

Motion Control of a Cable Robotic LED Light Fixture with IoT Connectivity

by

Negar Tavakoli

B.Sc., Shiraz University of Technology, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Mechatronic Systems Engineering
Faculty of Applied Sciences

© Negar Tavakoli 2022
SIMON FRASER UNIVERSITY
Spring 2022

Copyright in this work is held by the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.

Declaration of Committee

Name: Negar Tavakoli
Degree: Master of Applied Science (Engineering)
Title: Motion Control of a Cable Robotic LED Light
Fixture with IoT Connectivity

Committee: **Chair:** Behraad Bahreyni
Associate Professor, Mechatronic Systems
Engineering

Mehrdad Moallem
Supervisor
Professor, Mechatronic Systems Engineering

Ahmad Rad
Committee Member
Professor, Mechatronic Systems Engineering

Jiacheng (Jason) Wang
Examiner
Associate Professor, Mechatronic Systems Engineering

Abstract

This thesis focuses on the development of an Internet of Things (IoT) based control scheme for a 2-degree-of-freedom Cable-Suspended Parallel Robot (CDPRS) with a Light-Emitting diodes (LED) light panel as its end effector. It is recognized that smart and intuitive monitoring and control of a greenhouses is essential for a successful yield. To this end, we developed a monitoring and control system for light fixtures that can provide a solution to non-uniformity of light distribution for plant growth aimed while enhancing light energy efficiency and lowering energy cost. Ideally, the height and angle of the light panel should adjust when a physical parameter such as plant height changes. This has not been considered in conventional non-robotic lighting systems. Furthermore, conventional non-robotic light fixture setups cannot be used to optimize light uniformity during different stages of plant growth. The aim of this thesis is to address the above problems while developing an interface for data acquisition and data visualization. The proposed setup contains of two main parts: (i) Internet of Things (IoT); and (ii) Motion control unit, which is designed to control the translational and rotational motion of the robotic light fixture. The motion control unit consists of a Trajectory Planner, which receives the controller parameters and desired reference values via proposed IoT platform and trajectory controller, which is DC motor current control or DC voltage control. The trajectory is planned based on a 5th order polynomial motion profile, which can impose kinematics constraints such as position, velocity and acceleration. The numerical simulations of the proposed trajectory generator and controllers (PID and state feedback) demonstrated accurate performance in generating the trajectories that ensure kinematic constraints are met and can be accurately tracked using a dc motor controller. A 2-degree-of-freedom Cable-Suspended Parallel Robot (CDPRS) prototype was built and used to validate the performance of the motion control unit. A digital controller was implemented using Texas Instruments CC3220S-LAUNCHXL and was programmed using Energia Integrated Development Environment (IDE) and used to control two DC motors using current control and voltage control methods for trajectory tracking. Based on experimental results, the motioned control unit successfully tracked the trajectory each time a new set point was available. The controller can perform data acquisition and data transmission to a web server with the desired frequency.

Keywords: Cable-Driven Robots, Trajectory Planning, Trajectory Tracking, DC Motor Control, Cloud data logging, Internet of Things

Acknowledgements

First and foremost I am extremely grateful to my supervisor, Prof. Mehrdad Moallem for his invaluable advice, continuous support, and patience during my study. I would also like to express my gratitude to my father. Without his tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Table of Contents

Declaration of Committee	ii
Abstract	iii
Acknowledgements	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
1 Background and Motivation	1
1.1 Optimization of Lighting in Agricultural Greenhouses: Overview	1
1.1.1 Utilization of Light-Emitting Diodes (LEDs)	2
1.1.2 Optimizing the LED Grow Lights Intensity	3
1.1.3 Optimizing the LED Grow Lights Distance From Plants	4
1.2 IoT-based Data Logging, Data Analysis and Modeling the Greenhouses Environment: Overview	5
1.3 Thesis Objectives	6
1.3.1 Robotic LED Fixture Design For Light Optimization in The Smart Greenhouse Setup	6
1.3.2 IoT Platform Design in The Smart Greenhouse Setup	7
1.4 Thesis Structure	8
2 Architectural Framework For the IoT-Enabled LED Light Fixture	9
2.1 Computing Resources	9
2.1.1 Computing Components Type Model	10
2.1.1.1 Type A Computing Components Used in The Proposed Prototype	10
2.1.1.2 Type B Computing Components Used in The Proposed Prototype	10

2.1.1.3	Computing Resources Embedded in Sensors and Actuators (Type C)	12
2.1.1.4	Non-Computing Resources (Type D)	13
2.1.2	Computing Resources Models	15
2.2	Data Storage	18
2.2.1	Data Base	18
2.2.2	Information Life-cycle Management (ILM)	19
2.2.3	Data Type	19
2.3	LAMP Stack	21
2.3.1	Insert Data Into Database Table	21
2.3.2	Fetch Data From Database Table	22
2.4	Analytics and Computing	23
2.4.1	Advantage of Data Processing at The Edge of The Network (Edge Computing)	23
2.4.2	Advantage of Data Processing Power at The Cloud: Motivating Factors	24
2.5	Conclusion	24
3	Trajectory Generation for Steady-State Motion	25
3.1	Review of Current Trajectory Planer Schemes	25
3.2	Trajectory Planer Scheme	26
3.2.1	State-Space Representation of The Proposed Light Fixture	27
3.2.2	Trajectory Planner For Linear Movement (x)	31
3.2.3	Trajectory Planner For Rotational Movement (θ)	33
3.3	Conclusion	35
4	Trajectory Tracker Design	36
4.1	Cascaded Trajectory Tracking Design (Motor Current Control)	36
4.1.1	DC Motor Current Controller or Inner Controller Design	37
4.1.2	Force Controller or Outer Controller Design	39
4.1.2.1	PD Control	39
4.1.2.2	Optimal State-Space Control	42
4.2	Trajectory Tracking using DC motor voltage control Design	44
4.2.1	Definition and calculation of v_x	47
4.2.2	Definition and calculation of v_θ	47
4.2.3	Calculating motor voltage for each motor	48
4.3	Conclusion	48
5	Simulation Studies	49
5.1	Discretization	49
5.2	Simulation of designed trajectory planner	50

5.3	MATLAB Simulations for Cascaded Trajectory Tracking Design	52
5.3.1	MATLAB Simulation for The DC Motor Current PID Control (Inner Controller in Cascaded Trajectory Tracking Design)	53
5.3.2	MATLAB Simulation for The Trajectory Tracker: Cascaded PD Con- trol and Motor Current Control	54
5.3.3	MATLAB Simulation for The Trajectory Tracker: Cascaded State- Space Control and Motor Current Control	56
5.4	MATLAB Simulations for The Trajectory Tracker: DC Motor Voltage Control	58
5.5	Conclusion	60
6	Experimental Results	61
6.1	Experimental Setup	62
6.1.1	Sensor Nodes	62
6.1.2	Actuators	63
6.2	The Internet of Things Reference Model for Motion Control	64
6.2.1	Python script: Fetch data from <i>sensorReadings</i> table	65
6.2.2	Python script: Add data to <i>desiredReferencesMotionControl</i> database table	65
6.3	Motion Control Results	67
6.3.1	Current Controller	67
6.3.2	Voltage Controller	70
6.4	Online PID Tuning	72
6.5	Conclusion	72
7	Conclusion	73
7.1	Future Work	74
	Bibliography	75
8	C Code For The Texas Instruments CC3220S-LAUNCHXL	79
9	Configuration of LAMP Server on Raspberry Pi	82
9.1	Install LAMP Server Packages	82
9.1.1	Install Apache2	82
9.1.2	Install PHP	82
9.1.3	Install MySQL	82
9.1.4	Install PhpMyAdmin	83
9.1.5	Install and setup FTP	83

List of Tables

Table 2.1	Computing components type model defined in IEEE Standard 2413-2019	10
Table 2.2	Models introduced by IEEE Standard for an architectural framework for the IoT	15
Table 5.1	Motor parameters	52
Table 5.2	Parameters of the light fixture	52
Table 5.3	Parameters of the motor current PID controller	53
Table 6.1	Parameters of the DC motor	63

List of Figures

Figure 1.1	Light spectrum output of a LED as a light source and light-wave used by plants during different growing stage	3
Figure 1.2	Process flow diagram of data acquisition, transmission and control unit for the proposed IoT application	7
Figure 2.1	Turbo Geared Motor Gearbox DC 12V Motor (40 RPM)	13
Figure 2.2	Computing resources model in the proposed prototype	16
Figure 2.3	Flowchart of switching between WiFi clients in control unit implemented on CC3220S-LAUNCHXL	17
Figure 2.4	Communication diagram describing the relationship between data acquisition, transmission and controller	18
Figure 2.5	Created tables in the database	19
Figure 2.6	Table structure of the sensorReading table	20
Figure 2.7	Table structure of the desiredReferencesMotionControl table	20
Figure 2.8	Software packages and communication protocols in the proposed setup	22
Figure 2.9	Desired reference values in the desiredReferencesmotionControl table	23
Figure 3.1	Free body diagram of the proposed cable-driven robotic light fixture	26
Figure 3.2	System described by state variables	27
Figure 3.3	Forces diagram.	28
Figure 4.1	Block diagram of a cascaded controller.	36
Figure 4.2	Block diagram of a feedback system with a PID controller.	37
Figure 4.3	Model of a DC motor	38
Figure 4.4	Second Controller (current control loop) in the cascaded trajectory tracker	39
Figure 4.5	Control values and force diagram	39
Figure 4.6	Block diagram of the implementation of observer and PD force control for the trajectory tracker	41
Figure 4.7	Block diagram of the cascaded SQL force controller and PI current controller	43
Figure 4.8	The electric equivalent circuit of the armature and the free-body diagram of the rotor	44

Figure 4.9	Motor’s angular velocity to linear velocity of the light panel conversion	45
Figure 5.1	The block diagram of the simulink’s digital clock	50
Figure 5.2	Trajectory planner block in Simulink	51
Figure 5.3	Trajectory generation of a linear path from 0.5m to 0.8m	51
Figure 5.4	Trajectory generation of a rotational path from 0.1rad to .15rad . .	51
Figure 5.5	DC motor block diagram	52
Figure 5.6	MATLAB Simulink for close loop DC motor current control	53
Figure 5.7	Step response of DC motor PID current controller	54
Figure 5.8	Cascaded height and angle control: PD as the outer controller . . .	54
Figure 5.9	Trajectory tracker using cascaded PD controller and motor current controller (Height Response)	55
Figure 5.10	Trajectory tracker using cascaded PD controller and motor current controller (Angle Response)	55
Figure 5.11	Cascaded height and angle control: LQR as the outer controller . .	56
Figure 5.12	Trajectory tracker using cascaded LQR controller and motor current controller (Height Response)	57
Figure 5.13	Trajectory tracker using cascaded LQR controller and motor current controller (Angle Response)	57
Figure 5.14	MATLAB Simulink for trajectory planner and trajectory tracker: DC motor voltage control	58
Figure 5.15	Trajectory tracker using cascaded LQR controller and motor current controller (Height Response)	59
Figure 5.16	Trajectory tracker using cascaded LQR controller and motor current controller (Angle Response)	59
Figure 6.1	The proposed experimental setup for motion control unit	61
Figure 6.2	The purposed experimental setup block diagram	62
Figure 6.3	IoT platform used in the proposed setup	64
Figure 6.4	IoT system block diagram used in the proposed setup	64
Figure 6.5	Python function to fetch data from <i>sensorReadings</i> database table	65
Figure 6.6	Python function to add data to <i>desiredReferencesMotionControl</i> database table	66
Figure 6.7	Desired references motion control database table with updated desired values	66
Figure 6.8	Circuit diagram of nand gates used to reduce number of PWM pins	67
Figure 6.9	Toshiba Quad 2 input NAND gates(74HC00AP)	68
Figure 6.10	Truth table of NAND gate	68
Figure 6.11	Circuit diagram of current control: Terminal Configuration and Functions	68

Figure 6.12	Trajectory tracker using cascaded PD controller and motor current controller (Height Response)	69
Figure 6.13	Trajectory tracker using cascaded PD controller and motor current controller (Angle Response)	69
Figure 6.14	Circuit diagram of Voltage control: terminal configuration and functions	70
Figure 6.15	Trajectory tracker using DC motor Voltage Control (Height Response)	71
Figure 6.16	Trajectory tracker using DC motor Voltage Control (Angle Response)	71
Figure 6.17	Database table containing updated desired PID values	72

Chapter 1

Background and Motivation

This chapter provides background and motivation for this study. Greenhouse farming can increase crop production by providing optimal climate conditions needed for plant growth. This chapter presents an introduction to the current monitoring and control methods used in agricultural greenhouses and utilization of innovative new technologies such as robotics, drones, LED lighting and monitoring sensors. Data acquisition is the first step of monitoring and control. To address the challenges of local Data Acquisition Systems (DAS) and accelerate the data collection, internet connectivity is employed in data acquisition applications making Internet of Things (IoT) enters agriculture field. This chapter also provides a discussion of IoT solutions for the next generations of precision agriculture is reviewed.

1.1 Optimization of Lighting in Agricultural Greenhouses: Overview

Greenhouses provide an infrastructure capable of protecting plants from harsh environment and external interference. Also, it can provide a structure where environmental variables can be logged, monitored and controlled. This results in plants growing under optimum conditions and maximizes their growth potential. In [34] advances in greenhouse automation and controlled environment agriculture is reviewed. Automation can focus on various areas of greenhouse management such as: (i) automatic detection of pests [25], [45] and greenhouse pest and disease management using optimization of lighting parameters [22], [49], (ii) auto-optimization plant watering and fertilizing [44], [48] and (iii) optimization control for greenhouse light environment [26], [50].

The growing challenge of food security when all people have sufficient, safe and nutritious food for an active and healthy life and the fact that Canada heavily relies on international imports to fulfill fresh vegetable demands is resulting in the greenhouse vegetable sector being Canada's fastest-growing horticulture sector [23]. Artificial light is required to augment natural light for year-round greenhouse fruit and vegetable production due to the

poor natural light conditions throughout the winter. During times of low solar radiation, greenhouses utilize additional lighting to increase crop production. Supplemental lighting enhances the amount of light in the greenhouse and balances the greenhouse's energy [16].

High-Pressure Sodium (HPS) and light Emitting Diodes (LED) are mainly studied in greenhouses as supplements to sunlight. In [35] radiation incident to and absorbed by a leaf under high pressure sodium and light emitting diodes with equal photosynthetic photon flux as radiation sources is studied. Results indicate that employing LED technology will end in slightly cooler leaf temperatures than leaves in greenhouses and under HPS fixtures. In [17] an experiment is conducted to investigate the effects of far-red LED light on plant growth and fruit yield of greenhouse tomato. This study shows beneficial effects of low dose of far-red light in early stage of fruit production of greenhouse tomatoes grown under HPS lighting. The fruit harvested from the plants exposed to far-red light also had higher carotenoid content.

Optimization of lighting can increase photosynthesis rates in plants during cultivation. In [21] two experiments were conducted to investigate the effects of photosynthetic flux density on a dwarf tomato cultivar ('Micro-Tom') at the vegetative growth stage. Based on the results, higher Photosynthetic Photon Flux Density (PPFD) causes a higher dry mass and a lower specific leaf area, but it does not affect the stem length. In [46] an experiment is performed to quantify the effect of LED inter-light and artificial HPS top-light on fresh and dry matter production and fruit quality of greenhouse tomatoes. The results show that the effect of additional LED inter-light was less at higher levels of HPS top-light.

Another application of lighting control is in the area of energy management and daylight harvesting. In [5] an optimisation process is design and implemented which significantly decrease the energy demand of an Indoor Vertical Farms (IVF). This achieved using electricity load shifting on IVFs resulting in a 16–26% reduction of artificial lighting costs for all months throughout the year. A review analysis presented in [11] seeks to overview current Controlled Environment Agriculture (CEA) practices as well as potential energy efficiency technologies that can enhance the sustainability and the profitability of the indoor farming industry.

1.1.1 Utilization of Light-Emitting Diodes (LEDs)

Utilization of Light-Emitting Diodes (LEDs) in controlled growth environments have come a long way in the past decade. High energy efficiency, long life time, controllable light intensity, controllable light color and capability to employ various illumination pattern are significant characteristics of LED-based illumination light sources. Therefore, it can be seen that they replace incandescent and gas discharge lamps in near future especially when it comes to vertical farming as LED's do not generate lots of heat. In [4] spectral characteristics of blue, green, red, far-red, and blue and red LED lights is studied during plant cultivation. The efficiency of LED technology has been studied and demonstrated in many experimental



Figure 1.1: Light spectrum output of a LED as a light source and light-wave used by plants during different growing stage

studies. In [38] an innovative supervisory and predictive control strategy to optimize the energy performance of the artificial lights of greenhouses is introduced in which a 50W LED is used as a light source.

Light's features such as light intensity, light's spectral power distribution and light uniformity are all environmental factors affecting plant growth. According to Erik Runkle, professor and floriculture extension specialist in the Department of Horticulture at Michigan State University, light uniformity problems still needs to be addressed especially in indoor and vertical farming [6]. Fig 1.1 shows the light spectrum output of a LED as a light source, which ranges roughly between 400 and 700 nanometers in visible wavelength. Plants mostly use the blue spectrum for the vegetation stage and the red spectrum for the flowering stage while mostly reflecting green light.

1.1.2 Optimizing the LED Grow Lights Intensity

Photosynthesis is caused by wavelengths in the range of Photosynthetic Active Radiation (PAR), rather than by electromagnetic radiation throughout the entire spectrum of light. PAR is the visible spectrum of 400nm-700nm. The light density a plant receives over time is represented by Photosynthetic Photon Flux Density (PPFD), which measures the amount of light (PAR) a plant receives over time. PPFD is measured in micro-moles [of photons] per square meter per second.

$$ppfd = \frac{\mu mol}{meter^2 sec} \quad (1.1)$$

PPFD heat maps usually provided by LED grow light manufacturers are used to determine the strength of LED light by showing PPFD readings at certain spots beneath the grow light. LED grow light's PPFD maps for the Luxx lighting 1000w DE HPS fixture can be seen on the product knowledge page [2]. Using an Ulbricht Sphere, measurements of 49-point tests was logged for three different distances from crops which illustrate the reverse relation between the distance of light source and the area to be illuminated.

Optimization of light intensity plays a major roll in energy optimization. Without enough light, a plant cannot photosynthesise fast enough regardless of suitable amount of water, carbon dioxide and other resources. Increasing the light intensity increases the rate of photosynthesis. The maximum rate is determined by the other factors involved such as concentration of carbon dioxide and temperature [28]. The intensity of light at different distances from a light source can be described using the inverse square law meaning that the intensity of light is inversely proportional to the square of the distance from the source. Therefore, when the light is moved 2.5 times as far from the plant it will receive %16 of the energy. Light intensity can be calculated using the formula in 1.2 where distance is measured in metres.

$$LightIntensity \propto \frac{1}{distance^2} \quad (1.2)$$

1.1.3 Optimizing the LED Grow Lights Distance From Plants

Light intensity is inversely proportional to square of distance from source. Hence, having a dynamic distance from light source can play an important roll in energy and cost optimization. Using 1.2 we can see that when the light is moved two times closer to crops, the duty cycle of light fixture can decrease four times while providing the same light intensity. Most LED manufactures such as Urbanvine provide the light intensity information in different heights according to the wattage of their led grow light[1]. It should be mentioned that LED light spectrum can be overwhelming to lots of plant species if it is too close to plants causing light bleaching. This impacts negatively the plant’s growth. In practice, optimized distance in which the effectiveness and safety is considered, is dynamic during plant growth. Therefore, monitoring and controlling the LED’s distance from plants has significant effect when it comes to energy optimization of greenhouse while making sure that plants get the power they need for photosynthesis. Flexibility in the reflector angle of the fixture, which can be changed to direct the light spread to the desired side of the plant canopy can be used to address the light uniformity issue. Also, the advantages of angled indoor plant light are emphasized in terms of eliminating light hot spot, decreasing energy consumption by reducing light loss, which is a takeaway from being able to place the grow lamps closer to plants without risking burning them. Some works are done regarding this solution, which are mentioned in following paragraph.

One solution is proposed by Gualala Robotics Inc. where light sources (grow lamps) are assembled on rails and therefore can be located anywhere along the rail enabling 30% more area coverage per grow lamp compared to stationary light fixture. Providing light uniformly is another achievement of using this configuration [24].

Jump Start Standing Lighting System (JSFC2KT) manufacture by Hydrofarm is another product which is designed to offer flexible angle adjustment to favor one side of the plant canopy [36]. A drawback of these systems is that the adjustment of the grow light is

manual, which requires the system to be unplugged. The other drawback is lack of a smart algorithm based on sensory feedback to determine the optimized height and angle of the light source in each grow stage while considering other environmental factors.

1.2 IoT-based Data Logging, Data Analysis and Modeling the Greenhouses Environment: Overview

Modeling and optimization of environment in greenhouses as well as crop models are essential for enhancing crop production as they have a noticeable effect on environmental management and control efficiencies. In [8] several types of models used in greenhouse environment's simulation and tuning methods to compute their parameters are reviewed. [18] has conducted a literature review on emerging areas and sophisticated controlled environment agriculture projects like the use of vertical farm and building-integrated indoor agriculture focusing on their environmental impact, energy use and efficiency.

The process of sampling signals that measure real-world physical phenomena and convert them into a digital is called Data Acquisition. Data Acquisition Systems (DAS) typically convert analog profiles into digital values for processing. In [9] Methods of Manufacturing Data Acquisition for Production Management is reviewed. Data provided by sensor stations and Data sets with greenhouse environmental inputs such as temperature, radiation, CO₂ levels, RH, and etc as quantities paired with known outputs (actual yield, growth, and water use), are building blocks of developing greenhouses environment models. In [10] Neural Network (NN) models were developed. Their work showed an improvement in growth model where the current day plus the following day data were used rather than each day separately. [33] shows the importance of having access to reliable, well stored and organized data for imputing missing tabular data collected from greenhouses. Sensor errors and other various data-loss conditions will end in missing data, which have a negative impact on machine learning algorithms modeling the greenhouse micro-climate.

[42] has conducted a review centring around communication technologies, its challenges and some specific issues associated with IoT-based smart agriculture. Mentioned paper has proposed categories of recommendations for IoT applications in smart agriculture. Internet of things, having both the capacity to integrate with data collection, monitoring and controlling side of the automation as well as interacting with internet can open the possibilities to improve the extracted information and enhance the understanding of the process. Internet of things infrastructure makes the ideas of smart home, smart plants and other smart services happen by connecting physical and virtual world in a reliable way. Internet of things being a concept for getting the data from any Wireless Sensor Network through the internet and providing real time monitoring and/or controlling has a significant place in greenhouse environment optimization.

In [30] the state-of-the-art research on IoT systems for optimized greenhouse environments is reviewed. In [13] a comprehensive Survey of Internet of Things for the future of smart agriculture is conducted. Another review is written in [39] in which a comparison between different wireless technologies in IoT-based environment monitoring systems is conducted. In [29] a study is conducted to review the potential application of sensors used in agriculture, to describe the layers of IoT in agriculture, discuss the existing sensing approaches for monitoring the agricultural parameters effectively, and deliver the general challenges encountered while implementing IoT systems. In [20] an emulation tool is developed which simulate the scenario in which the Kalman filter algorithm predicts the greenhouse sensor data. Then, the optimal parameters are computed and finally, the optimized parameters are utilized by the control unit to regulate the actuator's state to meet the desired settings in the indoor environment.

Designing accurate decision making and management algorithms resulted from sharing and collaboration of data and other resource can be seen in different agriculture applications. In [15] a monitoring system is proposed to detect the abnormal stress situations. [43] decrease the risk of agronomic and economic losses by early detection of pests by employment of Artificial Neural Networks (ANNs) and Adaptive Neuro Fuzzy Inference System (ANFIS) model in a rose greenhouse. In [30] a review is presented on state-of-the-art research on IoT systems for optimized greenhouse environments.

When thinking about IoT we should consider different areas such as data acquisition, data fusion, data storage, analytic and computing power as well as their physical location. Connection between sensory devices, actuators, and other embedded devices located in many physical locations to an internet network with potential of data storage, data exchange and data analysis is the main goal of IoT.

1.3 Thesis Objectives

Optimization of lighting can increase photosynthesis rate in plants during cultivation. The objectives of this thesis is to enable the light sources in smart greenhouses to react automatically to environmental features as well as growth profile. Besides, additional IoT-enabled feature will make it feasible for users to access the logged data for control and visualization purposes. Cloud-based data logging and cloud-based analytical services are introduced to monitor daily behavioral profiles.

1.3.1 Robotic LED Fixture Design For Light Optimization in The Smart Greenhouse Setup

One important criteria of a reliable lighting system is light uniformity. The current light fixtures used in greenhouses have a few drawbacks, which can be addressed using intelligent systems. A drawback of these systems is that the adjustment of the grow light is manual,

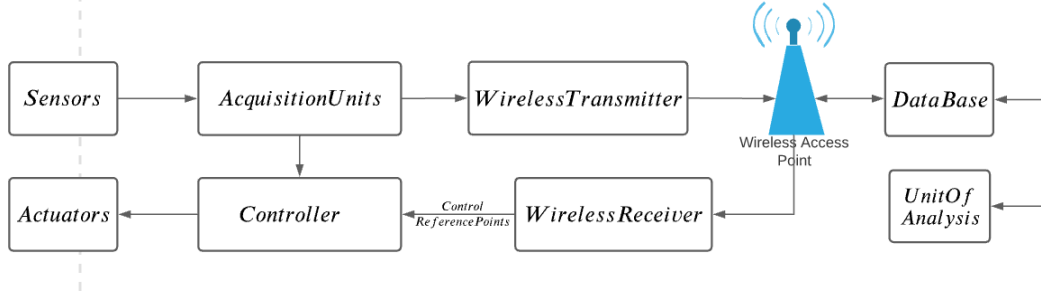


Figure 1.2: Process flow diagram of data acquisition, transmission and control unit for the proposed IoT application

which requires the system to be unplugged and re-adjust. Other drawback is the lack of a smart algorithm based on sensory feedback to determine the optimized height and angle of the light source in each grow stage.

The objectives of this research is to address these issues. Designing an IoT-based motion control of a cable robotic LED light fixture is a solution to light optimization based on plant grow profile.

