Carthesian Robot used for Drone Landing, Charging & Storing

LIȚOIU[1] Sebastian1[1], PETCU Vlad Adrian

[1]Facultatea:FIIR, Specializarea:Robotica, Anul de studii: IV, e-mail:litoiu.sebastian19@gmail.com

Conducător ştiinţific: S.L. Dr. Ing. Cozmin **CRÎSTOIU**
Conducător ştiinţific: S.L. Dr. Ing. Mario Andrei **IVAN**

The work consists of a modular Cartesian robot designed to intercept and store drones. The robot is electrically powered by 3 electric motors controlled by an Arduino microcontroller. It has 2 video cameras. The first camera captures the x and z coordinates of the drone, while the second camera captures the position on the y-axis. These cameras transmit the position of the drone to the Arduino board, which controls the axes to make the drone land on the pallet. After landing, the drone is stored in a dedicated space together with the pallet, which will also facilitate its charging.

1. Introduction

We wanted to build a robot that will be able to perform like a helipad that also will facilitate its charging. The robot is electrically powered by 3 electric motors controlled by an Arduino microcontroller. It has 2 video cameras. The first camera captures the x and z coordinates of the drone, while the second camera captures the position on the y-axis. These cameras transmit the position of the drone to the Arduino board, which controls the axes to make the drone land on the pallet. After landing, the drone is stored in a dedicated space together with the pallet, which will also facilitate its charging.

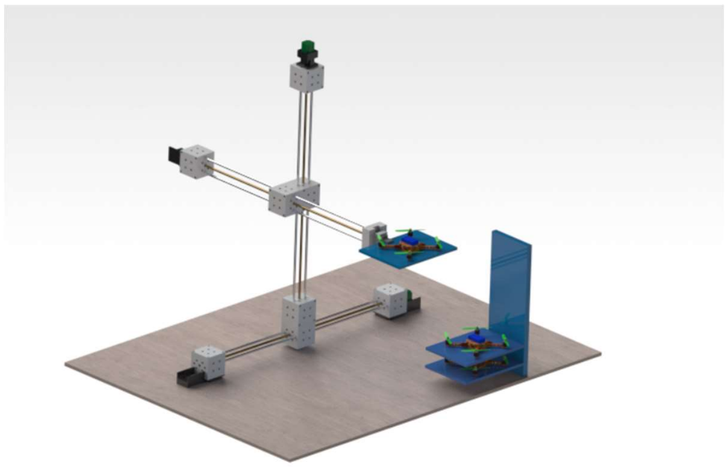


Figure 1 The CAD representation of the Project

2. Current Stage

The Robot is ready from an electro-mechanical standpoint. The computer vision software works very good in recognizing the elements that we've set it up for. We still need to do the serial communication between the cameras and the Arduino board, and also the Arduino processing for manipulating our data.

Each one of the 3 Axis is converting rotational energy from their own motors into prismatic movement. This is realized by having a nut lead-screw mechanism accompanied by 2 guiding rails, each having longitudinal bearings for a smoother action. The leadscrew has also 2 bearings attached for a smoother operation

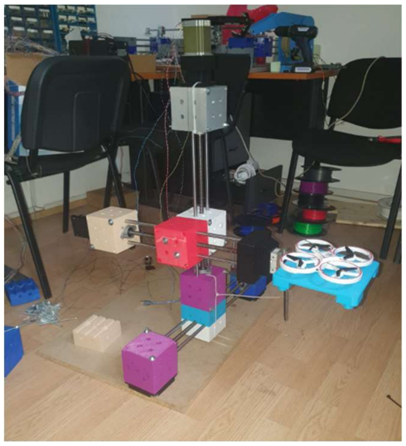


Figure 2 The State of the Project

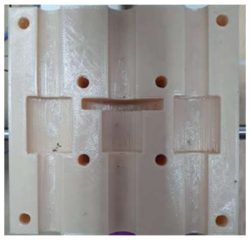


Figure 3 The Case for the Nut and Liniar Bearings

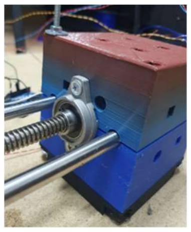


Figure 4 The Screw Bearings

Each one of the 3 motors is paired with an elastic coupling to be able to engage the screw, which also has been milled to the preferred diameter. Electric Motors will be driven using an Arduino Board and 3 TB6560 drivers.
To be able to triangulate the correct position of the drone in space we have used 2 cameras. One for obtaining the x and z coordinates and one for obtaining the y component. In the below example, the program is set to recognize the color bright green. After it finds it, it stores the desired values and sends the ones that we need into a file.

