

## TEAM 1 - iARM CONTROLLER FUNCTIONAL REQUIREMENTS

### 1 Purpose

This document presents the functional requirements for an accompanying controller to maneuver the Intelligent Arm Robotic Manipulator.

### 2 REASON FOR RE-ISSUE

ISSUE		REASON FOR RE-ISSUE
1	MR # iARMC.01	This is the first time the plan has been issued
2	MR # iARMC.02	This is the second draft of the requirements
3	MR # iARMC.03	This is the third draft of the requirements
4	MR # iARMC.04	This is the fourth draft of the requirements
5	MR # iARMC.05	This is the fifth draft of the requirements
6	MR # iARMC.06	This is the second draft of the requirements

The nomenclature used in this document is as follows:

- **REQxxx** denotes a **specific requirement** that must be met.
- **BACKxxx** denotes an **information** statement that may be useful in interpreting requirements and the numbering should match the requirement number.

### 3 Document References

The following document references are cited directly or may be useful in interpreting requirements, objectives and information statements contained herein:

- Creo 3.0 Parametric
- Exact Dynamics
- Serial Port Communications
- Arduino Manual
- Bourne Rotary Potentiometer Specification Sheet
- Jameco Potentiometer Specification Sheet
- Piher PTC10MH01-103A2020 Potentiometer Specification Sheet
- Honeywell 392JA10K Potentiometer Specification Sheet
- Dimensioned Part Sketches
- fprintf <https://www.mathworks.com/help/matlab/ref/fprintf.html>
- fscanf <https://www.mathworks.com/help/matlab/ref/fscanf.html>
- Serial.print <https://www.arduino.cc/en/Serial/Print>
- Arduino.read <https://www.arduino.cc/en/Serial/Read>

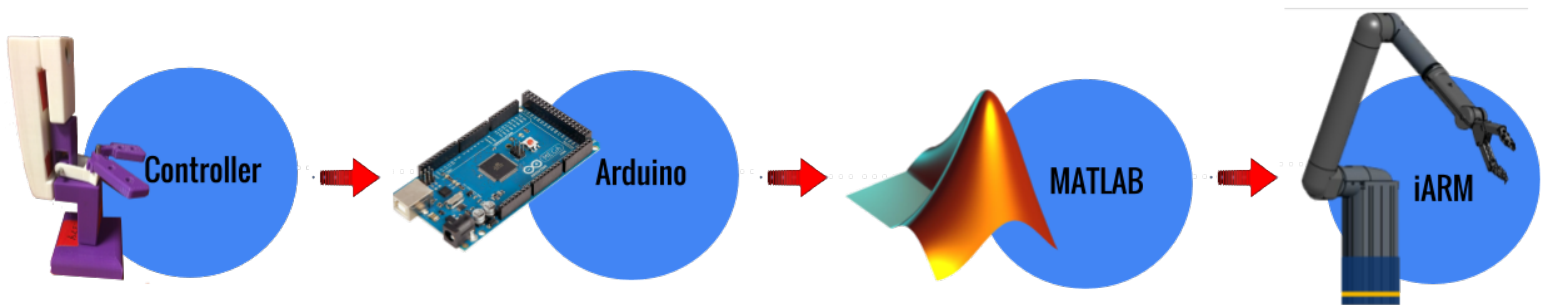
## 4 Overview

The iARM controller is going to be used to manipulate the Exact Dynamics Inc.'s Intelligent Arm Robotic Manipulator (iARM). The 7-degrees of freedom controller will be used to control the iARM by individuals with Arthrogryposis in order to achieve autonomy by providing mobility in order to perform daily living tasks involving their upper extremity. Ultimately, other than providing manual dexterity, this device will be used as a means for rehabilitation through movement re-training.

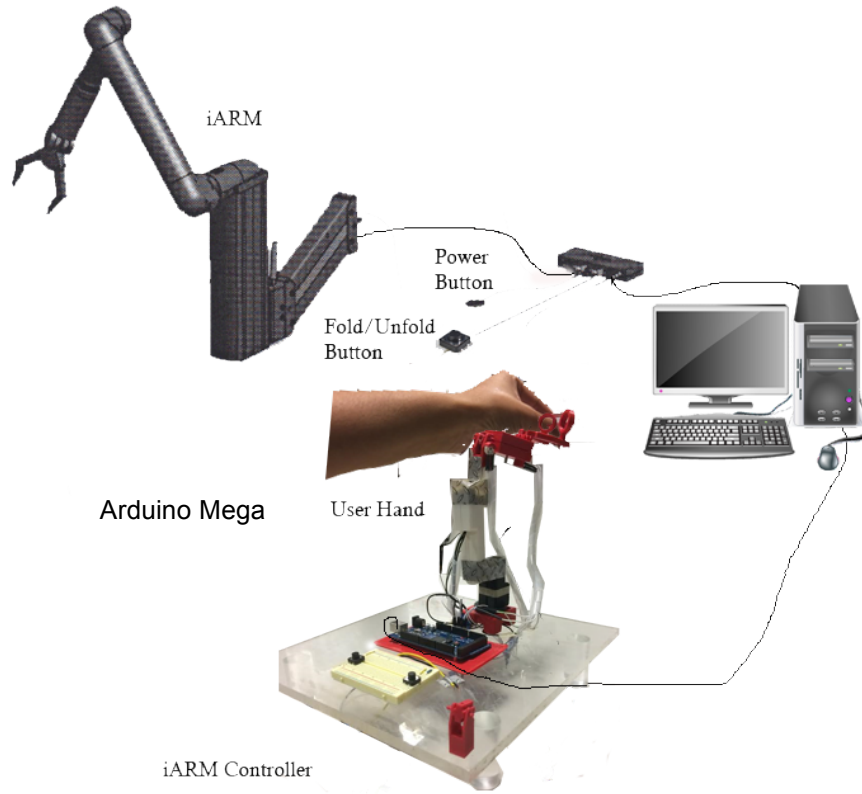
The complete system can be divided into three sub-systems which works together to provide mobility to the users. Shown in **Fig.1** are the main sub-systems: the controller, the Arduino, the computer (MATLAB) and the iARM robot. In the overall system, the input to the device will be a position minimal movement provided by the user through the controller. This input would then pass through the designed algorithm in order to produce the output, which is the similar magnified movement on the iARM robot.

By moving the end effector, the tool at the of the controller which is designed to interact with the environment, to any position in the reachable workspace, the user is able to provide an input. At that specified position, the shaft of the potentiometer at each joint of the controller is twisted to a certain point thereby adjusting the resistance of the potentiometer. The change of resistance then changes the voltage output of that specific potentiometer.

The output of the potentiometer is then connected to the analog pins of the Arduino microcontroller. The pins read the voltage difference created within the potentiometer as the resistance adjusts. The voltage readings are then converted to an iARM joint angle, which is sent from the Arduino to the computer to the iARM robot through a serial connection. Subsequently, once the iARM receives the commands, it's motors at each joint will adjust in to mimic the orientation of the controller.



**Fig. 1 - Overview of iARM controller system**



**Fig. 2: Overall Connection of the iARM**

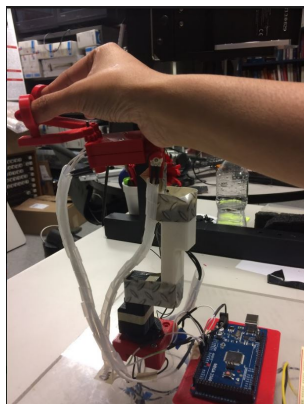
## 5 iARM Functional Requirements

This section provides the functional requirements for the iARM Controller.

### 5.1 *Physical/ Mechanical Requirements*

This section provides physical and environmental requirements for the iARM Controller

**REQ 100 Inputs to the controller:** The user shall input a position to the controller by moving the end effector to a designated position, as shown in **Fig. 3**



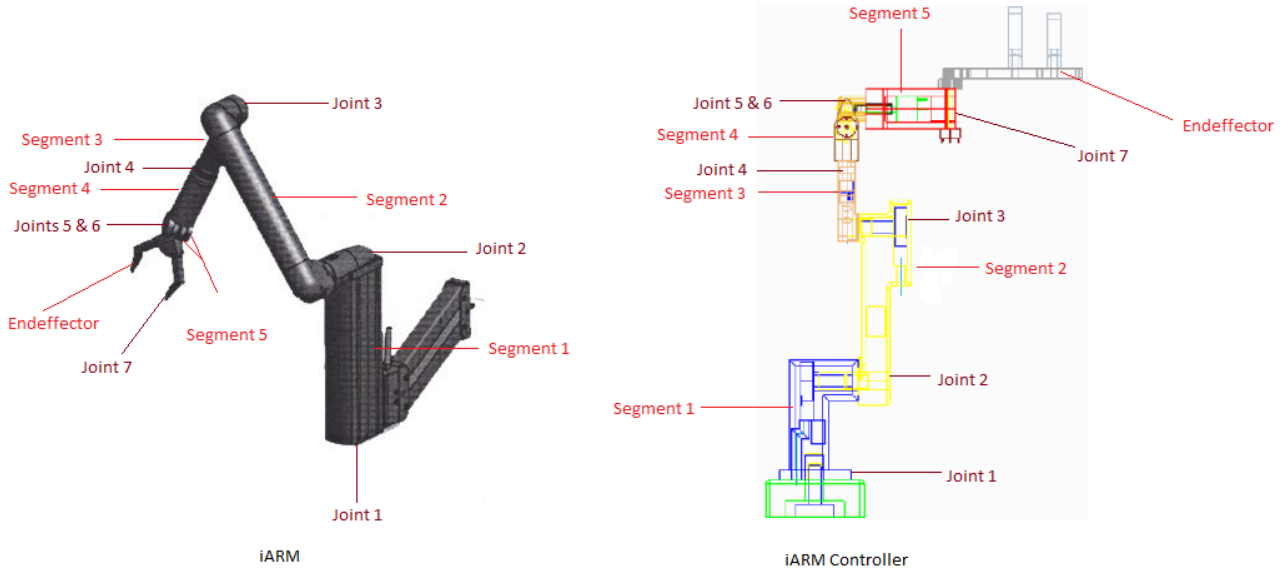
**Fig. 3: User Input – Fingers Holding the End Effector**

**REQ110 Proportionality of Controller:** The controller’s components shall be designed with a 25-28.5% proportionality of the iARM’s segment lengths, distance between the rotation axis of consecutive segments and width of the each segment, as shown in the **Table 1**. The location of each segment of the controller in reference to the iARM is shown in **Fig.4**.

