Low-cost Robotic Arm for Handicapped People

By

Hemant Kuruva

(20077636)

A project report submitted to the

Department of Electrical and Computer Engineering.

In conformity with the requirements for the Degree of

Masters of Engineering.

Queen's University

Kingston, Ontario, Canada

Lay Summary:

With technological advancements in the field of Biorobotics happening at a fast pace, it has become a necessity that much of it, if not all, is accessible to the public. One of the major factors that hinders this technology from being accessible to the public is the cost. The highly sophisticated technology, developed through extensive research and usage of high-quality materials, paves way to the surge in the price of the final product. This is the scenario when the low-cost variants, with few trade-offs, can be developed so that they are used by the very individuals the technology exists for.

The 'Low-cost Robotic Arm for Handicapped People' is a physical assistant developed for helping individuals with limited mobility. It is designed to be attached to the wheelchair of the physically challenged individual who can make use of external assistance. It enables the individual to perform everyday tasks such as picking objects, buying groceries, opening or closing the doors and other household tasks which can prove to be tiresome in the absence of an external assistance. The arm is controlled through, either intuitive joysticks or a smartphone application wirelessly.

Table of Contents:

1. Introduction

1.1 Background

The aim of this project is to design, build and test a robotic arm that can be mounted onto an electric wheelchair and used by an individual with limited mobility to shop groceries and perform other household tasks independently. The robotic arm should be controlled by either a couple of joysticks or a smartphone application. The client has set a desired budget of $2000 for the manufacturing of the robotic arm. The arm should be able to reach from 0.6 m to 1.2 m to allow reach at a variety of shelving heights. The arm should also be able to lift common grocery items.

1.2 Objective

The focus of the team is to design and build a product version of the robotic arm that will be controlled by joysticks attached to an Arduino microcontroller and also to develop a smartphone application which maneuvers the arm via Internet with negligible delay using Internet of Things technology. Tasks undertaken include the electrical and mechanical assembly of the arm, development of software to map the inputs from the joysticks to outputs performed by the actuators, development of the smartphone application and the testing of the arm's performance.

1.3 Scientific summary

The goal of this team was to design a responsive robotic arm which could assist the physically disabled people in household tasks. The arm is built to be attached to a wheelchair, with enough lifting power for an individual to use it while grocery shopping independently. This robotic arm contains four revolute joints and a prismatic joint making it an arm with 5 DOF (Degrees of Freedom). The team built, wired, and programmed the final product version of the arm. Further, the team also developed a smartphone application which could control the robotic arm in case the

joysticks, which are the primary controllers, are not a proper means of control for a specific individual.

This project is a new approach to a previous problem, attempted by a team of 5 undergraduate students under the project titled “Grizpaw controlled Robotic arm” as a MECH 462 Project. Though deriving the concept from the previous project, this project titled “Low-cost Robotic Arm for handicapped people” is a completely new implementation with advanced features not proposed in the previous project.

This robotic arm contains four revolute joints and a prismatic joint making it an arm with 5 Degrees of Freedom. The motors that will be part of this project are high torque motors with load torque capacities 11.7 Nm at the Shoulder joint, 4.9 Nm at the Base joint and 3.9 Nm at the Elbow joint. The motors are also equipped a worm geared gearbox which provide the self-locking capability to prevent reverse-drive in case of power failure. For the arm’s extension and retraction, a linear actuator of 10” stroke is used. At the end effector, a hand gripper is used which is powered by a hobby servo motor with maximum travel of 188°.

All the motors except the servo motor, are independently connected to Arduino Duemilanove via Cytron Dual channel DC motor driver with a maximum current rating of 10A, whereas the servo motor is directly connected to Arduino MEGA. There is another device part of this project, Particle Photon IoT device which is the heart of ‘Smartphone application control’ aspect of the arm. It uses Internet of Things technology to transmit and receive the signals.

The control system is fairly straight forward as it directly reflects the actions of the joysticks on to the motors. However, with upgraded hardware, namely motor encoders for position feedback, a more sophisticated control system can be implemented in further developments where the arm senses its position and acts accordingly.

There are two ways to maneuver the robotic arm, the first being a couple of dual-axis joysticks with buttons and the second being a smartphone application. One joystick is used to maneuver the shoulder motor and the base motor, the other being used for the elbow motor and the linear actuator. The buttons on the joysticks are used to hold and release the hand gripper so as to pick the objects the user intends to. The smartphone application, on the other hand, has an interface where a virtual joystick is provided to control the base and the shoulder motor, and virtual sliders to control the elbow motor and linear actuator. The application also has two virtual push buttons for the purpose of holding and releasing an object from the hand gripper.

2. Methodology

2.1 Theory

2.1.1 DC Motor

DC motors are motion components that take electrical power in the form of Direct Current and convert it into mechanical rotation. The motors do this through the use of magnetic fields that arise from the electric currents to spur rotation of a rotor fixed with an output shaft. Output torque and speed depends on the electrical input and motor design.

There are three main reasons for choosing DC Motors over Stepper motors. Firstly, stepper motors are more suitable in applications where precise positioning of the shaft is a requirement which is not the case with this project. Secondly, the bulkiness of the stepper motor pulls the arm towards

ground when working at larger tilt angles. More powerful the stepper motor is, the heavier the motor gets and eventually falling into a vicious circle where acquiring a lightweight, yet powerful stepper motor is not possible. Thirdly, the self-locking capability provided by the Worm gear gearbox of the DC motor used in this application prevents the arm from falling on its weight in case of power loss.

2.1.2 Worm Gear

The operation of worm gear motors can be broken down into two stages. Essentially, an electric motor provides the power to rotate the worm or screw gear. As the worm gear turns, as seen in Figure 1, it also turns the main gear, allowing for motion and torque to be produced.

Fig. 1: Worm gear

The first part of worm gear motors is the worm gear itself. This consists of a shaft with threading that spirals itself on the shaft. This worm gear is set into the splines of another gear. The rotational

speed of the worm gear would in turn rotate the gear it is attached to. The second part of the motor, as the name suggests, is the motor. The motor is attached to the worm gear, turning it at different speeds depending upon the amount of torque produced by the main gear. [3]

2.2 Projects from past years

The Grizpaw robotic arm team from the 2016-2017 academic year purchased a five DOF robotic arm from SainSmart and integrated it with the Grizpaw controller [3]. This setup was used to perform a series of tests to measure repeatability, learning curve, and task completion. It was determined that the Grizpaw controller can be successfully used to control a system composed of several motors. The team showed that the movements were repeatable within ±0.9 cm, that the control of the arm was improved as the team used the Grizpaw more, and that the Grizpaw can be used to move objects from place to place, although there was a significant degree of shakiness with the purchased arm. It was determined that while the script used for their tests worked well, it should be made more general to work with a variety of arms. The team also suggested that an ergonomic analysis of the Grizpaw might be useful to ensure that prolonged use of the Grizpaw does not result in any significant strain on the hand of the user

In the 2017-2018 academic year, a team of 5 undergraduate students built a wooden version of the robotic arm under the project titled “Grizpaw controlled Robotic arm” as a MECH 462 Project. The team used stepper motors at the revolute joints. Though largely successful, the arm failed to accomplish the objective of lifting due to its own weight consisting of heavy stepper motors and also a weak chassis built from lumber. The arm developed by the team can be seen in Figure 2.

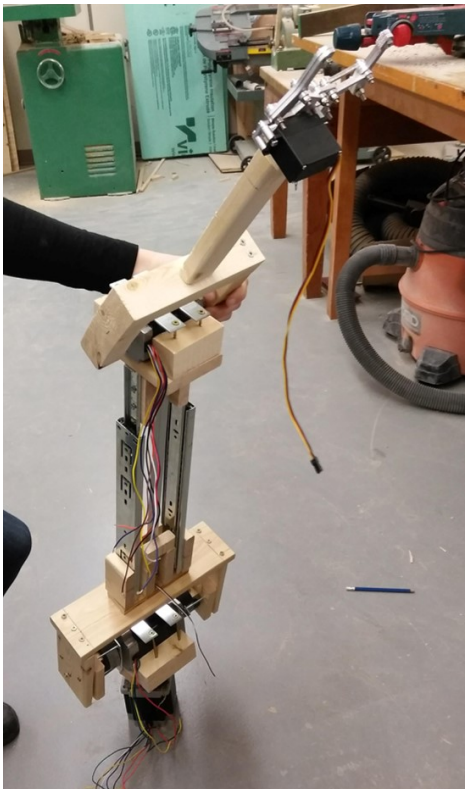


Fig. 2: Grizpaw controlled Robotic arm built by MECH 462 Project team

2.3 Approach

The chassis of the arm is built using aluminium extrusion bars of various sizes connected to each other with 90° internal brackets. The design contains a revolute joint at the base to turn the arm along the vertical axis to the ground, two revolute joints to bend and maneuver the arm, one revolute joint at the hand gripper to hold and release the objects, and one prismatic joint with a linear actuator to vary the length of the arm. A turntable with bearings is mounted directly to the top of the base motor, which allows for a smooth rotation of the arm about the vertical axis with limited torque requirements. This also prevents the robotic arm from inserting excessive amount of force on the shaft of the base motor. The shoulder joint assembly sits on the turntable and is mounted to the base motor shaft. The maximum torque requirement is at the shoulder joint as the maximum amount of arm's weight falls on the joint with the motor shaft's axis horizontal to the

ground. There is one ball bearing slider used in conjunction with a linear actuator for the same reason as with the turntable for the base motor, i.e. to assist in releasing the weight falling on the motor shaft. All the motors are mounted to their respective aluminium platforms using aluminium plates specially milled for the purpose. Mechanical connections between the U-shaped aluminum parts and the motor shafts are made possible with set screw mounting hubs, which are fixed directly onto the U-shaped aluminium parts. The base motor which is the bottom-most part of the arm is fastened to a large wooden table with an aluminum plate and aluminum extrusions acting as support.

Figures 3 to 5 are the 3D renderings of the arm model developed on Autodesk Fusion 360. The rendering was done on the native Fusion 360 Renderer.

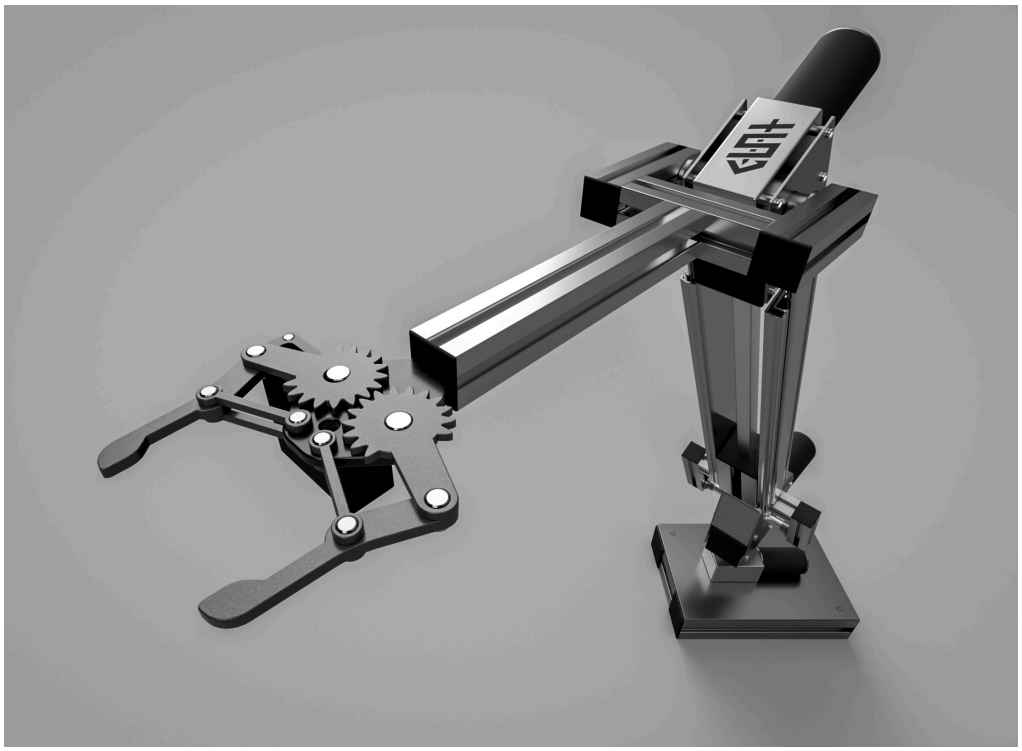


Fig. 3: Diagonal view of the arm.

Fig. 4: Side diagonal view of the arm.

