LUKE ARM - HUMAN EMULATING ROBOTIC HAND

PROJECT REPORT

EKLAVYA MENTORSHIP PROGRAM AT SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE , MUMBAI.

SEPTEMBER-OCTOBER 2022.

ACKNOWLEDGEMENT

WE ARE TRULY GRATEFUL FOR THIS WONDERFUL OPPORTUNITY THAT SRA HAS BESTOWED ON US. THANKYOU MENTORS AND SENIORS FOR YOUR GUIDANCE, SUPPORT, MOTIVATION. IT WAS IMPOSSIBLE TO ACHIEVE THIS WITHOUT YOU ALL. WE'LL LOVE TO WORK AND ACHIEVE FUTURE GOALS TOGETHER.

SPECIAL THANKS TO ;

OM SHELADIA
RISHIKESH DONADKAR

TEAM MEMBERS:

KAMAKSHI DHOKEY
+8879640952
k.dhokey@gmail.com

VAIDIC GUPTA
+918857826380
21vaidicgupta@gmail.com

1] PROJECT OVERVIEW :

In this project we will establish connection between two ESP32 boards (mainly focusing on the ESP-NOW protocol). One board will be the master and the other the slave, controlling servos as per command.

2] TECHNOLOGY USED:

1) **ESP-NOW**

ESP-NOW, is a protocol developed by Espressif. It is a fast communication protocol that can be used to exchange small messages (up to 250 bytes) between esp32 boards.

The protocol is similar to the low-power 2.4GHz wireless connectivity that is often deployed in wireless mouses. So, the pairing between devices is needed prior to their communication.

After the pairing is done, the connection is secure and peer-to-peer, with no handshake being required.

ESP-NOW is versatile and you can have one-way or two-way communication between two esp32 boards.

Why use ESP-NOW?

● It allows ESP devices to communicate directly without connecting to a WiFi network

● The pairing between devices is needed prior to their communication. After the pairing is done, the connection is secure and peer-to-peer. If suddenly one of your boards loses power or resets, when it restarts, it will automatically connect to its peer to continue the communication.

It does not require a router to connect two esp32 boards.Thus, it can be used at any remote places.

● It provides encrypted and unencrypted unicast communication.

● It does not need to connect to a Wifi access point which makes it a fast communication protocol .

● Maximum 20 nodes can be connected.

2) FLEX Sensors

Flex sensor is a variable resistor like no other. Its resistance value changes as we bend its body.

They are available in two sizes:

1) 2.2”

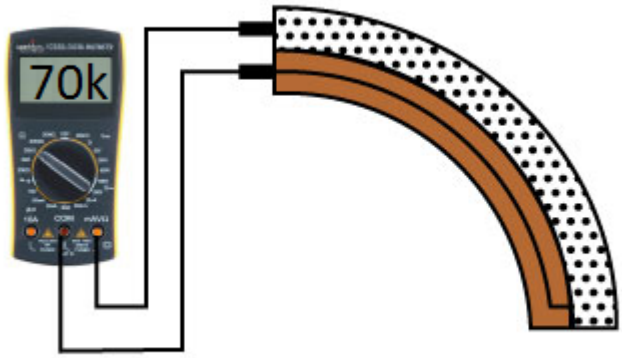
2) 4.5”

In our project we are using the 2.2” flex sensor

Working of Flex sensors-

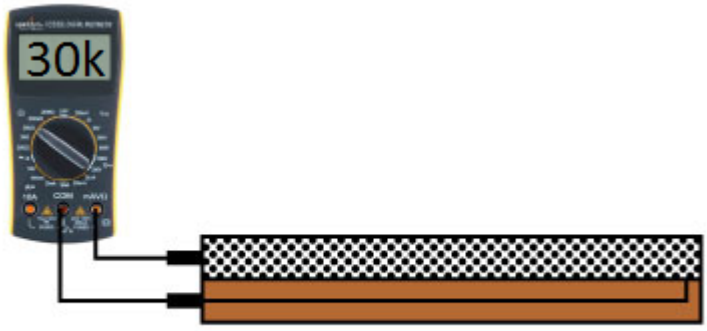
One side of the sensor is printed with a polymer ink that has conductive particles embedded in it. When the sensor is straight, the particles give the ink a resistance of about 30k Ohms. When the sensor is bent away from the ink, the conductive particles move

further apart, increasing this resistance (to about 50k-70K Ohms when the sensor is bent to 90°, as in the diagram below).