1.3.2 IoT Platform Design in The Smart Greenhouse Setup

Smart greenhouses are responsive to the weather condition. With utilizing sensors, actuators and microchips they can collect, analyze and manage the data for visualization and controller purpose. Achieving higher grow yield while the resources are being fully optimized, is the main goal of automation in smart greenhouse. Smart greenhouse uses Internet of things (IoT) during operation to connect a variety of subsystems, which typically operate independently, so that these systems can share information to optimize growing performance. Designing and implementing of an IoT platform with efficient and secure connection between the sensory nodes, analyzing units and database plays a key role in achieving this goal.

In the proposed system, data acquisition layer interfaces with physical sensory nodes and transfer collected data into a web server and a data base. Collected data will fed to an analytic unit, which outputs the desired height and angle for the light source based on the gathered data as well as previous knowledge about plant's grow profile. The motion control unit will request the desired set point (with optional frequency) via Hypertext Transfer Protocol (HTTP) request. Trajectory planer unit of the motion control unit will calculate a motion trajectory and the control unit is in charge of tracking this trajectory.

Fig 1.2 shows the process flow diagram of the Data acquisition, transmission and control for IoT applications.

1.4 Thesis Structure

This paper is consisted of seven chapters with topic related information and divided into two main sections: (i) IoT Platform Design and (ii) Robotic LED Fixture Design for a smart greenhouse. The First Chapter gives an overview of the thesis main goals and objectives and methodology, as well as introduction to the concept, background and motivation. Moreover, this chapter focuses on the introduction of some of the most important principles. A more detailed overview of literature, research methods, and online resources used are placed in this chapter.

In the Second Chapter, we will discuss the IoT concept and the platform design as well as data acquisition components, data handling and data management and analysis system. In particular, the Internet of Things section of this thesis outlines LAMP stack framework to enable the Data Management Systems using a server to host dynamic websites in, which the site data is stored in a MySQL database and dynamic content is processed by PHP, is introduced.

Interaction between this IoT platform and the motion control unit is also talked about where the server is able to provide the reference value for the motion controller via HTTP request. The thesis presents an IoT platform to efficiently answers the queries regarding desired values for the motion control unit.

Chapter 3 provides us with the steady-state representation of the light fixture. Theoretical part of Trajectory Generation for steady-state motion is provided in this section. In the next chapter Chapter Four cascaded trajectory tracking design using DC motor current control as the inner controller and both PD control and Optimal State control as outer controller is discussed. Also a DC motor voltage control is introduced for tracking the motion trajectory.

Chapter 5 provides us with the simulation study and Chapter 6 provides us with the practical and experimental results. Finally, in Chapter 7, summarized theoretical and practical parts and the comparison between different trajectory tacking methods is talked about. Information about the current challenges and issues, which technological sectors face and what could potentially create an issue with concept implementation in the future is also placed in this chapter.

Chapter 2

Architectural Framework For the IoT-Enabled LED Light Fixture

In this chapter the concept of Internet of Things (IoT) in a greenhouse is introduced. IoT-enabled smart greenhouse uses IoT during operation to connect its subsystems and enabling them to share information among themselves. This will end in optimized growing performance. This chapter introduces the subsystems used in the proposed prototype. Also, we can see how by utilizing sensors, actuators and microchips we can collect, analyze and manage data for visualization and control purposes.

2.1 Computing Resources

Internet of Things (IoT) is the larger vision of Machine-to-Machine (M2M) technology, which enables devices to communicate with one another. By employing M2M, mobile devices, embedded processors, smart sensors, actuators, and embedded processors can work together, talk to one another, take measurements, and make decisions, often bypassing human interaction [47]. In M2M, devices became connected, enabling IoT development and IoT infrastructure. On this foundation, IoT is built and developed.

According to the 2020 release of IEEE standard for an architectural framework for the Internet of Things (IoT), a simple definition of an IoT system is “a system of entities (including cyber-physical devices, information resources, and people) that exchange information and interact with the physical world by sensing, processing information, and actuating.” [4]. IEEE standard 2413-2019 for an architectural framework for the internet of Things introduces 4 computing components type and 3 computing resources model. Computing components types are type A standing for highly centralized computing components, type B standing for computing components connected to the network, type C standing for computing resources embedded in sensors and actuators and type D, which stands for non-computing resources in IoT architectural framework.

Type	Definition
Type A	Highly centralized computing components such as data centers and cloud servers
Type B	Computing components (other than Type A and Type C) connected to the network, for instance, gateways, PLCs, edge-cloud servers, and PCs
Type C	Computing resources embedded in sensors and actuators
Type D	Non-computing resources

Table 2.1: Computing components type model defined in IEEE Standard 2413-2019

Three computing resources models are: (i) centralized computing resources model, (ii) distributed computing resources model and (iii) combination of the two models.

2.1.1 Computing Components Type Model

Table 2.1 illustrate the name and summarize the main definition of each of the four component type. Details of the used components for each type are discussed in following subsections.

2.1.1.1 Type A Computing Components Used in The Proposed Prototype

- **Raspberry Pi:** Raspberry Pi was created by the Raspberry Pi Foundation and Broadcom. Although the initial objective of the Raspberry Pi project was to supply cheap tools for educating essential computer science in schools, it became much more well known for employments in robotic and mechanical technology. Raspberry pi is a single-board computer with a Linux-based operating system. Raspberry pi acts as a Type A in the presented architectural framework. The connection between Raspberry Pi and other computing components is wireless and through internet. This is achieved by enabling the Raspberry pi to perform as a web server.

In this thesis, LAMP stack (Linux, Apache, MySQL, PHP/Perl/Python), which is a popular web server software is implemented on the Raspberry Pi. For a web application to work smoothly, it has to include an operating system, a web server, a database, and a programming language. LAMP's components are completely interchangeable. In the proposed stack, LAMP has Raspbian as its Linux operating system, Apache2 as its web server software, MariaDB as its database and both Hypertext Preprocessor (PHP) and Python as the used programming language.

2.1.1.2 Type B Computing Components Used in The Proposed Prototype

The Type B component in this prototype has two role: (i) Data acquisition transmission systems (DATS) and (ii) Controller. Data acquisition transmission systems is the layer that interfaces with the physical sensory nodes or other data acquisition devices, which are

employed to measure quantities that describe different physical features in the environment such as light intensity, temperature, pressure, force, sound etc. Collected digital or analog raw data from sensory nodes usually in the form of electrical attributes (such as voltage, current, resistance) are fed as input to DATS, which goes through the first layer of data pre-processing by being converted to a manageable format.

Most well-known data acquisition boards can be named as: (i) Programmable Logical Controller for industrial and harsh environment and (ii) microprocessors for lighter applications, which are able to receive raw data and convert it to digital signals, which is readable for the processor unit. Different Sensory Network (SN) are employed in DATS based on applications required solutions and different protocols are used to standardize the communication. Outputs from sensors (nodes) need communication interface to pass the raw data to the central node where the pre-processing starts. Required Interface can be wired or wireless, which categorize the network to Wire and Wireless Sensor Network (WSN) or combination of both depends on method of sending data.

Wireless nodes are equipped with radio transceiver, which makes it possible for them to communicate via radio waves (wireless). Wired nodes usually need two pair of wire one for clock and one for data as they use serial communication and I2C protocol.

- **SimpleLink Wi-Fi CC3220S-LAUNCHXL Wireless Microcontroller LaunchPad:**

A SimpleLink Wi-Fi CC3220SF Wireless Microcontroller LaunchPad is a computing component used as a type B in this prototype, which oversees data gathering and controlling the process. This node is in the grow chamber and its I/O ports are connected to affordable wired sensors such as RGB sensor, Humidity sensor, temperature sensor, ultrasound sensor and gyroscope. In addition, wirelessly and using Representational State Transfer (REST) design the gathered sensory data is being send to the data center every 15 minutes. This component act as a client for the LED panel as well and continually send HTTP request to the LED panel, which have the actuators (LEDs) embedded in it. Using multitasking feature of CC3220S-LAUNCHXL we can send HTTP request to the LED server every 1 second and HTTP request to Database every 15 minutes. This board have the ability to communicate with sensor nodes through all the well known communication protocols.

- **Protocols:**

Protocols play important rules in data acquisition and data exchange process. Different protocols are designed for each layer of data transaction in the network, which act as quality assurance and practically, they are set of rules, which are being used in data communicate. Network protocols for the Internet of things in which embedded devices use Internet to exchange information do not need

to be as complex as network protocols for the Internet of people. As an example, transport layer of TCP/IP (Protocol at the heart of the internet) has two transport protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), being used for human interaction and Domain Name System (DNS and DHCP) respectively. UDP being a simpler protocol, can meet the requirement of transport layer for IoT. It is used for sensor data acquisition and remote control in transport layer of IP Smart Objects Protocol Suite. It is also showing better performance for real time data applications. Internet of things' Datalink, Network, Transport and Application Layer use sets of protocols like IEEE 802.15.4e MAC and PHY, 6LoWPAN, IPV6/IP adapted routing protocol, UDP, DTLS, CoAP, MQTT, JSON, XML, etc. respectively.

Criteria which are important in IoT industry' protocols can be listed as: quick and efficient cooperation between things, using less programming, power, memory and data transmission.

- * Protocols for IoT's Datalink Layer : IEEE 802, ZigBee
- * Protocols for IoT's Network Layer : IP version 4 (IPv4), IP version 6 (IPv6), Thread, 6LoWPAN
- * Protocols for IoT's Transport Layer : UDP, DTLS
- * Protocols for IoT's Application Layer: CoAP, Message Queuing Telemetry Transport (MQTT), MQTT-SN, STOMP, HTTP

2.1.1.3 Computing Resources Embedded in Sensors and Actuators (Type C)

Some sensory devices have embedded digital processor located in them, which can provide accurate results with low latency by offloading both timing requirements and processing power from the host processor and save valuable MIPS on the host processor for use in the application and simplify the software architecture.

- **MPU-6050 GY-521 Accelerometer and Gyroscope:** As the first integrated six-axis motion tracking device, the MPU-60X0 combines a Three-axis gyroscope and 3-axis accelerometer in a compact 4x4x0.9 mm package.

The MPU-60X0 is comprised of the key blocks and functions such as three-axis Micro Electro-Mechanical System (MEMS) rate gyroscope sensor with 16-bit Analog-to-Digital Converters (ADCs) and signal conditioning, Digital Motion Processor (DMP) engine, Sensor Data Registers, Interrupts, Primary I2C serial communications interfaces, Auxiliary I2C serial interface for third party magnetometer and other sensors. MPU-60X0 is equipped with a Digital Motion Processor (DMP) that handles the computation of motion processing algorithms. DMP acquires data from accelerometers, gyroscopes, and third party sensors such as magnetometers, and processes them.

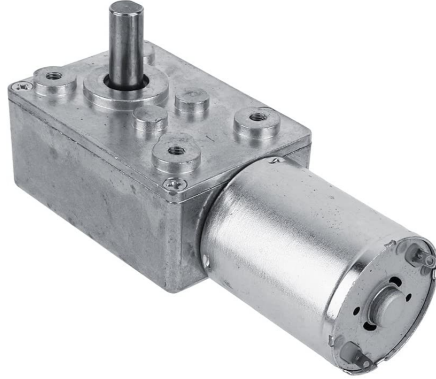


Figure 2.1: Turbo Geared Motor Gearbox DC 12V Motor (40 RPM)

The MPU's external pins are accessible to the DMP, which can be used for interrupt generation. The resulting data can be read from the DMP's registers or buffered [3].

- **Time-of-Flight Ranging Unit (VL53L0X) as a Distance Sensor:** Digital processing is the last operation inside the ranging sequence that computes, validates or rejects a ranging measurement. Part of this processing is performed internally while the other part is executed on the host by the API. At the end of the digital processing, the ranging distance is computed by VL53L0X. Functions such as signal value check (weak signal), cross-talk correction (in case of cover glass), final ranging value computation, offset correction are performed on the device itself. The host can perform some extra processing to improve range accuracy, for example, rolling average, hysteresis or any filtering.

2.1.1.4 Non-Computing Resources (Type D)

- **High Torque Turbo Geared Motor Gearbox DC 12V Motor (40 RPM):** DC motor transforms electrical energy into mechanical energy. The motor used in prototype is Gear DC motor with micro-turbine worm in witch changing the wiring-connection results into motor rotation change. Turbo worm geared motor with self-lock, that is, in the case of motor without electric, the output axis is fixed. This is a DC 12V voltage with 40rpm. The gearbox output shaft direction of the motor shaft is arranged vertically. The body of the motor output shaft relative to the general direction of gear motor short, Widely adapted to different installation dimensions.
- **TA7291P IC Motor Driver Toshiba:** The TA7291P is a Bridge Driver with output voltage control. Bridge driver is an electronic circuit that switches the polarity of a voltage applied to a load. It often used in robotics and other applications to allow DC motors to run forwards or backwards.

- **Logic ICs HD74LS00P:** The HD74LS00P is a logic gate contain four independent, 2-input NAND gates. The devices perform the Boolean function $Y = A.B$ or $Y = A + B$ in positive logic.

2.1.2 Computing Resources Models

IEEE Standard 2413-2019 introduces three computing resources models. These models and their block diagrams are illustrated in table 2.2. Utilizing the mentioned IEEE Standard,

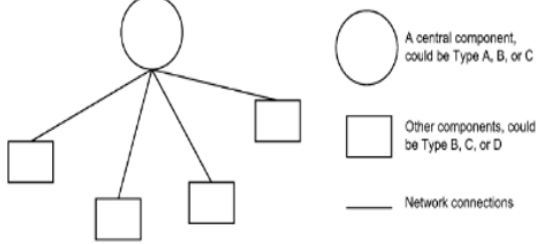
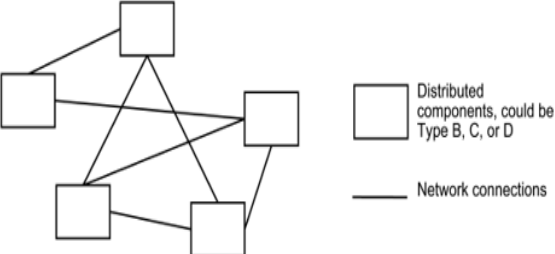
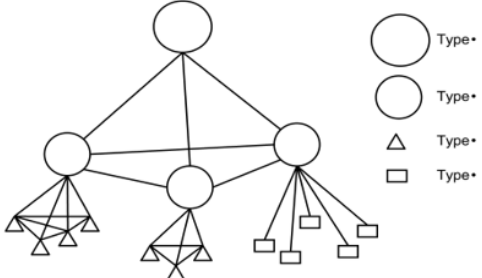
Model	Model's Block diagram
Centralized computing resources model	
Distributed computing resources model	
Combination of the two models	

Table 2.2: Models introduced by IEEE Standard for an architectural framework for the IoT

the hereby prototype has adapted the combination of the centralized computing resources model and distributed computing resources model. Fig 2.2 shows the computing resources in our setup.

In the proposed arrangement, a CC3220S-LAUNCHXL will become a client for two web servers. One is the web server on Raspberry Pi and the other one is the web server on the LED light panel.

- **Raspberry Pi client:** Raspberry Pi client requests the references for the motion control unit and post the sensor readings to data base for data storage.

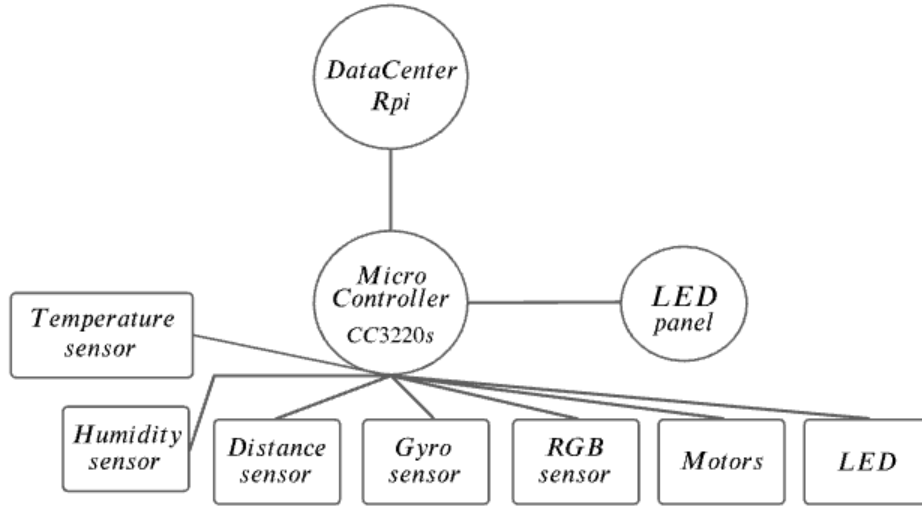


Figure 2.2: Computing resources model in the proposed prototype

- **LED panel client:** LED panel client post desired light intensity to the LED panel

It should be mentioned that the Texas Instrument CC3220S-LAUNCHXL cannot be the client for both servers at the same time. Fig 2.3 shows the logic flowchart for switching between two servers. Communication diagram perfectly describes the relationship between these computing resources (data acquisition, transmission and controller). As can be seen, a desired value is retrieved using Raspberry Pi HTTP request and is given to the controller to create a Pulse-width modulation (PWM) signal in order to produce desired output signal to control the motor's speed. PWM is a method of controlling analog devices with a digital output. PWM simulates an analog result by applying power in pulses, or short bursts of regulated voltage. Sensors capture physical features of the movement and environment and transfer them to acquisition unit, which will post these readings to the Raspberry Pi web server and database. Also, every time that there is a new light intensity value the controller will post the updates numbers to the LED panel.

Fig 2.4 shows the computing resources and their connection to each other in a form of a block diagram containing Data acquisition unit, transmission unit and control unit.

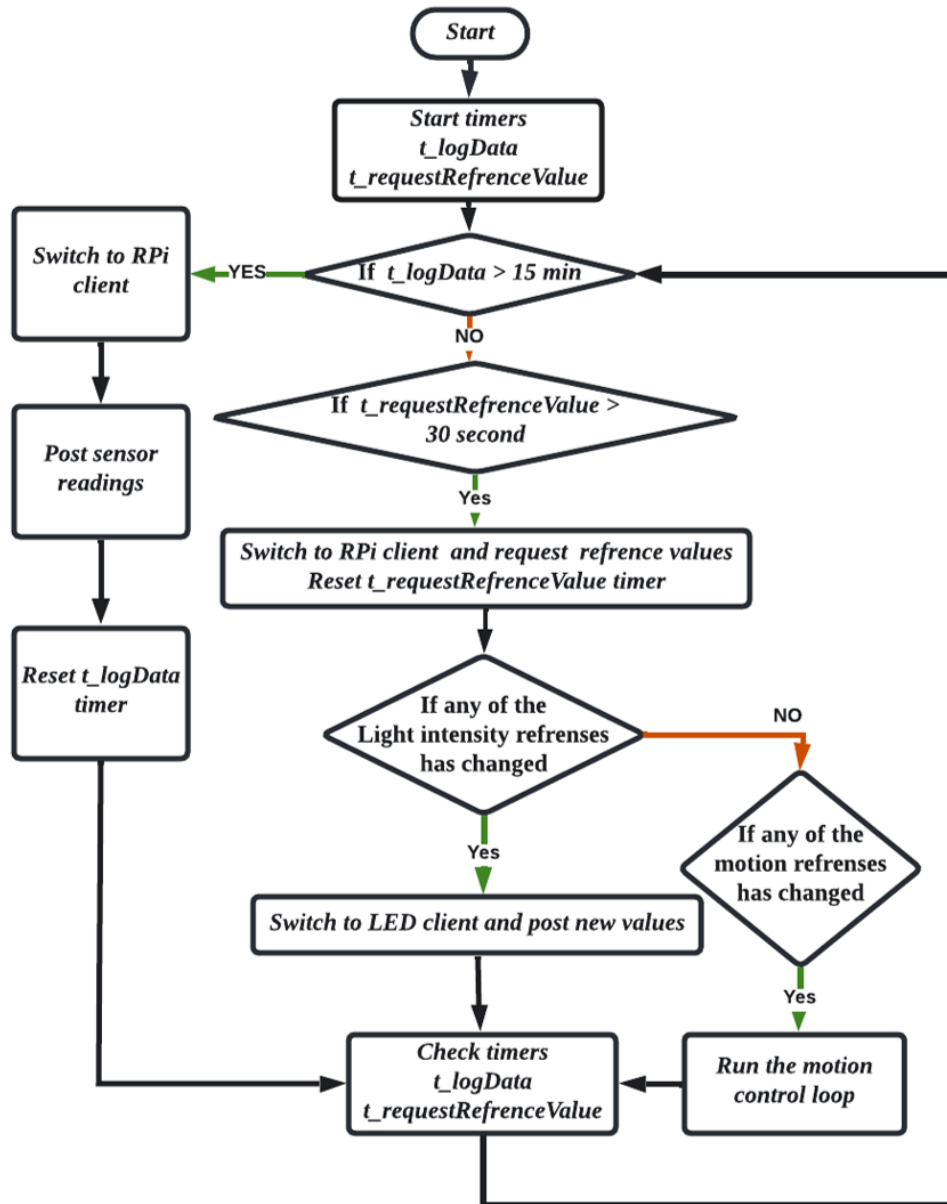


Figure 2.3: Flowchart of switching between WiFi clients in control unit implemented on CC3220S-LAUNCHXL

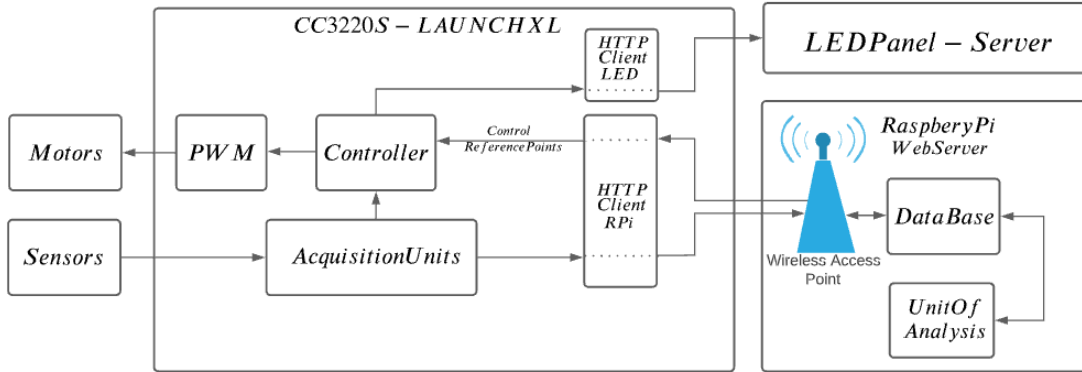


Figure 2.4: Communication diagram describing the relationship between data acquisition, transmission and controller

2.2 Data Storage

Data generated via one or more sensory device can be captured for the purposes of monitoring, decision making and information extraction in deeper processing levels. Gathering, cleaning and storing big amount of data coming from different sensors is one of the challenges that IoT developers try to overcome. This storing methodology should support scenarios in, which data might import or edit by users. The challenge here is to determine the time this data needs to be stored and kept. Keeping the data is a cost for organizations so it is reasonable to think of it as an important element. That's one reason data management tools are high demand and trending. Data maintenance's expenses grow by data growth and backup needs and they differ by the chose of storage hardware, retrieval speed and storage management. (To deal with data growth and keep costs manageable, many enterprises are turning to cloud storage, which greatly increases the importance of effective cloud storage management.

2.2.1 Data Base

A database is a collection of structured information, or data, usually stored electronically in a computer system. In this thesis MariaDB is used, which is a commercially supported fork of the MySQL relational database management system (RDBMS) and is an open-source software under the GNU General Public License. MariaDB is intended to maintain high compatibility with MySQL and exact matching with MySQL APIs and commands.

Most databases are controlled by a database management system (DBMS). Here, we use phpMyAdmin as a tool to handle the administration of MariaDB over the Web. phpMyAdmin is a free software tool written in PHP, which supports a wide range of operations

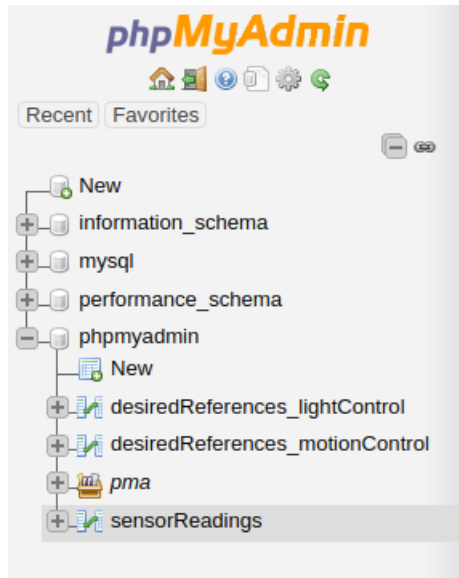


Figure 2.5: Created tables in the database

on MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface.

2.2.2 Information Life-cycle Management (ILM)

Lifecycle of data from creation and initial storage to the time when it becomes obsolete and is deleted can be managed by taking the purpose of the creation of that data into consideration. If the information has a finite purpose it should be deleted from storage network once it has served its purpose. Data may be re-used, archived or shared on the course of its lifecycle depends on the application. Not surprisingly, Data collected in the form of time series are extremely dynamic. Attributes of each data sample update frequently, which is the nature of the time series data. Studying the changes in these attributes over time is what makes it possible to model the behavior of the application under study and have it analysed.

2.2.3 Data Type

In computer programming, a data type is a classification that determines what a variable or object can store. Almost all computer programming languages, including C and C++, rely heavily on data types. To secure the proper outcome and an error-free software, programmers must reference and use data types correctly while creating computer programmers.

phpMyAdmin lets us assign a type to each observation easily. This is very important as the Type B node or CC3220S-LAUNCHXL in this prototype sends the sensor readings for each sampling using HTTP request. Anatomy of an HTTP Request has host, method, path, headers, HTTP version, query string, and request body. CC3220S-LAUNCHXL sends all

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	Id	bigint(20)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	Height	float			Yes	NULL		
<input type="checkbox"/>	3	Angle	float			Yes	NULL		
<input type="checkbox"/>	4	Temperature	float			Yes	NULL		
<input type="checkbox"/>	5	Humidity	float			Yes	NULL		
<input type="checkbox"/>	6	R1	float			Yes	NULL		
<input type="checkbox"/>	7	B1	float			Yes	NULL		
<input type="checkbox"/>	8	Par1	float			Yes	NULL		
<input type="checkbox"/>	9	R2	float			Yes	NULL		
<input type="checkbox"/>	10	B2	float			Yes	NULL		
<input type="checkbox"/>	11	Par2	float			Yes	NULL		
<input type="checkbox"/>	12	Time	timestamp			Yes	current_timestamp()		

Figure 2.6: Table structure of the sensorReading table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	Id	bigint(20)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	desiredHeight	float			Yes	NULL		
<input type="checkbox"/>	3	desiredAngle	float			Yes	NULL		
<input type="checkbox"/>	4	Time	timestamp			Yes	current_timestamp()		

Figure 2.7: Table structure of the desiredReferencesMotionControl table

the sensor readings in a form of one String in HTTP request body so we have to assign the data type to each observation using the database management system. Each time a content is being add to database table it will cast to assigned datatype.

