3D Robotic Arm

Xavier Samos I Casañe

2 of July 2023

Abstract: This text introduces the concept of a robotic arm and its various applications in different industries. It describes the structure and components of a robotic arm, including joints, motors, sensors, and cameras. The focus is on a specific type of robotic arm called the pick and place arm, which is designed for picking up and placing objects. The advantages and features of pick and place robotic arms are discussed, such as high-precision movement, multiple degrees of freedom, gripper or end effector, vision systems, flexibility, and speed. The objective of the project is to design and develop a small-scale pick-and-place robotic arm that can perform basic movements with small objects. The project includes tasks such as 3D printing the components, developing a control system, conducting testing, and programming the robot.
Keywords: Robotic arm, Arduino uno, pick and place, PWM, 3D printing.

Resumen: El siguiente texto presenta una visión general completa de un brazo robótico de recoger y colocar y sus aplicaciones en diversas industrias. Se describe la estructura y los componentes de un brazo robótico, incluyendo articulaciones, motores, sensores y cámaras. El enfoque se centra en un tipo específico de brazo robótico llamado brazo de recoger y colocar, diseñado para recoger y colocar objetos. Se discuten las ventajas y características de los brazos robóticos de recoger y colocar, como el movimiento de alta precisión, los múltiples grados de libertad, el agarre o efector final, los sistemas de visión, la flexibilidad y la velocidad. El objetivo del proyecto es diseñar y desarrollar un brazo robótico de recoger y colocar a pequeña escala capaz de realizar movimientos básicos con objetos pequeños. El proyecto incluye tareas como la impresión 3D de los componentes, el desarrollo de un sistema de control, la realización de pruebas y la programación del robot.
Palabras clave: Brazo robótico, Arduino uno, recoger y colocar, PWM, impressión 3D.

1 INTRODUCTION

A robotic arm is a type of mechanical arm that is capable of performing a wide range of tasks, such as moving and manipulating objects, welding, painting, assembling products, and performing inspections and tests. It is controlled by a computer program or by a human operator through a control panel.

The robotic arm [1] typically consists of a series of joints and links that give it the ability to move in a wide range of directions, similar to the human arm. The joints are controlled by motors, which are responsible for providing the power and movement to the arm. Some robotic arms are also equipped with sensors and cameras that allow them to perceive their environment and interact with it. Robotic arms can be found in many different settings, including factories, warehouses, research labs, and even hospitals.

Moreover, a robotic arm can be classified based on their degree of freedom (DOF) which refers to the number of independent axes of movement that the robotic arm has. Typically, robotic arms have between 3 and 7 DOF. One of the key points of industrial robotic arms is their versatility for supporting multiple applications, such as [2]:

- Palletizing - automate the process of placing goods or products onto pallets.
- Material handling - create a safe and efficient warehouse by ensuring goods and materials are properly stored, easy to find or transported correctly.
- Welding - excellent candidate for advanced robotics with vision and AI augmentation for inline quality inspection.
- Inspection - Enhancing robots with vision and AI systems, businesses can benefit from real-time inspection, helping to reduce waste and downtime.
- Pick and place - typically used in modern manufacturing and logistics.

However, the type of robotic arm that will be developed is the pick and place one [3] since it has great flexibility which could be used for a diferent purpouse in the future. Therefore, a pick and place robotic arm (Fig. 1) is a type

• E-mail de contacte: 1527224@uab.cat
• Menció realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Dolores Rexachs (DACSO)
• Curs 2022/23

of robotic arm that is specifically designed for picking up and placing objects. These types of robotic arms are often used in manufacturing and assembly operations, as well as in warehouses and distribution centres for material handling tasks.

Fig. 1: Pick and Place Robot

Some common features of pick and place robotic arms include:

1. High-precision movement: these robotic arms are able to move with a high degree of accuracy and repeatability, allowing them to pick up and place objects with a high level of precision.

2. Multiple degrees of freedom: pick and place robotic arms typically have multiple joints and axes of movement, allowing them to reach and move in a wide range of directions.

3. Gripper or end effector: the robotic arm is equipped with a gripper or end effector, which is a device at the end of the arm that is used to grapes and hold objects.

4. Vision systems: many of them include vision systems that allow the robot to locate and identify objects, which enables it to pick them up and place them in the correct location.

5. Flexibility: can be used to handle a wide range of object types and sizes, and can be programmed to perform a wide variety of tasks.

6. Speed and efficiency: they can work at a faster pace than humans, and can operate continuously without getting tired, which increases production efficiency and reduces labour costs.

1.1 Objective

The objective of this project is to design and develop a small-scale pick-and-place robotic arm capable of performing basic movements with small objects. The robotic arm will simulate real-case scenarios, such as assembly line operations or sorting and packaging tasks. Thus, it will be controlled through a wired controller, allowing manual operation and precise control of its movements. The control system, integrated with the robot, will provide seamless interaction and real-time feedback to the user, ensuring great performance and ease of use.

Additionally, the project aims to accomplish the following tasks:

1. Design and 3D print the components required for the robotic arm.

2. Develop and implement an efficient control system for the robotic arm.

3. Conduct thorough testing of the entire system to ensure its functionality and performance.

4. Program the robot with the necessary algorithms and instructions for precise and accurate object manipulation.

1.2 Organization

In addition to the project objectives, a hybrid methodology, as described in [4], has been adopted for the execution of this project. This methodology combines linear, sequential tasks with parallel activities to ensure efficient progress. Certain tasks have been specifically designed to maintain the expected quality of the final system.

For detailed information on the methodology and project planning, including a Gantt (Fig. 24) chart outlining the steps towards the successful realization of the 3D robotic arm, please refer to the appendix (Appendix B & Appendix A).

2 Theoretical Concepts

This section explains some theoretical concepts which are necessary to understand the next sections.

