Determination of cycle time constraints in case of link failure in closed loop control in Internet of Things

by

Arpit Ainchwar

Thesis submitted
In partial fulfillment of the requirements
For the Master's in Applied Science degree in
Mechanical Engineering

uOttawa

Department of Mechanical Engineering
Faculty of Engineering

University of Ottawa

Abstract

In today's era of the Internet of Things, it is crucial to study the real-time dependencies of the web, its failures and time delays. Today, smart grids, sensible homes, wise water networks, intelligent transportation, infrastructure systems that connect our world over are developing fast. The shared vision of such systems is typically associated with one single conception Internet of Things (IoT), where through the deployment of sensors, the entire physical infrastructure is firmly fastened with information and communication technologies; where intelligent observation and management is achieved via the usage of networked embedded devices.

The performance of a real-time control depends not only on the reliability of the hardware and software used but also on the time delay in estimating the output, because of the effects of computing time delay on the control system performance. For a given fixed sampling interval, the delay and loss issues are the consequences of computing time delay. The delay problem occurs when the computing time delay is non-zero but smaller than the sampling interval, while the loss problem occurs when the computing time delay is greater than, or equal to, the sampling interval, i.e., loss of the control output. These two queries are analyzed as a means of evaluating real-time control systems. First, a general analysis of the effects of computing time delay is presented along with necessary conditions for system stability. In this thesis, we will focus on the experimental study of the closed loop control system in the internet of things to determine the cycle time constraints in case of link failure.

Dedication

This thesis is dedicated to my parents without whose prayers, unmatchable hard work and support it wouldn't have been possible to pursue my dreams, their constant and undoubted support for me which helped in this hard but beautiful journey of mine, and last but not the least my younger sister, Rutuja, whose unabashed love for me, encourages me to achieve the goals.

Acknowledgements

I would like to express my deep gratitude to my supervisor Dr. Dan Necsulescu, who saw the capability in me and took me under his supervision and guided me through this tough journey. His support and encouragement made it possible for me to finish this work, his reviews and inputs always helped me improve different aspects of this thesis.

Also, I would like to thank all my colleagues in the lab Mohit Sain, Alireza Mirghesani, Vishal Koppula, Aliakbar Baadliwala and Hamid Reza Fallah whose ideas, help and support played a big part in the completion of this thesis.

Especially, I would like to thank my colleague Jasmeet Sigh for helping me out with the MPC simulation results which eventually helped me towards the partial fulfilment of my thesis work.

I would also like to thank my friends Jaydip Kathiriya, Amit Damani, Aswadh, Pranay Sharma, and Arindam Banerjee who stood with me and supported me in this journey.

A sincere thank you to Harpreet Dhaliwal and Tarundeep Dhiman for diligent proofreading of this thesis.

Most importantly I would like to thank my parents who always believed in me and supported my every decision, the Creator who made everything possible for me to achieve and gave me strength, courage and guidance to go through this phase of my life.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This research thesis outlines the Internet of Things (IoT) technology, a rapidly expanding and changing technology that is growing up to bring the next revolution in information systems and computing technologies in general. This thesis explores the key concepts of IoT, data delay and loss problems related to the communication technology. It gives the brief data to study about the internet latency and data loss problems, by performing extensive experimental work.

1.1 Motivation

The Internet of Things (IoT) depends on us. Embedded devices and sensors in automobiles, supermarkets, homes, watches, phones, electronic appliances, industrial and farm equipment, roads and bridges, and wearable technology are already making new kinds of information available and changing the way information is consumed, produced and experienced. IoT certainly portrays an excellent opportunity for improvements in information analysis. The links between IoT and data storage and processing as well as machine learning are obvious and earning attention already.

In this research thesis, we will concentrate on gathering data related to the data delay and data loss in Internet of Things (IoT) environment. We will introduce the experimental system architecture that enables IoT and allows us to gather data to study the internet latency along with the range of

representative applications. From these applications, we will abstract out the major challenges to realize the vision of Internet of Things (IoT) [1].

Internet of Things (IoT) Outline

1. IoT and its enabling technologies and trends
    a. Sensing
    b. Low power and passive technologies
    c. Connectivity
    d. History and positioning of IoT
        i. Synergy with mobile computing
        ii. Ad hoc and fixed sensor networks
        iii. Pervasive and ubiquitous computing
        iv. Participatory sensing
2. Representative applications of IoT
    a. Monitoring in enterprises and civil infrastructure
    b. Power systems
    c. Healthcare
    d. Context-awareness for user interaction
    e. Augmented reality
3. Architectures for the IoT
    a. Core: Power and connectivity
    b. Enablers: clouds; data storage and streaming; linked data
    c. Service abstractions for things and data
    d. Multiple administrative domains

4. Challenges to realizing the IoT
    a. Discovery: dealing with places and physical objects
    b. Delay tolerance and asynchrony
    c. Privacy and security
    d. Administration and sharing of things and data
5. Relevant MAS (multiagent systems) concepts and techniques
    a. Autonomy and heterogeneity
    b. Discovery and selection
        i. Service descriptions
        ii. Reputation: sharing and maintaining
        iii. Trust: engendering trustworthiness
    c. Achieving coherence and cooperation
        i. Sharing, fusing, revising information
        ii. Social choice and elements of voting theory
        iii. Incentives for promoting quality of information
    d. Constructing decentralized MAS
        i. Goals
        ii. Information-based protocols
        iii. Meaning-based protocols
        iv. Maintaining alignment
        v. Observability and compliance
    e. Governing interactions
        i. Models of stakeholder requirements

ii. Norms and normative relationships

iii. Identity

iv. Policies

6. Limitations of MAS concepts and techniques and directions for further research
    a. Accommodating sociotechnical systems
    b. Organizational modeling
    c. Engineering methodologies
    d. Modeling and evaluating dynamical properties

1.2 Research Objective

To increase the productivity of the Internet of Things (IoT) in wireless communication, an experimental study needs to be done on the IoT model, to gather and study the data related to the signal delay and loss. Study of data delay and data loss patterns in IoT environments will help to reduce its effects on the IoT applications and will allow taking preventive measures against the link failure and internet latency [2]. Certain scenarios are tested, and the approach will be refined by using the experimental results.

1.3 Thesis Outline

This thesis consists of 9 chapters that walk the reader through different stages of work related to the Internet of Things, the growing importance of the IoT in real life. This thesis is focused on developing the experimental setups to study the cycle time constraints in link failures in closed loop controls, to make the IoT technology more reliable.

Chapter 2 provides the literature review of the previously done work related with the IoT and previously published research on the internet link failures and the detailed study of the related work done by the scientists around the world. It also addresses the limitations and advantages of such work.

Chapter 3 discusses the Internet of Things in detail about its origin, drivers and applications, technology related to their implementation. Different types of communication models, a requirement of IPv6 and IoT challenges and its applications are also being discussed in this chapter.

Chapter 4 describes the various system configurations used to study the link failures; it also illustrates the different experimental setups developed to collect the data for the study of internet interruptions and gives in detail information about the components used in the experiments.

Chapter 5 outlines the software and design implementation, explains the software used (Arduino IDE) and its structure and functions and includes the code with its detailed explanation,

Chapter 6 detailed information about the experimental results, includes the graphical representation of the results obtained through the experimental study and contains the data drawn from the experiments, which is used for analyzing the results.

At last, Chapter 8 and 9 concludes the thesis by discussing the results obtained, conclusions drawn and ends by throwing light on the recommendations for the future work.

Chapter 2
Literature Review

A wireless sensor network (WSN) is a composition of nodes where each node is fitted with sensors to measure some physical phenomena like pressure, light, temperature, etc. WSNs are considered as one of the information collecting methods to build systems which will improve the efficiency of infrastructure. Compared with the wired solutions, WSNs emphasize on easier deployment and better versatility of devices. With the technological improvement of sensors, WSNs will become the key technology for IoT. Today's world is full of sensors in vehicles, in factories for controlling emissions, in homes, in smartphones, and even in the ground monitoring soil conditions in vineyards. Although it appears that sensors have been here for a while, research on WSNs began back in the 1980s, and it is only since 2001 that WSNs generated an enhanced interest from technical and research perspectives. This is due to the availability of economical, low powered miniature components like microcontrollers, processors, radios and sensors that were often integrated on a single chip (system on a chip(SoC)) [3].

The concept of the Internet of Things (IoT) was formed parallel to WSNs. The term Internet of Things was devised by Kevin Ashton in 1999 [4] and introduced to identifiable objects and their descriptions in an Internet-like structure. These "internet-like" structures can be composed of anything from industrial plants, machines, cars, specific parts of the larger system to animals, human beings and plants and even specifically their body parts. While IoT does not imply a communication technology, wireless communication technologies will play a significant part, and

WSNs have many applications in many industries. The small, sturdy, inexpensive and low powered WSN sensors will bring the IoT to even the smallest objects that can be installed in any environment, at reasonable costs. Integration of these objects into IoT will be a significant part of further development of WSNs.

A WSN can be defined as a system of nodes that cooperatively sense and may regulate the conditions, enabling interaction between persons or computers and the surrounding environment [5]. In fact, the activity of sensing, processing, and communication with an inadequate amount of power, generates a cross-layer perspective typically requiring the joint consideration of distributed signal/data processing, medium access control, and communication protocols [6].

Through integrating existing WSN applications as part of the infrastructure system, possible new applications can be recognized and improved to meet future technology and new market trends. For instance, WSN technology utilizations for the smart grids, smart water, intelligent transportation systems, and smart home generate enormous amounts of data, and this data can serve many purposes.

Time delays and data losses

Network delays are usually short. However, packets can get dropped in a network system. For reliable connections, software is necessary, which checks for losses and for resending. If a resend is required, the overall delay is doubled; another round-trip time is computed for a resend request and response. For higher speed, stable data transfer protocols the impact can be even greater.

The Delay Components

The following given equation describes the packet delay at a single node along its route from source to destination**.** Figure 2.1 shows the delay components in detail.

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc} = nodal processing

d_{queue} = queueing delay

d_{trans}= transmission delay

d_{prop}= propagation delay

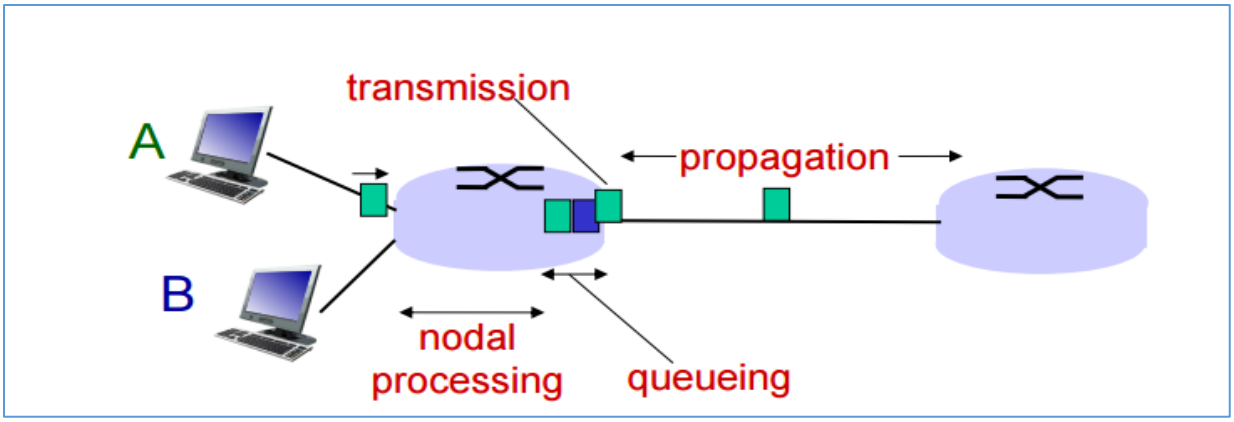

Figure 2.1 Delay components

(Source: Available. [online] http://www.eecs.yorku.ca/course_archive/2014-15/W/3214/LEC/queue.pdf)

Packet Switching: queueing delay, loss

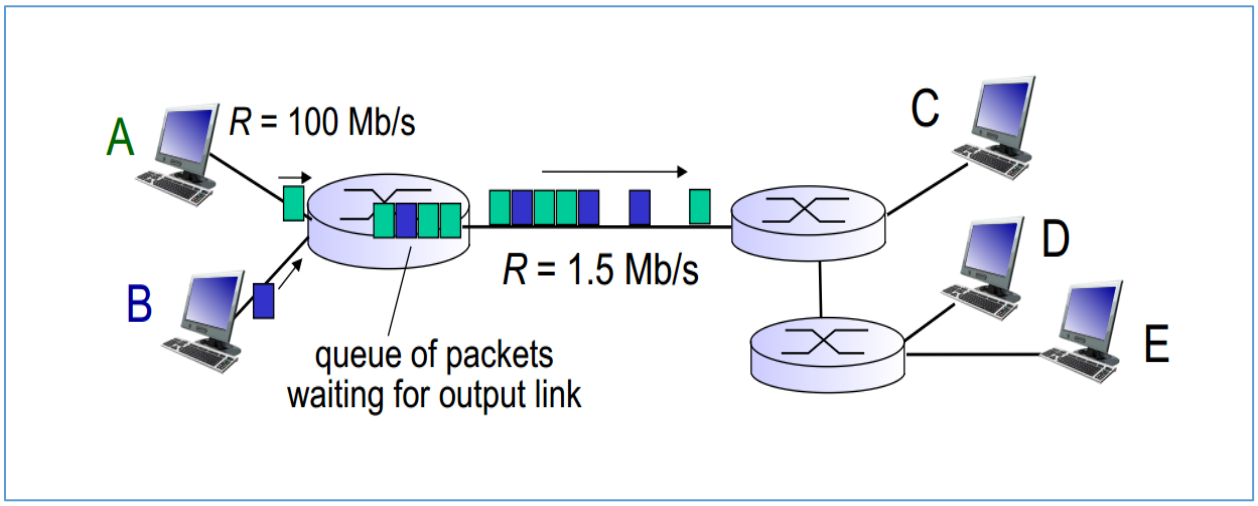

Figure 2.2 Queueing delay

(Source: Available. [online] http://www.eecs.yorku.ca/course_archive/2014-15/W/3214/LEC/queue.pdf)

Queueing and loss

- If arrival rate (in bits) to link exceeds transmission rate of link for a period

    1. Packets will queue, wait to be transmitted on link

2. Packets can be dropped (lost) if memory (buffer) fills up

Figure 2.2 shows how the queueing delay occurs.

- **Packets queue in router buffers**

1. packet arrival rate to link (temporarily) exceeds output link capacity
2. packets queue, wait for turn

Figure 2.3 shows the packets being queued up in router buffers.

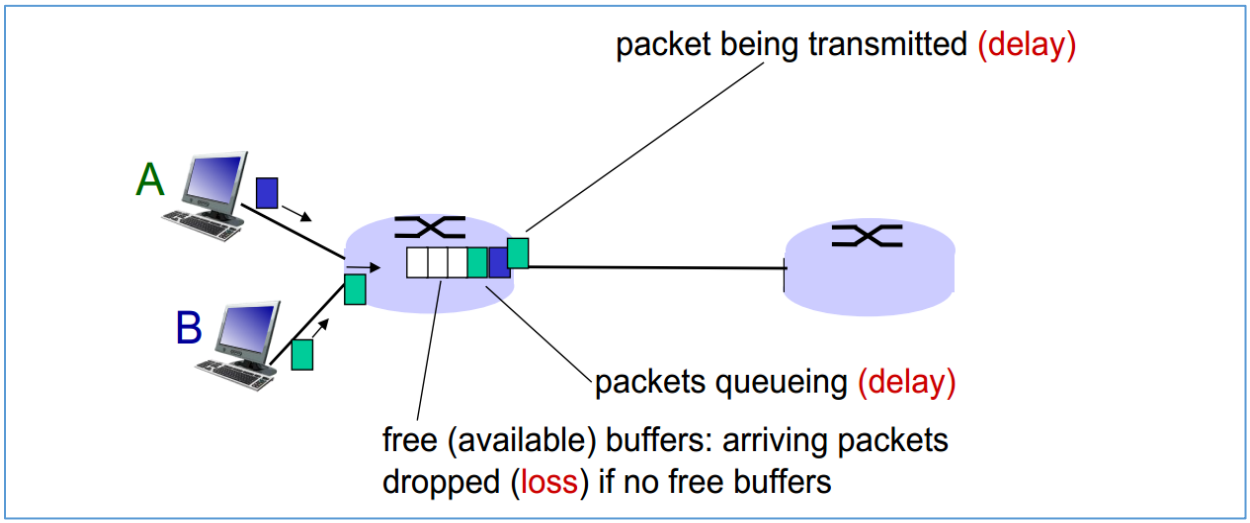

Figure 2.3 Packets queue in router buffers

(Source: Available. [online] http://www.eecs.yorku.ca/course_archive/2014-15/W/3214/LEC/queue.pdf)

Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all

Figure 2.4 describes the condition for packet loss.

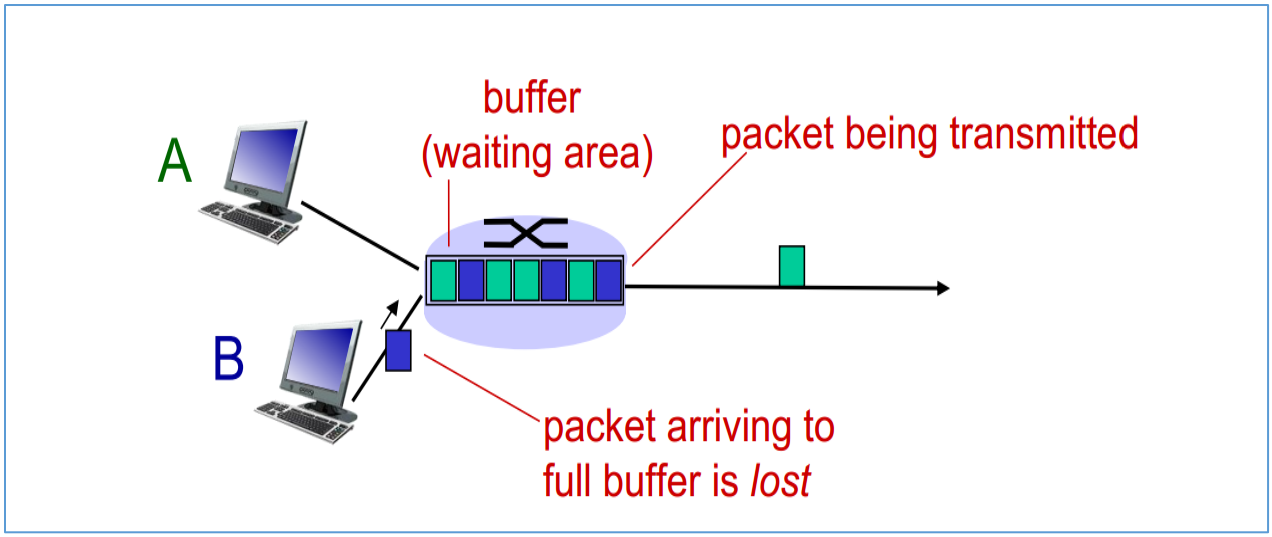

Figure 2.4 Packet Loss

(Source: Available. [online] http://www.eecs.yorku.ca/course_archive/2014-15/W/3214/LEC/queue.pdf)

Throughput

- throughput: rate (bits/time unit) at which bits transferred between sender/receiver
- instantaneous: rate at given point in time
- average: rate over longer period

Figure 2.5 describes the concept of throughput.

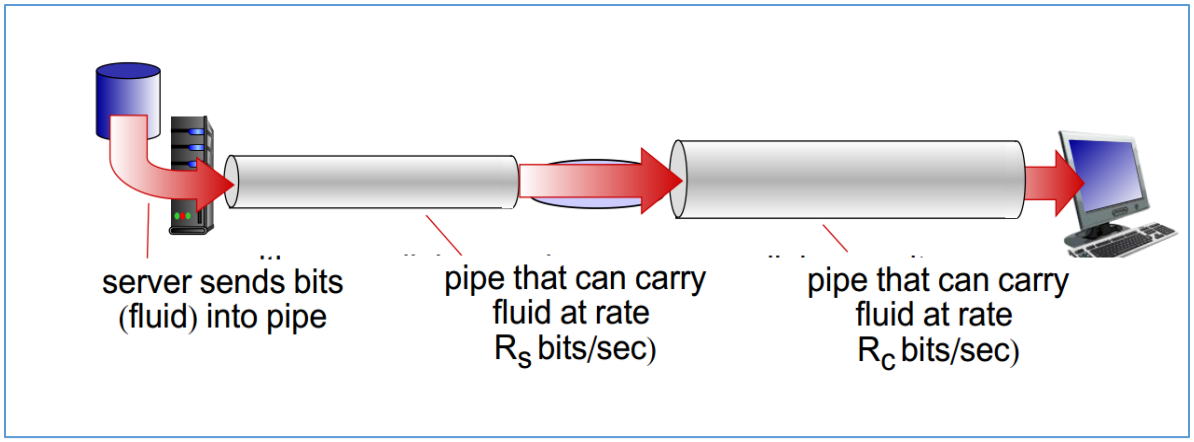

Figure 2.5 Throughput

(Source: Available. [online] http://www.eecs.yorku.ca/course_archive/2014-15/W/3214/LEC/queue.pdf)

Because of the adverse effects of the computing time delay on control system performance, the authenticity of a real-time digital control computer depends not only on the reliability of the hardware and software utilized but also on time delay computing the control output [7]. The delay and loss problems are classified as the effects of computing time delay for a given fixed sampling interval. When the computing time delay is non-zero but smaller than the sampling interval the delay problem occurs, while the loss problem occurs when the sampling interval is shorter than or equal to the computing time delay, i.e., loss of the control output. As a means of evaluating real-time control systems, these two problems are analyzed. First, along with necessary conditions for system stability a generic analysis of the effects of computing time delay is presented. On system stability and system performance, we show both qualitative and quantitative analysis of the computing time delay effects on a robot control system.

