

**A REPORT**  
**ON**  
**GRAPHICAL USER INTERFACE (GUI) OF LOCAL CONTROL PANEL**  
**FOR 3-AXIS GANTRY ROBOT**

**BY**

Abhishek Malav

2019ABPS0916P

**AT**

(Shalaka Connected Devices LLP, Pune)

A Practice School-I station of

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**(July, 2021)**

**A REPORT**  
**ON**  
**GRAPHICAL USER INTERFACE (GUI) OF LOCAL CONTROL PANEL**  
**FOR 3-AXIS GANTRY ROBOT**

**BY**

Abhishek Malav

2019ABPS0916P

Manufacturing Engineering

Prepared in completion of the  
Practice School-I Course Nos.  
BITS C221/BITSC231/BITSC241

**AT**

(Shalaka Connected Devices LLP, Pune)

A Practice School-I station of

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**(July, 2021)**

# Acknowledgements

Mr. Hemanth Kamat sir, CTO of Shalaka Connected Devices, is my industry mentor, and I appreciate his mentorship and unwavering support for the initiative. He conducted frequent meetings and thoroughly discussed the foundations and project needs. I'd like to express my gratitude to my teammates Agam, Ravi, Ashwin, and Saurabh for their help and contributions. Last but not least, I'd like to express my gratitude to my professor in charge, C P Kiran sir.

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE PILANI  
(RAJASTHAN)  
Practice School Division**

**Station:** Shalaka Connected Devices

**Centre:** Pune

**Duration:** 2 months

**Date of start:** 03/06/2021

**Date of Submission:** 21/07/2021

**Title of the Project:** Graphical User Interface (GUI) of local control panel for 3-axis Gantry Robot

**2019ABPS0916P**

**Abhishek Malav**

**Manufacturing Engineering**

**Mr. Hemanth Kamat**

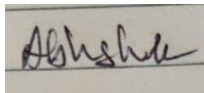
**Chief Technical Officer (CTO) of Shalaka Connected Devices**

**Prof. C P Kiran**

**Key words:** MQTT, GUI, 3-axis Gantry Robot, Tkinter, Edge Computer

**Project areas:** Embedded systems, IIOT, Connected devices

**Abstract:** This report summarises the primary outcomes and lessons learned thus far in the Practice School-1 (PS1) course. It also gives an outline of our work at Shalaka, a PS1 station of BITS Pilani, so far, as well as our aims for the next weeks.



Signature of Student

Date: 21/07/2021

Signature of PS Faculty

Date: 21/07/2021

# Table of Contents

1	Introduction.....	1
2	Overview.....	2
3	Requirements and Skills	
3.1	Technical Requirements	
3.2	Skills	
4	3-axis Gantry Robot.....	3
5	Python (Tkinter library based) GUI for Control Panel.....	5
6	Project Design.....	7
6.1	Objective.....	7
6.2	High Level Design.....	7
6.3	Data Structure.....	8
6.4	API Functions .....	8
6.4.1	setSpeedX(speedX).....	9
6.4.2	setSpeedY(speedY).....	10
6.4.3	setSpeedZ(speedZ).....	11
6.4.4	xMotorOn(speedX).....	12
6.4.5	xMotorOff().....	9
6.4.6	yMotorOn(speedY).....	
6.4.7	yMotorOff().....	
6.4.8	zMotorOn(speedZ).....	
6.4.9	zMotorOff().....	9
6.4.10	home().....	
6.4.11	update_label().....	
6.4.12	moveTo(x,y).....	
6.4.13	moveBy(x_sign,y_sign,x,y).....	9
6.4.14	pick().....	
6.4.15	place().....	

6.5	Control Panel Functionality	
6.5.1	Input Entry Boxes	
6.5.2	Display Panel	
6.5.3	Buttons	
6.5.3.1	Home	
6.5.3.2	Move To X,Y	
6.5.3.3	+ and -	
6.5.3.4	Pick/Place	
6.5.4	Core Logic	
6.6	Control Flow	
7	Project Timeline and Learning Outcomes.....	10
7.1	Schedule	
7.2	Insights Gained	
8	Scope of improvement	
9	Conclusion.....	
10	References.....	12

# 1 Introduction

Shalaka Connected Devices, an Embedded Systems and Internet of Things firm, is one of India's top electrical designers and manufacturers. It is also one of the first firms to accept and operate in the new hot sector of Industry 4.0, which originated in Germany and is the fourth generation of the industrial revolution. To grasp the significance of Industrie 4.0, we must first comprehend its past. The early 1800s saw the commencement of the industrial revolution, which brought with it a slew of changes. Manufacturing operations were mechanised with the use of water and steam power. This cleared the door for large production and assembly lines utilising electricity within a few years. The second generation, or second industrial revolution, was born out of this. Around the 1950s, the third generation arrived with the development of computers and automation. Finally, we now have Industrie 4.0, or the fourth generation, which uses IoT and machine learning to help existing systems become smarter and more autonomous. Our team is required to build a python based GUI for controlling the 3-axis gantry robot by user commands.

## 2 Overview

Here is the gist of the project that was assigned to us. Creating a python-based GUI to control gantry robots that accepts commands from users. We plan to make it using the Tkinter library.

The skill set required are :-

- Python programming is utilised to construct the Control panel and run desktop simulations on the edge computer.
- The data from the sensors, which is generally in.json format, will be received using object-oriented programming concepts .
- MQTT/COAP and other IoT protocols are used for communication between the physical factory and the cloud. We'll be utilising the MQTT protocol in our scenario, which will need the usage of Mosquitto, a MQTT broker.



## **3 Requirements and Skills**

### **3.1 Technical Requirements**

The developer's system must have:

1. Python (3.6 and above)
2. Integrated Development Environment (IDE), preferably vscode
3. Tkinter library installed (via pip command)

### **3.2 Skills**

Proficiency in python programming along with a clear understanding of python fundamentals is required. Familiarity with the IDE is needed for debugging and testing purposes. Additionally, knowledge of the tkinter library and its functions is required.

