PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Summer Internship Report on

"Automation of Diagnostics Laboratory"

Submitted by

Srinidhi B S (PES1UG20EC201)

June – July 2023

Under the Guidance of

Dr. Venkatarangan M J

Professor

Department of Electrical and Electronics

PES University

Bengaluru

Centre for Robotics, Automation and Intelligent Systems (cRAIS)
Room No. B-1212, 100 Feet Ring Road, BSK III Stage, Bangalore - 560085.

PES UNIVERSITY

CERTIFICATE

This is to certify that the Report entitled

'Automation of Diagnostics Laboratory'

is a bonafide work carried out by

Srinidhi B S (PES1202000899)

In partial fulfillment for the completion of 8th semester internship work in the Program of Study B.Tech in Electronics and Communication Engineering, under rules and regulations of PES University, Bengaluru during the period June – July 2023. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report.

Dr. Venkatarangan MJ
Faculty Mentor
Professor, EEE Department
PES University,
Bengaluru – 85.

DECLARATION

I, Srinidhi B S, hereby declare that the work entitled, ‘Title of the Internship’, was carried out under the guidance of **Dr. M.J. Venkatarangan**, Professor, Department of Electrical and Electronics, at the Centre for Robotics, Automation and Intelligent Systems (cRAIS) . This report is being submitted in partial fulfillment of the requirements for completion of 8 th Semester course work in the Program of Study, B.Tech in Electronics and Communication Engineering.

PLACE: BENGALURU

DATE: 31/7/2023

Name and sign: Srinidhi B S

[illegible]

ABSTRACT

In various diagnostic labs, the archiving of test tube samples is effectively performed using automated machines. Our project aims to automate the process of archiving blood samples of a Diagnostic Lab, Bengaluru. The project is undertaken in two parts. First part is to automate the complete automation using the robotic manipulator bought which is the main focus of the internship completed. The second part is to port the same application to the in-house built robot which could be gantry robot or a multi DOF. To achieve the first part, a, four axis robotic arm (Dobot Magician) is programmed using python programming language and then integrated with image processing techniques. Firstly, using a USB camera live feed data of the test tube samples placed in a test tube holder rack is captured. Using YoloV5 the test tubes are identified and the centroids of the test tubes are detected. With the help of the centroids, a matrix is generated in order to identify the presence of the test tube in the rack.

Next, the robotic arm (Dobot Magician) is commanded to attain the coordinate values of the centroids of test tube and perform the mechanism of pick and place. The firmware on the Dobot magician works on inverse kinematics to rotate the motors according to Cartesian co-ordinates Once the test tubes are archived the details of the patient on each test tube is determined using a barcode scanner to scan the barcode placed on the surface of the test tubes and is connected with a LIS (Laboratory Information System) software. The demonstration is prepared for picking up test tube one by one from a fixed location of rack, bar scanned to update LIS on back end and then categorise and place it in the destination rack for achieving and also to another destination rack if it is not tested for whatever reason.

Acknowledgement

We take this opportunity to express our deepest sense of gratitude and sincere thanks to everyone who helped us to complete this work successfully. We express our sincere thanks to **Dr. M.J. Venkatarangan**, Professor, Department of Electrical and Electronics, PES University, Bengaluru and **Dr. Sethuram D**, Professor, Department of Mechanical Engineering, PES University, Bengaluru for providing us with all the necessary facilities and support. We would also like to express our sincere gratitude to **Dr. Shikha Tripathi**, Chairperson, Department of Electronics and Communication Engineering, PES University, Bengaluru. Finally, we thank our friends and team mates who contributed to the successful fulfilment of this project work.

Srinidhi B S

Table of Contents

Table of Figures and Tables

1. INTRODUCTION

The project of "Diagnostics lab automation" which is to be implemented in the Diagnostics Laboratory is aimed at automating the process of archiving. After the sample is tested, the sample is stored for a fixed time period in cold storage, for re-tests and review. The test tubes are scanned in an LIS mapping each sample to a specific position in the corresponding tray whose code is also recorded, enabling one to trace back. The transfer is completed only if all tests assigned to the given sample have been completed.

The process of scanning and transferring the test tubes is currently being done manually. Our task is to automate the process of reducing human intervention effectively reducing error. The task is set to be performed by a robotic arm with 5 DOF. Image processing has been used to make the process efficient by reducing the number of work cycles to the number of test tubes present

