define BAUD 9600
#define BAUDRATE ((F_CPU/(BAUD*16UL))-1)

// Flag to raise for sending data to MATLAB
volatile uint8_t flag_0 = 0;

//initialization
void pin_init(void)
{
	//A0 is connected to flex sensor 1
	//A1 is connected to flex sensor 2
	//A3 is connected to flex sensor 3
	DDRB = 0xFF;
	//PB1 is connected to servo1
	//PB2 is connected to servo2
	//PB3 is connected to servo3
}

void timer_init(void){
	// Set initial timer value
	TCNT0 = 0; // Serial communication
	TCNT1 = 0; //
	// CTC mode with prescaler as 8 for TIMER1
	TCCR1A |= (1<<WGM11);
	TCCR1B |= (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
	// Set ICR1A as top value for 10ms duty cycle
	ICR1 = 4999;
	// CTC mode with prescaler as 1024 for TIMER0
	TCCR0B = (1<<WGM02)|(1<<CS02)|(1<<CS00);
	// Compare value for 10ms sampling rate
	OCR0A = 156;
	// Enable compare interrupt
	TIMSK1 |= (1<<OCIE1A);
	TIMSK0 = (1<<OCIE0A);
}

ISR(TIMER1_COMPA_vect){
	PORTB = 0xFF;
}

ISR(TIMER0_COMPA_vect){
	flag_0 = 1;
```

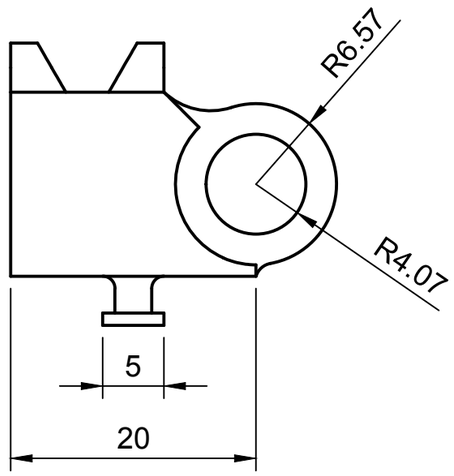
```
}

// Initialize ADC
void adc_init(void)
{
	// Set V_ref and set to read from pin ADC1
	ADMUX = (1<<REFS0);
	// Set prescaler to 128 and Enable ADC
	ADCSRA = (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)|(1<<ADEN);
}

// Read and return ADC value
uint16_t adc_read()
{
	// Return the ADC value
	return(ADC);
}

void uart_init(void)
{
	// Initialize baudrate registers
	UBRR0H = (BAUDRATE>>8);
	UBRR0L = BAUDRATE;
	// Initialize transmission and receiver pins
	UCSR0B = (1<<TXEN0)|(1<<RXEN0);
	// Set data bits to be 8-bits long
	UCSR0C = (0<<USBS0)|(1<<UCSZ00)|(1<<UCSZ01);
}

// Transmit data serially
void uart_transmit(unsigned char data)
{
	// Load to UART buffer
	UDR0 = data;
}

// Function to interpolate middle finger position
int sensor_middle_to_motor(int sense_pos)
{
	int motor_pos;
	motor_pos = (-4.7*sense_pos) + 2494;
	return motor_pos;
}

// Function to interpolate index finger position
int sensor_index_to_motor(int sense_pos)
{
	int motor_pos;
	motor_pos = (-4.3*sense_pos) + 2822.4;
	return motor_pos;
}

int main(void){
	// Variable for flexThumb
	int flexthumb = 0;
	// variable for flexmiddle
	int flexmiddle = 0;
	// variable for flexindex
```

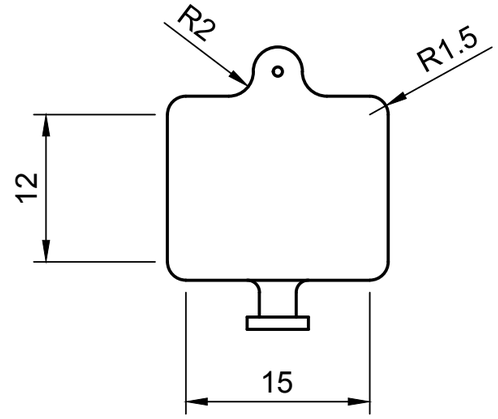
```
int flexindex = 0;
// variable to control PWM input for middle finger and index finger
int middle_count, index_count;
// Clear all interrupts
cli();
// Initialize the pins
pin_init();
// Initialize serial communication pins
uart_init();
// Initialize timer pins and interrupts
timer_init();
// Initialize PWM
pwm_init();
// Initialize ADC
adc_init();
// Initialize global interrupts
sei();
// Start conversion and wait for it to be done
ADCSRA |= (1<<ADSC);
// Filter initialization
struct Ring_Buffer_F buffer_thumb;
rb_initialize_F(&buffer_thumb);
struct Ring_Buffer_F buffer_pointing;
rb_initialize_F(&buffer_pointing);
struct Ring_Buffer_F buffer_middle;
rb_initialize_F(&buffer_middle);
// Initialize output char buffer
struct Ring_Buffer_C p_buf;
rb_initialize_C(&p_buf);
//main loop
while (1)
{
	if (TCNT1>100 && TCNT1<800)
	{
		if (TCNT1>=middle_count && bit_is_set(PORTB,PINB1))
		{
			PORTB &= ~(1<<PINB1);
		}
		if (TCNT1>=index_count && bit_is_set(PORTB,PINB2))
		{
			PORTB &= ~(1<<PINB2);
		}
	}