**Table 1-Proportion Between iARM and Controller**

Component	iARM dimensions (mm)	Controller Dimensions(mm)		
		25% Proportion	27% Proportion	28.5% Proportion
Segment 1 width	70	17.5	18.9	19.95
Segment 1 Length	110	27.5	29.7	31.35
Distance between segment 1 and 2 rotation axis	82	20.5	22.14	23.37
Offset of segment 2	72	18	19.44	20.52
Diameter of segment 2	90	22.5	24.3	25.65
Segment 2 length	465	116.25	125.55	132.525
Segment 3 diameter	44	11	11.88	12.54
Distance between segment 2 and 3 rotation axis	51	12.75	13.77	14.535
Segment 3 length	205	51.25	55.35	58.425
Segment 4 diameter	62	15.5	16.74	17.67
Segment 4 length	184	46	49.68	52.44
End Effector length	163	40.75	44.01	46.455

**Fig.4: iARM Controller & iARM Segment Assignments**



**BACK110** If the proportions are, less than 25% then the segments will not be able to house the potentiometers, if the segments are greater than 28.5%, it will not stay within the customer needs workspace. Proportions that do not fall between the range specified will require inverse kinematic calculations; hence it will not be a 1:1 movement translation between the controller and the iARM.

**REQ120 Segment Lengths:** The iARM Controller shall have 21 main components, not including the potentiometers. (Refer to the **Table 2** for the reference document corresponding to each segment.

**Table 2: References for Each Segment**

Segment	Reference
BASE	iARMCDD02
SEGMENT 1A	iARMCDD03
SEGMENT 1B	iARMCDD04
SEGMENT 2A	iARMCDD05
SEGMENT 2B	iARMCDD06
SEGMENT 3A	iARMCDD07
SEGMENT 3B	iARMCDD08
SEGMENT 4	iARMCDD09
CROSS	iARMCDD10
SEGMENT 5A	iARMCDD11
SEGMENT 5B	iARMCDD12
THUMB	iARMCDD13

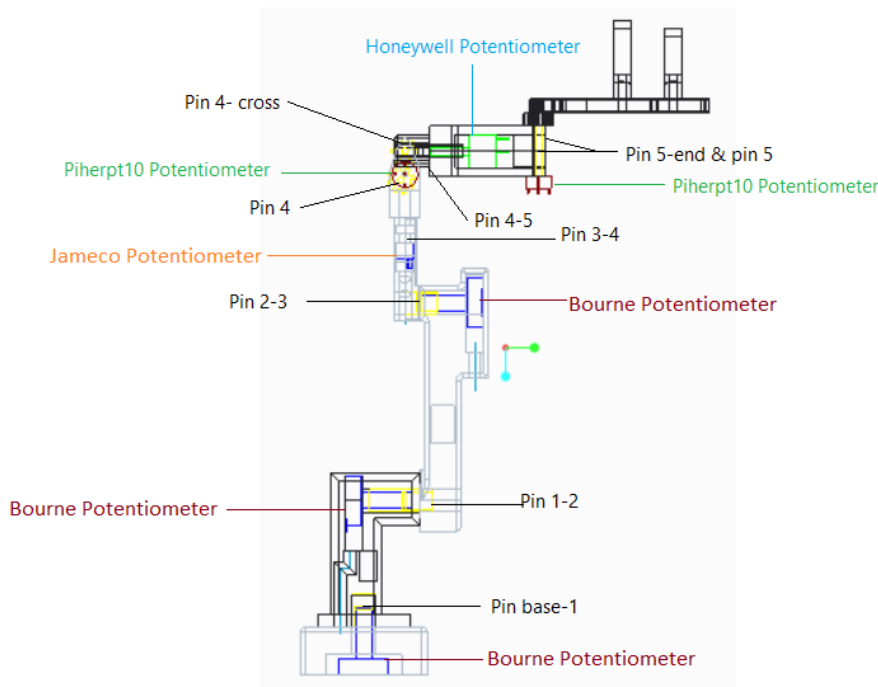
POINTER	iARMCDD14
RINGS	iARMCDD15
PIN BASE-SEGMENT 1	iARMCDD17
PIN SEGMENT 1-2	iARMCDD18
PIN SEGMENT 2-3	iARMCDD19
PIN SEGMENT 3-4	iARMCDD20
PIN SEGMENT 4 & CROSS	iARMCDD21
PIN CROSS-SEGMENT 5	iARMCDD22
PIN SEGMENT 5-END	iARMCDD23

**REQ130 Segment Length Proportionality:** Each dimension shall be proportional to the iARM itself.

**REQ140 Tolerance of Segments:** Each segment shall have a tolerance of  $\pm 2$ mm.

**REQ150 Joint Connection:** At each joint, a potentiometer- pin case complex that will be specific for each joint shall connect the segments. (Refer to **Fig.5** to view the location of each potentiometer within the controller.)

**BACK150** The potentiometer base will be static in the segments where they are encased and the pin case will be made static within the segments where they are enclosed.



**Fig. 5: Potentiometer Location**

**REQ160 Materials:** The iARM Controller shall be manufactured using Delrin.

**BACK160** Delrin has high abrasion resistance, low coefficient friction, high heat resistance, good electric and dielectric properties and low water absorption, which is ideal for this particular application.

**REQ170 Weight:** The iARM Controller shall weigh less than 5 pounds.

**BACK170** The iARM Controller's weight is **TBD**.

**REQ180 Strength:** The iARM Controller shall be fabricated to withstand the weight of the hand (2 lbs.) and arm movements generated by patients who suffer from Arthrogyrosis with wrist movement.

**BACK180** The amount of force produced by MD patients vary from force magnitudes of 2 N to 5 N.

## **5.2 Electrical Requirements**

This section provides the requirements for the electrical components within and pertaining to the iARM controller.

**REQ210 Potentiometer Function:** Each joint of the controller shall have a potentiometer which shall measure the amount of rotation in each joint. As each joint rotates, the shaft of the potentiometer shall twist and each rotation shall have an equivalent angle.

**REQ220 Potentiometer Specifications:** Bourne 3852A-282-103AL 10 K ohms - Single Turn Rotary Potentiometers with  $\pm 20\%$  shall be placed in joints 1, 2 & 3. The potentiometers located at joint 4 shall be Jameco ValuePro 3329P-1-103 1/2 Watt, 1/4 Inch, 10 k $\Omega$  Round Cermet Potentiometer with  $\pm 20\%$ . The potentiometers located at joints 5 & 7 shall be Piher PTC10MH01- 103A2020 10 Kohms 10mm Round. Honeywell 392JA10K 10K Ohm 10% 1/2W 3.18mm Potentiometers shall be placed at joint 6. The locations of the potentiometers are shown in **Fig 5**.

**BACK220** For joints 1, 2, & 3, the Bourne potentiometers will be used in order to provide support and strength to the base.

**REQ230 Arduino:** The controller shall need an Arduino Mega microcontroller.

**BACK230** The Arduino Mega will provide the necessary amount of pins compared to an Uno which has only 6 analog pins. The Mega has 16 Analog pins. The Arduino Mega 2560 microcontroller will be used to provide us 16 analog inputs that will allow for 7 potentiometers to be connected. It has an operating voltage of 5 volts and can accept an input voltage ranging from 6-20. It provides a DC current of 50 mA per I/O pin.

**REQ240 Terminal assignments for Bourne Potentiometer:** Each of the Bourne's potentiometer's terminals shall have 3 connections: one to the 5V pin of the Arduino, one to an analog pins of the Arduino, and one to the Ground pin of the Arduino. . The Bourne 3852A-282-103AL 10 k $\Omega$  1-Turn Rotary Potentiometer's terminal shall follow the designation shown in **Fig. 6**.



**Fig. 6: Terminal Assignment for Bourne potentiometer**

**REQ250 Terminal assignments for Jameco Potentiometer** Each of the Jameco potentiometer's terminals shall be connected to the 5V pin of the Arduino, one if the analog pins of the Arduino, and to the Ground pin of the Arduino. The Jameco ValuePro 3329P-1-103 ½ Watt, ¼ Inch, 10kΩ Round Cermet PPotentiometer's terminal will follow the designation shown in **Fig. 7**



**Fig. 7: Terminal Assignment for Jameco potentiometer**

**REQ260 Terminal assignments for Piher Potentiometer** Each of the Jameco potentiometer's terminals shall have 3 connections: one to the 5V pin of the Arduino, one to an analog pins of the Arduino, and one to the Ground pin of the Arduino. The Piher PTC10 10kΩ Round Cermet Potentiometer's terminal shall follow the designation shown in **Fig. 8**.

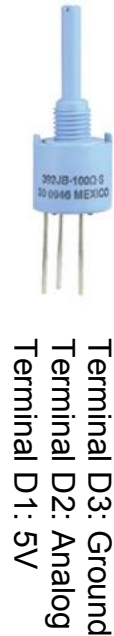
Terminal C1: 5V  
Terminal C2: Analog  
Terminal C3: Ground



**Fig. 8: Terminal Assignment for Piher potentiometer**



**REQ270 Terminal assignments for Honeywell Potentiometer** Each of the Honeywell potentiometer's terminals shall have 3 connections: one to the 5V pin of the Arduino, one to an analog pins of the Arduino, and one to the Ground pin of the Arduino. The 10K Ohm 10% 1/2W 3.18mm Potentiometer's terminal shall follow the designation shown in **Fig. 9**.