Conductive particles further apart - 70kΩ.

When the sensor straightens out again, the resistance returns to the original value. By measuring the resistance, you can determine how much the sensor is being bent.



Conductive particles close together - 30kΩ.

To study further about the flex sensors you can visit the following Links:

1. FreeRTOS

3] PROJECT IDEA:

Aim of the project is designing a cost effective arm housing 5 servos controlled by one slave esp-32 taking commands from a remote esp-32 by establishing wifi communication between the two,essentially one acting as a server and the other as a client.

The user will have to relay finger flex data to the arm and control the servos using client esp accordingly.

4] BASIC PROJECT DOMAINS:

Embedded C

5] WORKFLOW :

1)ESP-NOW protocol:

https://github.com/espressif/esp-idf/tree/master/examples/wifi/espnow

While using the above espnow-example to send data, you will come across a lot
of ‘extras’ on top of espnow data.
These ‘extras’ are not required to send the use the ESPNOW protocol, however to make the data more safe and reliable it is recommended that we use them.

We will require two esp devices for this project.

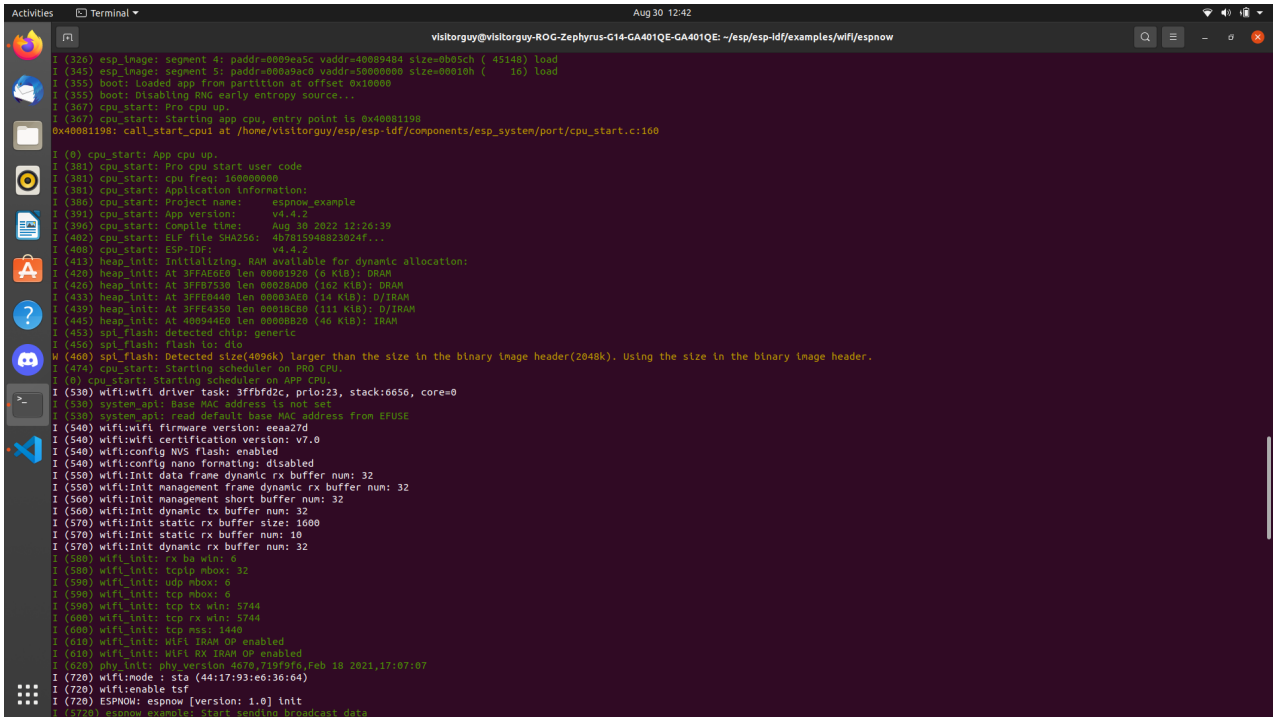
While sending data, these two devices must be on the same channel, must have the same primary and local master key.
The maximum data that can be transferred is of 250 bytes which can be changed under “Example Configuration”.
In our project we will be using the code snippets form the following ESPNOW
code.

