Design of an Assistive Low-Cost 6 d.o.f. Robotic Arm with Gripper

Vasile Denis Manolescu [1][0000-0002-3250-8066], Emanuele Lindo Secco [2][0000-0002-3269-6749]

[1]Robotics Laboratory, School of Mathematics, Computer Science & Engineering, Liverpool Hope University, L16 9JD, UK
20203547@hope.ac.uk, seccoe@hope.ac.uk

Abstract. The robotics industry is rapidly evolving driven by better and cheaper computer chips and affordable 3D printing technologies. All these aspects are a catalyst that helps in building new concepts and prototypes at a lower cost, easier and faster than ever before. This paper presents the entire process of building an articulated robotic arm with 6 Degrees Of Freedom (DOF) and a gripper, all controlled from a designed Arduino command centre. The project will go through the 3D designing process and the selection of different actuators. Then an optimization of the hardware options for controlling the motors and the software to operate the robotic arm is presented. Finally, advantages and drawbacks of the proposed architecture are discussed.

Keywords: Robotic Arm, Assistive Robotics, Low-Cost Robotics.

1 INTRODUCTION

The meaning of what is known to be a robotic arm has evolved significantly in the past 20 years. The concept was mostly referring to a pre-programmed mechanical arm found in the assembling industries, capable of doing a repetitive task, fast and without interruptions for long periods. Mainly, those tasks were actions like welding, painting, palletizing, screwing, picking and placing or other tasks that involved not very complicated movements [1]. Today, the meaning of a robotic arm is becoming gradually more structured, descriptive and specialized. While the machine capabilities became a lot more complex with extreme speeds and accuracy and a high level of efficiency. As for the future, relatively new concepts like artificial intelligence, machine learning and deep learning, present a promising opportunity for the robotics industry [2] [3] [4]. All these data-driven software-side techniques are capable of enhancing the capabilities of the robots more efficiently and offer them full autonomy around humans. When starting to design a robotic arm, the number of joints represents one of the key points. Terminology wise, the “axes movement” or the “degrees of freedom (DOF)” are to robots as the joints are to the human body. The end effector represents the working or the operation tools attached to the upper end of the robot, that effectively performs the task, for example - a griper. To be able to move the end effectors in a 3-dimensional space, at least 3 joints are necessary, while to change the angle of the tools, a minimum of 6 joints are required [5]. By the type of joint and working volume that the end effector can reach in a given space (known as the working envelope), the robots can be classified into 6 main categories, as it follows:

1. *Polar/spherical coordinate robots* rotate in 2-directions and move linearly in 1 direction: a base rotation, a shoulder rotation and a linear motion of the end effector joint.
2. *Cylindrical-coordinate robots* move linear in 2-directions and rotates in 1 direction: vertical linear motion (stroke), horizontal linear motion (reach), rotation motion (swing).
3. *Cartesian or Gantry coordinate robots* move linear by sliding on each X, Y, Z axis.
4. Articulated robots offer high flexible movements and it is the most common type of industrial robot. They present at least 3 serial joint linkages and performs only rotary motions.
5. *SCARA - Selective Compliance Assembly Robot Arm (SCARA)* are specialized in lateral movements and operations on level surfaces. Performs 2 rotary horizontal motions and 1 linear vertical motion.
6. *Parallel Link robots* have the main advantage is the speed obtained by the parallel joint linkage.

Other more recent types of robots are:

A. *Anthropomorphic robot* are those devices resembling the human arm, with biomimetic movement (usually including fingers as well).

B. *Collaborative robot or Cobots* are fully articulated robots enhanced by sensors and software to detect and avoid unnecessary collisions with humans and other objects. One of their main characteristics is the capacity to memorize and reproduce a set of movements.

The articulated type of robotic arm is the subject of this paper. The paper is organised as it follows: a *Design Set-up* section will present the main characteristic of the robot and the details of the designing process. The *Hardware* section will present a depth view of all the components integrated into the project and will systematically analyse the joint structure to understand the requirements and the functionality of each connection. Then, a *Software* section will go through the software used to make the 3D Design of the robot, the Arduino IDE environment and the code developed to give control to the robotic arm. *Results & Discussion* will point out the major problems encountered and offer an insight into how they have been managed. All the future improvements that can be applied to the robotic arm will be covered by the *Future Work* section. While, the final section, *Conclusion* - will create an objective overview of the whole project and reflect on the end result.

2 DESIGN SET-UP

The overall design inspiration for the robotic arm was taken from Gluon, developed by Innfos Drive Technologies Co (Fig.1). One of the primary design objectives was to make the robot to be modular, to be able to easily combine the 3D parts in various ways allowing future changes in the joint structure. This feature also offers a great advantage for future upgrades. The picture in Fig. 1 shows the similarities between those two designs. The final version is 75 cm tall with 55 cm working envelop.

All types of actuators have been tested in the pre-building process of the robot. The final version works with 2 bipolar stepper motors, one rotating the base and the other one drives the second joint, while the rest of the joints are each driven by 5 servo motors, the last one being the gripper. The gripper design was not a priority and therefore the 3D model was downloaded from the open sources grabcad.com library, re-scaled and 3D printed. All credit goes to its creator [6]. Additionally, a pressure sensor has been integrated into the gripper fingers. The purpose of that is to collect enough data and make a squeeze profile for different objects. At the moment it is only being used to trigger the gripper servo to stop squeezing when a pressure level is reached.