I²C Interface - The I²C [5] bus is a standard bidirectional interface that uses a controller (known as the master), to communicate with slave devices. A slave may not transmit data unless it has been addressed by the master. Each device on the I²C bus has a specific device address to differentiate between other devices that are not on the same I²C bus.

The physical I²C interface consists of the serial clock (SCL) and serial data (SDA) lines. Both SDA and SCL lines must be connected to Vcc through a pull-up resistor.

One data bit is transferred during each block pulse of the SCL. One byte consists of eight bits on the SDA line. A byte may either be a device address, register address, or data written to or read from a slave. Data is transferred to the Most Significant Bit (MSB) first.

Each byte of data (including the address byte) is followed by one ACK bit from the receiver. The ACK bit allows the receiver to communicate to the transmitter that the byte was successfully received and another byte may be sent.

Thus, the I²C protocol will be used to communicate the Arduino Uno board (master) with the SSD1306 and PCA9685 (slaves).

Pulse-Width Modulation - PWM [6] - Digital signals are signals that can be represented by a 0 or 1 while analog signals have a greater range of possible values. Some microcontrollers have an onboard digital-to-analog converter (DAC) in order to control analog devices. However, the DAC is relatively expensive to produce in terms of cost and it also takes up a lot of silicon area. That is why PWM or Pulse-Width Modulation is used instead, it is a technique to control analog devices, using a digital signal (Fig. 2).

In order to control the current delivered to a device, the power supply can be rapidly switched from the “on” position to the “off” one in a particular pattern. Keeping it “on” for a duration more than the “off” duration will raise the average power and vice-versa.

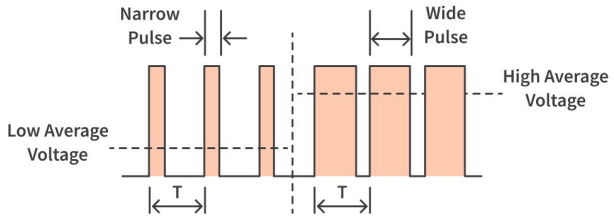

Fig. 2: Pulse and time period of a PWM signal

The switching on and off is the pulse. The duration for which the pulse is held at a high state is the pulse width. T represents the total time taken to complete a cycle. Modulation refers to the modification of the original signal to get our desired signal. Moreover, the duty cycle (1) describes the percentage of time a digital signal is “on” over an interval or period, which is represented in percentage (%).

$$D = \frac{T_{\text{on}}}{T} \cdot 100 \tag{1}$$

where
D = Duty cycle in Percentage
T_{on} = Duration of the signal being in the ”on” state
T = Total time taken to complete one cycle (Fig. 2)

$$V_{\text{avg}} = \frac{D}{100} \cdot V_{\text{max}} \tag{2}$$

where
V_{avg} = Average voltage of the signal
D = Duty Cycle in Percentage
V_{max} = Maximum voltage of the signal

Frequency is also a primary component that defines a PWM signal’s behaviour which will differ on each application.

3 Requirements

Functional requirements:

- Modular gripper design allowing for versatility in performing various tasks beyond pick and place operations.
- Wired controller for manual operation.
- Integration of a small display for visual feedback and user interaction.

Non-functional requirements:

- 6 degrees of freedom (DOF) to enable multi-axis movement.
- 3D printed robotic arm design.
- Servo motors with a range of motion of at least 180º (except for the gripper servo).
- Compact size with dimensions less than 40x40x40 cm (length, width, and height) in transport mode.

These requirements guide the development and ensure that the robotic arm meets the necessary functional capabilities and adheres to specific performance and operational criteria.

4 Design

4.1 Components

In order to have in mind a general view of the components, the **Wiring diagram** that will be used to connect the hardware is shown in Fig. 3.

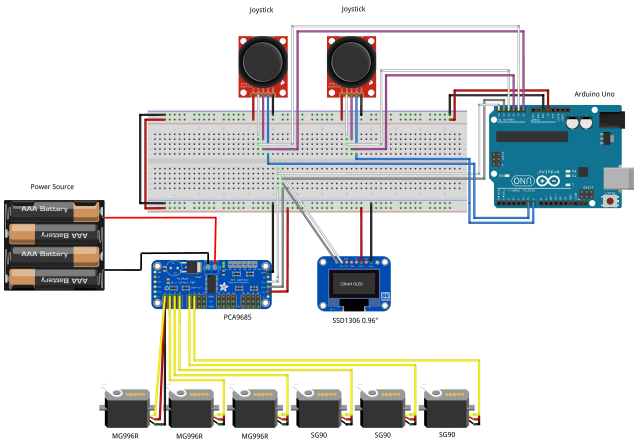

Fig. 3: Wiring diagram

Having said that, the details of the components are explained below:

MG996R (Fig. 4) - To effectively handle the leverage effect in the base, waist, and forearm of our 3D printed robotic arm, it is crucial to select a servo motor with good torque and robust metallic gears. This ensures optimal performance in these parts, making the MG996R Tower Pro servo motor [7] an ideal choice. With its impressive torque of 11 Kg/cm at 6V and a wide operating range of 0º to 180º (90º in each direction), this servo motor perfectly addresses the requirements. Additionally, its utilization of Pulse-Width Modulation (PWM) at a frequency of 50 Hz (20 ms) and a pulse width range of 500us to 2400us further complements its suitability for effectively operating the base, waist, and forearm.

Fig. 4: MG996R Servo Motor

SG90 (Fig. 5) - In order to ensure optimal performance in the arm, wrist, and gripper of our robotic arm, a servo motor that is compact, lightweight, and efficient is required. The SG90 Tower Pro servo motor [8] perfectly fits these

criteria. With its small size and lightweight design, it is an ideal choice for the end of the arm, where avoiding extra weight is crucial. The servo motor can rotate approximately 180º (90º in each direction) and operates with PWM at a frequency of 50 Hz (20 ms). This makes it well-suited for controlling the precise movements of the arm, wrist, and gripper. Moreover, as high power is not necessary for these components, the SG90 Tower Pro servo motor is a perfect match for our project requirements.