https://github.com/Mair/esp32-course/tree/master/_18_ESP_NOW/_18_esp_now

To establish the connection between the two esp-32 we need to know the MAC Addresses of both the ESP-32

To proceed with that we referred the following code,

You should get a similar output,



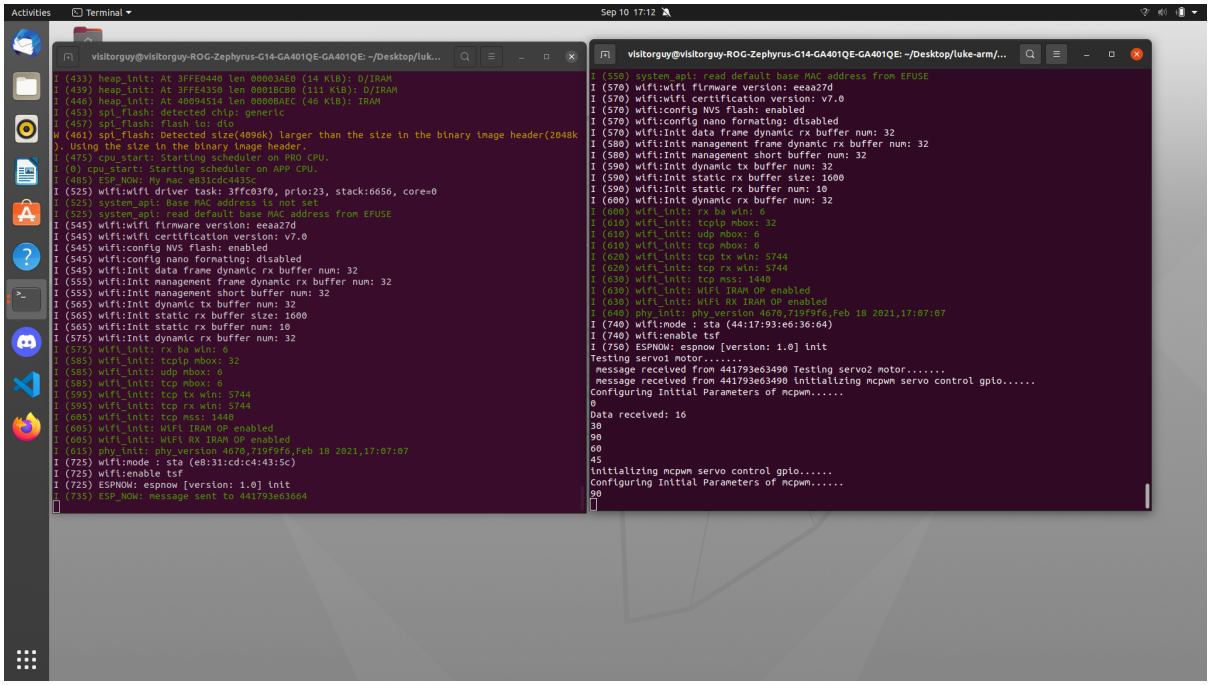
After we have acquired the MAC Addresses successfully we can now add slave and master ESP32
To move ahead with ESP-NOW communication we referred the following code

[ESP-NOW]

