Project Report

On

HEXAPOD

BY

THE TEAM OF

JYOTIPRAKASH MALLIK

MANAS KESHARWANI

PREYASU RAKSHIT

RAVI KUMAWAT

Table Of Contents:

1. Introduction

1,1 Project Description

A Hexapod is a 6-legged robot that uses various gait cycle patterns for locomotion. The robot mimics the walking pattern of 6-legged insects and provides a deep insight into the evolution of their walking patterns. The Project requires us to make a robot to execute various gait cycles on command. The Project also requires us to build a wireless communication system that connects the robot to a smartphone using an ESP 32 microcontroller.

1.2 Purpose

The purpose of the Project was to give a first-hand experience in developing a robot using hardware that was new to the team. The Project also aims to build Teamwork among the participants. It also seeks to provide an opportunity to create a modular robot built for plug-and-play applications.

1.3 Project Scope

The Project demands a robot made from Acrylic Sheet, which is Laser cut as per requirement. The Project also requires the development of an Inverse kinematics Solver Model, which is used to find various joint angles of the legs of the robot. Gait cycle generation will be done by hard-coding it into the microcontroller. The control system will be based on the esp-32 microcontroller, which will relay the data received to another microcontroller (Arduino Mega ADK). In the final stages of completion of the Project, we aim to eliminate the use of a second microcontroller (Arduino Mega ADK) to reduce the overall complexity of the Project.

2. Project Requirements Study

2.1Hardware Requirements

- The skeleton of the Hexapod is made from a Clear Acrylic sheet (5mm thickness). The 5mm thickness gives good enough structural integrity while being light enough not to overload the robot.
- The robot's brain will be an Arduino Mega ADK (chosen for its faster clock speed than Arduino Uno and Arduino Nano).
- For remote operation, an Esp-32 microcontroller is used. It provides both Wi-Fi and Bluetooth connectivity.
- Servo motors set the joint angles in the robot. We have chosen the MG995 servo motor for this purpose. It provides a peak of 12 kg-cm of stall torque and a wide range of operating voltage (4.8-7.2 V).
- To control the servos (18, to be precise), we have used two PCA9685 16-channel servo controllers. The communication between the controller and the microcontroller is achieved using the I2C protocol.

3. Project Design

3.1 CAD Model of the Robot

The CAD model of the Hexapod was taken from grabcad.com. This specific model was selected as it can be fabricated using acrylic sheets and uses the same mg995 servo motor for mobility. We modified the CAD model after the discovery of some structural flaws. These parts were later 3d-Printed. Also, the space constraint forced us to eliminate the top cover.

