Winter Internship

On

Industry 4.0 Automation and Robotics Implementation

Submitted by

Aditya Roy

Under the guidance of

Prof. Jayendran Venkateswaran

Industrial Engineering and Operations Research (IEOR)

IIT Bombay

February 6, 2026

Acknowledgment

I would like to express my sincere gratitude to all those who contributed to making this internship a meaningful and enriching experience.

I am deeply thankful to the Indian Institute of Technology Bombay (IIT Bombay) and the Department of Industrial Engineering and Operations Research (IEOR) for providing an exceptional research environment, along with the infrastructure and resources of the IEOR Learning Factory, which were essential for the successful execution of this project.

I am profoundly grateful to Prof. Jayendran Venkateswaran (JV Sir) for his invaluable guidance, mentorship, and constant encouragement throughout the course of this work. His vision and technical insights formed the foundation of the Industry 4.0 systems developed during the internship, including the conceptualization and implementation of the Dobot Express framework.

I would also like to extend my sincere appreciation to Mr. Sumanto (Sumanto Sir) for his continuous support, insightful technical discussions, and constructive feedback, which significantly contributed to the depth and quality of this work.

Finally, I gratefully acknowledge the FOSSEE (Free and Open Source Software for Education) project, under which this internship was undertaken. I am thankful to FOSSEE for providing the opportunity of employment during this internship, and for fostering an open-source–driven, collaborative learning environment that closely aligned with the philosophy and objectives of the custom software and automation systems developed as part of this project.

Contents

Acknowledgment **1**

1 Introduction **4**

2 Microcontrollers **6**

3 Dobot Magician **7**

3.1 Dobot Studio 9

3.2 Arduino and Python 10

3.3 pydobot - A Dobot Python Library 10

3.4 Projects with Dobot 11

4 Developing Dobot Express **13**

4.1 Overview 13

4.2 Architecture and Benefits 14

5 Industry 4.0 Demo **15**

5.1 System Overview 15

5.2 Conceptual Workflow 16

5.3 Key Subsystems 16

5.3.1 Physical Construction & Sensors 16

5.3.2 Importance of Calibration 16

5.3.3 Cloud-to-Factory Integration 16

6 Gesture Control Dobot **17**

6.1 System Architecture Overview 17

6.2 Pose Estimation and Mapping 17

6.3 Safety and Stability 18

6.4 Multithreaded Architecture 18

7 Conclusion and Future Scope **19**

Bibliography **20**

Chapter 1

Introduction

During the tenure of this internship, I had the privilege of working in the Department of Industrial Engineering and Operations Research (IEOR) at the Indian Institute of Technology Bombay, under the guidance of Prof. Jayendran Venkateswaran. The IEOR Learning Factory, where the internship was conducted, is an evolving laboratory initiative aimed at fostering research, experimentation, and hands-on learning in the domains of industrial automation, smart manufacturing, and the practical implementation of Industry 4.0 principles. The lab is designed to bridge the gap between theoretical concepts taught in classrooms and their real-world industrial applications, thereby providing students and researchers with a realistic and immersive manufacturing environment.

Industry 4.0 represents the fourth major industrial revolution, following mechanization, mass production, and automation. Unlike previous revolutions, Industry 4.0 is characterized by the seamless fusion of the physical and digital worlds. Traditional manufacturing systems, which were largely linear, rigid, and manually supervised, are now being transformed into intelligent, autonomous, and highly interconnected systems. This transformation is driven by the deep integration of advanced digital technologies, smart automation techniques, and real-time data intelligence across all levels of industrial operations.

At the core of Industry 4.0 lie systems that possess the ability to sense, communicate, analyze, and act autonomously or semi-autonomously. These capabilities are enabled through a combination of enabling technologies such as the Internet of Things (IoT), cyber-physical systems (CPS), artificial intelligence (AI), machine learning (ML), cloud and edge computing, industrial robotics, digital twins, big data analytics, and next-generation communication technologies such as 5G. Together, these technologies allow physical machines and processes to be digitally represented, monitored, and controlled in real time, thereby creating a tightly coupled cyber-physical manufacturing ecosystem.