- **Time series data analysis of IoT events:** Sensory nodes can read the data by being triggered by an event. The event can be time intervals, which lead to generation of time series data type. Data collected in the form of time series either on regular or irregular time basis is a valuable resource for organizations as it has the potential to be used as an input to the Machine Learning and Deep Learning algorithms.

While creating a new table we can add a column with `TIMESTAMP` data type. The MySQL `TIMESTAMP` is a data type that holds the combination of date and time. The format of a `TIMESTAMP` is `YYYY-MM-DD HH:MM:SS`, which is fixed at 19 characters. This will let us have the exact time for each observation when that observation is being add

to the database. The alternative way to add the time is to have the source that providing the data provide the captured time as well. In this prototype `TIMESTAMP` data type with default "current timestamp" is used.

As can be seen in fig 2.5 three separate data tables are defined. The `desiredReferences-motionControl` and `desiredReferenceslightControl` tables contains the reference values for motion control and light control units and the `sensorReadings` table contains the observations gathers via sensor nodes. Fig 2.6 and 2.7 illustrate the data types chosen for variables used in this project for `sensorReadings` table and `desiredReferencesmotionControl` table.

2.3 LAMP Stack

LAMP stack web server software commonly refers to the combination of Linux, Apache, MySQL and PHP where P portion is usually the scripting engine used to provide content from database tables and served through Apache. Here we use PHP scripting language that is especially suited to web development. However, the Python programming language is also employed for the purpose of data analysing. Python IED will act as client for the web server even though there are installed on one hardware. It should be mentioned that the web server and data analysis unit can be easily separated to different hardware and they are not depended. In that case we will have two Type A in our framework. The documentation of this server is in appendix A.

2.8 shows the software packages and communication protocols in this scenario. As can be seen, LAMP Server Packages, which are Apache2, MySQL, PHP should be installed on the Raspberry Pi. Also, phpMyAdmin needed to be installed to handle administration of MariaDB over the Web.

2.3.1 Insert Data Into Database Table

In this Prototype, two sources have permission to insert data to database tables. One is `phpmyadmin`, which handles the http requests received from `CC3220S-LAUNCHXL` Wireless Microcontroller and the second one is the main Python code running locally on Raspberry Pi.

Example for receiving new record from the python code for desired references for motion control: The main python code needs to establish a connection to database and add a new desired height and desired angle for the motion control unit into `desiredReferences-motionControl` table using `INSERT INTO table SET SQL` query, a new record will be created with a unique ID and current time 2.9. It should be noted that the connection between python script and database only happens if the admin password for database is provided and it gives php permission to access and add content to the database. As an example for receiving new record from the `CC3220S-LAUNCHXL` Wireless Microcontroller: The `phpmyadmin` needs to establish a connection to database and add the new record into

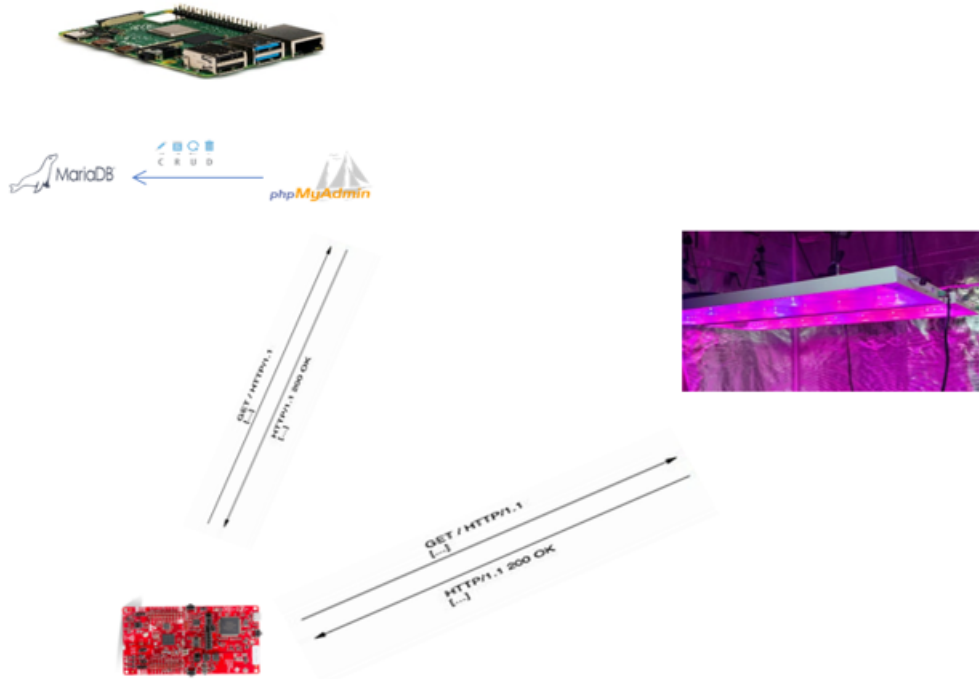


Figure 2.8: Software packages and communication protocols in the proposed setup

sensorReadings table using Using INSERT INTO table SET SQL query. Variables Height, Angle, Temperature, Humidity, R1, B1, Par1, R2, B2 and Par2 have their values being set with the value transferred by the CC3220S-LAUNCHXL Wireless Microcontroller. Column ID and TimeStamp get their value set automatically each time a new record is made. fig 2.6 shows the variables stored in this table.

2.3.2 Fetch Data From Database Table

As an example for fetching data from database table for the CC3220S-LAUNCHXL Wireless Microcontroller: Each time the CC3220S-LAUNCHXL is requesting the most updated desired height and desired angle for the motion control unit, Apache2 execute a php script file, which establish a connection to the database and the *desiredReferencesmotionControl*. The most Updated motion references values retrieve (Fetch) from Database and passes back to the Apache web server to send back to the CC3220S-LAUNCHXL.

It should be noted that the connection between PHP script and database only happens if the admin password for database is provided and it gives php permission to access and add content to the database.

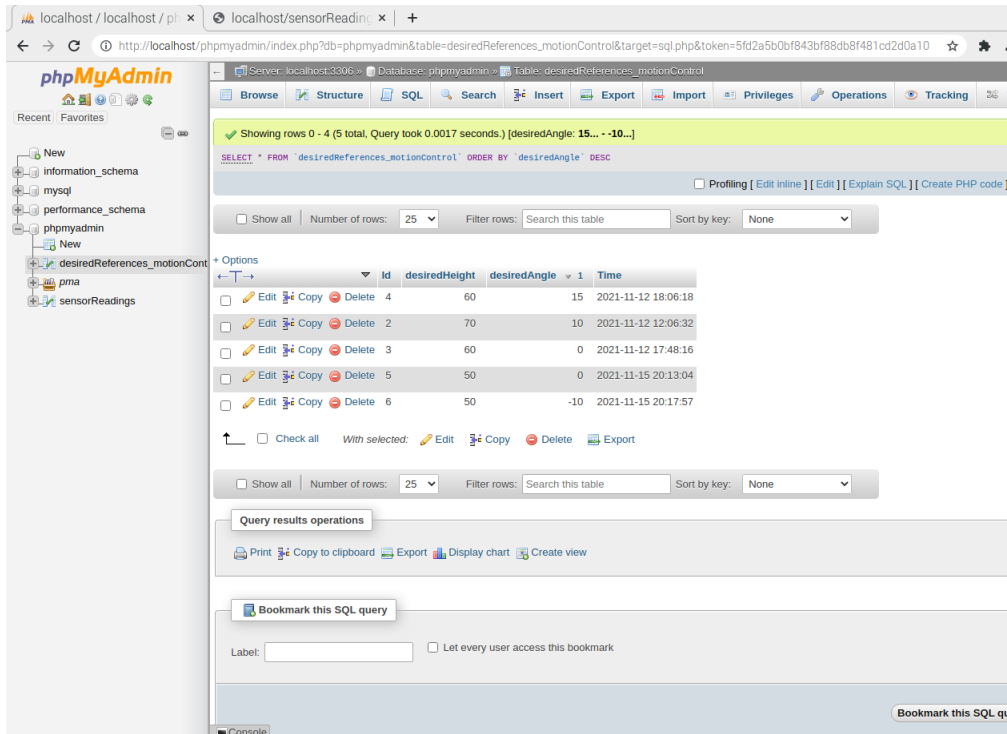


Figure 2.9: Desired reference values in the desiredReferencesmotionControl table

2.4 Analytics and Computing

After reading raw data from sensors, the processing step must be deployed. This processing can happen in different level of complication from calibrating the data coming form one sensory node to applying state of the art algorithms on the data and using them to update trained models. Sometimes a node need data from other nodes to have its raw readings calibrated. For example, a device used for measuring the level of salt in soil called EC meter has its reading highly correlated with temperature. This can be also an example of the preprocessing on data. The location of this computing node is important and can be categorized in different group. If devises send their reading directly to the cloud the processing is cloud computing and if a part of the processing happens where data is created in edge devices, Edge computing is being implemented. To design the best network that runs fast, smart and efficient, it is necessary to know the advantages and disadvantages of having the processing power in each site. The goal here can be finding the best threshold for dividing the Processing power between these sites.

2.4.1 Advantage of Data Processing at The Edge of The Network (Edge Computing)

Edge computing supports the applications in, which latency can lead to serious dysfunctions. Self driving cars can be an example in, which the decision can not wait the time needs to

send data to the cloud, have it processed and send back to actuators. Edge computing also plays an important role when it comes to Data anonymization by having the data encrypted before letting it leave the site. (Reale, 2017). Another advantage is the robust behavior to connectivity issues. Pre-process data before it is uploaded to the core datacentre for long-term processing is another take away of edge processing power. It is more efficient and cost friendly to have a central node capable of receiving readings from sensors than to have every node either sensor or actuator have the feature of sending or receiving data to and from internet. Computing node, beside being responsible for a part of processing is in charge of gathering the readings usually in a time series type (Team, 2018). To summarize Edge computing comes handy when:

- It is expensive to move all the data to the cloud
- Latency requirement for fast react. Edge Computing wins out over cloud processing when time sensitive events are happening.
- Regulatory reasons when you need to process the data where its produced without moving it off device.

2.4.2 Advantage of Data Processing Power at The Cloud: Motivating Factors

Fore most flexibility and most options in terms of compute and operating system. Whether or not edge computing does replace cloud computing is remained to be a subject of debate. Another advantage can demonstrate in a scenario where loss prevention is involved. Consider there is a theft of the device, so its reasonable to try to bring raw date back to the cloud as much as possible.

2.5 Conclusion

Achieving higher grow yield while the resources being fully optimized, is the main goal of IoT-based automation in smart greenhouse. Principle characters to enhance Industrial IoT platform performances are device management, data management, data analysis, security, industrial protocols, robustness, integration through development tools and APIs, predictive maintenance, remote access and energy optimization.

As it was demonstrated in this chapter, the proposed IoT prototype integrated into the robotic light fixture has the potential to feature all of these key players, however, in the proposed prototype the focus is on applying industrial and reliable development tools for data acquisition, data transmission, data logging, data analysis and remote access while utilizing the steady connection between all the components of the framework.

Chapter 3

Trajectory Generation for Steady-State Motion

In this chapter, control of a cable robotic LED light fixture is going to be discussed. A brief review of current control methods with the focus on trajectory planer Schemes is presented. System dynamics and its mathematical description is described and a trajectory is planned based on a five-order order polynomial motion profile, which can impose kinematics constraints such as position, velocity and acceleration.

3.1 Review of Current Trajectory Planer Schemes

The two-DOF cable-suspended parallel robot proposed in this prototype has a light panel, two DC motors and two cables in its structure. According to the number of cables and degree of freedom (DOFs), proposed robot is classified as a statically determined robot [41]. Each cable has its own DC motor as an actuator, which operates under torque control or voltage control of a proportional-integral-derivative (PID) controller. Proposed robot is designed to place the light panel in a desired height and angel with a smooth transnational and rotational movement while keeping the cables taut. To achieve this goal, a motion trajectory planner is proposed, which uses fifth-order polynomial trajectory profile of motion to impose constraints on position, velocity and acceleration. The advantages of employment a higher-order polynomial trajectory planner can be seen in [12].

To design this trajectory a State-space representation of a system is used. State-space representation of a system is a common and extremely powerful method of representing a system mathematically [19]. Some of the advantages of the state-space approach are the mathematical notation being simplified using vector equations, differential equations in state-space format being easier to solve with a digital computer, state-space formulation being applicable to both linear and non-linear systems as well as being applicable to multiple-input-multiple-output (MIMO) system. Therefore, the mathematical model of

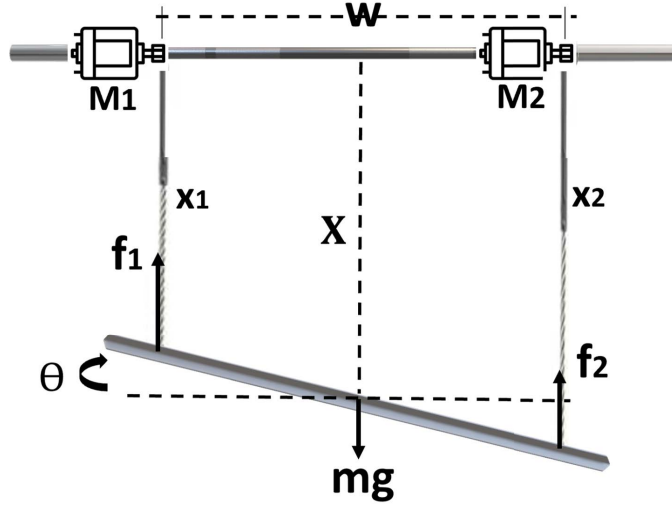


Figure 3.1: Free body diagram of the proposed cable-driven robotic light fixture

the proposed system is derived in a state-space form. A state-space feedback controller is designed for following the desired trajectory of tension forces.

Trajectory planning and trajectory tracking control are active fields in engineering and robotics research. In [51] a combination of sliding mode control and fuzzy logic control is developed to perform the trajectory tracking for a Cable Parallel Manipulator. Also, a servo controller is the employed method for tracking the designed trajectory path.

In[6] a reference trajectory describing a 21 cm by 21 cm square on a white board is successfully tracked using a 500 Hz visual feedback implemented with a 500 frames per second camera, an image processing algorithm using OpenCV library is proposed for a three-dof Cable-Driven Parallel Robot (CDPRS) which is implemented on a Raspberry Pi.

In [14] generic trajectory planner is proposed and validated using experimental result on a two-dof planar cable-suspended robot. This planner is based on a five-order polynomial to ensures continuity up to the acceleration level and can fulfil the dynamic constraints of the cables. Also, trajectory tracking is employed using two servo-controlled winches.

In the proposed prototype in this thesis, trajectory tracking is employed using DC motor's torque controller, state feedback controller and DC motor's voltage controller.

3.2 Trajectory Planer Scheme

The schematics and free body diagram of the two-dof cable-suspended robot proposed in this study is represented in fig 3.1. This figure illustrates the system's fixed reference frame with origin O, y-axis horizontal and pointing to the right, x-axis vertical and pointing downward and z axis perpendicular to x-axis and y-axis. This robot is consists of two cables, which connect two exit points M_1 and M_2 to a single end effector, which is a rectangular LED

panel with mass (m), width (w), height (h), length (l) as its characteristics. M_1 and M_2 are located at w from each other. Middle point between these two exit points is the origin location denoted by O . X is the distance between this origin point and the center of mass of the LED panel. Cables are parallel to each other and driven by two independent actuators, which are mounted at M_1 and M_2 . Each actuator is a DC motor. By controlling the pulling force on the two cables the distance X as well as the angle of the light panel can be controlled.

3.2.1 State-Space Representation of The Proposed Light Fixture

The concept of state of a dynamic system refers to a minimal set of variables, called state variables, which completely describe the system and its response to a given set of input data. A state-space model of a system has the following characteristics: (i) mathematical description of the system in terms of the minimum set of variables $x_i(t)$, $i = 1, \dots, n$, (ii) knowledge of values of these variables at the initial time t_0 and (iii) inputs of the system for the time $t \geq t_0$ are sufficient to predict the future state of the system. This definition indicates that the dynamic behavior of a state-driven system is entirely characterized by the response of a set of n variables $x_i(t)$, where n is defined as the order of the system [37].

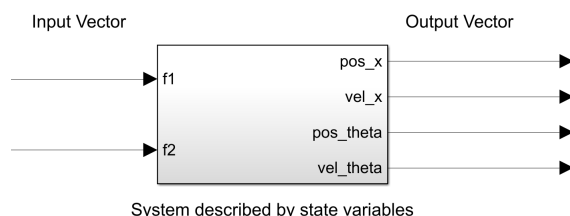


Figure 3.2: System described by state variables

State-space representation of the light fixture with the free body diagram shown in fig 3.1, can be obtained using the torque and tension equations. This system has two inputs $f_1(t)$ and $f_2(t)$, and four output variables $x(t)$, $\dot{x}(t)$, $\theta(t)$, $\dot{\theta}(t)$. Knowledge of its state variables at the initial time t_0 , which comes from initial sensor reading, and the knowledge of inputs $f_1(t)$ and $f_2(t)$ for $t \geq t_0$ is sufficient to determine all future behavior of the system. Fig 3.2 illustrates the inputs and outputs of this steady-state representation.

A standard form of equation of state is used throughout system dynamics and its mathematical description expressed as a set of first-order ordinary differential equations, known as state equations. To obtain state equations, kinematics of a two degrees of freedom (2-DOF) planar suspended cable-driven parallel robot (CDPR) is used.

As illustrated in fig 3.3, we can see the net torque τ produced at the center of mass of the panel is expressed as follows

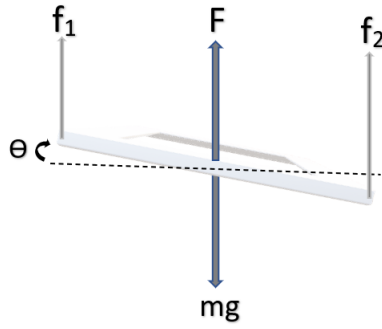


Figure 3.3: Forces diagram.

$$(f_1 \cos \theta)w - (f_2 \cos \theta)w = \tau \quad (3.1)$$

Where τ is net torque due to forces f_1 , f_2 acting on the light panel and w is the width of the panel. The equation can be written as bellow:

$$\tau = (f_1 - f_2)w \cos \theta \quad (3.2)$$

Net force acting on the center of mass of the light fixture is given as bellow:

$$f_{net} = mg - f_1 - f_2 \quad (3.3)$$

As the amount of torque required to produce an angular acceleration depends on the distribution of the mass of the object, torque is also equal to moment of inertia times angular acceleration.

$$\tau = I\ddot{\theta} \quad (3.4)$$

Where I is the moment of inertia of the light fixture along y axis and $\ddot{\theta}$ is the angular acceleration.

Newton's second law of motion states that the time rate of change of the momentum of a body is equal in both magnitude and direction to the force imposed on it. Therefore, we have:

$$f_{net} = m\ddot{x} \quad (3.5)$$

From 3.2 and 3.5 we have:

$$\begin{cases} (f_1 - f_2)w\cos\theta = I\ddot{\theta} \\ mg - f_1 - f_2 = m\ddot{x} \end{cases} \quad (3.6)$$

States are defined as x_1, x_2, x_3, x_4 , where x_1 is the distance x , x_2 is the rate of change in x , x_3 is the angle and x_4 is derivative of θ . In 3.7 the dynamic equations for these states are illustrated and 3.8 represents this dynamic in the Matrix form $\dot{x} = AX + BU$, which is the state space representation of a system where X is the state vector, A is the system matrix, B is the input matrix and U is the input vector.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \ddot{x} = \frac{mg-f_1-f_2}{m} = g - \frac{1}{m}(f_1 + f_2) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = \frac{(f_1-f_2)w\cos(x_3)}{I} = \frac{w\cos(x_3)}{I}(f_1 - f_2) \end{cases} \quad (3.7)$$

System dynamic in a matrix form is

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{-1}{m} & \frac{-1}{m} \\ 0 & 0 \\ \frac{w}{I} & \frac{w}{I} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + \begin{bmatrix} 0 \\ g \\ 0 \\ 0 \end{bmatrix} \quad (3.8)$$

Controllability and observability of this system needs to be taken into consideration. A linear time invariant (LTI) system is controllable if we can steer any initial state $x(t_0)$ to any final value $x(t_f)$ in a finite time t_f using a continues input $u(t)$ where $t_0 \leq t \leq t_f$. The linear system $\dot{x} = Ax + Bu$ is controllable if and only if the rank of the controllability matrix P which is defined as $[B \ AB \ A^2B \ A^3B \ \dots \ A^{n-1}B]$ is equal to n where n is the dimension of x . Controllability matrix of the system matrix in 3.8 is 4 therefore the system is controllable. A linear time invariant system is observable if the state at any instant can be determined by observing the output y over a finite interval of time. The linear system $\dot{x} = Ax + Bu$ is observable if and only if the rank of the observability matrix Q which is defined as $[C \ CA \ CA^2 \ \dots \ CA^{n-1}]$ is equal to n where n is the dimension of x . Observability will allow us to choose proper measurement such that the whole states can be estimated using the limited number of measurement resources. Observability matrix of the system matrix in 3.8 is only full rank if we consider the C matrix as $C = [1 \ 0 \ 0 \ 0; 0 \ 0 \ 1 \ 0]$ and have x and θ as the states which can be measured.

It should be noted that cable-suspended parallel robots use gravity to keep their cables taut, which results in reducing the number of actuators [27]. This necessitates the satisfying of cables working in tension only and not being able to push. Therefore, constraints must be imposed on the cartesian trajectory prescribed at the end effector such that f_1 and f_2 remain positive all the time.

From 3.6, we can obtain f_1 and f_2 as:

$$f_1 = \frac{1}{2} \left[\frac{I\ddot{\theta}}{w\cos\theta} + m(g - \ddot{x}) \right] \quad (3.9)$$

$$f_2 = \frac{1}{2} \left[\frac{-I\ddot{\theta}}{w\cos\theta} + m(g - \ddot{x}) \right] \quad (3.10)$$

As cables should be in tension and be pulled all the time therefore f_1 and f_2 should be always positive and we should plan the trajectory, which satisfy the following constraints:

$$\begin{cases} m(g - \ddot{x}) + \frac{I\ddot{\theta}}{w\cos\theta} > 0 \\ m(g - \ddot{x}) - \frac{I\ddot{\theta}}{w\cos\theta} > 0 \end{cases} \quad (3.11)$$

Let each panel be able to tile 90° . Therefore, θ_d is between 0° and 90° and we have $w\cos\theta_d > 0$ and we can write:

$$\begin{cases} m(g - \ddot{x}_d)w\cos\theta_d + I\ddot{\theta}_d > 0 \\ m(g - \ddot{x}_d)w\cos\theta_d - I\ddot{\theta}_d > 0 \end{cases} \quad (3.12)$$

Adding the two inequalities in 3.12 we can obtain that if $\ddot{x} < g$ the Constraints will be satisfied and forces f_1, f_2 will remain positive.

$$\ddot{x} < g \quad (3.13)$$

This bounded acceleration will result in

$$\ddot{\theta}_d < \frac{m(g - \ddot{x})}{I} w\cos(\theta_d) \quad (3.14)$$

Now if $\ddot{\theta}_d = \text{constant}$, we will have $\dot{\theta}_d = \ddot{\theta}_d t$ And $\theta_d = \frac{1}{2} \ddot{\theta}_d t_a^2$, Thus during the motion with $\ddot{\theta}_d$, starting from $\theta_d = 0$ to desired θ_d , with the travel time being t_a . $\cos\theta_d$ will be minimum at t_a .

$$\theta_d = \frac{1}{2} \ddot{\theta}_d t_a^2 \quad (3.15)$$

$$\ddot{\theta}_d < \frac{m(g - \ddot{x}_d)}{I} w \cos\left(\frac{1}{2}\ddot{\theta}_d t_a^2\right) \quad (3.16)$$

Therefore,

$$\frac{\ddot{\theta}_d}{w \cos\left(\frac{1}{2}\ddot{\theta}_d t_a^2\right)} < \frac{m(g - \ddot{x}_d)}{I} \quad (3.17)$$

Satisfying this condition ensures the cables being in tension while performing dynamic trajectories. Using 3.17, we can find the t_a that satisfy 3.13 given w , m , g , \ddot{x}_d , I and $\ddot{\theta}_d$.

In this application, we have two states that we need to design a trajectory for. One is the angle of the light fixture with respect to the horizontal plane. This angle can be varying from 0 to 90 degree. The other state is x , which is the distance between the light panel and the top of the light fixture (where the motors are mounted). To design the trajectory the 5th order polynomial motion profile is selected for both θ and x .

