DESIGN AND IMPLEMENTATION OF AN AUTOMATIC 4 DOF ROBOT ARM PICKER

Munyakabera jean Claude | 2019040146006 | esp32 control software
Ndizihiwe jean Paul |2019040146003 | Arduino servo control
Gutema Misgana | 2019040146010 | MATLAB simulation
Abigaba peter | 2019040146005 | CAD & 3D printing

JUNE 10, 2022
UESTC

INTRODUCTION

We wanted to build an arm that could be able to lift at least 1KG and could pick objects on its own.

Using a 4 DOF robot ensure that you can move the wrist independent of the rest of the arm and reach above or below where the arm is pointing.

We selected this project because it is challenging and we would learn multiple things at once, micro controllers, control theory and mechanical design.

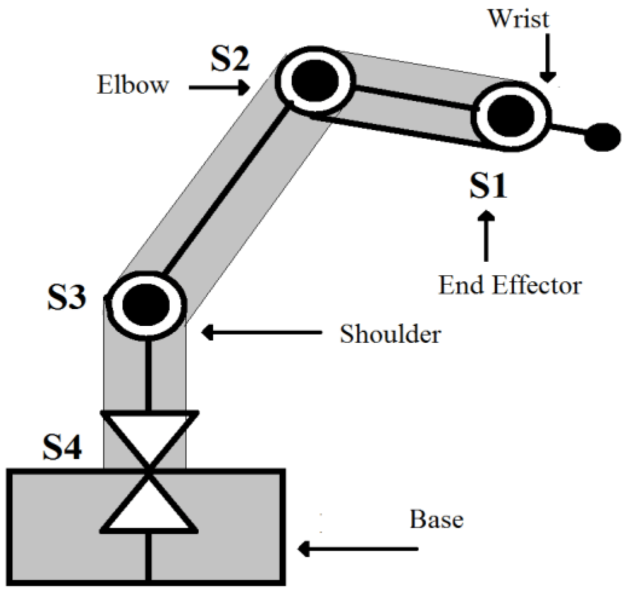
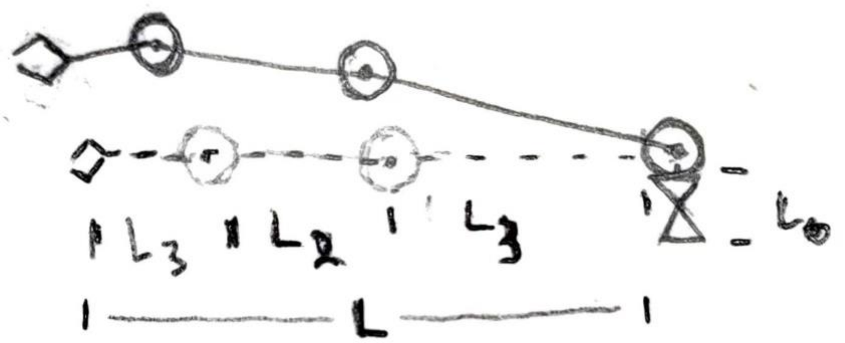


Figure 1: robot arm concept

We use an ultrasonic range sensor to identify where objects are and calculate which angles to send to the servos to reach that object, we planned to use inverse kinematics but after many trials we implemented a custom control logic and were able to reliably pick objects.

DESIGN

Torque and servo selection



$$L_{mass} = m_1 + m_2 + m_3 + P_{mass}$$

$$P_{mass} = \text{Payload mass}$$

$$P_{mass} \geq 1\,\text{Kg}$$

$$T = (L_1 + L_2 + L_3) \times (L_{mass} \times g)$$

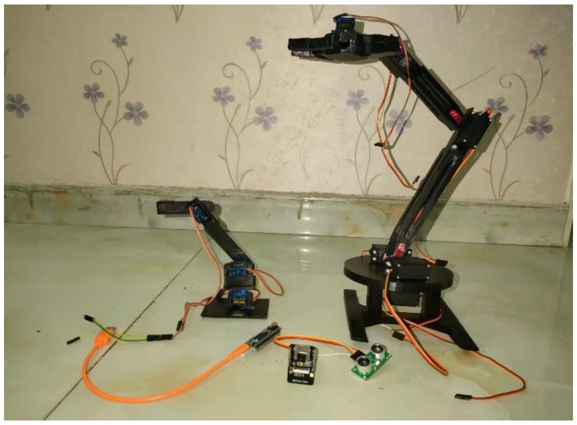
Figure 2: arm figure and torque equation