In a fully realized Industry 4.0 environment, machines, sensors, production lines, and supply chain components become "smart" and interconnected. Continuous

streams of data are generated from sensors embedded within machines and infrastructure, enabling factories to monitor operational performance at a granular level. This data-driven approach facilitates predictive maintenance, where potential failures can be detected and addressed before they result in costly downtime. Furthermore, intelligent control systems can dynamically optimize energy consumption, material usage, and production schedules, leading to significant improvements in efficiency, sustainability, and overall productivity. Automated inspection systems, powered by computer vision and machine learning, enhance product quality by identifying defects with high accuracy and consistency.

Another defining feature of Industry 4.0 is its ability to support mass customization. Unlike traditional mass production systems that prioritize uniformity and scale, Industry 4.0 systems are capable of producing personalized products at scale without sacrificing efficiency. This is achieved through flexible production lines, modular system design, and rapid reconfiguration of manufacturing processes based on real-time customer demand. As a result, industries can respond swiftly to changing market requirements while maintaining cost-effectiveness and operational agility.

Beyond the confines of the factory floor, Industry 4.0 extends into logistics, warehousing, supply chain management, and strategic business decision-making. The integration of digital technologies across the entire value chain enables end-to-end visibility, from raw material procurement to final product delivery. Such transparency allows organizations to make informed, data-driven decisions, improve coordination across departments, and enhance overall system resilience. However, this transformation also introduces new challenges, including concerns related to cybersecurity, data privacy, workforce upskilling, system interoperability, and the standardization of communication protocols across diverse devices and platforms.

The primary focus of the IEOR Learning Factory during this internship was divided into two major components. The first component involved the development of a dedicated B.Tech laboratory setup designed to support undergraduate education through hands-on exposure to automation and Industry 4.0 concepts. The second component focused on establishing a research-oriented experimental environment to explore the capabilities, limitations, and integration of various robotic and automation systems. This research setup served as a testbed for experimenting with real-world industrial scenarios, system integration challenges, and innovative automation workflows.

Over the course of this one-month internship, I gained extensive exposure to both hardware and software aspects of modern industrial systems, as well as the complex interactions between them. The experience involved working with microcontrollers, sensors, robotic arms, control software, and communication protocols, all of which had to function cohesively as part of a larger system. The internship offered a comprehensive blend of technologies unified by a single objective: the design and implementation of reliable, repeatable, and scalable components capable of performing Industry 4.0-oriented tasks.

Chapter 2

Microcontrollers

Microcontrollers are small, self-contained computers built onto a single chip, designed to control specific tasks in electronic devices. Unlike a regular computer CPU that needs external components to work, a microcontroller usually includes the processor (CPU), memory (RAM and Flash/ROM), and input/output peripherals like timers, ADC (Analog-to-Digital Converter), PWM (Pulse Width Modulation), communication modules (UART, I2C, SPI, CAN), and GPIO pins—all in one compact package.

They are widely used in embedded systems to read sensor data, make decisions based on programmed logic, and control actuators like motors, LEDs, relays, and displays. Microcontrollers are optimized for low power consumption, real-time operation, and reliability, which makes them perfect for applications such as home appliances, smartwatches, cars, drones, robots, IoT devices, medical instruments, and industrial automation. Popular microcontroller families include Arduino boards (based on AVR), ESP8266/ESP32 for Wi-Fi and IoT, STM32 for high-performance control, and PIC microcontrollers for general embedded use. In simple terms, a microcontroller is the "brain" inside many everyday electronic products, quietly running the code that makes them function automatically.

Arduino is one of the most famous microcontrollers not only in the teaching industry but also in daily usage. You will be surprised to discover that the Smart Led bulbs are based on the ESP8266, which is an extremely cheap but very profitable WIFI chip module. Microcontrollers were the base of our lab and from day one, we had extensively used Arduino to power our projects.

Our first project was to debug and assemble a robotic arm which involved extensive use of servos and Arduino. We had to debug the code, figure out servo functions and a lot more. This served as a starting point for our journey in the IEOR lab. We also had robotic arms which were 3D Printed and being assembled, coded and made functional for the B.Tech students.

Chapter 3

Dobot Magician

The Dobot Magician is an industry-grade robotic arm that was brought into the lab primarily for research purposes, and it quickly proved to be one of the most interesting as well as complex pieces of hardware one can come across in a robotics environment. What makes the Dobot Magician stand out is not just its ability to perform precise movements and execute automation tasks, but also the depth of learning it offers to anyone who gets the opportunity to operate and understand it.