<u>System Diagram</u>

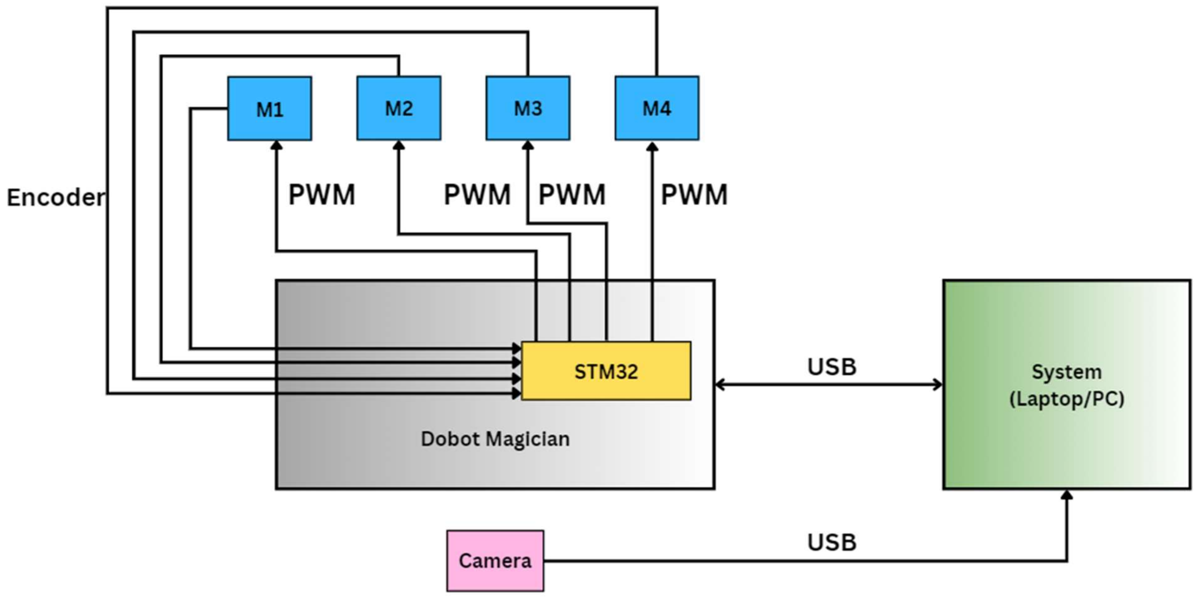


Figure 1 : System Diagram

M1,M2,M3 and M4 – motors corresponding to their respective joints

STM32 – 32 bit microcontroller used to control Dobot Movements

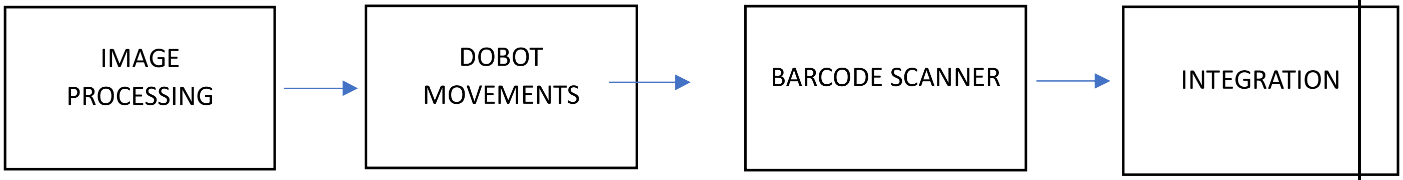
2. METHODOLOGY

I) Image Processing

II) Dobot Manipulation

III) Barcode Scanner

IV) Integration



I) IMAGE PROCESSING

1. Creating datasets using roboflow.
2. YOLOv5 framework is used for machine learning process encompassing the stages of dataset testing, training, and validation.
3. Centroid detection using python code.
4. Generation of matrix using python code.

1. Creating datasets using roboflow

We have used multiple images of the racks containing the test tubes as datasets for training the model. The datasets are created using roboflow, a platform which provides functionalities like dataset organization and annotation to prepare datasets for training machine learning models. We have used bounding box option with single-label classification and polygon

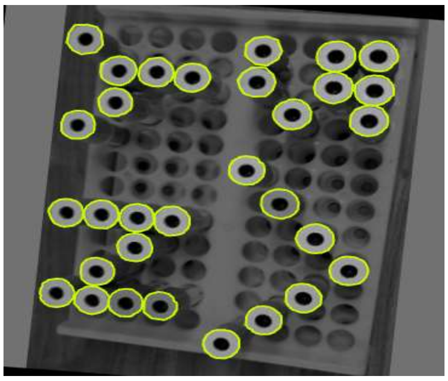
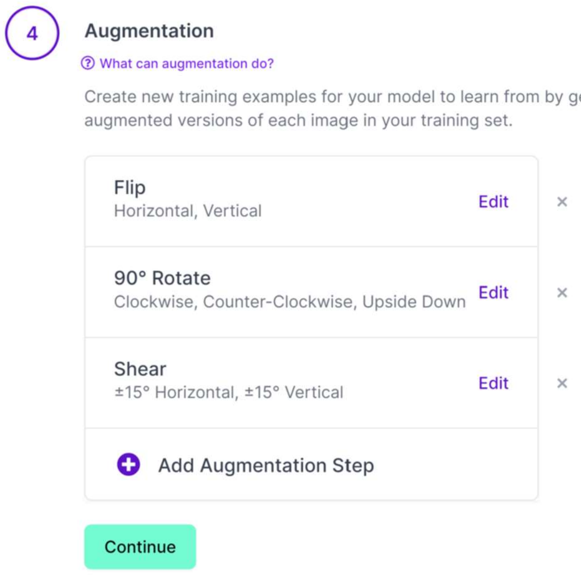


Figure 2 : Annotation

selection.

2. YOLOv5 framework

YOLOv5 framework is used for machine learning process encompassing the stages of dataset testing, training, and validation. Each object (test tube) is labelled using bounding boxes, using OpenCV. The precise model used is YOLOv5 version2. A dataset of 85 images have been used. Optimum resizing as per blank spaces present in the image is chosen to be 400x400. Along with rotation, brightness and orientation variations, grayscale and shearing of bounding boxes have been introduced during augmentation (Total images 235).