## 4 3-axis Gantry Robot

A Gantry Robot is an automated industrial system that can also be referred to as a Cartesian Robot or a Linear Robot. Gantry robots typically move in linear paths creating a three-dimensional cubic envelope of space it can work within. For this reason, the term Cartesian robot is ideal since the overall top-down working space is bound by two linear paths that intersect at right angles to create the working plane (typically referred to as X and Y.) When you add a third axis (typically referred to as Z), this creates the depth at which the end-of-arm-tool can operate inside the cubical envelope. While these three axes are the principal forms of movement, gantry robots may contain as few or as many axes as necessary to perform the required task.

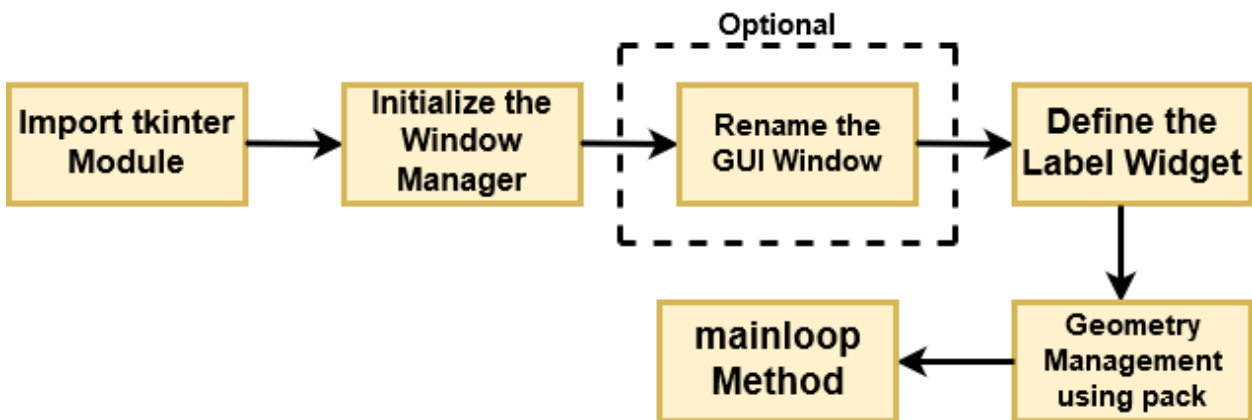


Gantry robot systems provide the advantage of large work areas and better positioning accuracy. Position accuracy is the ability of the robot to place a part correctly. Gantry robots are easier to program, with respect to motion, because they work with an X, Y, Z coordinate system. Another advantage is that they are less limited by floor space constraints. Because of their rigid lightweight structure, the Cartesian/Gantry Robots are very accurate and repeatable. Due to their simple structure, gantry robots are intuitive to program and easy to visualize when evaluating

new automation. Most of all Gantry robots are configurable. From a plethora of motor and gearbox choices to components and materials these robots are prepared to take on the challenges of damp hazardous and dirty environments. The Cartesian Coordinate Robot's relatively simple design and straightforward operation make it highly desirable in manufacturing. Because the individual axes can be easily replaced, downtime is reduced and maintenance costs are kept to a minimum. In addition, the entire system can be disassembled into its component parts for use in multiple single-axis applications. Most importantly, Cartesian Coordinate Robot systems are inexpensive compared to other more complex robots. Gantry robot's motion is generally controlled using stepper motors<sup>10/16</sup> Some variables/sensors used are motors/drivers, voltage & current supply, temperature, vibration, linear speed, direction, etc Some other companies that manufacture these robots are KUKA, ABB, Toshiba Machine Major companies in the gantry robot market are Toshiba Machine (Japan), Bosch Rexroth (Germany), Yamaha Motor (Japan), Macron Dynamics (US), IAI (Japan), Cimcorp (Finland), OMRON (Japan), Nordson (US).

## 5 Python (Tkinter library based) GUI for Control Panel

GUI platforms are user-friendly and efficient software programmes for completing any activity. Instead of instructions, they can operate an IOT device via graphicbased user inputs in an industry. There are several IDEs for developing a GUI, such Java-based NetBeans, C#- based VisualBasic, and so on. PyQt5, Tkinter, and PySimpleGUI are some of the Python libraries that may be used to construct a Graphical User Interface. Because of its simple and layered design, we will utilise the Tkinter library in our project. Importing the library, initialising the Window manager, renaming the window, adding labels, textboxes, buttons, checkboxes, radio buttons, and specifying their responsibilities are all part of Tkinter's tiered approach to creating a GUI. Use methods like pack() and put() to handle geometry. Use the grid() mainloop() function as an endless loop to keep the window open.



We'll need to establish some variables to receive and deliver signals from/to the IoT devices/sensors when building the GUI interface. These variables will also contain the variables that the user will provide to the gantry system in order for it to carry out its functions. These variables might include the gantry pin's coordinates, velocity, and the task (pick/drop) to be completed. Sensor inputs such as temperature sensors, IMU sensors (for coordinate data), and so on can be used as input variables.

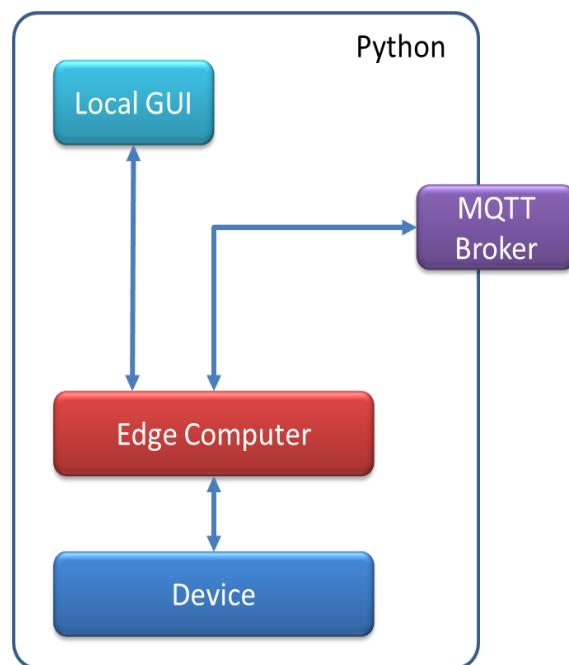
## 6 Project Design

### 6.1 Objective

The objective of the project is to create a functional model of the Gantry robot that can be used in a virtual embedded system to create a virtual device node. The model of the device in Python consists of a file containing the set of API functions that can be used by an application to set, get and reset values from the internal registers of the device. The second file consists of the front end GUI code, which places all the appropriate labels and their corresponding text box entries to take inputs from the user and display the current status of the Gantry robot.