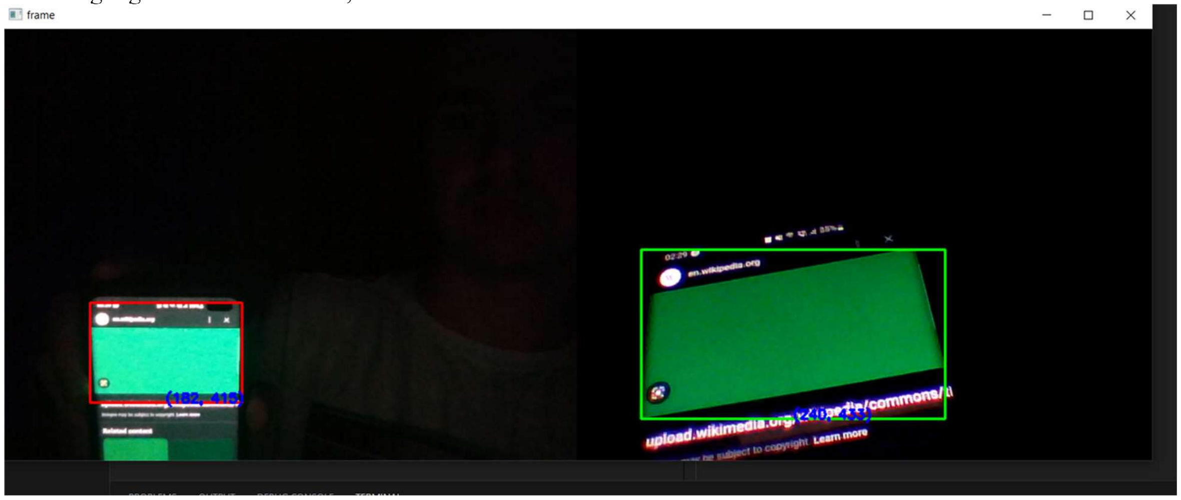
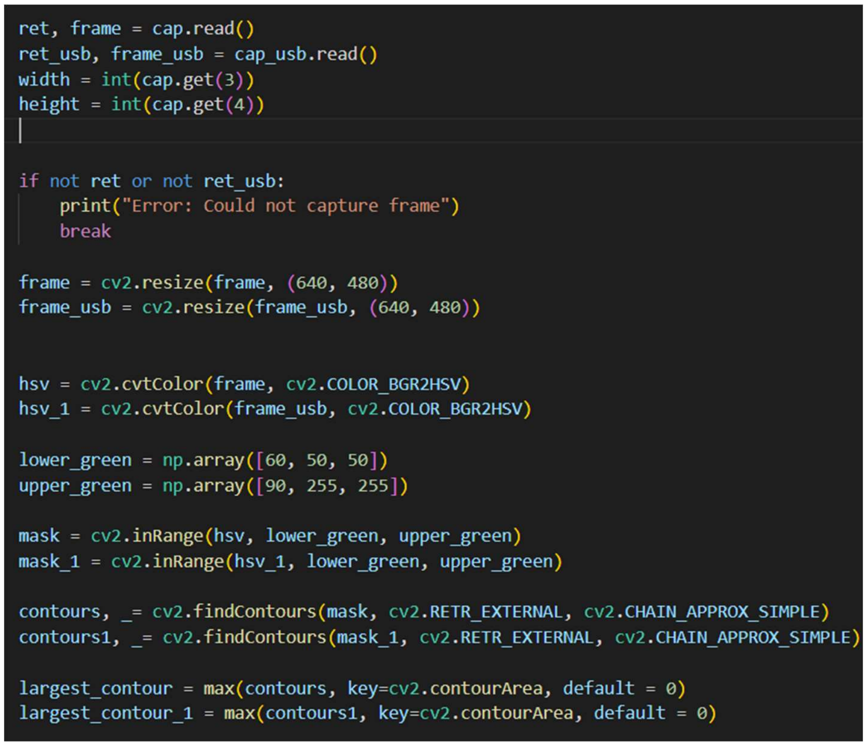
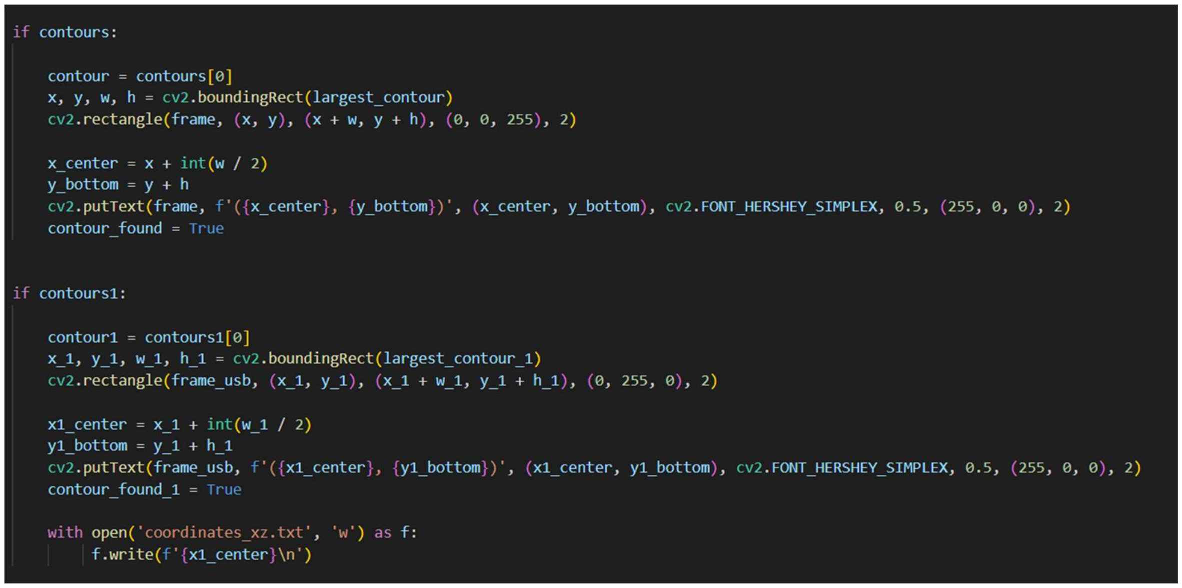