From basic control and calibration to advanced motion planning and real-world integration, every interaction with the Dobot Magician introduces new concepts and challenges that strengthen both technical knowledge and problem-solving skills. In many ways, it acts as a complete learning platform—bridging mechanical design, electronics, embedded systems, and software control into one unified experience. The entire journey of our internship revolves around the Dobot Magician, as it became the central hardware system through which we explored robotics in a hands-on and practical manner.

We began working with the Dobot Magician from Week 2, starting with understanding its structure, capabilities, and operating methods, and gradually moving toward deeper experimentation. Over time, our curiosity and motivation pushed us beyond simply using it as a tool, eventually leading us into reverse engineering its functioning, customizing its behavior, and exploring how it could be adapted for different applications. This progression allowed us to develop a stronger grasp of robotics systems as a whole, while also encouraging creativity and innovation. As the internship advanced, the Dobot Magician became the foundation for multiple exciting projects, helping us build impressive demonstrations and displays that reflected both our learning and our ability to apply engineering concepts in a real-world research setting.

Figure 3.1: Dobot Magician Arm (4 Joints)

3.1 Dobot Studio

The Dobot Magician was shipped with its own custom software called Dobot Studio. The Dobot Magician had the following features inbuilt, also to note, these features also defined the limitations of the hardware itself, and the extent we can push it for our own purposes.

- **Teaching and Playback**: A feature which basically replicates the last recoded coordinates to perform repetitive tasks.
- **Write & Draw**: This is one of the most interesting features which actually allows us to write anything with the Dobot Arm.
- **Blockly**: The Dobot can be controlled in 4 ways: Blockly, a very simplified block coding interface; Buttons given in Dobot Studio; Scripting: Writing python code inside the DobotStudio and lastly secondary development, which means to use python outside the DobotStudio environment.
- **LeapMotion**: Allows you to control the Dobot using a 3D animated model given with the DobotStudio.
- **Mouse**: Controlling the Dobot Arm using Mouse.
- **Laser Engraving**: A laser kit is specially provided with the Dobot Magician, to perform laser engravings.
- **3D Printing**: A 3D Printed Kit is also supplied.

Our main goal throughout the internship was to move beyond the limitations of DobotStudio and build a more flexible and powerful system by integrating our own custom sensors, microcontrollers, and external modules, ultimately creating demos where every component works together in perfect synchronization. DobotStudio is definitely a great starting point for beginners, as it provides an easy interface to understand the Dobot Magician's basic movements, coordinate system, and built-in operation modes.

However, for the tasks we aimed to perform, we required deeper control over the robot's behavior, the ability to connect and respond to real-time sensor inputs, and the freedom to design our own logic and decision-making systems. Many of our ideas demanded tight integration between the robotic arm and external hardware such as proximity sensors, RFID/NFC modules, cameras, and microcontrollers like Arduino, ESP32, or other embedded platforms. These components needed to communicate seamlessly, triggering the robot's movements dynamically based on sensor feedback and system conditions. In such scenarios, relying only on DobotStudio becomes restrictive.

3.2 Arduino and Python

One of the main hurdles of this internship was the fact that Arduino worked with Embedded C but for the complex tasks we needed to perform, we had to think of some way to move beyond the embedded C and access Arduino through other languages. Micropython was something we thought of initially but then it also didn't support all the standard python functions.

It is important to clarify that the "interpreter" we are referring to is not a full-fledged Python interpreter running directly on the Arduino. Instead, it works by listening for serial commands that follow a specific format and structure. In this setup, the Arduino acts more like an execution unit that waits for instructions, while the personal computer runs the actual Python program and sends commands over the serial interface. The computer then transmits these well-defined serial commands to control the Arduino's pins, sensors, and outputs, allowing the Arduino to perform actions based on what the Python script requests.

We came across this concept through an open-source GitHub library called **Python-Arduino-Command-API** (by Tristan Hearn), which served as a strong reference point for our implementation and experimentation. The library provides a clean and structured way to communicate with an Arduino board using Python, making it easier to build hardware-software systems that require real-time control and integration.

3.3 pydobot - A Dobot Python Library

In our efforts to control the Dobot Magician using Python outside of DobotStudio, we started exploring different third-party libraries that could help us establish direct programmatic control over the robotic arm. During this search, we came across one of the first and most commonly referenced libraries called **pydobot**. Over time, we also discovered that an improved successor to this library exists, known as **pydobotplus**.