### 6.2 High level Design

The architecture of the entire project undertaken by all BITS students at Shalaka is shown below:



Separate teams will be working on the Local GUI and Edge computer simulations. This report concerns the Local GUI team, whose task was to develop a complete and functional control panel using Python to successfully operate a 3-axis Gantry Robot.

## 6.3 Data Structure

The API functions are used to access Data Registers. These registers hold the data generated by the device or the data required by certain control operations performed by the device. In this particular case, the Data register is a Python Dictionary, which is a python data structure storing data in the form of keys and their corresponding values. The keys are the data variables that describe the complete state of the Gantry robot. These data variables are:

- Current coordinates
- Next coordinates
- Speed of the 3 axes motors
- Statuses of the 3 axes motors (ON/OFF)
- Status of the gripper (holding an object/free)

An example of a data register is shown below.

```
# Define a dictionary called register where each element contains a list of x,y,z values defining the state. MotorStatus contains a list of booleans to indicate x,y,z motor statuses. gripperStatus defines status of gripper using a string (default:free, pick, place)
register = {'current': [0, 0, 0], 'next': [0, 0, 0], 'speed': [0, 0, 0], 'motorStatus': [xMotor, yMotor, zMotor], 'gripperStatus': [False]}
```

## 6.4 API Functions

API functions (Application Programming Interface functions) are used to access and manipulate the data register. The following is a complete list of the API functions utilised in this project.

### 6.4.1 setSpeedX(speedX)

This function is used to control the speed of the x-axis motor. It only takes one argument, which is the speed value.

### 6.4.2 setSpeedY(speedY)

This function is used to control the speed of the y-axis motor. It only takes one argument, which is the speed value.

### 6.4.3 setSpeedZ(speedz)

This function is used to control the speed of the z-axis motor. It only takes one argument, which is the speed value.

### 6.4.4 xMotorOn(speedX)

This function sets the x motor status to True in the data register, then calls the setSpeedX function.

### **6.4.5 xMotorOff()**

The purpose of this function is to update the x motor status as False in the data register.

### **6.4.6 yMotorOn(speedY)**

The purpose of this function is to update the y motor status as True in the data register and consequently call the setSpeedY function.

### **6.4.7 yMotorOff()**

The purpose of this function is to update the y motor status as False in the data register.

### **6.4.8 zMotorOn(speedZ)**

The purpose of this function is to update the z motor status as True in the data register and consequently call the setSpeedZ function.

### **6.4.9 zMotorOff()**

The purpose of this function is to update the z motor status as False in the data register.

### **6.4.10 home()**

Returns all axes arms back to the home position (0,0,0), the order being x, then y, then z. Updates the current and next coordinate values in the data register.

### **6.4.11 update\_label()**

This is a helper function to provide motion and update the current position and motor status in real time in the GUI. Shows the real time information in the form of a label on the control panel.

### **6.4.12 moveTo(x, y)**

This function updates the current coordinate values in the data register to move the gantry robot to a certain point. It takes two parameters: the x and y coordinates.

### **6.4.13 moveBy(x\_sign, y\_sign, x, y)**

This function changes the gantry robot's position by a specified amount. It takes four arguments: the sign of x and y, as well as the distance it should travel along the x and y axes.

### 6.4.14 pick()

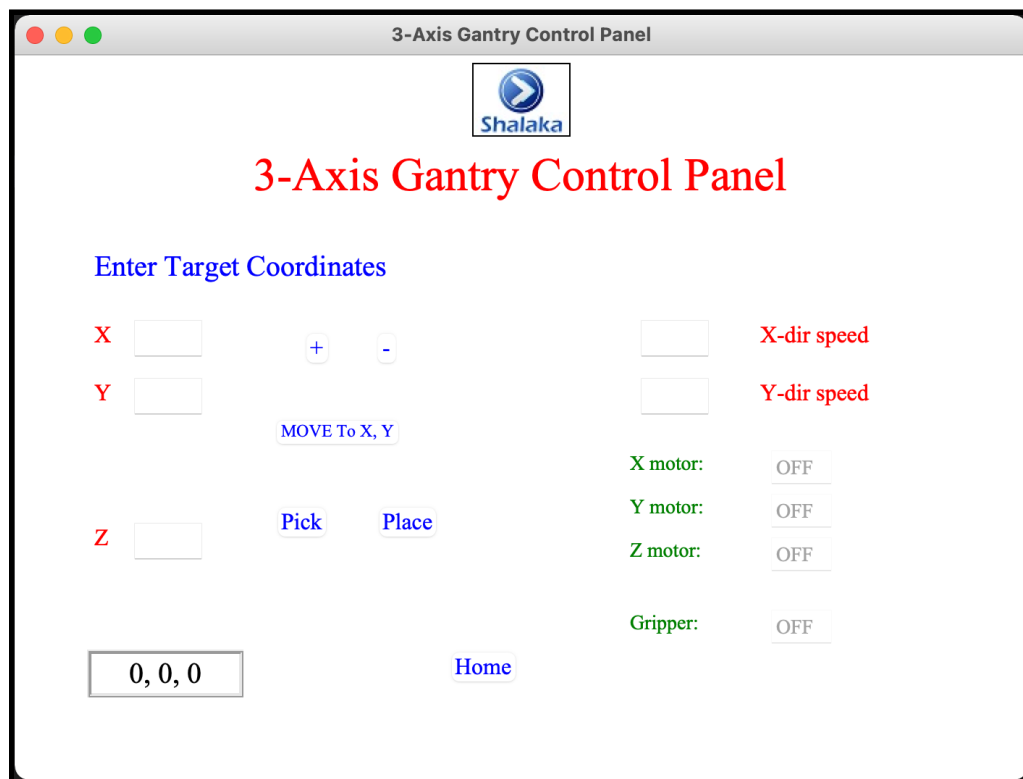
This function picks an object by setting the gripper status in the data register to True. It accepts one argument which is the z height.