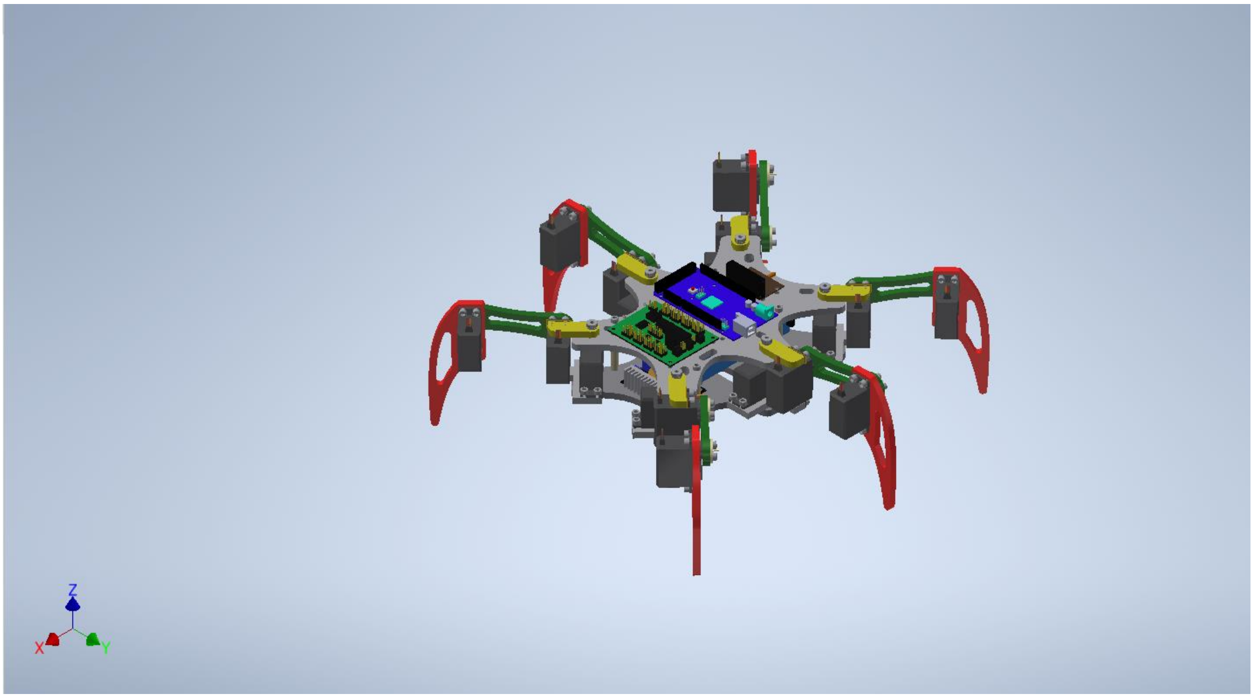


Figure 1 CAD from grabcad.com

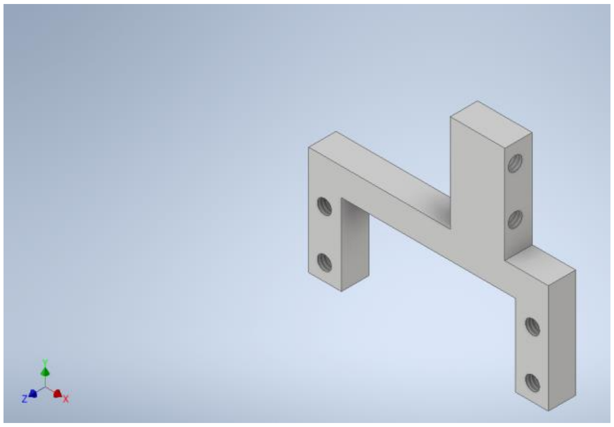


Figure 3 The part where the flaw was discovered.

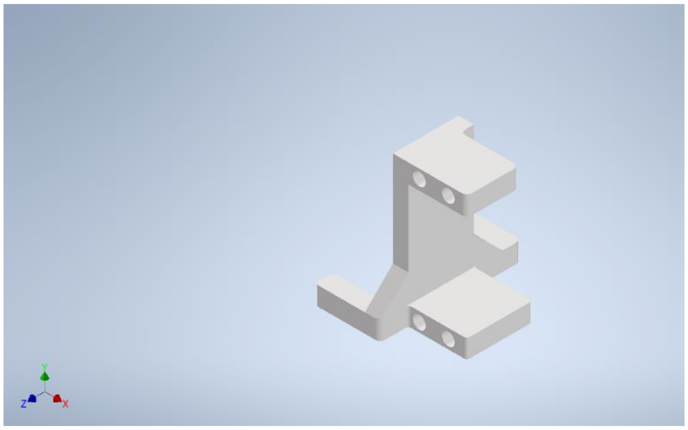


Figure 2 CAD of the part which was used in place of the previous part.

We also designed a separate mount to house the battery, electronics, and switches. The battery would go under the part, and the electronics would sit on top. Brass stand-offs were fitted on top to give some clearance to the components.