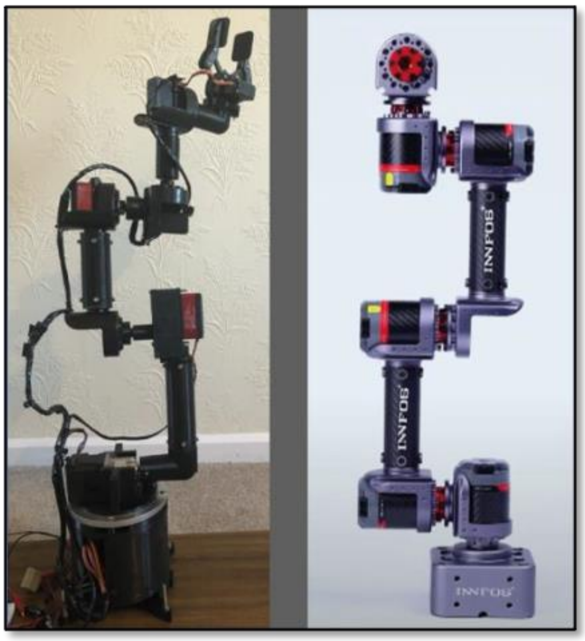


Fig. 1. The proposed design vs Gluon design (left and right panels, respectively).

The entire robotic arm is controlled by an Arduino Mega which offer the advantage of a larger number of connections: 54 digital input/output pins and 16 Analog inputs [7]. While in operation, the Arduino board is connected to a ~12V power supply and is linked to the followings:

1) one CNC shield expansion board holding two DRV8825 stepper motor drivers; This shield sits on top of the Arduino and is connected to a separate 12V power supply (Fig. 2).

2) a total of 6 dual-axis joystick modules which control the motors.

3) one push-button switch programmed to drive the motor in a pre-set position, defined in the code as home.

4) one PCA9685 servo driver module that is separately connected to 6V generated by a DC-DC Voltage Regulator from the 12V power supply.

More details about each of these components are presented in the next section of the paper.

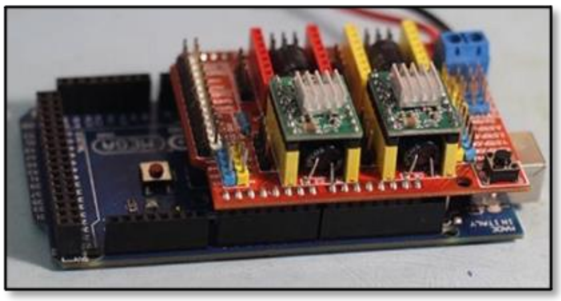


Fig. 2. The Arduino board with the CNC Shield and DVR8825 stepper drivers on the top.

3 HARDWARE

In this section we will look at the components used in building the robotics arm. The main goal is to analyse their purpose in the project and to justify their use. As a general reference, Fig. 3 illustrates the final degree of movement for each joint of the robotic arm, which are analysed in depth in the following paragraphs.

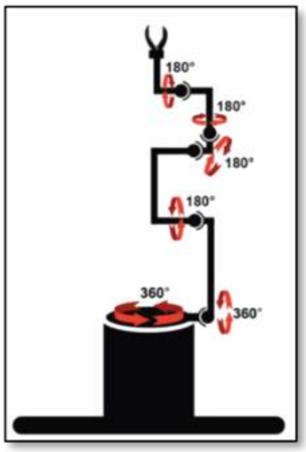


Fig. 3. Overview of the DOF angles and robot main movements.

3.1 Bipolar Stepper Motors

This section will analyse the stepper motors used to build the first 2 joints and will also talk about the 3D design around them.

Joint 1 – The Base. The base of the robotic arm is built using a *NEMA 17*, model *42HD4027* – which is a medium-size bipolar stepper motor. To operate, this actuator needs 3.3 V and a current rate of 1.5 A/phase. This makes it capable of producing 0.4 N/cm of torque. It also rotates with 1.8° per step, generating 200 steps per single revolution (360°). The 360° rotation of the base is a horizontal movement on the X-Z-axis. Counting the rest of the other joints and structure going up from the base, which weighs around 1.3 kg, the *42HD4027 stepper motor* generates enough torque to perform a smooth and stable movement.

The 3D model of the base is built in Autodesk® Fusion 360 and is designed to entirely encapsulate the stepper motor (Fig. 4 – left panel).

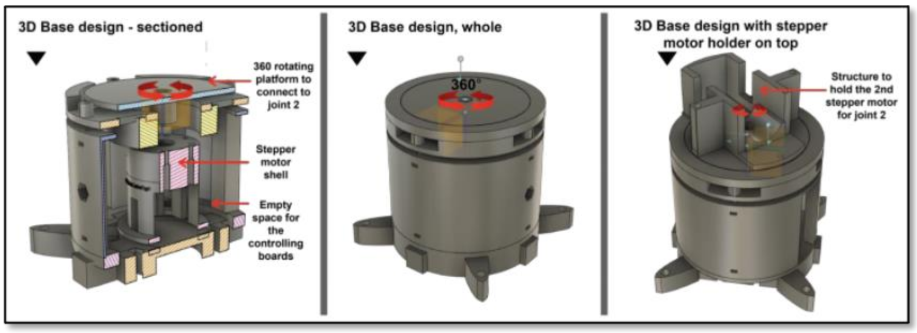

3D Base design - sectioned
360 rotating platform to connect to joint 2
Stepper motor shell
Empty space for the controlling boards
3D Base design, whole
360°
3D Base design with stepper motor holder on top
Structure to hold the 2nd stepper motor for joint 2

Fig. 4. Design of the robotic arm base (Fusion 360 ®)

Inside the base structure, the *42HD4027 stepper motor* holds itself in place by its own shell. This has been designed using one *ABEC-1* deep groove ball bearing, one *AXK* needle roller thrust bearing with vibration pads, one 3D printed shaft extension and one 3D printed outer housing. The bearings are installed on the motor shaft extension, while the housing closes down using the screws of the motor (Fig. 5 – left panel).