### 6.4.15 place()

This function places an object by setting the gripper status in the data register to False. It accepts one argument which is the z height.

## 6.5 Control Panel Functionality

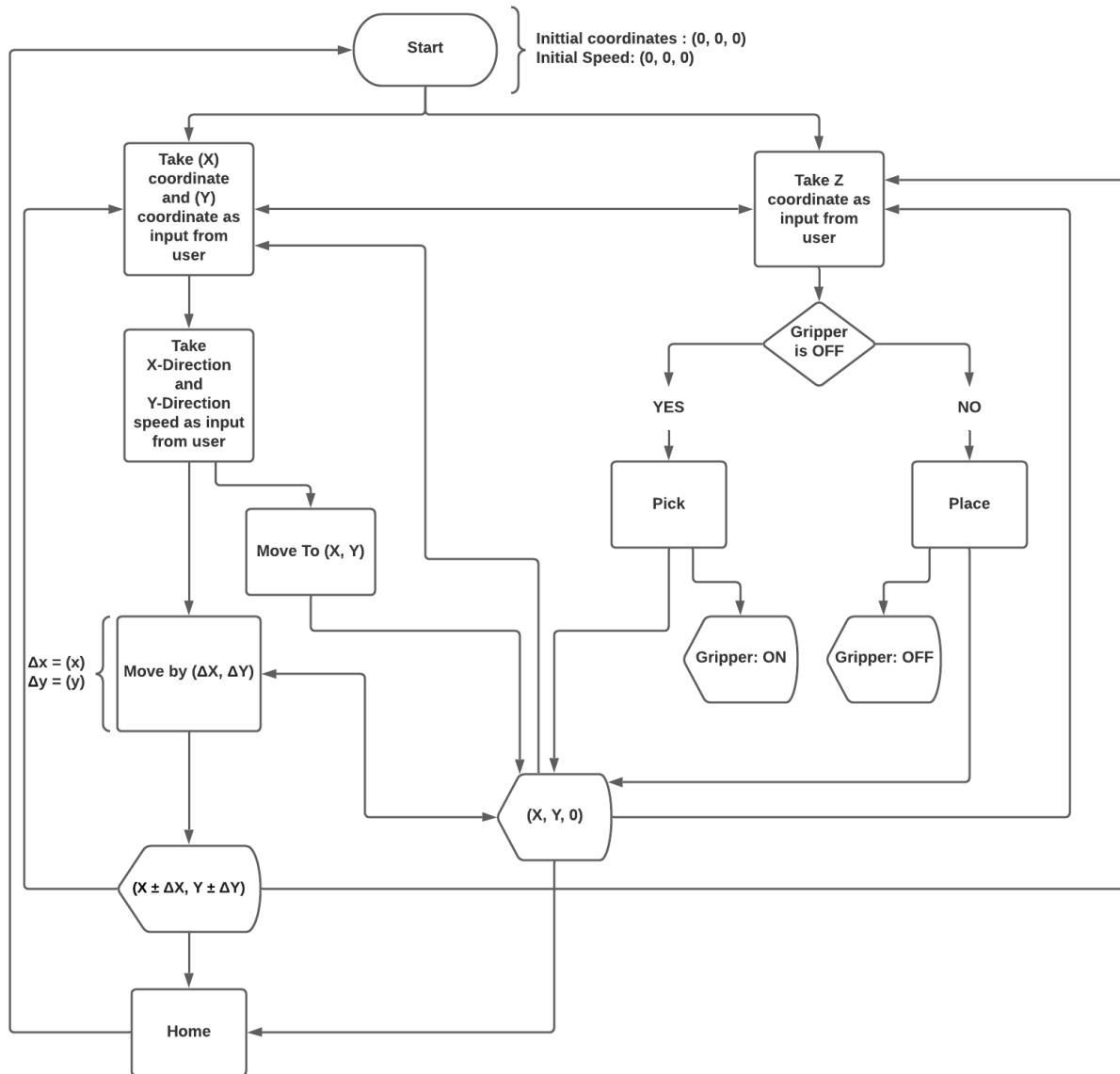
The GUI of the completed control panel looks like this



### 6.5.1 Control Flow

A top down view of the control flow of the panel is given below:





## 6.5.2 Input Entry Boxes

The control panel consists of 5 input entries:

1. X coordinate: This text box takes the x coordinate input and stores it in the data register.
2. Y coordinate: This text box takes the y coordinate input and stores it in the data register.
3. Z coordinate (height): This text box takes the z coordinate input and stores it in the data register.
4. X direction speed: This text box takes the x direction speed and stores it in the data register. The data register update happens when the motor is switched on, thus eliminating the need for a specific button.

5. Y direction speed: This text box takes the y direction speed and stores it in the data register. The data register update happens when the motor is switched on, thus eliminating the need for a specific button.