Data acquisition from sensors, data processing to generate control/display commands, and to output the results to the actuators/display devices, these steps can be thought as a three-stage or controller computer pipe of a real-time digital control computer. Although each of the three stages will take the time to complete, the research study done in this thesis is concerned only with the time taken by the most difficult stage, data processing, since the other two are much simpler and more static. More precisely, the time is taken to execute programs that implement control algorithms, called the computing time delay. By executing a sequence of instructions, a controller computer executes the underlying control algorithms. The reliability of a digital control system depends not only on the MTBF (mean time between failures) of the controller hardware and software, unlike analog control systems, but also on the delay in executing control algorithms on the controller computer.

The period from its trigger to the generation of a corresponding control command is defined as the execution time for a control algorithm. In a controlled system, it is an extra time delay that is introduced to the feedback loop. Because of the existence of resource sharing delays, conditional branches and processing exceptions, the execution time for the given control algorithm, or the computing time delay, is piecewise continuous, a random variable which is usually smaller than the corresponding sampling interval. Traditional controllers are designed without considering the computing time delay. Thus, it is very crucial to analyze the effects of computing time delay on control system performance when control algorithms are executed on digital computers.

The computing time delay is quite different from the usual system time delay and cannot be taken care of before putting a system in use due to its randomness caused by, for example, the data-dependent branches and loop counts in a program that implements the control algorithm under consideration.

The breakdown of delay equation makes it easy to analyze different delay components. Usually, network delays are small. However, lost packets in the network increase the impact of the delays. The computing time delay is considerably different from the usual system time delay and cannot be taken care of before establishing a system in use.

When the computing time delay is long versus the sampling interval (but small about the mission lifetime), it may severely affect control system performance. Depending on the magnitude of computing time delay about the sampling interval, its effects on the control system are categorized into either a delay or a loss problem. To be more precise, let ξ and T_s, denote the computing time delay and the sampling interval, respectively. A delay problem results when $0 < \xi < T_s$, and the loss problem occurs when $\xi \geq T_s$. The former represents the undesirable effects caused by a non-zero computing time delay smaller than the deadline (that is, the start of the next sampling interval) of a control algorithm or task, while the latter represents the case of no update of control output for one or more sampling intervals.

The sampling rate must be chosen precisely not only satisfying the requirements of the Shannon's sampling theorem [8] but also accomplishing the expected performance. A good example to

confirm this can be found in [9] where series of robot control experiments were conducted with different sampling rates. Time delay computation becomes more pronounced in the case of increasing the sample rate. However, these effects can not be neglected, particularly when the time constant of the plant is short and the order of the plant is high [10].

Before the IoT idea being widely accepted, many challenging issues need to be considered and both technological, as well as social knots, must be united. While guaranteeing privacy, trust and security, principal issues are making a complete interoperability of interconnected devices possible, rendering them with an always higher degree of smartness by enabling their adoption and autonomous behavior. Also, the IoT approach poses numerous new problems concerning the networking aspects [11].

In fact, the things composing the IoT will be characterized by low resources regarding both computation and energy capacity. Accordingly, the proposed solutions need to pay attention to support performance beyond the obvious scalability problems [12].

Chapter 3

Internet of Things (IoT)

3.1 Executive Summary

Internet of Things is an emerging subject of technological and economic significance. Consumer products, industrial and utility components, sensors and other everyday objects are being combined with internet connectivity, and powerful data analytic capabilities promise to transform the way we work, live and play.

However, at the same time, the IoT raises significant challenges that could stand in the way of realizing its potential. Technical difficulties such as connectivity and reliability are one of the many. The IoT involves a broad set of ideas that are complex and intertwined from different perspectives.

Connectivity Models: IoT implementations use various technical communications models, each with its individual properties. Four general communications standards described by the Internet Architecture Board comprises Device-to-Device, Device-to-Cloud, Device-to-Gateway, and Back-End Data-Sharing. This highlight the flexibility in the ways that IoT devices can combine and provide content to the user.

3.2 Introduction

In both the specialty press and the popular media, the Internet of Things is an important topic in the technology industry, policy and engineering circles and has become headline news. This technology is integrated into a broad spectrum of networked products, systems, and sensors, which take advantage of advancements in computing power, electronics miniaturization, and network interconnections to offer new capacities which were not earlier possible. The prospective impact of the "IoT revolution"- from new market opportunities and business models to concerns about technical interoperability, security and privacy has been discussed in abundance of news articles, reports and conferences.

New IoT products like home automation components, Internet-enabled devices and power management devices are moving us towards a vision of the "smart home", offering more energy efficiency and security. This is possible because of the large-scale implementation of IoT devices that promises to modify many perspectives of the way we live. The way healthcare services being delivered is transforming through personal IoT devices like wearable fitness and health monitoring devices and network enabled devices. This technology is proving very beneficial for people with disabilities and the elderly, allowing quality and improved level of independence at a lower cost. To minimize congestion and energy consumption, the idea of "smart cities" is realized through IoT systems like networked vehicles, intelligent traffic systems and sensors embedded in roads and bridges. By increasing the availability of information along the value chain of production using networked sensors in Internet of Things, technology offers the possibility to transform agriculture

industry and energy generation and distribution. However, IoT raises many issues and challenges that need to be considered and addressed for potential benefits to be realized.

Some companies and research organizations have suggested a wide range of predictions about the possible impact of IoT on the Internet and the economy during the next five to ten years. Cisco, for example, projects more than 24 billion Internet–connected objects by 2019 [13]; Morgan Stanley, however, projects 75 billion networked devices by 2020 [14]. Looking out further and raising the stakes higher, Huawei forecasts 100 billion IoT connections by 2025 [15]. McKinsey Global Institute suggests that the financial impact of IoT on the global economy may be as much as $3.9 to $11.1 trillion by 2025 [16]. While the variability in predictions makes any specific number questionable, collectively they paint a picture of significant growth and influence.

3.3 Internet of Things Communications Models.

It is beneficial to think about how IoT devices connect and communicate regarding their unique communication models from an operational perspective. The below discussion presents the framework of different communication models and explains main features of each model in the framework [17].

3.3.1 Device-to-Device Communications

This type of communication model describes two or more devices that connect directly and communicate with each other, rather than using an intermediary application server. Including IP (Internet Protocol) networks or the Internet these devices communicate over many types of networks. These devices often use protocols like Bluetooth [18], Z-Wave [19], or ZigBee [20] to establish direct device-to-device communication, as shown in Figure 3.3.

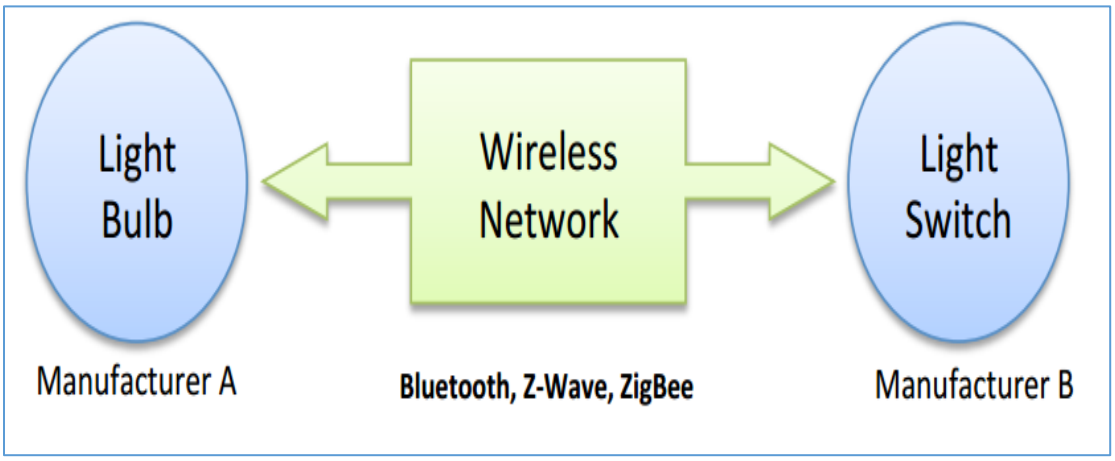

Figure 3.3 Example of device-to-device communication model.

(Source: Dave Thaler, Hannes Tschofenig, Mary Barnes. "Architectural Considerations in Smart Object Networking" Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. [Online]. Available: https://www.rfceditor.org/rfc/rfc7452.txt.)

These device-to-device systems allow devices that are based on a communication protocol to communicate and exchange their messages to achieve their function. This communication model typically uses small data packets of information systems to connect to devices with relatively low data rate requirements which are commonly employed in appliances such as home automation

systems. In a home automation scenario, residential IoT devices like light bulbs, thermostats, door locks and light switches usually send small numbers of information to each other.

These devices use device-specific data models that require redundant development efforts by device producers [21], but they also often have a direct relationship, they usually have built-in security and trust mechanism. This means that to implement device-specific data formats rather than open approaches that enable the use of standard data formats, the device manufacturers need to invest in development efforts.

This usually means that underlying device-to-device communication protocols are not agreeable, forcing the user to choose a group of devices that employ a general protocol, from the user's point of view. For example, ZigBee family of devices is not compatible with the family of devices using Z-Wave protocol. The user benefits from understanding that products within a family tend to communicate well, while these variances limit user choice to devices within an appropriate protocol family.

3.3.2 Device-to-Cloud Communications

The IoT device, in a device-to-cloud communication model, connects directly to an Internet cloud service as an application service provider to transfer data and control message traffic. This approach frequently takes advantage of existing communications mechanisms like traditional wired Ethernet or Wi-Fi connections to build a connection between the device and the IP network, which ultimately connects to the cloud service. As shown in Figure 3.4.

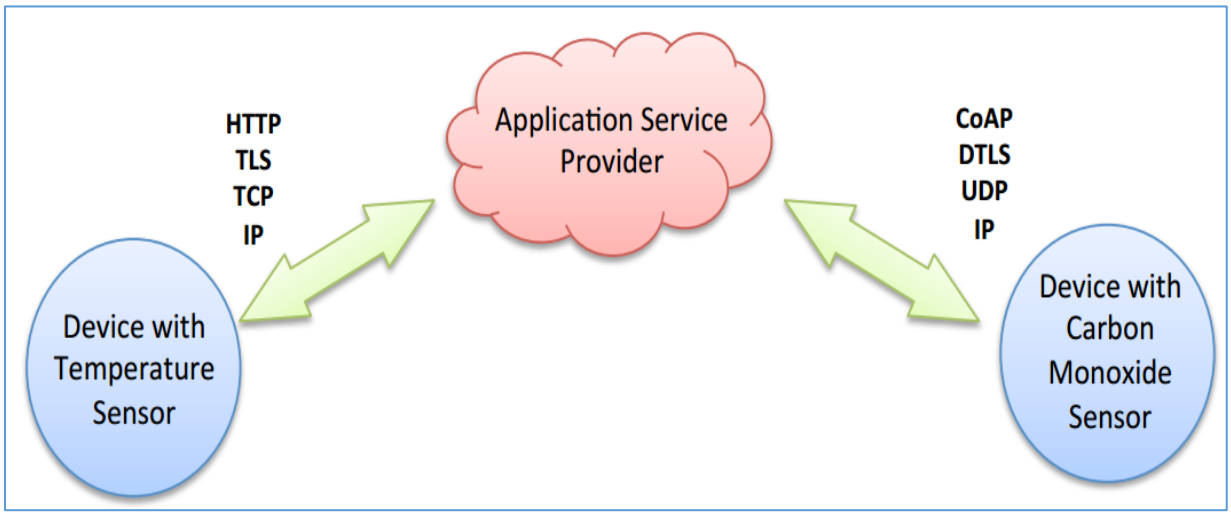

Figure 3.4 Device-to-cloud communication model diagram.

(Source: Dave Thaler, Hannes Tschofenig, Mary Barnes. "Architectural Considerations in Smart Object Networking" Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. [Online]. Available: https://www.rfceditor.org/rfc/rfc7452.txt.)

Some great consumer IoT devices like the Nest Labs Learning Thermostat [22] and the Samsung SmartTV [23] have employed this communication model. In the case of Nest Learning Thermostat, to a cloud-based database, the data is transmitted by the device where it can be utilized to analyze home energy consumption. The user is allowed the remote access to their thermostat via the Web interface or a smartphone using the cloud connection. The television uses an internet connection

to transfer user viewing information to Samsung for analysis and to enable the voice recognition features of the TV in case of the Samsung SmartTV. In these cases, the device-to-cloud model adds advantage to the end user by increasing the capacities of the device beyond its original features.

However, when venturing to integrate devices made by different manufacturers, interoperability difficulties may occur. Generally, the cloud service and device(s) are from the same vendor [24]. The device master or user may be tied to a cloud service if proprietary data protocols are applied between the device and the cloud service, restricting or preventing the use of alternative service providers. This is usually related to as "vendor lock-in'', a term that incorporates other facets of the relationship with the provider such as possession of and access to the data. At the same time, users can have confidence that devices designed for the platform can be integrated.

3.3.3 Device-to-Gateway Model

More typically, the device to gateway model or the device to application layer gateway (ALG) model, the IoT device connects by an ALG service as a channel to reach a cloud service. This means that there is an application software running on a local gateway device, which acts as an intermediate between the device and the cloud service and provides security and other functionality such as data protocol translation. The model is shown in Figure 3.5.

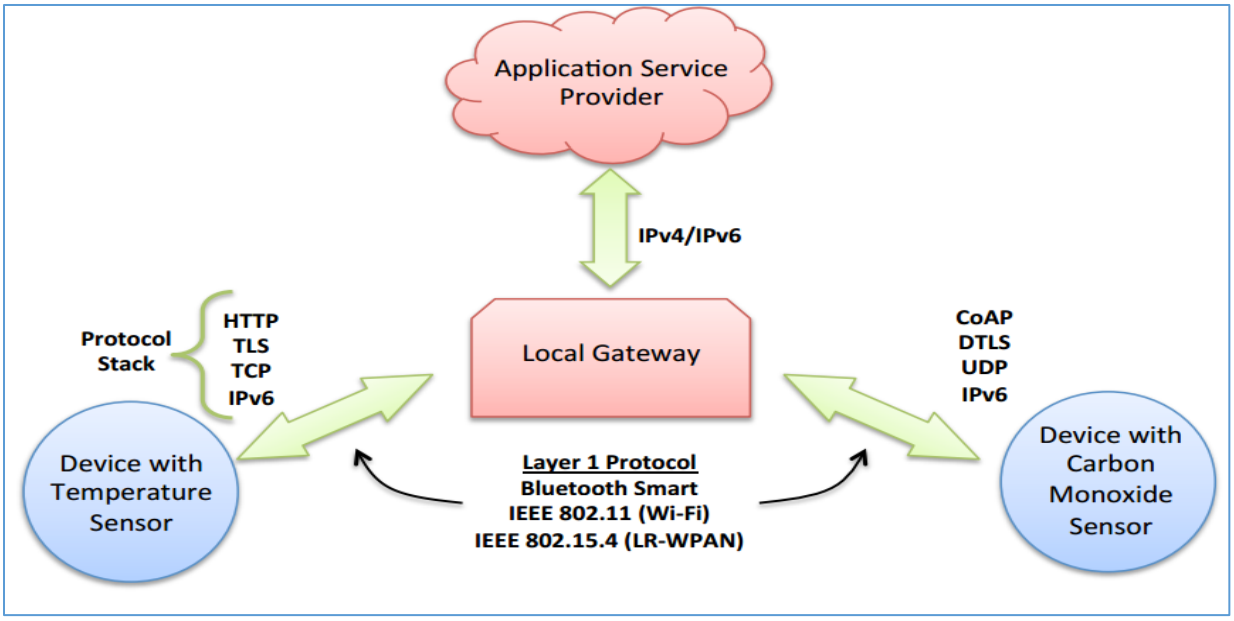

Figure 3.5* Device-*to-gateway communication model diagram.

(Source: Dave Thaler, Hannes Tschofenig, Mary Barnes. "Architectural Considerations in Smart Object Networking" Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. [Online]. Available: https://www.rfceditor.org/rfc/rfc7452.txt.)

In consumer devices, several forms of this model are formed. A smartphone running an app to communicate with a device and relay data to a cloud service is the local gateway device in many cases. This is often the model used with popular consumer objects like personal fitness trackers. They often rely on smartphone app software to work as a mediator gateway to connect the fitness device to the cloud as these devices do not have the native ability to connect directly to a cloud service.

Development of "hub" devices in home automation utilization is the other form of this device-to-gateway model. As they bridge the interoperability gap between devices themselves, they can also serve as a local gateway between different IoT devices and a cloud service.

For example, Z-Wave and Zigbee transceivers installed to communicate with both families of devices [25] in the SmartThings hub which is a stand-alone gateway device. Allowing the user to gain access to the devices using a smartphone app and an Internet connection, it then connects to the SmartThings cloud service.

3.3.4 Back-End Data-Sharing Model

In combination with data from other sources, the back-end data-sharing model relates to a communication structure that allows users to export and analyze smart objects data from a cloud service. This architecture supports “the [user’s] desire for granting access to the uploaded sensor data to third parties”.

This method can be considered as an expansion of the single device to cloud communication model. That can also lead to data pits where "IoT devices can upload data only to a single application service provider". A back-end sharing architecture permits the data gathered from individual IoT device data streams to be aggregated and analyzed.

For example, in charge of an office complex, a corporate user, involved in consolidating and analyzing the energy consumption and utility data produced by all the IoT sensor devices and Internet-enabled utility systems on the premises.

The data all IoT sensor or system produces sit in a stand-alone data pits, often in the single device-to-cloud model. To quickly access and analyze the data in the cloud produced by the whole spectrum of devices in the building, back-end data sharing method is very useful. Also, this kind of structure facilitates data portability needs. Robust back-end data sharing designs enable users to move their data when they switch between IoT services, breaking down traditional data pit boundaries.

The back-end data-sharing model implies a federated cloud services approach, applications programmer interfaces (APIs) are needed to achieve interoperability of smart device data received in the cloud. A graphical representation of this design is shown in Figure 3.6.

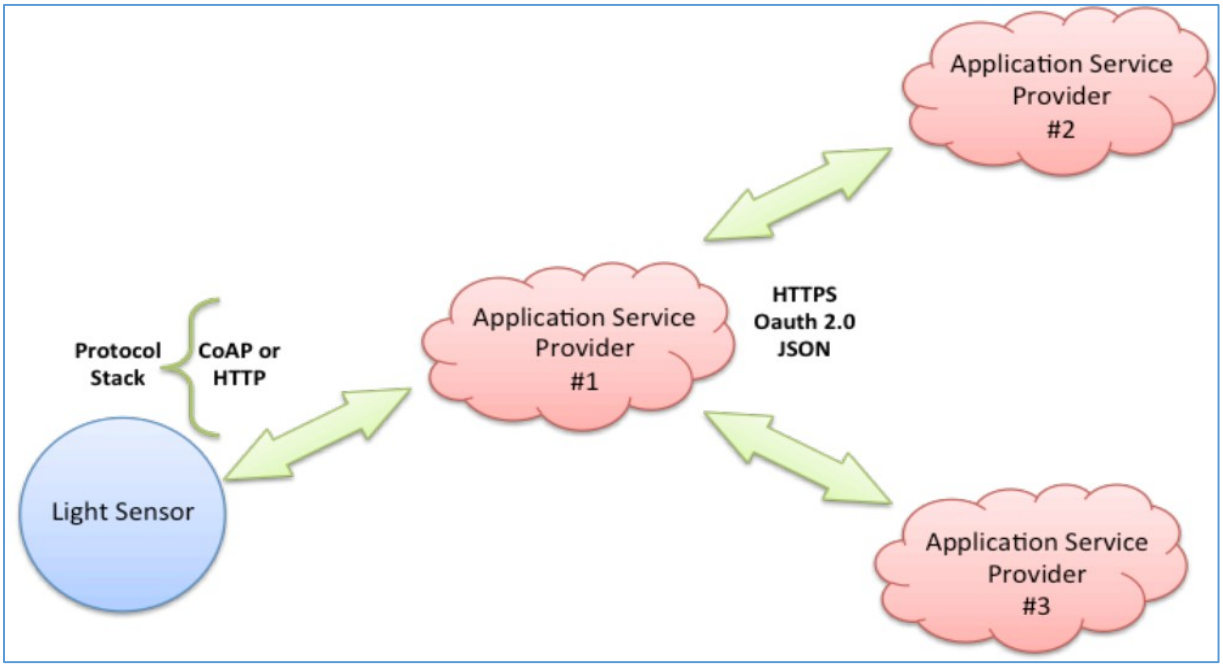

Figure 3.6 Back-end data sharing model diagram.

(Source: Dave Thaler, Hannes Tschofenig, Mary Barnes. "Architectural Considerations in Smart Object Networking" Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. [Online]. Available: https://www.rfceditor.org/rfc/rfc7452.txt.)

Chapter 4

System Configuration

4.1 Experimental Setup System 1 (LED and Photosensor open loop control)

Figure 4.1 shows an experimental setup to study the time delay and data losses in the given wireless network. The part A consists of (Arduino Uno + Wi-Fi Shield + LED) and the part B consists of (Arduino Uno + Wi-Fi Shield + Photocell). Both the parts are enclosed in the box to ensure the better readings. LED attached to the Arduino shield can be operated (blinked) continuously with regular time interval using the program running on the Arduino through IDE software.

The photocell connected to the other Wi-Fi shield will sense the blinking of the LED and sends the reading to the attached server, and it gets recorded subject to its time constraints. The recorded data can be studied for the time delays and the link failures. The timing of each data recorded let us know the time delay in execution of the loop. If the time delay is greater than that of the sampling interval, there is a loss of the data, which we can find out from the experimentally recorded data and the time related to it.