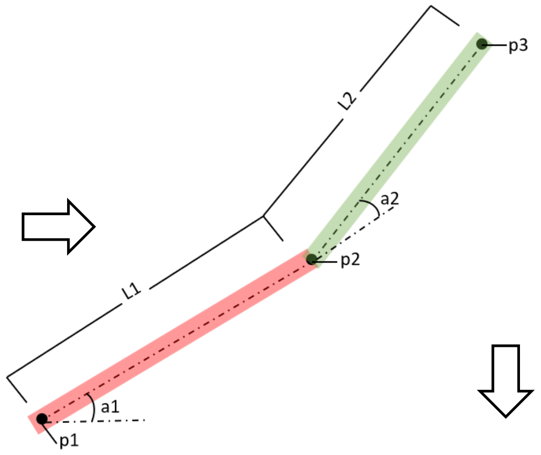
Since the joint base will be the one supporting all the arm and object it picked, we can simplify things by iterating over different arm lengths and computing the torque required to lift 1 kg, and omitting those iterations here, we reached at 30 cm and this would require a torque of **0.3*1.3*9.8=3.82Nm = 38.2kgcm (***mass of the links is 300g***)** and so we need a servo with a torque > than 38.2kgcm we decided to couple 2 20kgcm servo to make 40kgcm since there were high quality 20kgcm servos on sale/discount, the rest of the joints have shorter lengths to be able to handle less than 20kgcm torque and the base uses another 13kgcm servo but it does not need to because it experiences almost no torque, it was just convenient since the control structure will be unified.

Inverse and Forward Kinematics

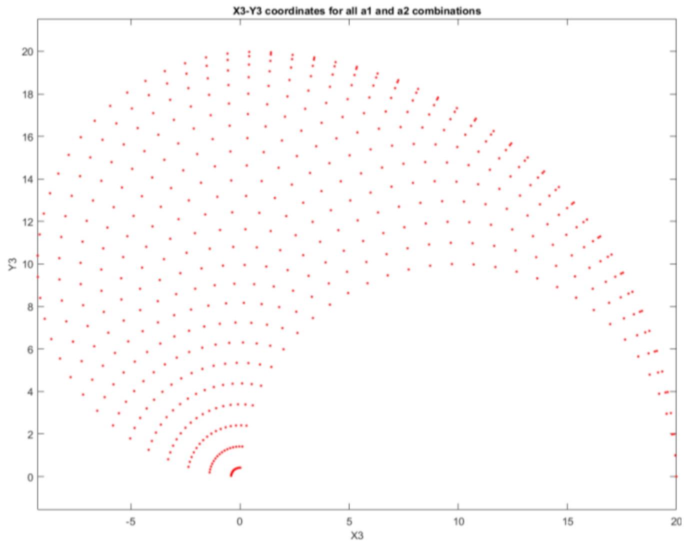
Inverse kinematics calculates what angles are needed to move in a certain x, y, z position, and forward kinematics takes angles and determine at what position the arm will reach.

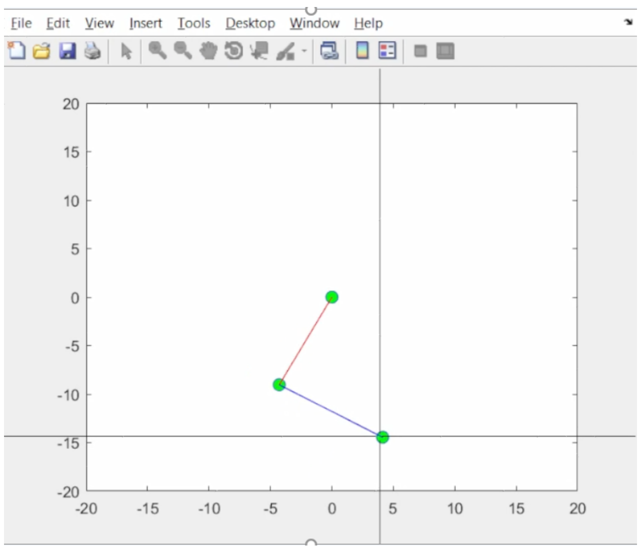
We started by testing a simple 2DOF arm then **made a Math model of it** then