### 6.5.3 Display panel

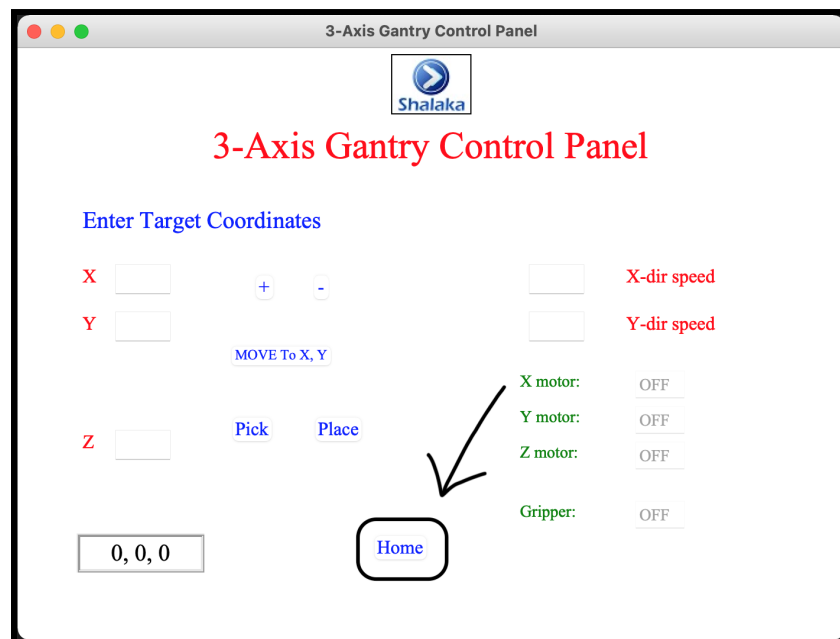
The display panel consists of:

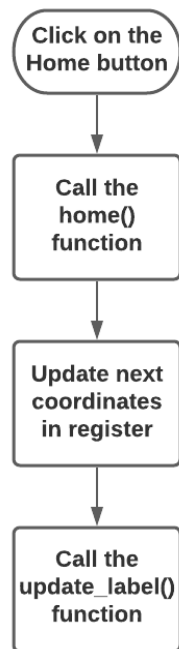
- Current position coordinates of the gantry robot (Bottom left)
- X-axis motor status (ON/OFF)
- Y-axis motor status (ON/OFF)
- X-axis motor status (ON/OFF)
- Gripper status (ON/OFF)

### 6.5.4 Buttons

There are a total of 6 buttons on the control panel. The functionality and programming behind these buttons are explained in detail below