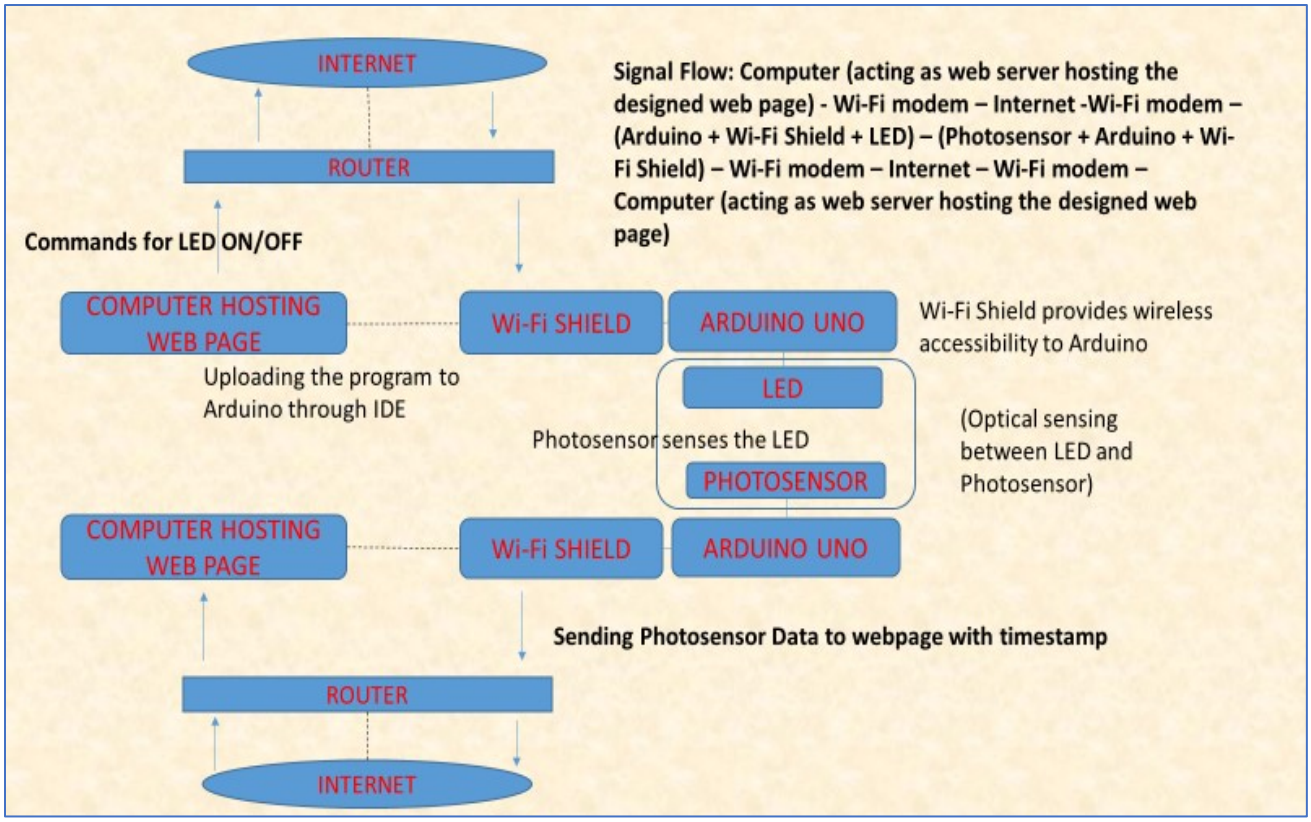

Figure 4.1 Experimental setup of system 1

Signal Flow: Computer (acting as web server hosting the designed web page) - Wi-Fi modem – Internet -Wi-Fi modem – (Arduino + Wi-Fi Shield + LED) – (Photosensor + Arduino + Wi-Fi Shield) – Wi-Fi modem – Internet – Wi-Fi modem – Computer (acting as web server hosting the designed web page).

4.2 System Configuration-Experimental Setup 1

The experiment requires two sets of Arduinos Uno™ and Arduino Wi-Fi Shield 101™ connected with each other in two separate parts. One system of (Arduino Uno + Arduino Wi-Fi Shield) is connected to the LED, while the other part contains the photocell.

4.3 Arduino Uno™

Arduino Uno is a board based on the microcontroller ATmega328P. It includes 14 digital input/output pins (of which six can be used as PWM outputs), and six pins are for analog inputs; it also has 16MHz quartz crystal, a power jack, a USB connection, a reset button and an ICSP header. It comprises everything needed to support the microcontroller. It needs to connect it with a computer USB cable, or we can power it with an AC-to-DC or battery. The UNO board is the reference model for the Arduino platform, the first in the series of USB Arduino Boards.

The Arduino board can be programmed with Arduino Software (IDE). The ATmega328 on this board comes preprogrammed with the bootloader which allowed you to upload the new code without any use of the external hardware programmer. It uses the original STK500 protocol. We can also program the microcontroller through the ICSP (In-Circuit Serial Programming) by bypassing the bootloader.

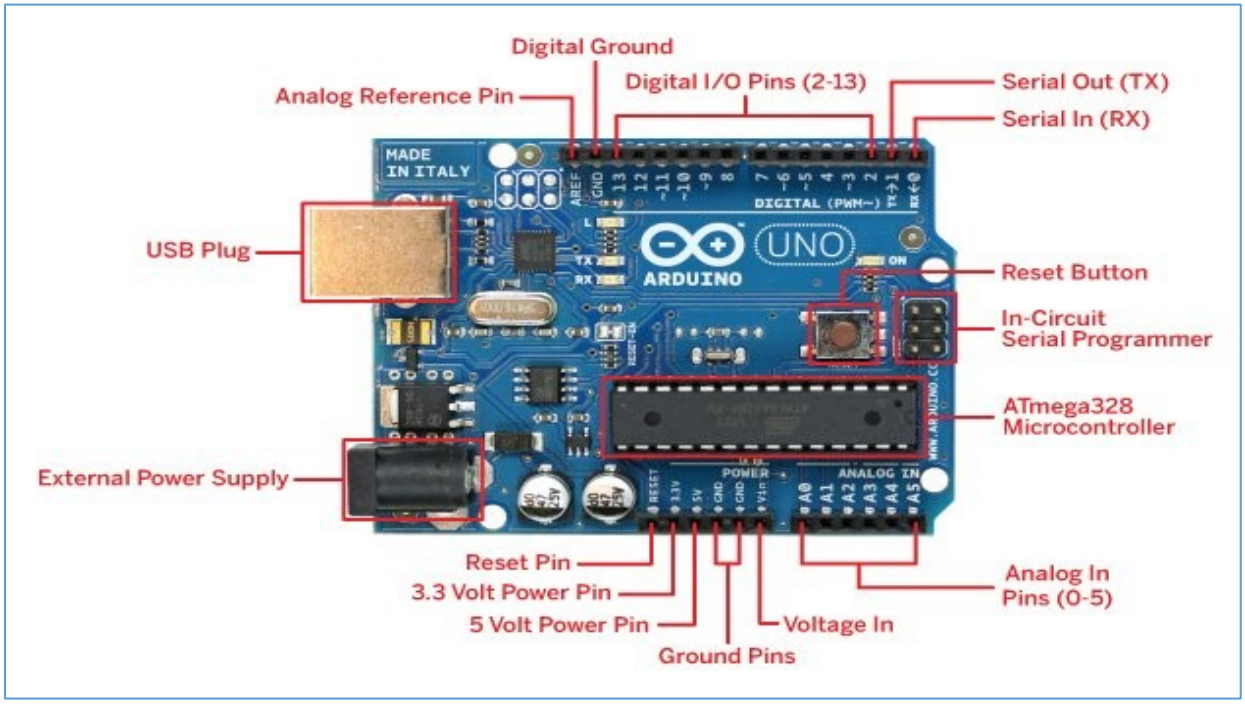

Figure 4.2 Arduino Uno™

(Source: Available. [online] https://www.robomart.com/image/catalog/RM0058/02.jpg)

Arduino UNO does not use the FTDI USB-to-serial driver chip; instead, it features the Atmega 16U2 programmed as a USB-to-serial converter. This microcontroller board can be powered externally or via the USB connection. The power source is selected automatically. Figure 4.2 shows in detail the components of Arduino Uno™.

4.3.1 Arduino Uno™ Technical Specifications:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital, I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by boot loader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

4.4 Arduino Wi-Fi 101 Shield™

Wi-Fi 101™ shield is a powerful IoT shield with crypto-authentication which allows you to wirelessly connect the Arduino, developed with ATMEL by using the IEEE 802.11 wireless specifications (Wi-Fi). Compliant with the IEEE 802.11 b/g/n standard the shield is based on the Atmel SmartConnect-WINC1500 module. The WINC1500 module is a network controller capable

of TCP and UDP protocols. This shield is designed specifically for the IoT applications and features a hardware encryption/decryption security protocol implemented by the ATECC508A Crypto-Authentication chip which is an ultra-secure method to provide key agreement for encryption/decryption.

- Operating voltage both 3.3V and 5V (supplied from the host board)
- Connection via: IEEE 802.11 b/g/n for up to 72 Mbps networks
- Encryption types: WEP and WPA2 Personal
- Support TLS 1.1 (SHA256)
- Connection with Arduino or Genuino on SPI port
- Onboard CryptoAuthentication by ATMEL

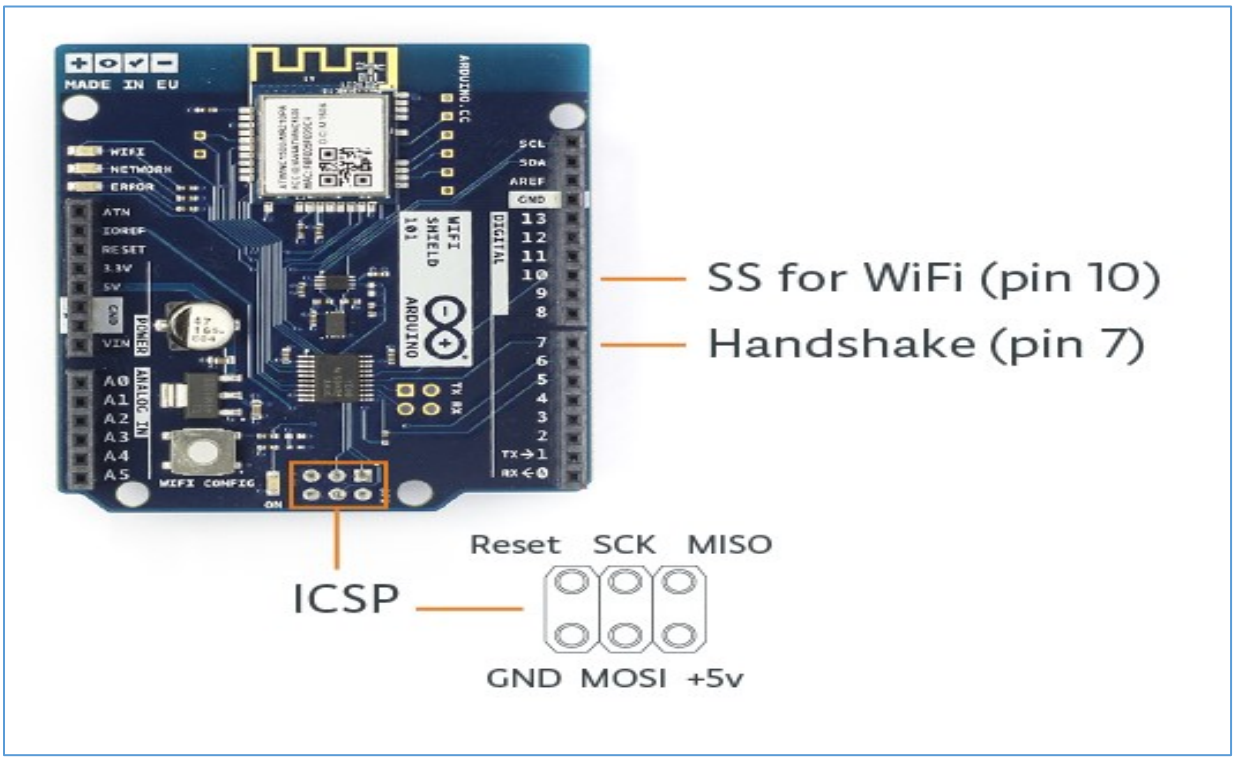

Figure 4.3 Arduino Wi-Fi 101 Shield™

(Source: Available [online]. https://www.arduino.cc/en/Guide/ArduinoWiFiShield101)

4.5 LED

Light-emitting diode (LED) is a light source with a two-lead semiconductor. It emits light when activated. It is a p-n junction diode. These are typically small of the size less than 1 mm^2. To shape its radiation pattern integrated optical patterns may be used. Figure 4.4 shows the components of LED in general.

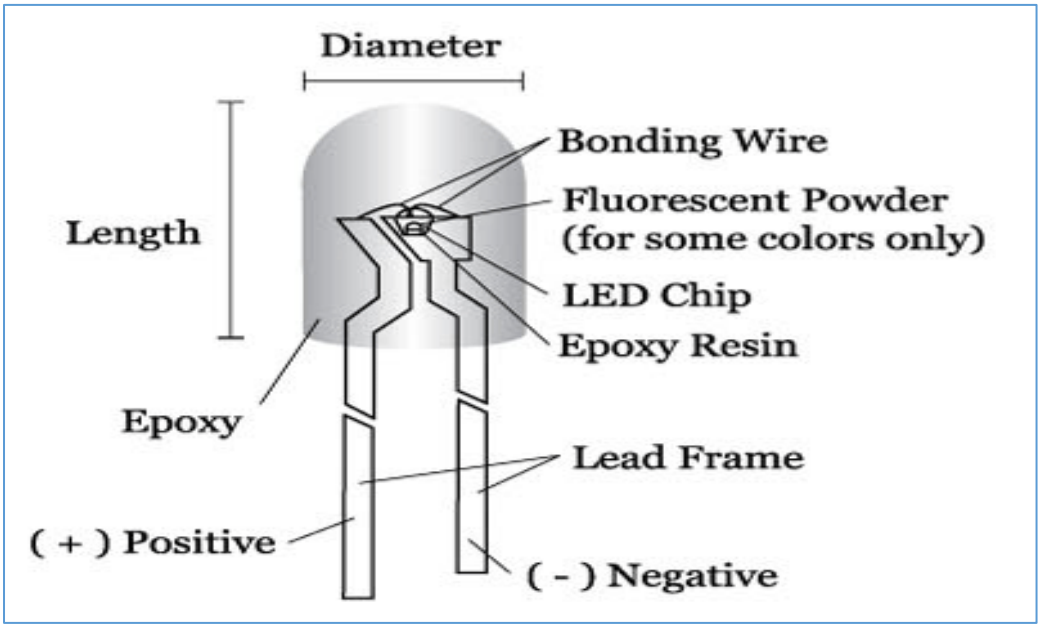

Figure 4.4 LED

(Available [online] http://demo.realgreenled.com/technology/Difference-between-round-LED-bulbs-and-concave-LED-bulbs.html)

Absorbed light energy can get converted into a proportional electric current through the P-N junction. In LED working principal, the same process is reversed, i.e., the, when the light energy applied to the, LED it emits light. This phenomenon is usually described as electroluminescence, which can be explained as under the influence of electric field, emission of light from a

semiconductor. As the electrons cross from the N-region and recombine with the holes existing in the P-region, the charge carriers combine in a forward-biased P-N junction. The conduction band of energy levels contains free electrons, while there are holes in the valence energy band. Thus, creating the energy difference, as the energy levels of the holes is less than the energy levels of the electrons. To recombine the holes and the electrons some portion of the energy must be dissipated. This energy is released in the form of heat and light.

Generally, for silicon and germanium diodes the electrons dissipate energy in the form of heat while, the electrons dissipate energy by emitting photons in the case of gallium arsenide phosphide (GaAsP) and gallium phosphide (GaP) semiconductors. The junction becomes the cause of light as it is emitted if the semiconductor is translucent, thus becoming a light emitting diode. However, in the case of the reverse biased junction, no light is produced by the LED, and the device gets damaged if the potential is high enough. Figure 4.5 describes the working principle of LED.

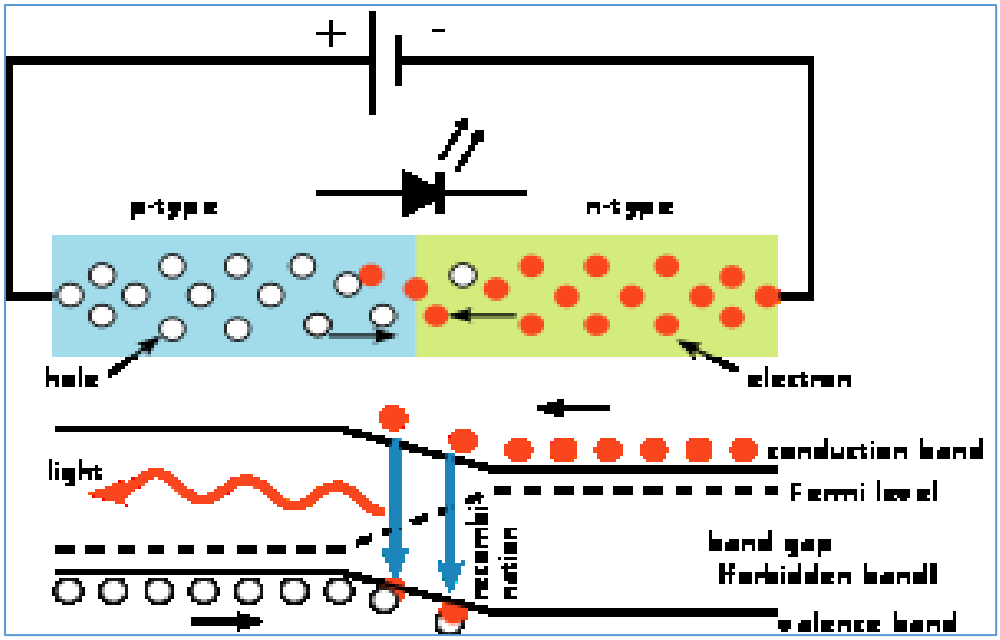

Figure 4.5 The inner working of an LED

(Source. Available [Online]: https://en.wikipedia.org/wiki/Light-emitting diode)

4.5.1 Efficiency and operational parameters

Typical indicator LEDs are devised to work within limits than 30–60 milliwatts (mW) of electrical power. LED-based lighting sources key advantage is high luminous efficiency. The efficiencies of the light-emitting diode are calculated in experimental conditions of the lab, held at low temperature. Real life efficiencies are much lower since the LEDs installed in real fixtures operate with the driver losses at high temperature.

Color	Wavelength range (nm)	Typical efficiency coefficient	Typical efficacy (lm/W)
Red	$620 < \lambda < 645$	0.39	72

Red-Orange	610 < λ < 620	0.29	98
Green	520 < λ < 550	0.15	93
Cyan	490 < λ < 520	0.26	75
Blue	460 < λ < 490	0.35	37

4.6 Photoresistor

The photoresistor is made of a high resistance semiconductor. It is a light-controlled variable resistor that reacts to the intensity of the light; in other words, it exhibits photoconductivity. It is useful in light-sensitive detector circuits. Different components of photoresistor are shown in Figure 4.6.

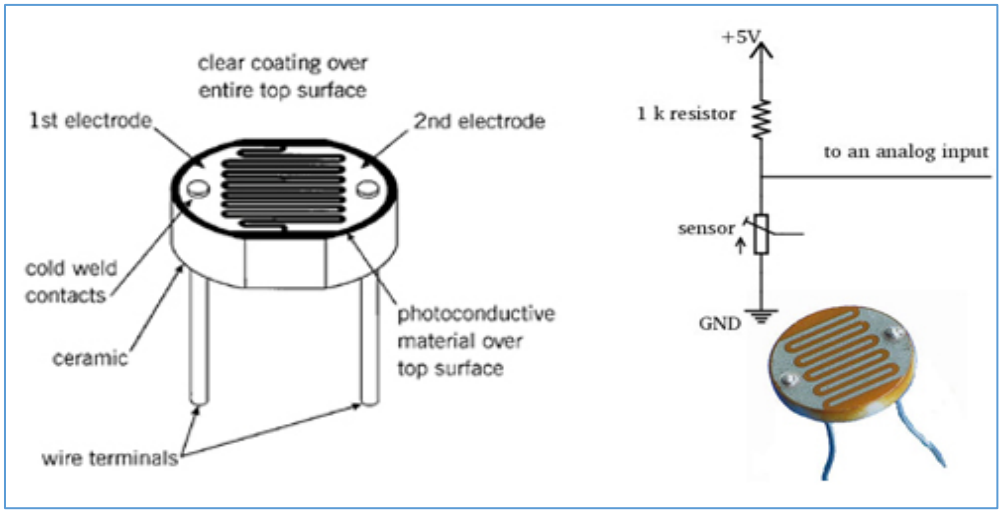

Figure 4.6 Photoresistor

(Source. Available [Online]: http://fabacademy.org/archives/2012/students/gispert.marc/18_pics/light_sensor.jpg)

Photoresistors are made of a high resistance semiconductor. Photoresistors possess high resistance in the dark while low resistance in the light. It possesses the resistance as high as several megaohms (MΩ) in the dark, while a photoresistor can have a resistance as low as few hundred ohms while in the light.

There are two types of the photoelectric devices categorized as intrinsic or extrinsic. Intrinsic semiconductors carry its charge carriers and are not an efficient semiconductor. In intrinsic devices, the photons must have enough energy to excite the electron across the entire bandgap, as the only available electrons are in the valence band. In extrinsic devices, they have impurities called as dopants. They have ground state energy closer to the conduction band. Hence, lower energy photons (i.e., lower frequencies and longer wavelengths) are enough to trigger the device, since the electrons do not have as far to jump. For example, if there are some phosphorous atoms(impurities) present in the sample of silicon, there will be extra electrons for the conduction.

4.7 Experimental Setup System 2 (DC Motor and Encoder open loop control)

Fig. 4.7 shows an experimental setup to study the time delay and data losses in the given global wireless internet network. The part A consists of (Arduino Uno + Wi-Fi Shield + DC Motor) and the part B consists of (Arduino Uno + Wi-Fi Shield + Encoder). DC Motor attached to the Wi-Fi shield can be operated (Turn On/Off) continuously for regular time interval using the program run on the Arduino through IDE software.

The Encoder connected to the other Wi-Fi shield will sense the variations in the velocity of the DC Motor and sends the reading to the attached server, and it gets recorded subject to its time constraints. The recorded data can be studied for the time delays and the link failures. The timing of each data recorded let us know the time delay in execution of the loop. If the time delay is greater than that of the sampling interval, there is a loss of the data, which we can find out from the experimentally recorded data.