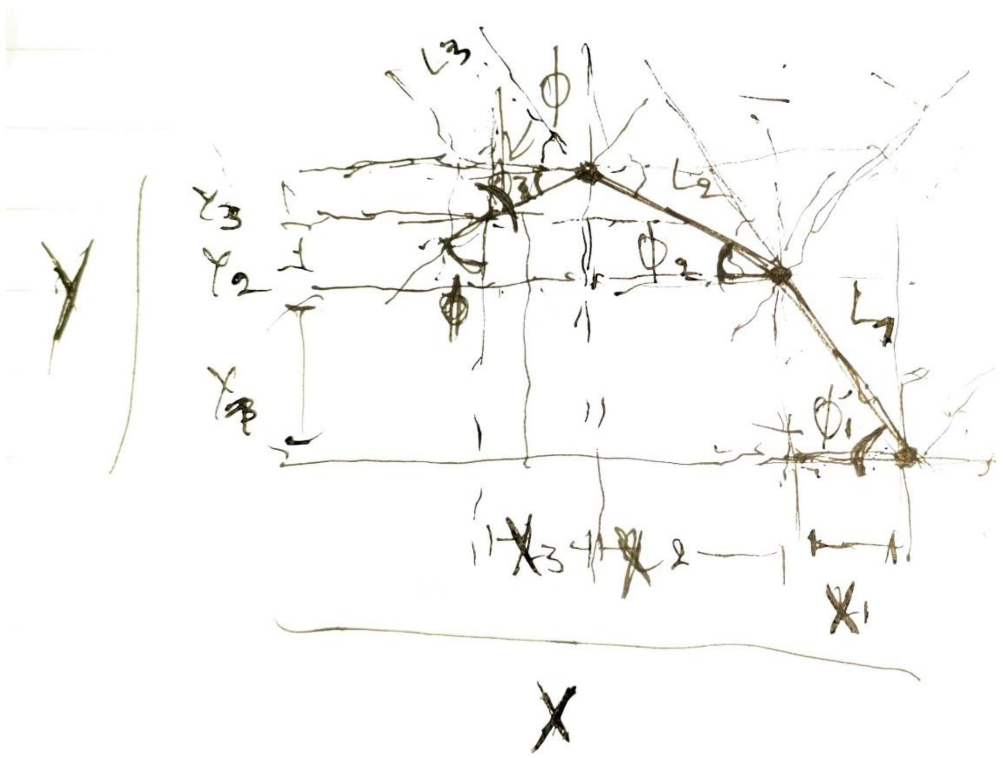
We got all possible position it can reach by **testing the simulator**





Forward Kinematics (2DOF) Inverse Kinematics (2DOF)

Forward Kinematics (2DOF)	Inverse Kinematics (2DOF)
$x3 = (L2\cos a1 + a2) + L1\cos a1$	$a_2 = \pm\cos^{-1}\dfrac{x_3^2 + y_3^2 - L_1^2 - L_2^2}{2L_1L_2}$
$y3 = (L2\sin a1 + a2) + L1\sin a1$	$a_1 = atan2(y_3, x_3) - \ atan2(L_2\sin a_2\,, L_2 + L_1\cos a_2)$



After testing we moved to 3 DOF inverse and forward kinematics formulation.

Inverse kinematics was found out to be and **Forward kinematics is**

We did 3 DOF because the base will rotate searching and the arm will just need to extend.