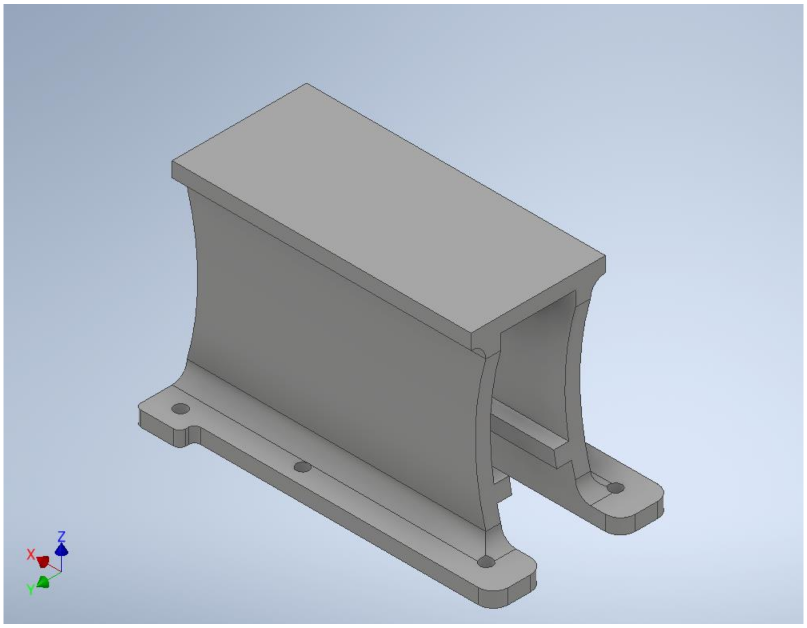
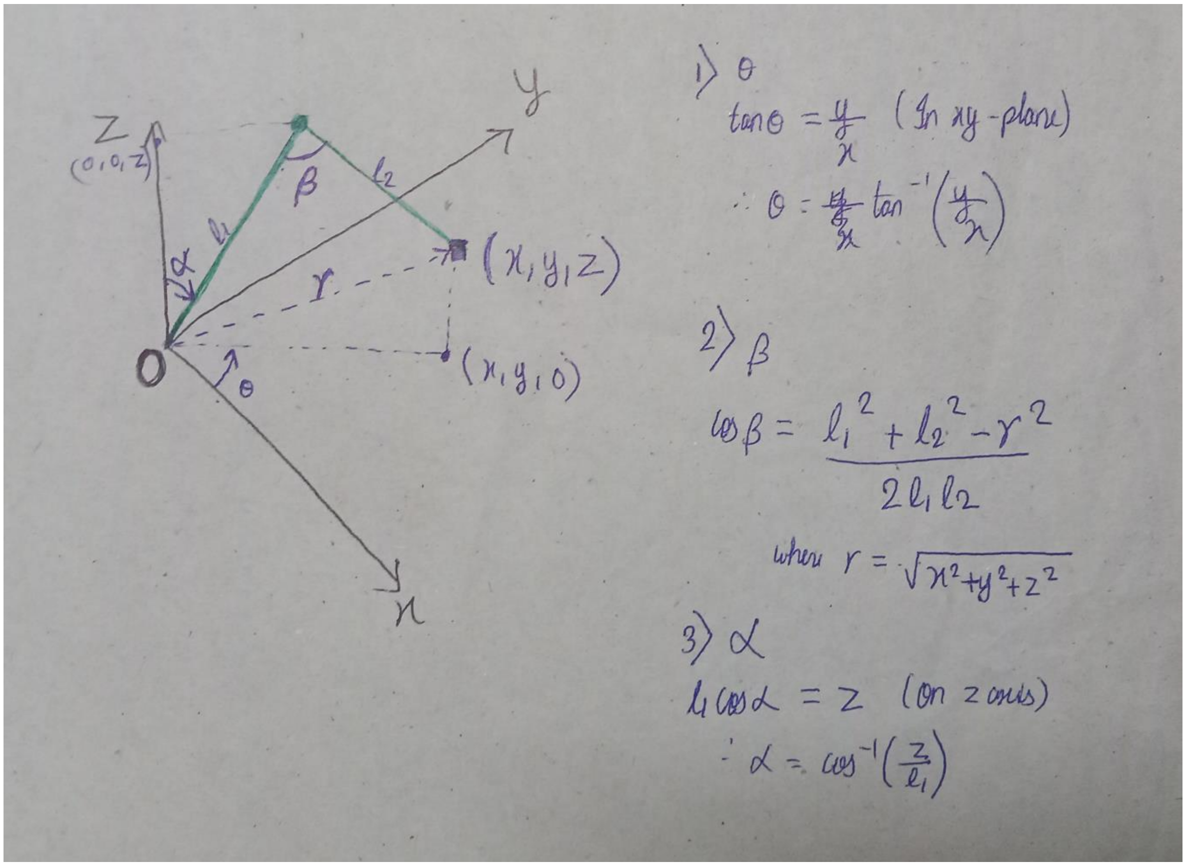