The DC Motor attached to the Wi-Fi shield can be operated from anywhere in the world using port forward method. In this method, the IP address of the Arduino shield on the local network being forwarded to the global network, which further can be used to operate the DC Motor attached to the Arduino board globally.

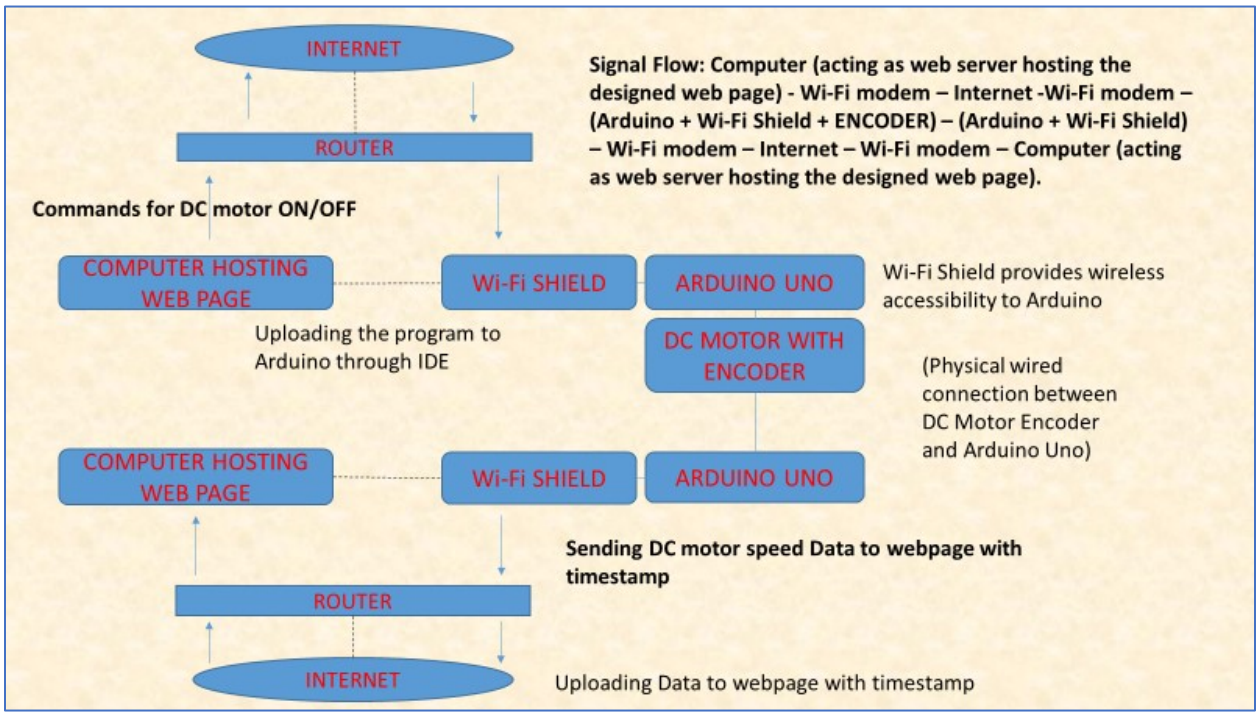

Figure 4.7 Experimental Setup System 2

4.8 L298N Dual H-Bridge Arduino Motor Driver

It is a low-cost motor driver board that can be used to drive two robot motors simultaneously. The Motor Driver is powerful enough to run motors from 5-35 Volts up to 2 Amperes per channel by using the popular L298N Dual H-Bridge Motor Driver Chip. The flexible digital input controls allow each motor to be fully independent with complete control over speed direction and braking action. L298N Dual H-Bridge Arduino Motor Driver is shown in Figure 4.8.

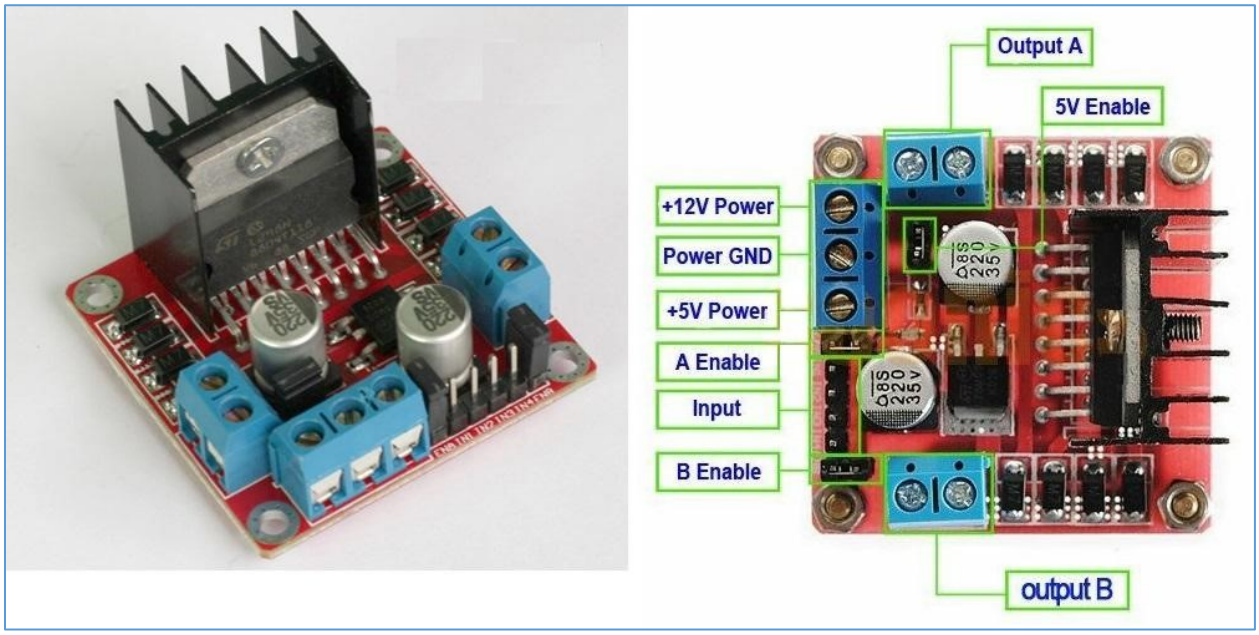

Figure 4.8 Dual H-Bridge Motor Driver

(Source: Available [online]: http://www.instructables.com/id/Control-DC-and-stepper-motors-with-L298N-Dual-Motor)

4.8.1 Features

- Popular L298N Dual H-Bridge Motor Driver chip

- Drivers motors from 5-35V at up to 2A per channel
- Provides 4 LEDs that reflect the state of the control logic
- Independent direction, speed and braking for each motor
- Screw terminals for easy connections to motors and power
- Includes a heavy-duty heat sink for maximum performance
- Easy to interface with most robot controllers
- Supports current sensing
- Handy power LED

4.8.2 Specifications

- Input voltage: 5-35V
- Output current: 2A per channel
- Control logic: standard 5V TTL
- Power consumption: 36ma for logic

Note: The 5V regulated power on pin 5 above is an output when the 5V_EN jumper is in place. Otherwise one must input 5V regulated power at pin 5 so that the circuit can operate properly. Do

not enable the onboard 5V regulator if you are supplying more than 16V to motors on pin 3 or the regulator will burn out.

4.8.3 How to use the L298N Dual H-Bridge Motor Driver?

The L298N Motor Driver board is the Dual H-Bridge type board, which can be used with the variety of robotic controllers. It has the potential to run motors from 5-35V at up to 2A peak. With the heavy-duty heat sink, it features a powerful L298N motor driver module. Other components of robot's systems such as an Arduino microcontroller can be powered by the 5V regulator provided on the driver board.

4.8.4 Usage

Follow the steps below to configure the motor controller board to work as a typical robot motor driver for use with two DC motors.

1. Attach your robot’s motors to the Motor A and Motor B screw terminals.
2. Connect the ENA and ENB to PWM capable digital outputs on your robot’s microcontroller.
3. Connect the IN1, 2, 3 and 4 pins to any digital outputs your robot’s microcontroller.
4. Apply 5-16V to the board by connecting positive (+) to the blue VMS screw terminal and ground (-) to the blue GND screw terminal.

4.8.5 Header pin assignments

Pin	Name	Description
1	ENA	Input to enable Motor A
2	IN1	Input to control Motor A
3	IN2	Input to control Motor A
4	ENB	Input to enable Motor B
5	IN3	Input to control Motor B
6	IN4	Input to control Motor B

4.8.6 Jumpers

Name	Description
5V_EN	Enable the onboard 5V regulator
U1	Enable Motor A input pin IN2 pull-up resistor (10K)
U2	Enable Motor A input pin IN2 pull-up resistor (10K)
U3	Enable Motor B input pin IN3 pull-up resistor (10K)

U3	Enable Motor B input pin IN4 pull-up resistor (10K)
CSA	Ties the Motor A current sense to ground
CSB	Ties the Motor B current sense to ground

4.8.7 Speed control

By connecting PWM (Pulse Width Modulation) outputs from the microcontroller to the ENA and ENB input pins on the motor driver board, the velocity of the motors can be regulated. ENA and ENB pin controls the respective motors connected to it. Power is output to the motor when these pins are HIGH. We can control the speed of the motor by using PWM, turning power ON and OFF very quickly. The higher the PWM duty cycle is, the faster the motor will turn. It is recommended to use a PWM duty cycle of 90% or less.

4.8.8 Direction control

Using the IN1, IN2, IN3 and IN4 input pins on the motor driver board, the direction of the motor is controlled.

IN1=HIGH and IN2=LOW-Motor A runs in a forward direction.

IN1=LOW and IN2=HIGH-Motor A runs in a reversed direction.

4.8.9 Stopping

To remove power from motors

ENA=LOW-Remove power from Motor A.

ENB=LOW- Remove power from Motor B.

To perform a quick braking action

ENA=LOW, IN1=LOW and IN2=LOW- Quick braking action for Motor A

ENB=LOW, IN3=LOW and IN4=LOW-Quick braking action for Motor B

The motors will come to an instant stop.

4.8.10 Motor Driver Truth Tables

Motor A truth table

ENA	IN1	IN2	Description
0	N/A	N/A	Motor A is off
1	0	0	Motor A is stopped (brakes)
1	0	1	Motor A is on and turning backwards

1	1	0	Motor A is on and turning forwards
1	1	1	Motor A is stopped (brakes

Motor B truth table

ENA	**IN3**	**IN4**	**Description**
0	N/A	N/A	Motor B is off
1	0	0	Motor B is stopped (brakes)
1	0	1	Motor B is on and turning backwards
1	1	0	Motor B is on and turning forwards
1	1	1	Motor B is stopped (brakes

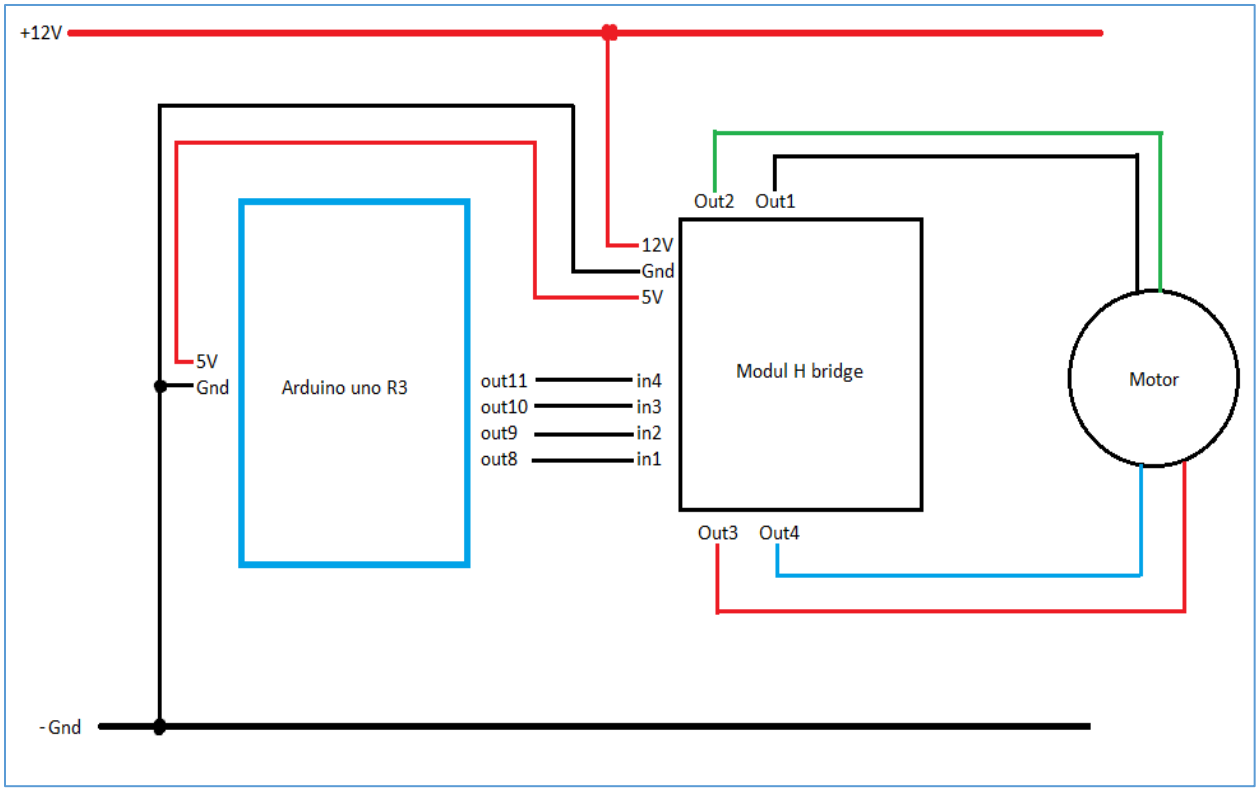

Figure 4.9 Dual H-Bridge Motor Driver Circuitry

(Source. Available. [online] http://forum.arduino.cc/index.php?topic=140274.0)

4.9 DC Motor

It is a type of an electrical rotary machine which converts direct current electrical energy into mechanical energy. Its types depend on the forces produced by the magnetic fields of the machine. Almost all types of DC motors have some internal mechanism, either electronic or electromechanical, to regularly switch the direction of current flow in part of the motor. Either changing the strength of the current in its field windings or by a variable supply voltage, a DC motor's speed can be controlled over a wide range. General components of DC motor are shown in Figure 4.10.

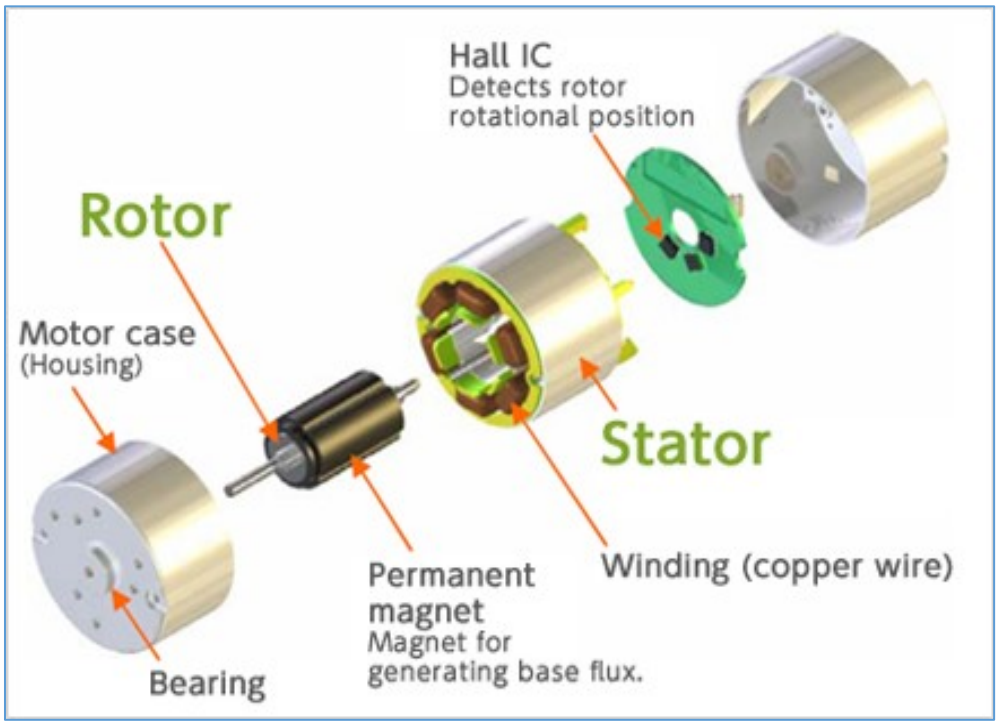

Figure 4.10 DC Motor

(Source. Available. [online] http://img.motors-biz.com/upload/tmp/1/210e0f97d456a786a436421d5a6891ed.jpg)

4.10 Rotary Encoder

In another term, rotary encoder also known as the shaft encoder, which is an electro-mechanical device. The rotary encoder converts the motion or angular position of a shaft or axle to an analog or digital code.

There are two main types of rotary encoders, incremental (relative) and absolute. The current position of the shaft is indicated by the output of absolute encoders by making them angle transducers, the information about the motion of the shaft is indicated by the output of the incremental encoders, which is elsewhere further can be processed into some other kind of information such as position, speed and distance.

4.10.1 Absolute and incremental encoders

When the power is removed from the system absolute encoder maintains position information. Immediately on applying power the position of the encoder is available. To maintain position accuracy, the system does not need to return to a calibration point. The relationship between the physical position and the encoder value of the controlled machinery is set at the assembly. An "incremental" encoder correctly registers changes in position but does not power up with a rigid relationship among encoder position and physical state.

To initialize the position measurement, the devices controlled by incremental encoders may have to "go home" to a fixed reference point. Fractional rotation is measured by the high-resolution wheel while the number of whole revolutions of the shaft is measured by the lower-resolution geared code wheels.

An absolute encoder contains multiple code rings with the various binary weightings, which by representing the absolute position of the encoder within one revolution provides a data word. This type of encoder is usually attributed to as a parallel absolute encoder. Figure 4.11 describes the different components of rotary encoder.

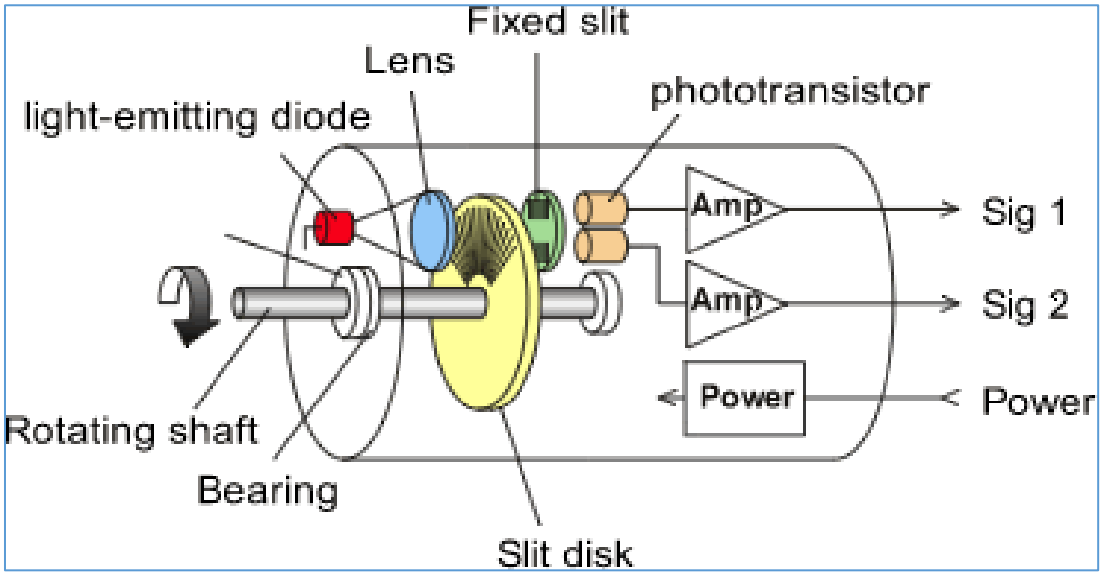

Figure 4.11 Rotary Encoder

(Source. Available [online].
https://www.onosokki.co.jp/English/hp_e/products/category/images/rp_principle_e.gif)

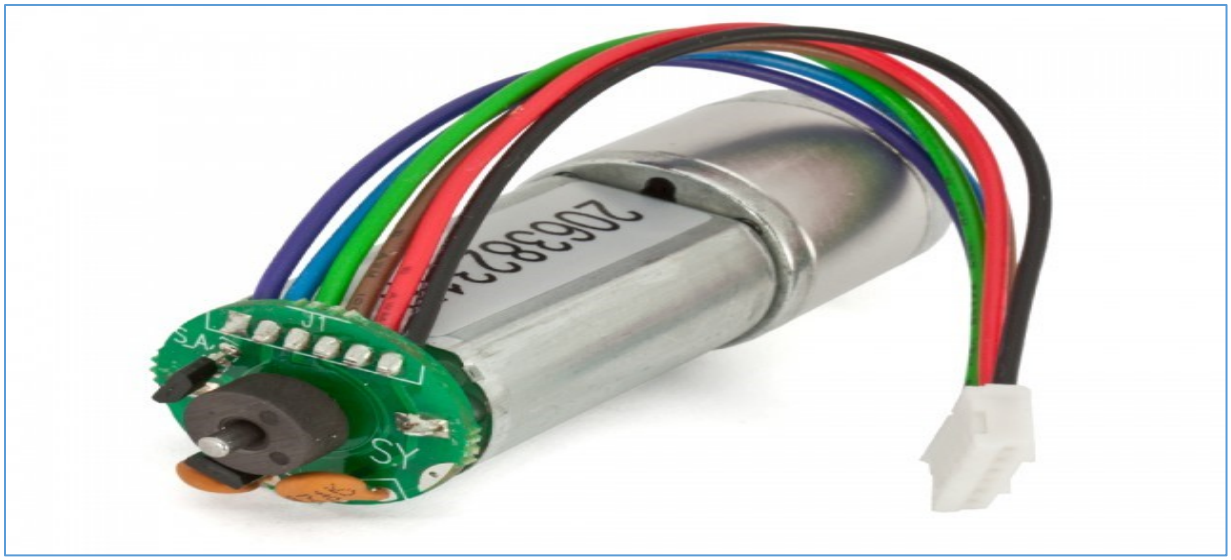

Figure 4.12 DC Motor Rotary Encoder.