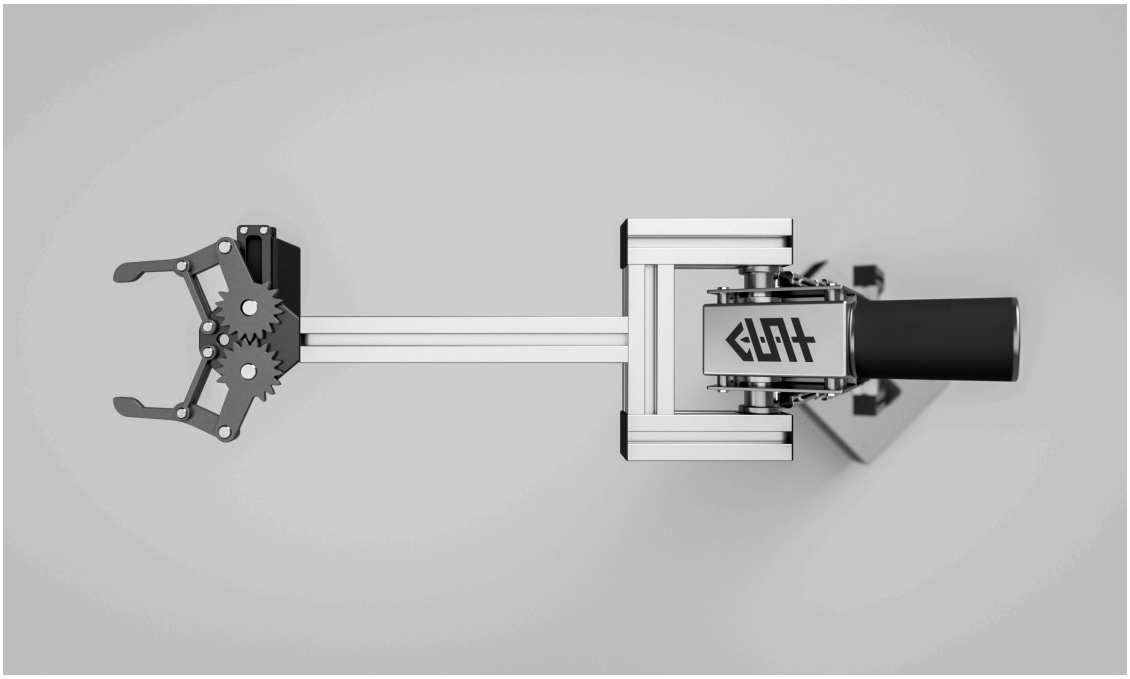


Fig. 5: Top view showing the U-shaped part.

2.4 Hardware

2.4.1 Parts and Materials

Table 1 shows a summary of the parts and materials used to create the robotic arm. The source column outlines how each part was supplied to the team. Most of the parts were ordered through Robotshop and Amazon and some parts were provided by the Project Supervisor as they were accessible through the University resources (i.e. Mechatronics Lab).

Table 1: Parts and materials list for the robotic arm

Sl. No.	Item	Description	Quantity	Source
1	IRUI Turbine DC Motor (Shoulder)	Worm Geared Double Shaft – 12 V – 3 A – 10 RPM – 11.7 Nm	1	Amazon
2	Uxcell DC Motor (Base)	Worm Geared Single Shaft – 12 V – 7 RPM – 4.9 Nm	1	Amazon
3	Uxcell DC Motor (Elbow)	Worm Geared Double Shaft – 12 V – 10 RPM – 0.85 A – 3.9 Nm	1	Amazon
4	HITEC Servo motor (Hand Gripper)	HS-425BB – 6 V – 188°	1	Prof. Surgenor
5	PA-07-10-5 Micro Linear Actuator	Progressive Automations – 10” stroke – 0.59”/s	1	Progressive Automations
6	Motor Driver	Cytron Dual Channel DC Motor Driver – 10 A – 5-30 V	3	Robotshop
7	Arduino Duemilanove	Lynxmotion - BotBoarduino	2	Robotshop
8	Arduino MEGA 2560	Arduino Clone	1	Team
9	Particle Photon	IoT Device	1	Team
10	Breadboard	1 Medium and 1 Small	2	Prof. Surgenor
11	Aluminum extrusion 40x40mm	Black – 500mm	2	Robotshop
12	Aluminum extrusion 20x20mm	Black – 300mm – (Pack of 4)	1	Robotshop
13	M4 Nut	For 20mm Aluminum extrusion (Pack of 10)	1	Robotshop
14	Internal Steel Connector	For 20mm Aluminum extrusion	2	Robotshop
15	90 Internal brackets	For 20mm Aluminum extrusion (Pack of 4)	3	Robotshop
16	90 External brackets	For 20mm Aluminum extrusion (Pack of 4)	3	Robotshop
17	DC Motors (Testing)	Worm Geared Single Shaft – 12 V – 1.14A – 20 RPM – 2.7 Nm	3	Amazon

18	Gripper	Lynxmotion Little Grip Kit	1	Prof. Surgenor
19	Set screw hub – 10mm	Actobotics	2	Robotshop
20	Set screw hub – 8mm	Actobotics	4	Robotshop
21	Joystick Breakout Board	2-Axis and 1 Button	2	Robotshop
22	MPU-6050	6 DOF Gyro Accelerometer IMU	2	Robotshop
23	3/8" x 1/4" Standoffs	Lynxmotion Nylon (Pack of 4)	1	Robotshop
24	3/4" x 1/4" Standoffs	Lynxmotion Nylon (Pack of 4)	1	Robotshop
25	M4 Bolts and Nuts	12 x 8mm and 14 x 12mm	26	Prof. Surgenor
26	M5 Bolts and Nuts	2 x 8mm and 8 x 12mm	10	Prof. Surgenor
27	6-32 Bolts and Nuts	6 x 12mm, 8 x 25mm and 1 x 45mm	15	Prof. Surgenor
28	Turntable	40mm circumference	1	Prof. Surgenor
29	Turntable Spacer	40mm Circ. and 7mm Height	1	Prof. Surgenor
30	Ball Bearing Drawer Slider	14" – 27.5"	1	Department Store
31	Aluminium plates	Of various dimensions for mounting motors and extrusion support	-	Prof. Surgenor
32	12 V Batteries	10 cells – Rechargeable NiMH Battery	2	Prof. Surgenor
33	6 V Batteries	5 cells – Rechargeable NiMH Battery	1	Prof. Surgenor
34	Single core wires	Of various lengths	-	Prof. Surgenor

2.4.2 Mechanical structure

The arm as seen in Figure 6, was assembled in a modular manner. The base, i.e. the aluminium base support, the base motor along with the shoulder motor and its platform are termed as 'Base module'. The U-shaped shoulder part, the linear actuator, the elbow motor and its support platform are termed as 'Shoulder module'. The U-shaped elbow part, along with the attached hand gripper and its Servo motor are termed as 'Elbow module'. The starting from the base motor, moving up to the end effector. Once the structure of the robotic arm was assembled, the team created a mount for the arm and its electrical components. Figures 7 to 11 illustrate the steps taken to complete the structural assembly of the robotic arm.

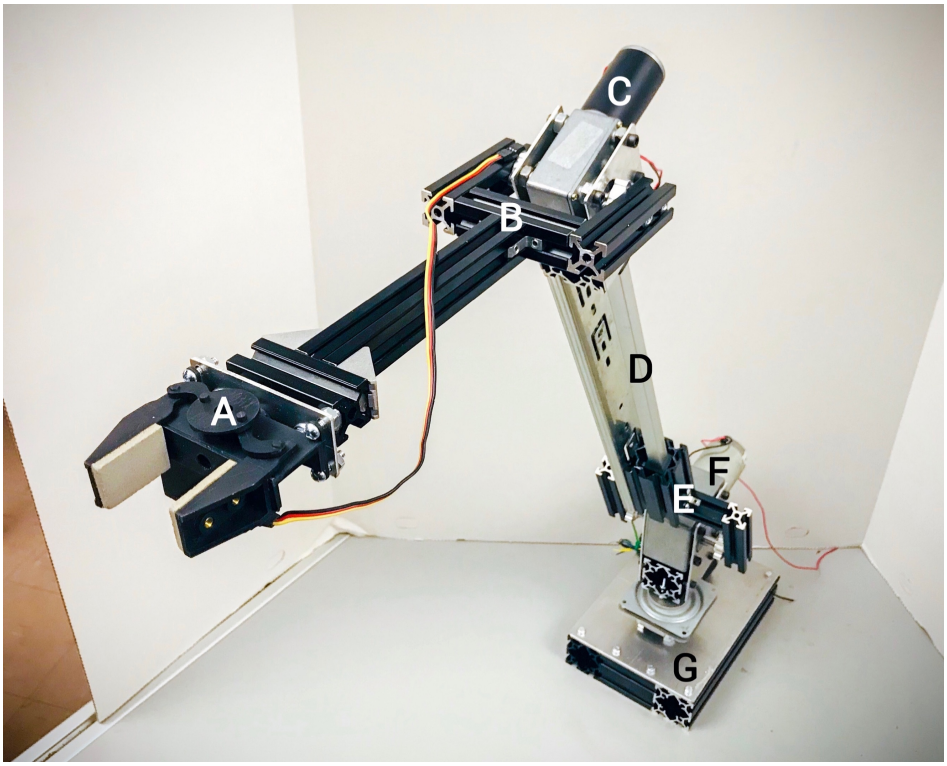


Fig. 6: A – Hand gripper; B – U-shaped elbow part; C – Elbow motor; D – Linear actuator; E – U-shaped shoulder part; F – Shoulder motor; G – Base platform

Note: Base motor is not visible due to alignment with the shoulder motor.

Fig. 7: Most of the parts used to build the robotic arm.