Fig. 5: SG90 Servo Motor

PCA9685 (Fig. 6) - To efficiently control the multiple servo motors in our robotic arm, we require a reliable and convenient solution. The PCA9685 [9] comes to our aid as an I²C-bus controlled 16-channel 12-bit PWM/Servo driver. This versatile component allows us to drive up to 16 servos using just two pins on the Arduino board, thanks to its I²C interface. The on-board PWM controller adds to its utility, enabling simultaneous control of all 16 channels. With six servo motors needed for our robot, the PCA9685 simplifies the control process by efficiently managing all the servos with minimal pin usage. Its compact design and advanced features make it an ideal choice for our project requirements.

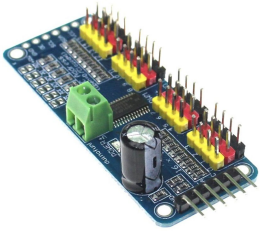

Fig. 6: PCA9685

SSD1306 (Fig. 7) - To enhance the user experience and provide valuable information about the robot's state, the SSD1306 [10] OLED display is incorporated into our design. With a resolution of 128x64 and a compact 0.96" screen size, this display offers clear and detailed visuals. The SSD1306 is controlled through I²C communication (it can also be modified for Serial Programming Interface), allowing easy integration into our system. It provides the capability to display text, graphs, and control each pixel individually. By utilizing this display,real-time information can be effectively conveyed to the user, ensuring they are aware of the robot's status and enhancing their interaction with the system.

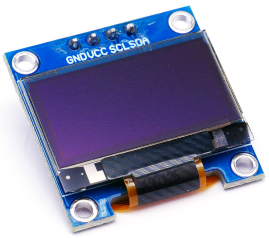

Fig. 7: SSD1306

Joystick (Fig. 8) - In order to enable intuitive control and interaction with the robotic arm, two joystick modules [11] are incorporated into the system. This module features two independent potentiometers—one for each axis (X and Y)—and a pushbutton. The potentiometers serve as input devices, providing control signals for the servo motors. These signals are utilized to precisely manipulate the arm's movements. Additionally, the joystick's pushbutton offers a convenient input for triggering specific actions or modes in the system.

Furthermore, the pushbuttons can also be utilized as input for the user interface provided by the SSD1306 OLED display [10]. This integration allows for an interactive user experience, where the pushbuttons can be translated into meaningful on-screen feedback or control parameters.

Fig. 8: Joystick

Arduino Uno (Fig. 9) - To meet the necessary requirements and efficiently control our 3D robotic arm, a microcontroller board with specific features is needed. The Arduino Uno [12] perfectly aligns with our needs, as it is based on the ATMega328P and offers 14 digital input/output pins (6 of which can be used as PWM outputs), 6 analog inputs (with 2 supporting I²C communications), a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Uno serves as the central programming unit, responsible for executing all the essential actions to move the robotic arm as desired. It facilitates communication with both the PCA9685 for servo control and the SSD1306 for OLED display interaction. By prioritizing the necessary requirements, such as digital and analog pins, communication interfaces, and essential functionalities, the Arduino Uno fulfills all the prerequisites for the project.

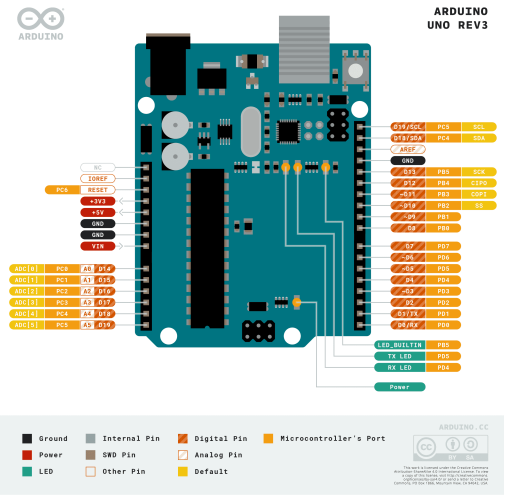

Fig. 9: Arduino Uno

4.2 Robotic arm

4.2.1 Design

There are many different computer-aided design (CAD) platforms on which the user can create the designs. Personally, I have used Onshape which is a FREE browser-based CAD platform that users can access on any web-connected device.

The platform is very intuitive and has many free online tutorials on how to use the different tools it offers. Nevertheless, everything ends up on creating sketches (Fig. 10) and then extruding (Fig. 11) them to form different shapes and parts (Fig. 12).

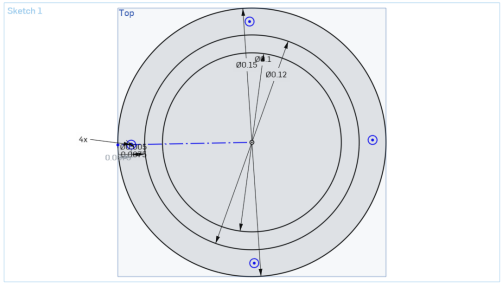

Fig. 10: Base sketch

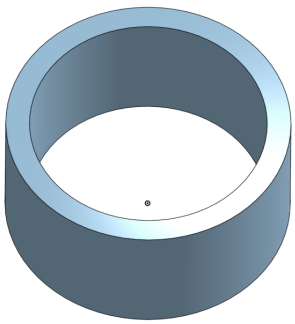

Fig. 11: Base extrude.

Once the different parts of the robot are designed, the correct procedure is to assemble (Fig. 13) them using the different assembly options provided by the CAD software.

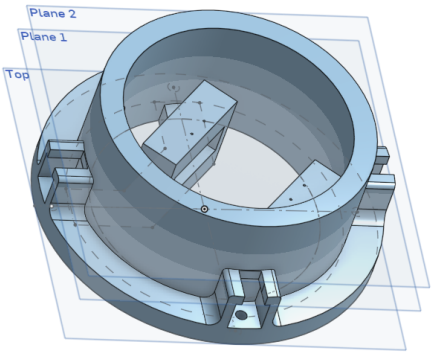

Fig. 12: Base.