4 Augmentation

What can augmentation do?

Create new training examples for your model to learn from by generating augmented versions of each image in your training set.

Flip
Horizontal, Vertical

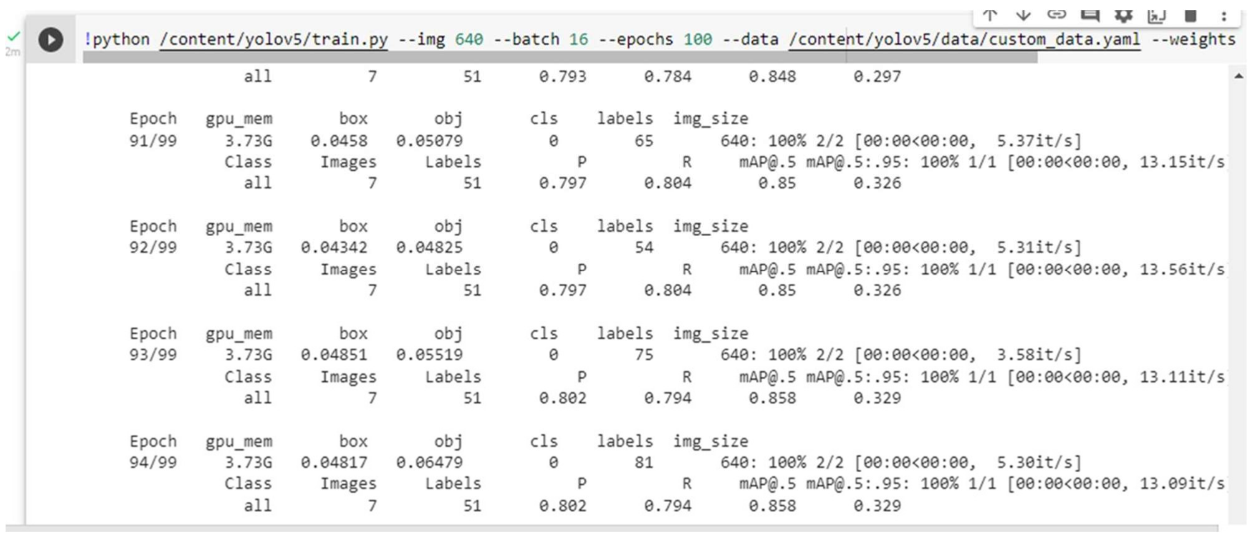
90° Rotate
Clockwise, Counter-Clockwise, Upside Down

Shear
±15° Horizontal, ±15° Vertical

Figure 3 : Dataset Generation

For training, verification and validation, data set was split split as 82% (193 images) for training, 9% (21 images) for validation and 9% (21 images) for verification.

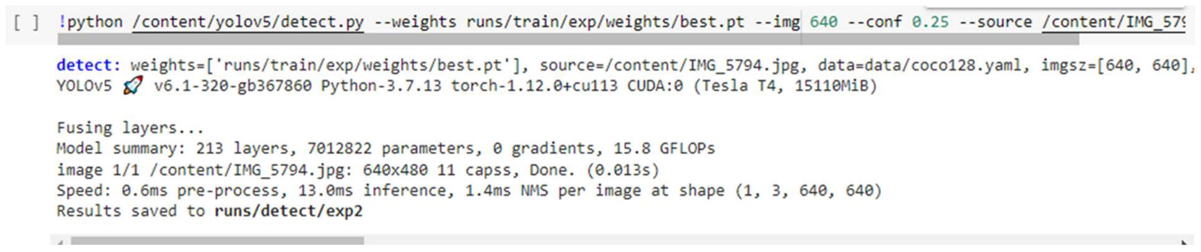
Training run for 100 epochs



```
!python /content/yolov5/train.py --img 640 --batch 16 --epochs 100 --data /content/yolov5/data/custom_data.yaml --weights
                 all          7         51      0.793      0.784      0.848      0.297

      Epoch    gpu_mem       box       obj       cls    labels  img_size
      91/99      3.73G    0.0458   0.05079         0        65       640: 100% 2/2 [00:00<00:00,  5.37it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.15it/s
                 all          7         51      0.797      0.804       0.85      0.326

      Epoch    gpu_mem       box       obj       cls    labels  img_size
      92/99      3.73G   0.04342   0.04825         0        54       640: 100% 2/2 [00:00<00:00,  5.31it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.56it/s
                 all          7         51      0.797      0.804       0.85      0.326

      Epoch    gpu_mem       box       obj       cls    labels  img_size
      93/99      3.73G   0.04851   0.05519         0        75       640: 100% 2/2 [00:00<00:00,  3.58it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.11it/s
                 all          7         51      0.802      0.794      0.858      0.329

      Epoch    gpu_mem       box       obj       cls    labels  img_size
      94/99      3.73G   0.04817   0.06479         0        81       640: 100% 2/2 [00:00<00:00,  5.30it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 13.09it/s
                 all          7         51      0.802      0.794      0.858      0.329
```