2) Controlling Servo Motor Using ESP-NOW protocol

Our next task is to control Servo motor using ESP-NOW
After we are done with ESP-NOW communication we can proceed with

interfacing the servo motor with the esp32
Here the master ESP32 will be sending the mcpwm data to the slave ESP32 using ESP-NOW protocol

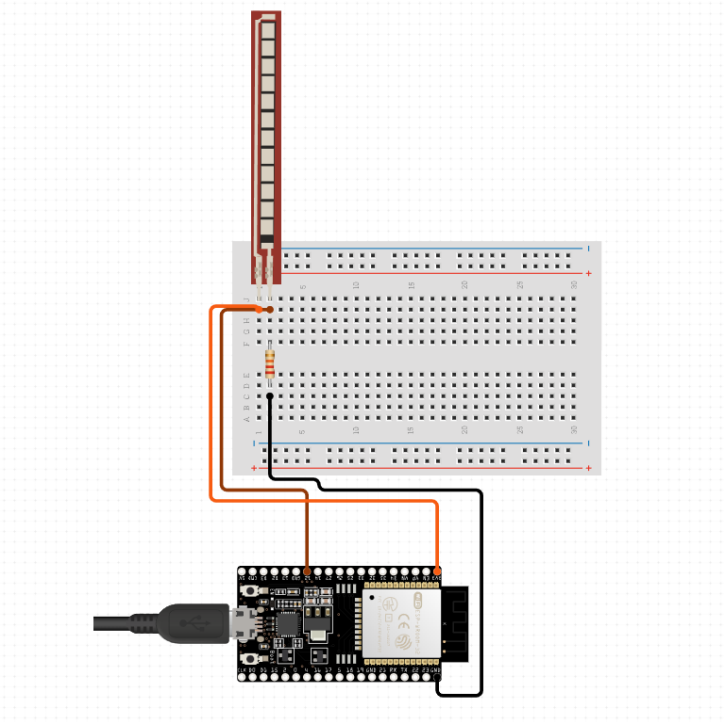


3) Working with flex sensors

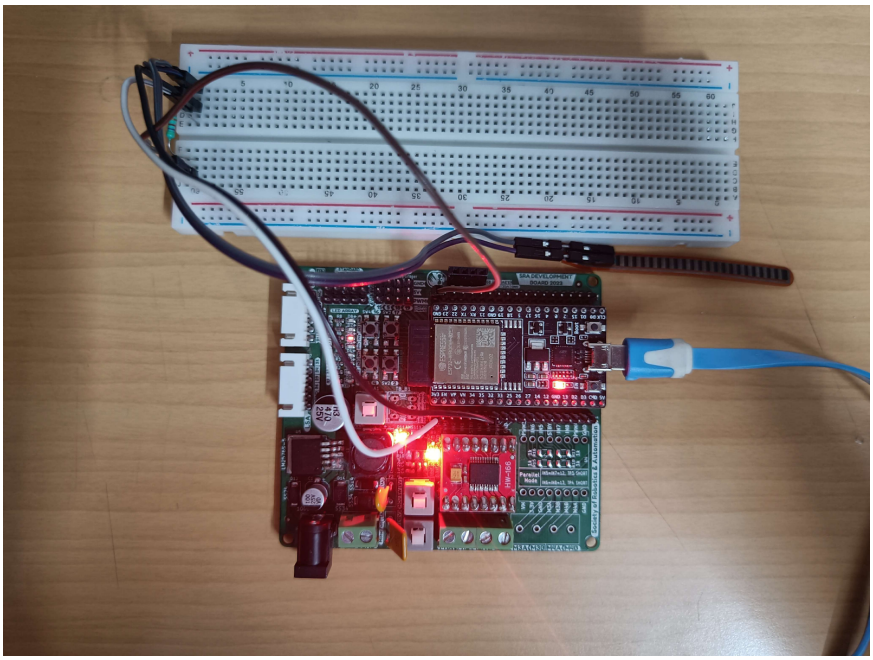
A) Acquiring ADC values of flex sensors

So to start working with flex sensors we'll first be intaking its voltage values using the ADC channel which will measure the analog input Voltage.
To intake its ADC values we refer the following code
ADC VALUE