3.2.2 Trajectory Planner For Linear Movement (x)

Assume a 5th order polynomial motion profile for the trajectory of x :

$$x_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (3.18)$$

Assume the velocity and the acceleration at rest in both initial and final position. Therefore, we have:

$$\begin{cases} x(0) = x_0 \\ \dot{x}(0) = 0 \\ \ddot{x}(0) = 0 \end{cases} \quad (3.19)$$

$$\begin{cases} x(t_f) = x_f \\ \dot{x}(t_f) = 0 \\ \ddot{x}(t_f) = 0 \end{cases} \quad (3.20)$$

from [31] we can have the coefficients of polynomial 3.18 as follow:

$$a_0 = x_0 \quad (3.21)$$

$$a_1 = 0 \quad (3.22)$$

$$a_2 = 0 \quad (3.23)$$

$$a_3 = \frac{10(x_f - x_0)}{t_f^3} \quad (3.24)$$

$$a_4 = \frac{-15(x_f - x_0)}{t_f^4} \quad (3.25)$$

$$a_5 = \frac{6(x_f - x_0)}{t_f^5} \quad (3.26)$$

Therefore, we can derive the equations for $x_d(t)$, $\dot{x}_d(t)$ and $\ddot{x}_d(t)$ as follow:

$$x_d(t) = x_0 + \frac{10(x_f - x_0)}{t_f^3} t^3 - \frac{15(x_f - x_0)}{t_f^4} t^4 + \frac{6(x_f - x_0)}{t_f^5} t^5 \quad (3.27)$$

$$\dot{x}_d(t) = \frac{30(x_f - x_0)}{t_f^3} t^2 - \frac{60(x_f - x_0)}{t_f^4} t^3 + \frac{30(x_f - x_0)}{t_f^5} t^4 \quad (3.28)$$

$$\ddot{x}_d(t) = \frac{60(x_f - x_0)}{t_f^3} t - \frac{180(x_f - x_0)}{t_f^4} t^2 + \frac{120(x_f - x_0)}{t_f^5} t^3 \quad (3.29)$$

Let $\tau = \frac{t}{t_f}$, we have:

$$x_d(\tau) = x_0 + 10(x_f - x_0)\tau^3 - 15(x_f - x_0)\tau^4 + 6(x_f - x_0)\tau^5 \quad (3.30)$$

$$\dot{x}_d(\tau) = \frac{30(x_f - x_0)}{t_f} \tau - \frac{60(x_f - x_0)}{t_f} \tau^2 + \frac{30(x_f - x_0)}{t_f} \tau^3 \quad (3.31)$$

$$\ddot{x}_d(\tau) = \frac{60(x_f - x_0)}{t_f^2} \tau - \frac{180(x_f - x_0)}{t_f^2} \tau^2 + \frac{120(x_f - x_0)}{t_f^2} \tau^3 \quad (3.32)$$

From 3.13 we have:

$$\ddot{x}_d(\tau) = \frac{60(x_f - x_0)\tau}{t_f^2} [1 - 3\tau + 2\tau^2] < g \quad (3.33)$$

To obtain the range of t_f we first should find the maximum of function $[\tau - 3\tau^2 + 2\tau^3]$ and plug it in 3.33

$$t_f^2 > 60(x_f - x_0) \frac{MAX(\tau - 3\tau^2 + 2\tau^3)}{g} \quad (3.34)$$

$$t_f > \sqrt{60(x_f - x_0) \frac{MAX(\tau - 3\tau^2 + 2\tau^3)}{g}} \quad (3.35)$$

As $\tau = \frac{t}{t_f}$, τ will remain between 0 and 1 ($0 \leq \tau \leq 1$). Therefore we have;

$$MAX(\tau - 3\tau^2 + 2\tau^3) = 0.0962 \quad (3.36)$$

Choosing t_f that fulfill the condition in 3.35 will result in having tension forces positive between t_0 and t_f .

3.2.3 Trajectory Planner For Rotational Movement (θ)

Assume a 5th order polynomial motion profile for the trajectory of θ :

$$\theta_d(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \quad (3.37)$$

Assume velocity and acceleration at rest in both initial and final position. We have:

$$\begin{cases} \theta(0) = \theta_0 \\ \dot{\theta}(0) = 0 \\ \ddot{\theta}(0) = 0 \end{cases} \quad (3.38)$$

$$\begin{cases} \theta(t_f) = \theta_f \\ \dot{\theta}(t_f) = 0 \\ \ddot{\theta}(t_f) = 0 \end{cases} \quad (3.39)$$

from [31] we can have the coefficients of polynomial 3.37 as follow:

$$a_0 = \theta_0 \quad (3.40)$$

$$a_1 = 0 \quad (3.41)$$

$$a_2 = 0 \quad (3.42)$$

$$a_3 = \frac{10(\theta_f - \theta_0)}{t_f^3} \quad (3.43)$$

$$a_4 = \frac{-15(\theta_f - \theta_0)}{t_f^4} \quad (3.44)$$

$$a_5 = \frac{6(\theta_f - \theta_0)}{t_f^5} \quad (3.45)$$

$$\theta_d(t) = x_0 + \frac{10(\theta_f - \theta_0)}{t_f^3} t^3 - \frac{15(\theta_f - \theta_0)}{t_f^4} t^4 + \frac{6(\theta_f - \theta_0)}{t_f^5} t^5 \quad (3.46)$$

$$\dot{\theta}_d(t) = \frac{30(\theta_f - \theta_0)}{t_f^3} t^2 - \frac{60(\theta_f - \theta_0)}{t_f^4} t^3 + \frac{30(\theta_f - \theta_0)}{t_f^5} t^4 \quad (3.47)$$

$$\ddot{\theta}_d(t) = \frac{60(\theta_f - \theta_0)}{t_f^3} t - \frac{180(\theta_f - \theta_0)}{t_f^4} t^2 + \frac{120(\theta_f - \theta_0)}{t_f^5} t^3 \quad (3.48)$$

Let $\tau = \frac{t}{t_f}$, we have:

$$\theta_d(\tau) = \theta_0 + 10(\theta_f - \theta_0)\tau^3 - 15(\theta_f - \theta_0)\tau^4 + 6(\theta_f - \theta_0)\tau^5 \quad (3.49)$$

$$\dot{\theta}_d(\tau) = \frac{30(\theta_f - \theta_0)}{t_f} \tau - \frac{60(\theta_f - \theta_0)}{t_f} \tau^2 + \frac{30(\theta_f - \theta_0)}{t_f} \tau^3 \quad (3.50)$$

$$\ddot{\theta}_d(\tau) = \frac{60(\theta_f - \theta_0)}{t_f^2} \tau - \frac{180(\theta_f - \theta_0)}{t_f^2} \tau^2 + \frac{120(\theta_f - \theta_0)}{t_f^2} \tau^3 \quad (3.51)$$

Where $0 \leq \tau \leq 1$. Hence the constraint on 3.17 is

$$\frac{\theta_d}{\cos(\theta_d)} < \frac{m(g - \ddot{x}_d)}{I} \quad (3.52)$$

By plugging 3.51 into 3.52 we have:

$$\frac{1}{t_f^2} \frac{60\tau(\theta_f - \theta_0)[1 - 3\tau + 2\tau^2]}{\cos(\theta_0 + 10(\theta_f - \theta_0)\tau^3 - 15(\theta_f - \theta_0)\tau^4 + 6(\theta_f - \theta_0)\tau^5)} < \frac{m(g - \ddot{x}_d)}{I} \quad (3.53)$$

Choosing t_f that fulfill the condition in 3.34 will result in having the tension forces positive between t_0 and t_f while performing the proposed 5 order trajectory.

3.3 Conclusion

This chapter proposed a cable robotic LED light fixture and its mathematical representation in a State-Space form. Using this representation a trajectory planner with five-order order polynomial motion profile is proposed. This mathematics shows the feasibility of controlling the proposed robotic system in order to place the light panel in a desired height and a desired angel with a smooth transnational and rotational movement while keeping the cables taut. The limitation and considerations for choosing the duration of trajectory is also provided.

Chapter 4

Trajectory Tracker Design

In this chapter, a DC motor current controller (cascaded control) and a DC motor voltage controller are proposed as the trajectory tracker unit. To precisely track specified trajectories, or be able to follow more general trajectories, many tracking control algorithms have been proposed. A robotic trajectory control has either electric motors and solenoids or it uses hydraulic system as actuator. Trajectory tracker controller calculates the control signal for the actuators based on their mathematical model usually in a form of an electrical signal. Actuators will convert the electrical signal into kinetic energy (rotational or linear motion) for the practical operations. Trajectory tracking controllers used in this chapter are proposed to make sure that the desired trajectory is being followed.

4.1 Cascaded Trajectory Tracking Design (Motor Current Control)

In a cascade control arrangement, there are two (or more) controllers with the output of the outer controller (first controller) providing the set point for the inner controller (second controller), the feedback loop for one controller nestling inside the other [7]. Fig 4.1 shows the block diagram for the cascade controller.

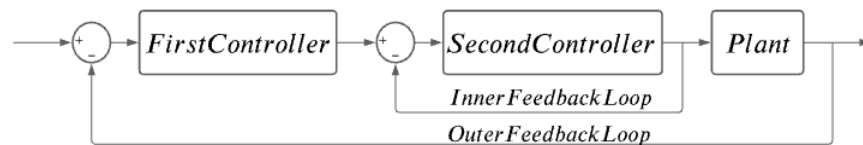


Figure 4.1: Block diagram of a cascaded controller.

In the previous chapter desired reference value in each step can be calculated by the trajectory planner. If we define h_r , \dot{h}_r , θ_r and $\dot{\theta}_r$ as the reference for desired height, desired

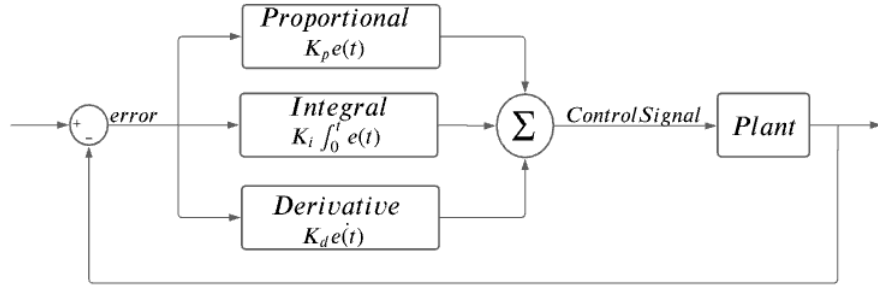


Figure 4.2: Block diagram of a feedback system with a PID controller.

rate of change for height, desired angle θ and desired rate of change for θ respectively, we can form our desired state vector X_d . Using the readings from distance sensor, gyro sensor and an observer we can calculate the error for all the states at each step time. First controller unit will calculate the torque and current needed for each motor. Second controller calculates the control signal to achieve that torque and current at each time step.

4.1.1 DC Motor Current Controller or Inner Controller Design

Speed control of the motor is required for tracking the desired current resulting in tracking the desired height and angle trajectory. These two controllers in charge of height/angle control and motor speed control are in series and cascaded. The proportional integral subsidiary controller (PID controller) has been broadly utilized for monitoring the armature current and controlling the DC motor speed. fig 4.2 illustrate the feedback PID Controller diagram, which generally utilized as a part of mechanical control frameworks. PID controller fundamentally analyze the error signal between measured control value and desired value and this error signal is used to calculate the control command. In this controller, reference point fed to controller. Using sensory feedback, the real control value will be monitored and its distance from desired value e , error over time *integrale* and rate of change in the error \dot{e} will gain the desired gain based on the desired respond and fed to the plant as the control input. equation 4.1 shows the mathematical expression of a PID controller.

$$U = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}} \quad (4.1)$$

Where P, I and D denote the coefficients for the proportional, integral, and derivative terms respectively.(tuning of the PID parameters using experimental techniques)

In the proposed control design here, motors current controller is a PID controller, which will be the second controller in the cascaded design in 4.1 Model of a DC motor is shown in fig 4.3.

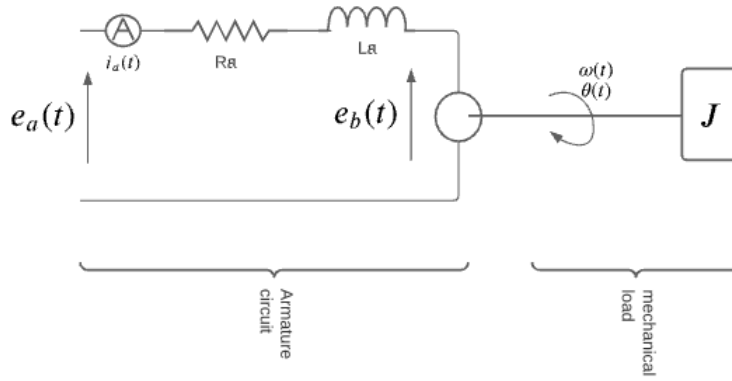


Figure 4.3: Model of a DC motor

The model for a DC-motor in frequency domain is derived as below: We have armature circuit as:

$$I_a(s) = \frac{1}{R_a + L_a s} (E_a(s) - E_b(s)) \quad (4.2)$$

where I_a is the armature current, R_a and L_a are the armature resistance and inductance respectively and applied voltage is E_a and Back EMF is E_b . Connection between mechanical/electrical parts are motor torque and back EMF as bellow:

$$T(s) = K_\tau I_a(s) \quad (4.3)$$

$$E_b(s) = K_b \Omega(s) \quad (4.4)$$

we have mechanical load as

$$\Omega(s) = \frac{1}{J s + B} (T(s) - T_L(s)) \quad (4.5)$$

Angular position will be

$$\Theta(s) = \frac{1}{s} \Omega(s) \quad (4.6)$$

Having the above equations we can achieve the block diagram of DC Motor and so we have the inner feedback loop, which is a motor speed control. the desired Current for each motor will come from the first controller in the cascaded scheme.

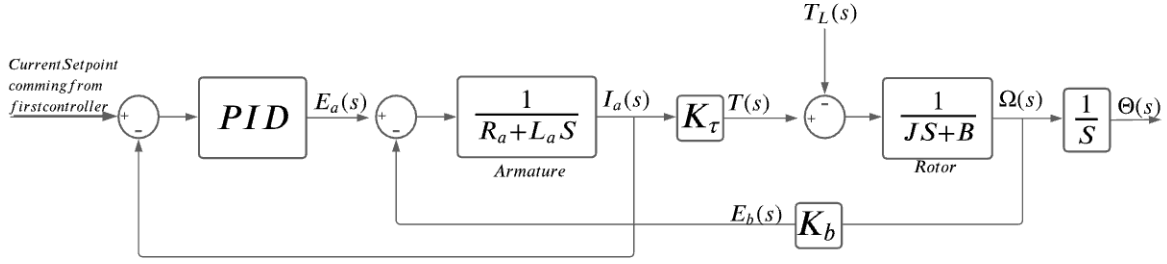


Figure 4.4: Second Controller (current control loop) in the cascaded trajectory tracker

4.1.2 Force Controller or Outer Controller Design

4.1.2.1 PD Control

A PD controller can calculate the desired current for each motor, which will be fed to the current control loop of each motor.

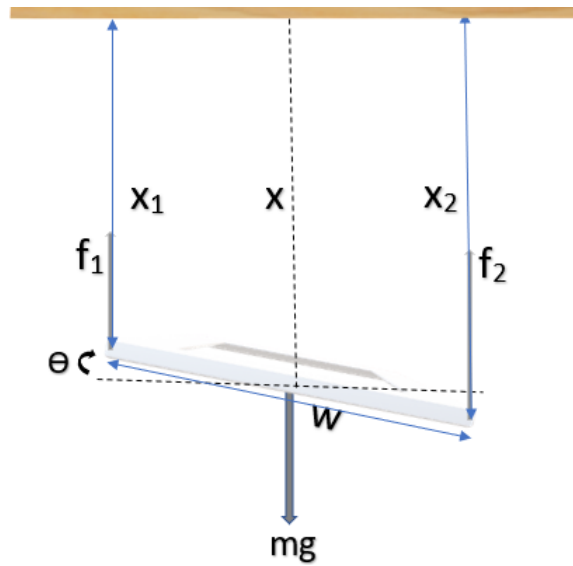


Figure 4.5: Control values and force diagram

Lets define u_x as bellow

$$u_x = f_1 + f_2 \quad (4.7)$$

Using 3.6, we can write:

$$m\ddot{x} = mg - u_x \quad (4.8)$$

if we define v_x as below

$$v_x = g - \frac{u_x}{m} \quad (4.9)$$

Then we have,

$$\ddot{x} = v_x \quad (4.10)$$

$$\ddot{x} - v_x = 0 \quad (4.11)$$

Lets define v_x as a quadratic equation

$$v_x = \ddot{x}_r + k_{dx}(\dot{x}_r - \dot{x}) + k_{px}(x_r - x) \quad (4.12)$$

so we have,

$$(\ddot{x}_r - \ddot{x}) + k_{dx}(\dot{x}_r - \dot{x}) + k_{px}(x_r - x) \quad (4.13)$$

Therefore, by defining $e_x = x_r - x$, we have:

$$\ddot{e}_x + k_{dx}\dot{e}_x + k_{px}e_x = 0 \quad (4.14)$$

using 4.9 we can find u_x as bellow:

$$u_x = g - m(\ddot{x}_r + k_{dx}(\dot{x}_r - \dot{x}) + k_{px}(x_r - x)) \quad (4.15)$$

Lets define

$$u_\theta = (f_1 - f_2) \cos(\theta) \quad (4.16)$$

Using 3.6, we can write:

$$I\ddot{\theta} = \cos(\theta)wu_\theta \quad (4.17)$$

if we define v_θ as below

$$v_\theta = \frac{w \cos(\theta)}{I}u_\theta \quad (4.18)$$

then,

$$\ddot{\theta} = v_\theta \quad (4.19)$$

$$\ddot{\theta} - v_{\theta} = 0 \quad (4.20)$$

Lets define v_{θ} as a quadratic equation

$$v_{\theta} = \ddot{\theta}_r + k_{d\theta}(\dot{\theta}_r - \dot{\theta}) + k_{p\theta}(\theta_r - \theta) \quad (4.21)$$

so we have,

$$(\ddot{\theta}_r - \ddot{\theta}) + k_{d\theta}(\dot{\theta}_r - \dot{\theta}) + k_{p\theta}(\theta_r - \theta) \quad (4.22)$$

Therefore, by defining $e_{\theta} = \theta_r - \theta$, we have:

$$\ddot{e}_{\theta} + k_{d\theta}\dot{e}_{\theta} + k_{p\theta}e_{\theta} = 0 \quad (4.23)$$

using 4.18 we can find u_{θ} as bellow:

$$u_{\theta} = \frac{I}{w \cos(\theta)} v_{\theta} \quad (4.24)$$

$$u_{\theta} = \frac{I}{w \cos(\theta)} (\ddot{\theta}_r + k_{d\theta}(\dot{\theta}_r - \dot{\theta}) + k_{p\theta}(\theta_r - \theta)) \quad (4.25)$$

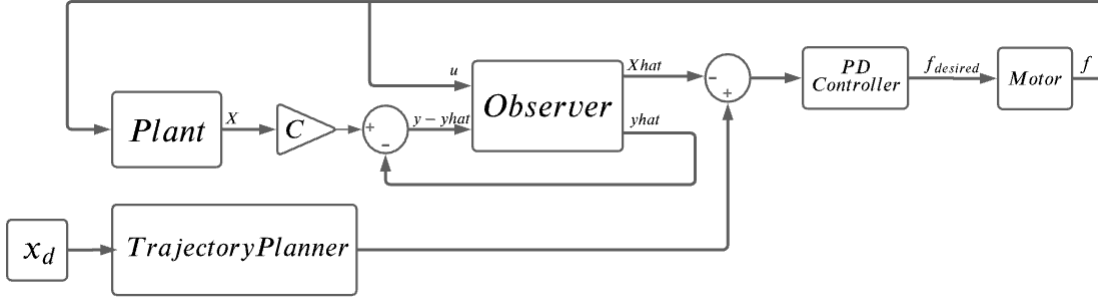


Figure 4.6: Block diagram of the implementation of observer and PD force control for the trajectory tracker

next we need to calculate desired motor current i_{d1} and i_{d2} using u_x and u_{θ} . We have defined $u_x = f_1 + f_2$ and $u_{\theta} = (f_1 - f_2) \cos(\theta)$ in the previous subsections, therefore, we can find f_1 and f_2 using these two equations. Fig 4.6 shows the block diagram of the implementation of observer and PD force control for the trajectory tracker.

$$\begin{cases} u_x = f_1 + f_2 \\ u_\theta = \cos(\theta)(f_1 - f_2) \end{cases} \quad (4.26)$$

Therefore,

$$\begin{cases} f_1 = \frac{1}{2}(u_x \cos(\theta) + u_\theta) \\ f_2 = \frac{1}{2}(u_x \cos(\theta) - u_\theta) \end{cases} \quad (4.27)$$

having the desired force we can calculate the desired current for each motor at given time step.

$$\begin{cases} i_{d1} = \frac{f_1}{\alpha} \\ i_{d2} = \frac{f_2}{\alpha} \end{cases} \quad (4.28)$$

Where i_{d1} is the desired current for M_1 and i_{d2} is the desired current for M_2 . α is a constant value of motor. These desired current will be fed into the current loop control of each motor, which is described in the following sections.

4.1.2.2 Optimal State-Space Control

For designing the State-Space Controller, first we should talk about the controllability and observability of the system. A linear time invariant (LTI) system is controllable if we can steer any initial state $x(t_0)$ to any final value $x(t_f)$ in a finite time t_f using a piece-wise continues input $u(t)$ where $t_0 < t \leq t_f$. The linear system $\dot{x} = Ax + Bu$ is controllable if and only if the rank of the controllability matrix P , which is defined as $\begin{bmatrix} B & AB & A^2B & A^3B & \dots & A^{n-1}B \end{bmatrix}$ is equal to n where n is the dimension of x . Controllability matrix of the system matrix in 3.8 is 4 therefore the system is controllable. A linear time invariant system is observable if the state at any instant can be determined by observing the output y over a finite interval of time. The linear system $\dot{x} = Ax + Bu$ is observable if and only if the rank of the observability matrix Q , which is defined as $\begin{bmatrix} C & CA & CA^2 & \dots & CA^{n-1} \end{bmatrix}'$ is equal to n where n is the dimension of x . Observability will allow us to choose proper measurement such that the whole states can be estimated using the limited number of measurement resources. Observability matrix of the system matrix in 3.8 is only full rank if we consider the C matrix as bellow to have x and θ as the states, which can be measured.

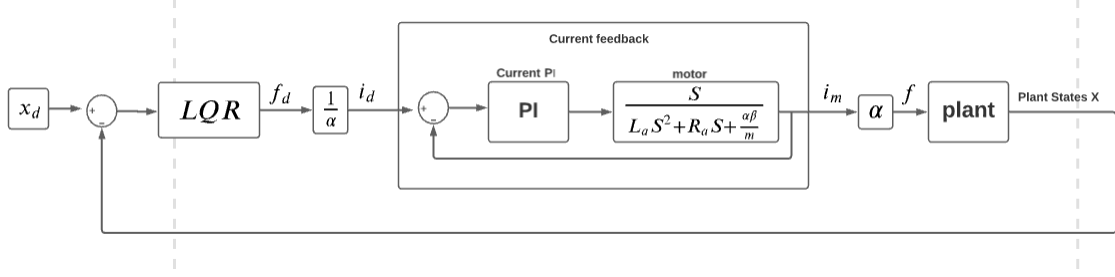


Figure 4.7: Block diagram of the cascaded SQL force controller and PI current controller

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.29)$$

To control the characteristics of the response of the system, a Full state feedback (FSF), or pole placement method is employed to place the closed-loop poles of a plant in desired locations in the s-plane. The system must be considered controllable in order to implement this method. As the Two-DOF Cable-Suspended Parallel Robot with given dynamics is controllable, we can place the poles in desired location and have the desired close loop characteristic equation.

Pole Placement based on LQR: The linear quadratic regulator (LQR) is a well-known design technique that provides practical feedback gains. It will help the designer achieve a compromise between good regulation and reasonably sized inputs. In other words Optimal control means finding the control law that minimizes the given performance index. For state regulator problem, performance index may be $J = \int (x^t q x + u^t r u) dt$ So, we must find $u(t)$ that minimizes J. Using these errors the LQR controller can calculate the forces needed for each motor for trajectory tracking. Fig 4.7 shows the control block diagram of these cascaded controllers.

Pole Placement based on time response: In a second-order system with transfer function $H(s)$, $s^2 + 2\xi\omega_n s + \omega_n^2 = 0$ will give two dominant poles s_1 and s_2 .

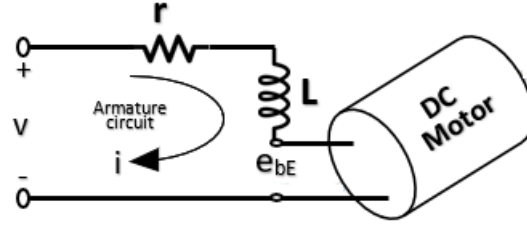


Figure 4.8: The electric equivalent circuit of the armature and the free-body diagram of the rotor

4.2 Trajectory Tracking using DC motor voltage control Design

Schematic of electromechanical system can be seen in fig 4.8. For this electromechanical model, we will assume that the armature inductance is neglectable ($L \approx 0$). To do so, the current in the armature should be able to build up quickly as the current in the armature winding reverses based on the commutator segment the specific coil is touching at that moment and it makes it look have AC voltage source although The supply is DC voltage.

Mathematical modeling of DC motor having voltage source v as input will be as follow:

$$v = ri + e_b \quad (4.30)$$

Where i is armature current, v is input voltage, r is armature resistance and e_b is the back-emf. Back emf e_b is the generator output of a motor, and is proportional to the motor's angular velocity ω .

$$e_b(t) = k_b \frac{d\theta_m(t)}{dt} = k_b \omega_m(t) \quad (4.31)$$

$$e_b(t) \propto \omega(t) \quad (4.32)$$

Motor's angular velocity will be converted to linear velocity of the light panel \dot{x} using a cord, which is wrapped around the rim as can be seen in fig 4.9. Therefore, back emf of the motor is proportional to the LED panel's linear velocity.

$$e_b \propto \dot{x} \quad (4.33)$$

Here we have two DC motor in, which we assume the coefficients between e_b and \dot{x} are the same and are a constant $\alpha/2$. This constant depends on motor constant, gear efficiency

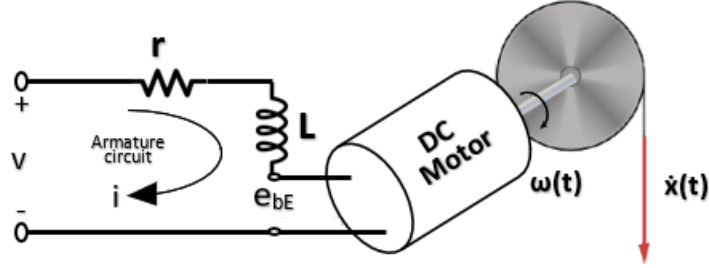


Figure 4.9: Motor's angular velocity to linear velocity of the light panel conversion

and gear ratio. Therefore we have:

$$e_b = \frac{\alpha}{2} \dot{x} \quad (4.34)$$

We also assume that both motors have the same parameters. Equation 4.30 for the two motor we have in this proposed setup will be as below:

$$\begin{cases} v_1 = ri_1 + \frac{\alpha}{2} \dot{x}_1 \\ v_2 = ri_2 + \frac{\alpha}{2} \dot{x}_2 \end{cases} \quad (4.35)$$

In the previous section, the desired reference value for the distance of the LED panel from the top of the structure where motors are mounted as well as its angle can be calculated in each step, which are $x_r(t)$ as the reference for height and $\theta_r(t)$ as the reference for θ . We can calculate the error for all the states at each step time. Using these errors a pi controller can calculate the desired current for each motor, which will be fed to the current control loop of each motor. States will be updated using the reading from distance and gyro sensor as well as an observer.

sine function gives the ratio of the length of the opposite side to the length of the hypotenuse.