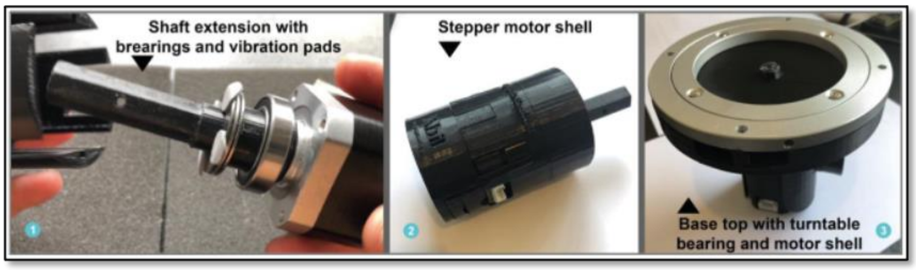


Fig. 5. Shell structure of the Stepper Motor number 1.

The new shaft extension goes inside the base, as shown in Fig. 4 – left panel, and connects to the rotating platform of the base (Fig. 4 – right panel). The rotating platform is built using a heavy-duty turntable bearing of 14 cm in diameter, which is attached to the holding structure of the second stepper motor of the second joint.

Joint 2. The second joint has a 360° rotation on the X-Y-axis, but when the arm is installed on a flat surface, the movement needs to be limited to 300° to avoid hitting the ground. Considering the weight of the arm structure and the up and down movement, the arm has a stress point in the stepper motor shaft of around 20 N/cm. Along the building process, the joint has been previously tested with a few other types of motors: high torque brushless geared DC motor, high torque servo motor and other stepper motors with less torque (see also Problems and Solution paragraph), but none were capable of handling the weight of the arm.

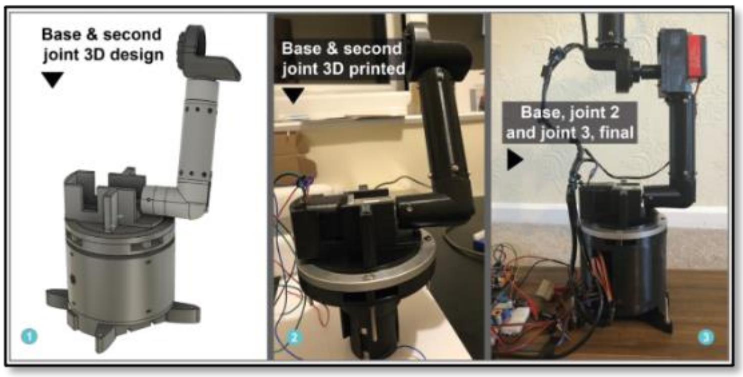


Fig. 6. 3D design vs the final 3D printed parts of the base, of joint 2 and of joint 3 (left, central and right panels, respectively)

The final version of the second joint uses a high torque *NEMA 17* bipolar stepper motor - model *17HS19-1684S-PG51*, having a planetary gearbox attached to it with a ratio of 51:1 (Fig. 6 – central panel). The *17HS19 stepper motor* operates at a current rate of 1.68A per phase with a rotation of 0.035° per step, generating approximately 10285 steps per revolution. This actuator is capable of producing around 400 N/cm of torque.

Comparing the requirements of the project with the capacity of the stepper motor, there is an unnecessary big difference in torque, but because accessible and reasonable price, the 17HS19 seemed to be the most viable solution available. The second joint connects with the third joint through a 3D printed 90° tube, which is directly installed on the joint-2 stepper motor shaft (Fig. 6). In the upper end of the tube, there is the joint-3 servo motor holder which will be discussed in the Servo section.

3.2 Stepper Motors Controller

This section will continue the talk about the parts that ensure the functionality and control of the 2 bipolar stepper motors presented above. These parts are the CNC shield and the DRV8825 stepper drivers.

CNC Shield V3. The CNC shields are specialized boards created around the Arduino microcontroller that offer integral solutions to build CNC machines. They offer the necessary power to drive multiple stepper motors and include functions like speed / direction control, stop, hold, micro-stepping, I2C and other more personalized functions [8].

Compared to other more dedicated options to control stepper motors, the CNC shield presents a few other advantages, like robust shape compatible with Arduino pinout, interface easy to work with, individual customizable controls, open-source design and reasonable price. According to the datasheet, the shield can operate with an input voltage of 12 to 36 V, in this project, it is connected to a 12 V DC power supply. On top, the shield can hold up to 4 stepper drivers like A4988, DRV8825, TCM 2100 etc. In this case, having to control just 2 stepper motors, only the X-slot and the Y-slot will each carry a DRV8825 driver. The *M0*, *M1* and *M2* pins are directly controlling the micro-stepping indexer of the driver attached to it (Fig. 7, left panel). More about that in the DRV8825 driver section below.

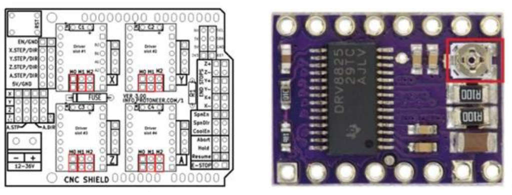


Fig. 7. the CNC shield V3 layout functions [9] and the current regulation potentiometer (red square) on the left and right panels, respectively.