You can refer the below link for connection
https://www.circuito.io/app?components=513,8606,360217

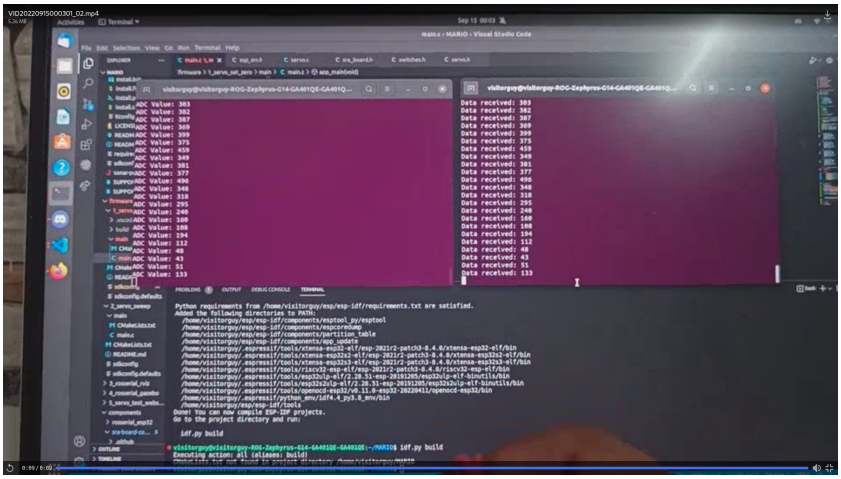


This is how our connection looked like



B) To send ADC values to the slave ESP32

We are already done with acquiring the ADC values for the flex sensors. Now our task is to interfacce that with the ESP-NOW protocol so we can send its value to the slave ESP-32



4) Controlling a Servo Motor using flex sensors

Here we'll be using the ADC values of the flex sensor obtained as a

parameter for the servo motors
We'll be using "If" condition to achieve this

5) Designing Part

This was undoubtedly our most challenging task :)

A) Designing the Arm

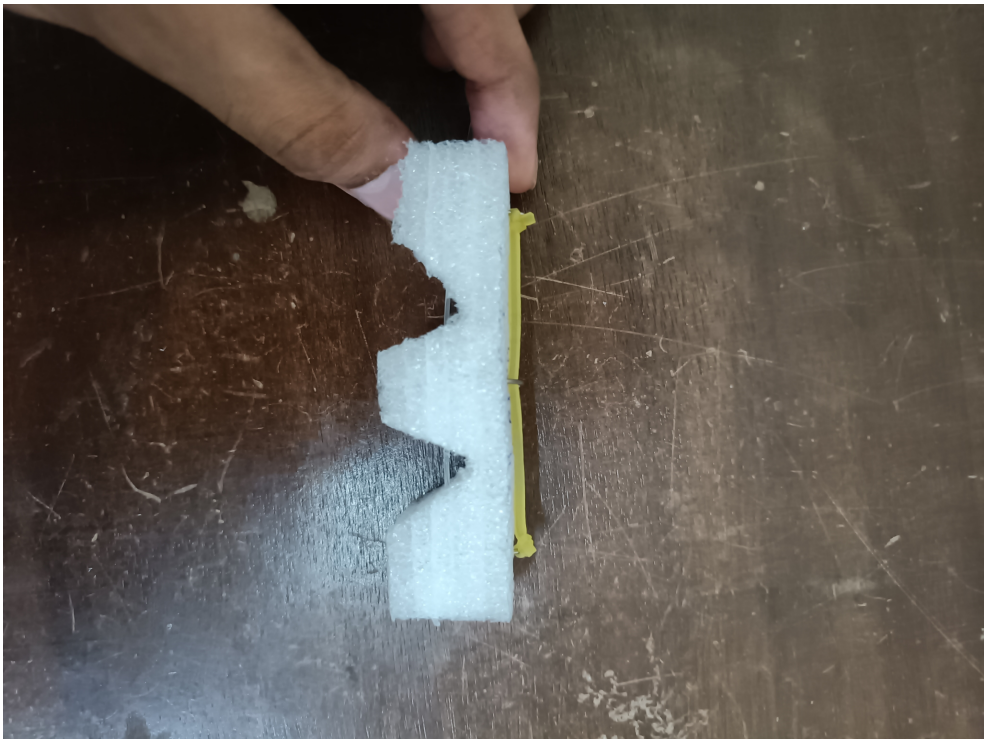
For designing the Arm we referred the below link
https://youtu.be/QOyghUxLdqE