As can be seen in 4.5 and using the trigonometric functions x_2-x_1 has direct relationship with the *sin* of rotational movement.

$$x_2 - x_1 = w \sin \theta \quad (4.36)$$

The Maclaurin expansion (the Taylor expansion about 0) of the sin trigonometric function is

$$\sin \theta = \theta - \frac{\theta^3}{6} + \frac{\theta^5}{120} - \frac{\theta^7}{5040} + \dots \quad (4.37)$$

Where θ is the angle in radians. The angles at, which the relative error is less than 1% for sin function are $\theta < 0.2441$ radians (13.99°). Therefore for any angle less than 13.99° , we can approximate $\sin \theta$ with θ . ($\sin \theta \approx \theta$). So, 4.38 can be approximated as follows

$$x_2 - x_1 \approx w\theta \quad (4.38)$$

Also we have:

$$x = \frac{x_1 + x_2}{2} \quad (4.39)$$

In 4.5 using Newton's second law of motion, we can write:

$$\ddot{x} = g - \frac{f_1 + f_2}{m} \quad (4.40)$$

$$I\ddot{\theta} = (f_1 - f_2)w \cos(\theta) \quad (4.41)$$

By adding the equations in 4.59 we have:

$$v_1 + v_2 = r(i_1 + i_2) + \frac{\alpha}{2}(x_1 + x_2) \quad (4.42)$$

$$v_1 + v_2 = r\beta(f_1 + f_2) + \alpha\dot{x} \quad (4.43)$$

$$v_1 - v_2 = r\beta(f_1 - f_2) - \frac{\alpha}{2}(x_2 - x_1) \quad (4.44)$$

Considering 4.38, we can write:

$$v_1 - v_2 = r\beta(f_1 - f_2) - \frac{\alpha}{2}\omega\dot{\theta} \quad (4.45)$$

By plugging 4.40 and 4.43 we can write

$$\ddot{x} = g - \frac{1}{m} \frac{1}{r\beta} (v_1 + v_2 - \alpha \dot{x}) \quad (4.46)$$

$$\ddot{x} - \frac{\alpha}{mr\beta} \dot{x} = g - \frac{1}{mr\beta} (v_1 + v_2) \quad (4.47)$$

By plugging 4.41 and 4.45 we can write

$$I\ddot{\theta} = \frac{1}{r\beta} (v_1 - v_2 + \frac{\alpha}{2} \omega \dot{\theta}) \omega \cos \theta \quad (4.48)$$

$$I\ddot{\theta} - \frac{\alpha\omega^2}{2r\beta} \dot{\theta} \cos \theta = \frac{\omega \cos \theta}{r\beta} (v_1 - v_2) \quad (4.49)$$

if $\cos \theta \approx 1$ then

$$\ddot{\theta} - \frac{\alpha\omega^2}{2r\beta} \dot{\theta} = \frac{\omega}{r\beta I} (v_1 - v_2) \quad (4.50)$$

4.2.1 Definition and calculation of v_x

Lets define $v_1 + v_2 = v_x$

$$\ddot{x} - \frac{\alpha}{mr\beta} \dot{x} = g - \frac{1}{mr\beta} v_x \quad (4.51)$$

Lets define v_x as a quadratic equation as bellow

$$v_x = -mr\beta(\ddot{x}_d + k_{dx}(\dot{x}_d - \dot{x}) + k_{px}(x_d - x) + k_{ix} \int_0^t (x_d - x) dt - g - \frac{\alpha}{mr\beta} \dot{x}) \quad (4.52)$$

Then equation 4.51 can be re-written as bellow

$$(\ddot{x}_d - \ddot{x}) + k_{dx}(\dot{x}_d - \dot{x}) + k_{px}(x_d - x) + k_{ix} \int_0^t (x_d - x) dt = 0 \quad (4.53)$$

Therefore, by defining $e_x = x_d - x$, we have:

$$\ddot{e}_x + k_{dx} \dot{e}_x + k_{px} e_x + k_{ix} \int_0^t e_x dt = 0 \quad (4.54)$$

4.2.2 Definition and calculation of v_θ

let $v_1 - v_2 = v_\theta$

$$\ddot{\theta} - \frac{\alpha\omega^2}{2r\beta}\dot{\theta} = \frac{\omega}{r\beta I}v_\theta \quad (4.55)$$

Lets define v_θ as a quadratic equation as bellow

$$v_\theta = \frac{rI\beta}{\omega}(\ddot{\theta}_d + k_{d\theta}(\dot{\theta}_d - \dot{\theta}) + k_{p\theta}(\theta_d - \theta) + k_{i\theta} \int_0^t (\theta_d - \theta)dt - \frac{\alpha\omega^2}{2rI\beta}\dot{\theta}) \quad (4.56)$$

Then equation 4.55 can be re-written as bellow

$$(\ddot{\theta}_d - \ddot{\theta}) + k_{d\theta}(\dot{\theta}_d - \dot{\theta}) + k_{p\theta}(\theta_d - \theta) + k_{i\theta} \int_0^t (\theta_d - \theta)dt = 0 \quad (4.57)$$

Therefore, by defining $e_\theta = \theta_d - \theta$, we have:

$$\ddot{e}_\theta + k_{d\theta}\dot{e}_\theta + k_{p\theta}e_\theta + k_{i\theta} \int_0^t e_\theta dt = 0 \quad (4.58)$$

4.2.3 Calculating motor voltage for each motor

$$\begin{cases} v_1 = \frac{v_x + v_\theta}{2} \\ v_2 = \frac{v_x - v_\theta}{2} \end{cases} \quad (4.59)$$

These desired voltages will fed into the voltage loop control of each motor.

4.3 Conclusion

In this section, tracking of the trajectory in a 2-DOF robotic manipulator is presented. Tracking time, initial and final positions co-ordinates are specified. Three control methods are is used to track the initial to final angular position of the joint generated by the trajectory generator. Simulations results show that controller control the end-effector effectively, and end-effector reaches from initial to final coordinates effectively.

Chapter 5

Simulation Studies

For evaluating the performance of the purposed trajectory planner and trajectory trackers, simulation studies were conducted and the results are demonstrated. The control systems are designed and evaluated via employment of MATLAB/Simulink software. In this chapter, the simulation studies for validating the control systems mentioned in previous chapters are going to be presented.

5.1 Discretization

For digital implementation, Simulinks' Digital Clock block 5.1, which Outputs simulation time at specified sampling interval is used. This block outputs the simulation time only at the specified sampling interval. Other times, the block holds the output at the previous value. This can simply digitally discretized the time domain. In time-domain, a general dynamic system is expressed as follows

$$\begin{aligned}\dot{x}(t) &= f(t, u, x) \\ y &= g(t, u, x)\end{aligned}\tag{5.1}$$

where x is the state of the system, u is the input of the system and y is the output of the system.

In (5.1), the dynamic system is described by sets of states expressed by first order differential equations. This dynamic system is then discretized digitally by approximating the continuous-time differential equations in discrete time as follows

$$\begin{aligned}x_{n+1} &= f'(t_n, u_n, x_n) \\ y_n &= g'(t_n, u_n, x_n)\end{aligned}\tag{5.2}$$

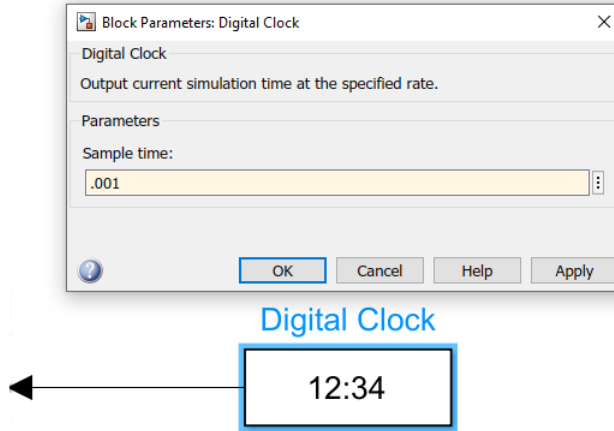


Figure 5.1: The block diagram of the simulink's digital clock

where n is an integer representing the discretized time, f' and g' are the approximation to calculate the next value of the state. The most important aspect of discretizing the control system is to make sure the method can accurately approximate the continuous dynamics. Here, The specified sampling interval of 10 millisecond is chosen to make the simulation as close as possible to the experimental setup in which the lowest response time amongst the sensors is 10 millisecond.

5.2 Simulation of designed trajectory planner

As it was described in chapter three motion profile for the trajectory of x and θ are

$$x_d(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \quad (5.3)$$

$$\theta_d(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \quad (5.4)$$

The coefficients of the polynomial can be calculated using initial value, desired final value, the duration of the motion and the sampling time (t). Digital Clock provides t for each step. Using a MATLAB function, which gets these values as input, and outputs system states x , \dot{x} , θ_d , $\dot{\theta}_d$, we can have the desired motion as the reference to trajectory tracker. 5.2 illustrate this subsystem in MATLAB.

An example of trajectory planner for translational motion from 0.5 meter to 0.8 meter and rotational motion from 0.1 radian to 0.15 radian during 20 second are illustrated in 5.3 and 5.4 respectively.

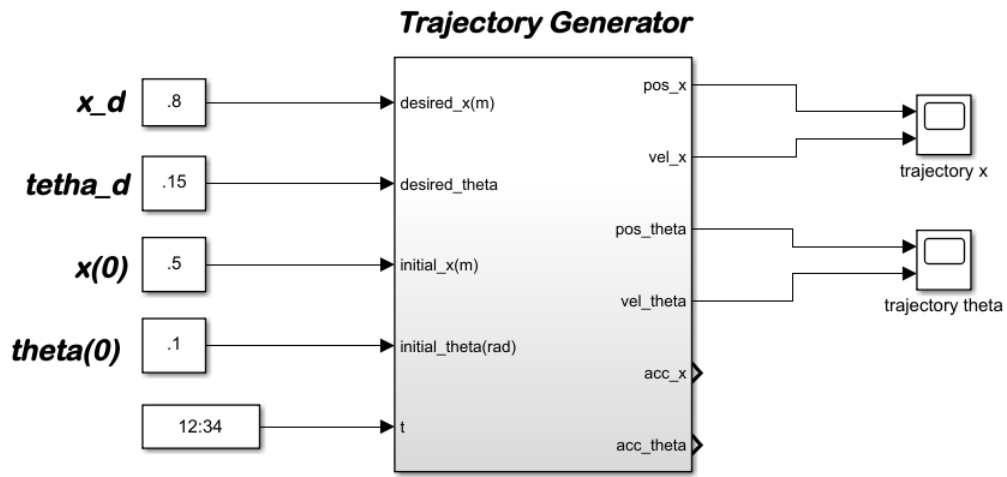


Figure 5.2: Trajectory planner block in Simulink

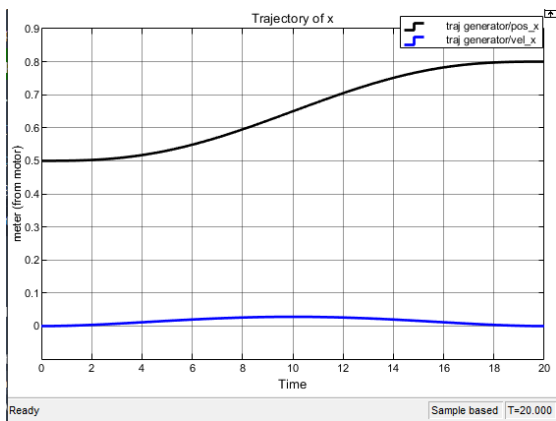


Figure 5.3: Trajectory generation of a linear path from 0.5m to 0.8m

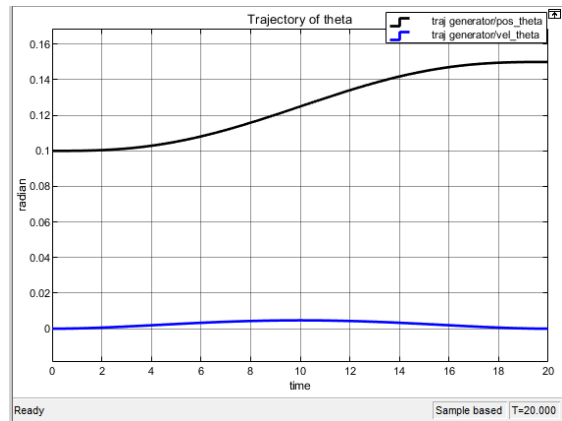


Figure 5.4: Trajectory generation of a rotational path from 0.1rad to .15rad

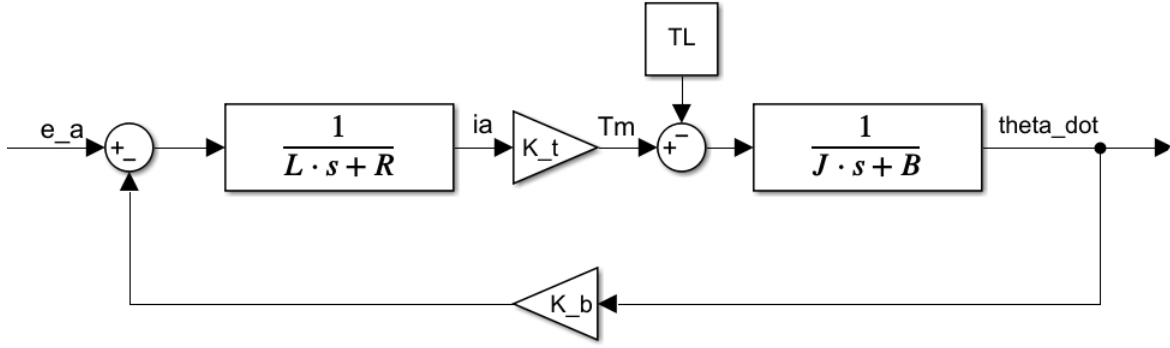


Figure 5.5: DC motor block diagram

Motor Parameters	Value
K_b	0.01 N.m/A
K_a	1.2 N.m/A
R	10 Ω
L	0.008 H
J	0.001 $K_g m^2/rad$
B	0.001 $K_g m^2/rad$

Table 5.1: Motor parameters

5.3 MATLAB Simulations for Cascaded Trajectory Tracking Design

Simulation of the DC motor is a crucial part in trajectory tracking simulation. The model for a DC-motor in frequency domain mentioned in Chapter 4.1.1 4.2 is used to simulate the DC motor, which is used as actuator in the process Fig 5.5. DC motor's Parameters are shown in table 5.1

To be able to simulate the LED light panel its physical attributes are measured and illustrated in table 5.2.

Parameters	Quantity	Simulation Value
m (Kg)	mass of the light panel	5
w (m)	Width of the light panel	1.2
h (m)	Height of the light panel	0.04
l (m)	length of the light panel	0.4
I $kg * m^2$	Moment of Inertia (Rectangle parallel to y-axis)	$m*(h^2 + l^2)/12$

Table 5.2: Parameters of the light fixture

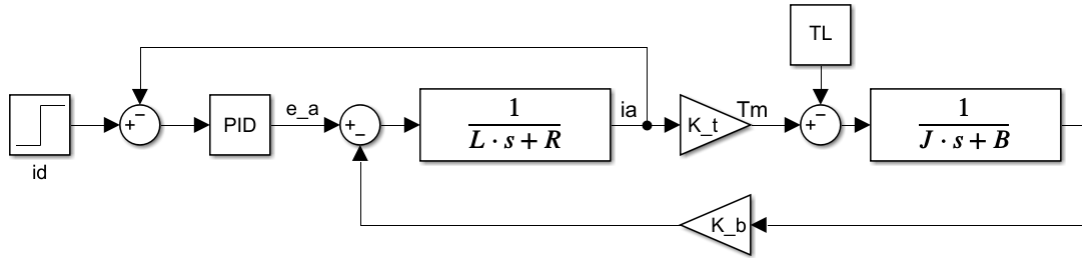


Figure 5.6: MATLAB Simulink for close loop DC motor current control

5.3.1 MATLAB Simulation for The DC Motor Current PID Control (Inner Controller in Cascaded Trajectory Tracking Design)

The Simulink model of a DC motor current control system using PID controller is shown in fig 5.6. For evaluating the PID controller we use a PID controller block $(P + I\frac{1}{s} + D\frac{N}{1+N\frac{1}{s}})$. P, I and D are proportional, integral and derivative gain constants respectively. Performance of the PID current controller purposed in Chapter 4.1 4.1 is demonstrated in this section.

To check the controller design, a step change in the current reference is used. This is implemented via a constant and step source blocks. The value of Kp_i , Ki_i and Kd_i in MATLAB prompt are tuned and shown in table 5.3.

Parameters of PID controller	Values
K_p	10
K_i	200
K_d	0

Table 5.3: Parameters of the motor current PID controller

Simulation has been ran for a reference current of 1A (for 3 sec, time step 1e-4). The response of the system to the step change in the reference current and its transient performance is shown in Fig 5.7.

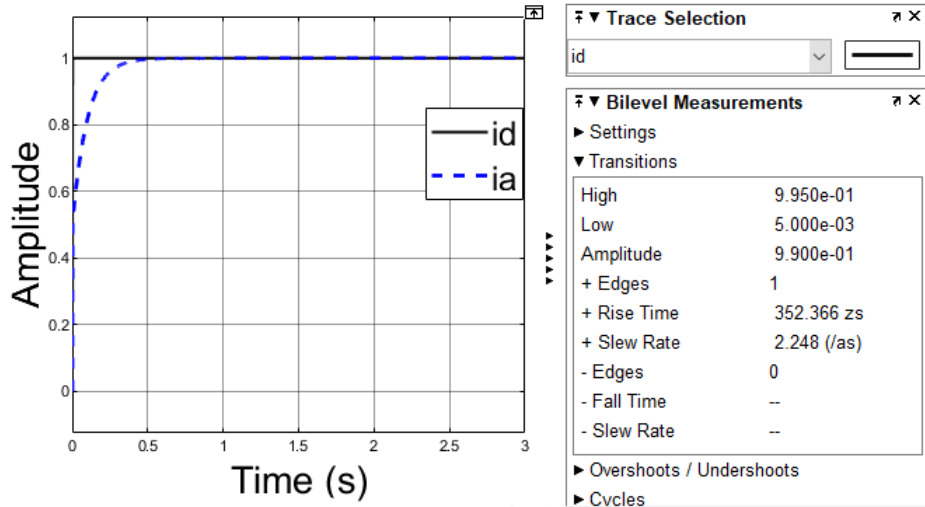


Figure 5.7: Step response of DC motor PID current controller

5.3.2 MATLAB Simulation for The Trajectory Tracker: Cascaded PD Control and Motor Current Control

Proposed cascaded control in 4.1 is simulated using MATLAB Simulink. Fig 5.11 shows the MATLAB Simulink design of the outer loop in the Cascaded control design. This PD controller outputs the control command, which is the desired force for each motor f_1 and f_2 . The desired motor force will convert to the desired current for the motor, which will be the input to the motor current PID control. The output of DC motor current controller will convert to force and feedback to the model of the LED plan and observer $M_1 f_1$ and $M_2 f_2$. Simulation has been ran for a translational motion from 0.5 meter to 0.8 meter

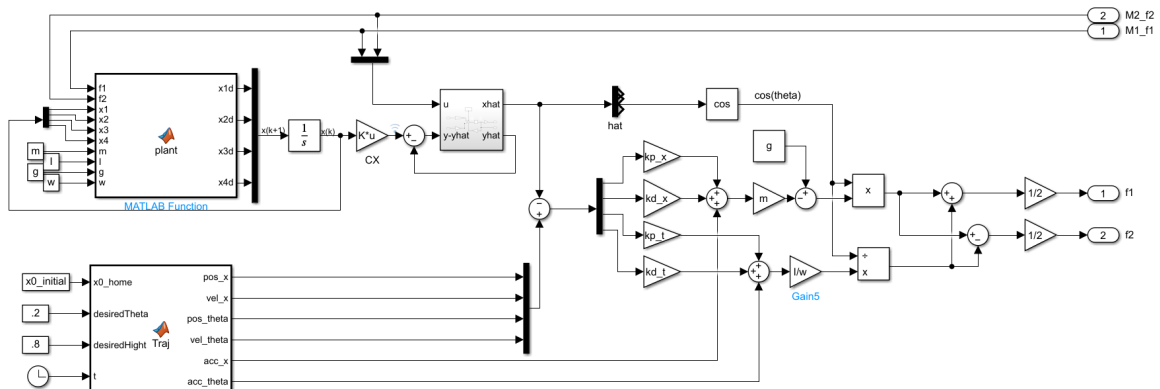


Figure 5.8: Cascaded height and angle control: PD as the outer controller

and rotational motion from .1 Radian to .15 Radian during 20 second. The result of the simulation of the system is shown in Fig 5.9 and 5.10 for translational and rotational motion respectively.

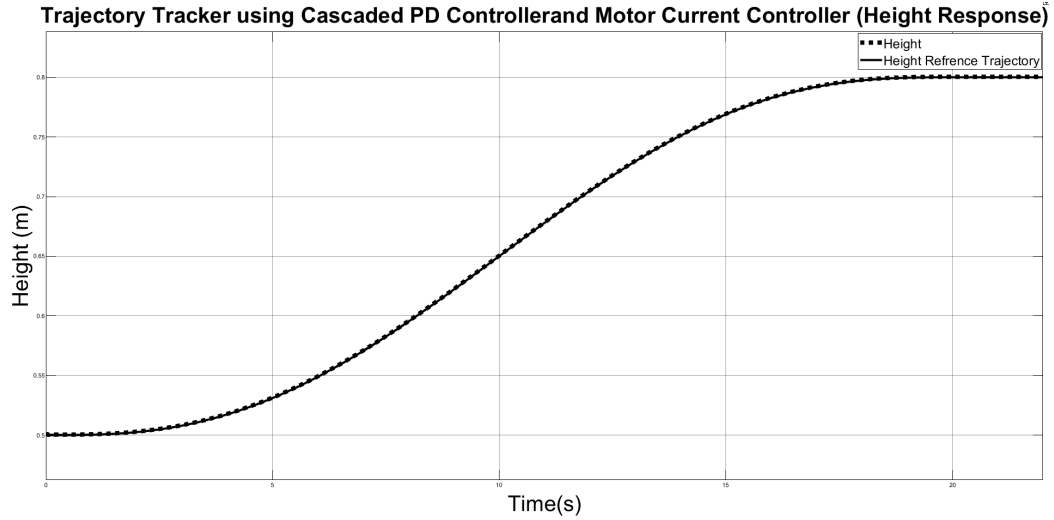


Figure 5.9: Trajectory tracker using cascaded PD controller and motor current controller (Height Response)

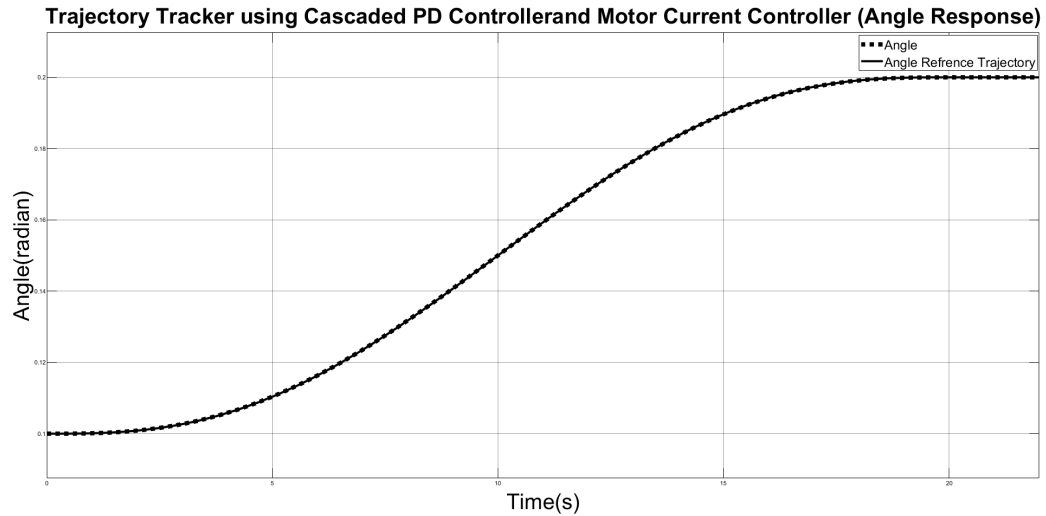


Figure 5.10: Trajectory tracker using cascaded PD controller and motor current controller (Angle Response)

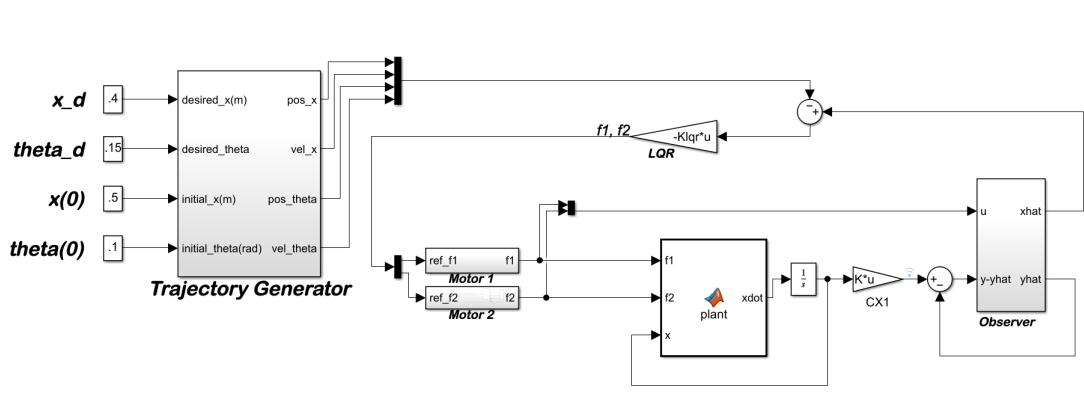


Figure 5.11: Cascaded height and angle control: LQR as the outer controller

5.3.3 MATLAB Simulation for The Trajectory Tracker: Cascaded State-Space Control and Motor Current Control

In this subsection proposed cascaded control in first section of Chapter Four is simulated using MATLAB Simulink. Fig 5.11 shows the MATLAB Simulink design of the outer loop in the Cascaded control design. This PD controller outputs the control command, which is the desired force for each motor.