DRV8825 Stepper Motor Driver. The *DRV8825* is an integrated motor driver that uses *N-channel MOSFETs* configured as 2 full *H-bridge* circuits. These are used to switch the flow of the current through the motor windings to control the speed and direction. A single driver can operate one stepper motor or two DC motors [10]. According to the datasheet the driver operates at 8.2 to 45V with a maximum current of 2.5A per phase [11]. In the project, the driver will get its voltage directly from the *CNC shield* which is connected to a 12V DC power supply. Compared to the *A4988* stepper driver, *DRV8825* supports micro-stepping down to 1/32 which makes it fully compatible with the CNC shield capabilities – that is the main reason why it was the final choice in this project.
After individual testing, both drivers end up being set to 4 micro-steps per step with *M0* and *M2* pins set as low and *M1* set as high. This configuration seems to move the stepper motors more smooth and natural than the other micro-stepping modes.

Setting up the stepper driver. Another very important feature in the driver configuration is the current regulation which needs to be set according to each stepper motor specifications. This is done from a small potentiometer located on top of the module (Fig. 7, right panel). The potentiometer needs to be manually set according to the manufacturer formula:

$$I_{FS}\,(A) = V_{REF}\,(V) \,/\, (\,A_V \times R_{SENSE}\,(\Omega)\,) \qquad (1)$$

where I_{FS} is the stepper motor maximum current rate per winding; I_{max} is the stepper motor total rated current; V_{REF} is the maximum voltage reference corresponding to the maximum current allowed to flow into the stepper motor; A_V is the current sense amplifier gain (according to the datasheet this parameter is a factor of x5); R_{SENSE} is the sense resistor value of the driver module, which in this case it is 0.2 Ω. After replacing the variables with the known values, given that $I_{max} = 2 \times I_{FS}$, it holds

$$I_{FS}\,(A) = \boldsymbol{V_{REF}}(\boldsymbol{V}) \,/\, (\,\boldsymbol{5} \times \boldsymbol{0.2}\,(\Omega)\,) \qquad (2)$$

It is important to notice that:

- ✓ V_{REF} is essentially determining the current limit per coil. If it is set too high it can overheat or burn both the motor and the driver. If it is set too low the motor loses steps or even stop moving.
- ✓ It is a good safety measure to adjust V_{REF} 10% lower.
- ✓ The R_{SENSE} variable can have different value, strictly depending on the manufacturer of the stepper driver.

Once the final formula is set, the next step is to determine the $\mathbf{V_{REF}}$ for each stepper motor and adjust the corresponding driver, namely:

Joint 1 - Stepper motor 42HD4027:

$$V_{REF} = (1.5 / 2) * 90\% = \mathbf{0.675\ V}$$

Joint 2 - Stepper motor 17HS19:

$$V_{REF} = (1.68 / 2) * 90\% = \mathbf{0.765\ V}$$

The final step is to physically adjust the current limit on the DRV8825 driver according to the calculus results.

3.3 Servo Motors

This section aims at discussing the selection of the 5 servo motors which were used for prototyping the joints 3 to 6 and the gripper.

Joint 3. The third joint is a relatively high-tension point in the robotic arm assembly. The minimum torque to move the upper structure is around 15 N/cm. The motion task is even harder when considering that there is an X-Y-axis movement, plus counting for a possible load carried by the gripper. The joint is capable of rotating 180° without restrictions.

To be able to properly operate this joint there was a need for a powerful servo motor. A great advantage for choosing a servo motor, in this case, was the balance between a lightweight actuator and a relatively high torque performance. The most significant disadvantages were the limited motion that can be achieved and the high acquisition cost.

In the end, the third joint was built with the *DS5260ssg servo motor,* which comes with a gearbox ratio of 279:1. According to the datasheet, it operates with a voltage of 6 to 8.4 V and it is capable of 60 kg/cm of torque.

The design for this joint is made to attach the servo motor to a 3D printed shell while extending the motor shaft from inside the shell to the next joint connector. This is done by using another 3D printed part – a pawn shaped shaft extension (Fig. 8). At the same time, between the servo shell and the shaft extension that comes out of it, there is an *ABEC-1* deep groove ball bearing that is used to minimise the bending and the pressure created by the upper joints upon the servo shaft (Fig. 8 – left panel).

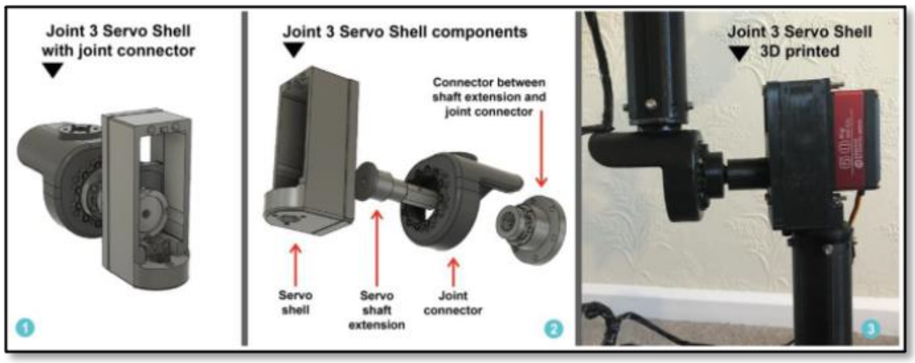

Joint 3 Servo Shell with joint connector
Joint 3 Servo Shell components
Connector between shaft extension and joint connector
Servo shell
Servo shaft extension
Joint connector
Joint 3 Servo Shell 3D printed
1
2
3

Fig. 8. design of the Joint 3 (left & central panel) and final assembly of the parts (right panel).

Joints 4, 5 & 6. Joint 4 is having a structure that weighs approximately 600 grams, that needs to lift off. Taking into consideration a possible gripper payload and the X-

Y-axis movement, the necessary torque to ensure a proper motion is approximately 10 N/cm. The joint can rotate 180° with no restrictions. The actuator used is a servo motor *DS3218MG* designed by DSServo with an operating voltage of 4.8 to 6.8 V and a gear-box ratio of 275:1.