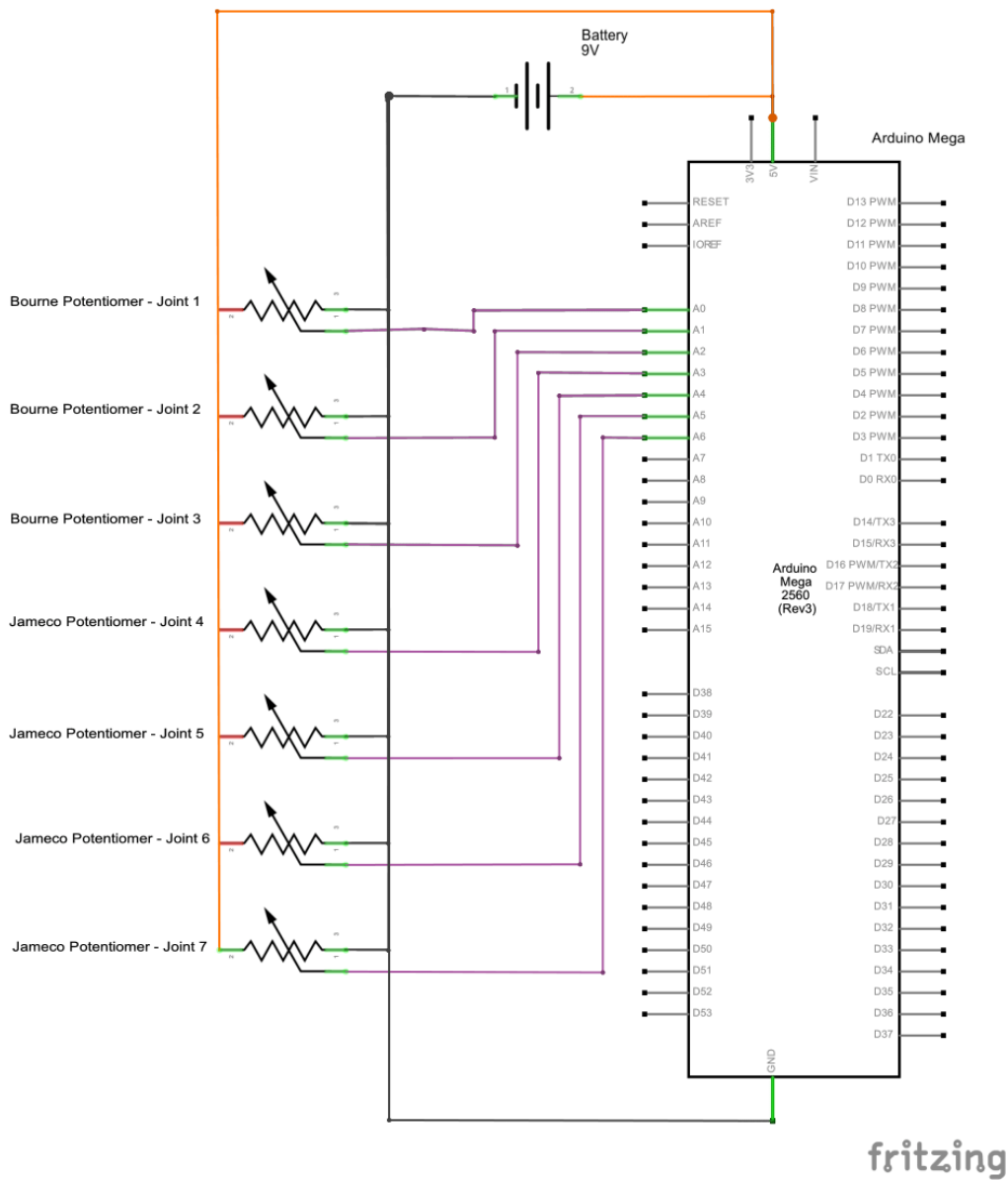


**Figure 9: Terminal Assignment for Piher potentiometer**

**REQ280 Circuit Design:** The potentiometers in the controller shall be wired based on the circuit diagram shown in **Fig.10**

**BACK280** The potentiometers will be connected in parallel with each other in order to provide equal voltage across each potentiometer.

**BACK281** The sketch was made using Fritzing, a software used for making wiring diagrams for Arduino



**Fig. 10: Circuit Diagram of the Controller & Arduino Mega**

### **5.3 Arduino Software Requirements**

This section provides the function requirement of the software that will provide the interface between the controller and MATLAB. The requirements are composed of 4 modules, which include assign variables, initialize the serial communication, read the voltages and send them to MATLAB through serial communication.

**REQ310 Assigning Variables:** The Arduino software shall create 16 different variables, seven (7) analog pins, seven(7) bit values, and two (2) different markers, one specifying the beginning and the other one specifying the ending of the data.

Joint Number	Pins	Unsigned integer 16 – Analog Pins	Variable = integer Bits
1	A0	Potentiometer1	Bits1
2	A1	Potentiometer2	Bits2
3	A2	Potentiometer3	Bits3
4	A3	Potentiometer4	Bits4
5	A4	Potentiometer5	Bits5
6	A5	Potentiometer6	Bits6
g (gripper)	A6	Potentiometer7	Bits7

Beginning Marker	End Marker
Int A=1999	Int B=2999

**REQ320 Initialize the Serial Communication of Arduino:** The software shall set the data in bits per second (baud) for serial data transmission using the syntax Serial.begin (speed). The baud rate is 115,200. Refer to **Flow Chart IARMCDD24-IARMCDD25** for initialization of baud rate.

**REQ330 Read the Analog Inputs:** The software shall read the 7 analog input values, which will be stored as integers refer to **Flow Chart IARMCDD24-IARMCDD25**, from potentiometers at each joint by using the analogRead( ) refer to **Document analogRead** function of Arduino.

**REQ340 Send the bits readings to MATLAB:** The Arduino software shall send the end marker first then the seven different bits readings and lastly send the beginning marker from the Arduino to MATLAB using the syntax Serial.println(). Refer to **Document Serial.println** function of Arduino.

## 5.4 MATLAB Software Requirements

This section provides the function requirements of the software that will provide the interface between Arduino and MATLAB. The requirements are composed of 7 modules which include, establishing serial communication between MATLAB and Arduino, establishing serial communication between MATLAB and the iARM, initialize the iARM, receiving values from Arduino, converting the bits to angles, reformatting to string, and turning off the controller and IARM.

**REQ410 Initialize the Serial Communication from Arduino:** The software shall assign the baud rate and the communication port for the Arduino. The baud rate is 115,200. Refer to **Flow Chart IARMCDD24-IARMCDD25** for initialization of baud rate.

**REQ420 Initialize the Serial Communication of the Iarm:** The software shall assign the baud rate, the communication port, stopbits, parity, and the flowcontrol of the iARM. The baud rate is 115,200, flowcontrol is none, parity is none, and databits is 8. Refer to **Flow Chart IARMCDD24-IARMCDD25 and the iARM manual.**

**REQ430 Turning on the iARM:** Once the iARM is on, the software shall send a command to the iARM, to “unfold” refer to **Document Exact Dynamics** by sending the string “u” using the syntax fprintf (file ID, format, string) refer to **Document fprintf.**

**REQ440 Receiving Values from Arduino:** The software shall receive the 7 different analog bit values as well as the starting and ending integers and store them in an array. The software uses the syntax fscanf(). Refer **Flow Chart IARMCDD24-IARMCDD25 and Document fscanf().**

**REQ450 Converting Bits:** The software shall convert the data bits read at each joint of the controller to a voltage using the equation below: refer to **Flow Chart IARMCDD24-IARMCDD25.**

$$\text{Voltage} = (5/1023) * \text{Bits}$$

**REQ460 Converting Voltages:** Using the linear equation  $y=mx+b$ , where m is the ratio of the Angles to the voltages values, x is the voltage received from the potentiometers, b is the offset of the potentiometers, and y is the corresponding joint angle shown in **Fig.11**. The input is the voltages read from each potentiometer and the output is the angle corresponding to each join. The values of m, y, and b vary depending on each joint. Each value of **Table 5** shall be defined as float-type variables in the software refer to **Document Float.**

$$\text{Angle} = m * \text{Volt} + b$$

**Fig. 11: Example of Conversion to Angle**

**Table 5: Angle Limits, Angle-Bit Conversion Factor, and Angular Offset**



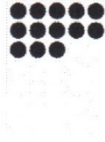

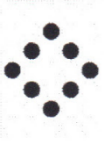

Joint	Min. Angle	Max Angle	m	b
1	0	300	0.29297	2525.3
2	0	300	0.29297	1328.12
3	0	300	0.29297	1909.06
4	0	300	0.29297	725.29
5	0	280	0.27344	2607.74
6	0	120	0.11718	1141.56
7	0	100	0.09765	20.12

**REQ470 Analyzing Outputs:** If the angle value produced by the algorithm is within the range shown in Table 4, the angle shall be sent as a command to the iARM by using the function fprintf (file ID, format, string) refer to **Document fprintf()**. If the angle is not within the range, the maximum angle of the joint shall be sent.

**REQ480 Command Syntax:** These commands assign goal position at each of the joints of the iARM. **JP** establishes the command type which is the joint position command, and it accepts seven parameters: six for specification of the amount of joint rotation, and one for specifying the



**Fig. 14: iARM Components**

					
Start-up	Electronics self-test	(Animated) Joint/motor self-test	(Flashing) Booting of iARM firmware	(Animated) Calibration/encoder read-out	iARM-PC Control is ready for your input

**Fig. 15: iARM Monitor Status Display**

**REQ530 Controlling the iARM:** The user shall move the controller to its desired position by inserting their fingers in the ring slots of the end effector as shown in **Fig. 3**.

## 7 Performance

This section should measure the efficiency of the controller.

**REQ610 Minimal Delay:** The MATLAB -Controller software shall send commands to the iARM within 380ms.

**BACK510** The controller's delay should only be 30ms and the iARM has a set delay of 350ms.

**REQ620 System Response:** The iARM will mimic the movement of the controller within 380 milliseconds.

## 8 Safety

This section should provide the user with a safety manual in which it will specify the limits of the controller

### 8.1 Mechanical Safety Requirements

**REQ710 Rotational Limits:** The controller shall have mechanical stops within each joint segment in order to prevent the complete rotation of the device.

**REQ720 Joint Locks:** Extrusions shall be present at specific locations of the controller to prevent joint locking of the controller to prevent it from being stuck on a certain position.

**REQ730 Initial Position Setting:** Attachments shall be placed at different locations at each joint of the controller in order for the controller to mimic the initial unfold position of the robot.

## **8.2 Electrical Safety Requirements**

**REQ810 Wire Conduits:** The wires of the controller shall be covered by Gardner Bender ½ inch spiral wrap to prevent damage and contact with the environment.

**REQ820 Arduino Case:** An Arduino case shall be designed in order to protect the Arduino from external damage such as water and physical contact.

## **8.3 Software Safety Requirements**

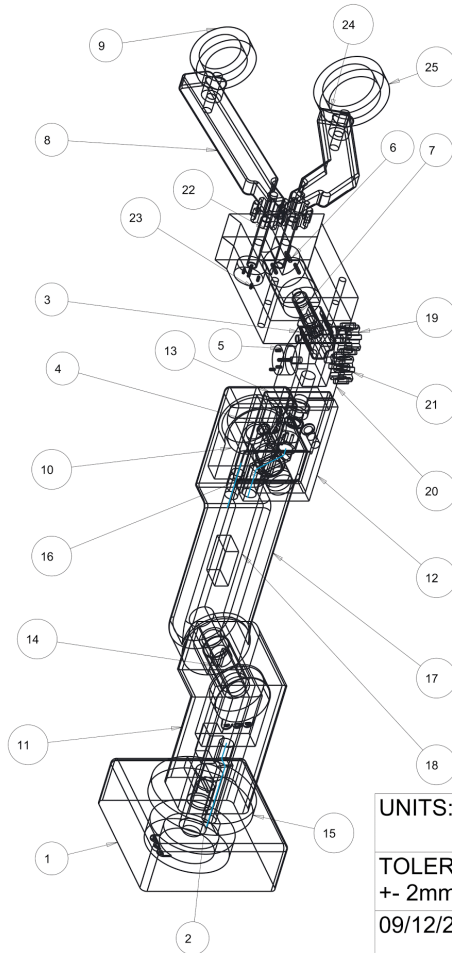
**REQ910 Software Delay:** The Arduino software shall have a delay, in which the iARM would be able to read all the angles that are sent, without overriding any other commands.