The main idea in LQR control design is to minimize the quadratic cost function of $\int (x^T Q x + u^T R u) dt$. It turns out that regardless of the values of Q and R, the cost function has a unique minimum that can be obtained by solving the Algebraic Riccati Equation. The parameters Q and R can be used as design parameters to penalize the state variables and the control signals. The larger these values are, the more you penalize these signals. Basically, choosing a large value for R means you try to stabilize the system with less (weighted) energy. This is usually called expensive control strategy. On the other hand, choosing a small value for R means you don't want to penalize the control signal (cheap control strategy). Similarly, if you choose a large value for Q means you try to stabilize the system with the least possible changes in the states and large Q implies less concern about the changes in the states.

Since there is a trade-off between the two, we keep R as I (identity matrix) and only alter Q. Klqr is calculated using a function in MATLAB (lqr(A,B,Q,R)) and the result is as follows:

$$Klqr = 1.0e + 04 * \begin{bmatrix} -5.0000 & -0.0290 & 0.0007 & 0.0007 \\ -5.0000 & -0.0290 & -0.0007 & -0.0007 \end{bmatrix} \quad (5.5)$$

Simulation has been ran for a translational motion from 0.5 meter to 0.8 meter and rotational motion from .1 Radian to .3 Radian during 20 second. The result of the simu-

lation of the system is shown in Fig 5.15 and 5.16 for translational and rotational motion respectively.

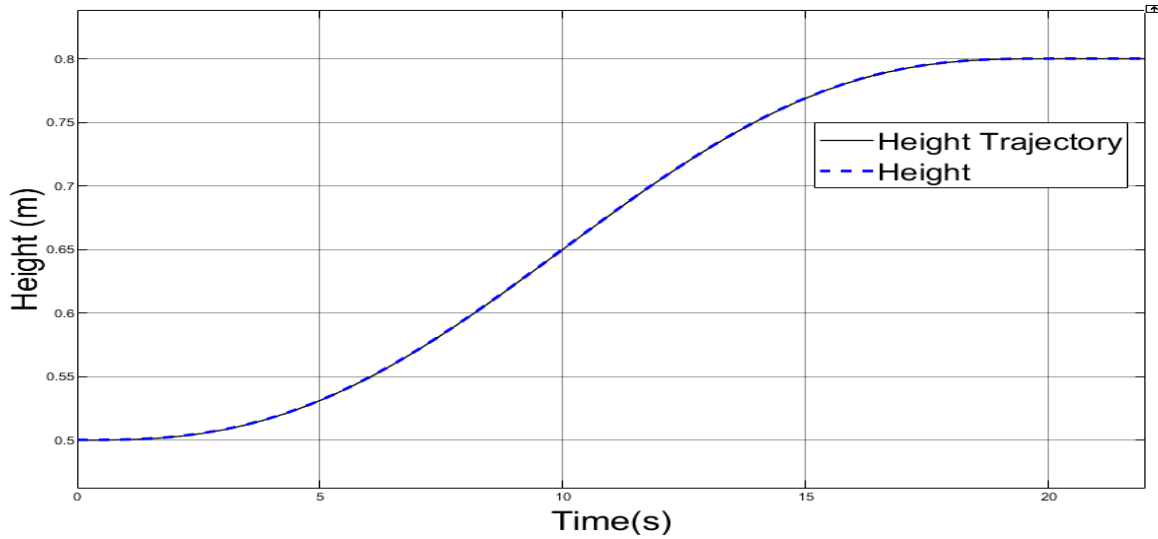


Figure 5.12: Trajectory tracker using cascaded LQR controller and motor current controller (Height Response)

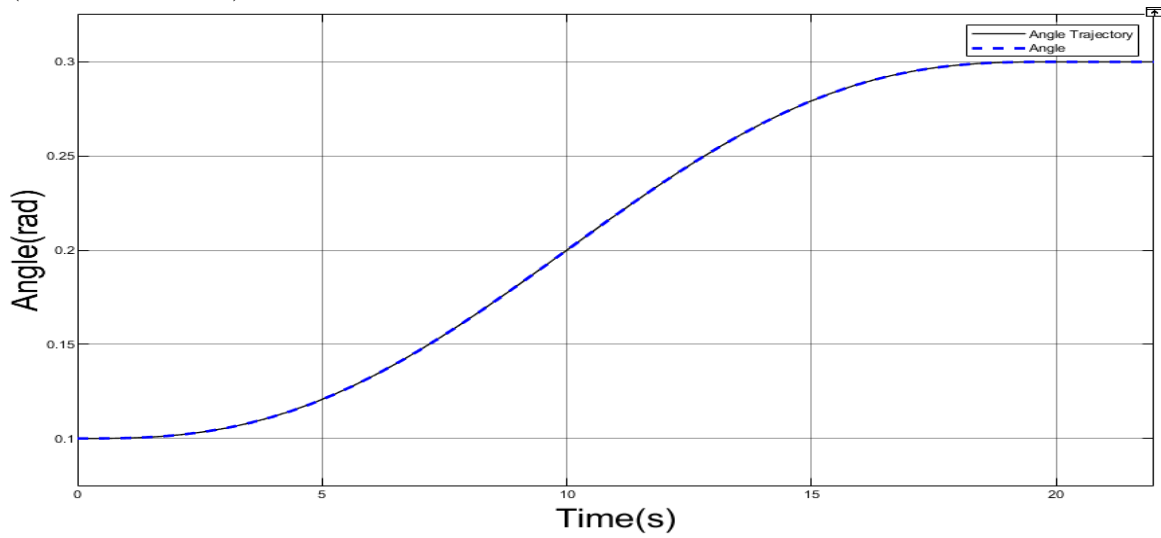


Figure 5.13: Trajectory tracker using cascaded LQR controller and motor current controller (Angle Response)

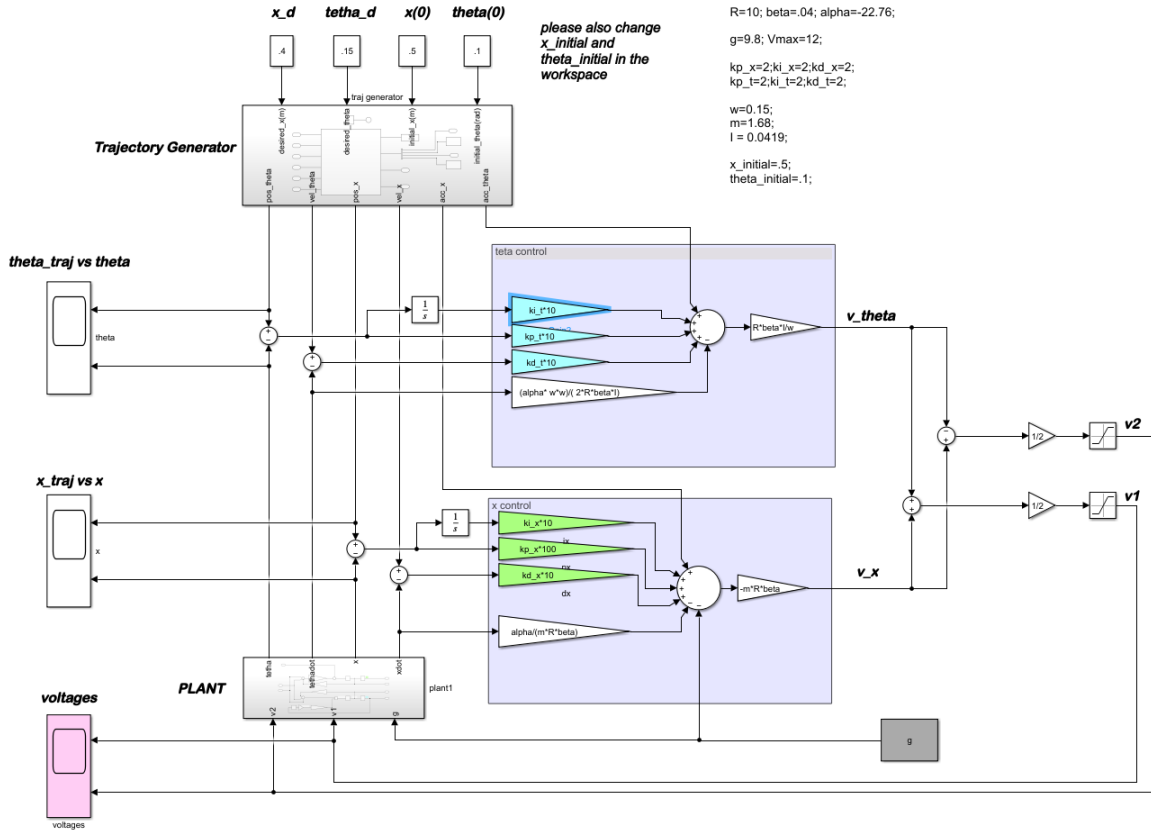


Figure 5.14: MATLAB Simulink for trajectory planner and trajectory tracker: DC motor voltage control

5.4 MATLAB Simulations for The Trajectory Tracker: DC Motor Voltage Control

Voltage control strategy is proposed to lower the prototype cost by eliminating current sensors and their amplifiers. Moreover, we can cut out four tuning parameters of gain and reference voltage for each low current sensors. Proposed DC motor voltage control in the Chapter 4.2 is simulated using MATLAB Simulink. Fig 5.14 shows the MATLAB Simulink design of the voltage control. This controller outputs the control command, which is the desired voltage for each translational and rotational motion v_{θ} and v_x .

Simulation has been ran for a translational motion from 0.5 meter to 0.4 meter and rotational motion from 0.1 Radian to 0.15 Radian during 20 second. The result of the simulation of the system is shown in Fig ?? and ?? for translational and rotational motion respectively.

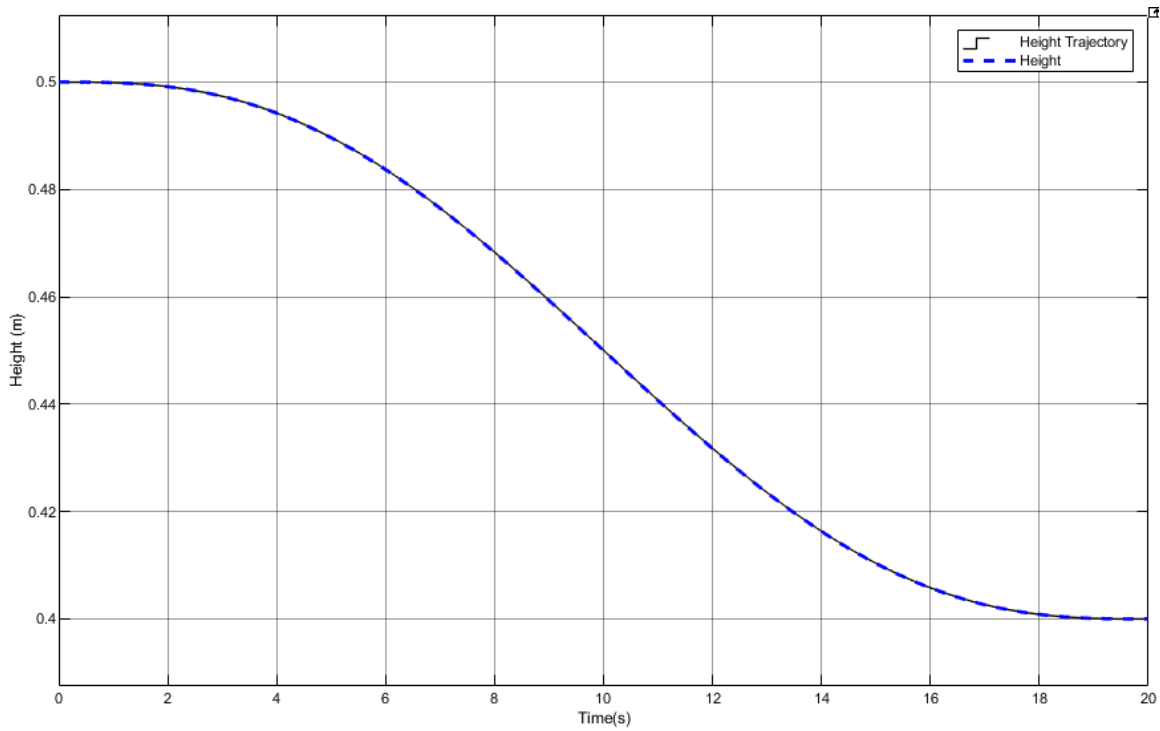


Figure 5.15: Trajectory tracker using cascaded LQR controller and motor current controller (Height Response)

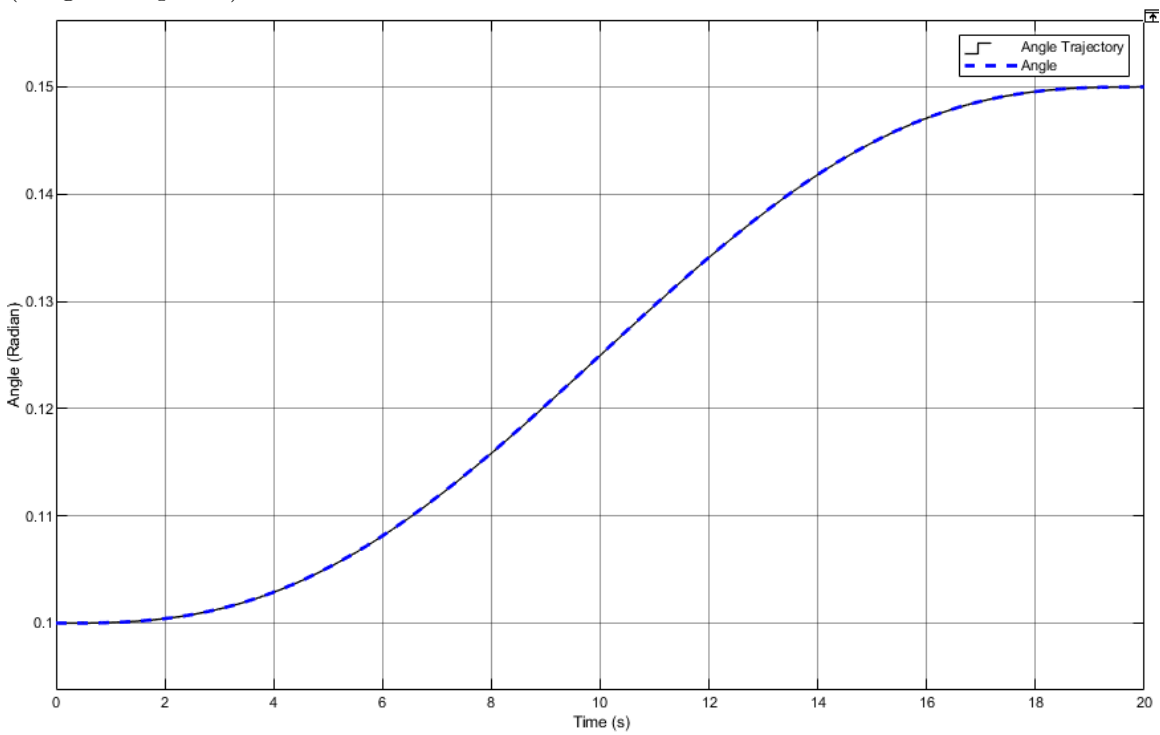


Figure 5.16: Trajectory tracker using cascaded LQR controller and motor current controller (Angle Response)

5.5 Conclusion

In this section, simulation for generating a trajectory and tracking of the trajectory in a 2-DOF robotic manipulator is presented. Three control methods being: (i) cascaded PD control and motor current control, (ii) cascaded state-space control and motor current Control and (iii) DC motor voltage control are simulated. Tracking time, initial and final positions for each scenarios as well as controller parameters and their values are specified. Simulations results show that controller control the end-effector effectively, and end-effector reaches from initial to final coordinates effectively.

Chapter 6

Experimental Results

In this chapter, the purposed control system in Chapter Two and Chapter Three of this thesis are experimentally evaluated and the results are presented. A flexible robotic tracking control test bed has been developed, which allows the implementation of various schemes.

Mobile robots have two mechanical subsystems: (i)actuators and sensors, and (ii) mechanical design. The control of these subsystems requires two electronic power stage as well as acquisition and control stage. The interaction between these stages is the controlling factor of a mobile robot [40].

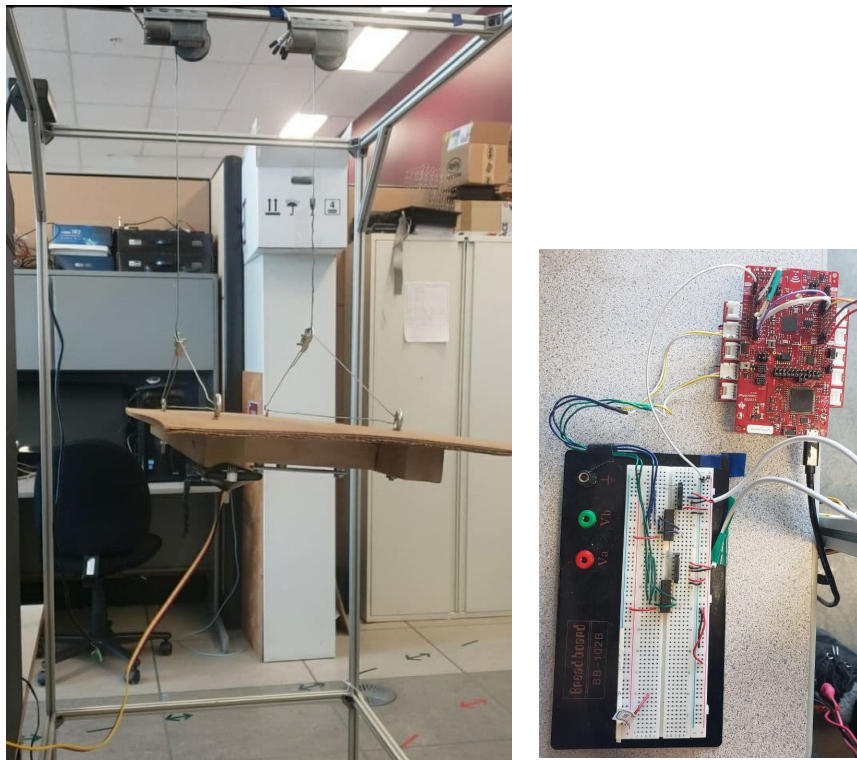


Figure 6.1: The proposed experimental setup for motion control unit

6.1 Experimental Setup

The purposed experimental setup, which is shown in fig 6.1 is designed to examine the proposed trajectory planer and trajectory tracker. This setup supports the LED panel, actuators and sensory devices. A set of two DC motors are located on the bar at the middle of the fixture's top. The LED panel is connected to these motors via two cable, The TOF sensor was mounted on another metal bar parallel to the motor bars to allow a distance between motros and sensor, which will lessen the noise coming from motors and effecting I2c Bus. The TOF sensor should be facing the LED panel. The MPU6050 was placed on the LED panel the LED panel wad connected to the CC3220S-LAUNCHXL's I2C bus with wire. The Texas Instrument CC3220S-LAUNCHXL was used as controller, which have a PWM frequency of 100k. The block diagram in fig 6.2 shows the experimental setup and the parts used as actuators and sensors.

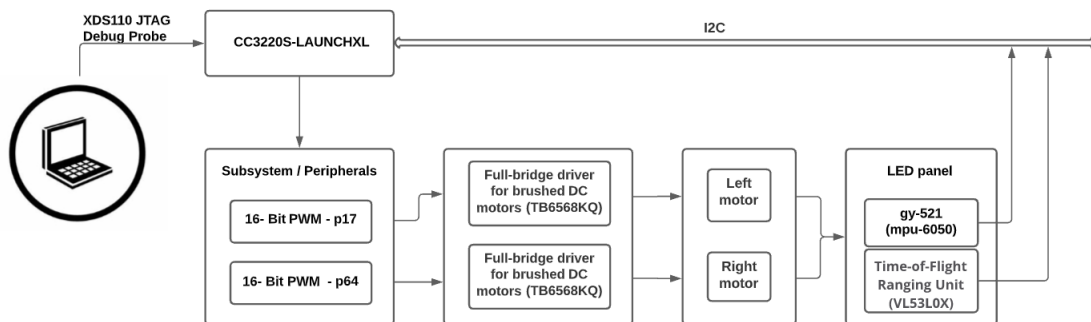


Figure 6.2: The purposed experimental setup block diagram

The dimension of the fixture is 200 cm, 200 cm by 200 cm for height, width and length respectively. The inner radius of the rim is 1 cm.

6.1.1 Sensor Nodes

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial communication bus invented in 1982 by Philips Semiconductors. It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication [32]. this communication interface is used for all the following sensors.

1. **Height Measurement:** To measure the distance between the LED panel and rims a M5Stack Time-of-Flight Ranging Unit (VL53L0X) is used, which is capable of Measuring absolute distances up to 2m with High precision. ToF sensors use a tiny laser to fire out infrared light where the light produced out will bounce off any object and return to the sensor. Based on the time difference between the emission of the light

Parameters	Quantity	Simulation Value
R (<i>ohm</i>)	Armature resistance	4.33
L (<i>H</i>)	Armature inductance	2.34e-3
i (<i>A</i>)	Armature current	0
e (<i>v</i>)	Back electromotive force	0
v (<i>v</i>)	Input voltage	0
J (<i>Kg.m²</i>)	Inertia torque of motor	0.08
B (<i>N.m.s/rad</i>)	Motor friction constant	0.001
K _m (<i>N.m.s/rad</i>)	Motor torque constant	2.18e-2
K _b (<i>V.s/rad</i>)	electromotive force constant	0
η (<i>Nm</i>)	Motor torque	0.9
n	Motor friction constant	6.3
r (<i>m</i>)	Radius of motor shaft	0.025

Table 6.1: Parameters of the DC motor

and its return to the sensor after being reflected by an object, the sensor is able to measure the distance between the object and the sensor. The VL53L0X integrates a leading-edge SPAD array (Single Photon Avalanche Diodes) and embeds ST’s second generation Flight Sense™ patented technology. By integrating a 940 nm VCSEL emitter (Vertical Cavity Surface-Emitting Laser) and infrared filters, the VL53L0X will achieve longer ranges, better immunity to ambient light, and robustness toward glass optical crosstalk.

2. **Angle Measurement:** Gyro + accelerometer gy-521 (mpu-6050)-built on the basis of the chip mpu6050. The module board also contains the necessary mpu6050 strapping, including the I2C interface pull-up resistors. The gyroscope is used to measure linear accelerometer speeds, and accelerometer-angular speeds. Sharing accelerometer and gyro allows you to determine the movement of the body in three-dimensional space.
3. **Light Measurement:** TCS34725 series Light, Color Sensor Sensor Evaluation Board, which has RGB and Clear light sensing elements. An IR blocking filter, integrated on-chip and localized to the color sensing photodiodes, minimizes the IR spectral component of the incoming light and allows color measurements to be made accurately.

6.1.2 Actuators

Two identical high torque turbo geared DC 12V Motor with 40 RPM with characteristic shown in table 6.1 are used in this prototype.

6.2 The Internet of Things Reference Model for Motion Control

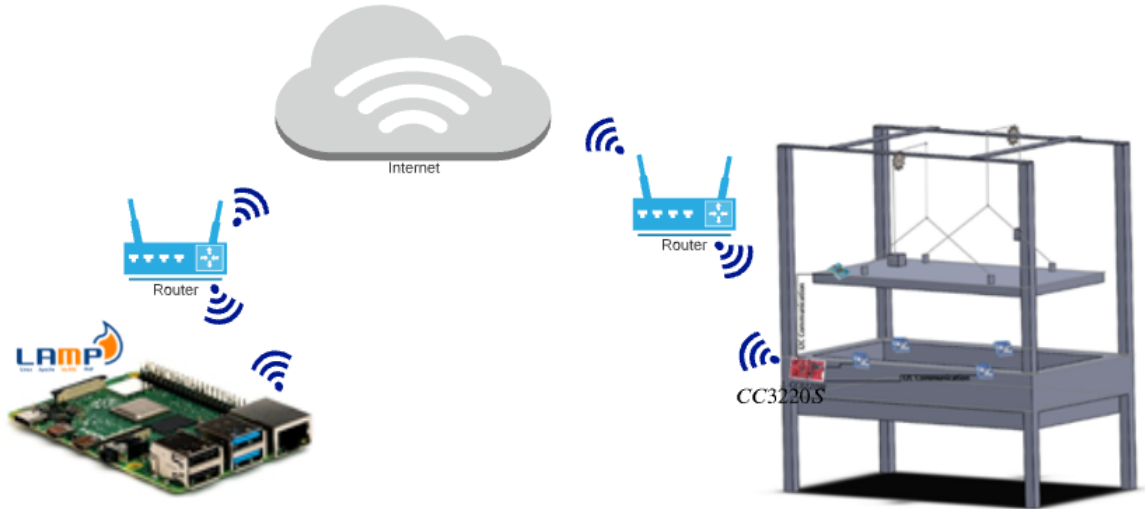


Figure 6.3: IoT platform used in the proposed setup

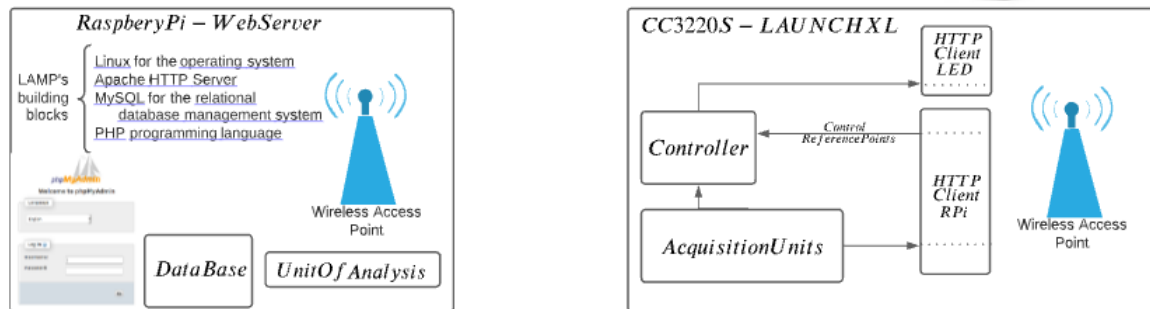


Figure 6.4: IoT system block diagram used in the proposed setup

Proposed IoT platform's performance is evaluated in this section. Fig 6.3 and 6.4 illustrate the IoT platform and IoT system block diagram.