These will make sure that everything fits together and the margins are correct before starting to print the parts. Moreover, designing the servo motors is a good practice because it will guarantee that everything works as expected.

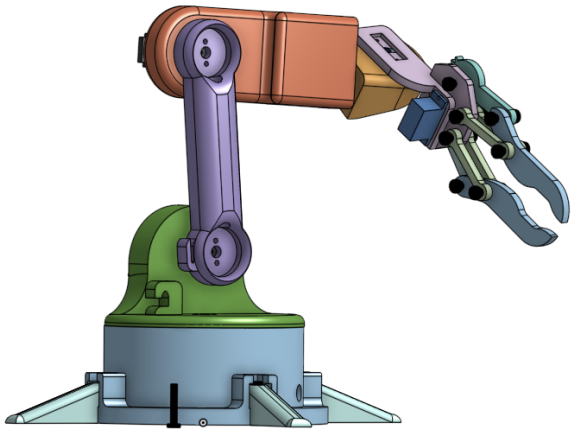

Fig. 13: Robot assembly.

As it can be seen on the assembly (Fig. 13), four legs have been added to provide better stability to the system. What's more, some structures have been added in the waist and forearm (green and violet pieces, respectively) to fit some rubber bands which will help the MG996R servo motor in charge of the forearm movement (it will reduce the stress on the motor since the gripper creates great leverage effect).

4.2.2 Assembly

The assembly is quite easy if the 3D model is accurate and the printing quality is acceptable. In this step it will be verified if the assembly simulation made with the CAD software was done correctly, otherwise, changes may need to be made to the 3D model.

Moreover, it is a good practice to print some parts and check that the fitting (everything assembles together correctly) is correct before printing all the other parts, the amount of redesigning and printing time will be reduced as future problems could be avoided.

It is to say that screws used in the assembly can damage the layers of the printed object (layers can separate from each other when applied force), specially sheet metal screws.

Once it is assembled, all the pins must be connected as shown on Section 4.1 Fig. 3, having said that, on Fig. 14 the full assembly of the robotic arm and the controller can be seen.

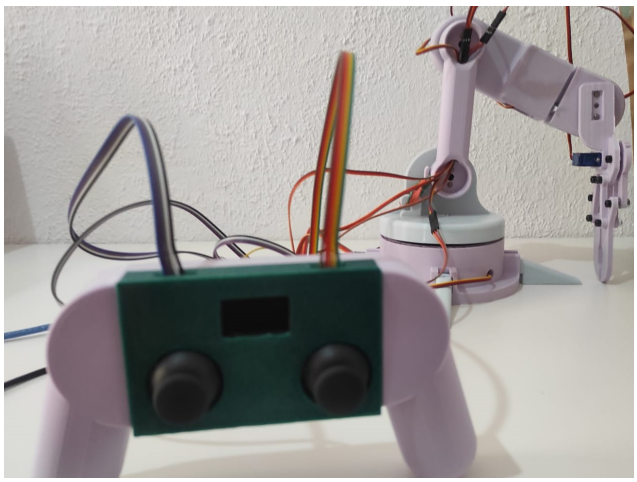

Fig. 14: Assembly of the robot and the controller.

4.3 Software

The software for the robot is written in C++ and is executed on an Arduino Uno microcontroller. The code can be found in Fig. C. However, understanding the code directly might be challenging, especially for inexperienced individuals. To facilitate comprehension, a software flow diagram is provided in Fig. 15, illustrating the sequence of operations. The software follows a simple and iterative procedure:

- Read joystick values: the program starts by reading the values from the joystick. These values indicate the desired movement or position of the robot.
- Process data and control servo motor: the read joystick values are then processed to determine how to control the chosen servo motor. Based on the desired movement or position, appropriate instructions are sent to the servo motor.
- Servo motor movement: the servo motor receives the instructions and moves accordingly. It adjusts its position or rotation based on the processed data from the joystick.
- Update display: finally, the display is updated with the current state of the robot. This includes information such as the position of the servo motor or any other relevant feedback.

This iterative procedure ensures that the software continuously reads joystick input, processes it, controls the servo motor accordingly, and updates the display to reflect the current state. By following this flow, the robot can be controlled effectively and provide visual feedback on its operations.

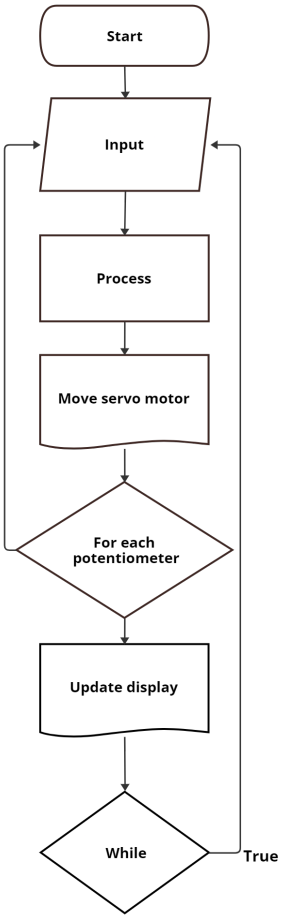

Fig. 15: Software flow diagram.

5 Implementation

On this section, the different hardware components will be tested individually in order to comply with the required specifications since not every hardware component may work as specified by the manufacturer.

Joystick module - As previously mentioned, the joystick module is formed by two potentiometers and a push-button. Therefore, if we connect the VRx and VRy pins of the module to two pins in the Arduino Uno board, we can proceed to read the analog value of the potentiometers. The read value is represented with a 10 bit value, which converted to a decimal number, ranging from 0 to 1023.