**REQ920 Software Safety Stop:** The program shall have an if statement that states if the angle is greater than the permitted joint angles along with the end effector then the software shall send the maximum allowed angle for those joints as shown in **Table 5**.

## **9 Documentation**

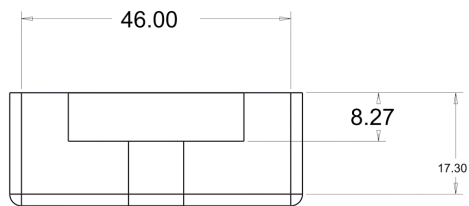
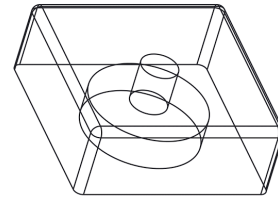
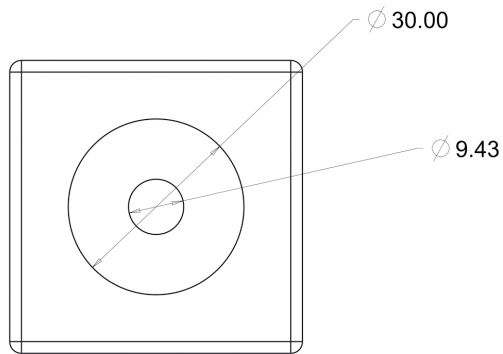
**REQ1010:** Documentation shall provide the schematics of the potentiometers, Arduino Mega, the drawings of each part of the controller the circuit diagram, and the Software Flowchart.

ITEM	DESCRIPTION	QTY
1	BASE	1
2	BASE_SEG1	1
3	CROSS	1
4	MINIPOT	1
5	NEW_POT	2
6	NEW_POT2	1
7	PIN_SEG5_SEG4	1
8	POINTER	1
9	POINTER_RING	1
10	POTENTIOMETER	3
11	SEG1B-TEST	1
12	SEG_3_P2_SEDIT	1
13	SEG_3_SEDIT	1
14	SEG1_2	1
15	SEG1A-TEST	1
16	SEG2_3	1
17	SEG2_PART1	1
18	SEG2_PART2	1
19	SEG4-1	1
20	SEG4_PART1	1
21	SEG4POT	1
22	SEG5-END	2
23	SEG5_PART1	2
24	THUMB	1
25	THUMB_RING	1

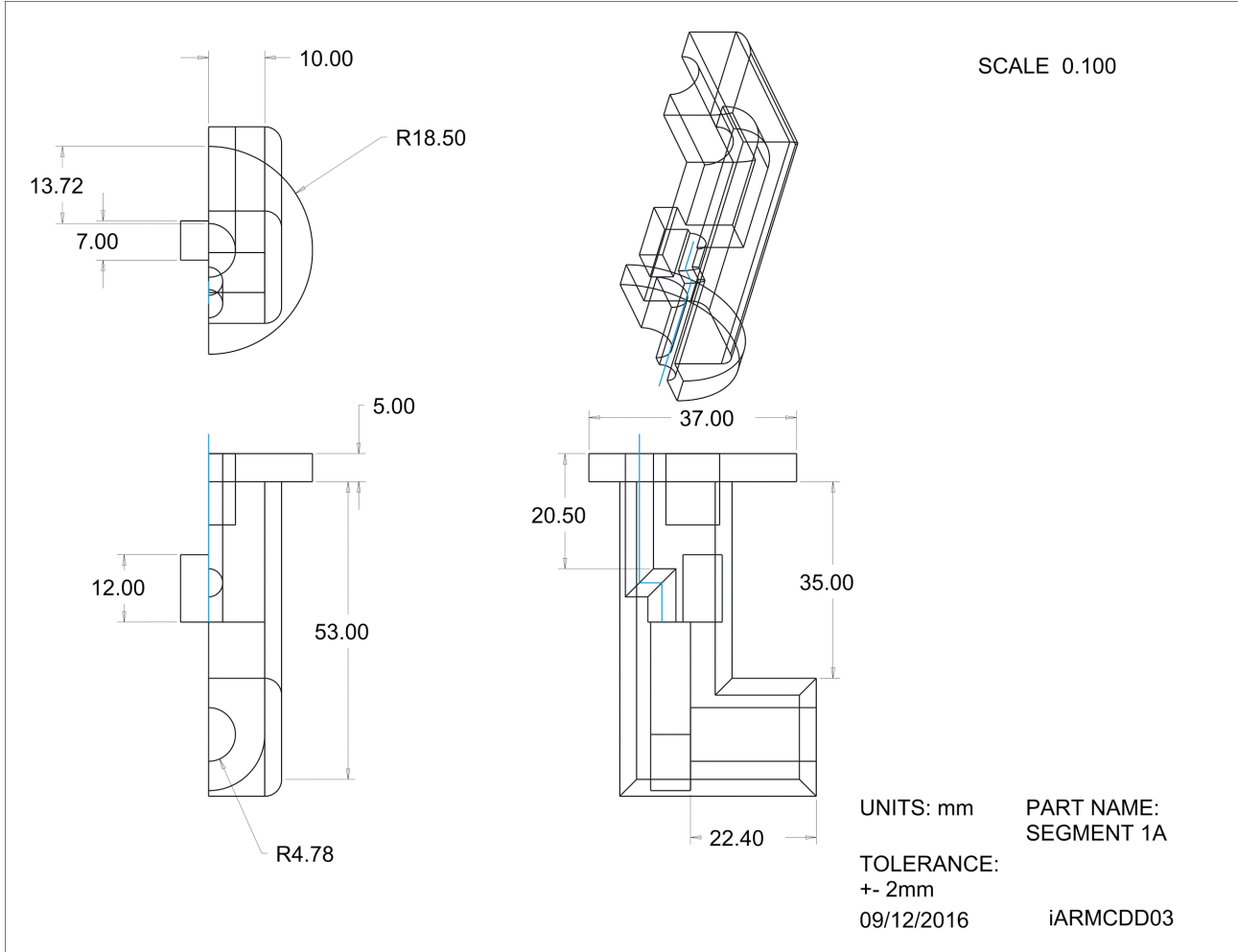


UNITS: mm	PART NAME: PROTOTYPE
TOLERANCE: +- 2mm	
09/12/2016	iARMCDD01

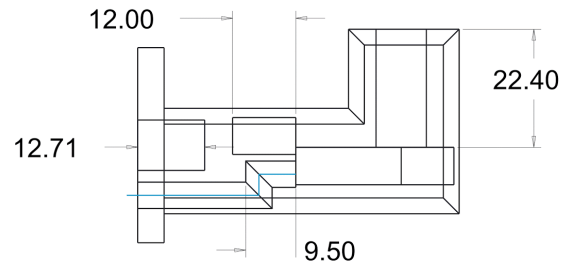
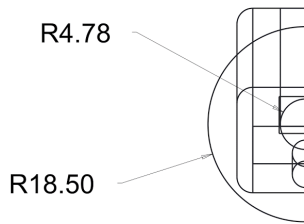
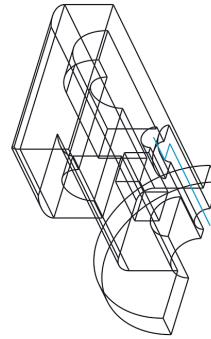
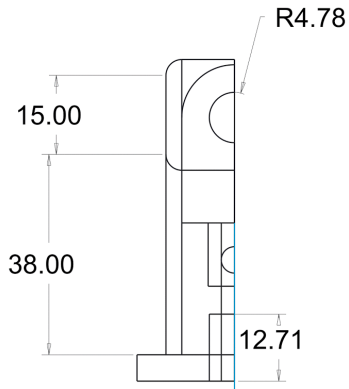