Figure 4 The battery/component mount

3.2 Inverse Kinematics Model

Inverse Kinematics in robotics is used to find the required joint angles of a system to get its end-effector to reach a particular position in space. In our case, the end-effector is the leg-tip of the robot. We need an inverse kinematics solver to calculate all the required angles of the leg joints for a specified coordinate in space. This reduces the length of the program written for the microcontroller. The explanation of the inverse kinematics solver developed is as follows.



1) θ

$\tan\theta = \frac{y}{x}$ (In xy-plane)

$\therefore \theta = \cancel{\frac{y}{x}} \tan^{-1}\left(\frac{y}{x}\right)$

2) β

$$\cos\beta = \frac{l_1^2 + l_2^2 - r^2}{2 l_1 l_2}$$

where $r = \sqrt{x^2+y^2+z^2}$

3) α

$l_1 \cos\alpha = z$ (on z axis)

$\therefore \alpha = \cos^{-1}\left(\frac{z}{l_1}\right)$

- In the above image, l1 and l2 are the lengths of the leg of the robot, which are known. (x,y,z) are the end-effector coordinates, in our case, the leg tip.
- α, β, θ are the angles that must be fed to the three servo motors in each leg to get the end effector to reach the coordinate (x,y,z).

3.3 Gait Pattern Generation

Walking in organisms follows a specific pattern known as Gait Pattern. In Hexapod, we have used one of the gait patterns which a spider uses. In this pattern, there will be three legs in the air at any time and three in contact with the ground. When viewed from the side, the tip of the leg seems to perform a closed semicircular loop. The same was implemented during the initial development phase, but due to the servo motors' lack of precise position control, the overall motion was jerky. Hence, we went with a closed triangular loop instead. This resulted in better motion and required less processing power.

For forward motion, the leg tip follows a triangular path, whose plane is parallel to the sides of the Hexapod. This motion is made by two sets of legs with a delay in their position in the loop. The two sets are as follows:

- Set 1: Left-front, Right-middle, Left-rear
- Set 2: Right-front, Left-middle, Right-rear

When Set 1 moves along the base of the triangular path, Set 2 moves along the non-base sides and vice-versa. This pattern results in forward/backward movement depending upon the direction (clockwise/counterclockwise) of tracing the triangular path.

For rotation, both sets trace the path in a different sense. Depending on which set chooses the clockwise and counterclockwise sense of tracing the path, we get clockwise or anti-clockwise rotation of the robot.

3.4 Wireless Control

We have used an esp32 microcontroller to establish wireless communications between the operator and the robot. The esp32 supports Station, AP, Station + Soft AP modes of Wi-Fi operation. The initial plan was to use it in Station mode and create a website where the robot could be controlled. But due to the inconsistency in features of different esp32, the code would run without lag on some models of esp32 only.

Another attempt was made to control the robot wirelessly using an app interface. Upon doing some research, we found an open-source application that provides various controls under a single application. The app's name is 'Controller,' Made by Invok Lab. The app uses the Station + Soft AP mode for communication.