Consequently, the expected value when the joystick is in the central position would be an analog value of 512, however, this is not true in all cases. Joysticks have springs that return the centre stick to the middle when let go. Unfortunately, this does not mean that the joystick will return to the expected value. A software compensation is required, called deadstick. Therefore, a certain percentage value must be considered that if the joystick moves, it will still read the value as no movement. Generally, 5% [13] deadstick is enough.

Furthermore, another problem is presented when testing the push-button implemented in the joystick module, a debounce input method is required.

When a button is pressed/released, the state is not changed from LOW to HIGH or vice-versa, the mechanical and physical characteristics affect the state of the button, which toggles between LOW and HIGH several times. This

phenomenon is called chattering, as can be seen on Fig. 16 [14]. The chattering phenomenon makes a single press that may be read as multiple presses.

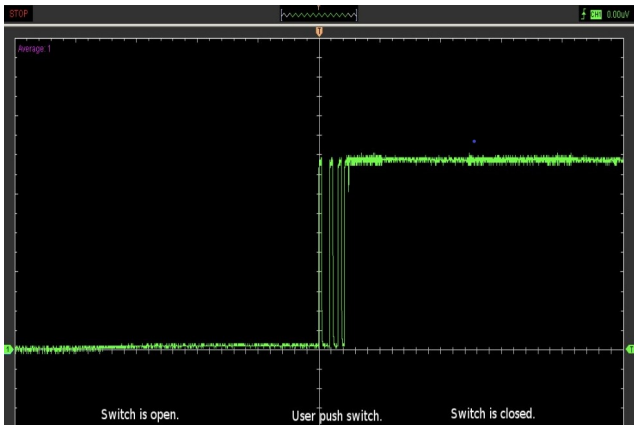

Fig. 16: Electrical Switch Output.

This can be easily solved by software (Fig. 17), the program just needs to keep track of the last read value of the button and add a bounce delay (the debounce delay can change if different hardware is used) to prevent the input fluctuations.

```
buttonState = digitalRead(BUTTON);
if(buttonState && !buttonLastState){
  flagLCD=!flagLCD;
  delay(300);
}
buttonLastState =  buttonState;
```

Fig. 17: Simple debounce code example.

Servo motors - The servo motors are supposed to work with PWM at 50 Hz. Nevertheless, this did not seem to work with the predefined voltage settings and, after testing with different parameters, the frequency used is 60 Hz with a maximum pulse width of 650 ms and minimum pulse width of 150 ms. These parameters will vary depending on the used voltage because they must comply with the average voltage delivered to the servo motors so they will function correctly in the range of 0º and 180º.

PCA9685 - The usage of the PCA9685 board is pretty straight forward, it has a specific library which smoothes the programming process. The only thing that is required is to connect (Fig. 18) the SDA and SCL pins of the board to the A4 and A5 pins of the Arduino Uno, respectively, as well as properly connecting the PCA9685 board to power.

The usage of this module simplifies the handling of the servo motors due to the fact it can generate the PWM signals by itself by just giving it the commands (Fig. 19). Additionally, there are more pins on the Arduino Uno board to be used, otherwise, we would require a bigger board with more input/output ports.

SSD1306 - The OLED display is also connected to the A4 and A5 ports of the Arduino Uno board, which implies that it is commanded by the Arduino Uno using the I²C interface. This display has a specific library to perform all the necessary operations.

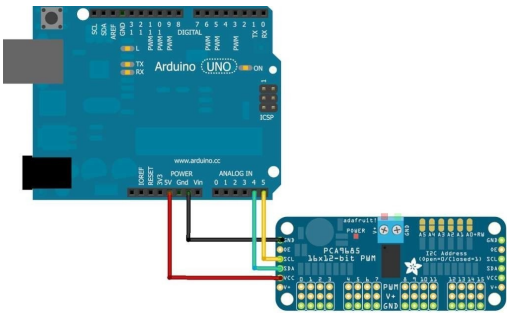

Fig. 18: PCA9685 test wire diagram.

```
#define minPulseWidth 150
#define maxPulseWidth 650
#define f 60

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

pwm.begin();
pwm.setPWMFreq(f);
// Pin number - on  - off
pwm.setPWM(0, 0, pulseWide);
```

Fig. 19: PCA9685 test program.

The only problem occurred with this hardware was that the manufacturer provided a wrong I²C address. The address had to be found manually through an exhaustive method (Fig. 20) (test each possible address sequentially until the slave replies to the master commands).

```
Serial.println("Scanning...");

nDevices = 0;
for(address = 1; address < 127; address++ )
{
  // The i2c_scanner uses the return value of
  // the Write.endTransmisstion to see if
  // a device did acknowledge to the address.
  WIRE.beginTransmission(address);
  error = WIRE.endTransmission();

  if (error == 0)
  {
    Serial.print("I2C device found at address 0x");
    if (address<16)
      Serial.print("0");
    Serial.print(address,HEX);
    Serial.println("  !");

    nDevices++;
  }
  else if (error==4)
  {
    Serial.print("Unknown error at address 0x");
    if (address<16)
      Serial.print("0");x
    Serial.println(address,HEX);
  }
}
```

Fig. 20: Exhaustive method to find I²C addresses.

6 Results

In this section, the capabilities of the 3D robotic arm are explored by performing various object manipulation tasks. By utilizing its precise movements and versatile grip, the robotic arm can interact with different small objects, including cubes, cilinders, and more (Fig. 21).

Through a combination of servo motor control, coordinated movements, and sensory feedback, the robotic arm demonstrates its ability to pick up, manipulate, and release these objects with precision. Each object presents unique challenges, requiring the arm to adapt its grip and movements accordingly.