UNITS: mm	PART NAME: BASE
TOLERANCE: +- 2mm	
09/12/2016	iARMCDD02



SCALE 0.090



UNITS: mm

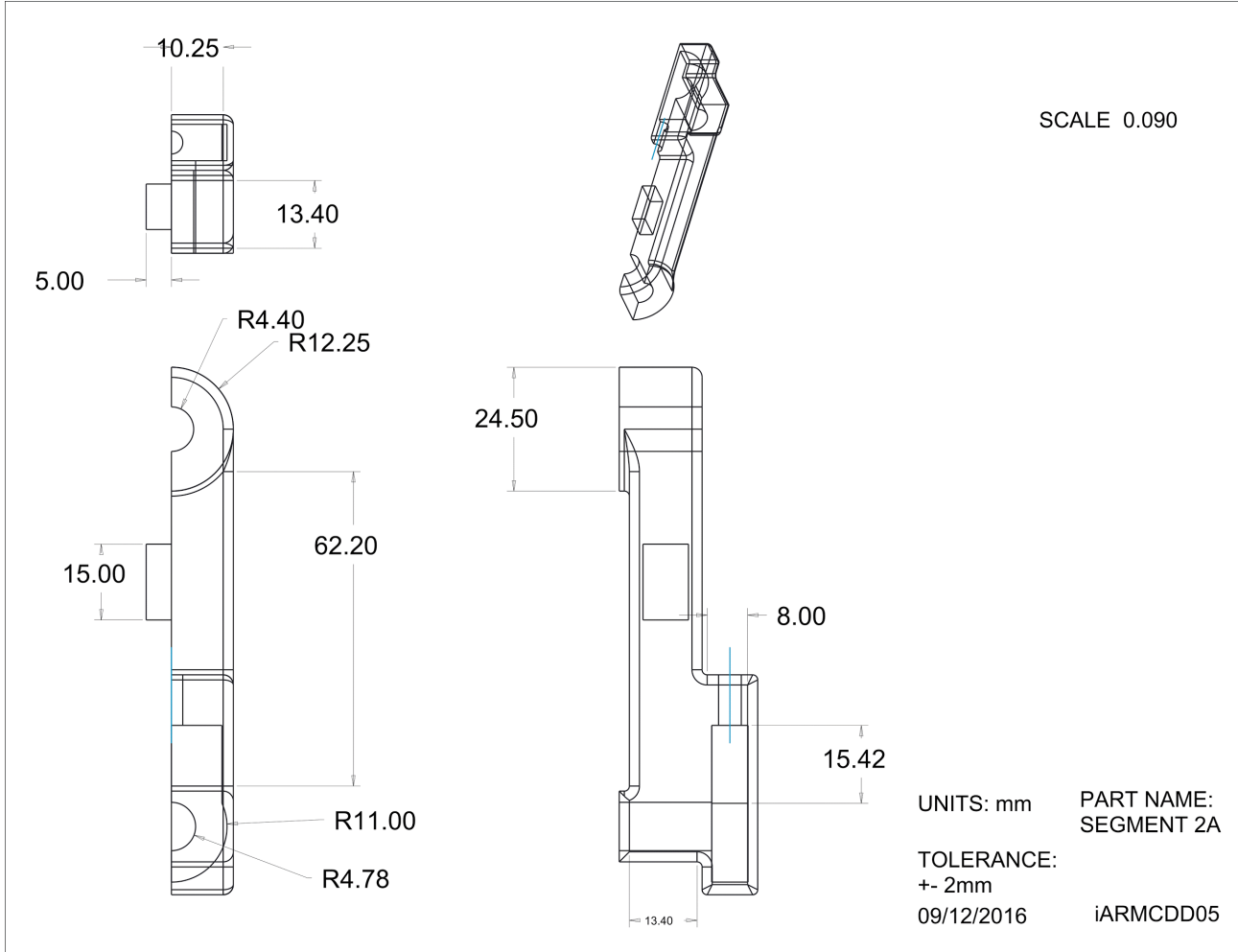
PART NAME:  
SEGMENT 1B

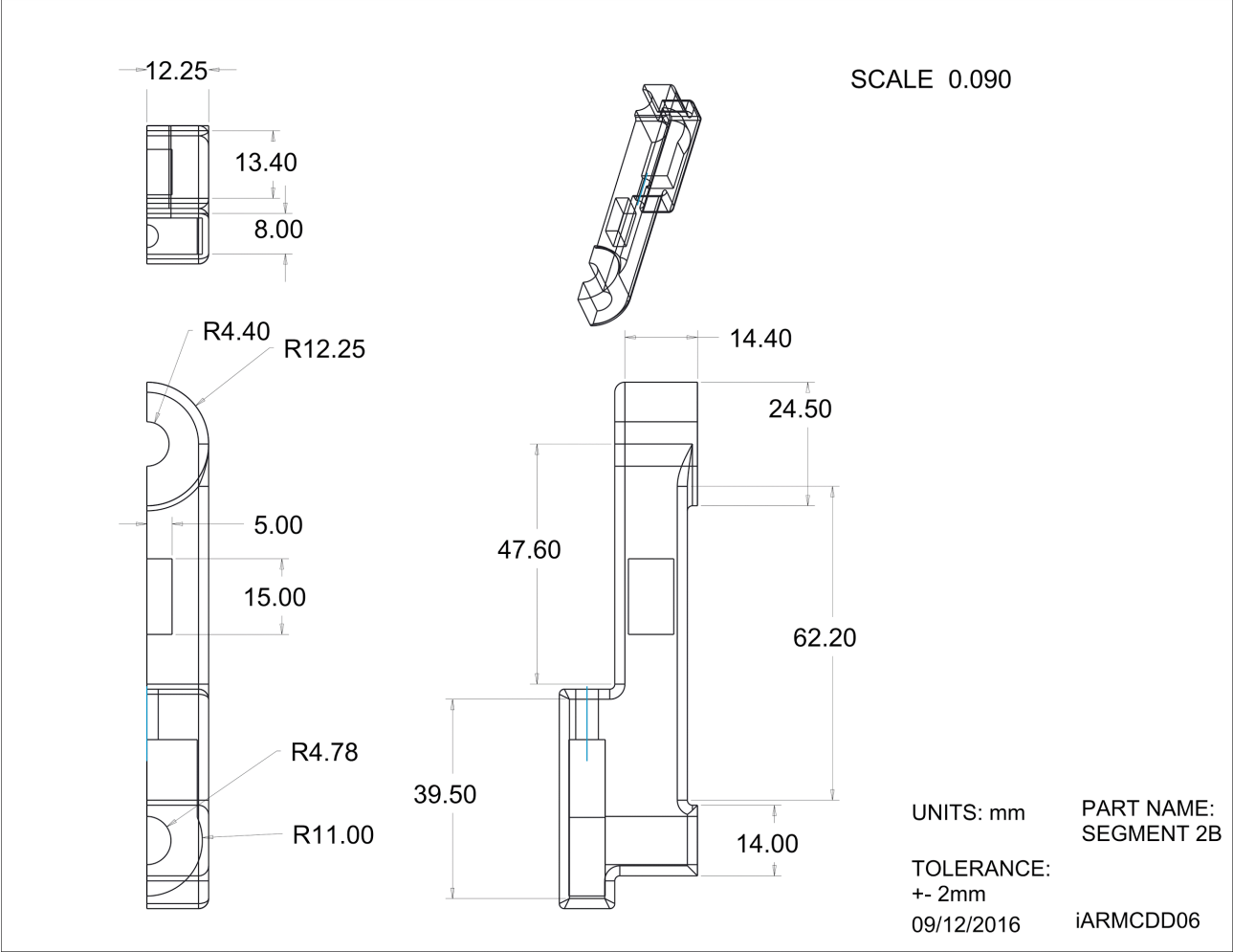
TOLERANCE:

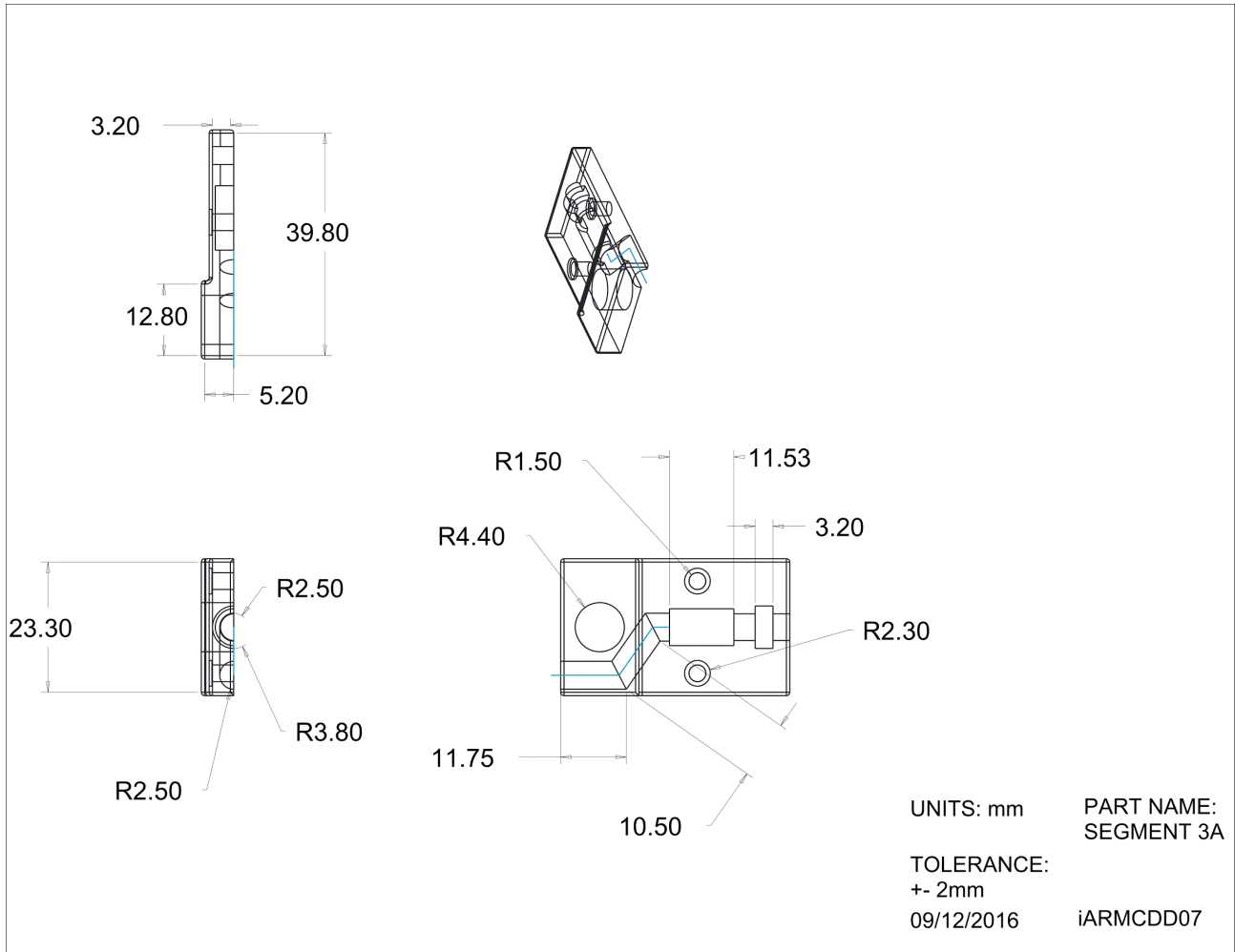
+/- 2mm

09/12/2016

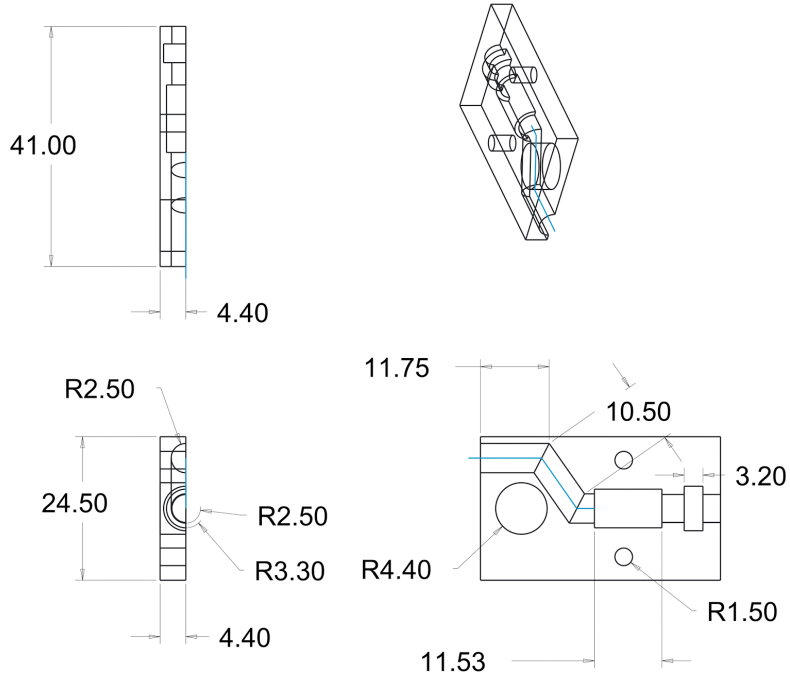
iARMCDD04



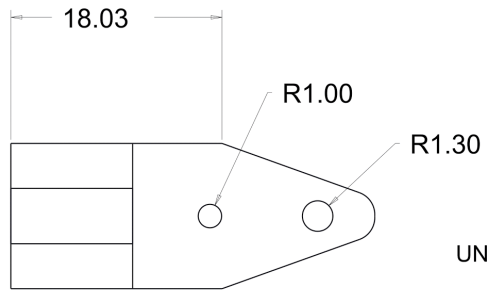
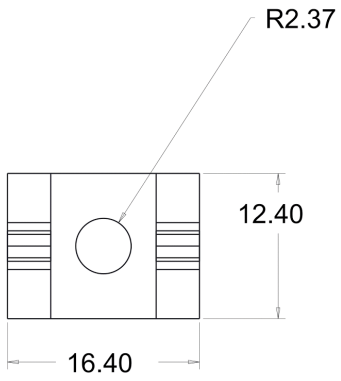
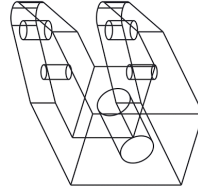
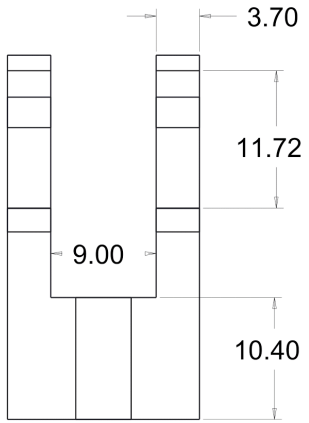