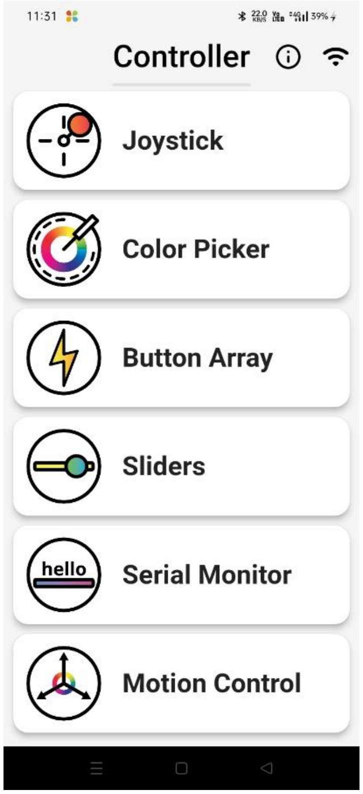
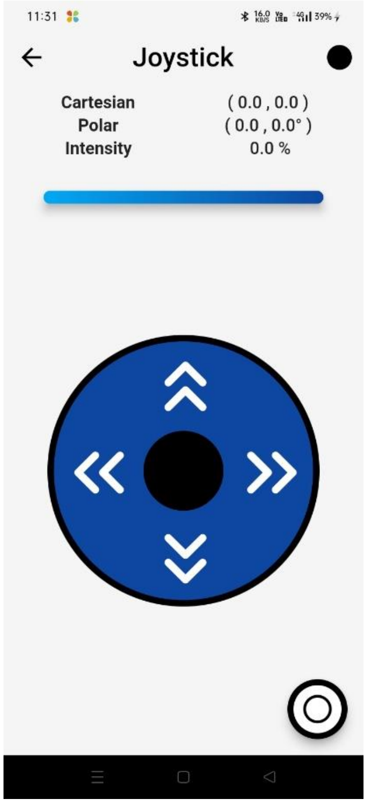
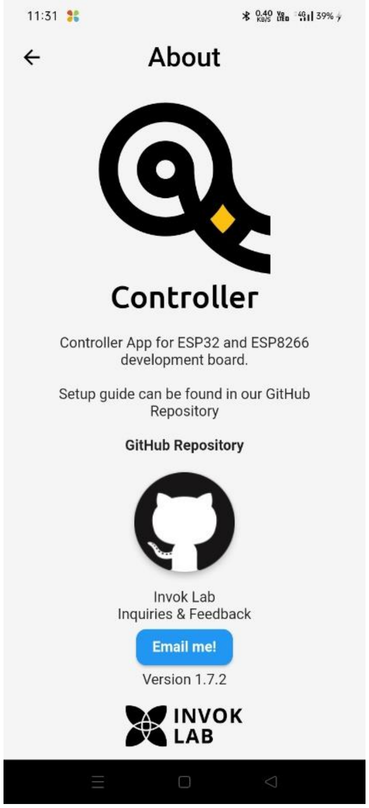


Figure 6 App interface



Cartesian (0.0 , 0.0)
Polar (0.0 , 0.0°)
Intensity 0.0 %

Figure 7 Joystick interface (used in Hexapod)



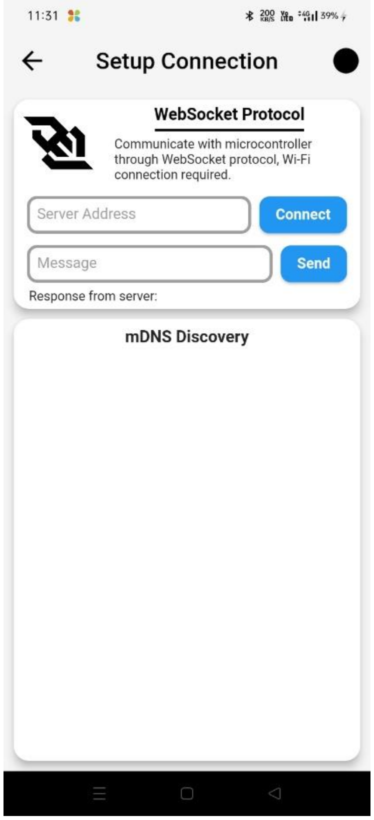
Controller App for ESP32 and ESP8266 development board.

Setup guide can be found in our GitHub Repository

Invok Lab
Inquiries & Feedback

Version 1.7.2

Figure 8 Further information about the App and link to its GitHub repository



WebSocket Protocol

Communicate with microcontroller through WebSocket protocol, Wi-Fi connection required.