6.2.1 Python script: Fetch data from *sensorReadings* table

Python script can request the data from sensor readings table where all the observations are stored. First, a connection is established between Python script and database using admin username and password. The host will be localhost by default as we are running the database on the same system. Second, using SQL SELECT command we can retrieve data. As an example the SQL query "SELECT R1 FROM sensorReadings ORDER BY Time DESC LIMIT 1" will bring back the latest record for the variable R1, which is the light intensity red captured with first RGB sensor. Fig 6.5 shows the SQL queries, database connection, cursor connection and cursor execute in this process.

```
22/12/2021                                sqlfetch.py *
1      import os
2      import mysql.connector as database
3      import time
4
5
6      #Define Sql Queries to Read the last Value in the Database
7      sqlAngle="SELECT Angle FROM sensorReadings ORDER BY Time DESC LIMIT 1"
8      sqlAngle_d="SELECT desiredAngle FROM desiredReferences_motionControl ORDER BY Time DESC
LIMIT 1"
9      sqlR1="SELECT R1 FROM sensorReadings ORDER BY Time DESC LIMIT 1"
10     sqlR2="SELECT R2 FROM sensorReadings ORDER BY Time DESC LIMIT 1"
11     sqlB1="SELECT B1 FROM sensorReadings ORDER BY Time DESC LIMIT 1"
12     sqlB2="SELECT B2 FROM sensorReadings ORDER BY Time DESC LIMIT 1"
13     #Begin Script Infinitely
14     while True:
15         connection = database.connect(
16             user="admin",
17             password="RamzeboorAdmin",
18             host="localhost",
19             database="phpmyadmin")
20     #The host will be localhost by default if you are running the database on the same
system.
21     cursor = connection.cursor()
22     # A cursor is a database object that retrieves and also updates data, one row at a
time, from a set of data
23
24     #Gather Values from Database
25     cursor.execute(sqlAngle)
26     last = cursor.fetchone()
27     for Angle in last:
28         angle = Angle
29
30     cursor.execute(sqlAngle_d)
31     last = cursor.fetchone()
32     for desiredAngle in last:
33         angle_d = desiredAngle
34
35     cursor.execute(sqlR1)
36     last = cursor.fetchone()
37     for R1 in last:
38         r1 = R1
39
40     cursor.execute(sqlR2)
41     last = cursor.fetchone()
42     for R2 in last:
43         r2 = R2
44
45
46     #Close Cursor
47     cursor.close()
48
49     #Disconnect from Database
50     connection.close()
~
```

Figure 6.5: Python function to fetch data from *sensorReadings* database table

6.2.2 Python script: Add data to *desiredReferencesMotionControl* database table

Python script can add data to *desiredReferencesMotionControl* table table where all the desired values for the motion control unit is being stored. First, a connection is established between Python script and database using admin username and password. The host will

be localhost by default as we are running the database on the same system. Second, using SQL INSERT command we can add data. As an example the SQL query "SELECT R1 FROM sensorReadings ORDER BY Time DESC LIMIT 1" will add desired Height. Fig 6.5 shows the function "add-data", which takes two inputs desired height and desired angle and add them to *desiredReferencesMotionControl* table.

```

1 import os
2 import mysql.connector as database
3 import time
4 connection = database.connect(
5     user="admin",
6     password="RamzeboorAdmin",
7     host="localhost",
8     database="phpmyadmin")
9 #The host will be localhost by default if you are running the database on the same system.
10 cursor = connection.cursor()
11 # A cursor is a database object that retrieves and also updates data, one row at a time, from a set of data
12
13 def add_data(desiredHeight, desiredAngle):
14     try:
15         statement = "INSERT INTO desiredReferences_motionControl (desiredHeight,desiredAngle) VALUES (%s, %s)"
16         data = (desiredHeight, desiredAngle)
17         cursor.execute(statement, data)
18         connection.commit()
19         print("Successfully added entry to database")
20     except database.Error as e:
21         print(f"Error adding entry to database: {e}")

```

```

hell
python 3.7.3 (/usr/bin/python3)
>> %Run addDataToDB.py
>> add_data(60,15)

```

Figure 6.6: Python function to add data to *desiredReferencesMotionControl* database table

Fig 6.7 shows that after calling *add – data(60,15)*, value 60 for height and 15 for angle are added to the *desiredReferencesMotionControl* table.




























	Id	desiredHeight	desiredAngle	Time
  	18	70	0	2021-11-18 12:43:19
  	19	60	15	2021-11-18 12:45:23
  	20	50	0	2021-11-18 12:49:01
  	21	70	10	2021-11-18 15:59:49
  	23	60	15	2021-11-18 16:19:45
  	24	60	15	2021-11-21 13:43:23
  	25	60	15	2021-11-21 13:46:49
  	26	70	0	2021-11-21 13:47:25
  	27	60	15	2021-11-21 13:50:24

Figure 6.7: Desired references motion control database table with updated desired values

6.3 Motion Control Results

The digital implementation of the continuous controller used in this prototype was explained in the last chapter. This discrete algorithms are implemented with the Texas Instrument CC3220S-LAUNCHXL Experimenter Kit, which has maximum clock speed of 80MHz. To program the controller in this MCU, Energia software is used.

The estimated parameters by the control loop were read using serial communication (SCI) with the Micro-controller. The data was read in the computer using serial port and sent to web server to store as well.

6.3.1 Current Controller

Fig 6.11 illustrate the circuit diagram of motor current control. One MPU6050 and one Ultrasonic Ranger are connected to I2C bus and digital input to measure LED panel's angle and distance respectively. Toshiba TB6568KQ Full-Bridge DC Motor Driver IC is used because current for motor cannot be supplied to the motors from the microprocessor. Microprocessors operate at low voltages and require a small amount of current to operate while the motors require a relatively higher voltages and current. Motor Driver IC PWM control enables driving DC motors with high thermal efficiency. Four operating modes are selectable via IN1 and IN2: clockwise (CW), counterclockwise (CCW), Short Brake and Stop. To be able to use only one PWM port of the microprocessor for each motor, we can use the bellow design:

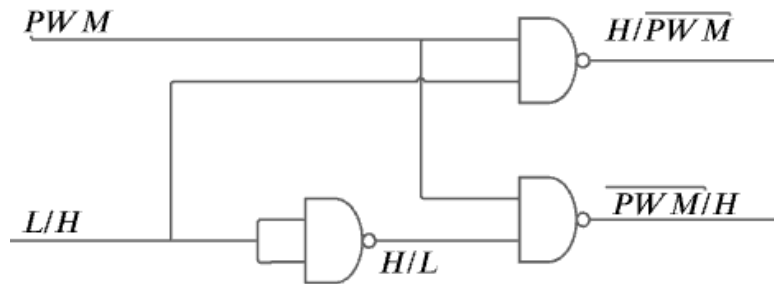


Figure 6.8: Circuit diagram of nand gates used to reduce number of PWM pins

TOSHIBA TC74HC00A digital integrated circuit with quad high speed 2-input NAND gate The TC74HC00A is used to implement this design shown in fig 6.9 and its truth table is shown in fig 6.10.

Proposed design makes it possible to configure one PWM pin for each motor (p17 for motor 1 and p64 for motor2). Also p16 and p53 are configured as output pins for controlling the direction of motor 1 and 2 respectively where HIGH is clockwise direction and LOW is counter clockwise direction. Analog input pins are used for current sensors. p59 for motor 1 and p58 for motor2 are configured as 10-bit ADC for analog inputs.

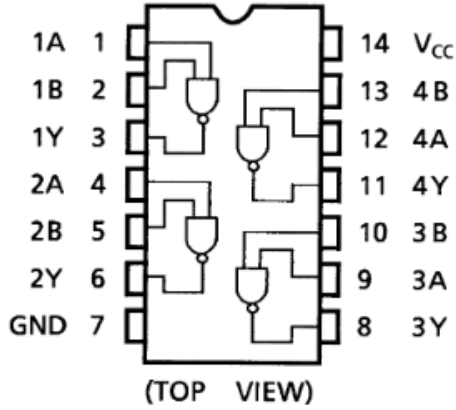


Figure 6.9: Toshiba Quad 2 input NAND gates(74HC00AP)

A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

Figure 6.10: Truth table of NAND gate

For each motor three pins for PWM, direction and current sensor are assigned at initialization (*initMotor(structmotor * motor – t, intpwm, intdir, intcurrentSensorPin)*). Fig 6.14 illustrates the circuit diagram of Proposed current control.

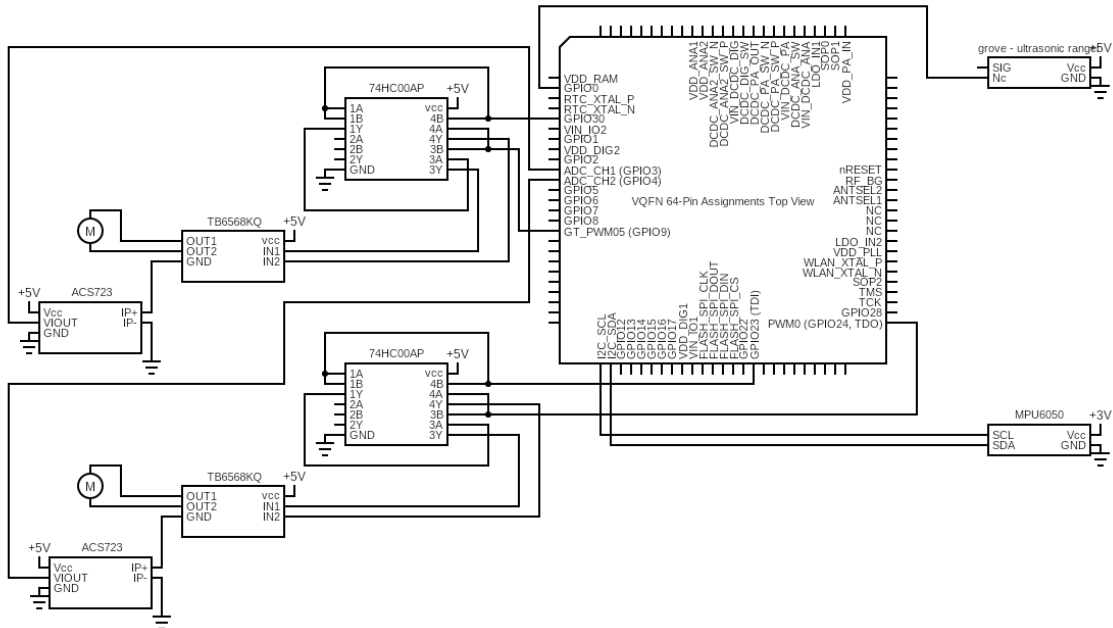


Figure 6.11: Circuit diagram of current control: Terminal Configuration and Functions

Applying the trajectory generator and running the proposed Cascaded PD Control and Motor Current Control for linear path from 0.4 meter to 0.6 meter and rotational path from 0 degree to -10 degree yielded the results shown in Fig 6.12 and Fig 6.13 respectively. Fig fig: shows the Circuit for the current control of DC motor.



Figure 6.12: Trajectory tracker using cascaded PD controller and motor current controller (Height Response)

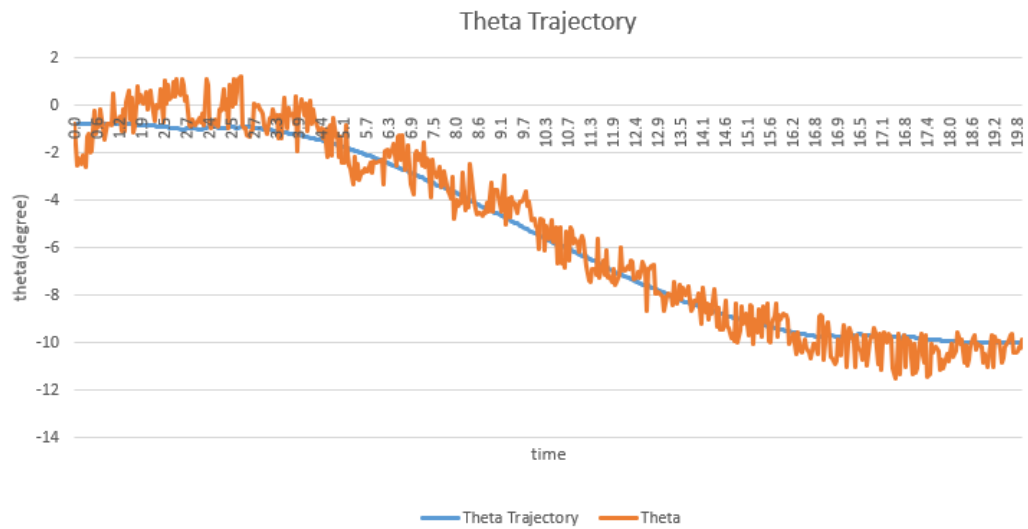


Figure 6.13: Trajectory tracker using cascaded PD controller and motor current controller (Angle Response)

6.3.2 Voltage Controller

Fig 6.14 illustrate the circuit diagram of motor voltage control. One MPU6050 and one VL53L0X are connected to I2C-SCL and I2C-SDA to measure Led panel angle and distance respectively. For each motor there is are assigned Pins for PWM and direction. Toshiba TB6568KQ Full-Bridge DC Motor Driver IC is used and TOSHIBA TC74HC00A Digital Integrated Circuit with Quad high speed 2-Input NAND Gate The TC74HC00A is used to implement this design illustrated in fig 6.9. One PWM pin for each motor (p17 for motor 1 and p64 for motor2) is configured. For each motor, one GPIO is configured as output pin for controlling the direction (p16 for motor 1 and p53 for motor2) where HIGH is clockwise direction and LOW is counter clockwise direction. For each motor two pins for PWM and direction are assigned at initialization (*initMotor(structmotor*motor-t, intpwm, intdir)*). Fig 6.14 illustrates the circuit diagram of Proposed voltage control.

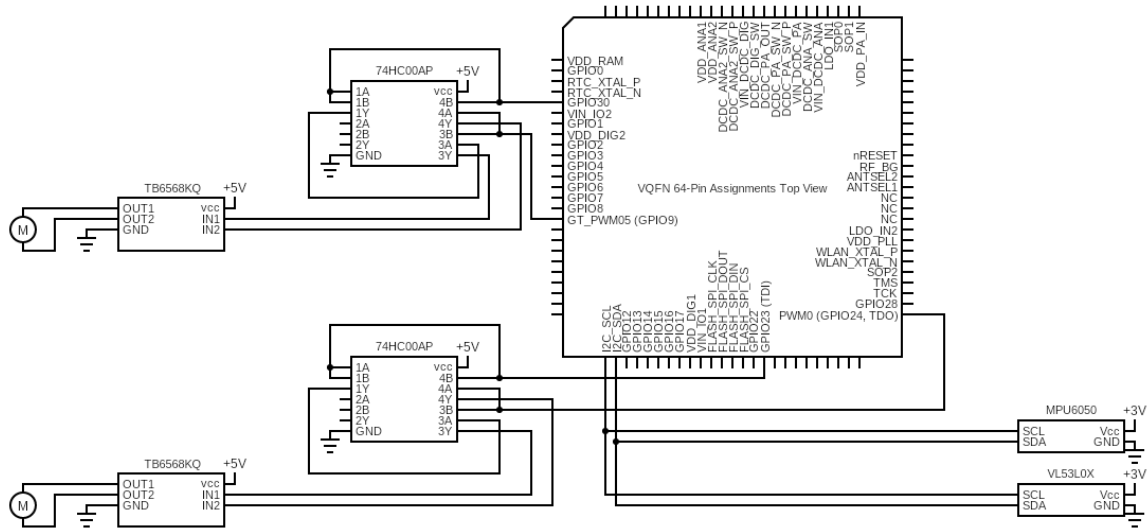


Figure 6.14: Circuit diagram of Voltage ctrl: terminal configuration and functions

Applying the trajectory generator and running the proposed DC Motor voltage Control for linear path starting from 0.6 meter to 0.5 meter and rotational path from 15 degree to 0 degree yielded the results shown in Fig 6.15 and Fig 6.16 respectively. Fig fig:circuitVoltage shows the Circuit diagram for the voltage control of DC motor.

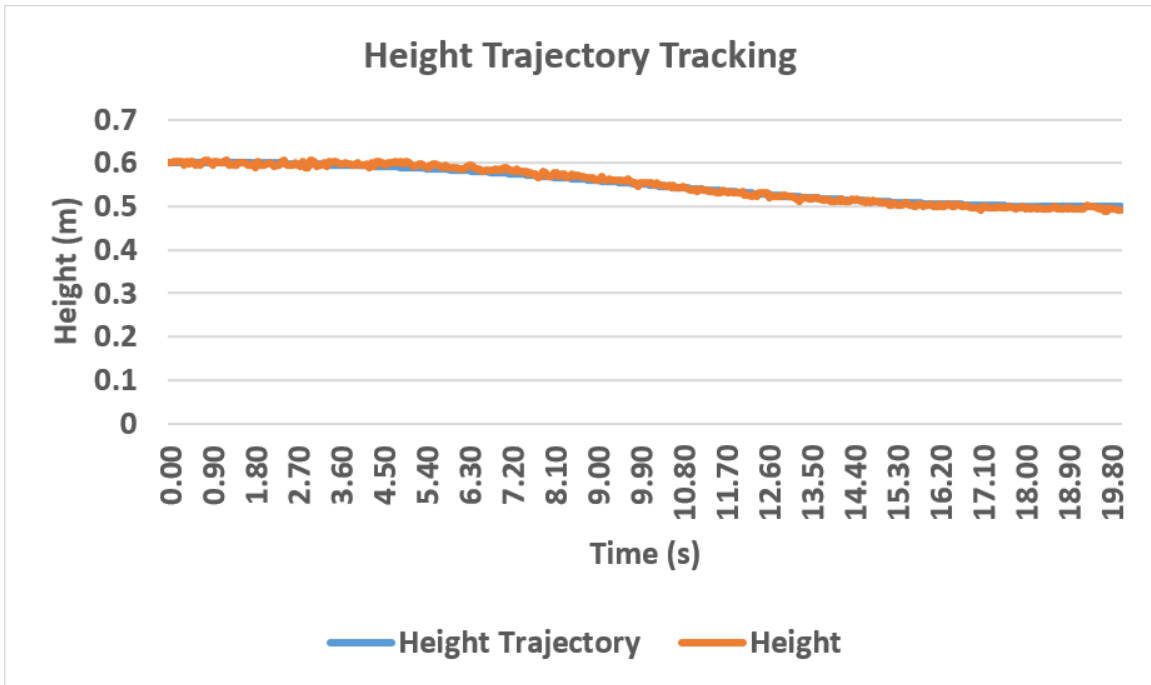


Figure 6.15: Trajectory tracker using DC motor Voltage Control (Height Response)

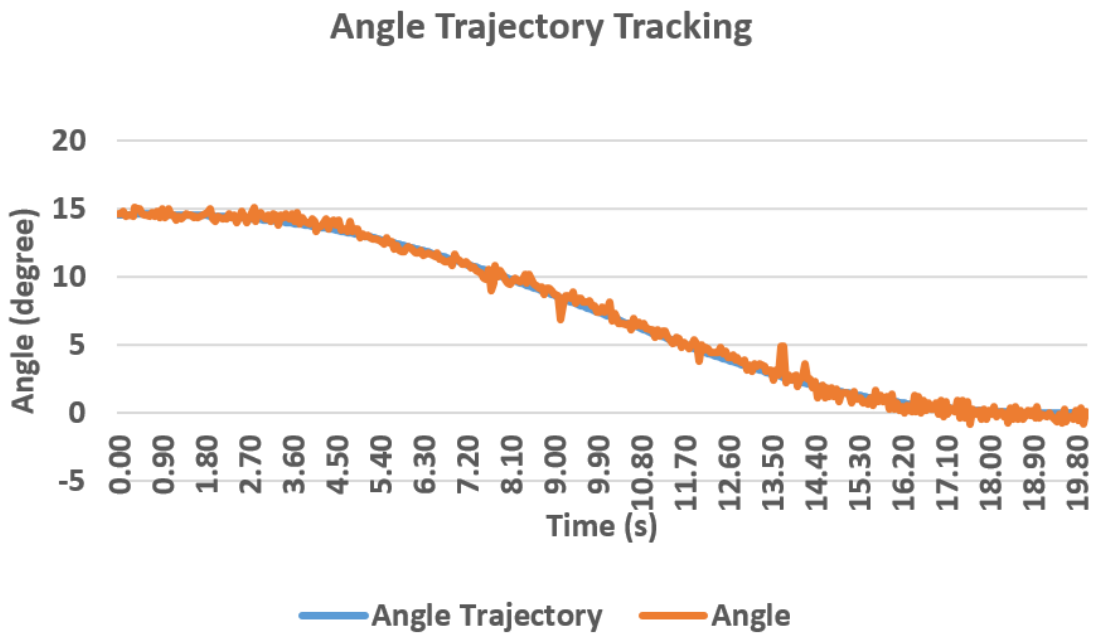
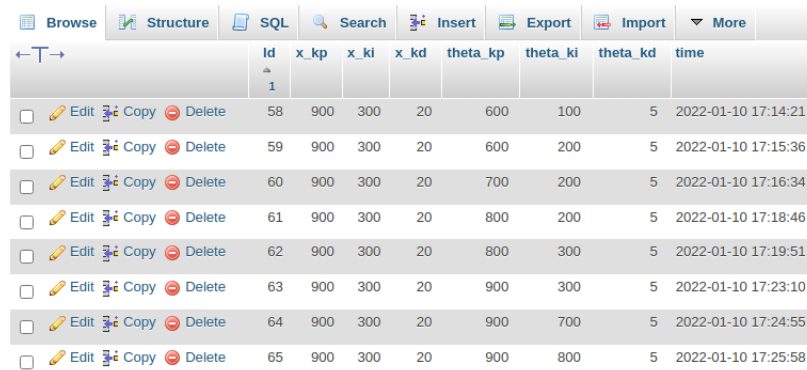


Figure 6.16: Trajectory tracker using DC motor Voltage Control (Angle Response)

6.4 Online PID Tuning

A modification is added to the system to enable online system tuning. In the proposed voltage control we have to tune 6 PID parameters. As can be seen in 4.2, calculating v_x needs 3 PID parameters 4.53 and calculating v_θ needs another 3 PID parameters 4.57. These values are provided to motion control unit upon request from server. In the database another table is configured to store the PID parameters. A Python function is responsible to add the most updated values to this table. Each time the motion control unit is triggered, It will request the latest PID values and updates the values in its PID structures. Fig 6.17 shows the SQL table containing PID parameters.



	Id	x_kp	x_ki	x_kd	theta_kp	theta_ki	theta_kd	time
	1							
<input type="checkbox"/>	58	900	300	20	600	100	5	2022-01-10 17:14:21
<input type="checkbox"/>	59	900	300	20	600	200	5	2022-01-10 17:15:36
<input type="checkbox"/>	60	900	300	20	700	200	5	2022-01-10 17:16:34
<input type="checkbox"/>	61	900	300	20	800	200	5	2022-01-10 17:18:46
<input type="checkbox"/>	62	900	300	20	800	300	5	2022-01-10 17:19:51
<input type="checkbox"/>	63	900	300	20	900	300	5	2022-01-10 17:23:10
<input type="checkbox"/>	64	900	300	20	900	700	5	2022-01-10 17:24:55
<input type="checkbox"/>	65	900	300	20	900	800	5	2022-01-10 17:25:58

Figure 6.17: Database table containing updated desired PID values

6.5 Conclusion

In this section, experimental result for generating a trajectory and tracking of the trajectory in a 2-DOF robotic manipulator is presented. Two control methods being: (i) cascaded PD control and motor current control and (ii) DC motor voltage control are implemented. Circuit diagrams of both control methods are provided in which the connection of all sensory units and actuators to the processing unit and the wiring diagram is presented.

Tracking time, initial and final positions for each motion scenario are specified. Experimental results show that although both controller control the end-effector so that it can reaches from initial to final coordinates effectively, DC motor voltage control performs smoother. It can be as a result of having less parameters to tune when it come to DC motor voltage control. Also it can be the result of using the low current version of the SparkFun current sensor breakout and the Texas Instruments CC3220S-LAUNCHXL. The current sensor board provides higher sensitivity to current by using an op-amp providing a voltage range of 0 to 5 volt as its output. But, the CC3220S-LAUNCHXL's analog to digital pins can not tolerate a voltage more than 1.8v and will cap the signal at 1.4v making it impossible to use the full potential of the low current sensor.

Chapter 7

Conclusion

It was shown that the interest in greenhouse energy management has increased recently by improvements in crop growth monitoring and light intensity control. The main integration challenge of these light intensity control is maintaining the optimized light intensity for each plant growth stage by performing advanced monitoring and control strategies. The control algorithm should react according to the environmental features to ensure the reliable and optimized control.

In this thesis, the performance of the IoT based two Degree of freedom robotic LED light fixture is analyzed. Using a mathematical model, the state-state presentation of a cable robotic LED light fixture is proposed. Also, feasibility of using IoT connectivity to enhance the data logging process as well as using cloud computing for advanced decision making for motion control reference values is discussed.

First, IoT framework is proposed and configured. The main purpose of this subsection is to provide the desired values for the motion control unit. The motion control unit have a trajectory planner, which accurately plans the linear and rotational movement for the LED light panel. Propose IoT platform is based on a web server which is implemented on a Raspberry Pi board. Web server uses Apache2 to handle HTTP requests coming from motion control unit which is implemented on a Texas Instruments CC3220S-LAUNCHXL board. All the desired values for motion control get stored in a database table. The database and its tables are managed phpMyAdmin and are password protected. The IoT framework can be used to store records and environmental features provided using sensors. In this prototype the CC3220S-LAUNCHXL is also in charge of data logging and data transfer to database.