Joint 5 and 6 are identically built and they use the same type of servo motor, the *Diymore MG996R*. These servo actuators operate with a voltage of 4.8 to 7.2 V and are capable of generating up to 15 kg/cm of torque.

Similar to joint 3, these three servo motors were chosen because they pack enough torque to operate the robotic arm and they are very light – weighting 55-60 grams.

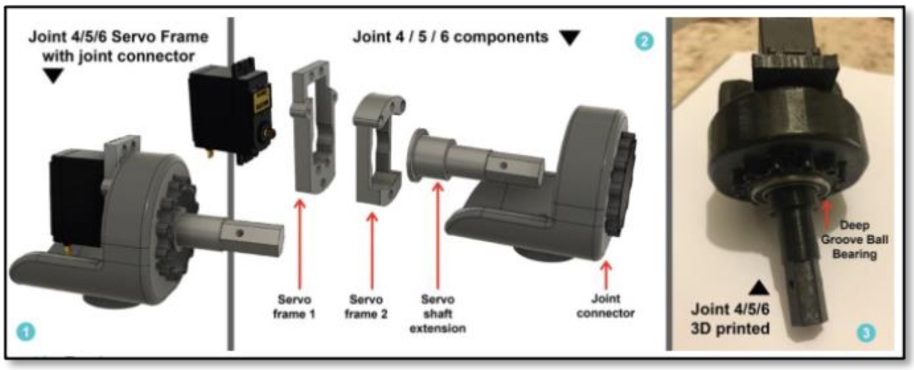


Fig. 9. top 3 joints – Inside and outside structure.

For all the three joints, the 3D design is similar and it is created to integrate the servos into the joint connectors as seen in Fig. 9. The motor connects to a 2 parts frame while the shaft is linked to a hexagonal pawn shaped shaft-extension. In the same way as in the third joint, between the shaft extension and the joint connector, there is a ball bearing to avoid bending and decrease the pressure (Fig. 9 – left panel).

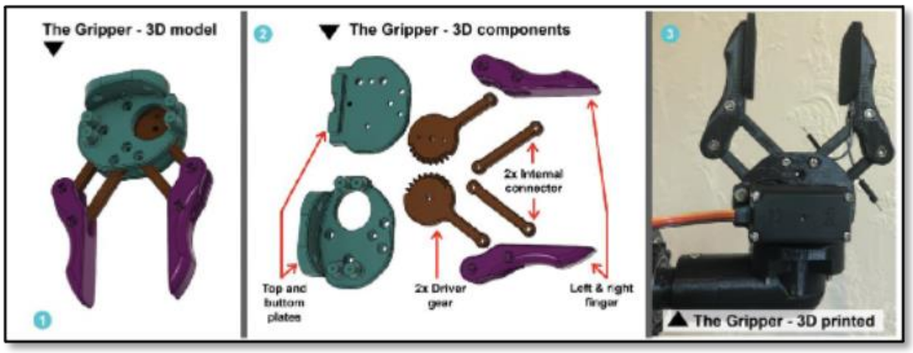


Fig. 10. The gripper design, from the 3D assembly (left and central panels) to the 3D printing manufacturing (right panel).

The end-effector or gripper. Because the construction of the griper was not a priority for the robotic arm, its design comes from a free online source [6]. The final 3D printed version is driven by a *Diymore MG996R* servo motor, the same actuator used in joint 5 and 6. The rotation of the motor is software limited to 120°, which is slightly less than the motion allowed by the mechanical parts.

Both gripper fingers are having anti-slipping pads and one of them is equipped with the *SF15-600 pressure sensor film* [12]. The sensor can sense and record the squeezing force applied to any grabbed object and it can be used to create gripping profiles for a different type of targets (Fig. 10).

3.4 Servo Motors Controller

This section of the paper goes through the components used to control the servo motors, which are the *PCA9685 servo driver* and the *DC-DC voltage regulator*.

PCA9685 Servo Motor Driver with I2C. The PCA9685 used in this project is a 16-channel I2C-bus protocol controller with a capacity of 12-bit output per channel and a fixed frequency (24-1526 Hz). Because of its features, the board can be used as *Pulse Width Modulation* (PWM) controller or to adjust the duty cycle, to individually drive up to 16 servo motors per chip [13].

Arduino communicates with the PCA board using I2C protocol through *Serial Data Line* (SDA) and *Serial Clock Line* (SCL) pin connection. This is how the direction and speed of the 5 servos are individually controlled (see also the Software section below).

To operate, the *PCA9685* gets 6 V from the 12V DC power supply using a dc-dc voltage regulator.

DC-DC Voltage Regulator. In order to simplify the development of the device, a single DC power supply was utilized, which is by default set to 12 V to be used by the CNC shield to control the stepper motors.

The *PCA9685 servo driver* is set to use the same power supply, but because it only needs 6 V, a *Buck DC-DC converter* is used to step down the voltage. The step-down switching regulator board is built based on the *XLSEMI XL4015 IC chip*, with 96% efficiency and capable of converting 8 to 36 V into 1.25 to 32 V [17]. The board can regulate the voltage using 2 push buttons and a 3-digits LED or from the *W503 trimmer potentiometer*.

3.5 Intuitive Control Panel (i.e. Dual-Axes Joystick Interface)

The centralized control of all the actuators is performed using 6 joysticks and 1 push-down button (Fig. 11). The 4-way directional 2-axis joystick is used as Analog user input and controller, and it is connected to the Arduino. The module consists of two perpendicular 10 kΩ potentiometers controlling the X and Y axis with the joystick movement. It includes springs to auto-return to the centre position. It also has an integrated joystick *push-down button* – which is used in this project as a home button with a pre-defined repositioning of each actuator.