	if (flag_0)
	{
		// Check if ADC conversion is done
		if (!(ADCSRA & (1<<ADSC)))
		{
			uint16_t adcValue = adc_read();
			// switch the pin
			switch (ADMUX)
			{
				case 0x40:
					flexmiddle = adcValue;
					if (flexmiddle<405) flexmiddle = 405;
					else if (flexmiddle>500) flexmiddle = 500;
```

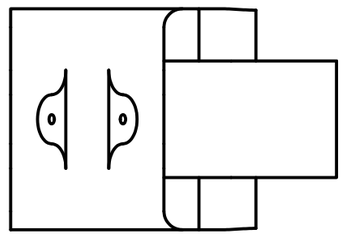
```
                                        middle_count =
sensor_middle_to_motor(flexmiddle);
                                        ADMUX = 0x41;
                                        break;
                                case 0x41:
                                        flexindex = adcValue;
                                        if (flexindex<505) flexindex = 505;
                                        else if (flexindex>620) flexindex = 620;
                                        index_count = sensor_index_to_motor(flexindex);
                                        ADMUX = 0x42;
                                        break;
                                case 0x42:
                                        flexthumb = adcValue;
                                        ADMUX = 0x40;
                                        break;
                                }
                                // start the next conversion
                                ADCSRA |= (1<<ADSC);

                                /*uint8_t *numByte = (uint8_t *) &flexindex;
                                rb_push_back_C(&p_buf, numByte[0]);
                                rb_push_back_C(&p_buf, numByte[1]);
                                rb_push_back_C(&p_buf, numByte[2]);
                                rb_push_back_C(&p_buf, numByte[3]);*/
                        }
                        // Reset flag for timer0
                        flag_0 = 0;
                }