SCALE 0.100



UNITS: mm      PART NAME:  
                         SEGMENT 3B  
TOLERANCE:  
+/- 2mm  
09/12/2016      iARMCDD08



UNITS: mm

PART NAME:  
SEGMENT 4

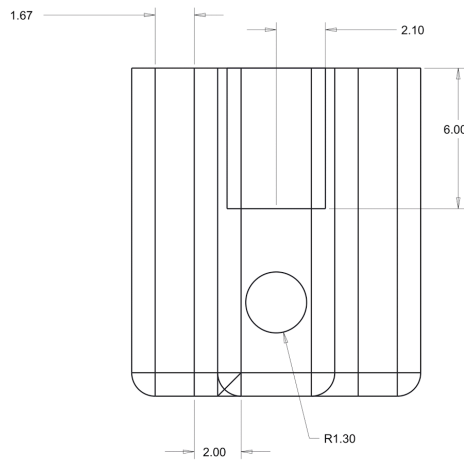
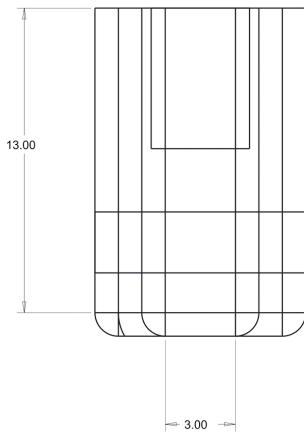
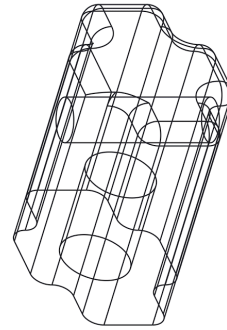
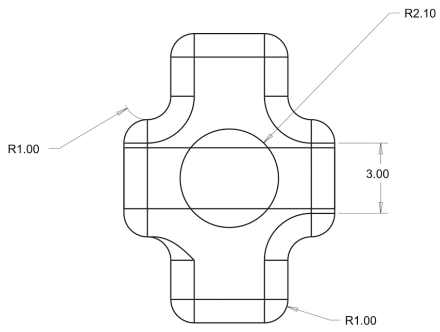
TOLERANCE:  
+- 2mm

09/12/2016

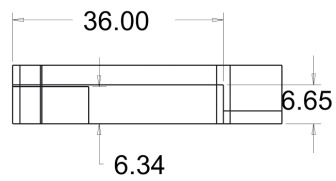
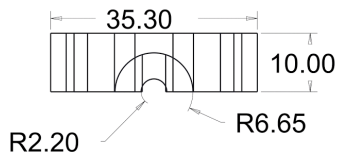
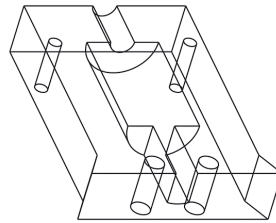
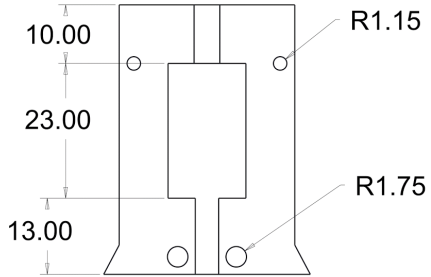
iARMCDD09



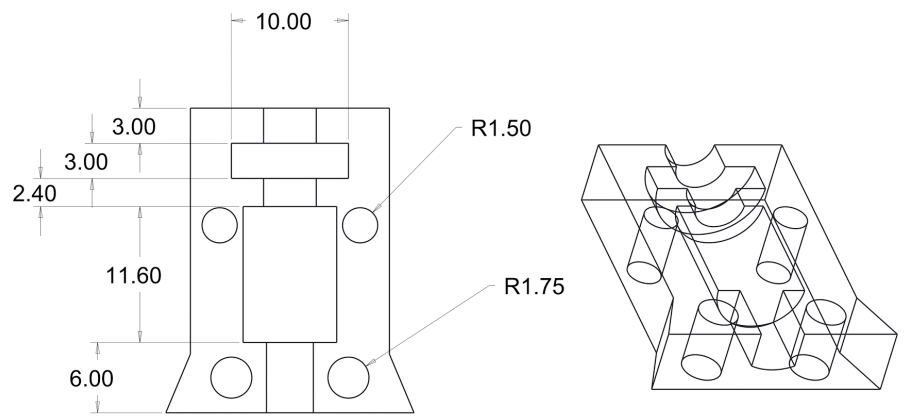
SCALE 0.400



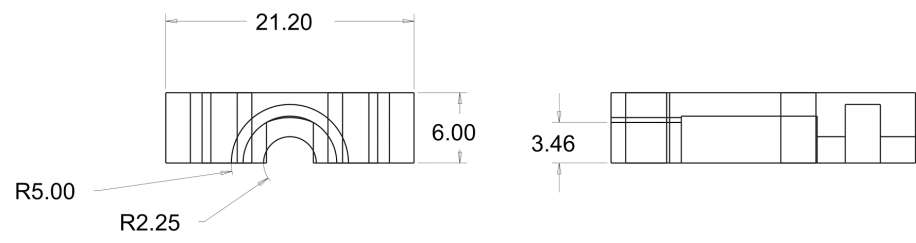
UNITS: mm    PART NAME: CROSS  
TOLERANCE: +- 2mm  
09/12/2016    iARMCDD10



UNITS: mm      PART NAME:  
 SEGMENT 5A  
 TOLERANCE:  
 +- 2mm  
 09/12/2016      iARMCDD11

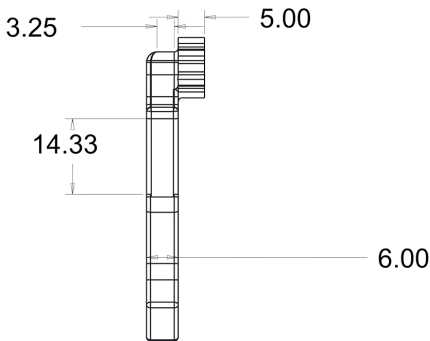
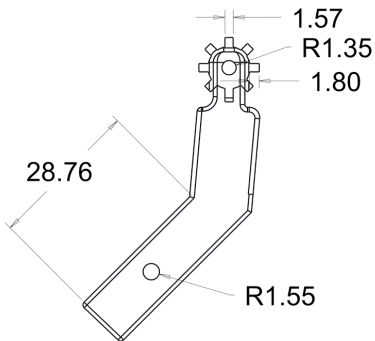
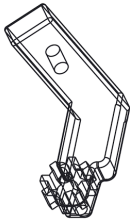
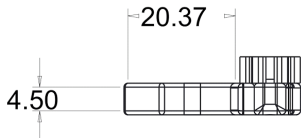


SCALE 0.200



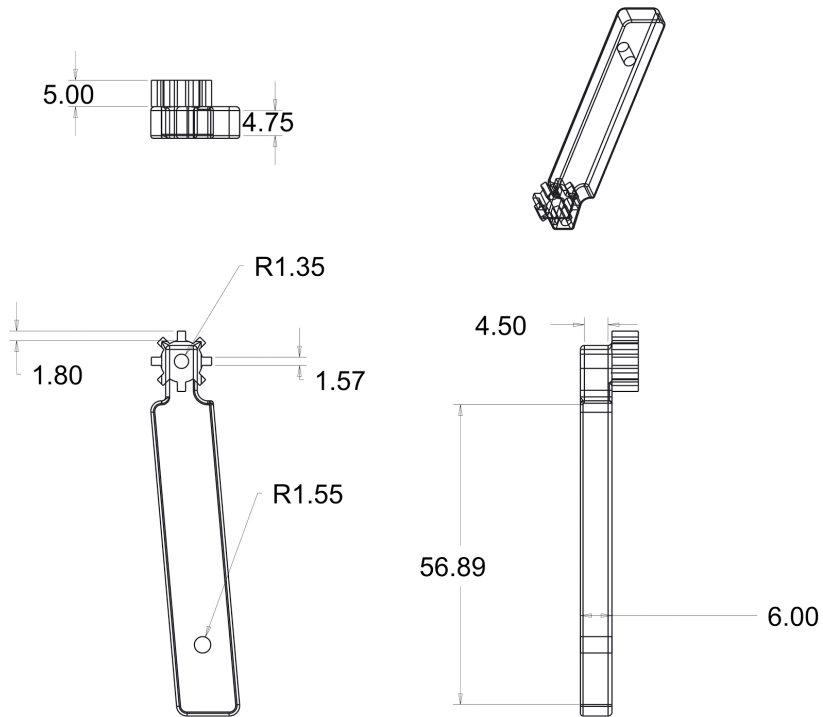
UNITS: mm      PART NAME:  
 SEGMENT 5B  
 TOLERANCE:  
 +- 2mm  
 09/12/2016      iARMCDD12