After many tests, inverse kinematics failed on both test and final robot arm, our own implementation worked in the simulator but failed on the robots, we tried various other libraries and they all failed (eg: fabrik2d), the issue being the servo angle configuration.

$$X_2 = X - L_3\cos(\phi)$$
$$Y_2 = Y - L_3\sin(\phi)$$

$$\cos(\varphi_2) = \frac{(X_2)^2 + (Y_2)^2 - (L_1)^2 - (L_2)^2}{2L_1L_2}$$

$$\cos(\varphi_1) = \frac{(L_1 + L_2\cos(\varphi_2))X_2 + L_2\sin(\varphi_2)Y_2}{(X_2)^2 + (Y_2)^2}$$

$$\varphi_3 = \varphi - \varphi_2 - \varphi_1$$

OF arm figu

$$X = L_1\cos(\varphi_1) + L_2\cos(\varphi_1 + \varphi_2) + L_3\cos(\varphi_1 + \varphi_2 + \varphi_3)$$
$$Y = L_1\sin(\varphi_1) + L_2\sin(\varphi_1 + \varphi_2) + L_3\sin(\varphi_1 + \varphi_2 + \varphi_3)$$
$$\varphi = \varphi_3 + \varphi_2 + \varphi_1$$

The new algorithm was made to be simple and straight forward it goes like:

given starting **angles A1,A2,A3** and final **angles A1’,A2’,A3’** with **step T**, take a reference joint then calculate the next step of the other joints relative to it, and iterate by T till you reach the final angles.

The result of this, is the arm move in a straight line, the gripper always facing forward till it reached the object.

IMPLEMENTATION

Cad, 3d Printing and Assembly

The whole robot was 3d printed, we used solid works and printed on an Anyubic mega s.