                /*// Check if output buffer has byte and UART serial is free
                if ((rb_length_C(&p_buf)!=0) && (UCSR0A & (1<<UDRE0)))
                {
                        // Load to UART buffer
                        uart_transmit(rb_pop_front_C(&p_buf));
                }*/
        }
        return 0;
}
```


10k FLEX_SENSOR-2.2"
R6
R5
VCC
R7
10k
R4
FLEX_SENSOR-2.2"
R8
10k
R3
FLEX_SENSOR-2.2"
GND
ATMEGA328-20P
S1
SPST
GND
R1
10k
C3
33pf
Y1
16MHz
C4
33pf
GND
PC6(/RESET/PCI14)
AGND
AREF
AVCC
PB6(XTAL1/TOSC1/PCI6)
PB7(XTAL2/TOSC2/PCI7)
GND
VCC
(PCI8/ADC0)PC0
(PCI9/ADC1)PC1
(PCI10/ADC2)PC2
(PCI11/ADC3)PC3
(PCI12/ADC4/SDA)PC4
(PCI13/ADC5/SCL)PC5
(PCI16/RXD)PD0
(PCI17/TXD)PD1
(PCI18/INT0)PD2
(PCI19/INT1/OC2B)PD3
(PCI20/XCK/T0)PD4