Figure 4 : Training

Testing



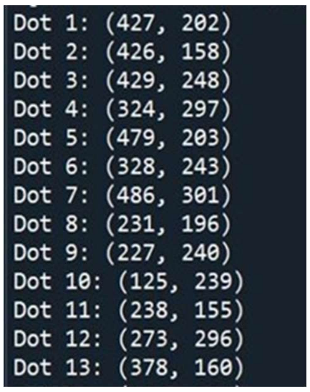
```
[ ] !python /content/yolov5/detect.py --weights runs/train/exp/weights/best.pt --img 640 --conf 0.25 --source /content/IMG_579

    detect: weights=['runs/train/exp/weights/best.pt'], source=/content/IMG_5794.jpg, data=data/coco128.yaml, imgsz=[640, 640],
    YOLOv5 🚀 v6.1-320-gb367860 Python-3.7.13 torch-1.12.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

    Fusing layers...
    Model summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
    image 1/1 /content/IMG_5794.jpg: 640x480 11 capss, Done. (0.013s)
    Speed: 0.6ms pre-process, 13.0ms inference, 1.4ms NMS per image at shape (1, 3, 640, 640)
    Results saved to runs/detect/exp2
```

Figure 5 : Testing

3.Centroid detection using python code

For each labelled object (test tube) in the datasets, the centroids are calculated by averaging the x and y coordinates of all the pixels belonging to that object. The centroids are used to determine the position of the test tube in the rack. YOLOv5 uses the technique of drawing bounding boxes around the roughly detected shape. The camera intrinsic matrix is calculated using a chess board. The camera matrix consists of fx,fy, cx, and cy (focal lengths and centres). By calculating the medians of the perpendicular bisectors of the adjacent sides, pint of intersection or centroid is calculated.

```
cX = int((topleft[0] + bottomright[0]) / 2.0)
cY = int((topleft[1] + bottomright[1]) / 2.0)
```

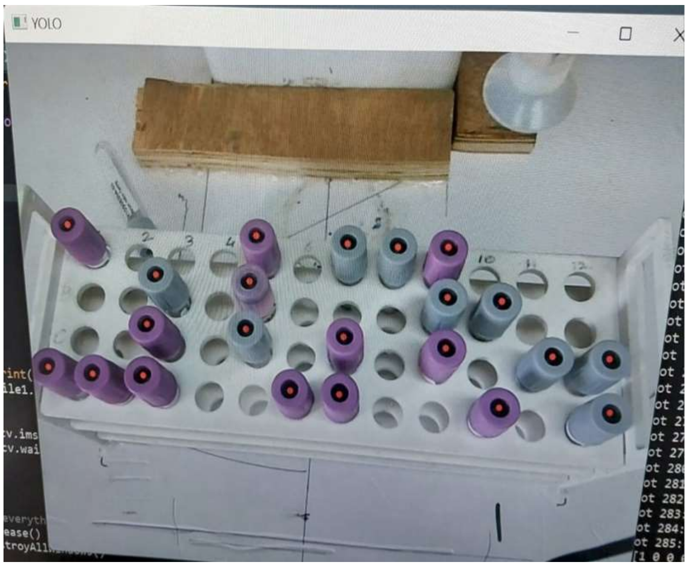


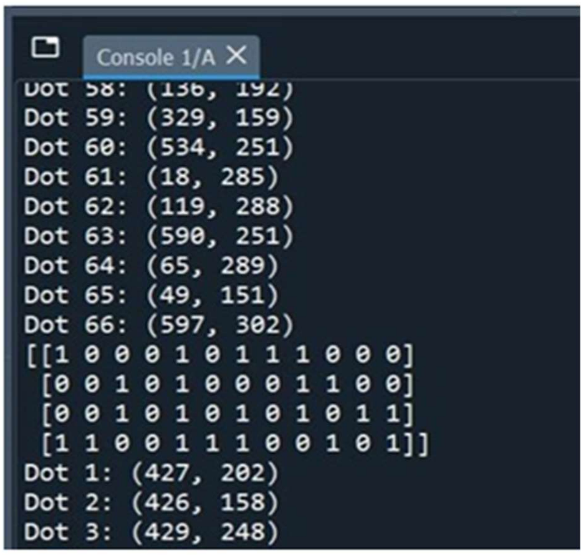
```
Dot 1: (427, 202)
Dot 2: (426, 158)
Dot 3: (429, 248)
Dot 4: (324, 297)
Dot 5: (479, 203)
Dot 6: (328, 243)
Dot 7: (486, 301)
Dot 8: (231, 196)
Dot 9: (227, 240)
Dot 10: (125, 239)
Dot 11: (238, 155)
Dot 12: (273, 296)
Dot 13: (378, 160)
```

Figure 6 : Centroid Generation

4. Generation of matrix using python code
By providing pixel co-ordinates of first slot, i.e., origin, we can approximately determine the positions of successive slots using slot interval distance also provided. By checking for centroids at these poisons, a binary matrix is formed using 1 as a presence flag.

Due to the video being dynamic, multiple co-ordinates are detected for the same test tube. Since these will not coincide with the predicted slot co-ordinates based on interval, we provide an offset of 10 pixels for the x-axis and 15 pixels for the y axis to deal with the same.





```
Dot 58: (136, 192)
Dot 59: (329, 159)
Dot 60: (534, 251)
Dot 61: (18, 285)
Dot 62: (119, 288)
Dot 63: (590, 251)
Dot 64: (65, 289)
Dot 65: (49, 151)
Dot 66: (597, 302)
[[1 0 0 0 1 0 1 1 1 0 0 0]
 [0 0 1 0 1 0 0 0 1 1 0 0]
 [0 0 1 0 1 0 1 0 1 0 1 1]
 [1 1 0 0 1 1 1 0 0 1 0 1]]
Dot 1: (427, 202)
Dot 2: (426, 158)
Dot 3: (429, 248)
```

Figure 7 : Real time detection

II) DOBOT MOVEMENTS

1. Determination of co-ordinates of all slots.
2. Reading the binary matrix to locate slots in which test-tubes are present.
3. Pick and place.
4. Test tube manoeuvre to scan barcode.