(Source. Available [online] https://encrypted-tbn0.gstatic.com/images?)

4.11 Recording of Data on Web page

Experimentally generated data gets recorded on the designed web page, which is linked to the experimental setup.

Figure 4.13 Web page Interface

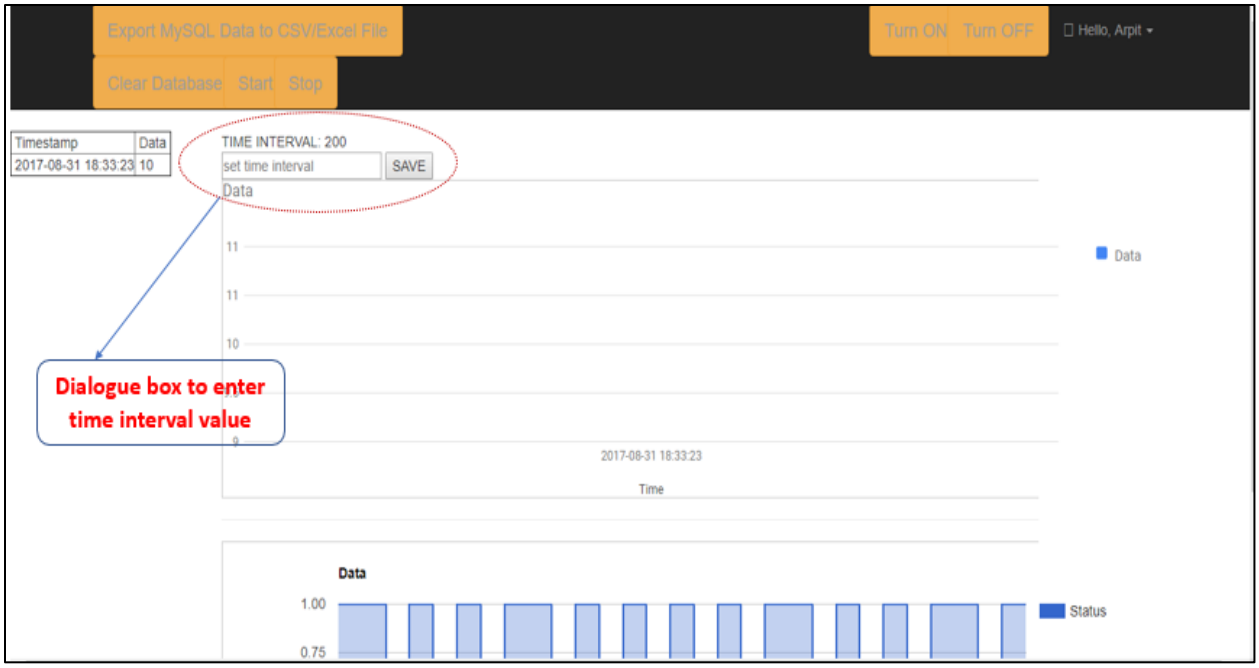

Figure 4.14 Time interval input dialogue box on Web page Interface

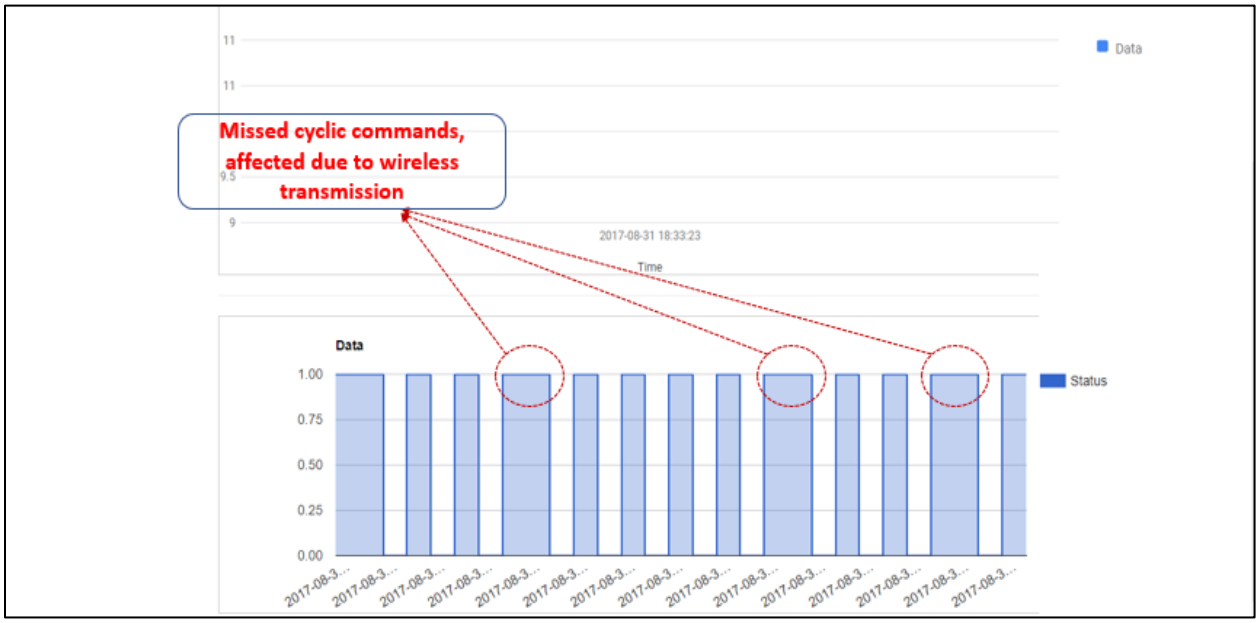

Figure 4.15 Missed cyclic commands showing on Web page Interface

Chapter 5

Software Design and Implementation

5.1 Arduino IDE Introduction

5.1.1 Structure

The basic structure of the Arduino programming language is simple and runs in at least two parts.

These two required parts, or functions, enclose block of statements.

```
void setup ( )
{
  statements;
}
void loop ( )
{
  statements;
}
```

setup ( ) and loop ( ) both functions are required for the program to work

where,

setup ( ) - is the preparation

loop ( ) – is the execution

At the very beginning of the program the setup function should follow the declaration of any variable It is used to set pinMode or initialize serial communication, is run only once, and is the first function to run in the program.

The next step is followed by the next function, which includes the code to be executed continuously-reading inputs triggering outputs. This specific function does the bulk of the work, as it is the core of all Arduino programs.

setup ( )

Once the program starts the setup ( ) function is called. It is used to initialize the pin modes, or begin serial. Even if there are no statements to run it must be included in program.

```
void setup ( )
{
  pinMode (pin, OUTPUT) ;          //sets the pin as output.
}
```

loop ( )

As the name suggests the loop ( ) function loops continuously, it is called after the setup( ) function, allowing the program to change, respond and control the Arduino board.

```
void loop ( )
{
   digitalWrite (pin, HIGH);      // turns 'pin' on
    delay (1000);                  // pauses for one second
    digitalWrite (pin, LOW);    // turns 'pin' off
```

```
  delay (1000)                    // pauses for one second
}
```

5.2 functions

It has a block of code with a name and consists of block of statements being executed when the function is called. To perform repetitive tasks and reduce clutter in program, custom functions can be written. By first declaring the function type, functions are declared, such as ‘int’ for an integer type function. The function type would be void if no value is to be returned.

```
type functionName (parameters)
{
  statements;
}
int delayVal ( )
{
  int v;                    //create temporary variable ‘v’
  v = analogRead (pot)   // read potentiometer value
  v /= 4;                   //converts 0-1023 to 0-255
  return v;                 //return final value
}
```

To make the code more readable, variables should be given descriptive names. It helps the programmer or anyone else to understand what the variables represent by reading the code.

5.3 variables

For later use by the program variables are used to store and name numerical value. As opposed to constants whose value never changes, variables are continuously changed numbers. For the value needing to be stored variables needs to be declared and optionally assigned.

```
int   inputVariable = 0;                  //declares a variable and assigns value of 0
inputVariable = analogRead(2)       //set variable to value of analog pin 2
```

We can justify if certain condition meets are not, once variables are assigned by analyzing the variables we declare.

```
if  (inputVariable < 100)  //tests variable if less than 100
{
  inputVariable = 100;    //if true assigns value of 100
}
delay (inputVariable);    //uses variable as delay
```

To make the code more readable, variables should be given descriptive names. Variable names like TiltSensor or pushButton help the programmer to interpret what the variables represents in the given code.

5.3.1 Variable declaration

Before using the variables in the program, all variables must be declared. Variable declaration means defining its value type, as in float, int, long etc., setting a specified name and optionally assigning an initial value. Variable value can be changed at any time in the program using arithmetic and various assignments.

5.3.2 Variable scope

Variable scope means the ability of the program to make use of variables in certain parts of programs. It can be declared before void setup at the beginning of the program.

A variable which can be seen and used by every function and statement in a program is known as a global variable. This variable is declared before the setup ( ) function at the beginning of the program.

A variable which is defined as the part of a for loop inside a function is known as the local variable. It can only be used and visible in the inside the function in which it was declared.

```
int value;                    // 'value' is visible to any function
void setup ( )
{
                              // no setup needed
}
void loop ( )
{
   for  ( int i=20; i<20);    // is only visible inside the for-loop
    {
       i++;
     }
     float f;                 //'f' is only visible inside loop.
}
```

5.4 Part A (System 1)

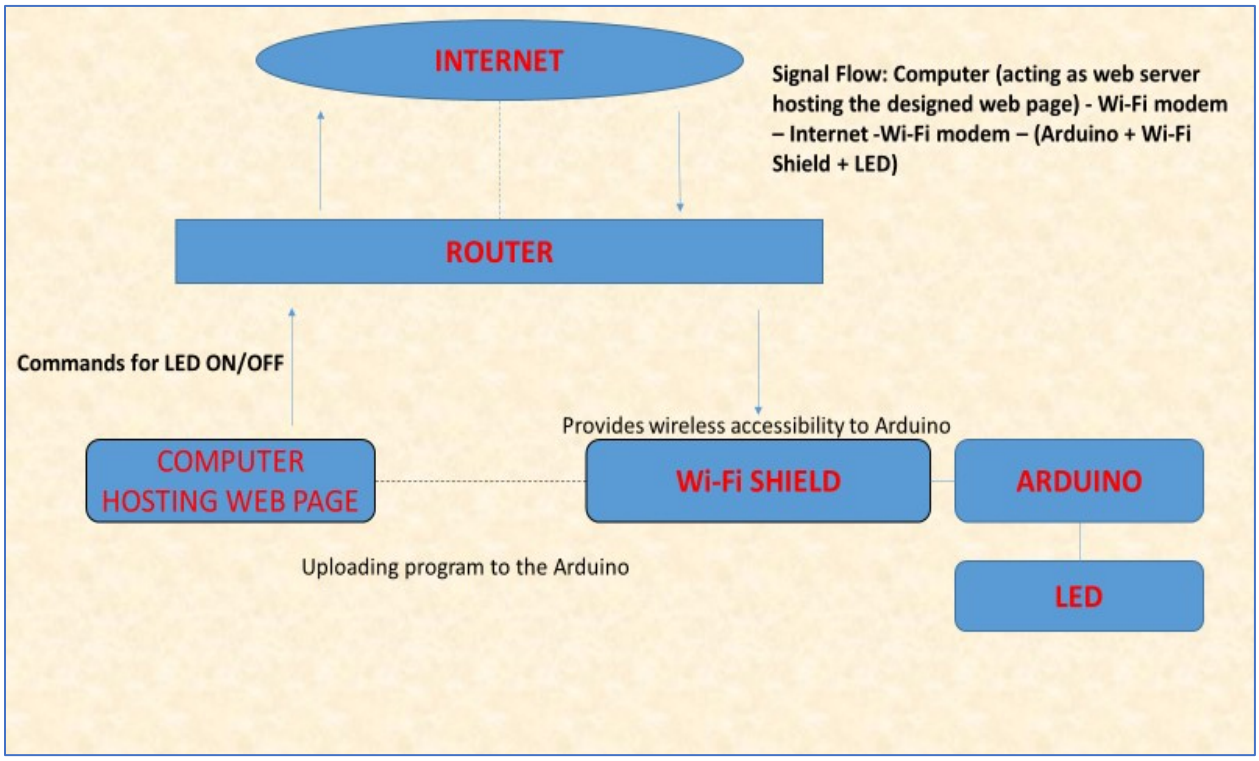

Figure 5.1 Part A (System 1)

This consists of (Arduino Uno + Arduino Wi-Fi Shield + LED), Arduino Wi-Fi Shield being used to provide the wireless access to the circuit, shown in Figure 5.1. The codes running will provide the IP address of the Arduino board present in the circuit. The given IP address is used to connect to the specific Arduino board in the circuit. Using this IP address commands are given to the LED, which is attached to the Arduino board having the provided IP address.

Part A of system 1 consists of (Arduino Uno + Arduino Wi-Fi Shield 101 + LED) connected in the circuit to each other. Arduino Wi-Fi shield is used to connect the Arduino circuit to the wireless

network. LED connected to the Arduino shield is programmed to blink continuously in time interval being set by the program code, which can be changed for performing experiments at different time intervals. Arduino board is connected to the computer system from which it gets power. Arduino IDE software module installed on the system is used to interact with the Arduino board and to upload the required code on the board to perform experiment.

5.4.1 Connecting Wi-Fi Shield to the wireless network.

This shows how to connect the Wi-Fi shield to the encrypted network with the Arduino Wi-Fi shield 101 and the Arduino Uno. The Arduino serial monitor will provide the information about the connection.

- **Hardware Required**

  - Arduino Wi-Fi Shield 101
  - Arduino Uno board

Digital pin 7-used as a handshake pin between the Wi-Fi Shield 101 and the Arduino board, and this pin should not be used to assign any task in the program.

We should have access to the networks SSID and password to connect the shield to the required network.

For networks using WPA/WPA2 Personal encryption, you need the SSID and password. The shield will not get connected to networks using WPA2 Enterprise encryption.

5.5 Part B (System1)

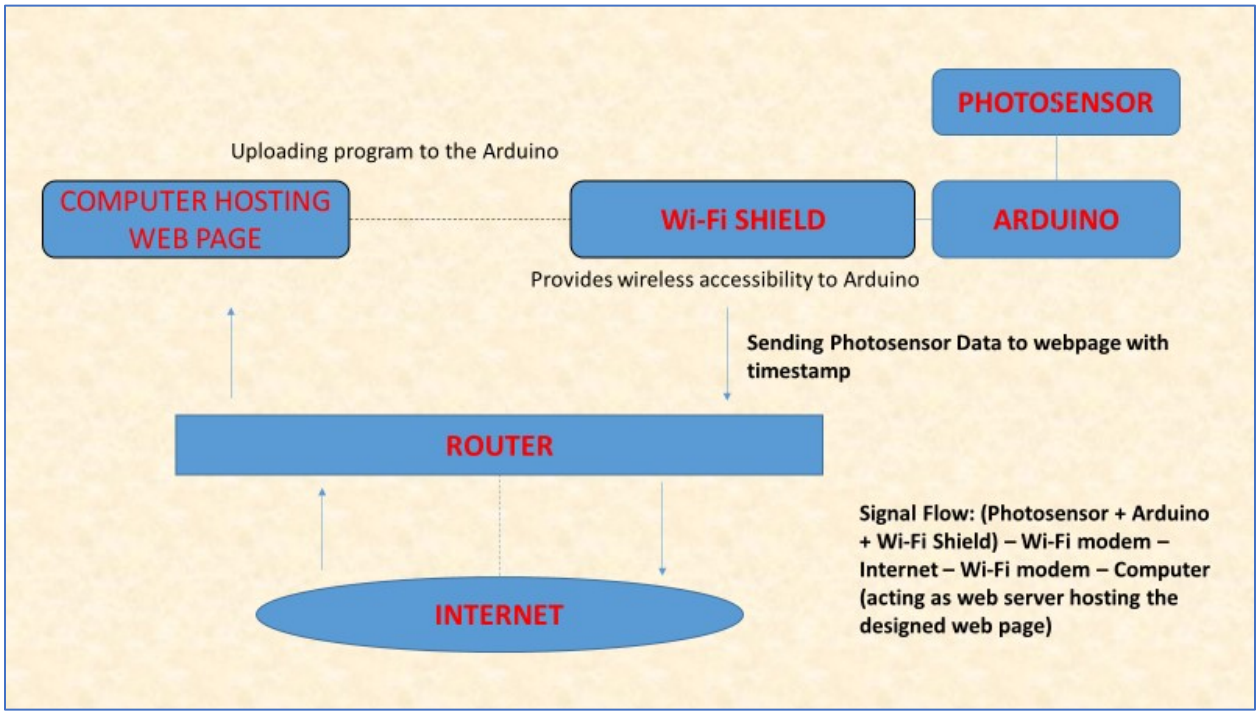

Figure 5.2 Part B (System 1)

Part B of system 1 consists of (Arduino Uno + Arduino Wi-Fi shield 101 + Photocell) connected in the circuit, as shown in Figure 5.2. The Wi-Fi shield connects the circuit to the wireless network same as the part 1. A photocell senses the blinking of LED and responds to it. The photocell attached to the circuit sends the data, acts as the Light Sensor, which generates an output signal signifying the intensity of light produced by the LED in the Part A. Part A and Part B of the system

are enclosed in the box to produce the better results. The photocell readings get uploaded and recorded in the server connected to the system, which can be used for further analysis to study the link failures and internet latencies.

5.6 System 1 (Part A + Part B)

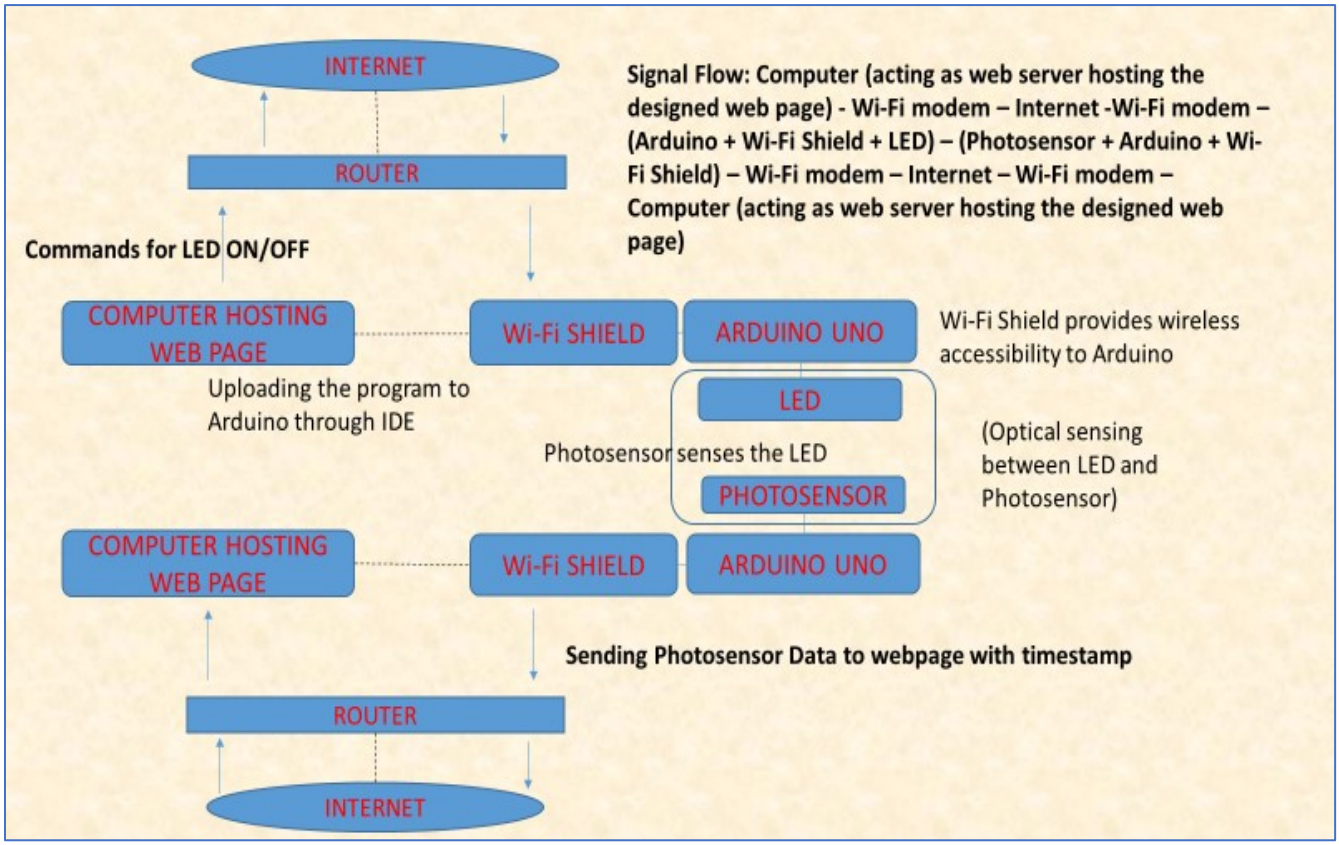

Figure 5.3 System 1

5.7 Part A (System 2)

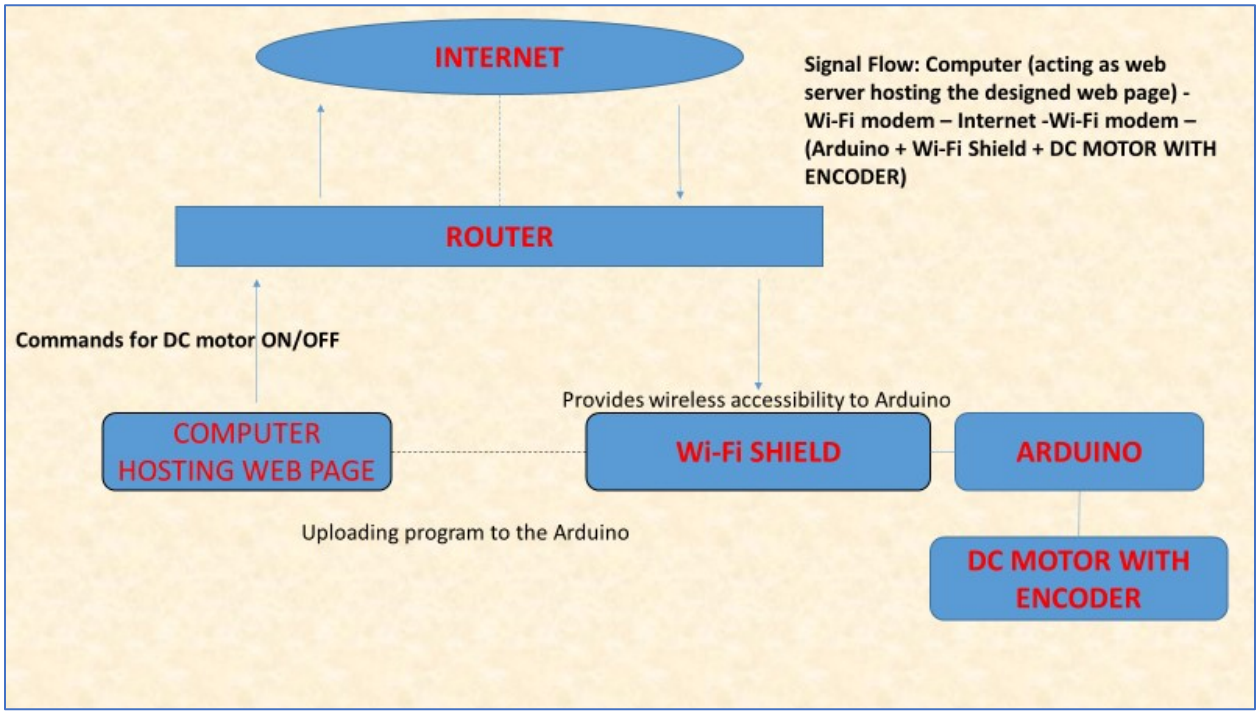

Figure 5.4 Part A (System 2)

Part A of system 2 consists of (Arduino Uno + Arduino Wi-Fi Shield 101 + DC Motor) connected in the circuit to each other, as shown in the Figure 5.4. Arduino Wi-Fi shield is used to connect the Arduino circuit to the wireless network. DC Motor attached to the Arduino shield is programmed to turn on/off continuously in time interval being set by the program code, which can be changed for performing experiments at different time intervals. Arduino board is connected to the computer system from which it gets power. Arduino IDE software module installed on the system is used to interact with the Arduino board and to upload the required code on the board to perform experiment.