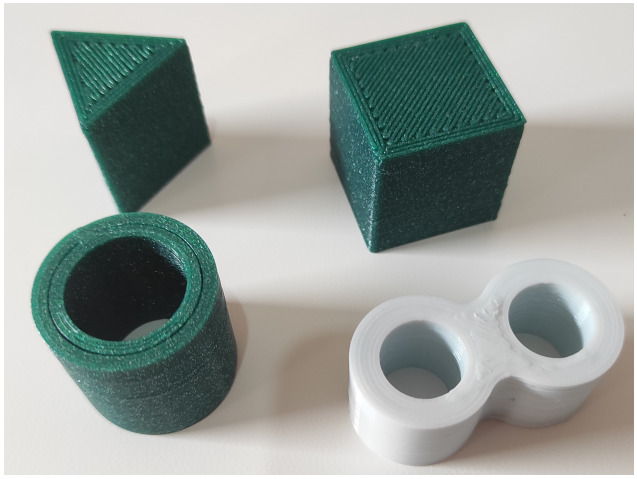

Fig. 21: Test objects.

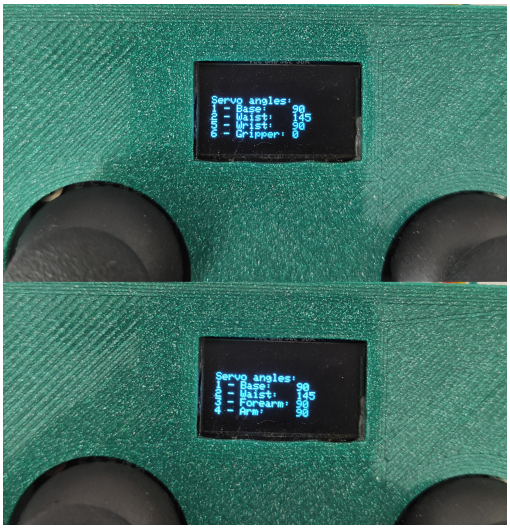

Fig. 22: Real-time feedback of servo motors degrees.

The robotic arm's control system, powered by the Arduino Uno and the PCA9685 servo driver, provides the necessary commands and positional data for executing the desired actions. This enables the arm to perform complex maneuvers and handle objects of varying shapes and sizes.

Furthermore, the integration of the joystick module and the SSD1306 OLED display allows for intuitive control and real-time feedback (Fig. 22), enhancing the operator's ability to monitor and adjust the arm's movements during object manipulation tasks.

By showcasing the robotic arm's performance with different objects, its versatility and potential applications in tasks such as pick-and-place operations, assembly processes, and more has been demonstrated. The successful manipulation of these objects highlights the precision and dexterity of the 3D robotic arm system.

The tests conducted on this Section 6 can be seen on the specifically created playlist [15].

7 Conclusion

The objectives of the project were successfully fulfilled as expected. The robotic arm demonstrated its capabilities through various object manipulation tasks, simulating real-world scenarios. However, it should be noted that the grip performance can be further improved, as the success of picking objects may depend on the skill of the operator and the characteristics of the objects themselves. Overall, the project serves as a solid foundation for future enhancements and advancements in the field of small-scale robotic arm development.

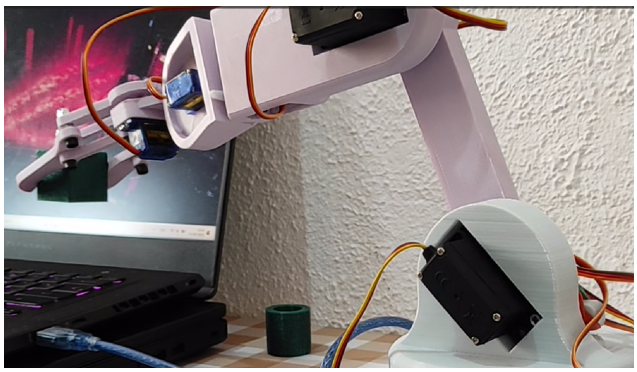

Fig. 23: Robot holding objects.

8 Future work

Some improvements that could be done to the project include exploring different materials for 3D printing. Instead of using Polylactic Acid (PLA), considering alternative materials with better cohesion and strength can enhance the printing quality and improve the fitting between parts. This can result in increased overall performance and the ability to handle larger and heavier objects.

In addition to the material upgrade, upgrading the printer itself can further improve the printing quality. Investing in a printer with advanced capabilities, such as higher precision and finer resolution, can enhance the fit and functionality of the printed parts. This upgrade can lead to improved printing results and overall better performance of the robot.

Another area of improvement is the servo motors used in the project. The current SG90 servo motors might have limitations in terms of torque and power, which can impact the gripping capability of the robot. Upgrading to larger servo motors with increased torque and strength can enhance the movement and gripping capabilities of the robot, allowing it to handle objects more effectively.

Furthermore, replacing the wire-controlled system with a wireless controller offers several advantages. By incorporating a WiFi module and leveraging the capabilities of the Arduino Uno, it is possible to establish wireless communication with computers or smartphones. Designing a dedicated application for wireless control can provide convenience and enable autonomous movements, while relieving computing tasks from the Arduino. This upgrade can enhance the flexibility, functionality, and ease of use of the robot.

By implementing these improvements, such as using different materials for 3D printing, upgrading the printer, servo motors, and implementing a wireless controller, the overall performance, precision, and control of the project can be significantly enhanced, resulting in a more capable and efficient robot.