Second, simulation for generating a trajectory and tracking of the trajectory is presented. Three control methods being: (i) cascaded PD control and motor current control, (ii) cascaded state-space control and motor current Control and (iii) DC motor voltage control are simulated. Tracking time, initial and final positions for each scenarios as well as controller parameters and their values are specified. Simulations results show that controller

control the end-effector effectively, and end-effector reaches from initial to final coordinates effectively.

And finally, the experimental result for fetching data from the webserver is provided. Having an updated value for either desired height or desired angle will trigger the motion control unit. Motion control unit will first create the coefficient of a fifth order polynomial. It will provide a new reference value for motor control unit at each sample time. Two control methods for DC motor are proposed and implemented on a test bed setup: (i) cascaded PD control and motor current control and (ii) DC motor voltage control. Experimental results show that although both controller control the end-effector so that it can reaches from initial to final coordinates effectively, DC motor voltage control performs smoother.

It should be mentioned that simulations were performed by Matlab/Simulink and experimental setup was developed and build in the SFU's mechatronic systems engineering lab to validate the performance of the purposed IoT-based control system. Circuit diagrams of both control methods are provided using Circuit-Diagram online tool in which the connection of all sensory units and actuators to the processing unit and the wiring diagram is presented.

7.1 Future Work

This work can be further developed in the future as suggestion in the followings

1. **Upgrading the actuator:** Using motor with encoder for accurate position feedback.
2. **Upgrading the sensors:** This work can be further improved using sensors with higher refresh rate.
3. **light prediction:** This work can be further improved to be able to utilize light sensors and weather forecast for determination of desired angle for light panel.
4. **Image processing:** This work can be further improved to be able to utilize image processing in tracking the plant growth and calculating the desired height for light panel.

Bibliography

- [1] 11 core guidelines to know before buying grow lights. <https://www.urbanvine.co>.
- [2] Dimlux vs luxx: Which is the best 1000 watt grow light. <https://www.dimluxlighting.com/knowledge>.
- [3] Mpu-6000 and mpu-6050 product specification ps-mpu-6000a-00. (PS-MPU-6000A-00), 8 2013. Rev. 3.4.
- [4] Ieee standard for an architectural framework for the internet of things (iot). *IEEE Std 2413-2019*, pages 1–269, 2020.
- [5] Dafni Despoina Avgoustaki and George Xydis. Energy cost reduction by shifting electricity demand in indoor vertical farms with artificial lighting. *Biosystems Engineering*, 211:219–229, 2021.
- [6] Jeremy Begey, Loic Cuvillon, Maximilien Lesellier, Marc Gouttefarde, and Jacques Gangloff. Dynamic control of parallel robots driven by flexible cables and actuated by position-controlled winches. *IEEE Transactions on Robotics*, 35(1):286–293, 2018.
- [7] William Bolton. *Instrumentation and control systems*. Newnes, 2021.
- [8] J Boaventura Cunha et al. Greenhouse climate models: An overview. In *Efita 2003 conference*, pages 823–829. Citeseer, 2003.
- [9] Grzegorz Ówikła. Methods of manufacturing data acquisition for production management-a review. In *Advanced Materials Research*, volume 837, pages 618–623. Trans Tech Publ, 2014.
- [10] David L Ehret, Bernard D Hill, Tom Helmer, and Diane R Edwards. Neural network modeling of greenhouse tomato yield, growth and water use from automated crop monitoring data. *Computers and electronics in agriculture*, 79(1):82–89, 2011.
- [11] Nicholas Engler and Moncef Krarti. Review of energy efficiency in controlled environment agriculture. *Renewable and Sustainable Energy Reviews*, 141:110786, 2021.
- [12] Ben Ezair, Tamir Tassa, and Zvi Shiller. Planning high order trajectories with general initial and final conditions and asymmetric bounds. *The International Journal of Robotics Research*, 33(6):898–916, 2014.
- [13] Othmane Friha, Mohamed Amine Ferrag, Lei Shu, Leandros A Maglaras, and Xiaochan Wang. Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies. *IEEE CAA J. Autom. Sinica*, 8(4):718–752, 2021.

- [14] Clement Gosselin, Ping Ren, and Simon Foucault. Dynamic trajectory planning of a two-dof cable-suspended parallel robot. In *2012 IEEE International conference on Robotics and Automation*, pages 1476–1481. IEEE, 2012.
- [15] Ping Guo, Puwadol Oak Dusadeerungsikul, and Shimon Y Nof. Agricultural cyber physical system collaboration for greenhouse stress management. *Computers and electronics in agriculture*, 150:439–454, 2018.
- [16] X. Hao. Latest development in lighting greenhouse vegetables. 2019.
- [17] X Hao, C Little, JM Zheng, and R Cao. Far-red leds improve fruit production in greenhouse tomato grown under high-pressure sodium lighting. In *VIII International Symposium on Light in Horticulture 1134*, pages 95–102, 2016.
- [18] E Iddio, L Wang, Y Thomas, G McMorrow, and A Denzer. Energy efficient operation and modeling for greenhouses: A literature review. *Renewable and Sustainable Energy Reviews*, 117:109480, 2020.
- [19] Birgit Jacob and Hans J Zwart. *Linear port-Hamiltonian systems on infinite-dimensional spaces*, volume 223. Springer Science & Business Media, 2012.
- [20] Faisal Jamil, Muhammad Ibrahim, Israr Ullah, Suyeon Kim, Hyun Kook Kahng, and Do-Hyeun Kim. Optimal smart contract for autonomous greenhouse environment based on iot blockchain network in agriculture. *Computers and Electronics in Agriculture*, 192:106573, 2022.
- [21] Xinglin Ke, Hideo Yoshida, Shoko Hikosaka, and Eiji Goto. Optimization of photosynthetic photon flux density and light quality for increasing radiation-use efficiency in dwarf tomato under led light at the vegetative growth stage. *Plants*, 11(1):121, 2022.
- [22] H Marjolein Kruidhof and Wade H Elmer. Cultural methods for greenhouse pest and disease management. In *Integrated Pest and Disease Management in Greenhouse Crops*, pages 285–330. Springer Dordrecht, The Netherlands, 2020.
- [23] Gabriel LaPlante, Sonja Andrekovic, Robert G Young, Jocelyn M Kelly, Niki Bennett, Elliott J Currie, and Robert H Hanner. Canadian greenhouse operations and their potential to enhance domestic food security. *Agronomy*, 11(6):1229, 2021.
- [24] lightrail3. How does angled indoor plant light change grow results? *URL: <https://www.lightrail3.com/angled-indoor-plant-light-change-grow-results/>*.
- [25] Matheus Cardim Ferreira Lima, Maria Elisa Damascena de Almeida Leandro, Constantino Valero, Luis Carlos Pereira Coronel, and Clara Oliva Gonçalves Bazzo. Automatic detection and monitoring of insect pests—a review. *Agriculture*, 10(5):161, 2020.
- [26] Tan Liu, Qingyun Yuan, and Yonggang Wang. Hierarchical optimization control based on crop growth model for greenhouse light environment. *Computers and Electronics in Agriculture*, 180:105854, 2021.

- [27] Jordan M Longval and Clément Gosselin. Dynamic trajectory planning and geometric analysis of a two-degree-of-freedom translational cable-suspended planar parallel robot using a parallelogram cable loop. *Journal of Mechanisms and Robotics*, 11(2):020903, 2019.
- [28] Fangfang Ma, Lara J Jazmin, Jamey D Young, and Doug K Allen. Isotopically nonstationary ^{13}C flux analysis of changes in arabidopsis thaliana leaf metabolism due to high light acclimation. *Proceedings of the National Academy of Sciences*, 111(47):16967–16972, 2014.
- [29] R Madhumathi, T Arumuganathan, and R Shruthi. Internet of things in precision agriculture: A survey on sensing mechanisms, potential applications, and challenges. In *Intelligent Sustainable Systems*, pages 539–553. Springer, 2022.
- [30] Chrysanthos Maraveas and Thomas Bartzanas. Application of internet of things (iot) for optimized greenhouse environments. *AgriEngineering*, 3(4):954–970, 2021.
- [31] F Merat. Introduction to robotics: Mechanics and control. *IEEE Journal on Robotics and Automation*, 3(2):166–166, 1987.
- [32] Adrianus PMM Moelands and Herman Schutte. Two-wire bus-system comprising a clock wire and a data wire for interconnecting a number of stations, August 25 1987. US Patent 4,689,740.
- [33] Taewon Moon, Joon Woo Lee, and Jung Eek Son. Accurate imputation of greenhouse environment data for data integrity utilizing two-dimensional convolutional neural networks. *Sensors*, 21(6):2187, 2021.
- [34] Redmond R Shamshiri, Fatemeh Kalantari, KC Ting, Kelly R Thorp, Ibrahim A Hameed, Cornelia Weltzien, Desa Ahmad, and Zahra Mojgan Shad. Advances in greenhouse automation and controlled environment agriculture: A transition to plant factories and urban agriculture. 2018.
- [35] Wesley C Randall and Roberto G Lopez. Comparison of bedding plant seedlings grown under sole-source light-emitting diodes (leds) and greenhouse supplemental lighting from leds and high-pressure sodium lamps. *HortScience*, 50(5):705–713, 2015.
- [36] JSFC2KT Instructions revised. Jump start t5 24w 2’ standing lighting system. URL: <https://www.hydrofarm.com/p/jump-start-t5-24w-2-standing-lighting-system/jsfc2kt>, 2016.
- [37] Derek Rowell. State-space representation of lti systems. URL: <http://web.mit.edu/2.14/www/Handouts/StateSpace.pdf>, 2002.
- [38] Gianluca Serale, Luca Gnoli, Emanuele Giraudo, and Enrico Fabrizio. A supervisory control strategy for improving energy efficiency of artificial lighting systems in greenhouses. *Energies*, 14(1):202, 2021.
- [39] Snehal R Shinde, AH Karode, and Dr SR Suralkar. Review on-iot based environment monitoring system. *International Journal of Electronics and Communication Engineering and Technology*, 8(2):103–108, 2017.

- [40] Ramón Silva-Ortigoza, C Márquez-Sánchez, Mariana Marcelino-Aranda, Magdalena Marciano-Melchor, Gilberto Silva-Ortigoza, R Bautista-Quintero, ER Ramos-Silvestre, JC Rivera-Díaz, and D Muñoz-Carrillo. Construction of a wmr for trajectory tracking control: Experimental results. *The Scientific World Journal*, 2013, 2013.
- [41] Lewei Tang, Xiaoqiang Tang, Xiaoling Jiang, and Clement Gosselin. Dynamic trajectory planning study of planar two-dof redundantly actuated cable-suspended parallel robots. *Mechatronics*, 30:187–197, 2015.
- [42] Wen Tao, Liang Zhao, Guangwen Wang, and Ruobing Liang. Review of the internet of things communication technologies in smart agriculture and challenges. *Computers and Electronics in Agriculture*, page 106352, 2021.
- [43] Ahmad Tay, Frédéric Lafont, and Jean-François Balmat. Forecasting pest risk level in roses greenhouse: Adaptive neuro-fuzzy inference system vs artificial neural networks. *Information Processing in Agriculture*, 8(3):386–397, 2021.
- [44] Pradyumna K Tripathy, Ajaya K Tripathy, Aditi Agarwal, and Saraju P Mohanty. My-green: An iot-enabled smart greenhouse for sustainable agriculture. *IEEE Consumer Electronics Magazine*, 2021.
- [45] Tomáš Tureček, Pavel Vařacha, Alžběta Turečková, Václav Psota, Peter Jank, Vít Štěpánek, Adam Viktorin, Roman Šenkeřík, Roman Jašek, Bronislav Chramcov, et al. Scouting of whiteflies in tomato greenhouse environment using deep learning. In *Agriculture Digitalization and Organic Production*, pages 323–335. Springer, 2022.
- [46] Michel J Verheul, Henk FR Maessen, Martina Paponov, Anush Panosyan, Dmitry Kechasov, Muhammad Naseer, and Ivan A Paponov. Artificial top-light is more efficient for tomato production than inter-light. *Scientia Horticulturae*, 291:110537, 2022.
- [47] David S Watson, Mary Ann Piette, Osman Sezgen, and Naoya Motegi. *Machine to machine (M2M) technology in demand responsive commercial buildings*. River Publishers, 2020.
- [48] You Wu, Shicheng Yan, Junliang Fan, Fucang Zhang, Wenju Zhao, Jing Zheng, Jinjin Guo, Youzhen Xiang, and Lifeng Wu. Combined effects of irrigation level and fertilization practice on yield, economic benefit and water-nitrogen use efficiency of drip-irrigated greenhouse tomato. *Agricultural Water Management*, 262:107401, 2022.
- [49] Jihong Zhang, Huyin Li, Maorong Liu, Huan Zhang, Hai Sun, Hongtuo Wang, Lin Miao, Miaomiao Li, Ruihao Shu, and Qilian Qin. A greenhouse test to explore and evaluate light-emitting diode (led) insect traps in the monitoring and control of trialeurodes vaporariorum. *Insects*, 11(2):94, 2020.
- [50] Kai Zhang, Jihua Yu, and Yan Ren. Research on the size optimization of photovoltaic panels and integrated application with chinese solar greenhouses. *Renewable Energy*, 182:536–551, 2022.
- [51] Bin Zi, Baoyan Duan, Jingli Du, and Yuanying Qiu. Trajectory tracking sliding mode control of a cable parallel manipulator based on fuzzy logic. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 9203–9207. IEEE, 2006.

Chapter 8

C Code For The Texas Instruments CC3220S-LAUNCHXL

In this section, the Energia code used to implement the purposed motion control unit is provided. This sample code only shows the sequence and logic used in the microprocessor and the libraries, modules, variables, structures and functions are not included for simplicity. It should be noted that the C is used to implement the controller.

```
1 // Include libraries here. For simplicity, this part is removed.
2 // Put all the variables, structures and IP addresses here. For
  simplicity, this part is removed.
3
4 void setup() {
5     // point to the light panel which we want to control
6     led_t = &led1;
7
8     Wire.begin(); // initiate the Wire library and join the I2C bus
9
10    WiFi.begin(ssid, ssid_password); // attempt to connect to WiFi
11
12    initGyroscope(); // initialize the gyroscope
13
14    // initialize motors and assign pwm and direction pins to them
15    // initMotor(struct motor *motor_t, int pwm, int dir)
16    initMotor(&motor1, 31, 32);
17    initMotor(&motor2, 29, 28);
18
19    httpRequest_delay_ms = millis(); // start timer for HTTP
    request
20    sendingToDB_delay_ms = millis(); // start timer for data
    logging
21 }
22 void loop() {
```

```

23 MPU6050Readings(); // read gyroscope
24 VL53L0XReadings(); // read distance
25
26 // connect to web server
27 if (client_RPi.connect(RpiServer, 80)) {
28     // fetch the desired motion values from Phpmyadmin Database
29     fetchingDesiredMotionValuesFromPhpmyadminDatabase();
30 } else {
31     Serial.println("mysql_connection_failed");
32 }
33
34 // if we have an updated value, run the motion control
    algorithm
35 if(incomingFloatAngle != lastIncomingFloatAngle ||
    incomingFloatHight != lastIncomingFloatHight){
36     lastIncomingFloatAngle = incomingFloatAngle;
37     lastIncomingFloatHight = incomingFloatHight;
38     heightTrajectoryArray.readyForInitialization = true;
39 }
40
41 if (heightTrajectoryArray.readyForInitialization) {
42
43     // Initializing trajectoryArray profile. This function runs
        only once each time a new trajectoryArray is needed.
44     initTraj(&heightTrajectoryArray, incomingFloatHight,
        trajectoryDuration);
45     led_t->x = heightTrajectoryArray.state;
46
47     initTraj(&thetaTrajectoryArray, incomingFloatAngle,
        trajectoryDuration);
48     led_t->theta = thetaTrajectoryArray.state;
49
50     // assign the home values
51     x[0] = led_t->x;
52     xdot[0] = 0;
53     theta[0] = led_t->theta;
54     thetadot[0] = 0;
55     X << x[0], xdot[0], theta[0], thetadot[0];
56 }
57 if (heightTrajectoryArray.initialized) {
58     // reset initialized flags
59     heightTrajectoryArray.readyForInitialization = false;
60     heightTrajectoryArray.initialized = false;
61     thetaTrajectoryArray.initialized = false;
62     t = 0;
63     while (t <= trajectoryDuration) {
64         whileloopStartTime = millis();

```

```

65 trajectoryArrayPlanner(&heightTrajectoryArray);
66 trajectoryArrayPlanner(&thetaTrajectoryArray);
67
68 // update the reference vector (assumed to be of the same
    dimension as the observation y)
69 r << heightTrajectoryArray.position, heightTrajectoryArray.
    velocity, thetaTrajectoryArray.position,
    thetaTrajectoryArray.velocity;
70
71 // call voltage control unit
72 voltageController(&pidX, &pidTheta);
73
74 // update states
75 X << led_t->x, led_t->xdot, led_t->theta, led_t->thetadot;
76
77 stopMotor(&motor1); stopMotor(&motor2);
78 led_t->xKm1 = led_t->x; // save last distance state
79 VL53L0XReadings(); // read distance
80 led_t->thetaKm1 = led_t->theta; // save last theta state
81 MPU6050Readings(); // update theta in radian
82 velocityObserver(); // update xdot and thetadot
83
84 // create pwm signal
85 pwmGenerator(&motor1, vm1);
86 pwmGenerator(&motor2, vm2);
87
88 // next time step
89 t = t + samplingTime;
90
91 // wait for sampling time to reach its value
92 while (millis() - whileloopStartTime < samplingTime) {}
93 }
94
95 // stop the motors
96 stopMotor(&motor1); stopMotor(&motor2);
97
98 // disable the motion tracing unit
99 heightTrajectoryArray.initialized = false;
100 }
101 }

```

Chapter 9

Configuration of LAMP Server on Raspberry Pi

9.1 Install LAMP Server Packages

Update Raspbian Sudo apt-get update Sudo apt-get upgrade -y

9.1.1 Install Apache2

Sudo apt-get install apache2 -y // -y flag will automatically install required and suggested apache 2 packages Sudo a2enmod rewrite Sudo service apache2 restart Allow .htaccess overrides in the /var/www directory Sudo nano /etc/apache2/apache2.conf Change “AllowOverride None” to “AllowOverride All” and restart apache service Sudo service apache2 restart Find your ip address using “ifconfig” it will be 192. 75.X.X. Using any device, you will see the apache page if you enter this ip address in your browser.

9.1.2 Install PHP

Sudo apt-get install php libapache2-mod-php -y Now delete the default Apache2 Debian page and restart apache service Sudo rm index.html Sudo service apache2 restart

9.1.3 Install MySQL

Sudo apt-get install mariadb-server mariadb-client php-mysql -y Sudo service apache2 restart

9.1.4 Install PhpMyAdmin

Sudo apt-get install PhpMyAdmin -y • Automatic configuration? Choose apache2 with spacebar, hit Tab then Enter • Configure database for PhpMyAdmin with dbconfig-common? Yes • PhpMyAdmin application password -[Enter new password] [confirm new password]

9.1.5 Install and setup FTP

Install and setup VSFTPD and lock-down user pi to the /var/www folder. sudo apt-get install vsftpd -y // Install the FTP service Open the sudo nano /etc/vsftpd.conf file and comment out the following two options:

```
1 local_enable=YES   change to:  #local_enable=YES
2 ssl_enable=NO      change to:  #ssl_enable=NO
```

Then add this code to the very bottom of the file:

```
1 # CUSTOM
2 ssl_enable=YES
3 local_enable=YES
4 chroot_local_user=YES
5 local_root=/var/www
6 user_sub_token=piftp
7 write_enable=YES
8 local_umask=002
9 allow_writeable_chroot=YES
10 ftpd_banner>Welcome to my Raspberry Pi FTP service.
```

Now we need to add a dedicated FTP user called piftp, so we enter the following commands and answer the questions as shown:

```
1 sudo adduser piftp
2 [enter password]
3 [confirm password]
4 Full Name: [type your name]
5 Room Number: [press enter]
6 Work Phone: [press enter]
7 Home Phone: [press enter]
8 Other: [press enter]
9 Is this information correct?: y [press enter]
10 Important Note:
11     password:
12     PiftpRamzeoboor68
13     FullName =Negar
14     Room=1
15     work=2
16     Home=3
```

We also need to add user piftp to the www-data group, give ownership of the /var/www folder to www-data user and group, change user piftp's home folder to same, and loosen some permissions on the /var/www folder:

```
1 sudo usermod -a -G www-data piftp
2 sudo usermod -m -d /var/www piftp
3 sudo chown -R www-data:www-data /var/www
4 sudo chmod -R 775 /var/www
5 Restart the VSFTPD service:
6 sudo service vsftpd restart
7 Login to FTP (VSFTPD) Service
8 Sudo apt-get install filezilla
```

Open the filezilla by opening the application menu in Raspberry Pi , selecting internet, then FileZilla In FileZilla go to File tab, then site manager, new site and insert below:

```
1
2 Host      192.xxx.x.xxx (IP address of your Pi with no prefix)
3 Port      21
4 Protocol  FTP (File Transfer Protocol)
5 Encryption Use explicit FTP over TLS if available
6 Logon Type Normal (username & password)
7 Username  piftp
8 Password  [enter password]
```

Make a table in mysql Go to phpmyadmin make a table with all the needed variables. Here, I named the table sensorReadings. after opening the table, you may see this warning: Warning in ./libraries/sql.lib.php613 count(): Parameter must be an array or an object that implements Countable to solve this:

```
1 sudo sed -i "s/|\s*\((count(\$analyzed_sql_results\
2 ['select_expr'\])\)/|_(\1)/g" /usr/share/phpmyadmin/libraries/sql.
lib.php
```

Finally, put all the php files in the /var/www/html folder (Fig 9.1).

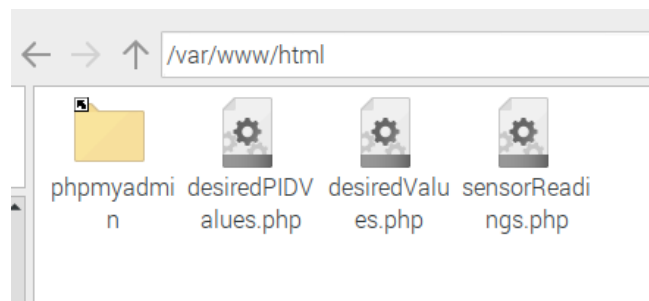


Figure 9.1: PHP files in /var/www folder

Here we can see the PHP file for sensor readings:

```
1 <?php
2 class sensorReadings{
3     public $link = '';
4     function __construct($Height, $Angle, $Temperature, $Humidity,
5         $R1, $B1, $Par1, $R2, $B2, $Par2){
6         $this->connect();
7         $this->storeInDB($Height, $Angle, $Temperature, $Humidity, $R1,
8             $B1, $Par1, $R2, $B2, $Par2);
9     }
10    function connect(){
11        $this->link = mysqli_connect('127.0.0.1','admin','
12            RamzeoboorAdmin') or die('cannot_connect_to_the_BD');
13        mysqli_select_db($this->link, 'phpmyadmin') or die('cannot_
14            connect_to_the_BD');
15    }
16    function storeInDB($Height, $Angle, $Temperature, $Humidity, $R1
17        , $B1, $Par1, $R2, $B2, $Par2){
18        $query = "insert_into_sensorReadings_set_Height='". $Height."',_
19            Angle='". $Angle."',_Temperature='". $Temperature."',_Humidity
20            ='". $Humidity."',_R1='". $R1."',_B1='". $B1."',_Par1='". $Par1.
21            "' ,_R2='". $R2."',_B2='". $B2."',_Par2='". $Par2."";
22        $result = mysqli_query($this->link, $query) or die('Errant_
23            query:_' . $query);
24    }
25 }
26 if($_GET['Height'] != '' and $_GET['Angle'] != '' and $_GET['
27     Temperature'] != '' and $_GET['Humidity'] != '' and $_GET['R1'
28     ] != '' and $_GET['B1'] != '' and $_GET['Par1'] != '' and
29     $_GET['R2'] != '' and $_GET['B2'] != '' and $_GET['Par2'] != '
30     '){
31     $sensorReadings = new sensorReadings($_GET['Height'], $_GET['
32         Angle'], $_GET['Temperature'], $_GET['Humidity'], $_GET['R1'
33         ], $_GET['B1'], $_GET['Par1'], $_GET['R2'], $_GET['B2'],
34         $_GET['Par2']);
35 }
36 ?>
```

Here we can see the PHP file for PID values:

```
1 <?php
2 $con=mysqli_connect("127.0.0.1","admin","RamzeoboorAdmin","
3     phpmyadmin");
4 // Check connection
5 if (mysqli_connect_errno()) {
6     echo "Failed_to_connect_to_MySQL:_" . mysqli_connect_error();
7 }
```

```

7 $query = "SELECT_x_kp,_x_ki,_x_kd,_theta_kp,_theta_ki,_theta_kd_
    FROM_desiredPIDvalues_ORDER_BY_Time_DESC_LIMIT_1";
8 $result = mysqli_query($con,$query);
9
10 while($row = mysqli_fetch_array($result)) {
11     echo ",";
12     echo $row['x_kp'];echo ",";
13     echo $row['x_ki'];echo ",";
14     echo $row['x_kd'];echo ",";
15     echo $row['theta_kp'];echo ",";
16     echo $row['theta_ki'];echo ",";
17     echo $row['theta_kd'];
18 }
19 ?>

```

Here we can see the PHP file for desired motion values:

```

1 <?php
2 $con=mysqli_connect("127.0.0.1","admin","RamzeoboorAdmin","
    phpmyadmin");
3 // Check connection
4 if (mysqli_connect_errno()) {
5     echo "Failed_to_connect_to_MySQL:_ " . mysqli_connect_error();
6 }
7 $query = "SELECT_desiredAngle,_desiredHeight_FROM_
    desiredReferences_motionControl_ORDER_BY_Time_DESC_LIMIT_1";
8 $result = mysqli_query($con,$query );
9
10 while($row = mysqli_fetch_array($result)) {
11     echo $row['desiredAngle'];
12     echo $row['desiredHeight'];
13 }
14 ?>

```