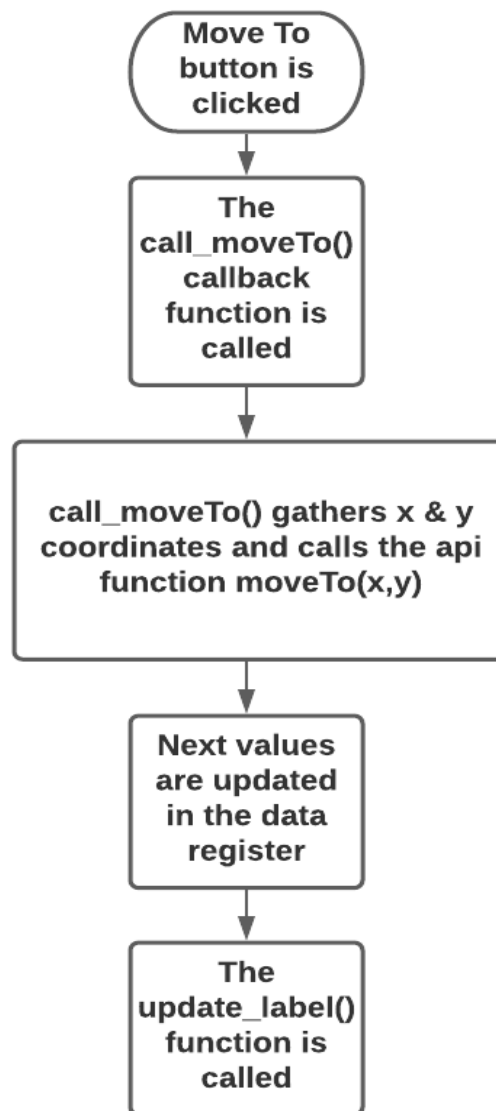
#### 6.5.4.1 Home





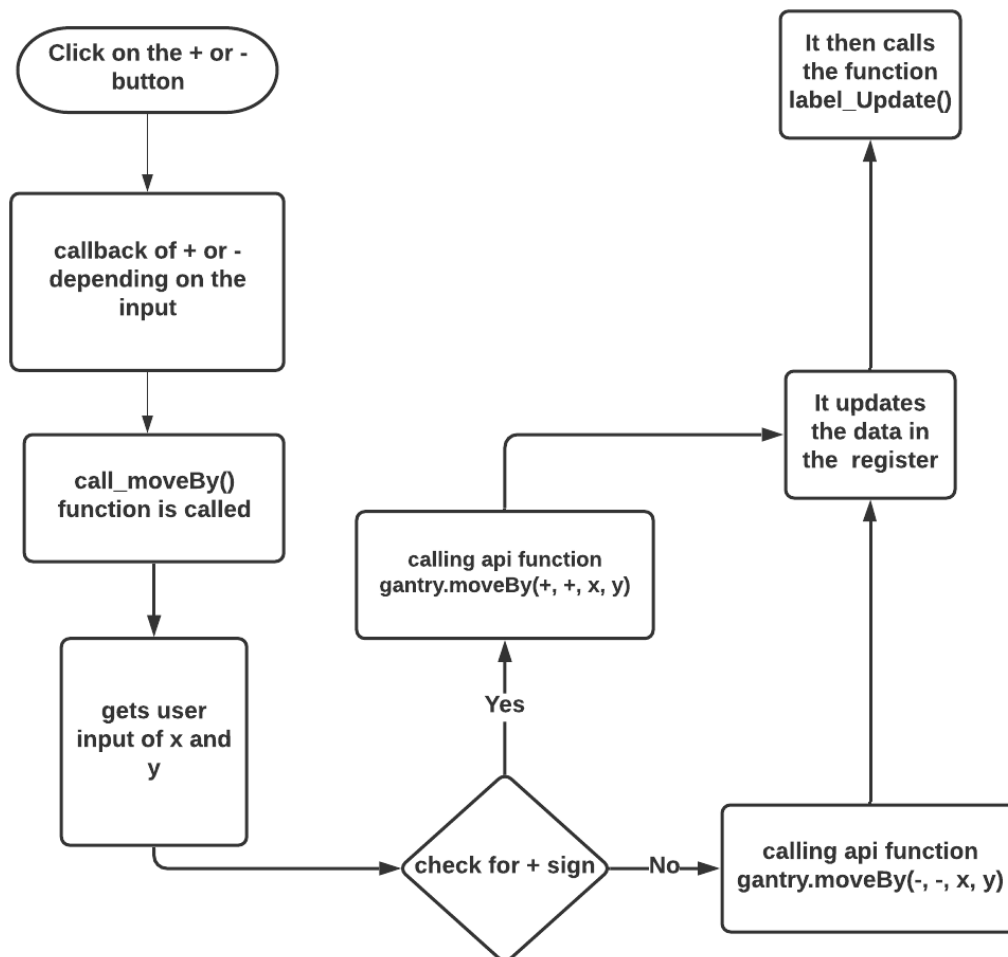
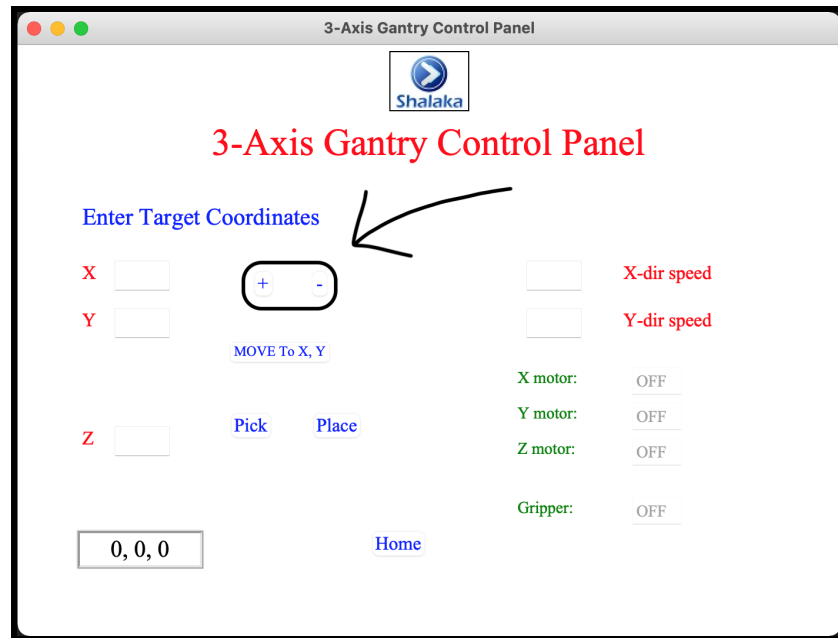
The home button serves the purpose of returning all arms of the gantry robot back to its origin coordinates, that is, (0,0,0). Upon clicking this, the *home()* api function is called, which updates the next coordinates to (0,0,0) in the register and calls the *update\_label()* api function, which in turn handles the movement of the arms back to origin. The live status of the gantry robot position is shown in the display panel at the bottom left.

#### **6.5.4.2 Move To X, Y**



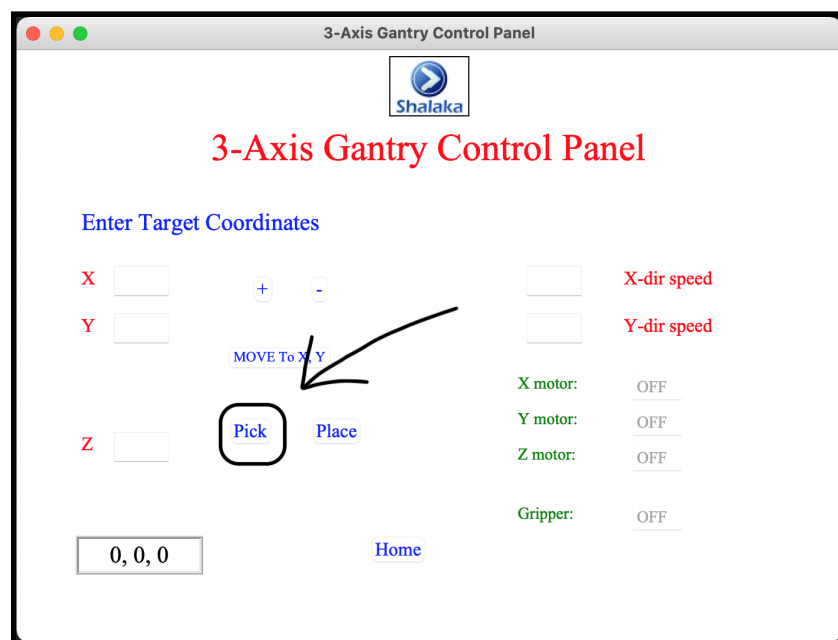
Clicking on the “Move To X,Y” button elicits a callback function called *call\_moveTo()*. This callback function gathers the input coordinates from the user and uses them to call the api function *moveTo(x,y)*. Inside the api function, the next values in the data register are updated and lastly, *update\_label()* is called to update the real time coordinates on the panel.