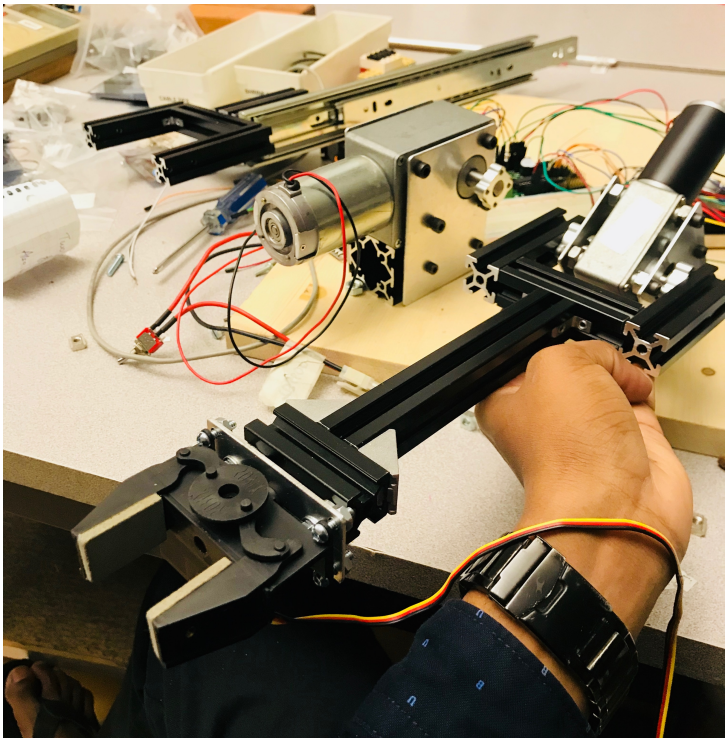


Fig. 8: Elbow module close-up

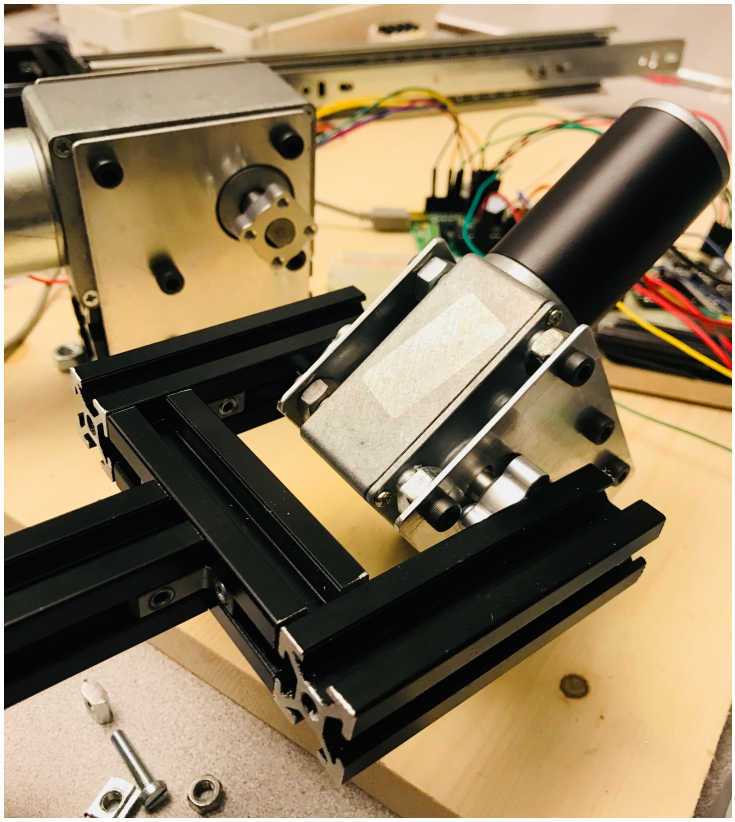


Fig. 9: U-shaped part of Elbow module

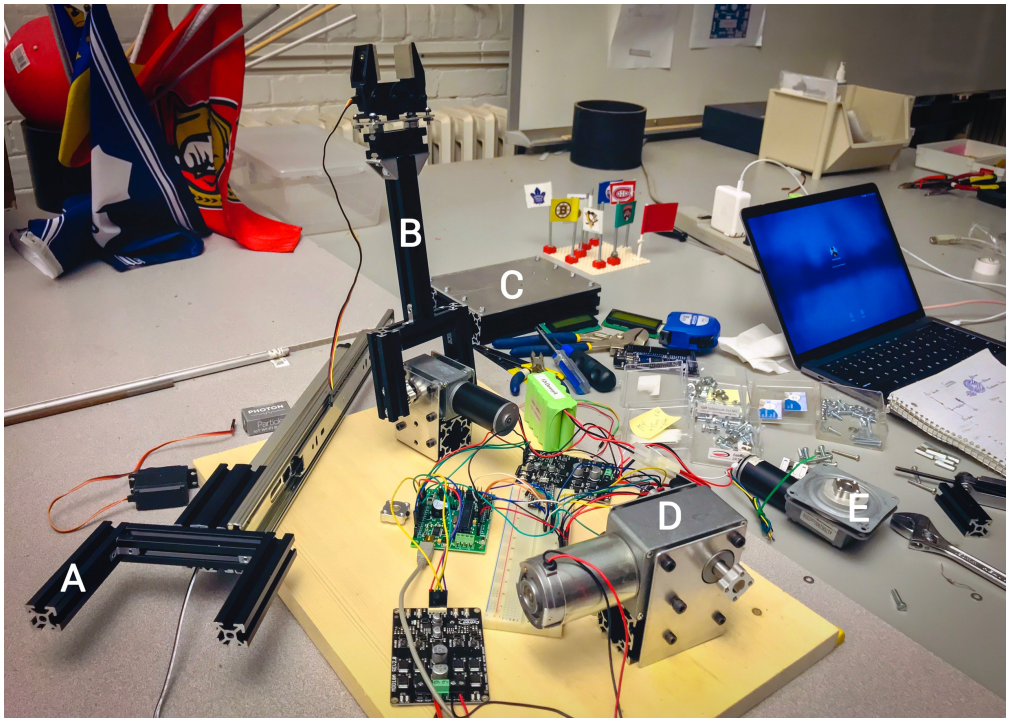


Fig. 10: A – U-shaped shoulder part; B – U-shaped elbow part along with Elbow motor and platform; C – Aluminium Base platform; D – Shoulder motor along with its platform; E – Base motor

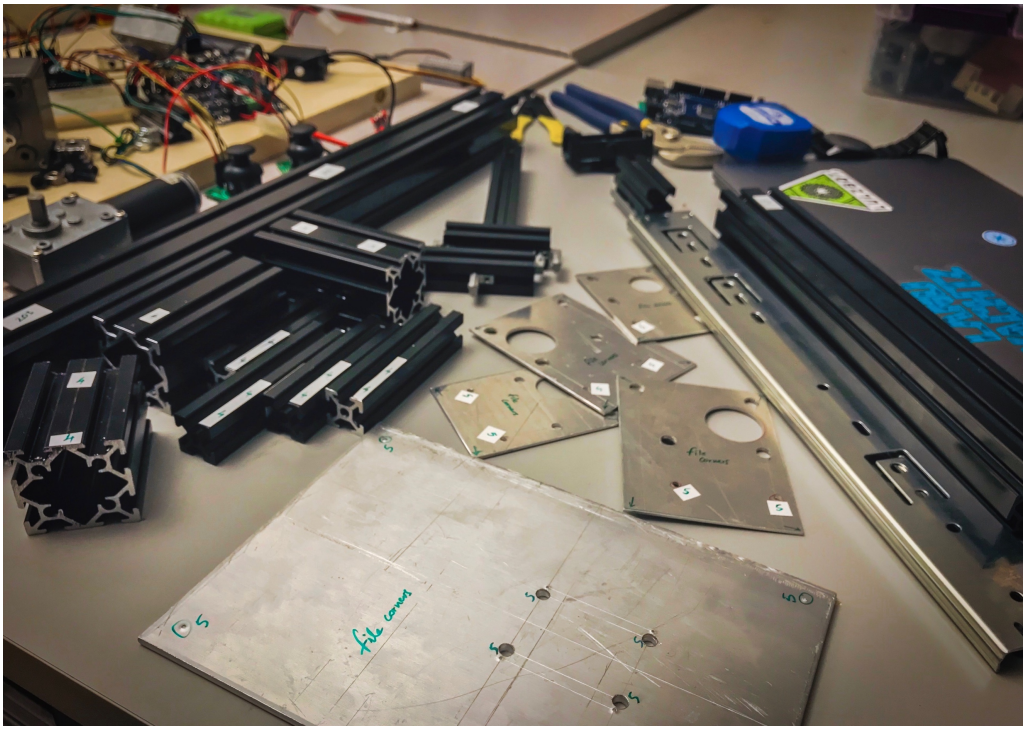


Fig. 11: Black aluminium extrusions can be seen to the left, Large base aluminium plate to the bottom, Motor mounting plates in the middle, and Ball bearing slide to the right

2.4.3 Mechanical specifications - Weight and dimensions

The weight of the robotic arm was found out to be around 3580 g including the base platform made of aluminium extrusion bars. The weight breakdown can be seen in Table 2.

Table 2: Weights of the components of the arm

Sl. No.	Component	Weight
1.	40 x 40 Aluminium extrusion (750 mm x 1.26 g/mm)	945 g
2.	20 x 20 Aluminium extrusion (624 mm x 0.47 g/mm)	294 g
3.	Elbow motor	390 g
4.	Shoulder motor	1000 g
5.	Base motor	370 g
6.	Linear actuator	181 g
7.	Miscellaneous (Mounting plates, hand gripper, servo motor &c.)	400 g
Total		**3580 g**

Note: This weight breakdown excludes the weight of the wooden board and its electrical components.

The total length of the robotic arm is 95 cm when the linear actuator is fully contracted and 120.4 cm when fully extended. Further, the closest object to which the arm can reach is 25 cm.

2.5 Electricals

The electrical portion of the arm consists of three worm geared DC motors, a linear actuator, and a servo motor. Worm geared DC motors are chosen to ensure self-locking. The base motor is a worm-geared, single shaft DC motor rated at 4.9 Nm, the shoulder motor is a worm-geared, double shaft DC motor rated at 11.7 Nm, and the elbow motor is a worm-geared, double shaft DC motor rated at 3.9 Nm. For the arm's extension and retraction, a linear actuator of 10" stroke is used. At

the end effector, a hand gripper is used which is powered by a hobby servo motor with maximum travel of 188°. All the motors except the servo motor, are independently connected to Arduino Duemilanove via Cytron Dual channel DC motor driver with a maximum current rating of 10A, whereas the servo motor is directly connected to Arduino MEGA. Servo motor is made independent from the main circuit so as to prevent it from signal disturbances produced during the control of other motors.

There are two ways to maneuver the robotic arm, the first being a couple of dual-axis joysticks with buttons and the second being a smartphone application. One joystick is used to maneuver the shoulder motor and the base motor, the other being used for the elbow motor and the linear actuator. The buttons on the joysticks are used to hold and release the hand gripper so as to pick the objects the user intends to. The smartphone application, on the other hand, has an interface where a virtual joystick is provided to control the base and the shoulder motor, and virtual sliders to control the elbow motor and linear actuator. The application also has two virtual push buttons for the purpose of holding and releasing an object from the hand gripper.

Smartphone controlled maneuvering is achieved by implementing Internet of Things technology by using Particle Photon IoT device. It is connected to both the Arduinos such that the signals from the smartphone application are used to control the arm in place of joysticks.

The power source for the arm is multiple NiMH batteries with 6V and 12V rating. The 12V battery is used for powering DC motors and linear actuators whereas the 6V battery is used for powering the servo motor. Both the sources are equipped with switches to control the circuit. Figure 12 shows the wooden board with all the electrical components placed over it.

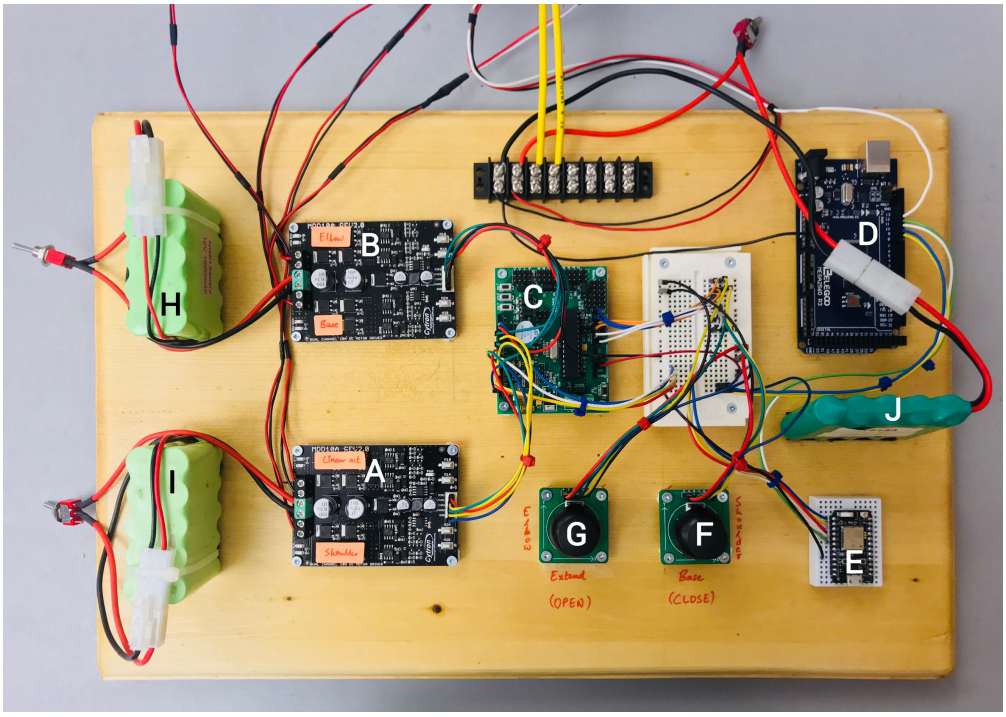


Fig. 12: A – Cytron Motor driver 1 ; B – Cytron Motor driver 2; C – Arduino Duemilanove; D – Arduino MEGA; E – Particle Photon; F – Joystick 1; G – Joystick 2; H – 12V NiMH Battery; I – 12V NiMH Battery; J – 6V NiMH Battery

The complete wiring diagram of the circuit is provided in Appendix B.

2.6 Software

Source code for Arduino Duemilanove and Arduino MEGA was developed on Arduino IDE while that for Particle Photon was developed on Particle IDE. The code setup follows standard practices, setting up pin modes, libraries, and initialization print statements. Two different versions of code were developed, one for the joysticks control and the other being smartphone controlled. In the future versions of the arm, a single code can be developed such that either the smartphone or the joysticks control the arm but not both.

The void loop of the code is a series of if/else blocks corresponding to different motor actions. At the start of void loop, in case of joystick-control, the program reads the X and Y values from the

joysticks and keeps comparing the values with previously-set tolerance values to assess the direction in which the joystick is being moved so as to intuitively reflect the action on to the robotic arm. See Appendix C for the commented code for Arduino Duemilanove, Arduino MEGA. See Table 3 for a summary of the joysticks control scheme.

Table 3: Joysticks control scheme

Joystick No.	X-axis	Y-axis	Button
1	Base (CW)	Shoulder	Gripper Close
2	Extend	Elbow	Gripper Open

The actuators' subroutines use the analogWrite() function to write the speed of rotation to the actuator. The more extreme the X and Y values of the joystick, the faster the motors rotate. The servo subroutine uses Pulse Width Modulation to write a position to the servo enabling it to open or close the hand gripper.

Figure 13 shows the joysticks setup on the wooden board.