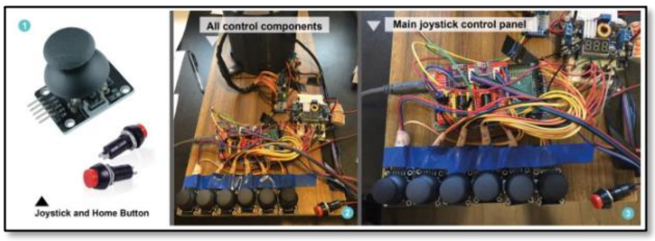


Fig. 11. The 2-axis, 4 directions Joystick interface with the push-buttons (left panel) and the main control panel details (central and right panels).

4 SOFTWARE

All the 3D designing and 3D printing have been possible using Autodesk Fusion 360. This software represents a complete package for product development offering a wide range of 3D designing tools (3D CAD), engineering tools (CAM) and manufacturing tools (CAE) [16]. The programming of the Arduino Mega 2560 board has been developed by using the Arduino IDE interface, which is a powerful, open-source C/C++ Programming Language environment developed for the Arduino products, but not limited to it.

4.1 The Arduino code

The servo motors controlling is developed using an external library specially developed for the PCA9685 driver board. The library cannot be downloaded and installed using the conventional way from Arduino IDE, the Library Manager [14]. It needs to be manually added to the default library folder of the IDE. Once installed, the library gets called. In practice, the most important aspect done by this library is to partition the internal registers on the PCA9685 chip to allow I2C communication for commands.

Global Variables and Pinouts. The global variables section in the code is split between those related to stepper motor control and those related to servo motor control. The first declared are the Arduino pinouts used to control the direction and the speed for both stepper motors.

The variables *x_dirPin / x_stepPin* are commanding the base direction and speed, while y*_dirPin* and *y_stepPin* are the same commands for the next joint. These first 2 actuators will be controlled by the same joystick, the base is related to the X-axis with up-down joystick movement, while joint-1 is related to the Y-axis with left-right joystick movement. The joystick data given by the potentiometer movement is read by the Arduino Analog pins *A0* and *A1* and it is stored by the *vrx_data* and *vry_data* variables. The variables *x_steps_per_rev* and *y_steps_per_rev* represent the number of steps per a 360° rotation for each stepper motor. These are used together with the joystick data to synchronize in mapping the motion.

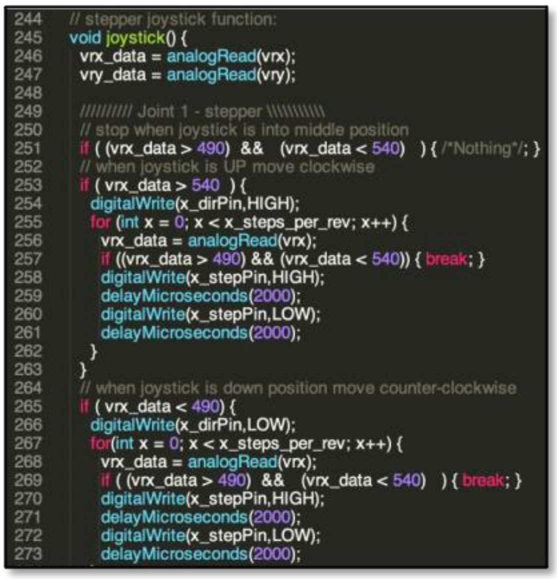
Concerning the servo motors global variables declaration, the first step is to define the *Arduino I2C master* address that will be used to communicate with the PCA board, as it follows: the code defines the Arduino Mega pinouts *A8* to *A11* for each joint, which are used to read the positioning data from each joystick. The data from those *Analog pins* are paired with the *Servo_Position* variables to map and send motion commands to the servo motors via the PCA board.

Every joystick press-down button defined by the *homeButton_* variable gets detected by the Arduino pins 22, 24, 26, 28 and 30. This command will automatically drive the related servo motor to its pre-defined initial position. The same thing happens with the *allHome* variable connected to pin 40, which is related to a universal push-down button that drives all servo motors, at the same time, to their initial position.

Setup function. The first four lines of code in the setup function is setting the *_stepPin* and *_dirPin* pins as output (sending data). While all home button pins are set to read the data and detect when the buttons are pressed.

Code line 58 is initializing the *PCA9685* by turning the Boolean variable *SERVO_MODE* to true. This variable is located in the library. The next line of code is activating all the library functions and features by turning the sleep mode off (false).

Loop function. One of the primary commands of the main loop is to run another function called *Joystick*. This dictates the two stepper motors rotation based on the joystick motion. The joystick potentiometer registers its movement with values between 0 to 1023, both on the X- and Y- axis. When it is not used, it auto-sets itself into a default middle position corresponding to a value of around 512, in this case between 490-540. Therefore, starting with the base stepper motor, initially, the function reads the X and Y data position of the joystick and writes it to the data variables assigned to it. The first if statement is associating the joystick default position values (490 to 540, including offsets), to the motor standby mode (code line 251, Fig. 12). The second if statement is associating the joystick up position (detected when the data position is greater than 540) to the right rotation of the actuator (lines 253-254, Fig. 12). At the same time, using a for loop, it keeps updating the joystick position and checks if it gets released to the default state, which would trigger the command of stopping the motor. Finally, if nothing is detected, the actuator will keep rotating to the right at the same speed (lines 258-261, Fig. 12).