Response from server:

Figure 5 Interface for connecting to esp32.

3.5 Power Distribution

The initial electrical setup of the controllers and the components were as follows:

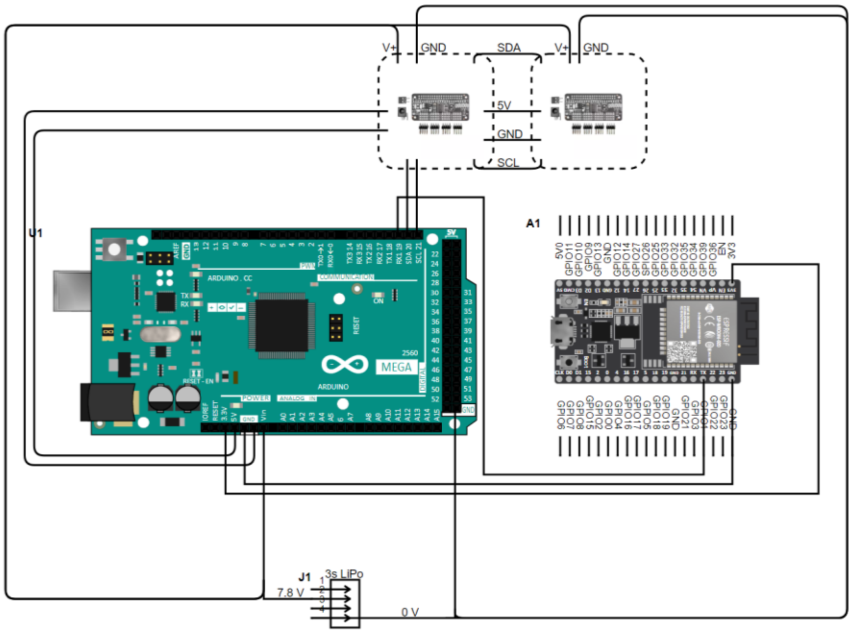


Figure 9 Schematic of the initial circuit, used Arduino Mega 2560 for processing.

In this configuration, the whole robot runs on 7.8 V lines. Though this configuration puts load only on two cells of the 3s LiPo battery pack, it is most straightforward in terms of complexity. Initial test runs were successful with this configuration, but during one of the tests, upon powering the system on, two servos stopped working along with the esp32 microcontroller. The prime suspect we could think of was High Inrush Current which may have caused the system to malfunction.

We used two esp32 microcontrollers to create a new configuration following this test run. The overall setup was the same as before, with the Arduino Mega being replaced by another esp32. We also made the power lines of the PCA9685 run through voltage regulators. This was done to separate the high current lines from the low power-consuming components (esp32 and the logic of PCA9685). The resultant schematic is as follows.

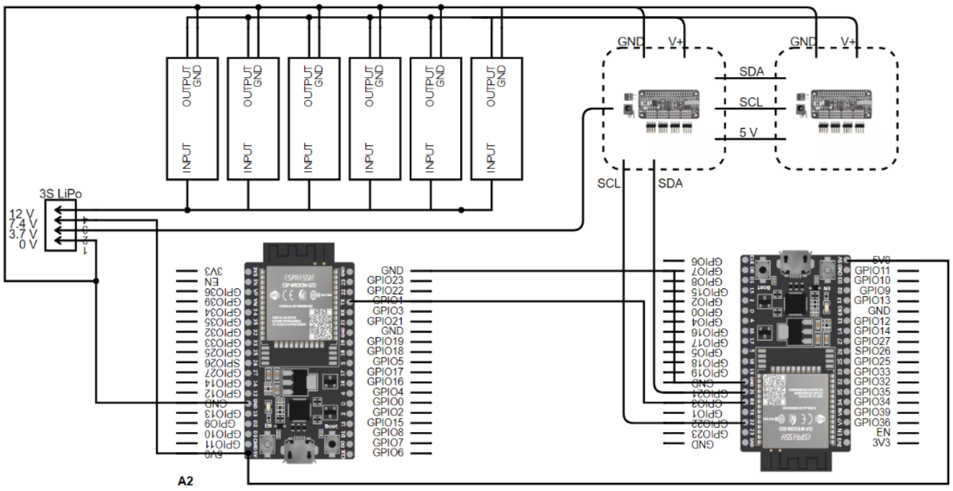


Figure 10 Schematic of circuit which uses esp32 for processing.