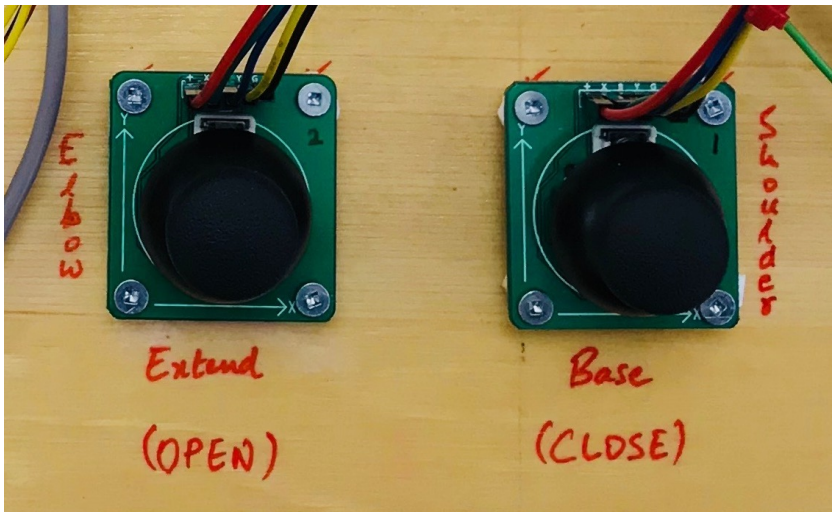


Fig. 13: Joysticks setup

The smartphone application was implemented by using a framework called Blynk. Blynk is a framework over which the user can add the tools like sliders, joysticks, push buttons graphically without the need to write a code. A single piece of code is needed to be flashed on Particle Photon such that it gives Blynk application the access to Particle Photon IoT device. This piece of code, provided in Appendix C.2 was generated on Particle Web IDE and flashed onto the Particle Photon from the same place. A screenshot of the application is shown in Figure 14.

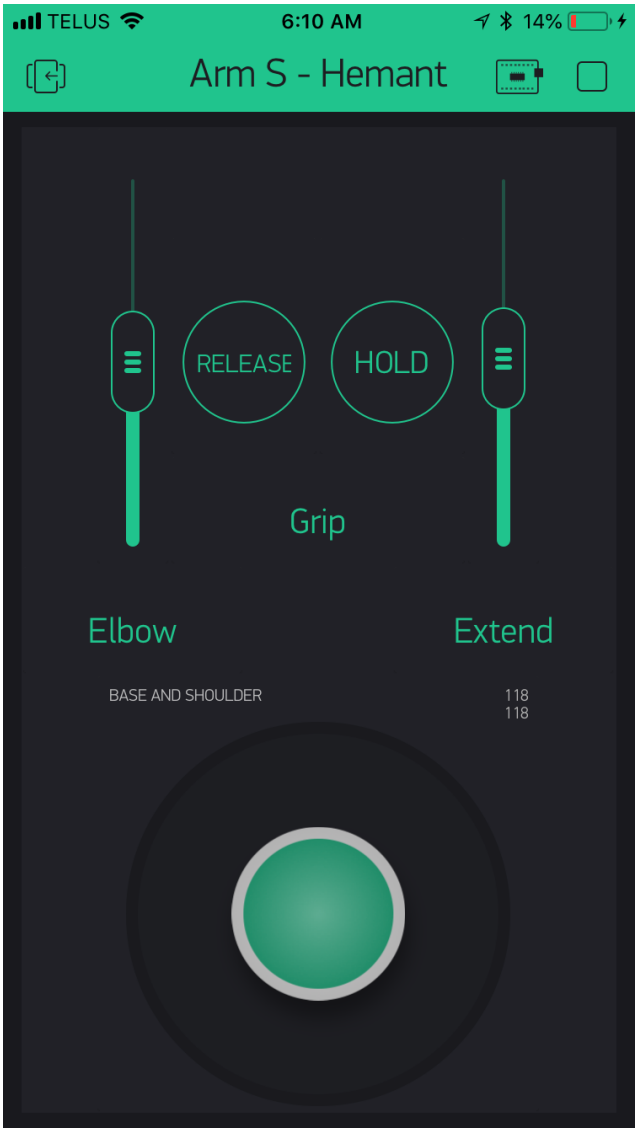


Fig. 14: Blynk Smartphone application interface

The control system is fairly straight forward as it simply reflects the actions of the joysticks on to the motors. However, with upgraded hardware, namely motor encoders for position feedback, a more sophisticated control system can be implemented where the arm senses its position and acts accordingly. The movement of the arm can be fully automated in further versions so that the arm picks objects with intuitive joysticks.

2.7 Financial analysis

The goal of this project was to design a low-cost robotic arm with capabilities discussed in the beginning of the report. According to our initial estimates, the project cost could have been around $1,000 CAD but the whole project was completed in just $802 CAD.

Table 4 shows a detailed breakdown of all costs associated with the robotic arm. Parts were purchased by multiple people for this arm, so the final column indicates the purchaser of the component.

Table 4: Detailed list of components purchased

Sl. No.	Item	Description	Quantity	Price
1.	IRUI Turbine DC Motor (Shoulder)	Worm Geared Double Shaft – 12V – 3 A – 10 RPM – 11.7 Nm	1	137.03
2.	Uxcell DC Motor (Base)	Worm Geared Single Shaft – 12V 7 RPM– 4.9 Nm	1	47.23
3.	Uxcell DC Motor (Elbow)	Worm Geared Double Shaft – 12V – 0.85 A – 10RPM – 3.9 Nm	1	43.65
4.	PA-07-10-5 Micro Linear Actuator	Progressive Automations – 10” stroke – 0.59”/s	1	113.39
5.	Motor Driver	Cytron Dual Channel DC Motor Driver – 10 A – 5-30 V	3	102.11
6.	Arduino Duemilanove	Lynxmotion - BotBoarduino	2	91.98
7.	Aluminum extrusion 40x40mm	Black – 500mm	2	43.10
8.	Aluminum extrusion 20x20mm	Black – 300mm – (Pack of 4)	1	22.17

9.	M4 Nut	For 20mm Aluminum extrusion (Pack of 10)	1	6.17
10.	Internal Steel Connector	For 20mm Aluminum extrusion	2	5.70
11.	90 Internal brackets	For 20mm Aluminum extrusion (Pack of 4)	3	13.86
12.	90 External brackets	For 20mm Aluminum extrusion (Pack of 4)	3	6.00
13.	DC Motors (Testing)	Worm Geared Single Shaft – 12 V – 1.14A – 20 RPM – 2.7 Nm	3	67.90
14.	Set screw hub – 10mm	Actobotics	2	13.78
15.	Set screw hub – 8mm	Actobotics	4	27.56
16.	Joystick Breakout Board	2-Axis and 1 Button	2	6.06
17.	MPU-6050	6 DOF Gyro Accelerometer IMU	2	21.02
18.	3/8" x 1/4" Standoffs	Lynxmotion Nylon (Pack of 4)	1	6.38
19.	3/4" x 1/4" Standoffs	Lynxmotion Nylon (Pack of 4)	1	6.78
20.	Ball Bearing Drawer Slider	14" – 27.5"	1	20.00
Total (Including taxes)				**$802**

As seen above, the total cost of the arm was approximately $802. This is much less expensive than most robotic arms currently in the market today. As the arm is built using aluminum extrusions, the arm is structurally very strong and is capable of sustaining relatively heavy loads. The redundant expenditure includes three DC motors for testing worth $67.90 and two MPU6050 IMUs worth $21.02. Similarly, the cost of Arduino MEGA and Particle Photon IoT device was not accounted as it was sourced from the team. Hence, the savings and expenditures closely cancel out each other. Further, when built on an industrial level for mass production, the cost of production reduces by a large margin, at the same, labour cost adds up. Finally, one can say that, it costs significantly lower than the robotic arms already present in the market.

3. Testing and Results

3.1 General testing and troubleshooting

Initially, test motors of lower power independently connected to the Arduino, were used to build the main code. The motors were initially tested individually to ensure proper functioning of the

motors free of applied loads. With this set-up, all five motors could be controlled with one Arduino board and code for each motor was developed.

During the testing process, a problem was encountered with the working of servo motor at the hand gripper. The servo motor is controlled using the select buttons provided on the joysticks. As servo motor was kept stationary and other motors were being manoeuvred, servo motor was receiving junk values via joysticks' buttons due to which the servo motor was stuttering. This stuttering is enough to drop down an object which is held by the hand gripper. As the source of these signals was joysticks' buttons, many trials were conducted to resolve the issue of junk values but to no avail. As a final resort, the joysticks buttons are connected to a secondary microcontroller sourced from the team, i.e. Arduino MEGA so that the disturbances caused by the other motors does not influence the hand gripper. The result is a perfectly working hand gripper free of issues.

One can observe that the 3D Render has two ball-bearing slider draws whereas the final product has only one. The objective of this was to reduce the weight and eventually increasing the load capacity of the arm. The functionality of the arm extension is observed to be not influenced by the removal of one slide draw.

One of the major software issues faced by the team was with Arduino and Particle integration. As a means to control the robotic arm using the smartphone application, analog signals had to be transmitted to the Arduino from Particle. The team could witness the signal value reflect on the Particle but not on the Arduino and the robotic arm. Using the Serial monitor to check the signal value, gibberish values were being received making the Arm's motors to stutter. After a long inspection for the possible sources of error in the source code and other hardware components, the team developed a code to average the gibberish values for a small period of time. As a result, the

averaged value tracked the analog value as the robot was being controlled by the smartphone application.

It became clear that a PWM approximation of the analog value was being transferred not the actual analog signal. Later on, the team understood that the pin through which the Particle was sending the analog signal to the Arduino, didn't have the analogwrite() support as it didn't have an inbuilt DAC. There are many ways to convert PWM values to smooth analog values using coding techniques. One of the ways is to use pulseIn() function. After implementing the fix, the smartphone control worked.

Many such problems were tackled by creatively singling out the problems, finding the solution after having a backup of the previous code, and properly integrating the fix into the main code, one after the other.

3.2 Software practices

Code was developed and tested in a methodical way using best practice for naming conventions and commenting to ensure progress through project is easy to trace. Sections of the code are distinctly separated from each other to ensure that the person going through the code does not face difficulty in assessing or improving it.

As further code was developed and issues arose, these test codes were constantly re-uploaded to confirm motors still operated as intended. This allowed for easier troubleshooting, since it quickly became clear whether hardware or software issues had arisen. Using the test codes, the team could zero in on the sources of the errors and change the code accordingly.

3.3 Results

After the arm was assembled, wired properly, and tested using the final version of the software, a series of performance analyses were done.

3.3.1 Speed Test

The purpose of these tests was to determine the operating times of each automated component through a series of basic movements.

For each motor, and the performance test was performed five times in a row. The time per individual motion was then calculated as the total trial time divided by the number of repetitions. This method was used in order to reduce human error from timing a very short motion. The results are summarized in Table 5.

Table 5: Average time readings for each actuator.

Motor	Movement range	Time taken (s)
Elbow	180°	3s Up and 2.8s Down
Shoulder	90°	1.6s Up and 1.2s Down
Base	180°	4.2s
Linear actuator	10 in.	17s Up and 16.5s Down

As these speeds were very high, the speed was programmatically decreased to reduce the damage inflicted to the chassis. The percentage of the actual speed retained and the new time readings are noted in the Table 6.

Table 6: The percentage of the actual speed retained and the new time readings for each actuator.

Motor	Movement range	Percentage	Time taken (s)
Elbow	180°	75%	2.2s Up and 2.1s Down

Shoulder	90°	50%	0.7s Up and 0.6s Down
Base	180°	70%	2.9s
Linear actuator	10 in.	100%	17s Up and 16.5s Down

3.3.2 Current draw

After finding out the ideal motor speeds for optimal physical performance, the next test that was conducted is Current draw experiment. An ammeter was connected in series with the shoulder motor's terminal to measure the amount of current being drawn by the shoulder motor while being operated at various loads. Figure 15 shows various weights that were used in the testing process and weight being suspended down the arm's hand gripper.

Fig. 15: Experimental setup consisting of various loads

Shoulder motor was selected for this experiment is because it is the actuator which sustains the maximum torque. The current draw follows the applied torque at the motor's shaft. The measurement findings can be seen in Table 7.

Table 7: Current vs Load

Load	Arm extension (Linear actuator)		
	0" (Zero)	**5" (Half)**	**10" (Full)**
No load	1.48 A	1.84 A	2.20 A
100g	1.70 A	2.06 A	2.35 A
200g	1.78 A	2.16 A	2.45 A
300g	1.84 A	2.25 A	2.62 A
400g	1.92 A	2.36 A	2.80 A
500g	2.03 A	2.53 A	3.05 A
600g	2.20 A	2.75 A	-

The final weight of 600g was not added to the arm at full extension due to concerns about its structural capacity.

3.3.3 Maximum load

After noting down all the readings, an experiment was conducted to know the maximum load which the robotic arm is capable of handling. In the initial series of load addition with completely retracted arm, it was found that the arm is capable of lifting 1000g of weight. As the load capacity certainly depends on the length of extension, it was found that a fully extended arm is capable of lifting a maximum weight of only 600g.