SCALE 0.090

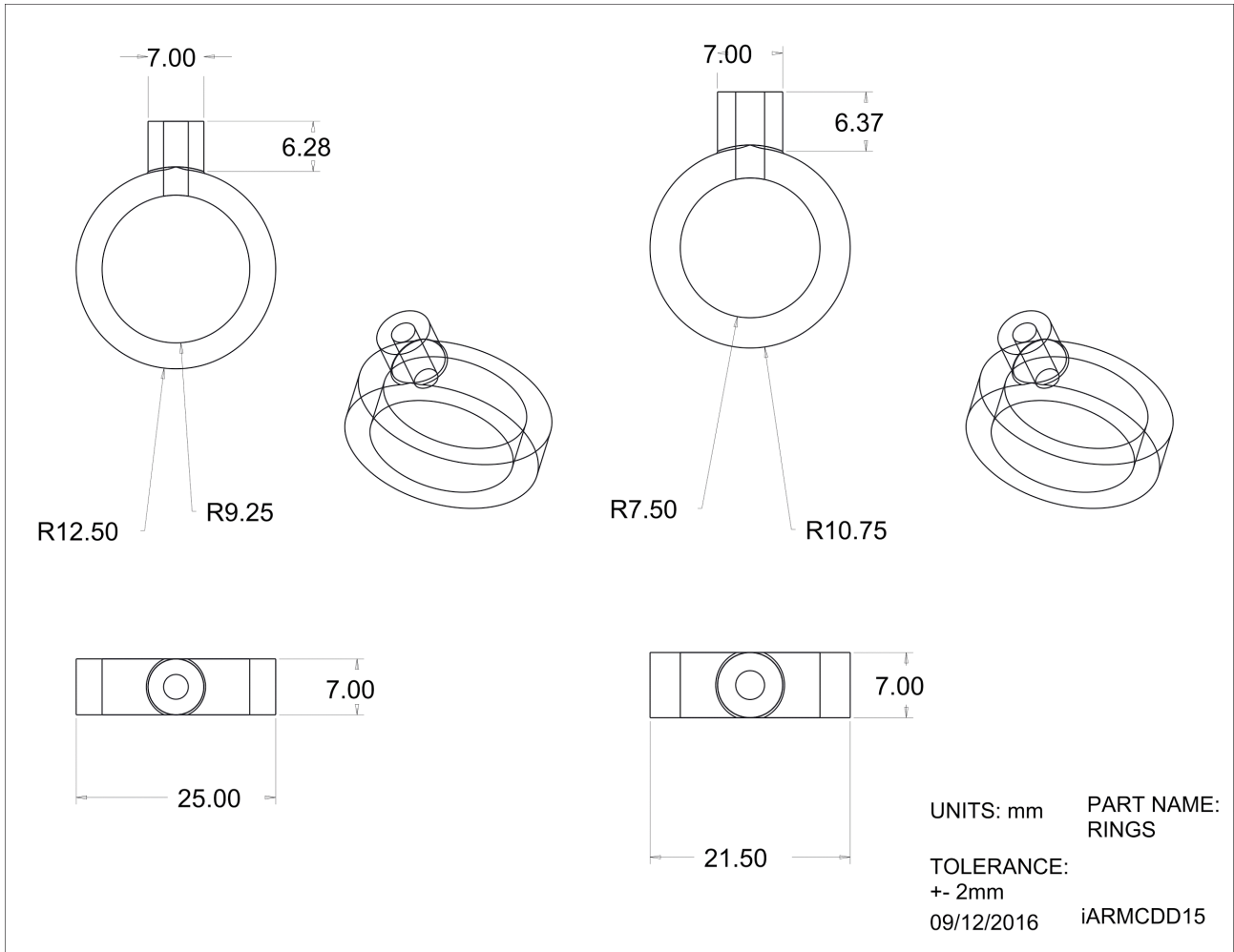


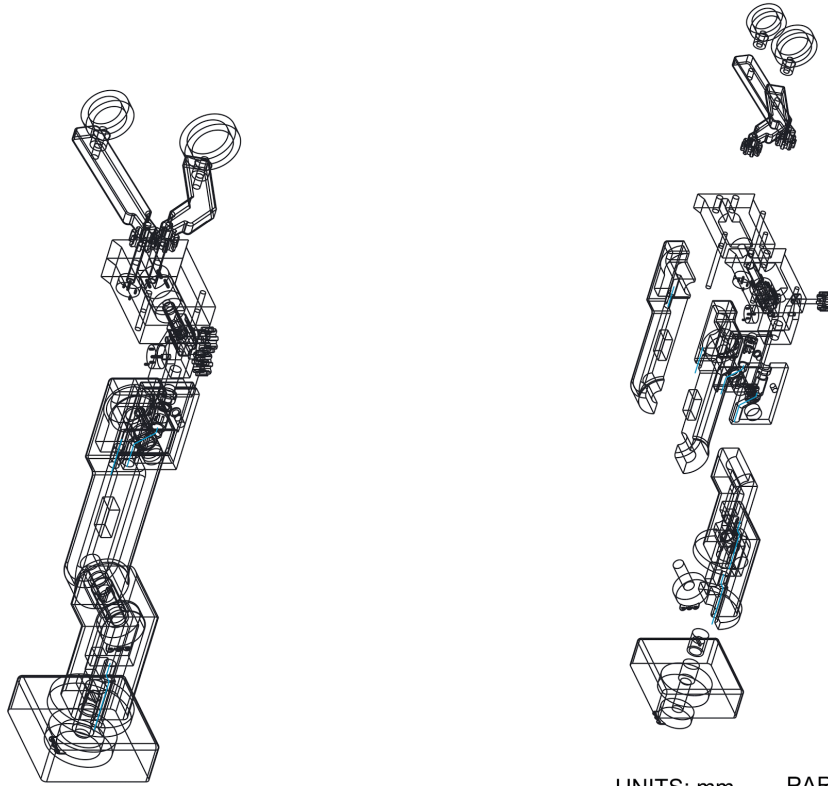
UNITS: mm    PART NAME:  
                 THUMB  
TOLERANCE:  
+/- 2mm  
09/12/2016    iARMCDD13

SCALE 0.090



UNITS: mm      PART NAME:  
                    POINTER  
TOLERANCE:  
+/- 2mm  
09/12/2016      iARMCDD14





UNITS: mm

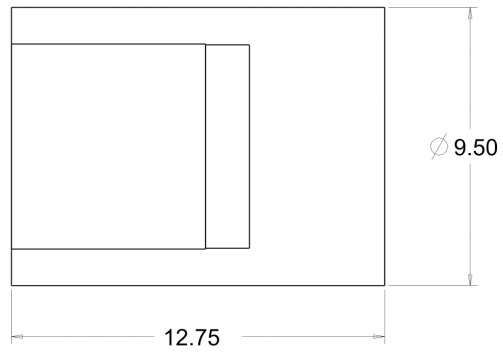
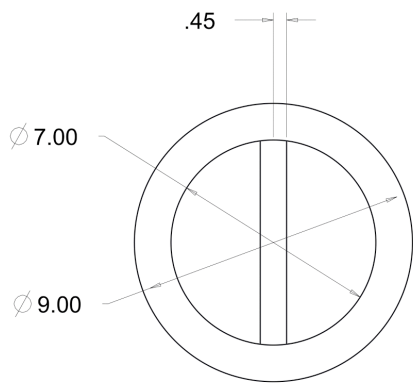
PART NAME:  
Potentiometer Locations

TOLERANCE:

+/- 2mm

09/16/2016

iARMCDD16



SCALE 0.500

UNITS: mm

PART NAME:  
PIN base-Segment 1

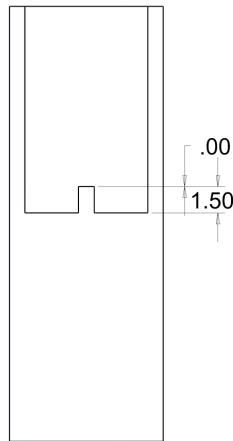
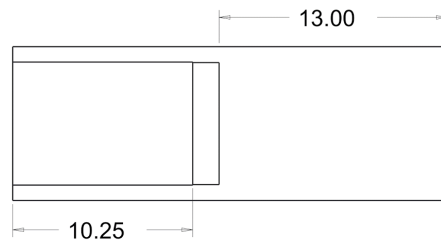
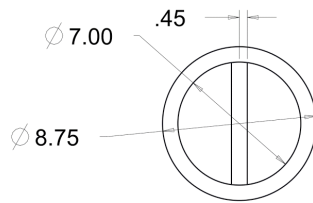
TOLERANCE:

+/- 2mm

09/16/2016

iARMCDD17





SCALE 0.300

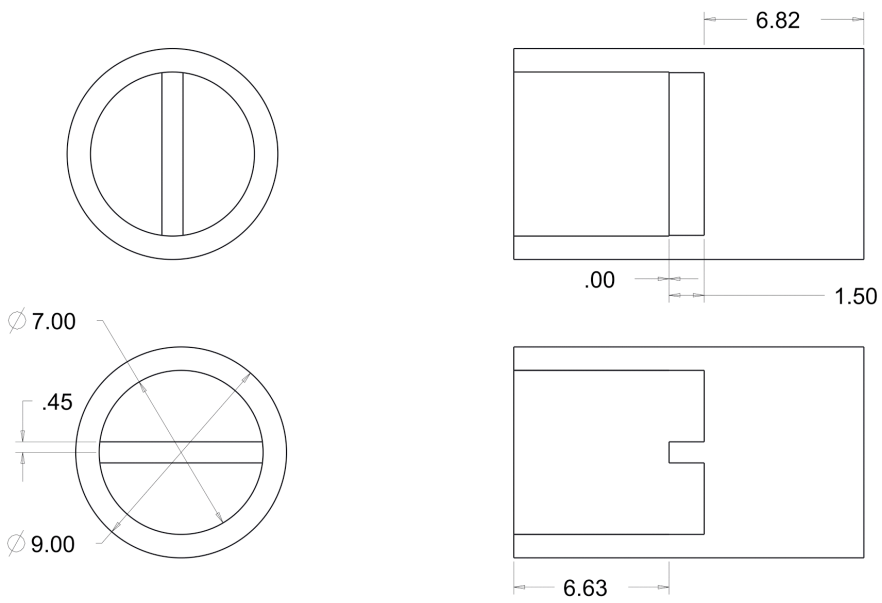
UNITS: mm

PART NAME:  
PIN Segment 1 -2

TOLERANCE:  
+- 2mm

09/16/2016

iARMCDD18



SCALE 0.400

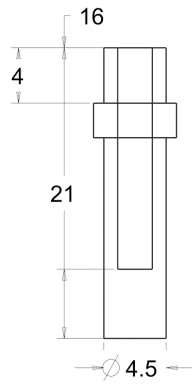
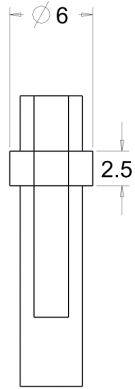
UNITS: mm