DOBOT Magician is a multifunctional small-sized desktop robotic arm for practical training education, deploying multiple functions, such as 3D printing, laser engraving, etc. It has 4-axes of motion and 4 Degrees of Freedom.

Module	Version
Dobot Studio	1.9.4
Dobot Firmware	3.7.0

Table 1 : Dobot Studio Version details

The CP210x USB to UART COM Port driver was used for device operation. In this project, we have used the suction cup and air pump for pick and place operations. The signal (SW1 and GP1) and power lines of the pump were connected to the Dobot via EIO (Extended I/O) pins located at the base of the machine. The suction was interfaces with those on the forearm (slot1/GP3) articulated to Joint 4.

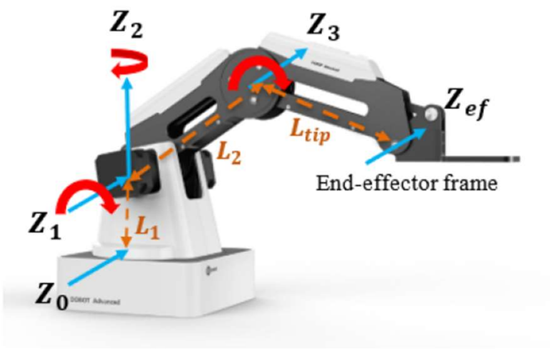
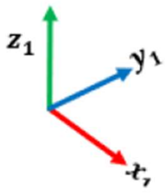


Figure 8 : Dobot Magician DOF



The controls were handled using the python script editor IDE provided in Dobot Studio. For additional python operations, the required library modules were added to the Ide path of Dobot Studio.

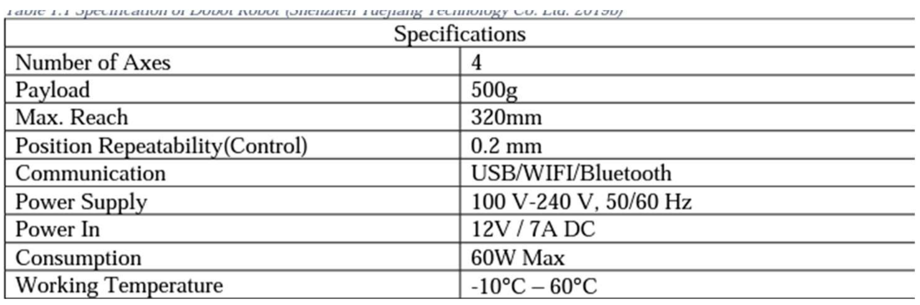


Table 1.1 Specification of Dobot Robot (Shenzhen Yuejiang Technology Co. Ltd. 2019b)

Specifications	
Number of Axes	4
Payload	500g
Max. Reach	320mm
Position Repeatability(Control)	0.2 mm
Communication	USB/WIFI/Bluetooth
Power Supply	100 V-240 V, 50/60 Hz
Power In	12V / 7A DC
Consumption	60W Max
Working Temperature	-10°C – 60°C

Table 2 : Dobot Magician Hardware limitations

The controller estimates required joint torques to ensure Dobot's end-effector follow the desired position, velocity and acceleration. The general layout of the control architecture for 'Cartesian control of Dobot robot' is given in Fig-2 where θd, $\theta' d$,and $\theta'' d$ represent desired joint angles, velocity and acceleration respectively.

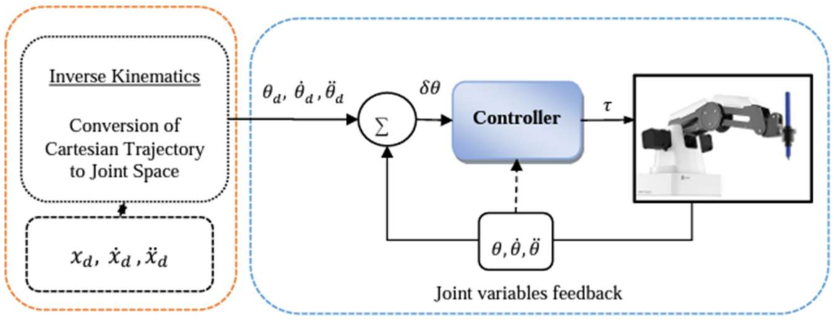
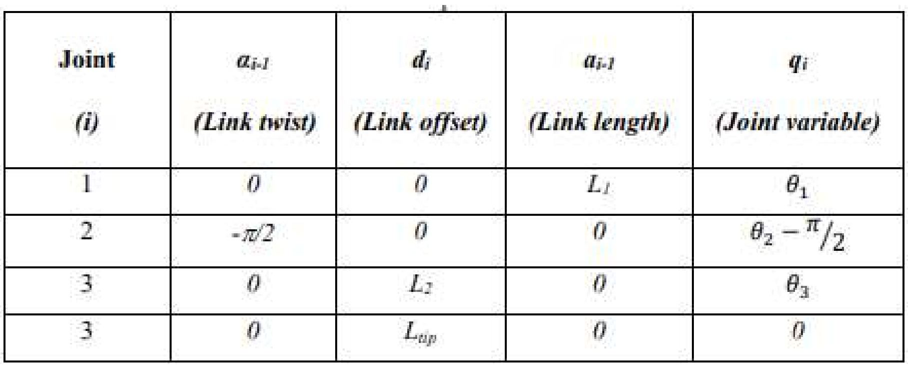