(PCI21/T1/OC0B)PD5
(PCI22/AIN0OC0A)PD6
(PCI23/AIN1)PD7
(PCI0/ICP/CLKO)PB0
(PCI1/OC1A/OC1S)PB1
(PCI2/SS/OC1B)PB2
(PCI3/MOSI/OC2A)PB3
(PCI4/MISO)PB4
(PCI5/SCK)PB5
MISO
SCK
RST
VCC
MOSI
GND
ISP1
R2
220
LED1
GND
VCC
SERVO
GND
V+
SIG
M3
SERVO
GND
V+
SIG
SERVO
GND
V+
SIG
M1
GND
IC2
IC1
7806T
VI
VO
GND
VCC
VCC
VCC
C1
10uf
C2
10uf
GND

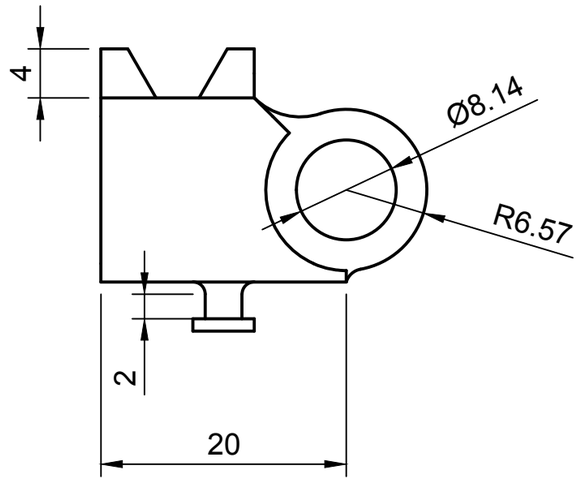

7806T
IC1
ATMEGA328-20P
IC2
C1
10uf
C2
10uf
C3
33pf
C4
33pf
R2
R6
R7
R8
LED1
GND

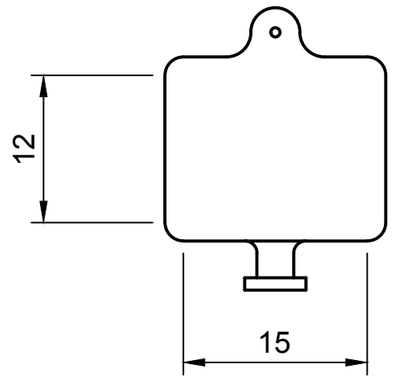


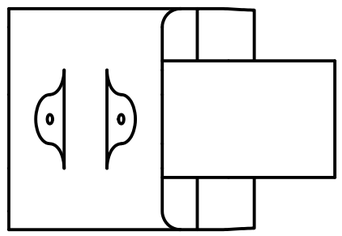


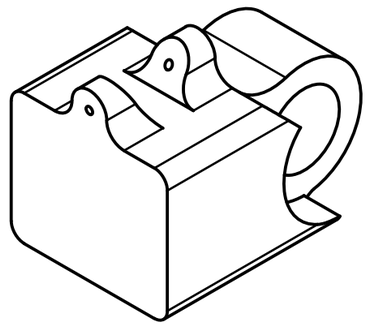


Dept.	Technical reference	Created by Ramprasad Rajagopalan 5/4/20	Approved by
		Document type	Document status
		Title Distal_phalanx_index	DWG No.
			Rev. / Date of issue / Sheet 1/1