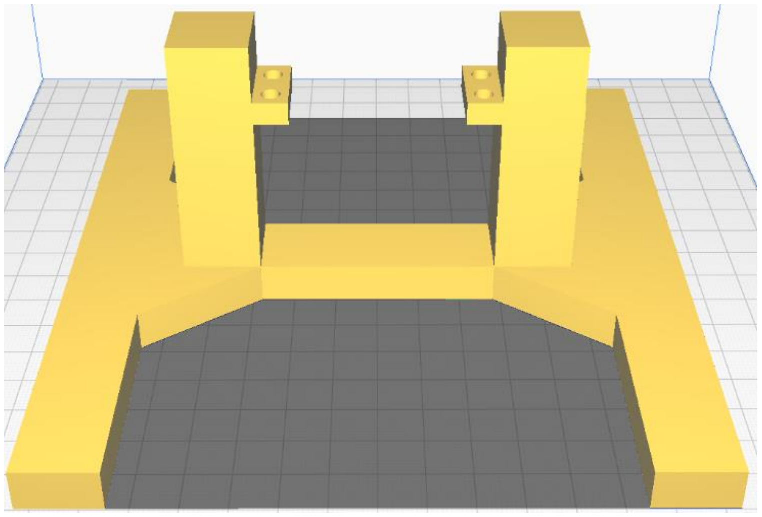


Figure 4: base

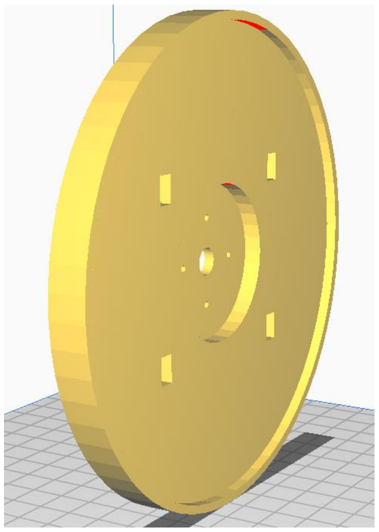


Figure 5: upper base

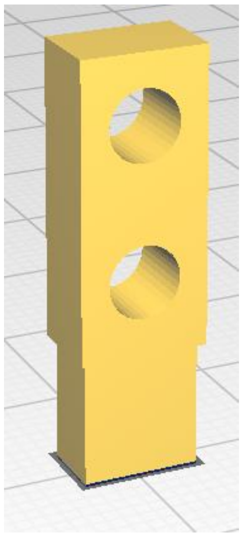


Figure 7: servo mount

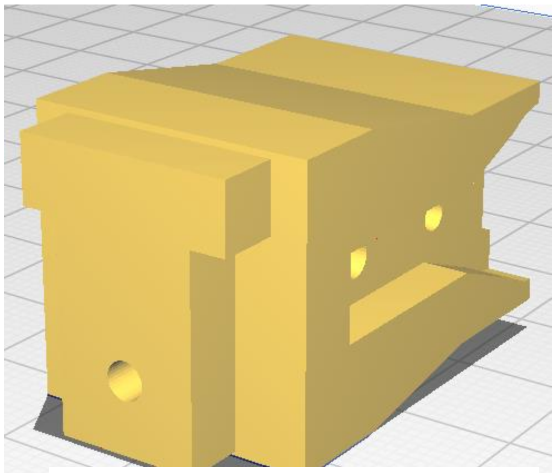


Figure 6: wrist arm

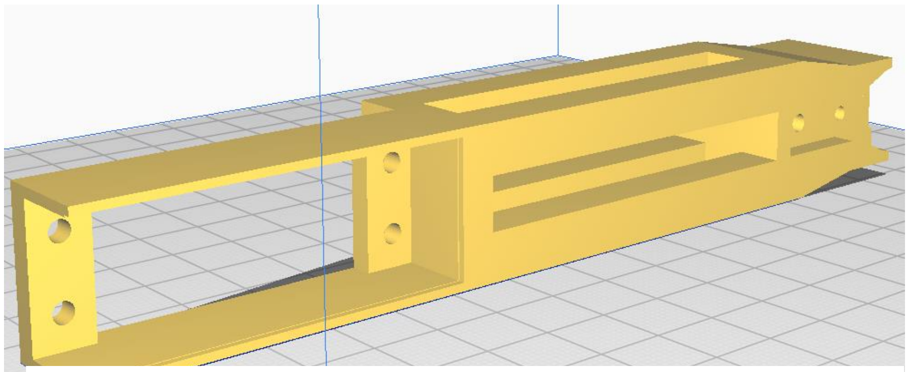


Figure 7: servo holder arm