REFERENCES

[1] RS Components. (2023, Jan, 20). A Complete Guide to Robotic Arms [Online]. Available: https://uk.rs-online.com/web/content/discovery/ideas-and-advice/robotic-arms-guide

[2] Intel. (2023, Jan. 19). Industrial Robotic Arms: Changing How Work Gets Done [Online]. Available: https://www.intel.com/content/www/us/en/robotics/robotic-arm.html

[3] EDS Robotics. (2020, Sep, 14). Pick and place y su impacto en la industria [Online]. Available: https://www.edsrobotics.com/blog/pick-and-place-que-es/

[4] Luis Paiva. BairesDevBlog. (2023, Feb, 06). The Best Project Management Methodology for Software Development [Online]. Available: https://www.bairesdev.com/blog/best-pm-methodology/

[5] Jonathan Valdez, Jared becker. (2015, Jun). Understanding the I²C bus [Online]. Available: https://www.ti.com/lit/an/slva704/slva704.pdf

[6] Jayesh Upadhyay, CircuitBread. (2022, Feb, 22). What is a PWM signal?[Online]. Available: https://www.circuitbread.com/ee-faq/what-is-a-pwm-signal

[7] Electronicoscaldas. (2023, Feb, 20). MG996R High torque Metal Gear Dual Ball Bearing Servo [Online]. Available: https://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf

[8] Imperial College London. (2023, Feb, 20). Servo Motor SG90 Datasheet [Online]. Available: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

[9] Bill Earl. (2023, Feb, 20). Adafruit PCA9685 16-Channel Servo Driver [Online]. Available: https://learn.adafruit.com/16-channel-pwm-servo-driver?view=all

[10] Solomon Systech. (2008, Apr). SSD1306 [Online]. Available: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

[11] Components 101. (2023, Feb, 20). Joystick Module [Online]. Available: https://components101.com/modules/joystick-module

[12] Arduino. (2023, Feb, 20). Arduino Uno Rev3 [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3

[13] Schindler Electronics. (2023, Mar, 08). Joysticks and how to use them [Online]. Available: https://www.schindlerelectronics.com/joysticks

[14] Jens Christoffersen, All About Circuits. (2015, Sept, 03). Switch Bounce and How to Deal with [Online]. Available: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

[15] Xavier Samos Casañe, (2023, May, 21). 3D Robotic Arm, playlist, Youtube, 2023. [Online]. Available: https://www.youtube.com/playlist?list=PL6xftA-OOhNYFMYyMoiAi2DdkdOemMghb

APPENDIX

A PROJECT PLANIFICATION

The project planification is outlined in Fig. 24, the steps that will guide us towards the successful realization of our 3D robotic arm are:

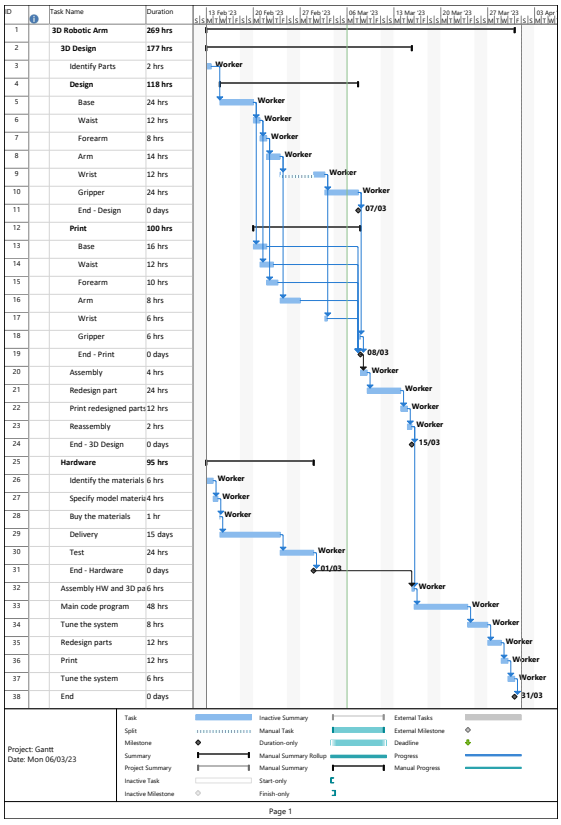

Fig. 24: Gantt diagram.

1. Identify the required materials. Research the type of material to be used (without specific brands and models). For example, a board, number of sensors and its type (analog, digital...).

2. Specify the material model (based on market disponibility and material quality). For example, the programmable board X can be used since it fulfils the requirements to connect all the required sensors, the sensor Y must be used to have a linear response in the system.

3. Buy the components. Components can be bought online from different websites, however, prices, quality and delivery time will change drastically depending on the manufacturer and shipping place.

4. Design the 3D robotic arm. The robotic arm has the next main parts:

   - Base.
   - Waist.
   - Forearm.

- Arm.
- Wrist.
- Gripper.

5. Print the designed pieces. The pieces can be re-designed if necessary; the usage of different pieces gives modularity to the system.
6. Individual tests of the bought elements. Tests will be handled individually to ensure its expected usage.
7. Assembly of the robotic arm.
8. Program the main code of the system.
9. Test the system. In this part, redesign of the 3D parts may be required, hence, they must be printed again and assembled.
10. Tune the system. Minor improvements will be done to increase the system performance.
11. System validation. The capabilities of the 3D robotic arm are further demonstrated and evaluated through object manipulation tasks (Section 6).

B METHODOLOGY

The main methodology is based on the hybrid [4] methodology since some tasks and phases are completed in a linear, sequential manner while others are done in parallel. It must be said that there are some tasks specifically designed to ensure that the quality of the final system is as expected.