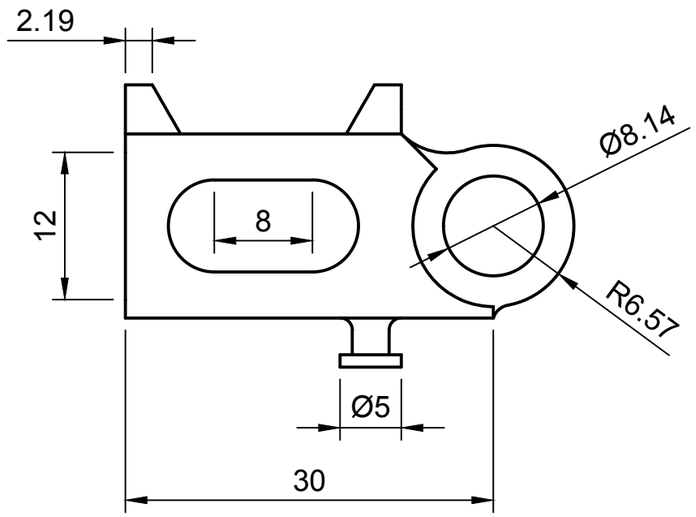


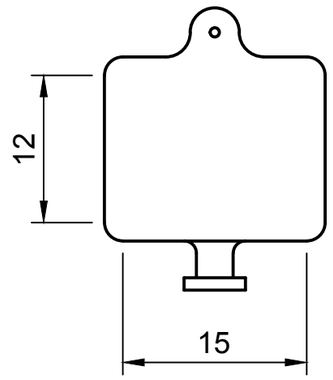


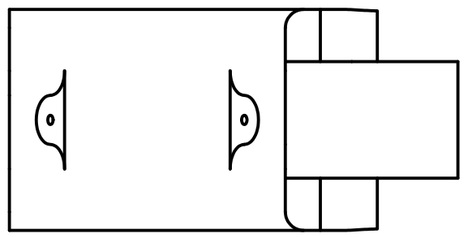


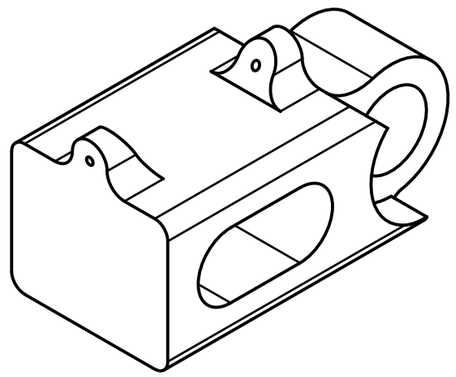


Dept.	Technical reference	Created by	Approved by
		Ramprasad Rajagopalan 5/4/20	
		Document type	Document status
		Title: Distal_phalanx_middle	DWG No.
			Rev. / Date of issue / Sheet: 1/1

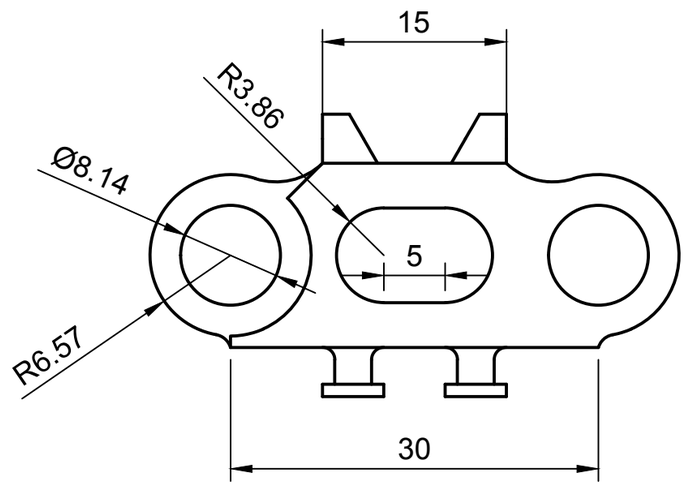


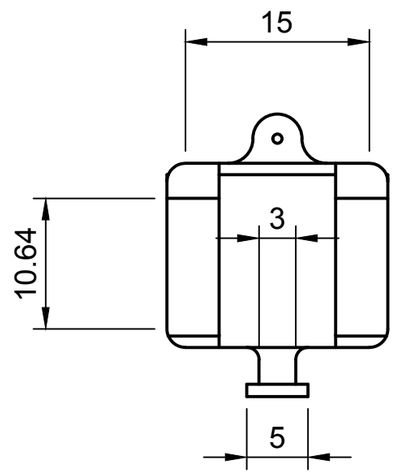


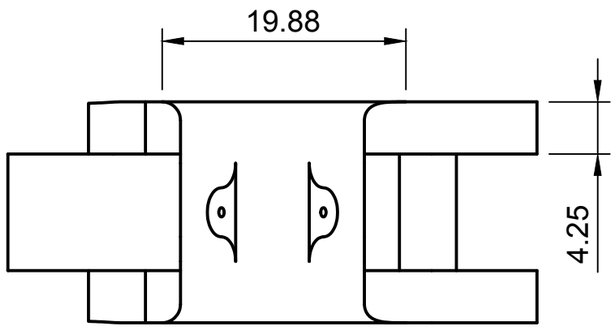


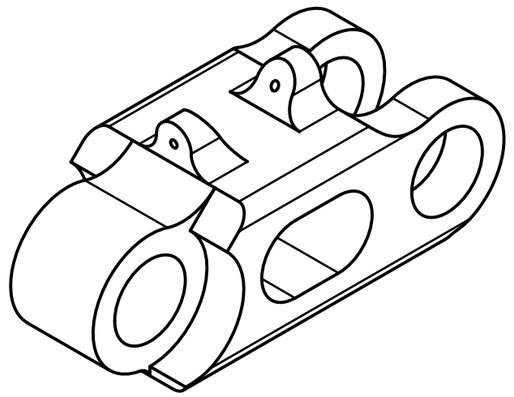


Dept.	Technical reference	Created by Ramprasad Rajagopalan 5/4/20	Approved by
		Document type	Document status
		Title Distal_phalanx_thumb	DWG No.
		Rev. / Date of issue	Sheet 1/1

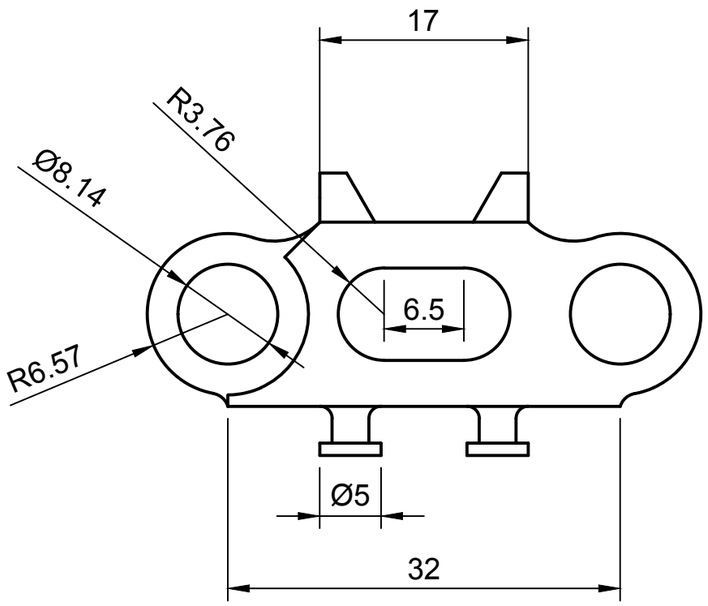