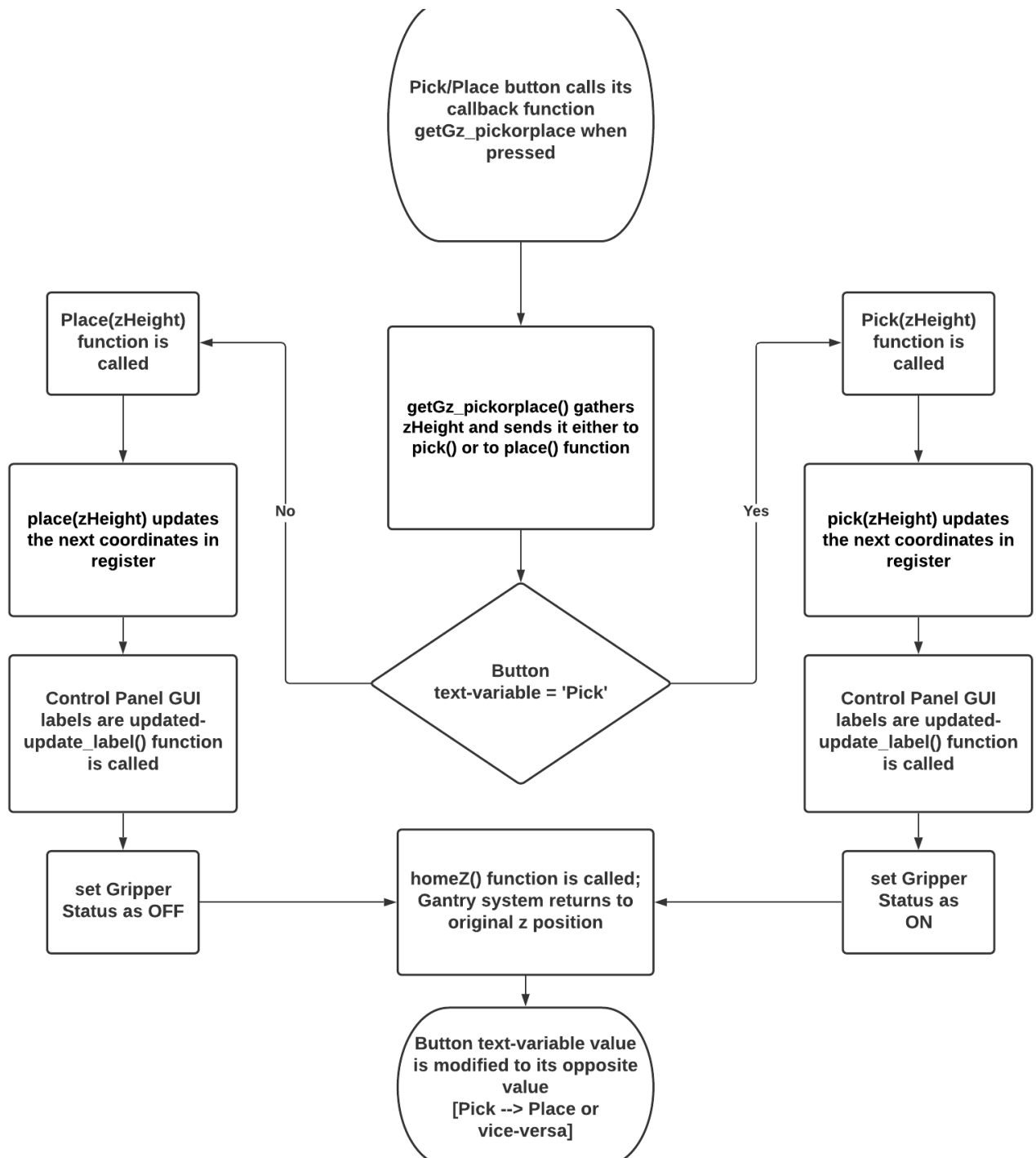
### 6.5.4.3 + and -



The entry of X and Y coordinates is first filled with the relative displacement of the coordinates which are generally less than the absolute displacement. After entering the values, the + /- button is clicked, which triggers the callback function *call\_moveBy()*. The purpose of this function is to get the user inputted values of relative displacement, and this callback function finally calls the api function *moveBy()* with the user inputted values as arguments , which updates the data in the register based on the displacement and the sign mentioned by the user and lastly updates the label by calling *update\_Label()*.

#### 6.5.4.4 Pick/Place





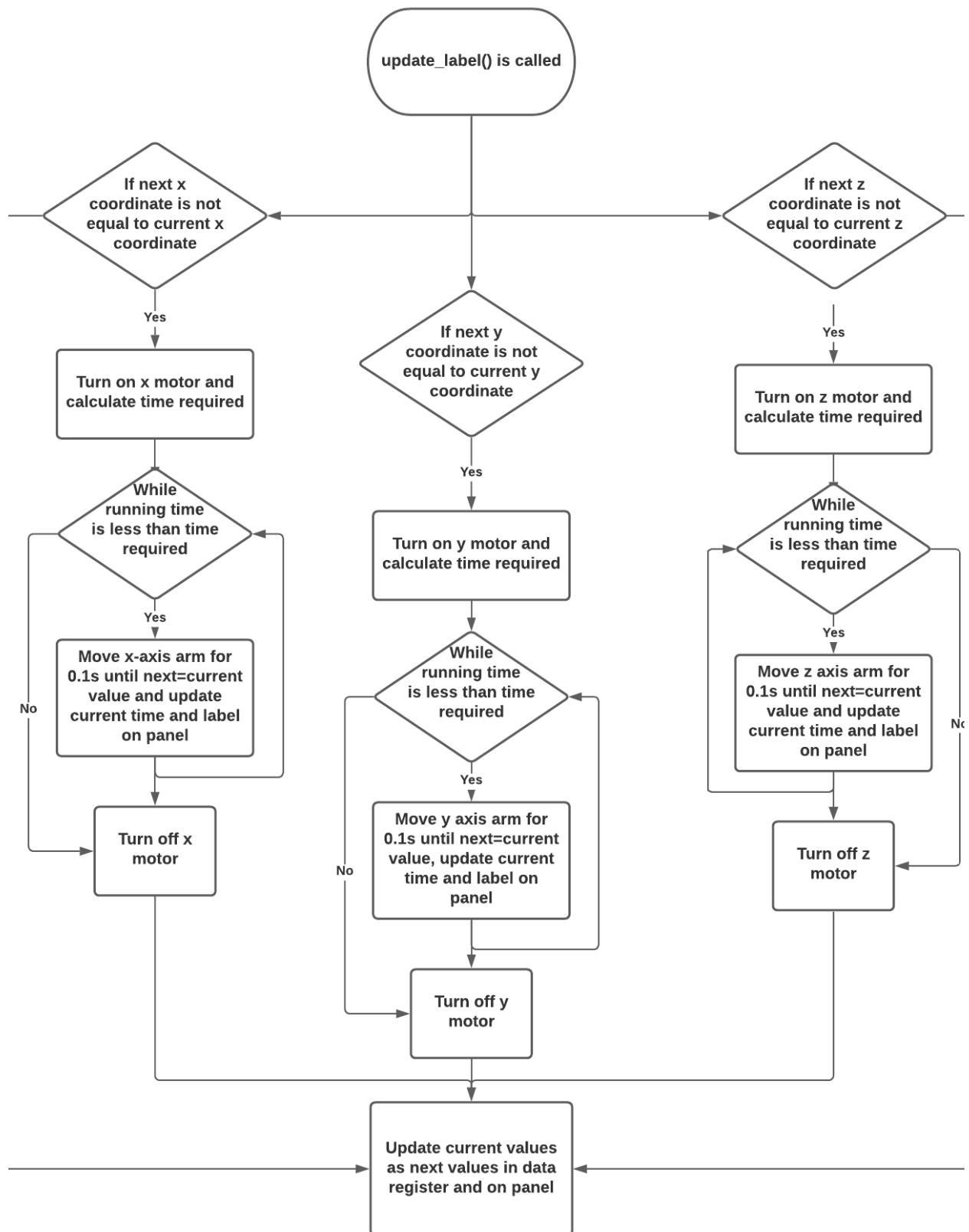
The Pick button in the Control Panel calls the *getGz\_pick()* function which gathers the z-coordinate height (which the gantry system needs to travel to reach the object to be picked) from the z-coordinate input text box and then calls the *pick(zHeight)* function. The *pick(zHeight)* function takes the z-coordinate value as parameter input to update the new coordinates of the gantry and pick the object. New x and y coordinates are set to be the same as current values and then the z-coordinate value is updated. Finally, this function sets the gripper status as ON to