The limitation to the load, contrary to the popular belief in most cases, is the arm's chassis not the motor's capacity. The motors are capable of lifting weights well above the load capacity found in this experiment. As the weights were being added, there was an extreme force being applied at the U-shaped part of the shoulder motor. The 90-degree brackets eventually broke making the team to replace them with new ones.

Hence, it can be concluded that with a stronger chassis, the robot would be capable of lifting weights in excess of 1500g at the end effector. This makes the robot apt for everyday groceries, where the robotic arm is can easily help in picking grocery items.

3.3.4 Pick and place test

The robotic arm also underwent many real-life test scenarios which could assess its capability in real world. The robotic arm was tested both using joysticks and smartphone application. In both cases, the arm could pick and place the test object, i.e. a pill box in a container within a relatively small amount of time. With the user getting more used to the control, the task times could be further lessened.

Figures 16 and 17 illustrate two scenarios where the arm was controlled using joysticks and the smartphone application respectively. The videos of both the tests can be made available on demand.

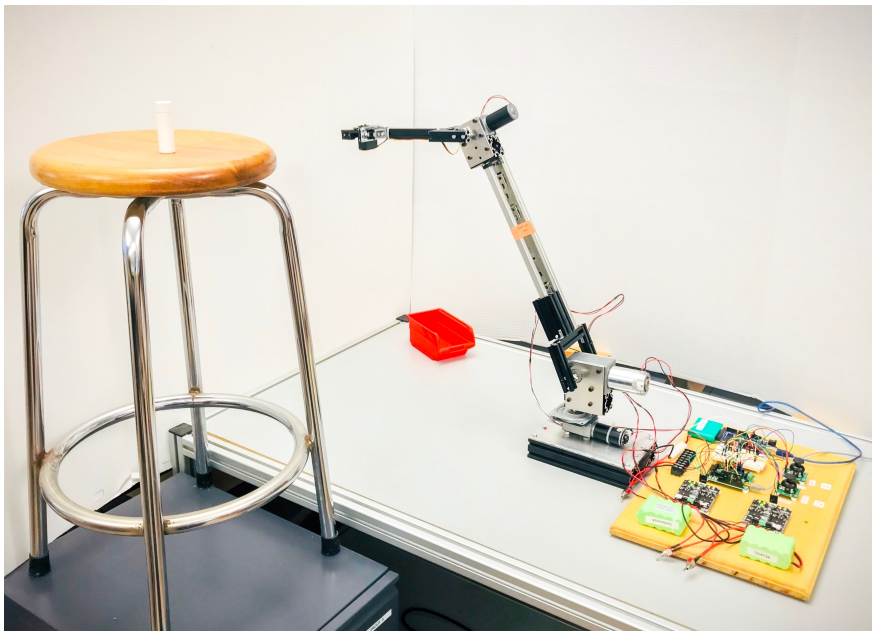


Fig. 16: Pill box test using joysticks

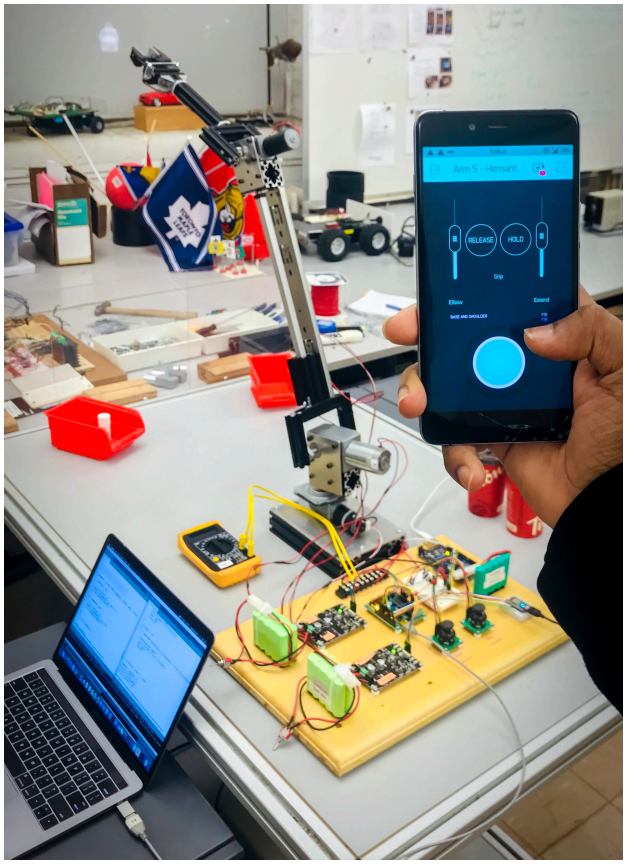


Fig. 17: Pill box test using smartphone application

4. Conclusions and Recommendations

4.1 Conclusion

The developed robotic arm is capable of performing the tasks of picking and placing the objects from one place to another within an acceptable time duration. Hence, it can be concluded that the arm will be successful in real world where the physically handicapped person will use it to shop groceries and will use it for other related tasks.

The arm was assembled for $802, which is vastly lower than the robotic arms already available in the market in the present day. Testing the robotic arm revealed that the actuators were able to move at a reasonable speed, with each of the revolute joints able to turn within two seconds, and the linear actuator taking relatively longer to fully extend and retract, around 16.5 seconds each way.

Yet, some improvements can be made to make the robotic arm more efficient and useful, the details of which are discussed in the Recommendations section.

4.2 Recommendations

In the testing phase, the 90 degrees brackets broke when the robotic arm was put to the high load test (600g load with arm fully extended). The next version of this arm can have even more stronger chassis which is capable of sustaining much larger weights compared to the present one. This can be achieved by removing the modularity and building much of the chassis from one piece of metal, especially the U-shaped parts of both Shoulder and Elbow module.

In the electrical part, there is a redundant microcontroller which was used only to isolate the DC motor signals from the servo motor terminals. Hence, the next version of the arm can use a different approach to tackle this problem without using a new microcontroller.

The smartphone application can be equipped with a vibration feedback which alert the use of extremities. Further, the whole smartphone-based control can be implemented using other means, such as Bluetooth technology or Wi-Fi. In this project, IoT was used due to a couple of reasons, the first being the ready availability of the Particle Photon IoT device with the team, and the second, experience of the team in developing IoT based projects on Particle Photon.

In the control system part, a better control can be developed which makes use of sensors' data to locate the object when approached by the hand gripper, and picks the object with proper understanding of the force being applied in order to not destroy the object. This makes the robotic arm more intuitive and helps the user in reducing the time required for the task-completion.

Acknowledgements:

I am very thankful to my supervisor Prof. Brian Surgenor for his guidance, support, and motivation throughout the project.

I am grateful to my parents for providing me support and encouragement. I also like to thank Queen's University for this MEng. Project.

References

1. Eitel, Lisa. What are dc motors and how do definitions vary? Technical summary for engineers. 4 October 2011. Retrieved from https://www.motioncontroltips.com/dc-motors/.
2. Medi, Norman. Quick Learning – How DC Motors Work. 1 March 2018. Retrieved from https://meee-services.com/quick-learning-dc-motors-work/
3. How Worm Gear Motors Work. 21 July 2017. Retrieved from https://bauergmc.com/how-worm-gear-motors-work.html
4. Cass, K., Fumat, S., Futterer, T., Holm, Q., & Beaudoin, M. MECH 462 Final Report: Grizpaw Controlled Robotic Arm. 11 April 2018.

Appendix A - Motor Specifications and Datasheets

A.1: IRUI Turbine DC Motor (Shoulder) - (Worm Geared Double Shaft – 10 RPM – 11.7 Nm)

Note: Due to the unavailability of datasheet, a screenshot of the product page is provided.

Product description

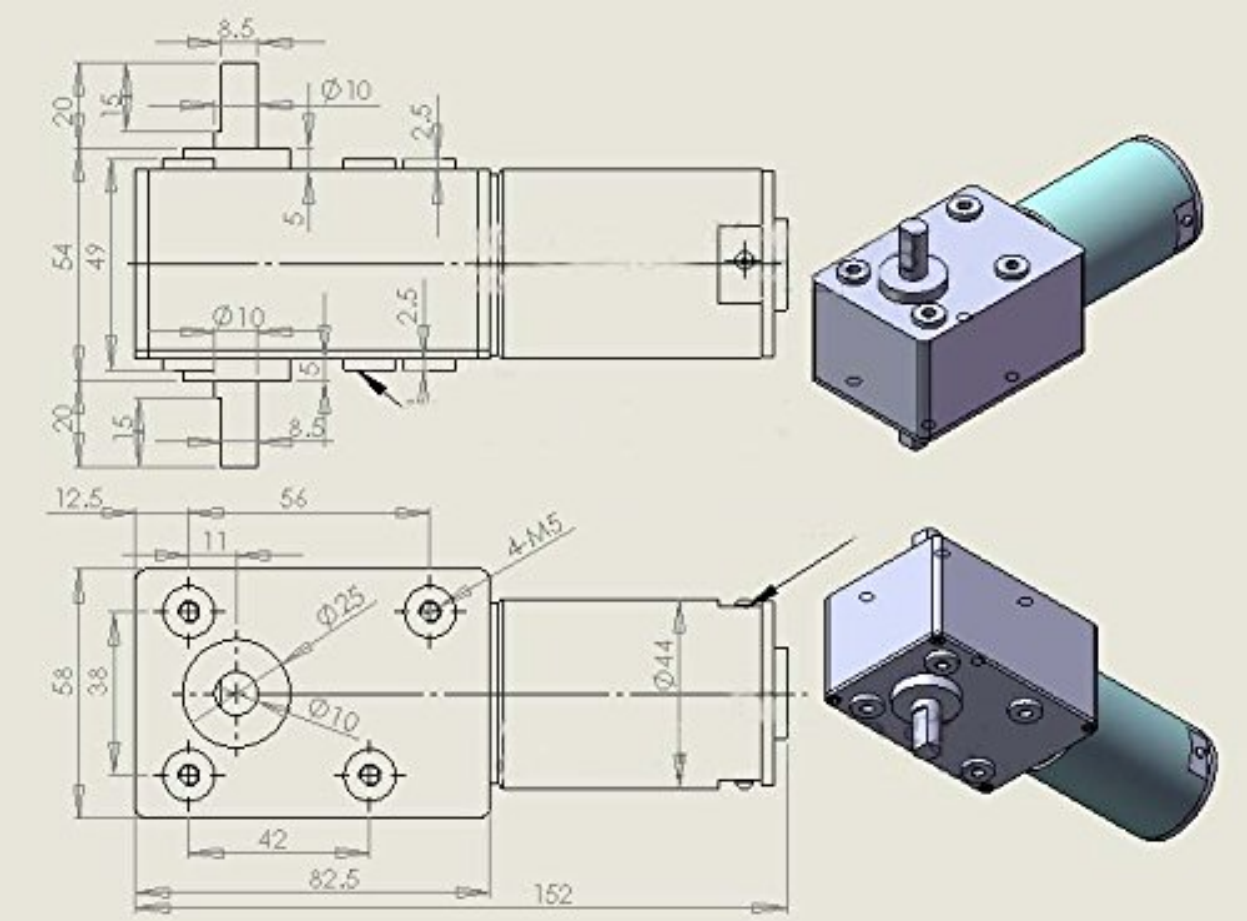
Size :12V 3A 10RPM 120Kg.cm | Color Name:Dual Shaft

Notice: This motor have single and dual output shaft, the pictures just show the dual output shaft worm speed reduction gear motor, if need the single our shaft gear motor, please just choose it in the Classification. Many thanks.

Descriptions:
This motor is a mininature with DC turbine worm speed reduction gear motor, which could change the wiring of the positive & negative and the rotation of the motor.

Specifications:
Product Name: DC Worm Speed Reduction Gear Motor
Output Shaft: Dual Model: GW600-R10D
Rated Voltage: 12V
No-load Speed: 10rpm
Rated Speed: 7.6rpm
Rated Torque: 120kg.cm
Reated Current: 3A
Body Size: 152 x 58 x 54mm
Shaft Diameter: 10mm
Net Weight: 1000g

Package Includes:
1 x Gear Motor



A.2: **Uxcell DC Motor (Base) -** (Worm Geared Single Shaft – 7 RPM – 4.9 Nm)

Note: Due to the unavailability of datasheet, a screenshot of the product page is provided.