Inverse Kinematics
Conversion of Cartesian Trajectory to Joint Space
$x_d, \dot{x}_d, \ddot{x}_d$
$\theta_d, \dot{\theta}_d, \ddot{\theta}_d$
Σ
$\delta\theta$
Controller
τ
$\theta, \dot{\theta}, \ddot{\theta}$
Joint variables feedback

Figure 9 : Inverse Kinematics

Dobot Kinematics and interface

Kinematics

DH Parameters for dobot magician



Joint (i)	α_{i-1} (Link twist)	d_i (Link offset)	a_{i-1} (Link length)	q_i (Joint variable)
1	0	0	L_1	θ_1
2	$-\pi/2$	0	0	$\theta_2 - {}^{\pi}/_{2}$
3	0	L_2	0	θ_3
3	0	L_{up}	0	0

Table 3 : DH Parameters

Interface

Dobot Magician comes with its own software, DobotStudio, to perform the built-in operations. The manufacture of Dobot Magician gives users SDKs in different programming languages for secondary development of the Dobot Magician. In this project, we have made use of the python script available on-platform.

DLL files

While the Dobot API allowed provided a platform for the execution of python commands, in order to get access to the MCU of the motherboard of the Dobot Magician, DLL files or Dynamic-Link Library files are required they contain a set of data and code to execute certain operations in Windows operating systems. This was done by the DobotDll.dll file.

Dobot controller supports two types of commands: immediate command and queue command. We mainly utilized the queued commands which are executed by the Dobot controller maintaining the sequence of receiving the command.

1. Determination of co-ordinates of all slots

Given that the Dobot was capable of linear motion in the x, y, and z axes, with 1 unit movement being about 0.35mm, we provided the coordinate values of the left-top most slot anointing it as the origin (ox, oy). Additionally, the slot interval of 21 units for x and y axes were also provided. With the given information, the Dobot was able to compute the coordinates of any slot using indices provided.

$$\boxed{x, y = ox+(i*ds), oy+(j*ds)}$$

where x and y are co-ordinate values of the slot, ox and oy origin coordinates, ds- the distance between slots being 21 units. For the movements of the z axes, the raised position for which the test tube is not an obstruction and lowering height at which it is securely placed for the given setup are provided and used via flagging.

2.Reading matrix to locate slots in which test-tubes are present and pick place

Reading the binary matrix which has been generated by the image processing segment, the indices of all the slots where a test-tube is present is noted. The methods mentioned in the section above, the coordinates of the test tube are determined and the test tube is picked up using the suction module. After the scanning mentioned in (section III) is completed, the tubes are placed in an adjacent tray using the same coordinate mapping mechanism mentioned above.

Vacuum Suction Cap	Cap Diameter	20 mm
	Pressure	-35 kPa

Figure 10 : Suction cup Hardware Specifications

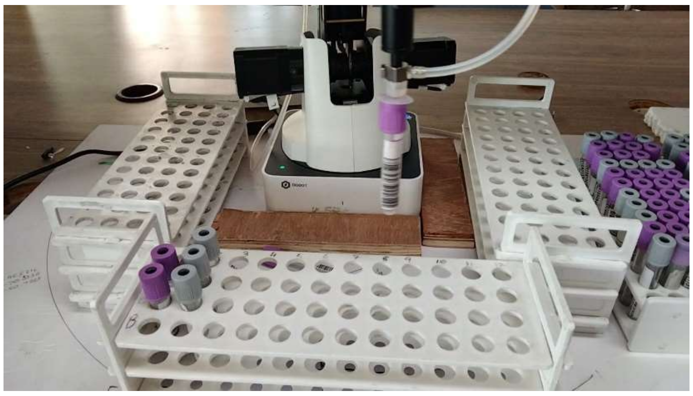


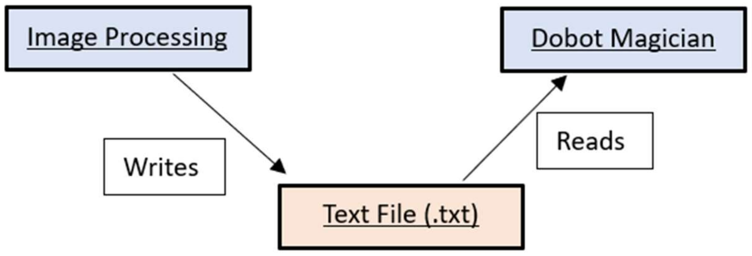
Figure 11 : Work in Action

3.Test tube manoeuvre to scan barcode.