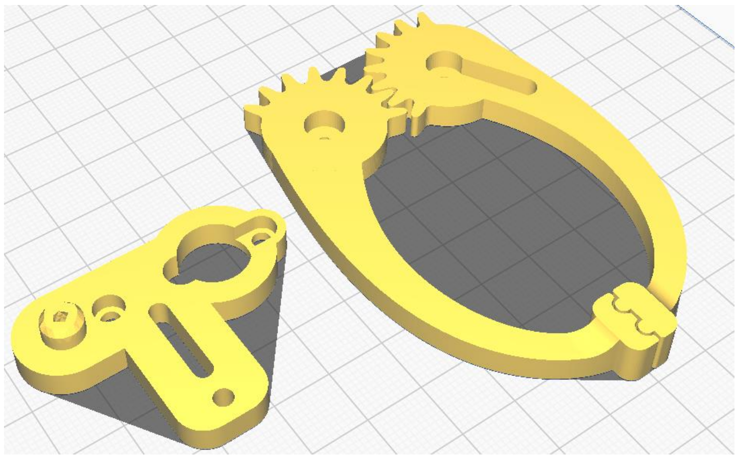


Figure 8: Gripper

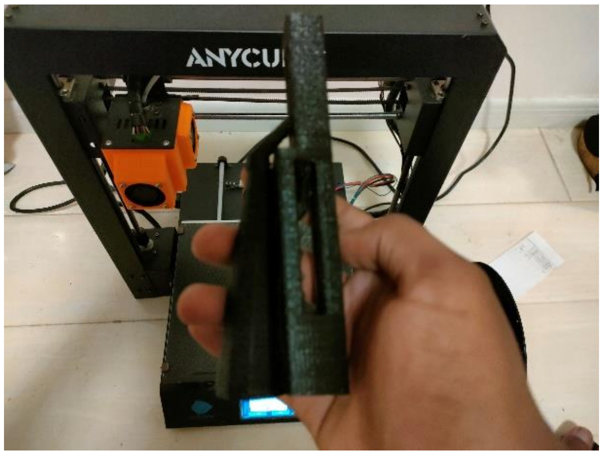


Figure 9: elbow arm just printed

Figure 10: after removing supports

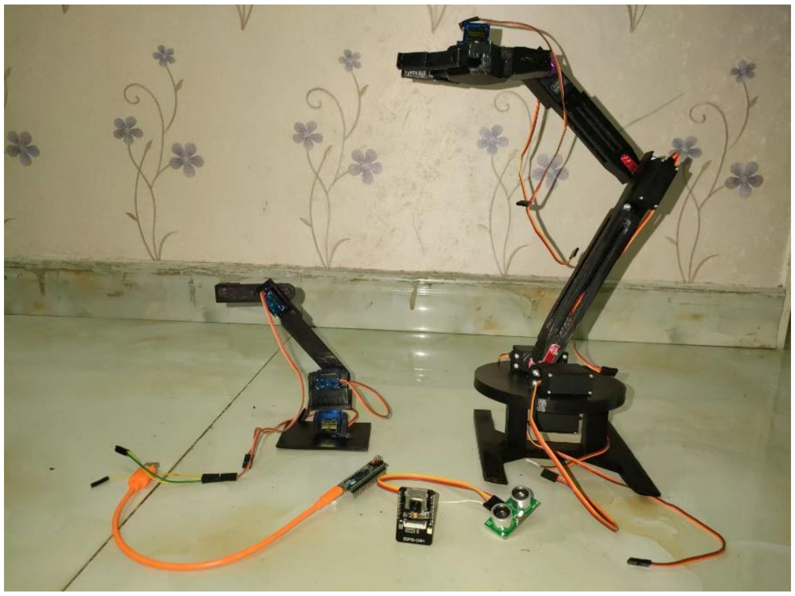


Figure 11: arm fully assembled

Circuit

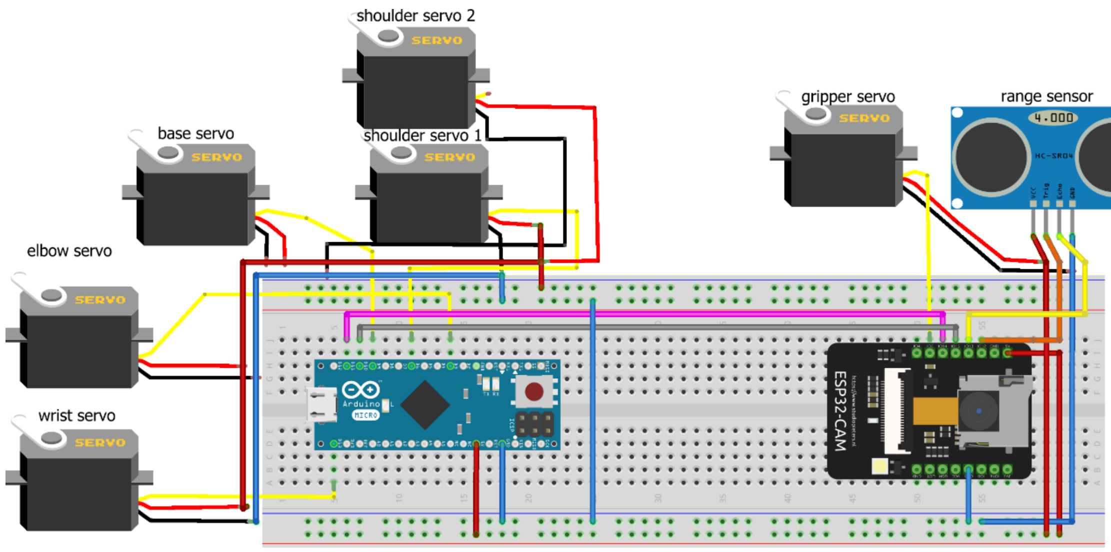
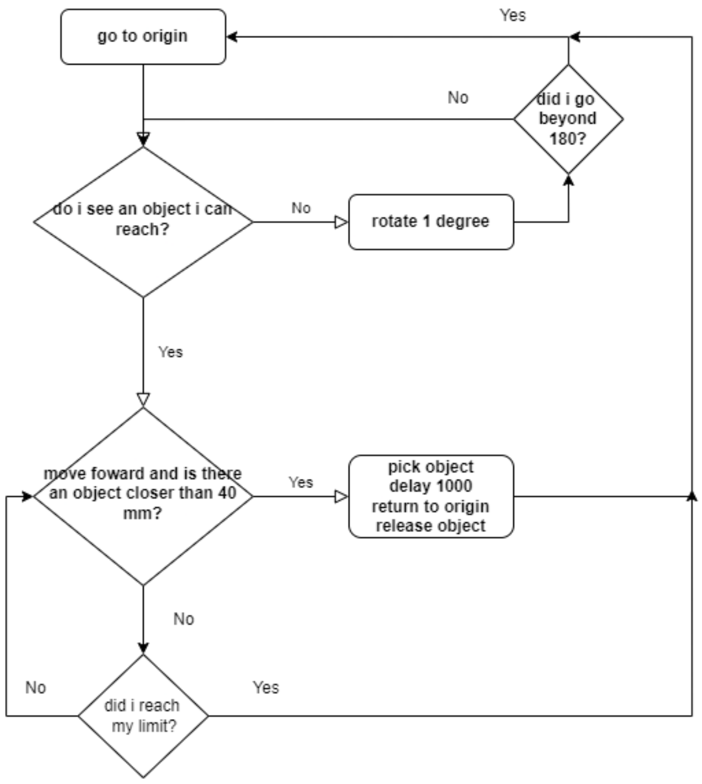