5.8 Part B (System 2)

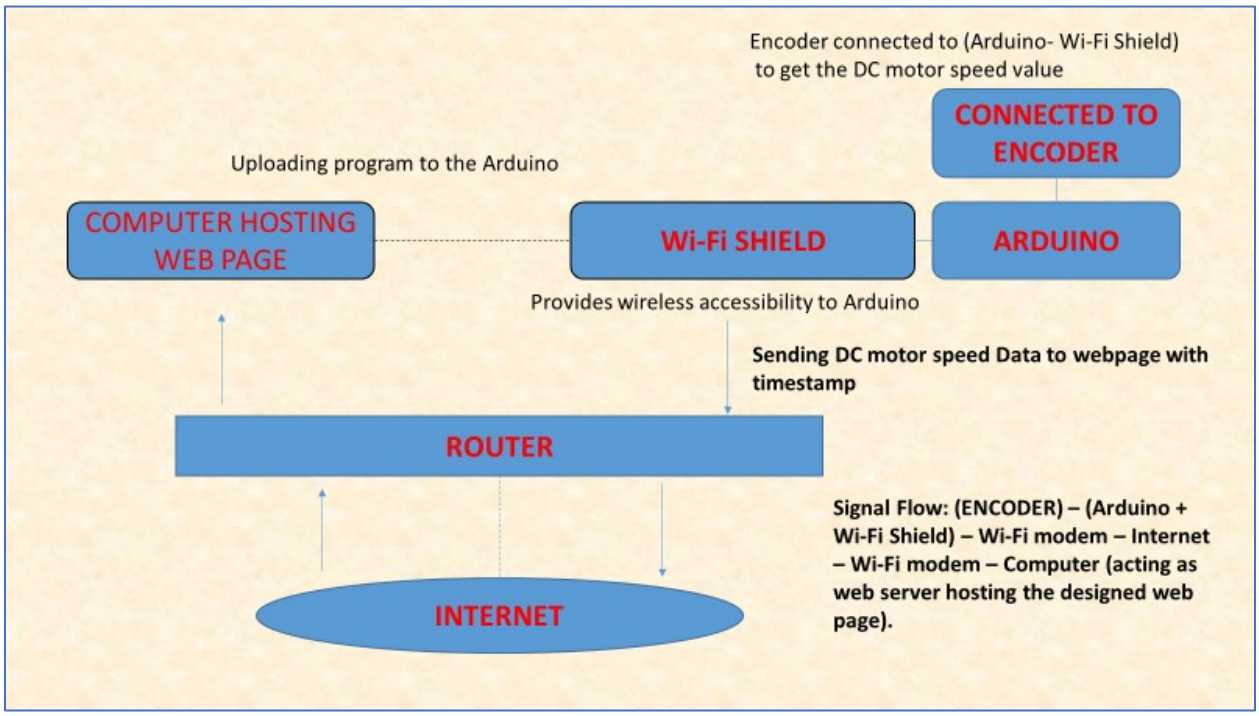

Figure 5.5 Part B (System 2)

Part B of system 2 consists of (Arduino Uno + Arduino Wi-Fi shield 101 + Encoder) connected in the circuit. The Wi-Fi shield connects the circuit to the wireless network same as the part 1, shown in Figure 5.5. An encoder senses the variations in the speed of the DC Motor. The encoder sends the velocity data to the server and it gets recorded with the time constrained to it. Part A and Part B of the system are connected to each other to produce the better results. The encoder readings get uploaded and recorded in the server connected to the system, which can be used for further analysis to study the link failures and internet latencies.

5.9 System 2 (Part A + Part B)

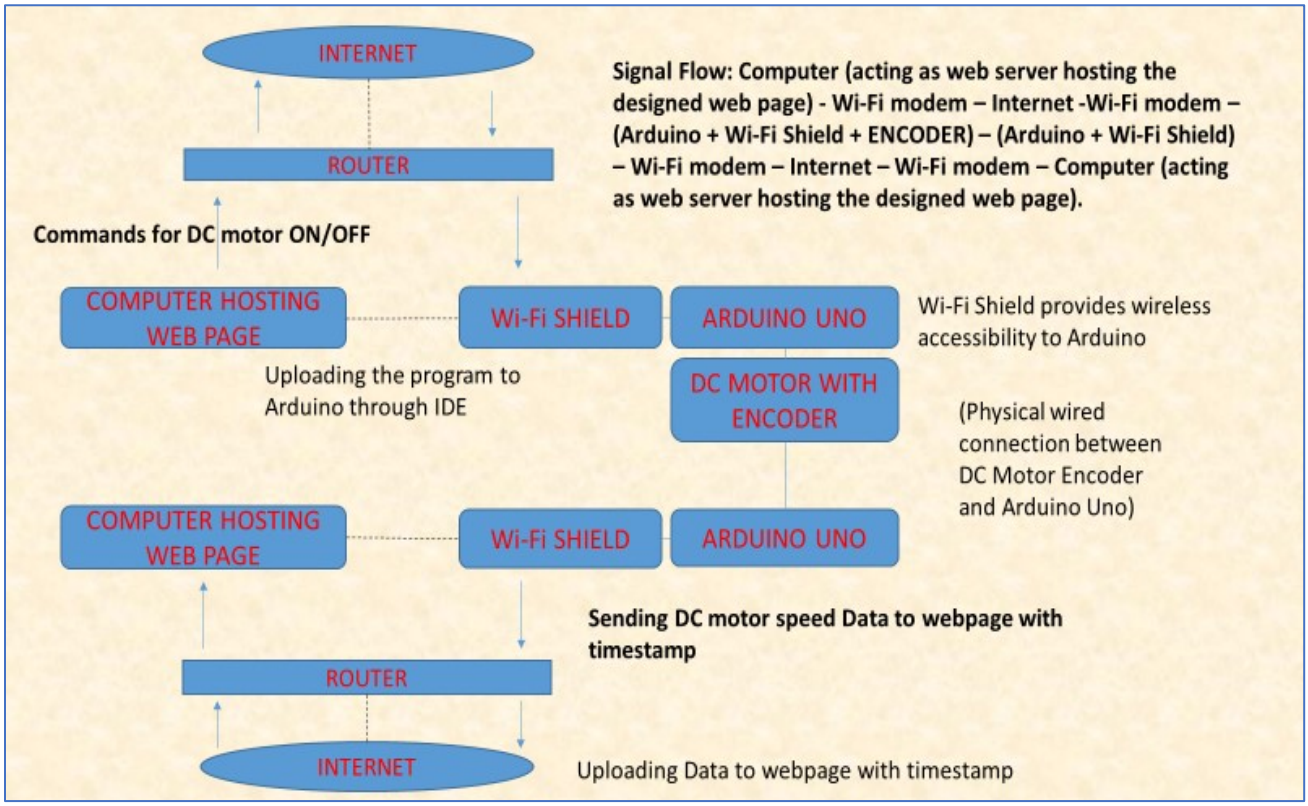

Figure 5.6 System 2

Chapter 6

Results and Discussions

6.1 Experimental data and graphical representation for results of system 1

The increasing implementation of Internet of Things has made it crucial to thoroughly analyze the effects of time delay and data losses for the given wireless environment to study the link failures and find out cycle time in case of link failures. This computing time delay is varied from system time delay; it is a random delay resulting from the execution of control programs.

Consequences of computing time delay are categorized as the delay and loss problems which can be analyzed for cycle time to link failure. The experimental setup described in this thesis concentrates on the development of the practical approach to study the closed loop link failure problems and eventually find its randomness characteristics and cycle time to link failure for executing it for different time intervals. This study will help in determining the suitable time interval or frequency for performing the closed loop control accounting for data losses and link failures.

Fig. 6.1, shows the recorded data from examining the experiment. Time constraint is associated with each registered data which ultimately helps to find the delay and data losses which can be

used in determining the suitable time interval for performing the closed loop control with reduced effects of failures.

In the graphs, the x-axis represents the time of the data recorded while the y-axis denotes the intensity of the LED light. The flat lines on the x-axis show the data losses as there is no data recorded at that time constraint.

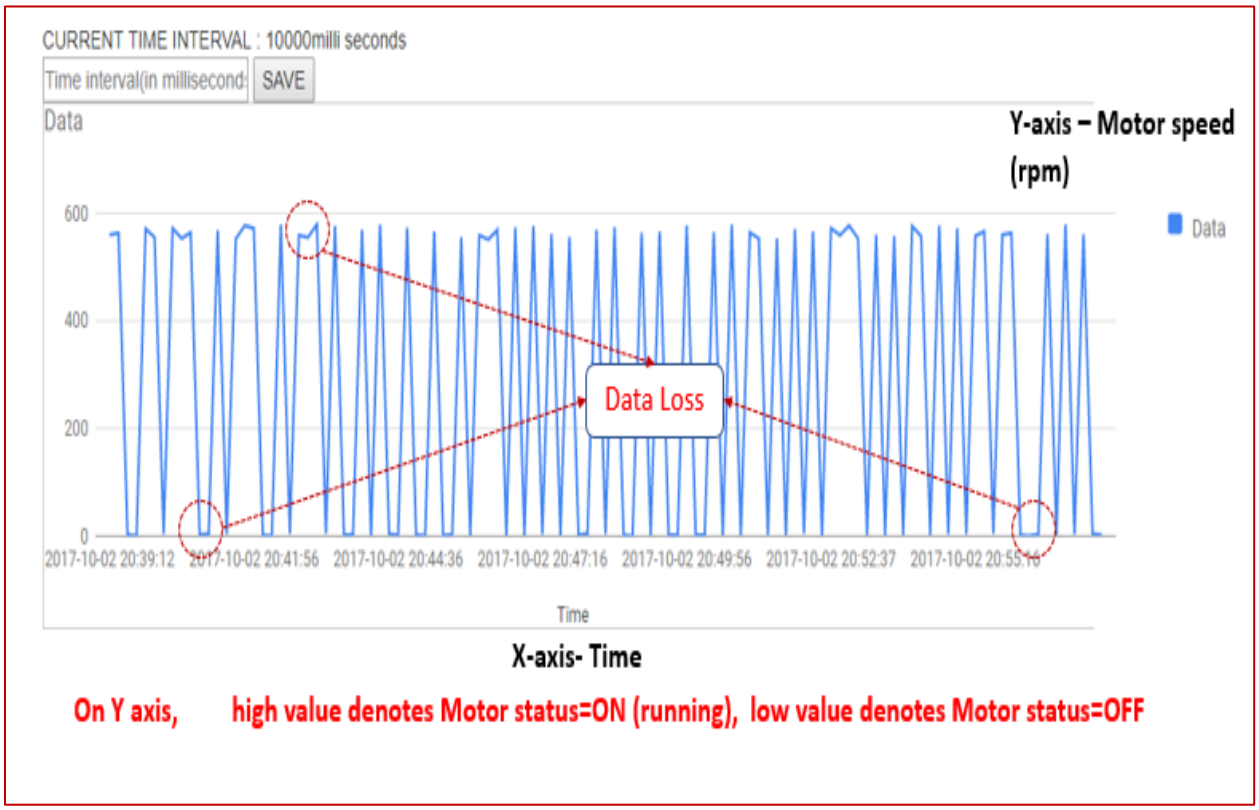

Figure 6.1 Experimental results with time constraint

Each experimentally recorded data in the Fig. 6.1 has its time constraint. The collected data for the sampling interval shows the randomness throughput the period. From the figure, we can observe that sometimes the time constraint of two continuous data is greater than the fixed sampling interval which shows the time delay, while for certain consecutive readings the time difference is in multiples of sampling intervals which shows us the loss of data.

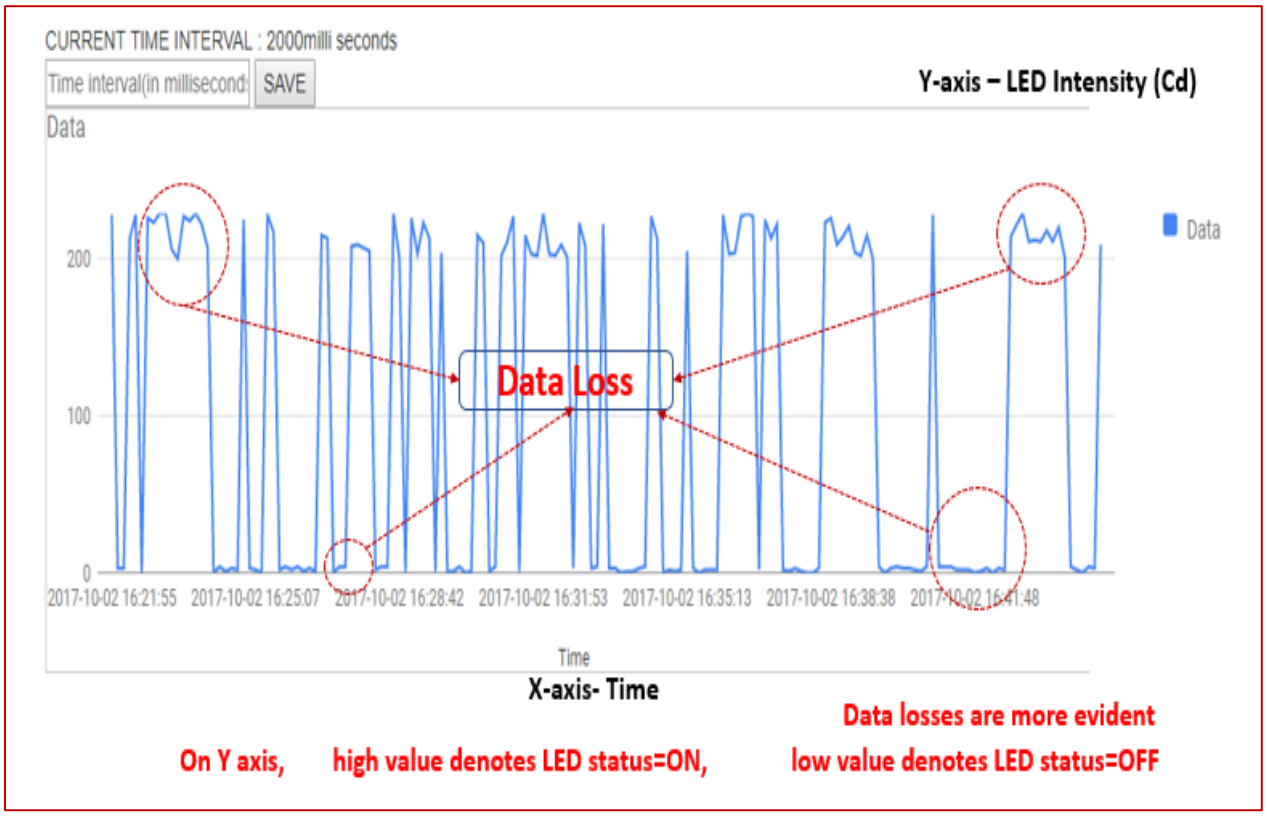

Figure 6.2 Experimental graph results for 2 seconds' time interval

Table 6.1: Data for 2 seconds' time interval

Timestamp	Data
10/2/2017 16:21	228
10/2/2017 16:22	3
10/2/2017 16:22	3
10/2/2017 16:22	213
10/2/2017 16:22	228
10/2/2017 16:22	0
10/2/2017 16:22	226
10/2/2017 16:22	223
10/2/2017 16:22	229
10/2/2017 16:23	229
10/2/2017 16:23	206
10/2/2017 16:23	200
10/2/2017 16:23	227
10/2/2017 16:23	224
10/2/2017 16:23	229
10/2/2017 16:23	222
10/2/2017 16:24	207
10/2/2017 16:24	0
10/2/2017 16:24	4
10/2/2017 16:24	1
10/2/2017 16:24	3
10/2/2017 16:24	2
10/2/2017 16:24	225
10/2/2017 16:24	3
10/2/2017 16:25	2
10/2/2017 16:25	0
10/2/2017 16:25	228
10/2/2017 16:25	217

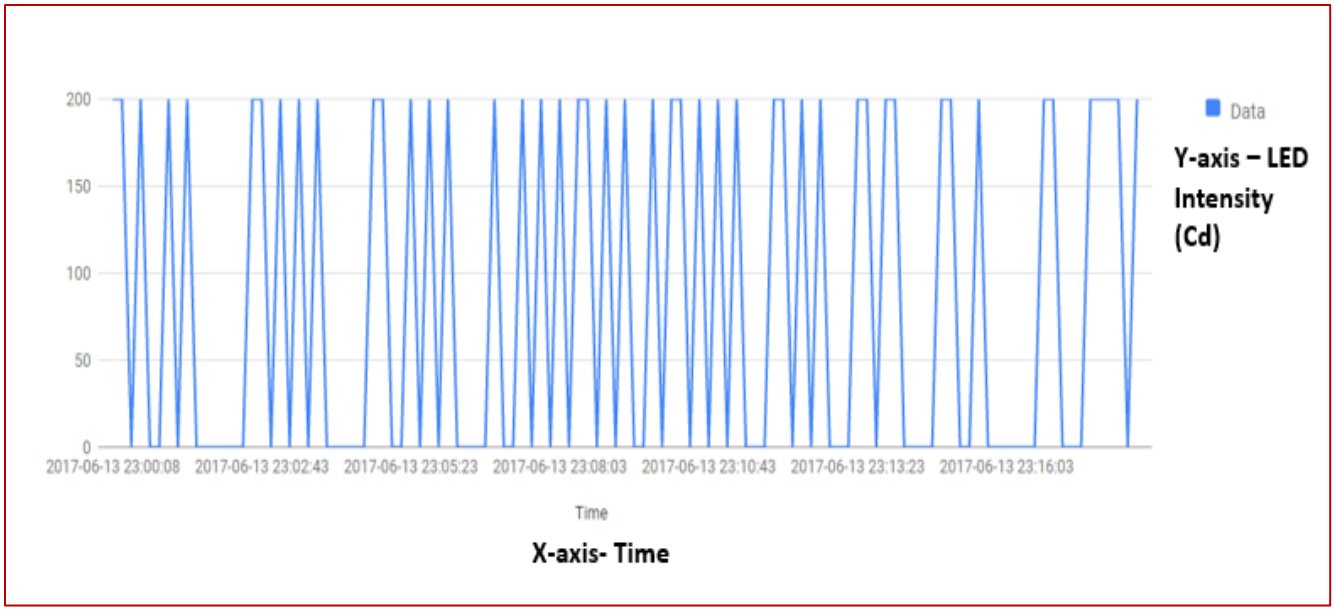

Figure 6.3 Experimental graph results for 4 seconds' time interval

Table 6.2: Data for 4 seconds' time interval

Timestamp	Data
6/13/2017 22:24	200
6/13/2017 22:25	200
6/13/2017 22:25	0
6/13/2017 22:25	0
6/13/2017 22:25	0
6/13/2017 22:25	0
6/13/2017 22:25	0
6/13/2017 22:25	0
6/13/2017 22:25	200
6/13/2017 22:26	200
6/13/2017 22:26	200
6/13/2017 22:26	0
6/13/2017 22:26	0
6/13/2017 22:26	0
6/13/2017 22:26	0