The scanner (Retsol LS500 barcode scanner) is a commercial scanner which can scan codes present virtually in front of it. In order to ensure the barcode (which is present on a single side of the tube) is captured, the joint4 placed above the suction cup is made to rotate in steps till the barcode is noted. Upon a successful scan, the sample is placed in the destination rack.

III) Integration

The matrix generated by the image processing segment is written into a .txt file and retrieved by the Dobot Studio IDE using file handling. This is done in order to ensure persistence of data. In order to ensure compatibility, the required library modules were added to the Ide path of Dobot Studio. The three modules namely, image processing, barcode scanning and LIS access are integrated using modularity of python by importing them in cascade and then calling the required functions



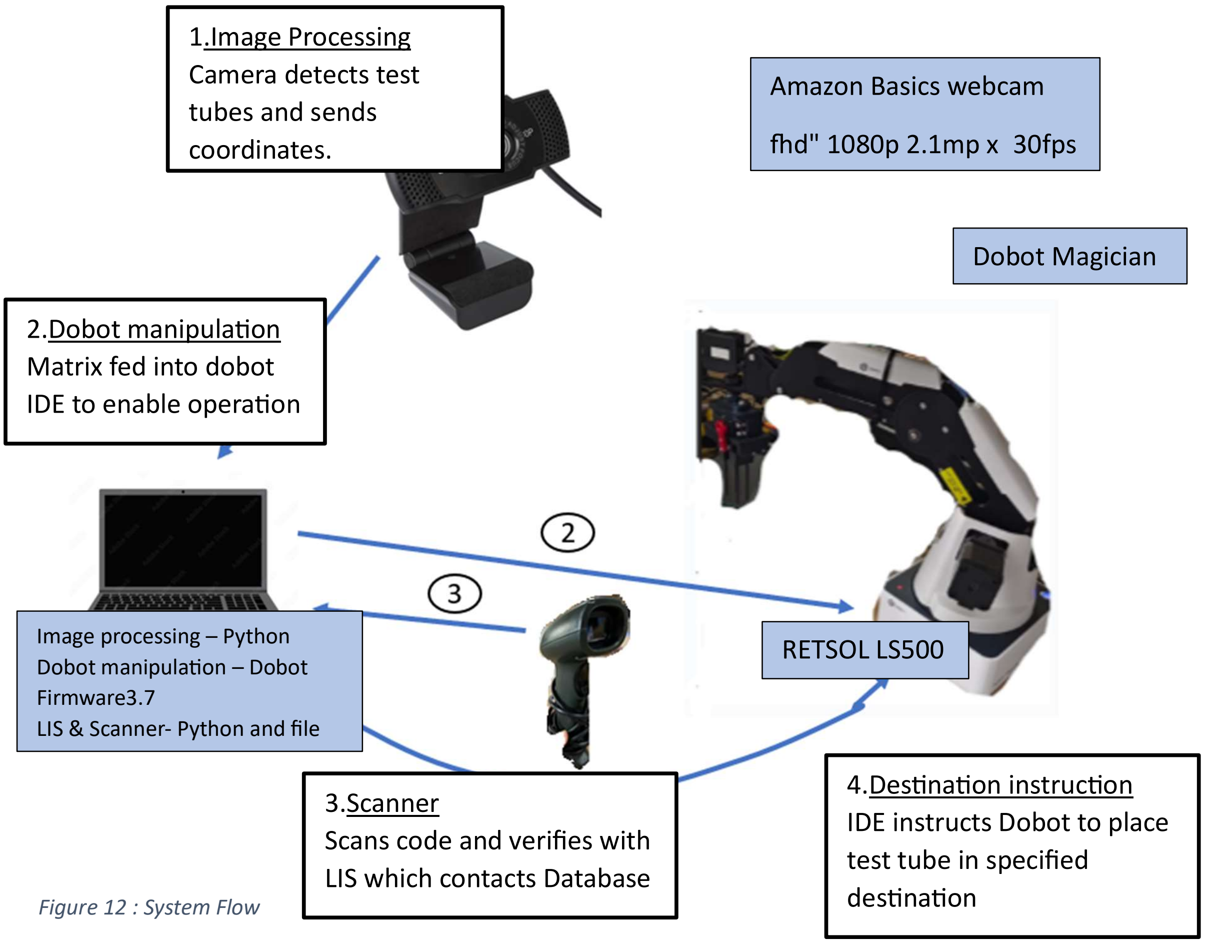

1.Image Processing
Camera detects test tubes and sends coordinates.
Amazon Basics webcam
fhd" 1080p 2.1mp x 30fps
Dobot Magician
2.Dobot manipulation
Matrix fed into dobot IDE to enable operation
2
3
Image processing – Python
Dobot manipulation – Dobot Firmware3.7
LIS & Scanner- Python and file
RETSOL LS500
3.Scanner
Scans code and verifies with LIS which contacts Database
4.Destination instruction
IDE instructs Dobot to place test tube in specified destination

Figure 12 : System Flow

3. Conclusion

Conclusion

"Successfully automated the process of archiving of test tube samples using image processing and robotic arm and successfully integrated the two. The project can help to serve for industrial purposes in diagnostics."