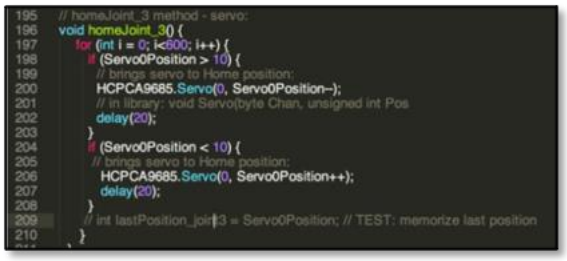


```
// stepper joystick function:
void joystick() {
  vrx_data = analogRead(vrx);
  vry_data = analogRead(vry);

  ////////// Joint 1 - stepper \\\\\\\\\\\\
  // stop when joystick is into middle position
  if ( (vrx_data > 490)  &&   (vrx_data < 540)   ) { /*Nothing*/; }
  // when joystick is UP move clockwise
  if ( vrx_data > 540  ) {
    digitalWrite(x_dirPin,HIGH);
    for (int x = 0; x < x_steps_per_rev; x++) {
      vrx_data = analogRead(vrx);
      if ((vrx_data > 490) && (vrx_data < 540)) { break; }
      digitalWrite(x_stepPin,HIGH);
      delayMicroseconds(2000);
      digitalWrite(x_stepPin,LOW);
      delayMicroseconds(2000);
    }
  }
  // when joystick is down position move counter-clockwise
  if ( vrx_data < 490) {
    digitalWrite(x_dirPin,LOW);
    for(int x = 0; x < x_steps_per_rev; x++) {
      vrx_data = analogRead(vrx);
      if ( (vrx_data > 490)  &&   (vrx_data < 540)   ) { break; }
      digitalWrite(x_stepPin,HIGH);
      delayMicroseconds(2000);
      digitalWrite(x_stepPin,LOW);
      delayMicroseconds(2000);
```

Fig. 12. The *Joystick* function.

The same logic is used to corelated the joystick down-positioning with the movement of the actuator to the left. The joystick function finishes by adapting the same algorithm design to the joystick left-right movement and the rotation of the second stepper motor.



```
// homeJoint_3 method - servo:
void homeJoint_3() {
    for (int i = 0; i<600; i++) {
      if (Servo0Position > 10) {
        // brings servo to Home position:
        HCPCA9685.Servo(0, Servo0Position--);
        // in library: void Servo(byte Chan, unsigned int Pos
        delay(20);
      }
      if (Servo0Position < 10) {
       // brings servo to Home position:
        HCPCA9685.Servo(0, Servo0Position++);
        delay(20);
      }
     // int lastPosition_joint3 = Servo0Position; // TEST: memorize last position
   }
```

Fig. 13. The *Return to Home* function.

Servo Motors controls. When it comes to servo motors control, the main loop checks first if any of the *homeButton* or *allHome* buttons are being pressed. The check is done by 5 if statements having the negation argument *!digitalRead()*, which detects any change of state. If a press of a button gets detected it will execute one of the *homeJoint* functions. This function is adapted for each joint, but it generally works the same way: a for loop is used with a given maximum value greater than the servo motor steps per a full revolution – this is determined by testing each servo. No matter in what position the actuator is, it will always loop back to the initial position of the shaft (Fig. 13).

Linking Servo Motor with Joystick. The last part of the main loop is a series of if statements grouped to define the movement of each servo motor. The logic of the algorithm works the same for each joint. The joystick potentiometer value is read, and the following 2 events may occur:

1. The value leans towards the upper direction (i.e., greater than 600) and the servo motor position is less than its maximum possible value (420), then the servo position is increased by 1 step to the right.
2. The value leans towards the down direction (i.e., less than 400) and the servo position is greater than its minimum (10), then the servo motor position is decreased by 1 step to the left.

In both cases, the loop is delayed by 10 ms and then it sends the new position value to the PCA board in order to be effectively executed by the actuator.

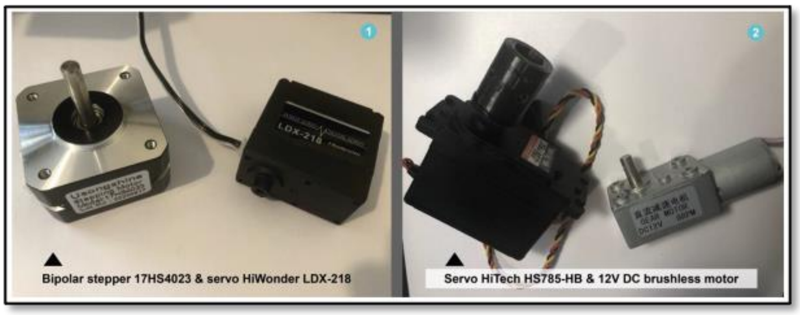

1
2
Bipolar stepper 17HS4023 & servo HiWonder LDX-218
Servo HiTech HS785-HB & 12V DC brushless motor

Fig. 14. Selection of the actuators (see details in the Results section below)

5 RESULTS & DISCUSSION

A set of preliminary trials was performed in order to check the reliability and robustness of the robot. During these preliminary tests, two main problems were noticed:

The first problem regards the fact that, initially, the robotic arm was supposed to be built using a small size stepper motor for all joints, but after testing the actuator on the 3D printed structure, the results proved that the motor was not capable of handling such a weight. The next option to solve this issue was to try using a 12V high torque, reversible DC motor, which on paper looked like a viable solution. But again, after testing on the structure of the arm, it also failed to lift the weight of the upper joints. Finally, the solution was to use the *17HS19 bipolar stepper motor,* which proved to be more than capable to handle the joint requirements. Additionally, to use this 600 gr actuator, the robotic arm base had to be redesigned (Fig. 14, above).