To understand why these libraries are important and how they actually work, it is necessary to understand how the Dobot Magician is designed to operate in the first place. In reality, the Dobot Magician is primarily meant to be used with **DobotStudio**. The core abilities of the Dobot Magician—such as motion control, kinematics, command execution, and device communication—are implemented inside the robot's official compiled library, **DobotDll.dll**.

Libraries like **pydobot** and **pydobotplus** attempt to recreate this same functionality in Python. They do not provide control by inventing a new communication protocol from scratch, but rather by leveraging the existing command structure that DobotStudio already uses. This also brings us to the reality that there is a huge

limitation of being able to expand these libraries which will cause us to think of alternative options.

3.4 Projects with Dobot

We were systematically introduced to the Dobot Magician through a structured learning approach that emphasized building a strong foundation before progressing toward more complex and advanced robotic tasks. This pedagogical strategy ensured that fundamental concepts such as coordinate systems, motion planning, and end-effector control were thoroughly understood prior to implementing higher-level automation workflows. Each project was designed to incrementally increase in complexity, allowing us to develop confidence and technical proficiency while minimizing operational errors and safety risks.

The Dobot Magician supports a wide range of interchangeable attachments, commonly referred to as end-effectors, which significantly enhance its versatility and applicability in industrial automation scenarios. In our laboratory setup, we had access to multiple end-effectors, including a mechanical gripper for handling solid objects, a suction cup module for lightweight pick-and-place tasks, a laser module for engraving operations, and a 3D printer module for additive manufacturing demonstrations. The availability of these end-effectors enabled us to explore diverse use cases and gain insight into how a single robotic platform can be adapted for multiple industrial functions.

The first task performed using the Dobot Magician was a basic pick-and-place operation, which serves as a foundational exercise in robotics. To execute this task successfully, it was necessary to manually hard-code the Cartesian coordinates corresponding to both the pick location and the drop location. This process required careful calibration of the robot's workspace, precise measurement of object positions, and an understanding of the robot's coordinate reference frame. The motion type employed for this operation was Point-to-Point (PTP) movement, in which the robot transitions between predefined spatial coordinates in a controlled and repeatable manner.

The Dobot Magician provides multiple motion planning strategies within PTP movement, each suited to different operational requirements. Linear movement enables the end-effector to follow a straight-line trajectory in Cartesian space, which is particularly useful for tasks requiring consistent tool orientation or precise path control. Jump movement introduces a vertical lift before horizontal translation, reducing the risk of collisions with surrounding objects or fixtures, making it suitable for cluttered workspaces. Joint movement, on the other hand, prioritizes efficient joint rotation, allowing the robot to reach the target position through the most mechanically optimal configuration, often resulting in faster execution times.

Once basic pick-and-place operations were mastered, the tasks were extended

to include stacking operations, where multiple objects had to be placed accurately on top of one another. Stacking introduced additional challenges such as maintaining positional accuracy across repeated cycles, compensating for cumulative height offsets, and ensuring stable placement to prevent toppling. These exercises highlighted the importance of precision, repeatability, and error minimization in robotic manipulation tasks.

Further complexity was introduced through the integration of color detection mechanisms using external sensors. In these experiments, objects of different colors were identified prior to manipulation, and the robot's actions were dynamically adjusted based on sensor input. This marked a transition from static, pre-programmed motion sequences to semi-autonomous, decision-based behavior. The robot was required to interpret sensor data, determine the appropriate pick or drop location, and execute the corresponding motion sequence. Such tasks closely resemble real-world industrial sorting and quality control applications, where robotic systems must respond intelligently to variations in input conditions.

Through these progressively advanced projects, we moved from simple coordinate-based control to more intelligent and adaptive automation workflows. The integration of stacking, color detection, and conditional logic demonstrated how robotic systems can be transformed from basic mechanical manipulators into responsive cyber-physical systems. These experiments not only strengthened our understanding of robotic kinematics and motion planning but also provided practical exposure to sensor integration, system calibration, and real-time decision-making—key elements of modern Industry 4.0 manufacturing environments.

- **Linear Movement**: The Dobot tries to move along a straight path in Cartesian space.
- **Jump Movement**: The robot lifts upward first and then moves toward the target position before lowering down again.
- **Joint Movement**: The robot focuses on moving the joints efficiently to reach the target position.