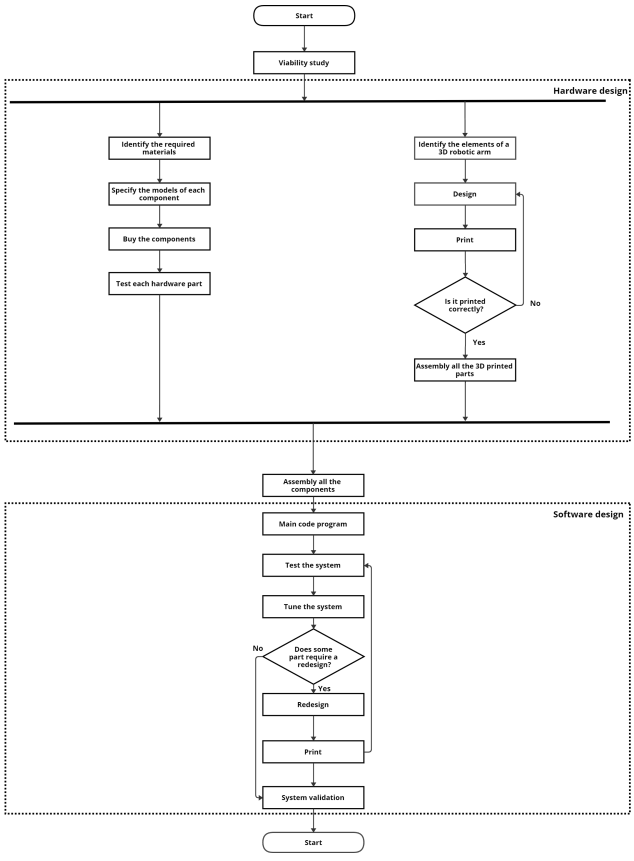

Fig. 25: Work methodology.

C ARDUINO CODE

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_GFX.h>\
#include <Adafruit_PWMServoDriver.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
// Declaration for a SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
#define OLED_RESET     -1   // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C   // Address 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
#define LOGO_WIDTH    88
#define LOGO_HEIGHT   35
#define minPulseWidth 150
#define maxPulseWidth 650
#define f 60

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

/*Digital input pin, used to toggle servo motors*/
#define BUTTON 6
/*
  Joystick inputs  - Servo outputs on PCA9685 - Motor Degree Value
  Base        A0 - 0 - 0
  Shoulder    A1 - 1 - 145
  Elbow       A2 - 2 - 90
  WristYaw    A3 - 3 - 90
  WristPitch  A6 - 4 - 90
  Gripper     A7 - 5 - 0
*/
int joyIn[] = { A0, A1, A2, A3};
int mOutput[] = { 0, 1, 2, 3, 4, 5};
int sDegrees[] = {90, 145, 90, 90, 90, 0};
bool buttonLastState = false;
bool buttonState = false;
bool flag = false;
bool flagLCD = true;

void setup() {
  pwm.begin();
  pwm.setPWMFreq(f);
  Serial.begin(9600);
  pinMode(BUTTON, INPUT_PULLUP);

  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  drawbitmap(); // Draw a small bitmap image
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("Moving servo motors to default values..."));
  delay(1000);
  //Set servo motors to initial position
  for(int i = 0; i<5; i++)
    moveServoDeg(sDegrees[i], i);
  textLCD();
}

// Convert joystick position to servo motor angle
int getAngle(int i){
  int joyVal, degrees;
  joyVal = analogRead(joyIn[i]);
  if(!flag)
    i+=2;
  if(joyVal > 530){
    if(i != 5)
      degrees = map(joyVal, 512, 1023, 0, 3);
    else
      degrees = map(joyVal, 512, 1023, 0, 30);
  }
  else if(490 > joyVal){
    if(i !=5)
      degrees = map(joyVal, 0, 512, -3, 0);
    else
      degrees = map(joyVal,0, 512, -30, 0);
  }
  else
    degrees = 0;
  return degrees;
}

// Move servo to specific degree
void moveServoDeg(int degrees, int i){
  int pulseWide, aux;
  if(i==5){ // Gripper servo motor
    aux = map(degrees, 0, 180, 0, 35);
    pulseWide = map(aux, 0, 180, minPulseWidth, maxPulseWidth);
  }
  else
    pulseWide = map(degrees, 0, 180, minPulseWidth, maxPulseWidth);
  pwm.setPWM(mOutput[i], 0, pulseWide);
}
void loop(){

  for(int i=0; i< 2; i++){
    sDegrees[i] += getAngle(i);
    if(sDegrees[i]>180)
      sDegrees[i]=180;
    else if(sDegrees[i]<0)
      sDegrees[i]=0;
    moveServoDeg(sDegrees[i], i);
  }
  // Servo selector
  if(flag)
    for(int i=2; i<4; i++){
      sDegrees[i] += getAngle(i);
      if(sDegrees[i]>180)
        sDegrees[i]=180;
      else if(sDegrees[i]<0)
        sDegrees[i]=0;
      moveServoDeg(sDegrees[i], i);
    }
  else
    for(int i=4; i<6; i++){
      sDegrees[i] += getAngle(i%4+2);
      if(sDegrees[i]>180)
        sDegrees[i]=180;
      else if(sDegrees[i]<0)
        sDegrees[i]=0;
      moveServoDeg(sDegrees[i], i);
    }

  // Button servo selector + button debounce
  buttonState = digitalRead(BUTTON);
  if(buttonState && !buttonLastState){
    flag=!flag;
    flagLCD=!flagLCD;
    Serial.print("Button: ");
    Serial.println(flag);
    delay(300);
  }
  buttonLastState =  buttonState;
  textLCD();
}
// UAB Logo
static const unsigned char PROGMEM logo_bmp[] = { ...
};
void textLCD(void){
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 20);
  if(flagLCD){
    display.println("Servo degrees: ");
    display.println("1 - Base:    " + String(sDegrees[0]));
    display.println("2 - Waist:   " + String(sDegrees[1]));
    display.println("5 - Wrist:   " + String(sDegrees[4]));
    display.println("6 - Gripper: " + String(sDegrees[5]));
  }else{
    display.println("Servo degrees: ");
    display.println("1 - Base:    " + String(sDegrees[0]));
    display.println("2 - Waist:   " + String(sDegrees[1]));
    display.println("3 - Forearm: " + String(sDegrees[2]));
    display.println("4 - Arm:     " + String(sDegrees[3]));
  }
  display.display();      // Show initial text
}
void drawbitmap(void){
  display.clearDisplay();
  display.drawBitmap(
    (display.width()  - LOGO_WIDTH ) / 2,
    (display.height() - LOGO_HEIGHT) / 2,
    logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
  display.display();
  delay(1000);
}
```