The new configuration also has flaws; the voltage regulators convert the 12V input to usable 6V by converting the extra energy into heat. We fabricated a heatsink to help dissipate the heat generated. The esp32 to the left of the figure connects to the smartphone app and relays the instructions received through the Rx line to the esp32 on the right. The second esp32 (on the right) does the necessary computation and sends the joint angles to the servo controllers using the I2C protocol (This protocol uses the SDA and SCL pins of the esp32).

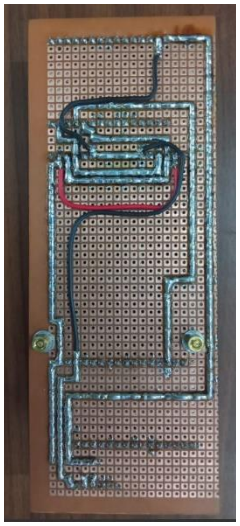


Figure 11 Final PCB Bottom View

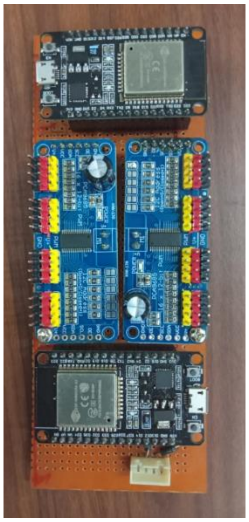


Figure 12 Final PCB Top View

4. Key Learning Points

4.1 Hardware Level

- Checking for possible structural flaws before the fabrication of the part comes in handy and saves a lot of time in the development phase.
- We discovered a way of making holes in acrylic sheets without ruining the drill bit with melted acrylic. The proper way is to use low RPM with high pressure.
- The Servo motors, which use potentiometers as position control, provide a meager angle resolution. Using ones with magnetic encoders offer very high resolution.
- We discovered that only those MG995 servos with a brass shaft have metal gears inside, while others with a non-brass shaft have plastic gears inside.
- Controlling servos with PCA9685 gives issues while holding a position; if some disturbance comes, the servo oscillates about its holding position. This issue does not come when we drive it directly from a microcontroller.

4.2 Software Level

- WebSocket protocol of the esp32 is susceptible to the model of the esp32 under consideration. Also, it needs an active internet connection to do so.
- The soft AP mode of esp32 consumes a lot of energy and causes the chip to heat up.
- Serial communication between esp32 and Arduino requires a [Serial.flush();] command if we want to use the latest data received on the Rx line. But doing so in communication between esp32 to esp32 does not help. In our case, we had to stop the Serial communication once the data was read and restart it again when we needed another reading.
- Using WI-Fi and I2C communication simultaneously makes the Wi-Fi reboot. Hence we used two separate esp32.

5. Future Scope

We could implement the following in the Hexapod :

- Use Servo motors which use magnetic encoders for position control. One such example is SC15 from WaveShare.
- Create a feedback system to interrupt the microcontroller whenever the leg touches the ground, enabling locomotion in rough terrain.
- Attach a depth-sensing camera to create an object-following robot.
- Transitioning from acrylic to 3D printed skeleton to reduce the overall weight of the robot.
- Using a 2S battery or implementing a buck converter instead of a voltage regulator.

6. References

Grabcad: https://grabcad.com/

Arduino IDE: https://www.arduino.cc/en/software

HCPCA9685 Library Source:
https://forum.hobbycomponents.com/viewtopic.php?t=2034

Controller Application:
https://play.google.com/store/apps/details?id=com.invokcontroller.app

Schematic creator

DigiKey Scheme-it: https://www.digikey.com/schemeit/project

Source Code to the Project: https://github.com/jpmiitd/Project_Hexapod.git