Chapter 4

Developing Dobot Express

4.1 Overview

As we continued to push forward with increasingly complex experiments and real-world robotic applications, it became evident that the existing libraries available for controlling the Dobot robotic arm were not designed with deep experimentation and fine-grained control in mind. While the official SDKs and community-built wrappers were useful for basic motion commands, pick-and-place demos, and introductory robotics workflows, they often hid crucial low-level details behind high-level abstractions.

To overcome these constraints, we set out to build **Dobot Express**, a Python library focused on flexibility, control, and clarity. Dobot Express is built directly on top of DobotDll.dll, reusing Dobot's official compiled C library rather than attempting to reverse-engineer or replace it. By interfacing directly with the C-based DLL, we were able to expose a much larger portion of the Dobot API to Python in a controlled and predictable manner.

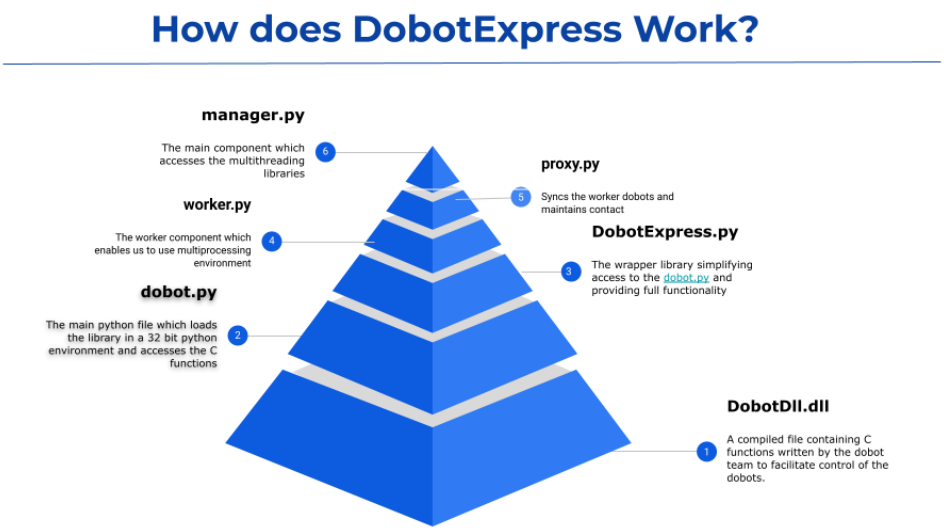

Figure 4.1: DOBOT Express Architecture

4.2 Architecture and Benefits

Developing Dobot Express was a challenging and deeply educational experience. It required a strong understanding of foreign function interfaces, careful handling of C data structures in Python, and precise management of memory and pointers to avoid instability or crashes.

For **researchers**, Dobot Express provides a transparent and programmable control layer that enables experimentation with motion planning, kinematics, trajectory optimization, feedback control, and human–robot interaction.

For **students and educators**, Dobot Express serves as a powerful learning platform that bridges theory and practice. Concepts such as coordinate systems, inverse kinematics, timing synchronization, and control loops can be explored hands-on using a real robotic arm.

Crucially, Dobot Express strikes a deliberate balance between **ease of use and low-level access**. Users are not forced to choose between a closed, beginner-only interface and a complex, hardware-centric control stack.

Chapter 5

Industry 4.0 Demo

This project demonstrates a miniaturized Industry 4.0 manufacturing workflow designed and deployed in the IEOR Laboratory. The setup integrates robotic arms, sensors, microcontrollers, real-time cloud communication, and material handling systems to simulate a modern smart factory environment. The objective of this demonstration is to showcase how customer-driven digital orders can be transformed into physical manufacturing actions through a tightly coordinated cyber-physical system.

5.1 System Overview

The complete setup consists of multiple independent components, each with its own configuration and calibration requirements:

1. Two robotic arms (Picker and Writer)
2. Linear rail system
3. Storage module with gravity-fed slots
4. Color sensing subsystem
5. Microcontroller-based control layer
6. Servo-actuated gripper platform
7. Conveyor-based logistics unit
8. Cloud-connected ordering application

5.2 Conceptual Workflow

The logical flow of the system is as follows:

1. A customer places an order via a mobile or web application.
2. The order specifies a **product configuration**, primarily identified by color.
3. The system determines which storage slot contains the requested item.
4. The **Picker robot**, mounted on a linear rail, retrieves the correct block.
5. The block is placed onto a **central gripper workstation**.
6. The **Writer robot** performs a marking or writing operation on the block.
7. The Picker robot retrieves the processed block.
8. The block is deposited onto a conveyor system for logistics or dispatch.