Figure 12: circuit diagram designed

Software Structure



go to origin
Yes
No
did i go beyond 180?
do i see an object i can reach?
No
rotate 1 degree
Yes
move foward and is there an object closer than 40 mm?
Yes
pick object delay 1000 return to origin release object
No
No
Yes
did i reach my limit?

Figure 13: control software flowchart

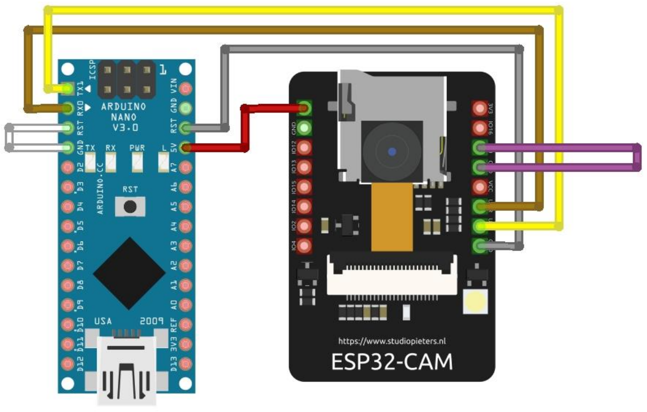


Figure 14: esp32 to Arduino serial connection

We used 4 futaba 20 kg servos, 1 mg995 servo for the base and 1 sg90 servo for the gripper

At the base we used 2 futaba 20kg.cm servos to reach the required 40kg.cm torque and we find out that they had to be offset by 180 degrees to make them move together.

We used Arduino's servo library to control the servos, when the esp32 sends rotation angle as 360 we retain the angle we are on.

We need a 2 pins for an ultrasonic range sensor, and 6 for the servos, and perhaps a camera, so we chose the control board to be an esp32-cam running at 240 Mhz with 4 MB flash but the esp32-cam has a few usable pins, so we decided to use an Arduino micro to control the servos and get command from the esp32-cam over serial.

We run a loop that check if the esp32-cam sent data and then parse it as follows

Input from esp32-cam: 90.0,0.0,10.5,0.0\n

Use **strtok** to split input where char is "," or "\n" into 4 substrings (shoulder, elbow, wrist and base angles), then use **atoi** to make the substrings into float and send those to the servos.

The esp32 follows the loop described in figure 13 where it constantly check for objects it can reach by reading the nearest object from the range sensor, if it's further than 30 cm it rotates to find other objects and keeps doing that until it sees an object it can reach there, it stops rotating and then extend the arm and when it is very close to the object (5-3 cm) it closes and gripper and picks it up, of course the esp32 controls directly only the gripper and the range sensor, everything else it controls indirectly through the Arduino.