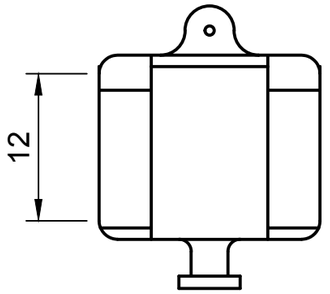


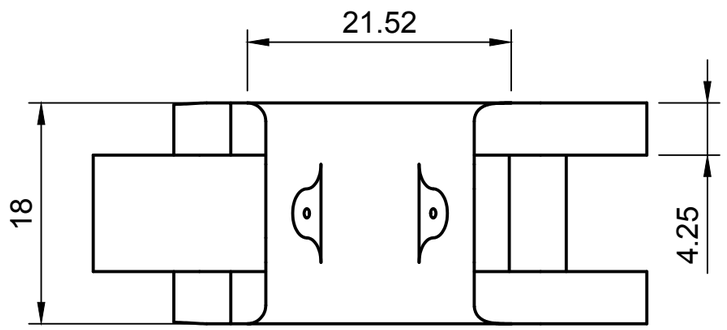


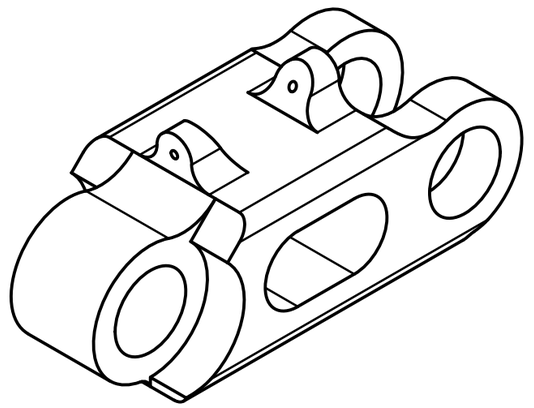


Dept.	Technical reference	Created by	Approved by
		Ramprasad Rajagopalan 5/4/20	
		Document type	Document status
		Title: Middle_phalanx_index	DWG No.
			Rev. / Date of issue / Sheet: 1/1

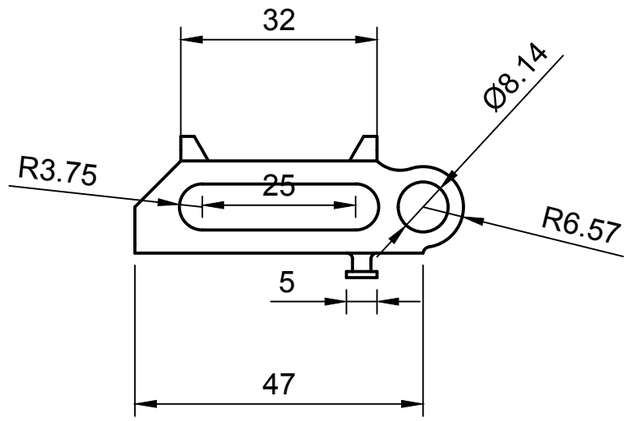


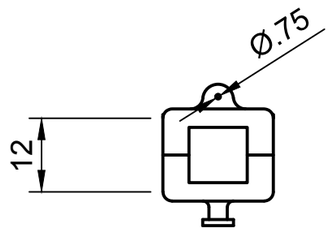


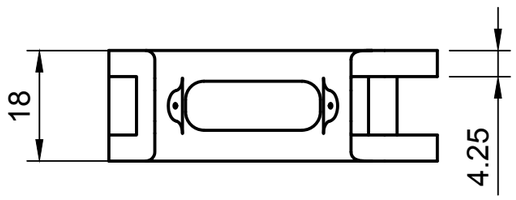


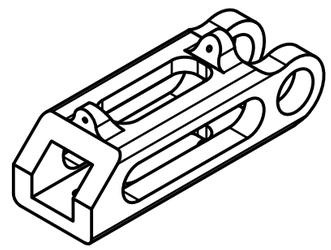


Dept.	Technical reference	Created by Ramprasad Rajagopalan 5/4/20	Approved by
		Document type	Document status
		Title MIddle_phalanx_middle	DWG No.
			Rev. / Date of issue / Sheet 1/1

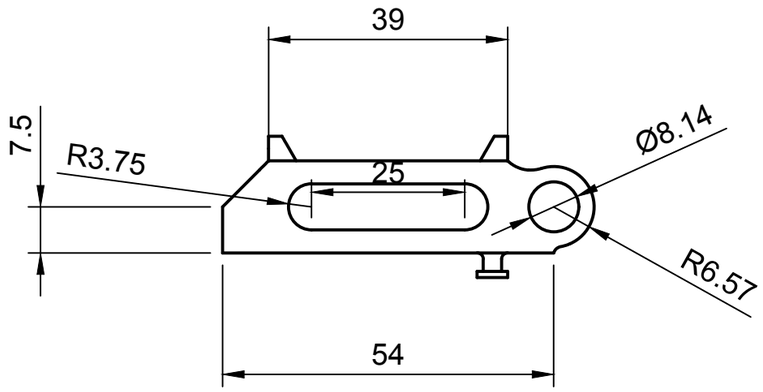