5.3 Key Subsystems

5.3.1 Physical Construction & Sensors

The storage area is a custom-built structure constructed using cardboard, designed to function as a gravity-fed slot mechanism. Two color sensors are mounted at the rear of the storage unit to identify block color and availability.

5.3.2 Importance of Calibration

The key lesson demonstrated by this setup is that **robotic automation is highly sensitive to physical placement**. Because joint angles, rail positions, and end-effector paths are pre-calibrated, even small positional changes can cause collisions or missed picks.

5.3.3 Cloud-to-Factory Integration

One of the most significant aspects of this demo is the **real-time connection between cloud data and physical automation**. Orders are placed digitally via an application, and the factory reacts instantly. This architecture reflects modern **smart manufacturing systems**, where ERP, MES, and shop-floor automation are tightly coupled.

Chapter 6

Gesture Control Dobot

One of the most compelling and intuitive experiments conducted in the IEOR laboratory involved controlling a robotic arm using human gestures. Unlike traditional robotic control methods that rely on joysticks, teach pendants, or predefined scripts, this experiment explored direct human–robot interaction (HRI) by mapping human body movements to robotic motion in real time.

6.1 System Architecture Overview

The system is composed of four major layers:

1. Vision Input Layer
2. Pose Estimation Layer
3. Motion Mapping & Filtering Layer
4. Robot Control Layer

6.2 Pose Estimation and Mapping

The captured video stream is processed using a **human pose estimation model**, which identifies key anatomical landmarks of the human body. The system focuses specifically on the **Right wrist position** and **Left wrist position**.

The experiment uses a **Cartesian mapping approach**:

- Right hand vertical motion → Dobot forward/backward movement

- Right hand horizontal motion → Dobot left/right movement
- Left hand vertical motion → Dobot up/down movement

6.3 Safety and Stability

One of the most critical aspects of this experiment is **safety**. The robot's movement is constrained within predefined Cartesian boundaries to prevent overextension or collisions.

To mitigate jitter from human gestures, the system incorporates a combination of **temporal smoothing** and **deadband filtering**. Temporal smoothing applies gradual interpolation between successive target positions, while deadband filtering ignores minor, unintentional hand movements.

6.4 Multithreaded Architecture

To ensure real-time performance, the system separates vision processing and robot command execution into different threads. This prevents camera lag from affecting robot motion and ensures consistent command timing.

Chapter 7

Conclusion and Future Scope

This internship provided an immersive and comprehensive practical experience at the intersection of robotics, industrial automation, and the core principles of Industry 4.0. Working within the IEOR Learning Factory, the focus shifted rapidly from theoretical understanding to hands-on system integration.

A significant outcome was the realization of the limitations inherent in off-the-shelf control software like DobotStudio, which necessitated the development of custom solutions. This led to the creation of **Dobot Express**, a dedicated Python library built on the official DLL, providing the fine-grained, low-level control essential for serious research and complex integration.

The knowledge gained was successfully demonstrated in two major projects: a fully operational **Industry 4.0 Mini-Factory Demo** and a cutting-edge **Gesture Control System**. In summary, this tenure transformed abstract engineering concepts into tangible, functional systems. It reinforced the criticality of seamless hardware-software integration, the power of custom programmatic control, and the fundamental role of reliability and robustness in deploying real-world robotic solutions.

Bibliography

[1] sammydick22, "pydobotplus," GitHub Repository. Available: https://github.com/sammydick22/pydobotplus/tree/master/pydobotplus

[2] L. Mesas, "pydobot," GitHub Repository. Available: https://github.com/luismesas/pydobot

[3] T. Hearn, "Python-Arduino-Command-API," GitHub Repository. [Online]. Available: https://github.com/thearn/Python-Arduino-Command-API

[4] Dobot, "Dobot Magician - Desktop Intelligent Training Robot," Official Product Page. [Online]. Available: https://www.dobot-robots.com/products/education/magician.html

[5] Dobot Express, "dobotexpress 0.1.0," Python Package Index. Available: https://pypi.org/project/dobotexpress/0.1.0/