A second problem that was noticed during the tests was about the 3rd and 4th joint actuators: finding the proper motor for the next two joints proved to be challenging. After furtherly testing the 12 V DC motor and a couple of other servo motors, none of them was capable of lifting or moving the arm properly because of the weight. Then a solution was to try the highest torque servo available and gradually go for a lower torque servo as going up to the next joints. In the end, the strategy gave positive results, and all the servo actuators were performing well.

6 FUTURE WORK

One of the near-future plans is to use the pressure sensor that is already been included in the gripper structure and to develop an algorithm capable of using that data to better control the end effectors. Moreover, the PCA9685 cannot properly be used when dealing with various servo motors of different voltages. A new option for a new servo driver needs to be explored as well the development of a Human and Graphical User Interface [15] [18].

The modular design of the robotic arm makes it easy for other future upgrades. The major challenge that will help the project reach its true potential is to develop an encapsulated type of joint around a high torque actuator and incorporate into the same space features like integrated encoder, temperature sensors, high gearbox ratio for high precision motion, feedback with hall sensors, backlash correction and vibration dissipation housing. These are all planned to be tested and implemented in the near future.

Another important update could be to integrate cameras and proximity sensors into the robotic structure and feed the data into an AI type of algorithm that can help the robot become socially interactable, aware of obstacles and being able to receive and reproduce commands.

7 CONCLUSION

This project started with the goal of exploring a variety of actuators and other controlling hardware by building a 6 DOF robotic arm. 3D designing proved to be a fairly creative and satisfying phase, while 3D printing has been more demanding and the most time-consuming task. The electronic side of the project ended being the one that required a lot of research and documentation, done in parallel with intensive testing and problem-solving situations, but one of the most engaging and complex. Meanwhile, the software development has been a great task of logical thinking, testing, debugging and documenting. The final result of this work shows that building an operational robotic arm is becoming easier and more accessible through lower-cost components and better technologies. In other words, building a robotic arm is a complex and continuous process of challenges and improvements, but it is also a process of self-development.

ACKNOWLEDGEMENTS

This work was presented in coursework form in fulfilment of the requirements for the BEng in Robotics for the student Vasile Denis Manolescu from the Robotics Lab, School of Mathematics, Computer Science and Engineering, Liverpool Hope University.

REFERENCES

1. Builtin, Available at: https://builtin.com/roboticsgrabcad, last accessed 2021/09/02.
2. Magenes G, Ramat S, Secco EL: Life-like Sensorimotor Control: from Biological Systems to Artifacts, Current Psychology of Cognition, Vol. 21(4-5), pp. 565-596 (2002).
3. Magenes G, Secco EL: Teaching a robot with human natural movements. In: XI Winter Universiads, Conference on Biomechanics and Sports, 135-144, Italy (2003).
4. Secco EL, Visioli A, Magenes G: 'Minimum Jerk Motion Planning for a Prosthetic Finger', Journal of Robotic Systems, 21(7): 361-368 (2004).
5. Kawasaki Robotics. Available at: https://robotics.kawasaki.com/ja1/xyz/en/1803-01/, last accessed 2021/09/01.
6. Gripper design 3D model, Open source use. Available at: https://grabcad.com/library/dripper-4, last accessed 2021/09/02.
7. Arduino, Arduino Mega. Available at: https://www.arduino.cc/en/Main/arduinoBoardMega/, last accessed 2021/09/02.
8. All3dp, Arduino CNC shield guide. Available at: https://all3dp.com/2/arduino-cnc-shield-buyers-guide/, last accessed 2021/09/01.
9. Osoyoo, CNC shield V3 guide. Available at: https://osoyoo.com/2017/04/07/arduino-uno-cnc-shield-v3-0-a4988/, last accessed 2021/09/01.
10. Northwestern_University, Northwestern Robotics and Biosystems, 2009. Available at: http://hades.mech.northwestern.edu/index.php/Driving_a_high_current_DC_Motor_using_an_Hbridge, last accessed 2021/09/01.
11. Texas_Instruments, Texas Instruments - DRV8825 driver. Available at: https://www.ti.com/product/DRV8825, last accessed 2021/09/02.
12. Secco EL, Moutschen C: A Soft Anthropomorphic & Tactile Fingertip for Low-Cost Prosthetic & Robotic Applications, In: EAI Transactions on Pervasive Health and Technology, 4, 14, 2018.
13. nxp.com – Datasheet - PCA9685 Servo Driver. Available at: https://www.nxp.com/docs/en/data-sheet/PCA9685.pdf , last accessed 2021/09/02.
14. hobby.component.com, HCPCA9685 library. Available at: https://forum.hobbycomponents.com/viewtopic.php?f=58&t=2034, last accessed 2021/09/02.
15. Chu TS, Chua AY, Secco EL: A Wearable MYO Gesture Armband Controlling Sphero BB-8 Robot, In: HighTech and Innovation Journal, 1(4), 179-186, http://dx.doi.org/10.28991/HIJ-2020-01-04-05, last accessed 2021/09/02.
16. Autodesk Fusion 360, 2020. 3D CAD, CAM, CAE. Available at: https://academy.autodesk.com/course/120630/introduction-cad-and-cae-fusion-360, last accessed 2021/09/02.
17. LCSC. XL4015 Datasheet, 2014. Available at: https://datasheet.lcsc.com/szlcsc/1811081616_XLSEMI-XL4015E1_C51661.pdf, last accessed 2021/08/20
18. Secco EL, Scilio J: Development of a symbiotic GUI for Robotic and Prosthetic Hand, In: Intelligent Systems Conference (IntelliSys), Amsterdam, The Netherlands (2020).

APPENDIX

A video demonstration of the working robotic arm is available at the following link
https://www.youtube.com/watch?v=NlBRkktJQAw