List of materials Required:
1) Foam
2) Rubberbands
3) Pins
4) Nylon Wires
5) Hot Glue Gun
6) Glue
7) Plastic Straws

For designing the arm and the fingers we decided to move on with Styrofoam
But as it was not readily available anywhere and also it was a bit pricey we decided to

We decided the arm and fingers using foam

For Fingers:

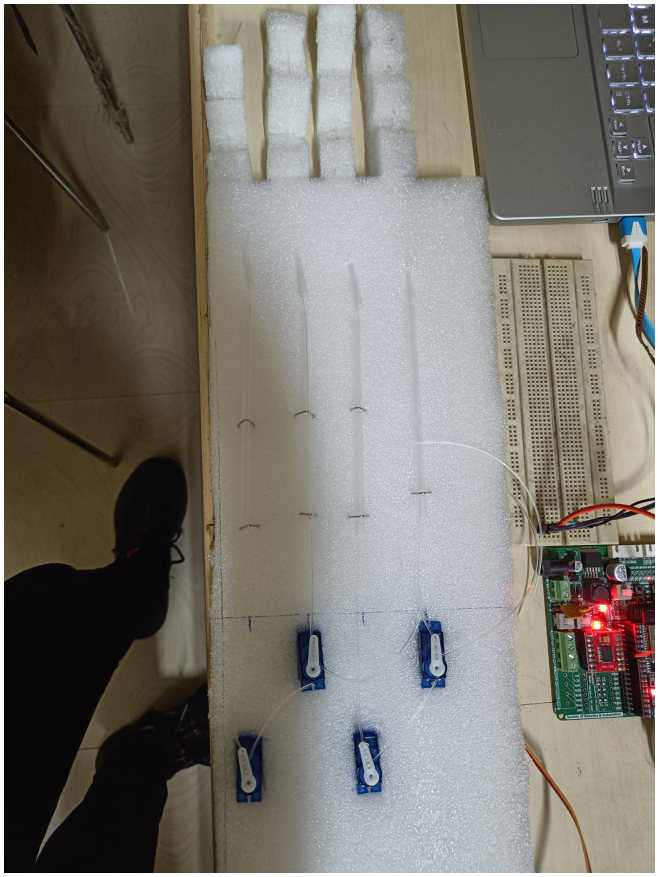


- We made two cuts in every finger for it to bend
- For the finger to retrace back to its original position we used rubber-bands
- The servo and the finger is connected with the means of a nylon string

For the Arm:

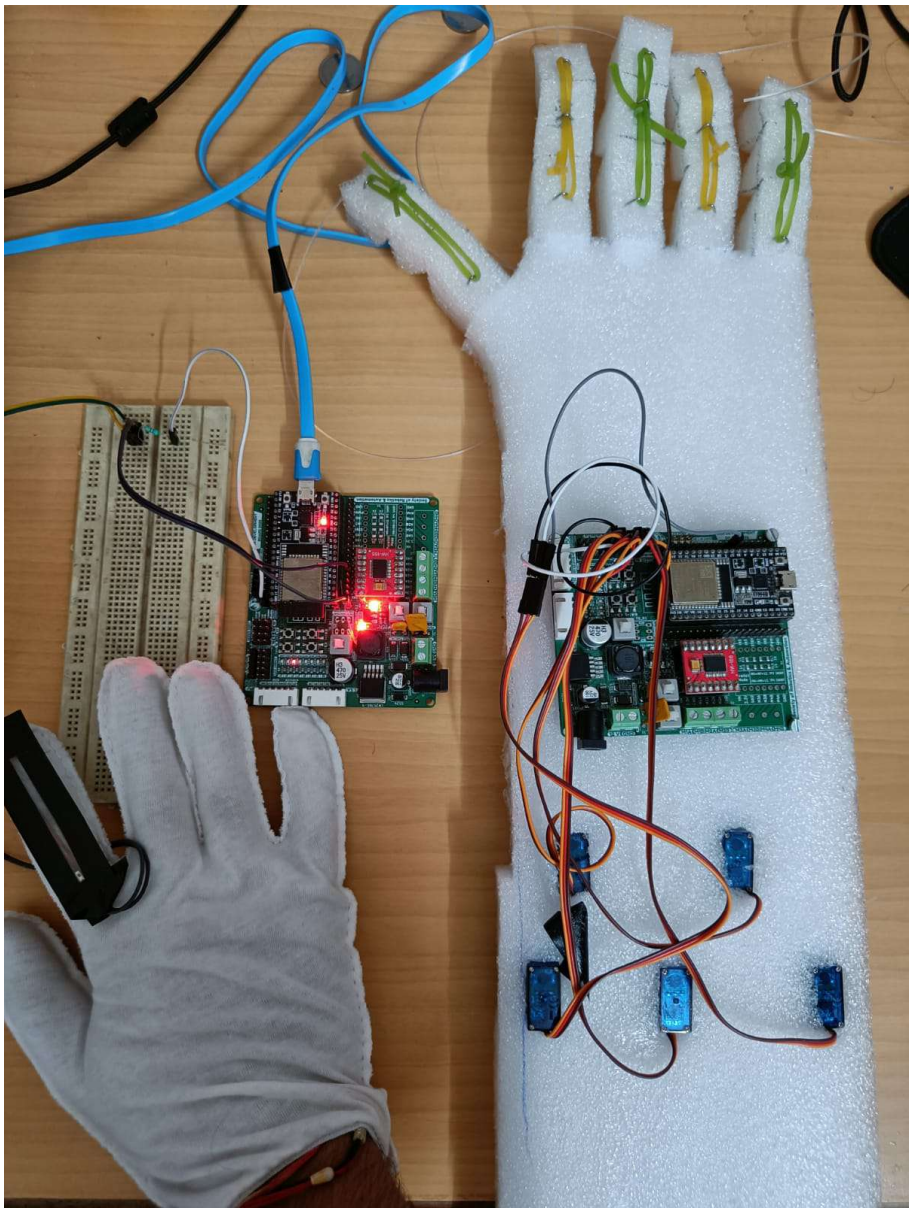
Referred the diagram below with the given dimensions

Our Arm and finger together turned out to be like this



For the Glove:
For this you can refer the same link too
Glove

Our Arm and glove, all together looked something like this



6] CHALLENGES FACED

- We were unable to use more flex sensors due to unavailability of velostat paper, and high cost of flex sensors
- We didn't get accurate readings from flex sensors made up of pencil shade.

7] FUTURE WORK

- use four more flex sensors to control each servo motor individually.

8] REFERENCE LINKS:

1) Embedded C :

freeRTOS Playlist:

https://youtube.com/playlist?list=PLXyB2ILBXW5FLc7j2hLcX6sAGbmH0Jx X8

freeRTOS study-group
https://sravjti.in/embedded-systems-study-group/week6/week6.html

API reference:
https://www.freertos.org/a00106.html

API reference for espnow:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/ne twork/esp_now.html

Esp-idf project build system

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build system.html

2) Designing

https://youtu.be/QOyghUxLdqE

https://youtu.be/5z5evInThP4

3) Servo motor control

https://github.com/espressif/esp-idf/blob/master/examples/peripherals/mcpwm/mcpwm_servo_control/main/mcpwm_servo_control_example_main.c

4) ADC flex sensor testing

https://esp32tutorials.com/esp32-adc-esp-idf/