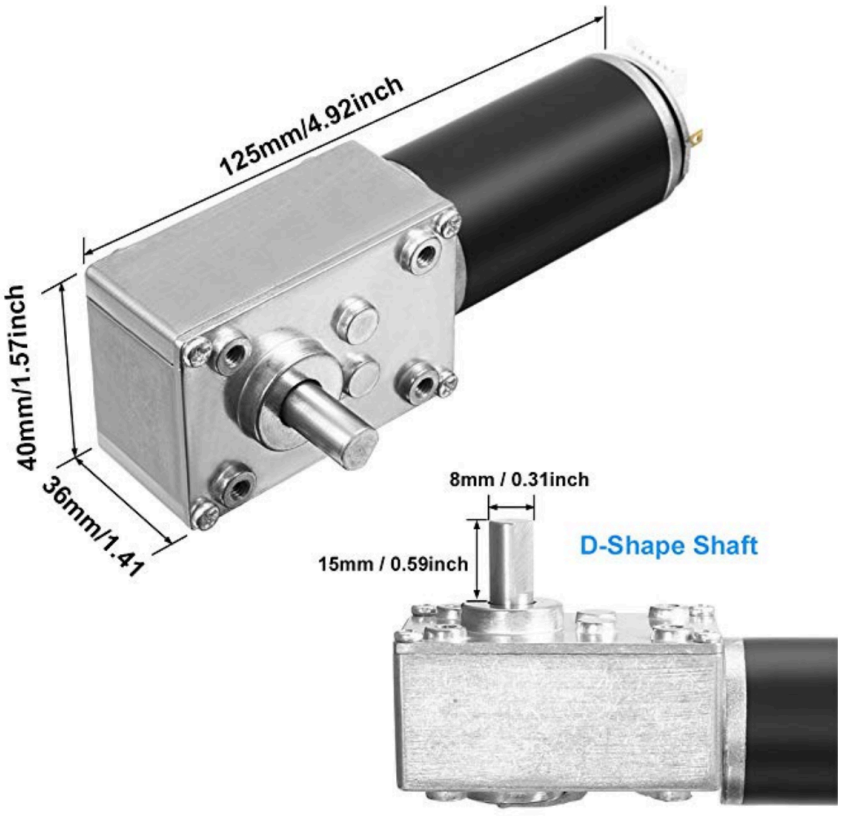
Product description

Description
DC 12V 7RPM 50Kg.cm Self-Locking Worm Gear Motor With Encoder And Cable, High Torque Speed Reduction Motor

Specification:
Voltage: DC12V
No-Load Speed: 7rpm
Reduction Ratio: 1:522
Torque: 50Kg.cm
Error: ±10%
Wire Length: 20cm / 7.87inch
Output Shaft Diameter: 8mm / 0.31inch D-type
Output Shaft Length: 15mm / 0.59inch
Total Size(not include shaft):40*36*125mm / 1.57*1.41*4.92inch

Wirings:
Notice:
1. The yellow and green wires are the speed signal output line, and they can't be connected any power supply, or it will be burnt.
2. Positive and negative power supply of encoder do not allow connect wrong;voltage is 3.3-5V.
3. Control CW or CCW of motor by changing the positive and negative power supply of encoder (change Red Wire and White Wire connect)
Red Wire - positive power supply of motor(+)
White Wire - negative power supply of motor(-)
Black Wire - negative power supply of encoder(-) (positive and negative power supply of encoder do not allow connect wrong;voltage is3.3-5V)
Blue Wire - positive power supply of encoder(+) (positive and negative power supply of encoder do not allow connect wrong;voltage is 3.3-5V)
Yellow Wire - signal feedback (11 signals per motor turns a circle)
Green Wire - signal feedback (11 signals per motor turns a circle)

Package Include:
1 x motor , 1 x 6-Wire Connector



A.3: **Uxcell DC Motor (Elbow) -** (Worm Geared Double Shaft – 10 RPM – 3.9 Nm)

Note: Due to the unavailability of datasheet, a screenshot of the product page is provided.

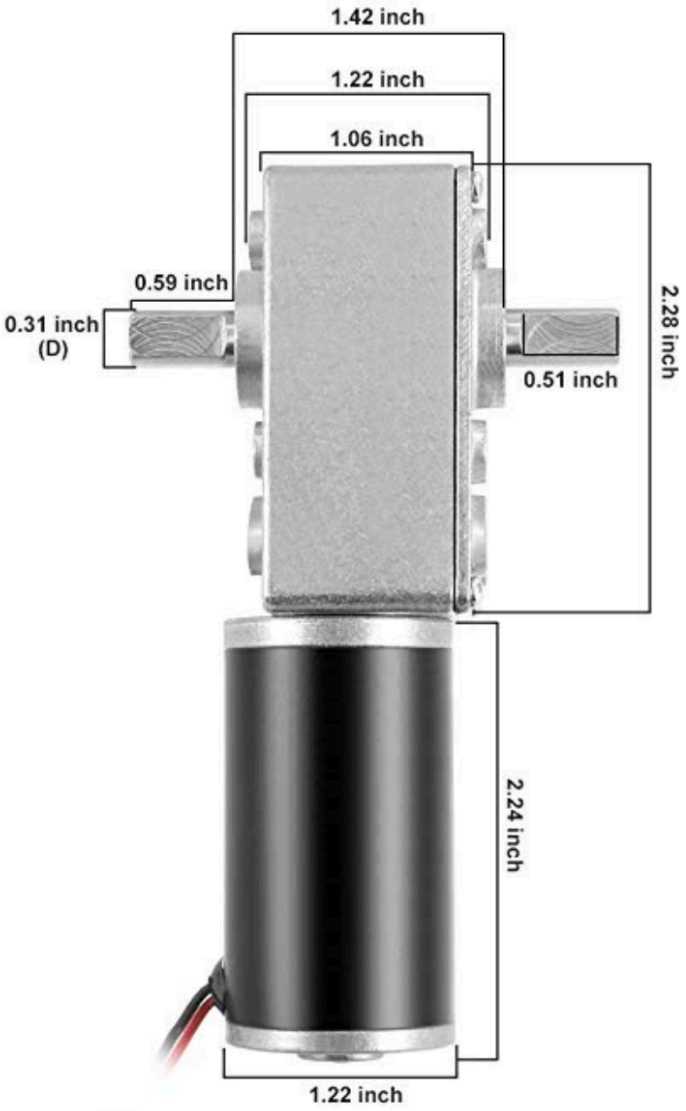
Product description

This type is miniature Worm Gear DC Motor, which can change shaft rotation direction while the wiring positive andnegative be changed. It is also have addition two characteristics:
1 With self-locking, the output shaft can not rotation when switch off, that is self-locking.
2 Gearbox output shaft and motor shaft are come to be a rectangle, it is widely used in various of occasions that require special installation size.

Specification:
Rated Voltage: 12V DC;
No-Load Speed: 10 RPM;
Output Shaft: 8mm x 15mm/1.22inch x 2.24inch(D*L);
DC Motor Body Dimensions: 31 x 57mm/ 1.22inch x 2.24inch(D*L);
Gear Box Dimensions: 58 x 40 x 35mm/ 2.28inch x 1.57inch x 1.37inch(L*W*H);
Wire Length : 20cm;

Applications: Boat, Car, Electric Bicycle, Fan, Home Appliance, BBQ, Machinery and equipment, automation instruments, electric curtain, stage lighting, counter, faucet, safe, wet towel dispenser, vending machines, automatic monitoring and other modern art.

Package Contents:1 x Gear Box Motor



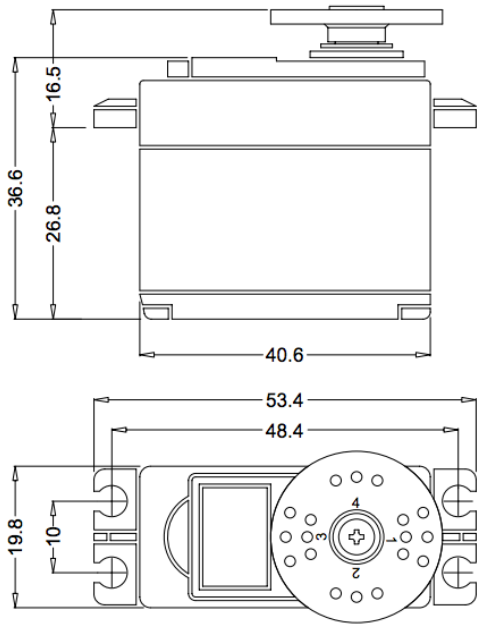
A.4: HITEC Servo motor (Hand Gripper) - (HS-425BB – 6 V)

°

ANNOUNCED SPECIFICATION OF HS–425BB STANDARD DELUXE BALL BEARING SERVO

1.TECHNICAL VALUES

CONTROL SYSTEM	: +PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	: 4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	: -20 T0 +60° C	
TEST VOLTAGE	: AT 4.8V	AT 6.0V
OPERATING SPEED	: 0.21sec/60° AT NO LOAD	0.16sec/60° AT NO LOAD
STALL TORQUE	: 3.3kg.cm(45.82oz.in)	4.1kg.cm(56.93oz.in)
OPERATING ANGLE	: 45°ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	: CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
CURRENT DRAIN	: 8mA/IDLE AND 150mA/NO LOAD RUNNING	
DEAD BAND WIDTH	: 8usec	
CONNECTOR WIRE LENGTH	: 300mm(11.81in)	
DIMENSIONS	: 40.6x19.8x36.6mm(1.59x0.77x1.44in)	
WEIGHT	: 45.5g(1.6oz)	



2.FEATURES
3-POLE FERRITE MOTOR
LONG LIFE POTENTIOMETER
DUAL BALL BEARING
INDIRECT POTENTIOMETER DRIVE

3.APPLICATIONS
AIRCRAFT 20-60 SIZE
30 SIZE HELICOPTERS
STEERING AND THROTTLE SERVO FOR CARS
TRUCK AND BOATS

A.5: PA-07-10-5 Micro Linear Actuator - (Progressive Automations – 10” stroke – 0.59”/s)

PROGRESSIVE AUTOMATIONS
FORWARD THINKING

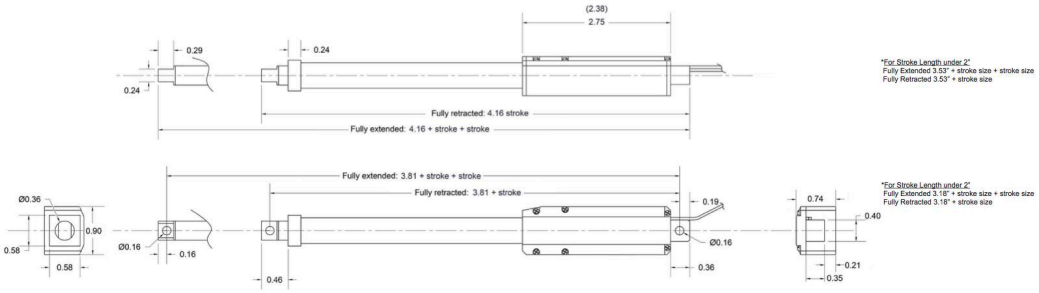
PA-07 MICRO LINEAR ACTUATOR

HOW TO ORDER
1.800.676.6123
sales@progressiveautomations.com
www.progressiveautomations.com

SPECIFICATIONS

- Input Voltage: 12VDC
- Current: 0.24A at full load
- Load Capacity: 5lbs
- Static load: 6.5lbs
- Stroke Length: 0.50” to 12”
- Mounting Holes: 0.16” diameter
- Screw: Acme Screw
- Duty Cycle: 10%
- Operational Temperature: -0°C~50°C (-32°F~122°F)
- Limit Switch: Built in, Non -Adjustable
- IP Grade: IP66
- Low Noise: db<45(A)
- Certification: CE and ROHS
- Housing: Plastic and Aluminum Alloy
- Gear: Plastic Gears
- Wire Length: 8”

DIMENSIONS IN INCHES



PROGRESSIVE AUTOMATIONS
FORWARD THINKING

PA-07 MICRO LINEAR ACTUATOR

HOW TO ORDER
1.800.676.6123
sales@progressiveautomations.com
www.progressiveautomations.com

12 VDC CURRENT VS LOAD

HOLE TO HOLE DIMENSIONS

Stroke Size (in inches)	Fully Retracted (in inches)	Fully Extended (in inches)
0.5	3.68	4.18
0.8	3.98	4.78
1	4.18	5.18
2	5.81	7.81
4	7.81	11.81
6	9.81	15.81
8	11.81	19.81
10	13.81	23.81
12	15.81	27.81

12 VDC SPEED VS LOAD

PRODUCT ACCESSORIES

- Will work with most of PA’s AC & DC controls boxes
- Mounting Brackets: BRK-07 (for each end);
- Rocker Switches: any PA’s Rocker Switches
- Longer Wires: AC-01
- Foot Controls: PDL-01 and PDL-03