Figure 5 The GUI with 2 views and their respective coordinates



```
ret, frame = cap.read()
ret_usb, frame_usb = cap_usb.read()
width = int(cap.get(3))
height = int(cap.get(4))


if not ret or not ret_usb:
    print("Error: Could not capture frame")
    break

frame = cv2.resize(frame, (640, 480))
frame_usb = cv2.resize(frame_usb, (640, 480))


hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
hsv_1 = cv2.cvtColor(frame_usb, cv2.COLOR_BGR2HSV)

lower_green = np.array([60, 50, 50])
upper_green = np.array([90, 255, 255])

mask = cv2.inRange(hsv, lower_green, upper_green)
mask_1 = cv2.inRange(hsv_1, lower_green, upper_green)

contours, _= cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours1, _= cv2.findContours(mask_1, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

largest_contour = max(contours, key=cv2.contourArea, default = 0)
largest_contour_1 = max(contours1, key=cv2.contourArea, default = 0)
```

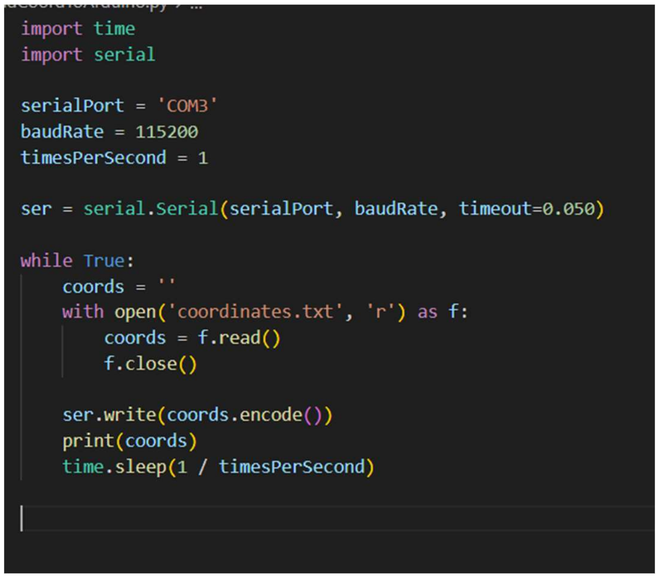
Figure 6 Code for sizing the view and defining preliminary contours.



```python
if contours:

    contour = contours[0]
    x, y, w, h = cv2.boundingRect(largest_contour)
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    x_center = x + int(w / 2)
    y_bottom = y + h
    cv2.putText(frame, f'({x_center}, {y_bottom})', (x_center, y_bottom), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
    contour_found = True


if contours1:

    contour1 = contours1[0]
    x_1, y_1, w_1, h_1 = cv2.boundingRect(largest_contour_1)
    cv2.rectangle(frame_usb, (x_1, y_1), (x_1 + w_1, y_1 + h_1), (0, 255, 0), 2)

    x1_center = x_1 + int(w_1 / 2)
    y1_bottom = y_1 + h_1
    cv2.putText(frame_usb, f'({x1_center}, {y1_bottom})', (x1_center, y1_bottom), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
    contour_found_1 = True

    with open('coordinates_xz.txt', 'w') as f:
        f.write(f'{x1_center}\n')
```