PART NAME:  
PIN Segment 2 -3

TOLERANCE:  
+/- 2mm

09/16/2016

iARMCDD19



SCALE 6.000

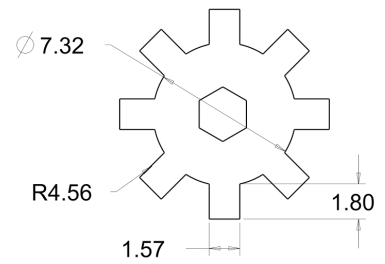
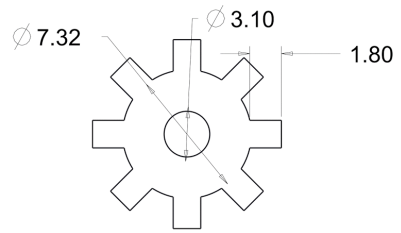
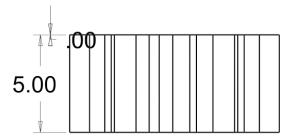
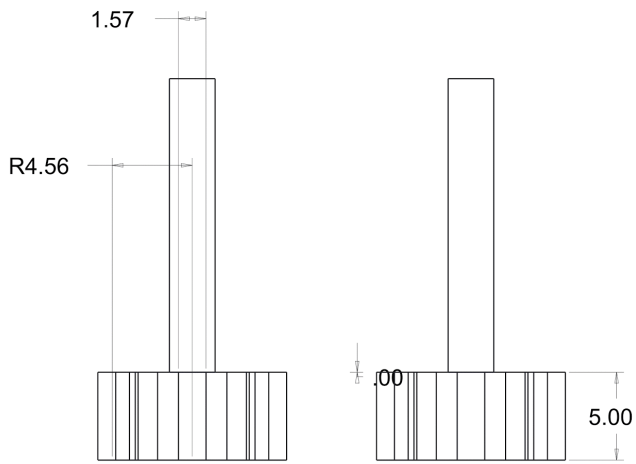
UNITS: mm

PART NAME:  
PIN Segment 3 -4

TOLERANCE:  
+- 2mm

09/16/2016

iARMCDD20



SCALE 0.300

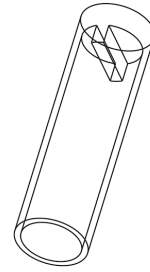
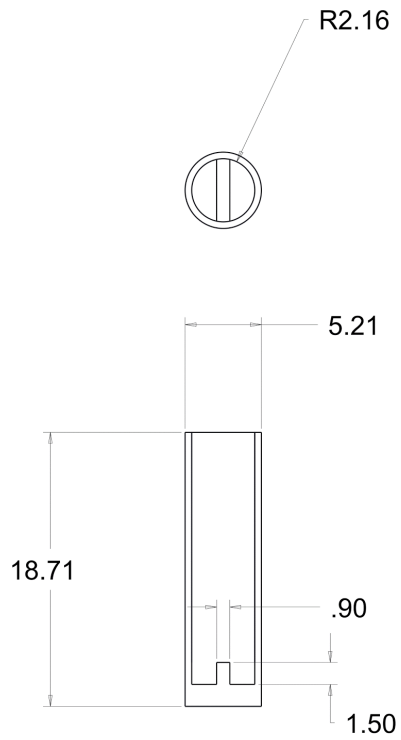
UNITS: mm

PART NAME:  
PIN Segment 4&4-cross

TOLERANCE:  
+- .2mm

09/16/2016

iARMCDD21



UNITS: mm

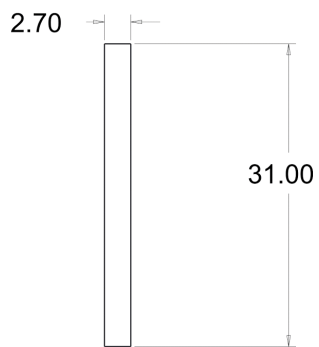
PART NAME:  
PIN Cross-Segment 5

TOLERANCE:  
+- 2mm

09/16/2016

iARMCDD22

R1.35



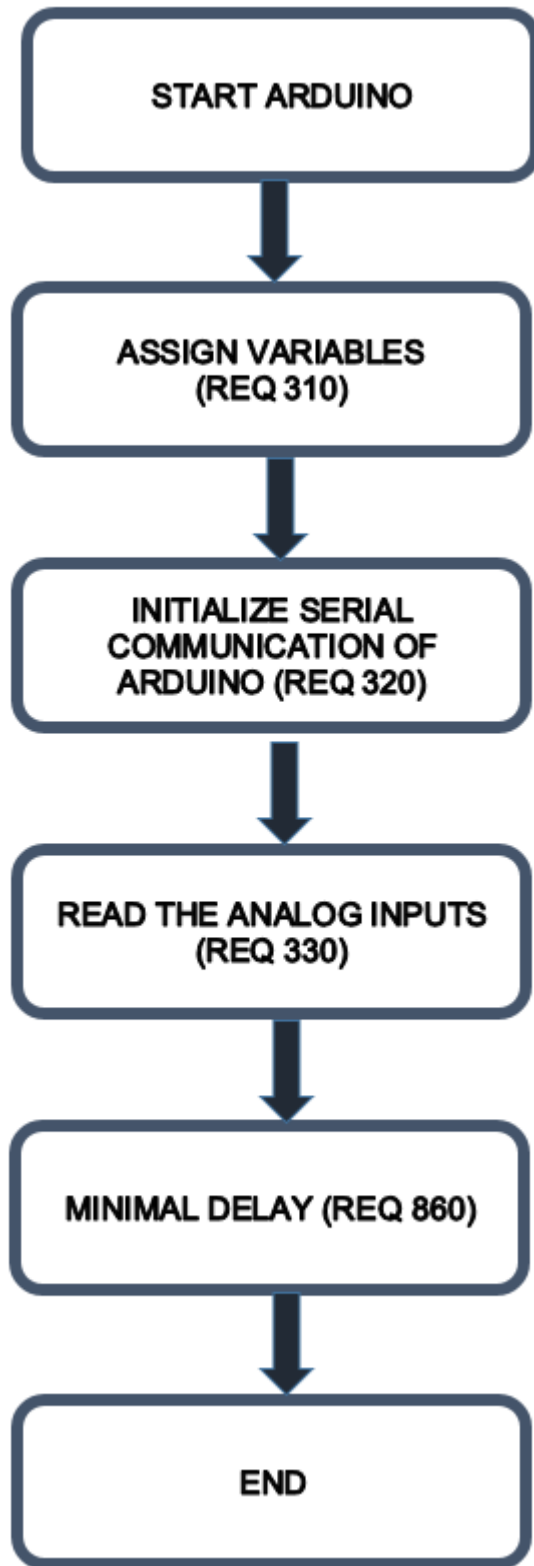
UNITS: mm

PART NAME:  
PIN Segment 5-end

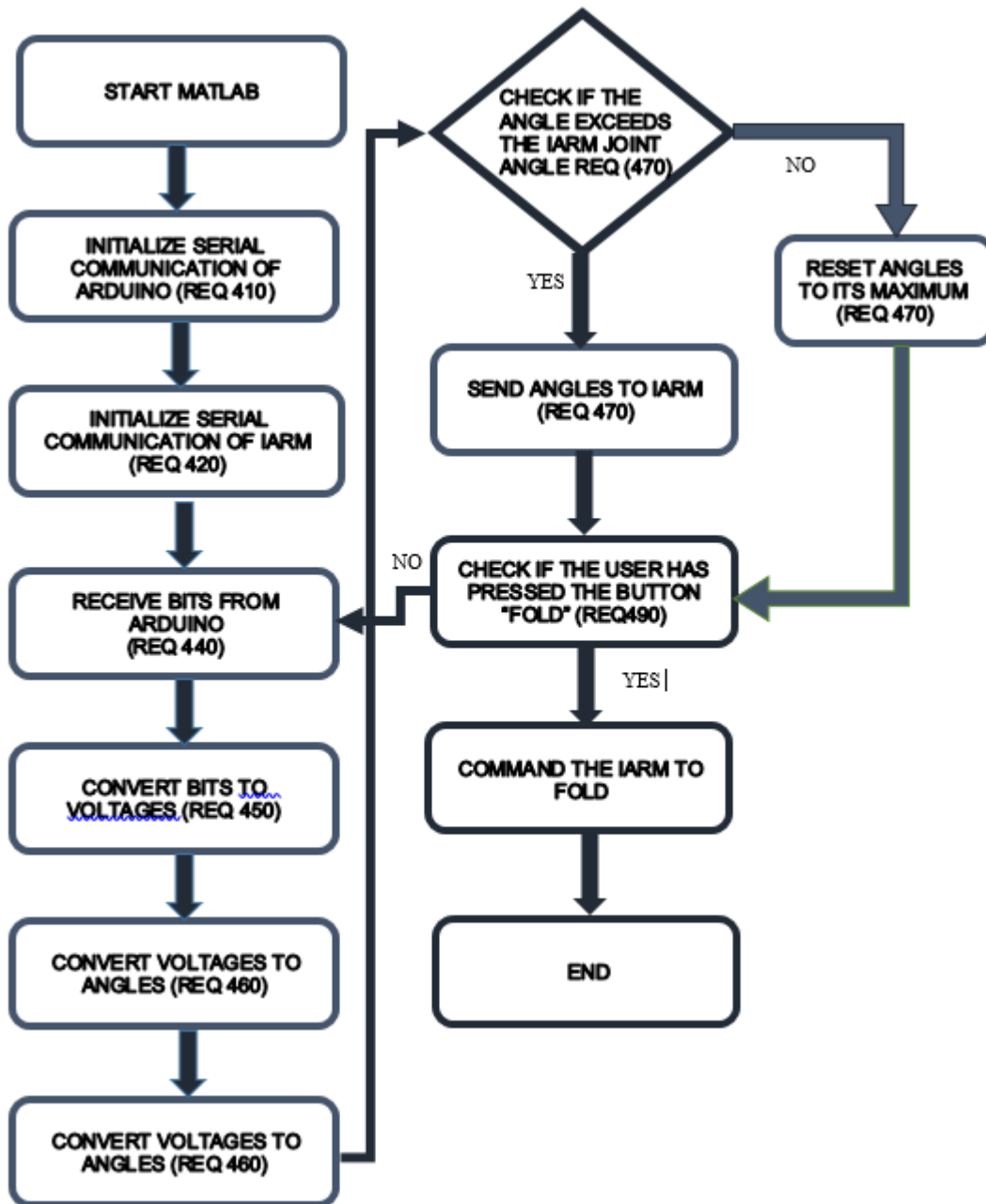
TOLERANCE:  
+- 2mm

09/16/2016

iARMCDD23



iARMCDD24



iARMCDD25