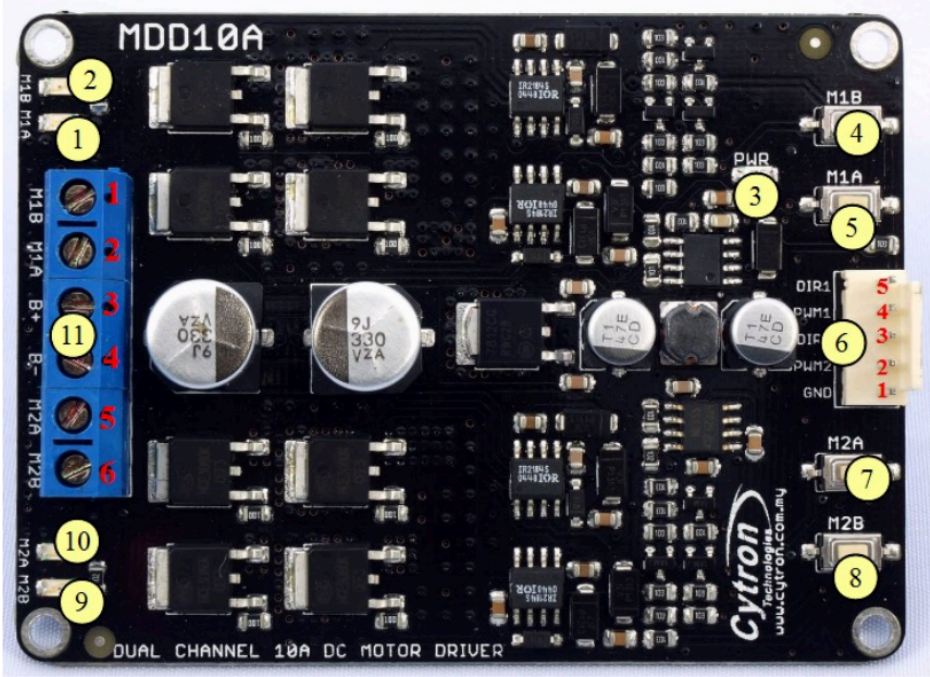
SPEED IN INCHES PER SECOND

Forces (lbs)	No Load	Full Load
5	0.59	0.55

Page 2 of 5

A.6: Cytron Dual Channel DC Motor Driver – 10 A – 5-30 V

5. BOARD LAYOUT



1. Red LED M1A – Turns on when the output M1A is high and output M1B is low. Indicates the current flows from output M1A to M1B.

2. Red LED M1B – Turns on when the output M1A is low and output M1B is high. Indicates the current flows from output M1B to M1A.

3. Green LED – Power LED. Should be on when the board is powered on.

4. Test Button M1B – When this button is pressed, current flows from output M1B to M1A and motor will turn CCW (or CW depending on the connection).

5. Test Button M1A – When this button is pressed, current flows from output M1A to M1B and motor will turn CW (or CCW depending on the connection).

3. PRODUCT SPECIFICATION AND LIMITATIONS

Absolute Maximum Rating

No.	Parameters	Min	Typical	Max	Unit
1	Power Input Voltage	5	-	25	V
2	I_{MAX} (Maximum Continuous Motor Curren	-	-	10	A
3	I_{PEAK} – (Peak Motor Current) *	-	-	30	A
4	V_{IOH} (Logic Input – High Level)	3	-	5.5	V
5	V_{IOL} (Logic Input – Low Level)	0	0	0.5	V
6	Maximum PWM Frequency	-	-	20	KHz

* ***Must not exceed 10 seconds.***

6. Input

Pin No.	Pin Name	Description
1	GND	Ground
2	PWM2	PWM input for speed control (Motor 2)
3	DIR2	Direction input (Motor 2)
4	PWM1	PWM input for speed control (Motor 1)
5	DIR1	Direction input (Motor 1)

The truth table for the control logic for motor 1 and motor 2 are as follow:

PWM	DIR	Output A	Output B
Low	X(Don't care)	Low	Low
High	Low	High	Low
High	High	Low	High

7. Test Button M2A – When this button is pressed, current flows from output M2A to M2B and motor will turn CW (or CCW depending on the connection).

8. Test Button M2B – When this button is pressed, current flows from output M2B to M2A and motor will turn CCW (or CW depending on the connection).

9. Red LED M2B – Turns on when the output M2A is low and output M2B is high. Indicates the current flows from output M2B to M2A.

10. Red LED M2A – Turns on when the output M2A is high and output M2B is low. Indicates the current flows from output M2A to M2B.

11. Terminal Block – Connect to motor and power source.

Pin No	Pin Name	Description
1	Motor 1 Output B	Connect to motor 1 terminal B
2	Motor 1 Output A	Connect to motor 1 terminal A
3	POWER +	Positive Supply
4	POWER -	Negative Supply
5	Motor 2 Output A	Connect to motor 2 terminal A
6	Motor 2 Output B	Connect to motor 2 terminal B

Appendix B: Wiring diagram

Note: Four single-channel motor drivers are used in the wiring diagram instead of two double-channel motor drivers.

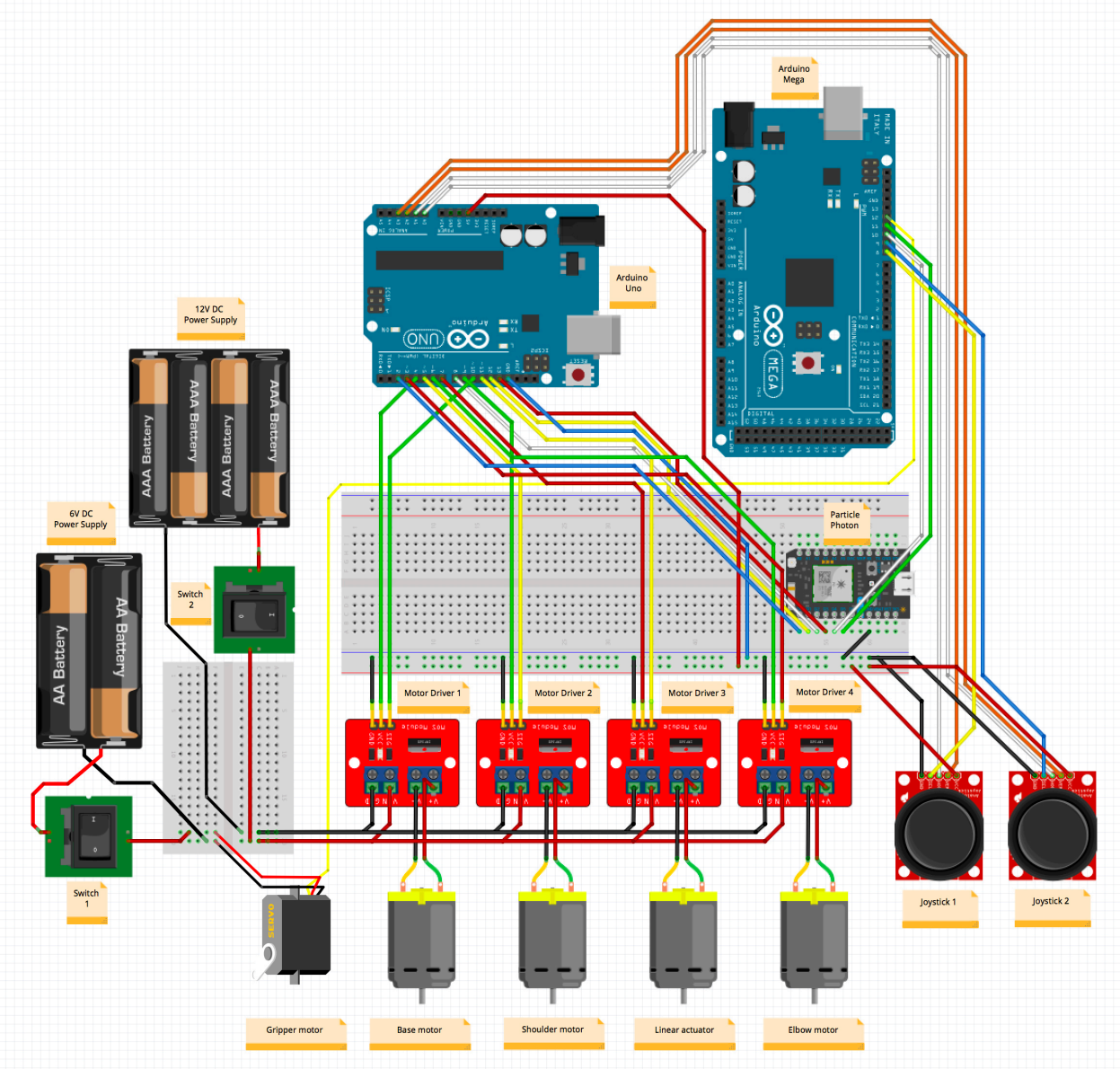


Fig. 18: Wiring diagram

Appendix C: Source code

C.1: Arduino

C.1.1: Source code for Arduino Duemilanove - Joysticks control

```
/////JOYSTICKS' PINS/////
int pin1x = 0;
int pin2x = 1;
int pin1y = 2;
int pin2y = 3;
int pin1btn = 18;
int pin2btn = 19;

/////JOYSTICKS' VARIABLES/////
int joy1x = 0;
int joy2x = 0;
int joy1y = 0;
int joy2y = 0;
int joy1btn = 0;
int joy2btn = 0;

/////ELBOW MOTOR PINS/////
int pwmElbow = 3;
int dirElbow = 6;
int i;

/////LINEAR ACTUATOR PINS/////
int pwmLin = 11;
int dirLin = 7;
int l;

/////SHOULDER MOTOR PINS/////
```

```
int pwmShould = 5;
int dirShould = 9;
int j;

/////BASE MOTOR PINS/////
int pwmBase = 10;
int dirBase = 4;
int k;


void setup()
  {
    pinMode(pwmElbow, OUTPUT);
    pinMode(dirElbow, OUTPUT);

    pinMode(pwmLin, OUTPUT);
    pinMode(dirLin, OUTPUT);

    pinMode(pwmShould, OUTPUT);
    pinMode(dirShould, OUTPUT);

    pinMode(pwmBase, OUTPUT);
    pinMode(dirBase, OUTPUT);


    pinMode(pin1btn, INPUT);
    pinMode(pin2btn, INPUT);

//    pinMode(servoPin, OUTPUT);

    //myservo.attach(12);
```

```
    Serial.begin(9600);
    Serial.println("Joystick control started");
  }

void loop()
  {
    joy2x = analogRead(pin2x); //495
    joy2y = analogRead(pin2y); //507
    joy1x = analogRead(pin1x);
    joy1y = analogRead(pin1y);
    joy1btn = digitalRead(pin1btn);
    joy2btn = digitalRead(pin2btn);

////////// JOYSTICK 1 - Y AXIS //////////
    if (joy1y > 527)
      {
        digitalWrite(dirShould, LOW);
        joy1y = map(joy1y, 527, 1023, 0, 100);
        analogWrite(pwmShould, joy1y);
      }

    else if (joy1y < 510)
      {
        digitalWrite(dirShould, HIGH);
        joy1y = map(joy1y, 0, 510, 120, 0);
        analogWrite(pwmShould, joy1y);
      }

    else
      {
        analogWrite(pwmShould, 0);
      }
```

```
////////// JOYSTICK 1 - X AXIS //////////
      if (joy1x > 530)
      {
        digitalWrite(dirBase, HIGH);
        joy1x = map(joy1x, 530, 1023, 0, 180);
        analogWrite(pwmBase, joy1x);
      }

    else if (joy1x < 510)
      {
        digitalWrite(dirBase, LOW);
        joy1x = map(joy1x, 0, 510, 180, 0);
        analogWrite(pwmBase, joy1x);
      }

    else
      {
        analogWrite(pwmBase, 0);
      }

////////// JOYSTICK 2 - Y AXIS //////////
    if (joy2y > 505)
      {
        digitalWrite(dirElbow, LOW);
        joy2y = map(joy2y, 505, 1023, 0, 200);
        analogWrite(pwmElbow, joy2y);
      }

    else if (joy2y < 485)
      {
        digitalWrite(dirElbow, HIGH);
```

```
        joy2y = map(joy2y, 0, 485, 200, 0);
        analogWrite(pwmElbow, joy2y);
      }

    else
      {
        analogWrite(pwmElbow, 0);
      }

////////// JOYSTICK 2 - X AXIS //////////
      if (joy2x > 510)
      {
        digitalWrite(dirLin, LOW);
        joy2x = map(joy2x, 510, 1023, 0, 255);
        analogWrite(pwmLin, joy2x);
      }

    else if (joy2x < 490)
      {
        digitalWrite(dirLin, HIGH);
        joy2x = map(joy2x, 0, 490, 255, 0);
        analogWrite(pwmLin, joy2x);
      }

    else
      {
        analogWrite(pwmLin, 0);
      }
  }
```