collect/pick the object, updates relevant labels of the GUI interface and returns back to its original z-coordinate height (z-home).

### **6.5.5 Core Logic**

The core logic of our program lies in the `update_label()` api function, as shown below:





Attached below is a code snippet showing the definition of the update\_label() function:

```
def update_label():  
    """  
    A helper function to provide motion and update the current position and motor status in  
    real time in the GUI. Shows the real time information in the form of a label on the control panel.  
    """
```

## 7 Project Timeline and Learning Outcomes

### 7.1 Schedule

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
	5/31/2021	6/7/2021	6/14/2021	6/21/2021	6/28/2021	7/5/2021	7/12/2021	7/19/2021
Induction								
Project Planning								
Project Kick off								
Programming								
Testing								
Documentation								
Deployment								

### 7.2 Insights Gained

Since day 1, our mentor Mr. Hemanth Kamat sir has organized daily or alternate day meetings, to introduce and discuss our PS-1 project. As shown above, the first 3 weeks involved a one-way communication between us and him, where he introduced us to the idea of Industrie 4.0 and its relevance, Shalaka's involvement in Industrie 4.0, and the industrial working environment. Various learning requirements like IoT tools, developer tools, devices (eg IMU, HVAC control), etc and learning activities like IoT concepts and the industrial application development processes were taught. We were made aware of the Product Development Life Cycle, which consists of:

1. Needs Identification
2. Conceptualization

3. Requirements Gathering
4. Design and Development
5. Integration
6. Manufacturing
7. Marketing
8. Support and Maintenance
9. Upgradation
10. Retirement

Any product goes through these above 10 stages.

Additionally, the core IoT structure was taught. The 2 main protocols involved in IoT (CoAP and MQTT), and the history behind why these 2 protocols emerged as the most widely used protocols captured our interest. The passage of data via the MQTT broker and client, with their subscription and publishing model showed us the cloud architecture involved in developing an Industrie 4.0 application. After a thorough understanding of the background concepts, we were divided into teams, to tackle the enormity of the project, and there started the development life cycle.

## **8 Scope of Improvement**

Although we have successfully covered all the aspects of the problem statement of the project, there are a few aspects which could be added.

- The integration of the control panel with the cloud using MQTT broker was not implemented instead the current project involves more of desktop based simulation
- Dynamic sizing and alignment of the text boxes, buttons and labels of the control panel could be implemented to overcome disparity in window sizes
- Functions like Stop and Drop could also be implemented in the GUI.

## 9 Conclusion

This project aimed to develop a Graphical User Interface for the operation of a 3-axis Gantry Robot. This called for implementing functionality such as taking speeds and coordinates input, moving the arms of the Gantry robot, picking and placing objects, and finally displaying the real time status of the robot. The Python programming language was chosen for this project and the tkinter library for developing GUIs was used. An IDE was required for testing and debugging purposes, and this prompted the use of vscode by Microsoft, a well developed and supported IDE. The project was displayed to the mentor with a demo showcasing all the features and functionality of the control panel, and tested thoroughly by the team members.

*Having met all requirements within the stipulated time, the Control Panel for a 3-axis Gantry Robot project is said to be complete.*

# 10 References

- Python documentation - [3.9.6 Documentation](#)
- Mosquitto documentation - [Authentication methods](#)
- Youtube Tutorials on python & tkinter
- About Gantry robots –  
[Gantry Systems: Working Outside the Envelope](#)  
<https://www.sagerobot.com/gantry-robots/>
- Tkinter GUI guide- [Create UI using Tkinter in Python](#)
- Tkinter Canvas- [python - Tkinter Canvas creating rectangle - Stack Overflow](#)
- StrVar() in Tkinter: [what is the difference between a variable and StringVar\(\) of tkinter](#)
- Resizable method in Tk- [resizable\(\) method in Tkinter | Python](#)
- [Shalaka logo](#)
- [Shalaka Connected Devices – Embedded Design and IoT](#)