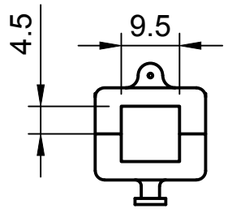


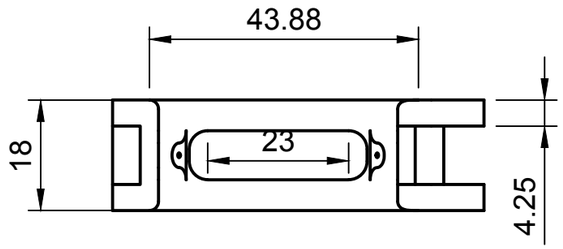


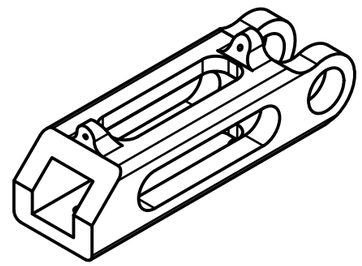


Dept.	Technical reference	Created by Ramprasad Rajagopalan 5/4/20	Approved by
		Document type	Document status
		Title Proximal_phalanx_index	DWG No.
		Rev. / Date of issue	Sheet 1/1

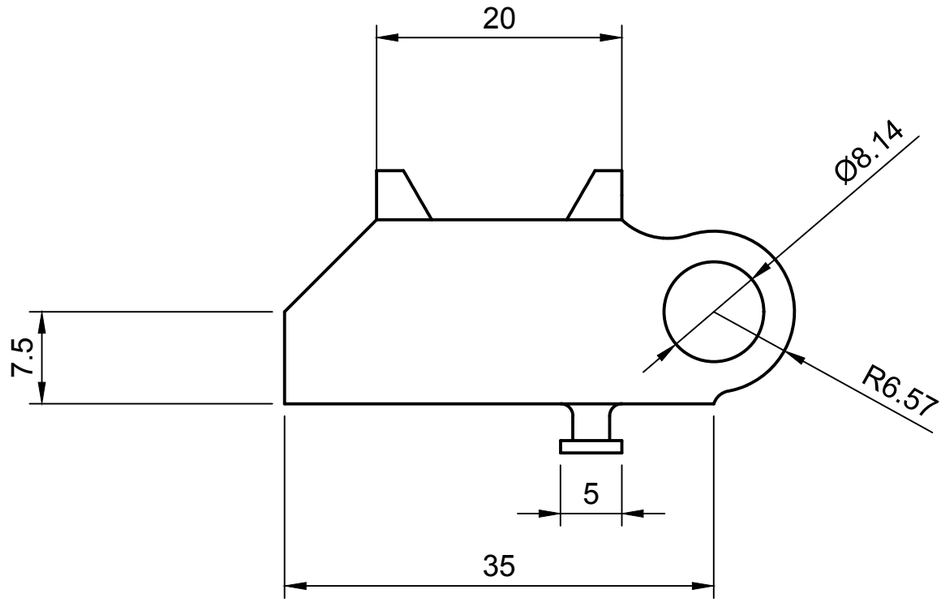


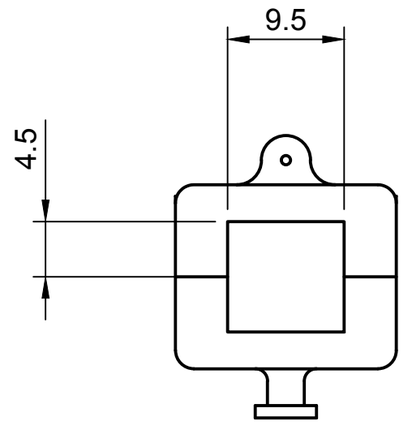


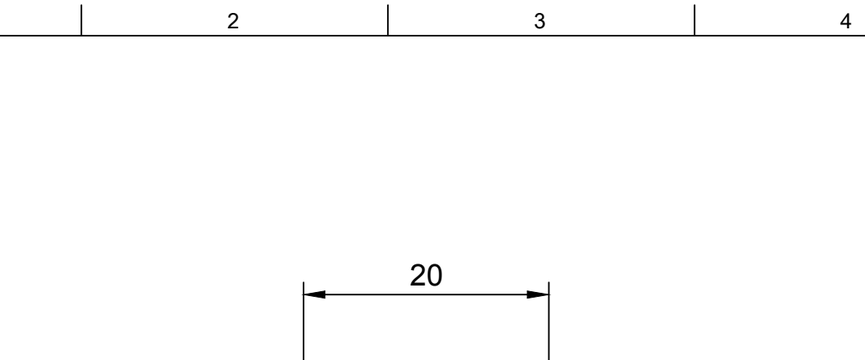


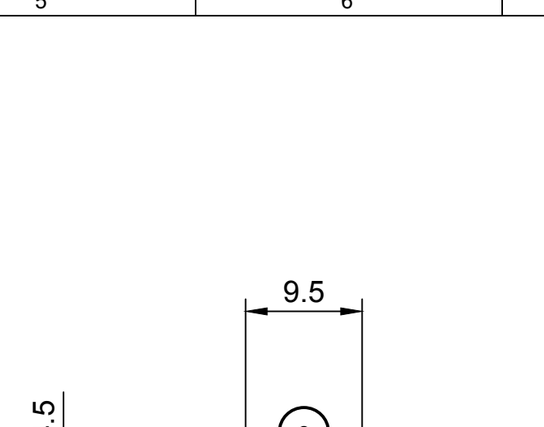


Dept.	Technical reference	Created by Ramprasad Rajagopalan 5/4/20	Approved by
		Document type	Document status
		Title Proximal_phalanx_middle	DWG No.
			Rev. / Date of issue / Sheet 1/1

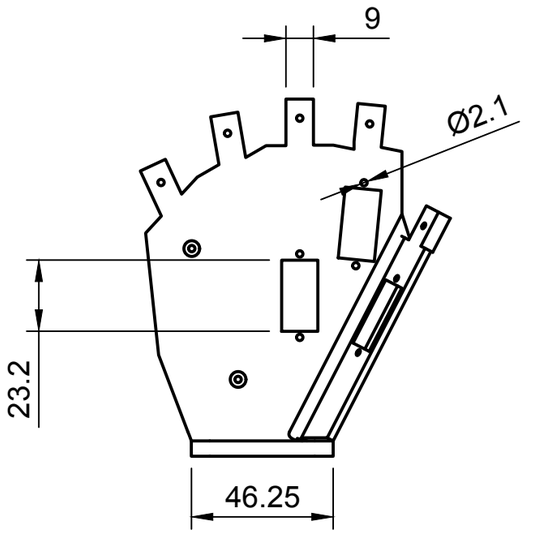


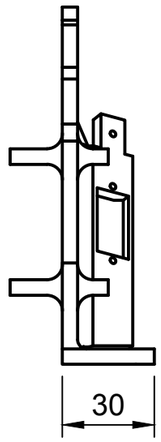