C.1.2: Source code for Arduino MEGA - Joysticks control

```
int servoPin = 12;
int angle = 0;
int pulseWidth = 0;
int token = 28;

/////JOYSTICKS' PINS/////
int pin1x = 0;
int pin2x = 1;
int pin1y = 2;
int pin2y = 3;
int pin1btn = 8;
int pin2btn = 9;

/////JOYSTICKS' VARIABLES/////
int joy1x = 0;
int joy2x = 0;
int joy1y = 0;
int joy2y = 0;
int joy1btn = 0;
int joy2btn = 0;

void setup()
{
  pinMode(pin1btn, INPUT);
    pinMode(pin2btn, INPUT);

    pinMode(servoPin, OUTPUT);

    //myservo.attach(12);
```

```
    Serial.begin(9600);
    Serial.println("Joystick control started");
}

void loop() {
    joy1btn = digitalRead(pin1btn);
    joy2btn = digitalRead(pin2btn);

    if (joy1btn == LOW)
      {
        for (angle=token; angle<=165 && (digitalRead(pin1btn)) == LOW &&
(digitalRead(pin2btn)) == HIGH; angle++)
        {
          servoPulse(servoPin, angle);
          Serial.print(angle); Serial.println("HOLD");
          token = angle;
          //delay(20);
        }
      }

      if (joy2btn == LOW)
      {
        for (angle=token; angle>=28 && (digitalRead(pin2btn)) == LOW &&
(digitalRead(pin1btn)) == HIGH; angle--)
        {
          servoPulse(servoPin, angle);
          Serial.print(angle); Serial.println("RELEASE");
          token = angle;
          //delay(20);
        }
      }
    }
```

```
  void servoPulse(int servoPin, int angle)
{
  pulseWidth = (angle*10) + 575;
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseWidth);
  digitalWrite(servoPin, LOW);
}
```

C.1.3: Source code for Arduino Duemilanove - Smartphone control

```
/////PWM READS/////
byte pinAppShould = 2;
byte pinAppElbow = 12;
byte pinAppBase = 8;
byte pinAppLin = 13;

/////SHOULDER VARIABLES/////
int pwmAppShould = 0;
int mapShould = 0;
int oldShould = 0;

/////ELBOW VARIABLES/////
int pwmAppElbow = 0;
int mapElbow = 0;
int oldElbow = 0;

/////BASE VARIABLES/////
int pwmAppBase = 0;
int mapBase = 0;
int oldBase = 0;

/////LINEAR ACTUATOR VARIABLES/////
int pwmAppLin = 0;
int mapLin = 0;
int oldLin = 0;

/////JOYSTICKS' PINS/////
int pin1x = 0;
int pin2x = 1;
int pin1y = 2;
int pin2y = 3;
```

```
int pin1btn = 18;
int pin2btn = 19;

/////JOYSTICKS' VARIABLES/////
int joy1x = 0;
int joy2x = 0;
int joy1y = 0;
int joy2y = 0;
int joy1btn = 0;
int joy2btn = 0;

/////ELBOW MOTOR PINS/////
int pwmElbow = 3;
int dirElbow = 6;
int i;

/////LINEAR ACTUATOR PINS/////
int pwmLin = 11;
int dirLin = 7;
int l;

/////SHOULDER MOTOR PINS/////
int pwmShould = 5;
int dirShould = 9;
int j;

/////BASE MOTOR PINS/////
int pwmBase = 10;
int dirBase = 4;
int k;

void setup()
```

```
{
  pinMode(pwmElbow, OUTPUT);
  pinMode(dirElbow, OUTPUT);

  pinMode(pwmLin, OUTPUT);
  pinMode(dirLin, OUTPUT);

  pinMode(pwmShould, OUTPUT);
  pinMode(dirShould, OUTPUT);

  pinMode(pwmBase, OUTPUT);
  pinMode(dirBase, OUTPUT);



  pinMode(pin1btn, INPUT);
  pinMode(pin2btn, INPUT);

  pinMode(pinAppShould, INPUT);
  pinMode(pinAppElbow, INPUT);
  pinMode(pinAppLin, INPUT);
  pinMode(pinAppBase, INPUT);

  Serial.begin(115200);
  Serial.println("Program started");
}


void loop()
{
 pwmAppShould = pulseIn(pinAppShould, HIGH);
 pwmAppElbow = pulseIn(pinAppElbow, HIGH);
 pwmAppBase = pulseIn(pinAppBase, HIGH);
```

```
 pwmAppLin = pulseIn(pinAppLin, HIGH);

    ////////////////////////////////////////////////////
    ////////// APP JOYSTICK 1 - Y AXIS - SHOULDER //////////
    ////////////////////////////////////////////////////

  if ((pwmAppShould < (oldShould + 200) && pwmAppShould > (oldShould -
200)) || oldShould == 0)
    {
      if (pwmAppShould > 1100)
      {
        digitalWrite(dirShould, LOW);
        Serial.print(pwmAppShould); Serial.print(" Greater ");
Serial.print(oldShould); Serial.println(" Part 1");
        mapShould = map(pwmAppShould, 1100, 1800, 0, 35);
        analogWrite(pwmShould, mapShould);
      }

    else if (pwmAppShould < 700)
      {
        digitalWrite(dirShould, HIGH);
        Serial.print(pwmAppShould); Serial.print(" Lower ");
Serial.print(oldShould); Serial.println(" Part 2");
        mapShould = map(pwmAppShould, 0, 700, 50, 0);
        analogWrite(pwmShould, mapShould);
      }

    else
      {
        analogWrite(pwmShould, 0);
        Serial.print(pwmAppShould); Serial.print(" Middle ");
Serial.print(oldShould); Serial.println(" Part 3");
```

```
      }
      oldShould = pwmAppShould;
    }

      else
      {
      Serial.print(pwmAppShould); Serial.println(" Random");
      oldShould = pwmAppShould;
      }

    ////////////////////////////////////////////////
    ///////// APP JOYSTICK 1 - X AXIS - BASE ///////
    ////////////////////////////////////////////////

  if ((pwmAppBase < (oldBase + 200) && pwmAppBase > (oldBase - 200))
|| oldBase == 0)
    {
      if (pwmAppBase > 1100)
      {
        digitalWrite(dirBase, HIGH);
        mapBase = map(pwmAppBase, 1100, 1800, 0, 70);
        analogWrite(pwmBase, mapBase);
      }

    else if (pwmAppBase < 700)
      {
        digitalWrite(dirBase, LOW);
        mapBase = map(pwmAppBase, 0, 700, 70, 0);
        analogWrite(pwmBase, mapBase);
      }

    else
```

```
      {
        analogWrite(pwmBase, 0);
      }

      oldBase = pwmAppBase;
    }

    else
      {
        oldBase = pwmAppBase;
      }

    ////////////////////////////////////////////////////
    ////////// APP JOYSTICK 2 - Y AXIS - ELBOW //////////
    ////////////////////////////////////////////////////

  if ((pwmAppElbow < (oldElbow + 200) && pwmAppElbow > (oldElbow -
200)) || oldElbow == 0)
    {
      if (pwmAppElbow > 1350)
      {
        digitalWrite(dirElbow, LOW);
        mapElbow = map(pwmAppElbow, 1350, 1960, 0, 75);
        analogWrite(pwmElbow, mapElbow);
      }

    else if (pwmAppElbow < 650)
      {
        digitalWrite(dirElbow, HIGH);
        mapElbow = map(pwmAppElbow, 0, 650, 90, 0);
        analogWrite(pwmElbow, mapElbow);
      }
```

```
    else
      {
        analogWrite(pwmElbow, 0);
      }

    oldElbow = pwmAppElbow;
    }

    else
    {
      oldElbow = pwmAppElbow;
    }

    ////////////////////////////////////////////////////
    ////////// APP JOYSTICK 2 - X AXIS - LINEAR ////////
    ////////////////////////////////////////////////////

  if ((pwmAppLin < (oldLin + 300) && pwmAppLin > (oldLin - 300)) ||
oldLin == 0)
    {
      if (pwmAppLin > 1150)
      {
        digitalWrite(dirLin, LOW);
        //Serial.print(pwmAppLin); Serial.print(" Greater ");
Serial.println(oldLin);
        mapLin = map(pwmAppLin, 1150, 1960, 5, 250);
        analogWrite(pwmLin, mapLin);
      }

    else if (pwmAppLin < 850)
      {
```

```
        digitalWrite(dirLin, HIGH);
        //Serial.print(pwmAppLin); Serial.print(" Lesser ");
Serial.println(oldLin);
        mapLin = map(pwmAppLin, 0, 850, 250, 5);
        analogWrite(pwmLin, mapLin);
      }

    else
      {
        analogWrite(pwmLin, 0);
        //Serial.print(pwmAppLin); Serial.print(" Middle ");
Serial.println(oldLin);
      }

      oldLin = pwmAppLin;
    }

    else
      {
        oldLin = pwmAppLin;
      }

}
```

C.1.4: Source code for Arduino MEGA - Smartphone control

```
int servoPin = 12;
int angle = 0;
int pulseWidth = 0;
int token = 28;

/////JOYSTICKS' PINS/////
int pinOpen = 10;
int pinClose = 11;

/////JOYSTICKS' VARIABLES/////
int varClose = 0;
int varOpen = 0;

void setup()
  {
    pinMode(pinClose, INPUT);
    pinMode(pinOpen, INPUT);

    pinMode(servoPin, OUTPUT);

    Serial.begin(9600);
    Serial.println("Joystick control started");
   }

void loop()
  {
    varClose = digitalRead(pinClose);
    varOpen = digitalRead(pinOpen);
```

```
    if (varClose == LOW)
      {
       for (angle=token; angle<=165 && (digitalRead(pinClose)) == LOW &&
(digitalRead(pinOpen)) == HIGH; angle++)
          {
            servoPulse(servoPin, angle);
            Serial.println("HOLD");
            token = angle;
          }
      }

    if (varOpen == LOW)
      {
       for (angle=token; angle>=28 && (digitalRead(pinOpen)) == LOW &&
(digitalRead(pinClose)) == HIGH; angle--)
          {
            servoPulse(servoPin, angle);
            Serial.println("RELEASE");
            token = angle;
          }
      }

  }


  void servoPulse(int servoPin, int angle)
{
  pulseWidth = (angle*10) + 575;
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseWidth);
  digitalWrite(servoPin, LOW);
}
```

C.2: Particle Photon

Note: Customize the Smartphone application and its components after installing Blynk application on Android/iOS > Replace the authentication token in this code with the authentication key provided by Blynk application > Flash the code to Particle Photon using Particle Web IDE.

```
/*************************************************************
 * Blynk is a platform with iOS and Android apps to control
 * Arduino, Raspberry Pi and the likes over the Internet.
 * You can easily build graphic interfaces for all your
 * projects by simply dragging and dropping widgets.
 *
 *   Downloads, docs, tutorials: http://www.blynk.cc
 *   Blynk community:            http://community.blynk.cc
 *   Social groups:              http://www.fb.com/blynkapp
 *                               http://twitter.com/blynk_app
 *
 * Blynk library is licensed under MIT license
 * This example code is in public domain.
 *
 * WARNING: It is recommended to use SparkCorePolledTimer library
 *         to make periodic actions (similar to SimpleTimer on Arduino).
 *
 *************************************************************/
//#define BLYNK_DEBUG // Uncomment this to see debug prints
#define BLYNK_PRINT Serial
#include "blynk/blynk.h"

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "82da29bf446d4bd7988f73e7c571e83b";
```

```
// Attach a Button widget (mode: Switch) to the Digital pin 7 - and control the built-in blue led.
// Attach a Graph widget to Analog pin 1
// Attach a Gauge widget to Analog pin 2

// No coding required for direct pin operations!

void setup()
{
    //Serial.begin(9600);
    delay(5000); // Allow board to settle
    Blynk.begin(auth);
}

// Attach a Button widget (mode: Push) to the Virtual pin 1 - and send sweet tweets!
BLYNK_WRITE(V1) {
    if (param.asInt() == 1) { // On button down...
        // Tweeting!
        // Note:
        //   We allow 1 tweet per minute for now.
        //   Twitter doesn't allow identical subsequent messages.
        Blynk.tweet("My Particle project is tweeting using @blynk_app and it’s awesome!\n @Particle #IoT #blynk");

        // Pushing notification to the app!
        // Note:
        //   We allow 1 notification per minute for now.
        Blynk.notify("You pressed the button and I know it ;)");
    }
}
```

```
// Attach a ZeRGBa widget (mode: Merge) to the Virtual pin 2 - and
control the built-in RGB led!
BLYNK_WRITE(V2) {
    int r = param[0].asInt();
    int g = param[1].asInt();
    int b = param[2].asInt();
    if (r > 0 || g > 0 || b > 0) {
        RGB.control(true);
        RGB.color(r, g, b);
    } else {
        RGB.control(false);
    }
}
void loop()
{
    Blynk.run();
    if (ModeBtnPressed()) {
        Blynk.notify("Mode button was pressed");
    }
}

// *** Utility functions

bool ModeBtnPressed() {
    if(millis() > 5000) {
        if(BUTTON_GetDebouncedTime(BUTTON1) >= 50) {
            BUTTON_ResetDebouncedState(BUTTON1);
            return 1;
        }
    }
    return 0;
}
```