Figure 7 Finding the color, putting a rectangle around it and getting its coordinates.

We've managed to send data to the Arduino by creating a python program that communicates through serial with our board.



```python
import time
import serial

serialPort = 'COM3'
baudRate = 115200
timesPerSecond = 1

ser = serial.Serial(serialPort, baudRate, timeout=0.050)

while True:
    coords = ''
    with open('coordinates.txt', 'r') as f:
        coords = f.read()
        f.close()

    ser.write(coords.encode())
    print(coords)
    time.sleep(1 / timesPerSecond)
```

Figure 8 Serial Communication with the Arduino

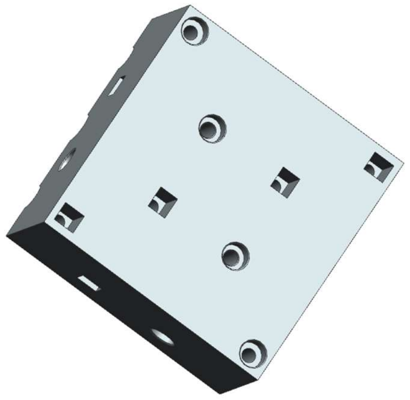


Figure 9 Cad representation of the case

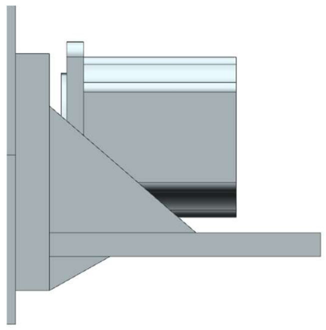


Figure 10 Y axis motor mount

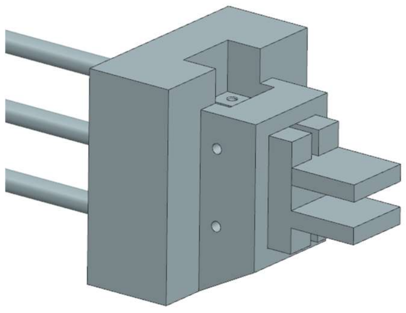


Figure 11 Gripper Case

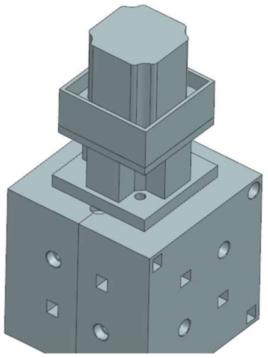


Figure 12 Z axis Motor Mount

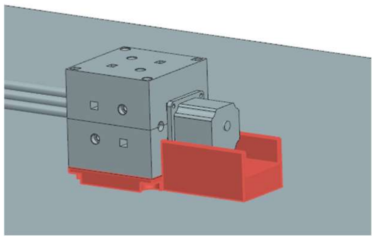


Figure 13 X axis Motor Mount

3. Conclusion

To conclue, we have learned a lot from this project and we are thrilled to finnish it and also be able to develop new and complex robots. We still need to complete the followings :

- Finishing the serial communication
- Programming the Arduino for data manipulation
- Finishing the design for the docking station
- Implementing wireless charging into the docking station
- Actuating the gripper using an electro-valve

4. Bibliography

1. https://www.youtube.com/watch?v=MvEpi4FDhuI&list=PLUb-vZlf8Y40hHA_D_AWWIaXbTFFewU_M&ab_channel=OleksandrStepanenko