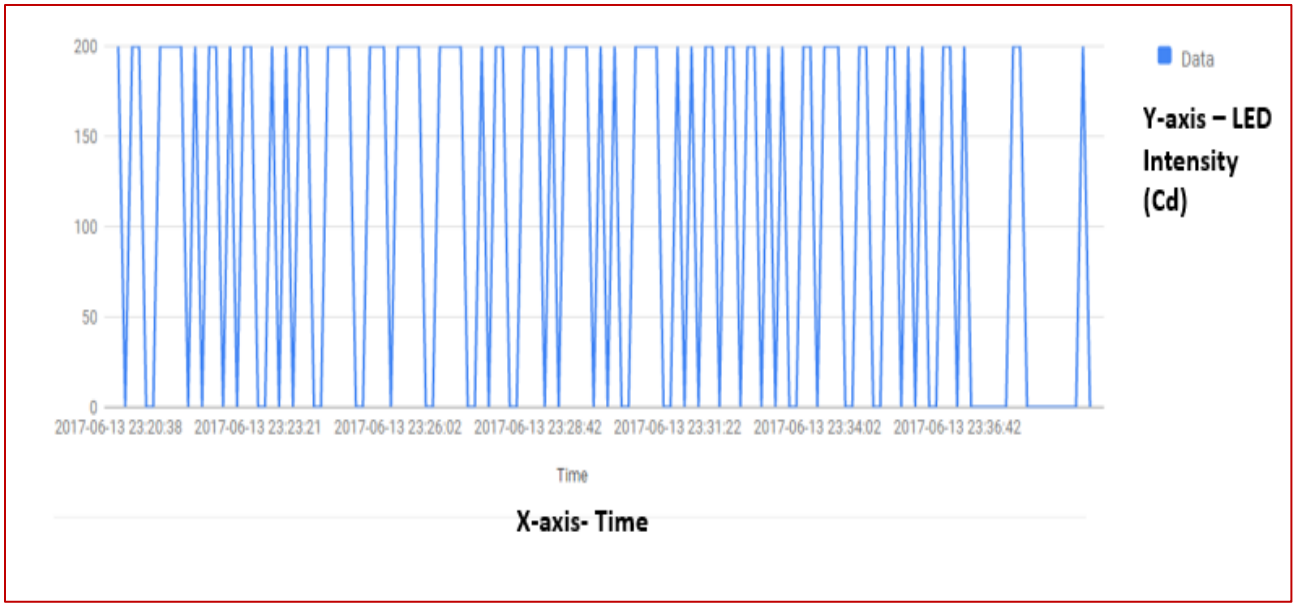

Figure 6.4 Experimental graph results for 5 seconds' time interval

Table 6.3: Data for 5 seconds' time interval

Timestamp	Data
6/13/2017 23:00	200
6/13/2017 23:00	200
6/13/2017 23:00	0
6/13/2017 23:00	200
6/13/2017 23:00	0
6/13/2017 23:00	0
6/13/2017 23:01	200
6/13/2017 23:01	0
6/13/2017 23:01	200
6/13/2017 23:01	0
6/13/2017 23:01	0
6/13/2017 23:01	0
6/13/2017 23:02	0
6/13/2017 23:02	0
6/13/2017 23:02	0
6/13/2017 23:02	200
6/13/2017 23:02	200
6/13/2017 23:02	0
6/13/2017 23:03	200

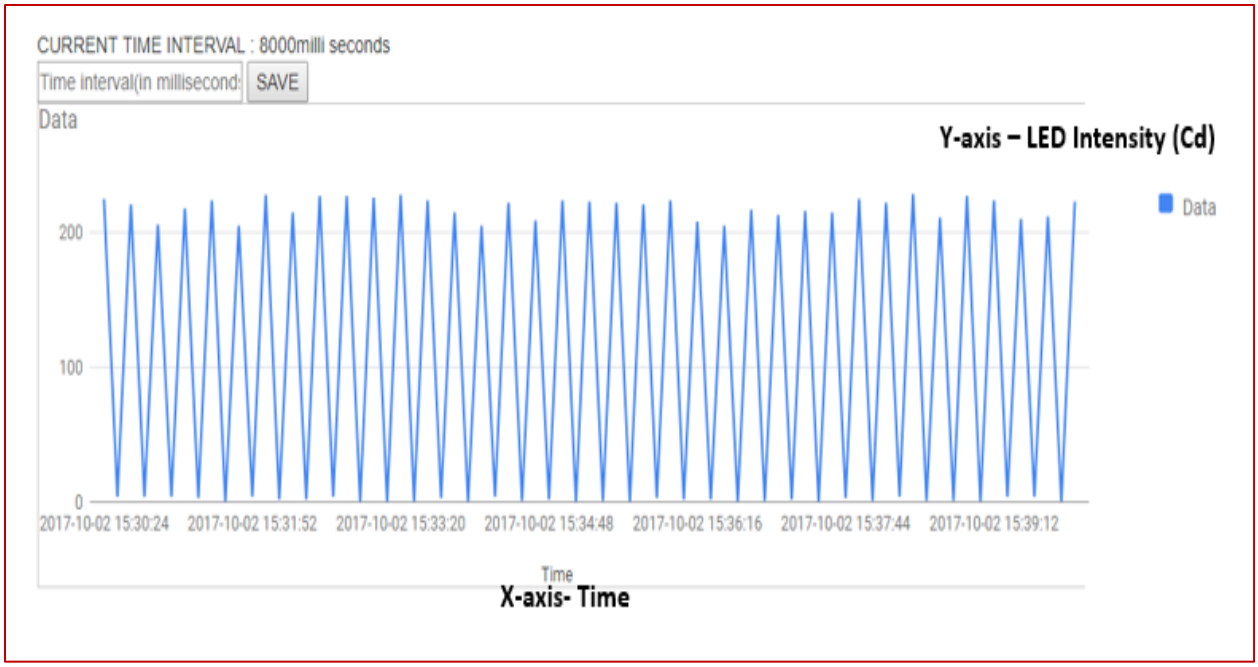

Figure 6.5 Experimental graph results for 8 seconds' time interval

Table 6.4: Data for 8 seconds' time interval

Timestamp	Data
10/2/2017 15:30	225
10/2/2017 15:30	4
10/2/2017 15:30	221
10/2/2017 15:30	4
10/2/2017 15:30	206
10/2/2017 15:31	4
10/2/2017 15:31	218
10/2/2017 15:31	3
10/2/2017 15:31	224
10/2/2017 15:31	0
10/2/2017 15:31	205
10/2/2017 15:31	4
10/2/2017 15:32	228
10/2/2017 15:32	2
10/2/2017 15:32	215
10/2/2017 15:32	2
10/2/2017 15:32	227
10/2/2017 15:32	4
10/2/2017 15:32	227

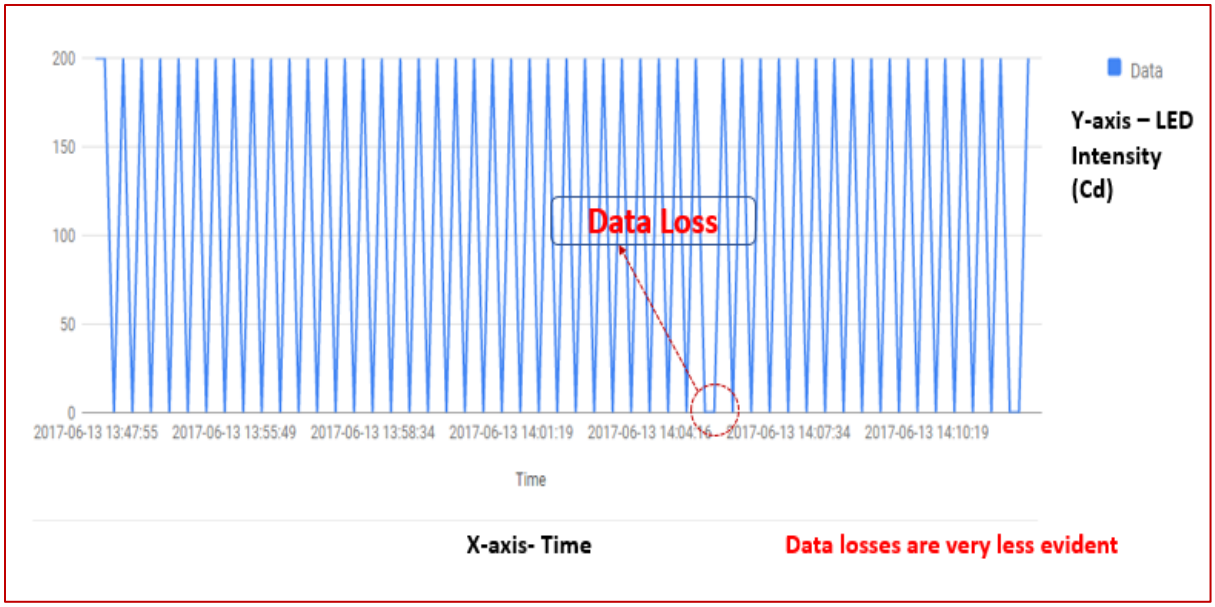

Figure 6.6 Experimental graph results for 10 seconds' time interval

Table 6.5: Data for 10 seconds' time interval

Timestamp	Data
6/13/2017 23:40	200
6/13/2017 23:40	0
6/13/2017 23:40	200
6/13/2017 23:40	0
6/13/2017 23:40	200
6/13/2017 23:40	200
6/13/2017 23:41	0
6/13/2017 23:41	200
6/13/2017 23:41	0
6/13/2017 23:41	200
6/13/2017 23:41	200
6/13/2017 23:42	0
6/13/2017 23:42	0
6/13/2017 23:42	200
6/13/2017 23:42	200
6/13/2017 23:42	200
6/13/2017 23:42	200
6/13/2017 23:43	200
6/13/2017 23:43	200
6/13/2017 23:43	200
6/13/2017 23:43	0
6/13/2017 23:43	200

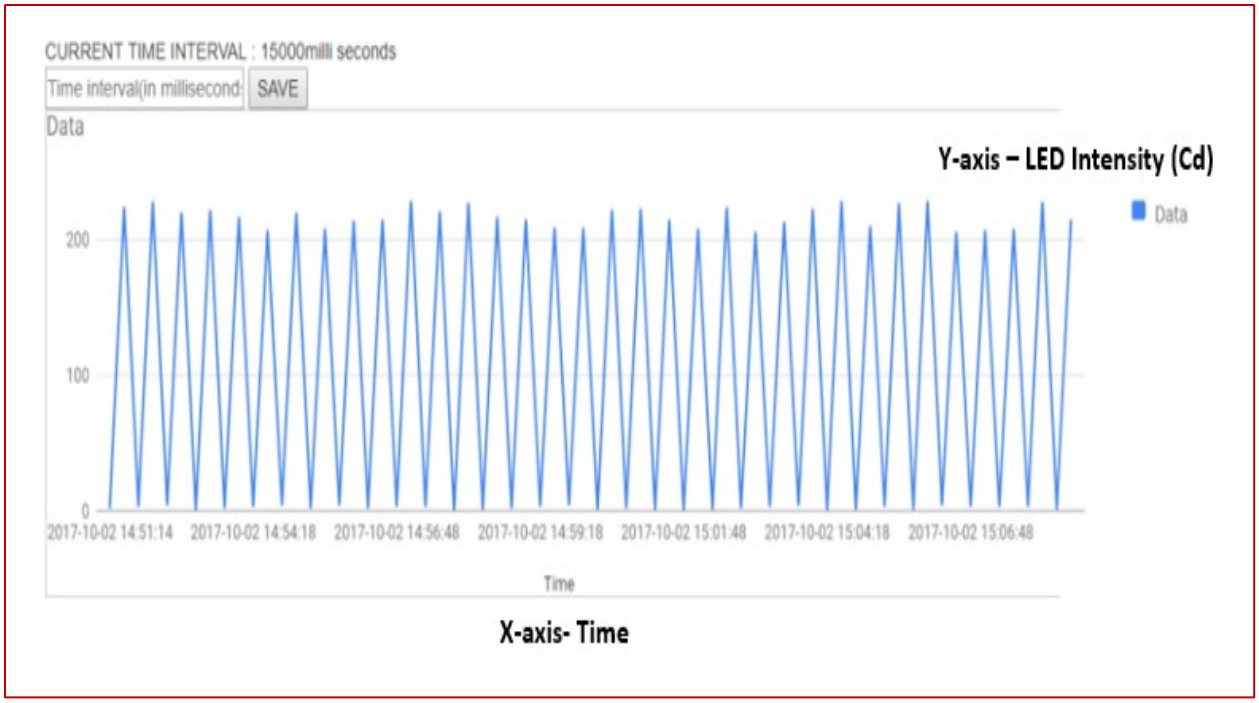

Figure 6.7 Experimental graph results for 15 seconds' time interval

Fig. 6.2, Fig. 6.4 and Fig. 6.6 shows the experimentally recorded data for the 2, 5 and 10 seconds' time interval of LED blinking respectively, each recorded data has its own time constraint. As we observe the figures thoroughly the data delay and data loss are more evident in the case of 2 seconds' time interval as compared to the other two graphs. As we compare the results of 5 seconds' and 10 seconds' time interval data delay and loss are more apparent in 5 seconds' graph than the graph represents the 10 seconds' time interval. This shows that as the time interval for closed loop control system increases there are lesser chances of data delay and data losses resulting in fewer failures of the system.

The experimental results show that, as the frequency of the closed loop control system increases, the more the chances of the failures because of the data delay and data loss as compared to the closed loop cycles of higher time intervals, i.e., lower frequency.

6.2 Experimental data and graphical representation for results of system 2

The following graph shows the velocity of DC motor recorded in regular intervals. The x-axis represents the time constraint while the y-axis denotes the velocity. The flat lines on x-axis show the data losses as there are no data recorded at that time, thus the flat lines.

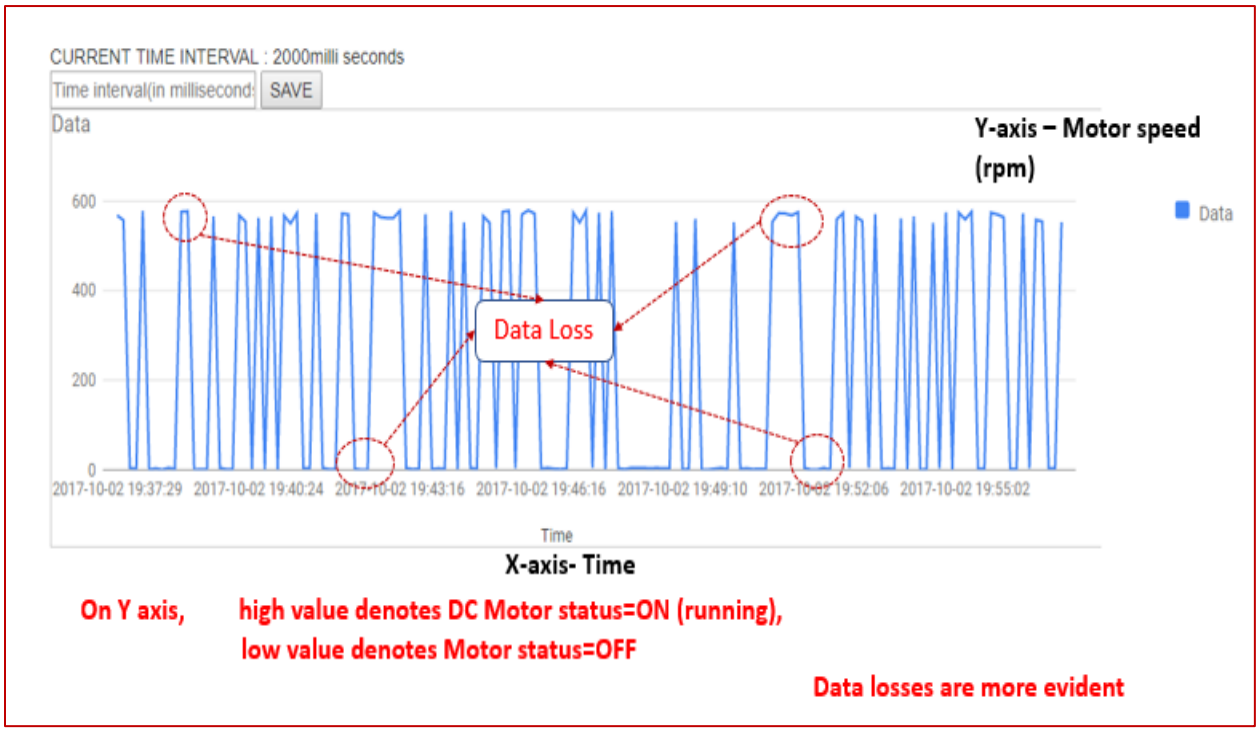

Figure 6.8 Experimental graph results for 2 seconds' time interval

Table 6.6: Data for 2 seconds' time interval

Timestamp	Data
10/2/2017 19:37	568
10/2/2017 19:37	557
10/2/2017 19:37	4
10/2/2017 19:37	2
10/2/2017 19:38	578
10/2/2017 19:38	0
10/2/2017 19:38	3
10/2/2017 19:38	0
10/2/2017 19:38	4
10/2/2017 19:38	2
10/2/2017 19:38	576
10/2/2017 19:38	577
10/2/2017 19:39	0
10/2/2017 19:39	1
10/2/2017 19:39	1
10/2/2017 19:39	566
10/2/2017 19:39	4
10/2/2017 19:39	1
10/2/2017 19:39	1
10/2/2017 19:40	568
10/2/2017 19:40	554
10/2/2017 19:40	2
10/2/2017 19:40	563
10/2/2017 19:40	0
10/2/2017 19:40	565
10/2/2017 19:40	0
10/2/2017 19:40	567
10/2/2017 19:41	550
10/2/2017 19:41	573
10/2/2017 19:41	3
10/2/2017 19:41	3

Figure 6.9 Experimental graph results for 4 seconds' time interval

Table 6.7: Data for 4 seconds' time interval

Timestamp	Data
10/2/2017 19:03	4
10/2/2017 19:04	553
10/2/2017 19:04	550
10/2/2017 19:04	554
10/2/2017 19:04	0
10/2/2017 19:04	2
10/2/2017 19:05	556
10/2/2017 19:05	574
10/2/2017 19:05	572
10/2/2017 19:05	550
10/2/2017 19:05	563
10/2/2017 19:05	0
10/2/2017 19:05	1

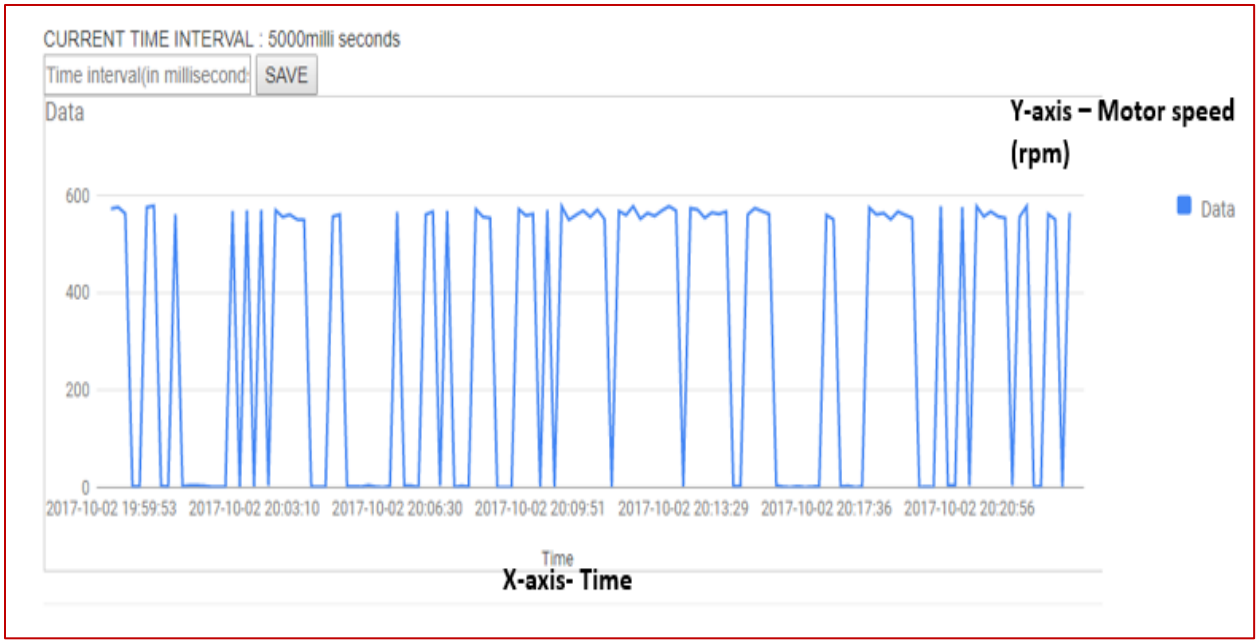

Figure 6.10 Experimental graph results for 5 seconds' time interval

Table 6.8: Data for 5 seconds' time interval

Timestamp	Data
10/2/2017 19:11	3
10/2/2017 19:11	4
10/2/2017 19:12	557
10/2/2017 19:12	572
10/2/2017 19:12	2
10/2/2017 19:12	562
10/2/2017 19:12	567
10/2/2017 19:12	563
10/2/2017 19:12	550
10/2/2017 19:13	579
10/2/2017 19:13	552
10/2/2017 19:13	0
10/2/2017 19:13	0
10/2/2017 19:13	0