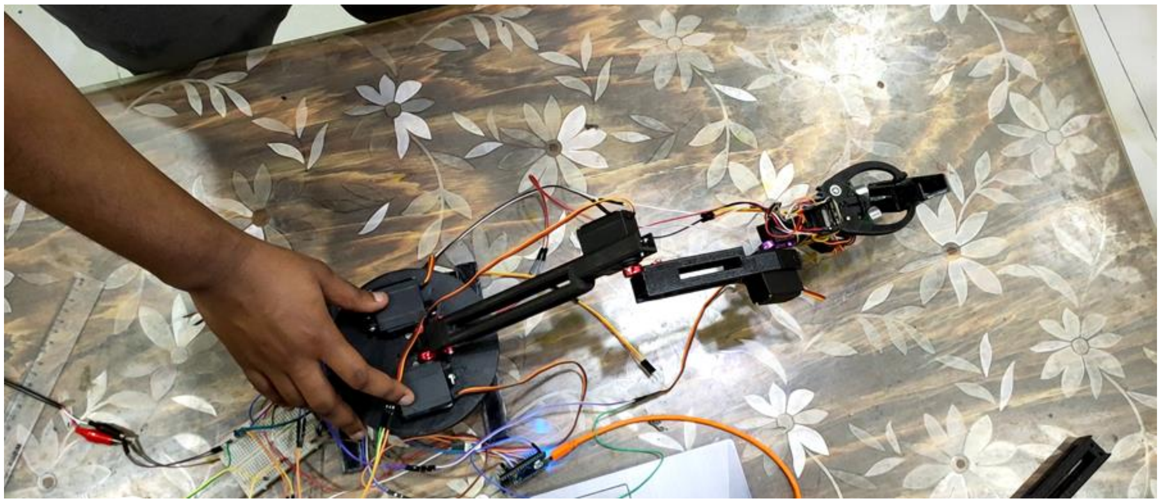


Figure 15: robotic arm picking an object

Conclusions and Future Work

The issues that it has are many, it is not stable due to the mobile servo mounts, the base servo’s encoder is not very accurate to move at 1 degree increments(mg995), the algorithm used is very limited, the esp32 restarts most times when the gripper servo closes due to a current surge, the servos are not supported on both sides, the part that holds the servos are weak and break easily, the base cannot stay straight when the arm is fully extended, unmanaged wiring and more.

But it has and while we keep working on it will keep being a great learning tool, which we can keep iterating over and over learning new things while making it perform better.

To conclude it successfully but not efficiently achieved the task of automatically picking up objects by detecting where they are.

The future work is massive, as the issues are, not only will we fix the issues but also add more functionalities like using the camera, i2c and wireless capabilities of the esp32.

References

1. https://peakd.com/hive-196387/@juecoree/forward-and-reverse-kinematics-for-3r-planar-manipulator
2. https://www.instructables.com/3D-Printed-2-Servo-Robot-Arm/
3. https://www.instructables.com/Tertiarm-3d-Printed-Robot-Arm/
4. https://www.stlfinder.com/model/robot-arm-servo-motors-ArFMnuH7/4049214/
5. https://duino4projects.com/record-and-play-3d-printed-robotic-arm-using-arduino/
6. https://robotics.stackexchange.com/questions/14504/robotic-arm-select-servo-motors
7. http://fabacademy.org/2020/labs/kamakura/students/hiroaki-kimura/assignments/week16/simple-stick-arm-robot
8. https://uk.pi-supply.com/products/ultrasonic-distance-sensor-hc-sr04
9. https://www.robotshop.com/community/tutorials/show/robot-arm-torque-calculator

10. https://blog.robotiq.com/how-to-calculate-a-robots-forward-kinematics-in-5-easy-steps
11. https://demonstrations.wolfram.com/DenavitHartenbergParametersForAThreeLinkRobot/
12. http://www.roboanalyzer.com/
13. https://web.archive.org/web/20160531030510/http://freespace.virgin.net/hugo.elias/models/m_ik2.htm
14. https://github.com/khoih-prog/ESP_WiFiManager#howto-use-analogread-with-esp32-running-wifi-andor-bluetooth-btble