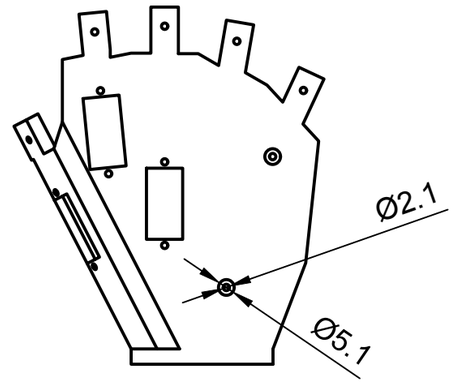


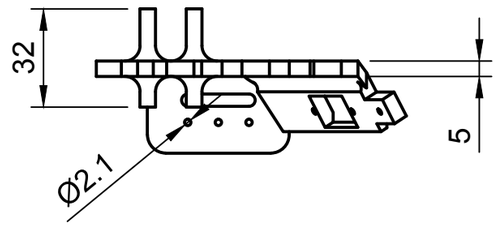


Dept.	Technical reference	Created by Ramprasad Rajagopalan 5/4/20	Approved by
		Document type	Document status
		Title Proximal_phalanx_thumb	DWG No.
			Rev. / Date of issue / Sheet 1/1

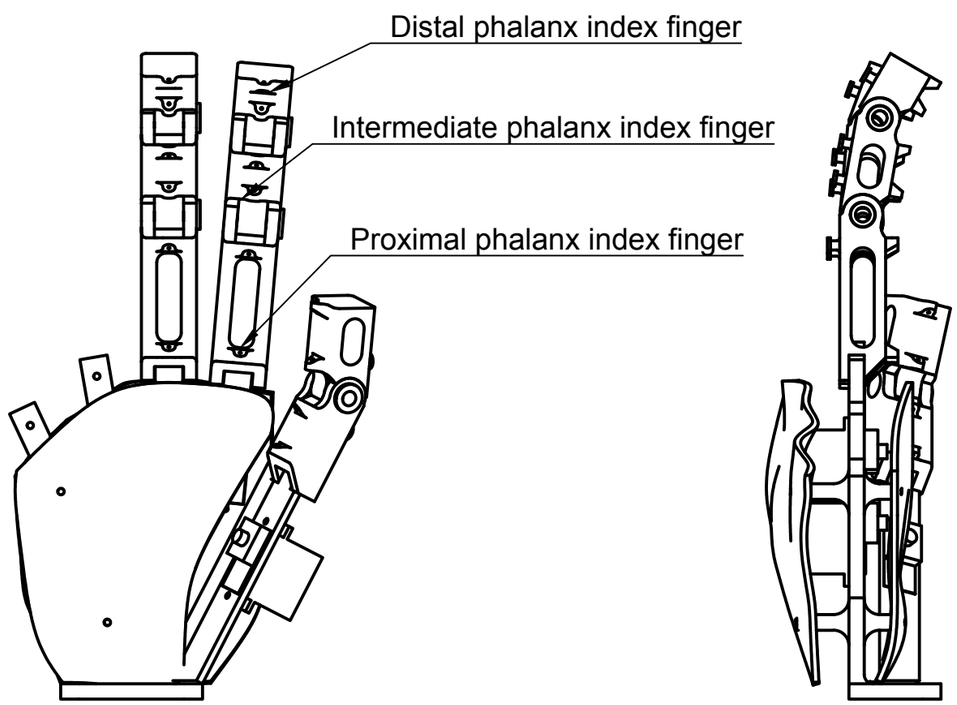


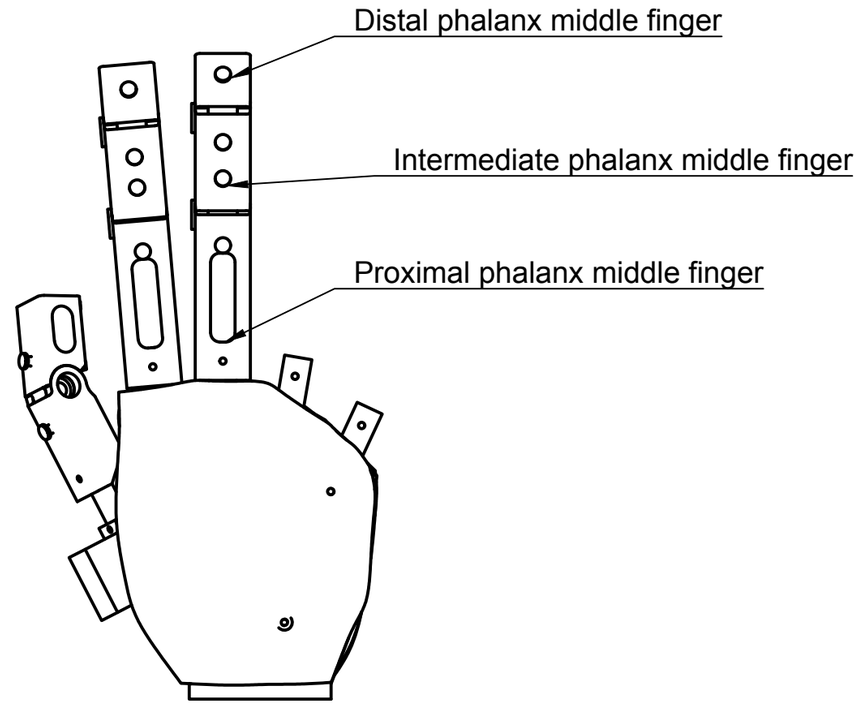


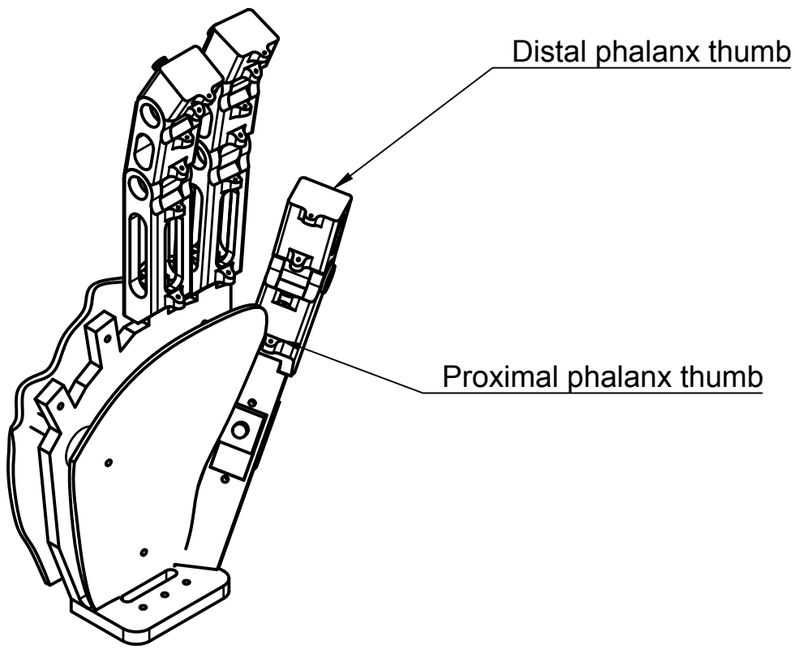




Dept.	Technical reference	Created by	Approved by
		Ramprasad Rajagopalan 5/4/20	
		Document type	Document status
		Title: Palm_right_hand_mod	DWG No.
			Rev. / Date of issue / Sheet: 1/1







Dept.	Technical reference	Created by Ramprasad Rajagopalan 4/28/20	Approved by	
		Document type	Document status	
		Title Hand_mod	DWG No.	
			Rev.	Date of issue / Sheet 1/1