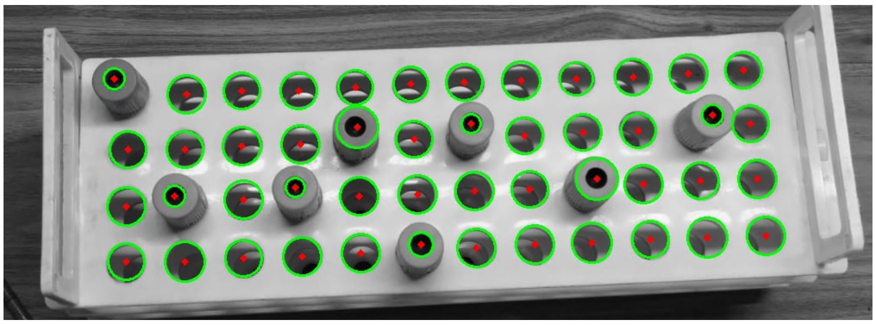


Figure 13 : Border Detection

Furthermore, test tubes are even mapped solely based on their color(grey and purple being the two colors)

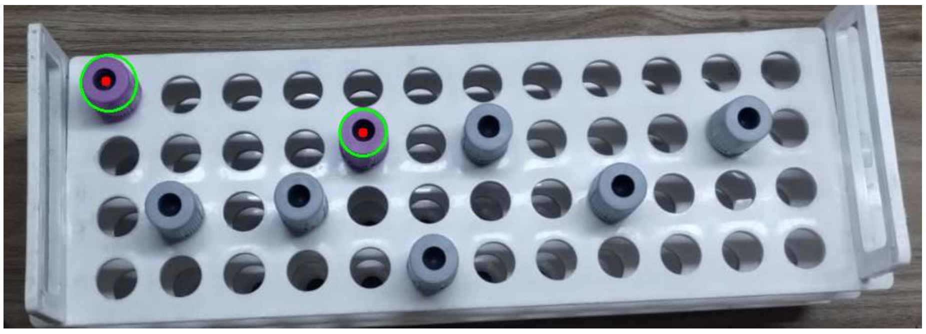


Figure 14 : Colour Classification

Integrated system comprising Dobot Magician, Web Camera, and scanner modules

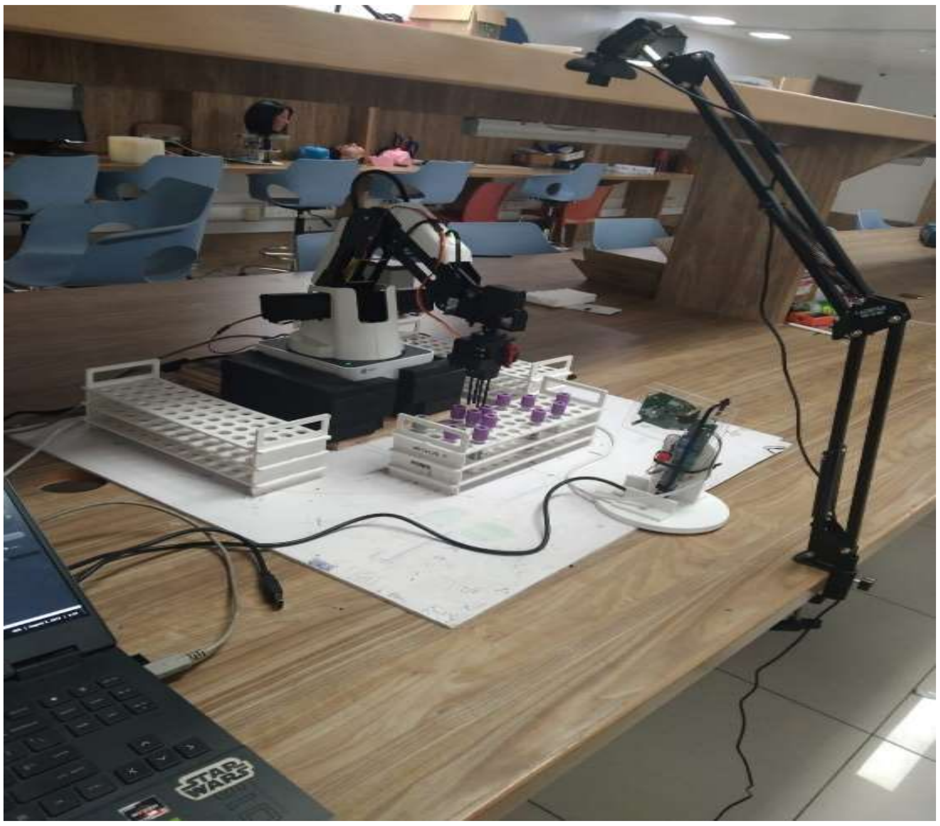


Figure 15 : Integrated System

4. References

1. https://blog.roboflow.com/introducing-bounding-box-level-augmentations
2. https://www.dobot-robots.com/service/download-center
3. https://github.com/luismesas/pydobot
4. https://forum.dobot.cc/t/importing-python-modules-to-dobotstudio-script-editor/136
5. https://github.com/SERLatBTH/StarterGuide-Dobot-Magician-with-python/blob/master/README.md
6. https://electronicsworkshops.com/2020/08/10/object-detection-image-processing/
7. https://www.researchgate.net/figure/Simulation-of-DOBOT-manipulator-in-Matlab-environment_fig7_321954826
8. https://ieeexplore.ieee.org/document/8337386
9. chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://dc.uwm.edu/cgi/viewcontent.cgi?article=3257&context=etd