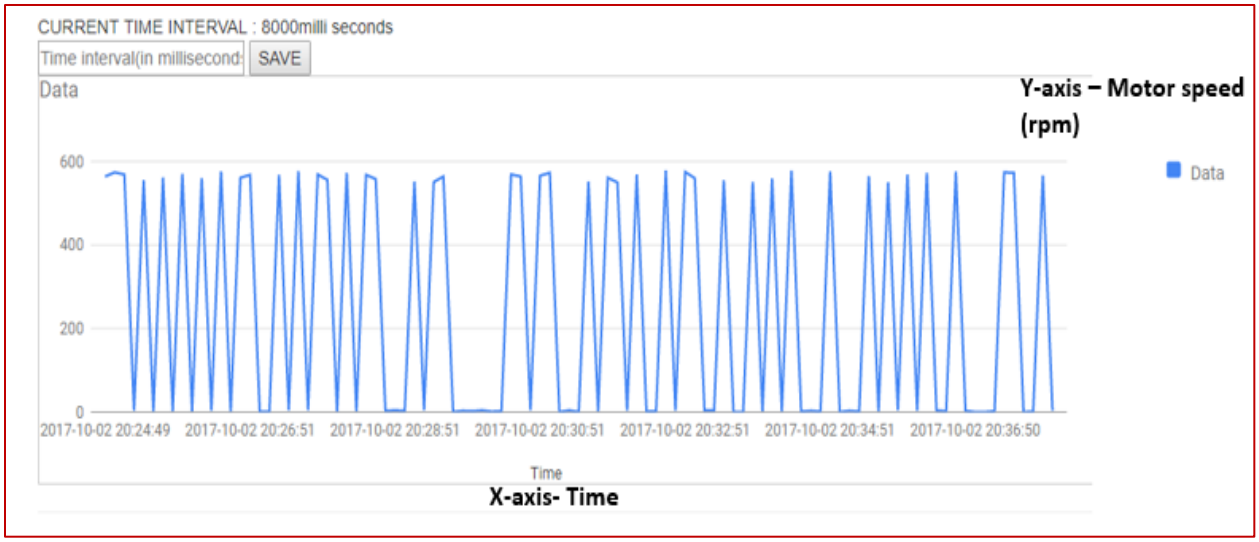

Figure 6.11 Experimental graph results for 8 seconds' time interval

Table 6.9: Data for 8 seconds' time interval

Timestamp	Data
10/2/2017 20:24	564
10/2/2017 20:24	574
10/2/2017 20:25	569
10/2/2017 20:25	3
10/2/2017 20:25	555
10/2/2017 20:25	1
10/2/2017 20:25	562
10/2/2017 20:25	2
10/2/2017 20:25	571
10/2/2017 20:26	2
10/2/2017 20:26	560
10/2/2017 20:26	3
10/2/2017 20:26	576
10/2/2017 20:26	1
10/2/2017 20:26	561
10/2/2017 20:26	568
10/2/2017 20:26	1
10/2/2017 20:27	1
10/2/2017 20:27	568
10/2/2017 20:27	4
10/2/2017 20:27	577
10/2/2017 20:27	4
10/2/2017 20:27	569
10/2/2017 20:27	556
10/2/2017 20:28	1

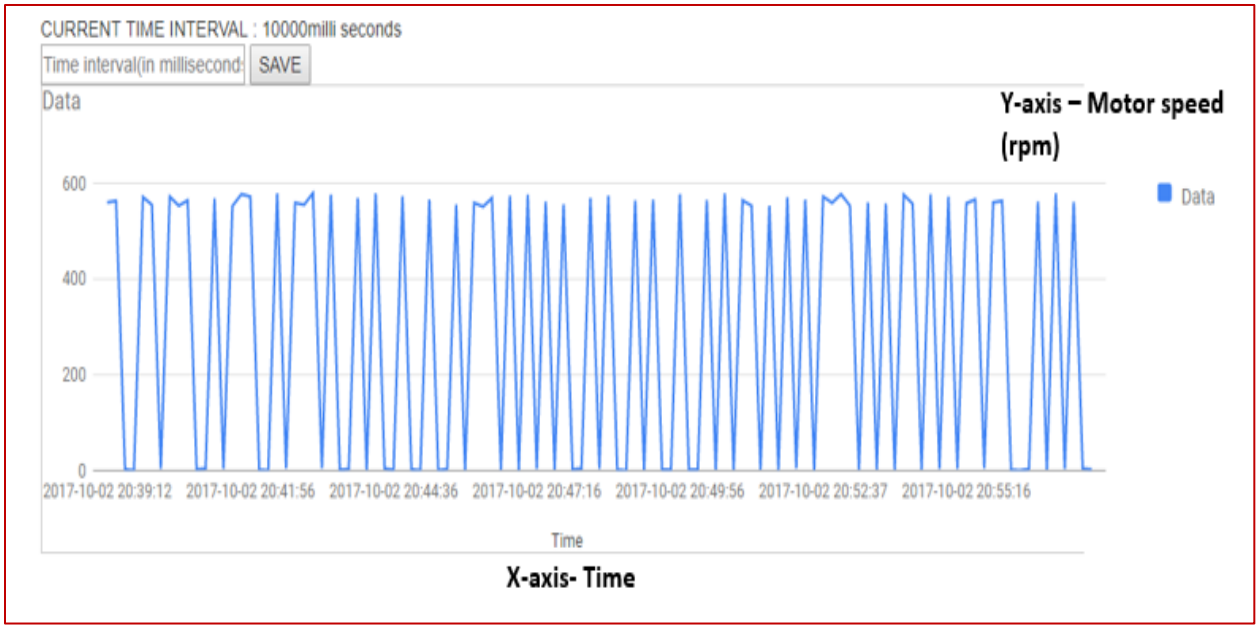

Figure 6.12 Experimental graph results for 10 seconds' time interval

Table 6.10: Data for 10 seconds' time interval

Timestamp	Data
10/2/2017 20:39	560
10/2/2017 20:39	563
10/2/2017 20:39	2
10/2/2017 20:39	0
10/2/2017 20:39	571
10/2/2017 20:40	555
10/2/2017 20:40	3
10/2/2017 20:40	572
10/2/2017 20:40	553
10/2/2017 20:40	564
10/2/2017 20:40	2
10/2/2017 20:41	4
10/2/2017 20:41	568
10/2/2017 20:41	3
10/2/2017 20:41	552
10/2/2017 20:41	577
10/2/2017 20:41	572
10/2/2017 20:42	1
10/2/2017 20:42	0
10/2/2017 20:42	579
10/2/2017 20:42	4

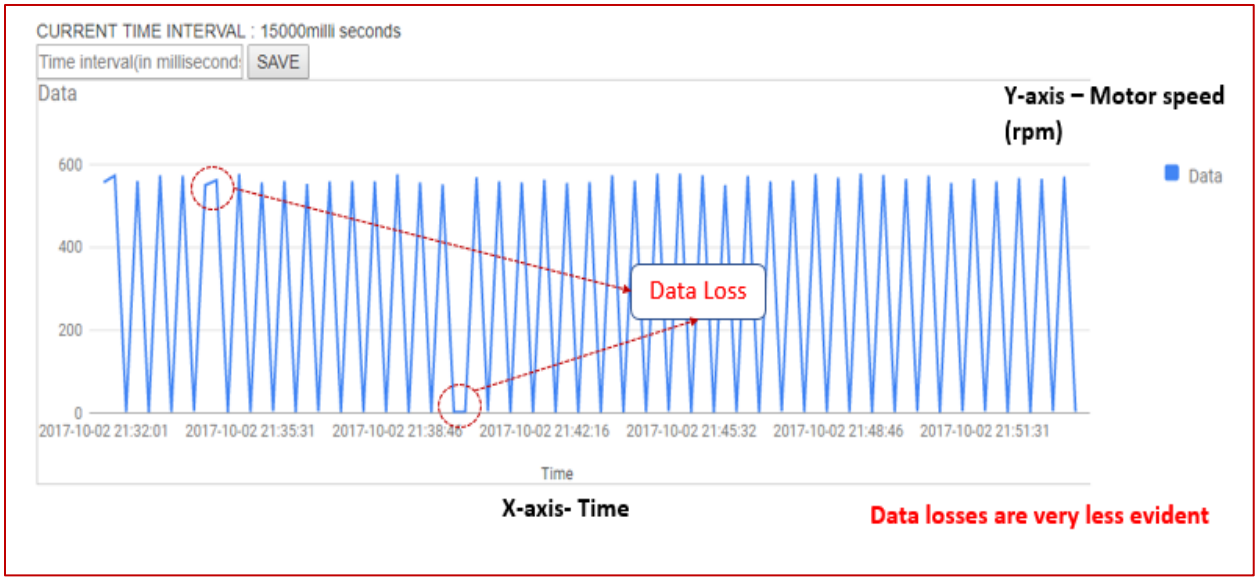

Figure 6.13 Experimental graph results for 15 seconds' time interval

Table 6.11: Data for 15 seconds' time interval

Timestamp	Data
10/2/2017 21:09	570
10/2/2017 21:09	4
10/2/2017 21:09	555
10/2/2017 21:09	3
10/2/2017 21:09	565
10/2/2017 21:10	2
10/2/2017 21:10	578
10/2/2017 21:10	3
10/2/2017 21:10	542
10/2/2017 21:11	4
10/2/2017 21:11	550
10/2/2017 21:11	0
10/2/2017 21:11	567
10/2/2017 21:12	2
10/2/2017 21:12	579
10/2/2017 21:12	3
10/2/2017 21:12	568
10/2/2017 21:13	4
10/2/2017 21:13	571
10/2/2017 21:13	2
10/2/2017 21:13	567
10/2/2017 21:14	4
10/2/2017 21:14	571
10/2/2017 21:14	2

Chapter 7

Model Predictive Control

It is a type of control in which, at each sampling instant, the current control the action is obtained by solving online. In this method, it uses the current state of the plant as the initial state; the optimization yields, the first control in this sequence and an optimal control is applied to the system.

These controllers rely on dynamic models of the process, most often by system identification linear empirical models are obtained. The main advantage of MPC is that while keeping in account future time slots, it allows the current time slot to optimize. MPC can take future control actions because it can anticipate the future events. This predictive ability is missing in PID controllers.

Most MPC systems are based on the concept of producing values for process model and other determinations. In MPC construction there is a feedback or feedforward path to compute the process measurements. There are various forms are available to make model predictive controller:

Feedback MPC- This type of MPC mitigates shrinkage of the feasible region.

Robust MPC- Through this type of MPC we get feasibility and stability of the system.

Decentralized MPC- For very fast response decentralized MPCs are used.

Pre-computed MPC- In this MPC the optimization is an off-line process. Mainly the parameters are solved by linear or quadratic programming.

7.1 Model predictive control structure

There are three main components in model predictive control structure

1. The process model

2. The cost function

3. The optimizer

In the process model, according to the manipulated control variables, the information about the controlled process and the prediction of the response is calculated. Then bye the minimization of cost function error is reduced. Basic structure of MPC is described in Figure 7.1.

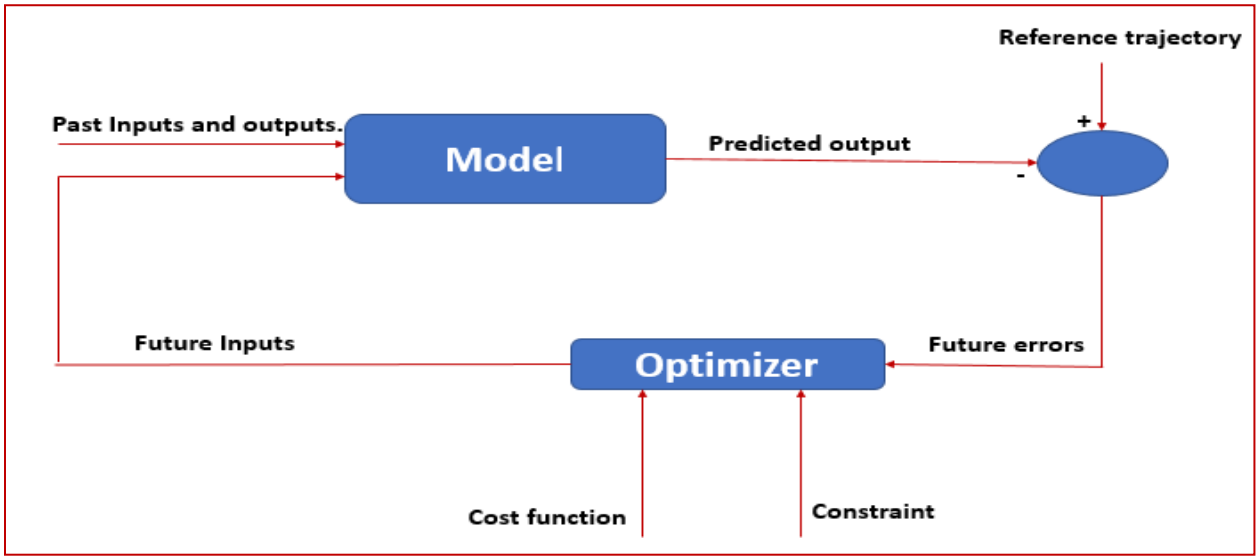

Figure 7.1 Basic structure of Model Predictive Controller

(Source: P. E. Orukpe, "Basics of Model Predictive Control", ICM, EEE-CAP, Imperial College, London April 14, 2005)

In the last step, various types of optimization techniques are used and the output gives to the input sequence for the next prediction horizon.

Model predictive control is a multivariable control algorithm that uses:

- An internal dynamic model of the process,
- A history of past control moves, and
- An optimization cost function J over the receding prediction horizon to calculate the optimum control moves.

The optimization cost function is given by:

$$J = \sum_{i=1}^{N} w_{xi}(r_i - x_i)^2 + \sum_{i=1}^{N} w_{xi}\Delta u_i{}^2$$

(Source. [Online]: https://www.mathworks.com/help/mpc/ug/optimization-problem.html)

where,

x_i = i-th control variable

r_i = i-th reference variable

u_i = i-th manipulated variable

w_{xi} = weighting coefficient reflecting the relative importance of x_i

7.2 Time delay and data loss compensation using MPC

The wireless NCS (Network control system) is a closed loop system consisting of a Plant (Actuator), sensor to provide feedback and a controller. The wireless input is communicating with the DC motor using shared network. Wireless network is not a stable network, it always consists of delays and even data losses.

Transfer function

$$\frac{Y(s)}{R(s)} = \frac{G_c(s).G_p(s)e^{-t_1 s}}{1 + G_c(s).G_p(s)e^{-t_1 s}}$$

(Source. [online]: http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling)

t_1 is the time delay at the input,

$G_c(s)$ is the transfer function of the controller,

$G_p(s)$ is the transfer function of the DC motor without the time delay.

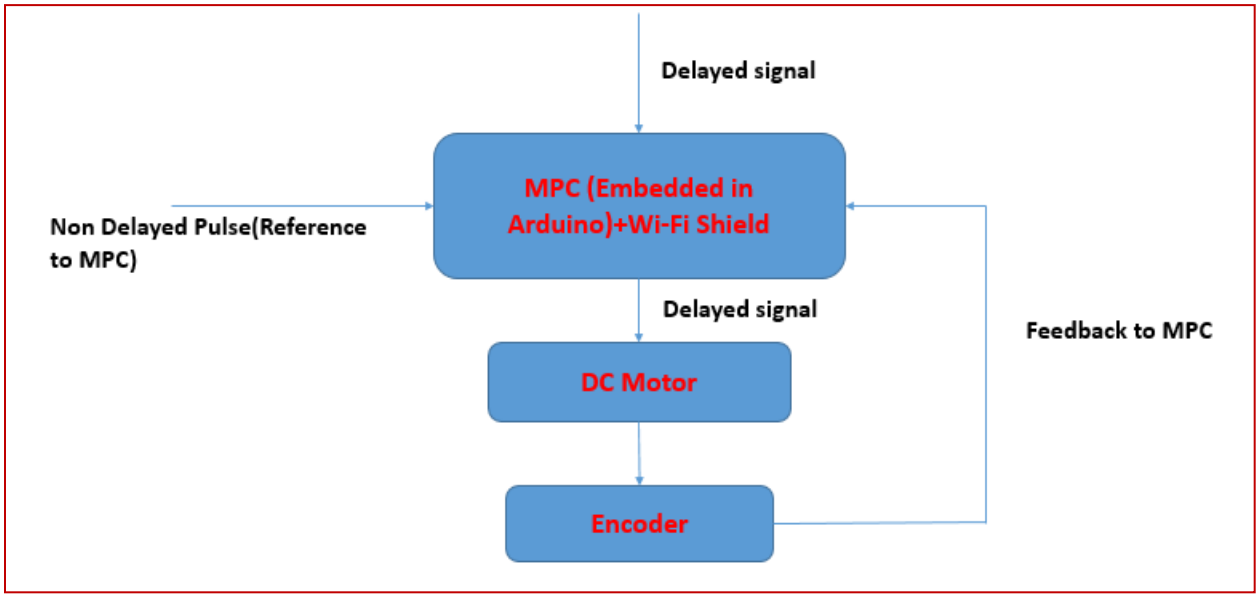

Figure 7.2 Closed loop architecture of network delayed signal

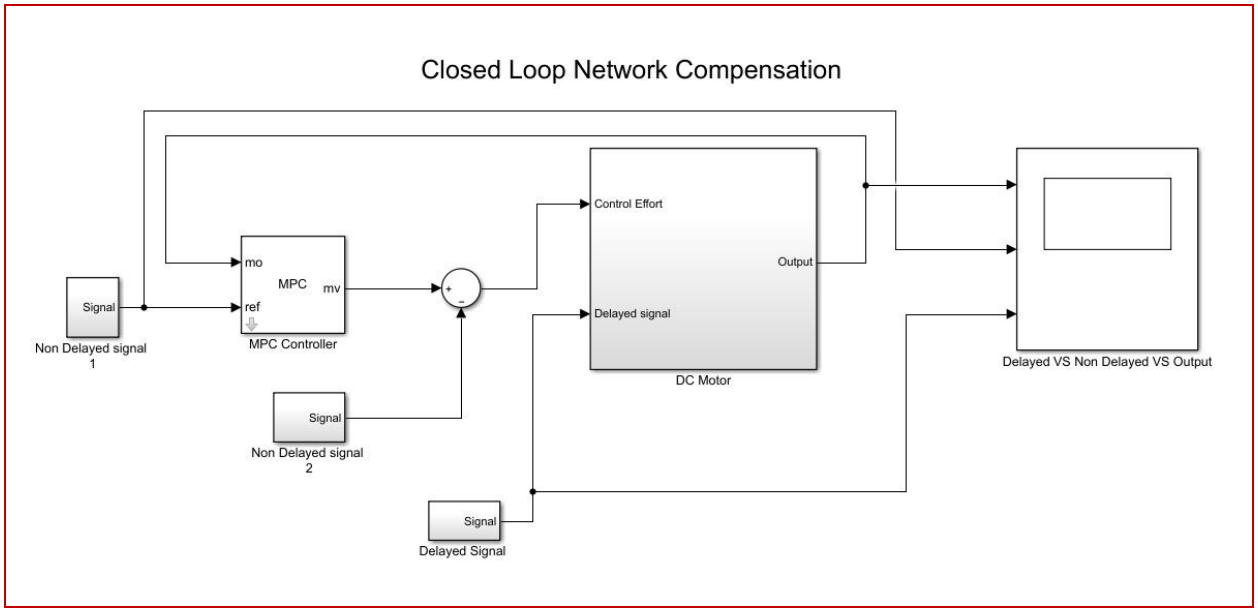

Figure 7.3 Closed loop network compensation using MPC

To compensate the network latency and the data losses Model Predictive Control has been used. The property of the Model Predictive controller to predict the future will be exploited in order to compensate the losses in the network. In our system as described in the Fig 7.2 below the delay gets introduced when the Wi-Fi shield gets the signal from the webpage. Data packets also gets lost when the latency exceeds a certain limit.

The input in our system is in the form of a pulse with a constant width throughput the cycle. When the pulse is high, DC motor gets a signal to turn on depending on the time period of the pulse and it turns off when the pulse is low. Due to latency in network, our input to the motor suffers time delay. Let's assume the case of 0.5 sec time delay, two situations arise either the motor will stay high for extra 0.5 sec or will stay low for 0.5 sec. The worst possibility is the loss of the data,

making our actuator to stay on low or high for the 3NT seconds (where T is the time period of the pulse, N number of data packets lost).

Model Predicative controller which is a robust controller will be used to compensate for network losses and delays. MPC should produce the control effort in such a way that will command motor as per the non-delayed ideal signal.

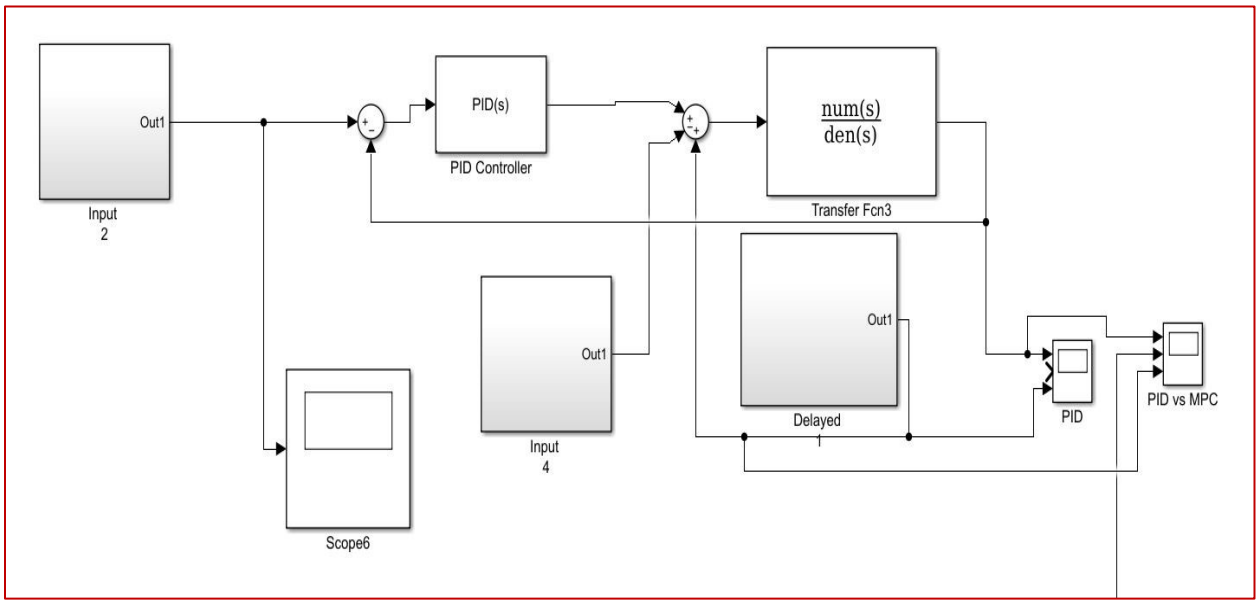

Figure 7.4 Closed loop network compensation using PID

Chapter 8

Simulation Results and discussions

Model Predictive Control has been used to compensate the delay of the system in the network. This chapter present the simulation performance evaluation of our proposed scheme.

The transfer function and parameters of the DC motor used to produce the simulation results are as follows:

Transfer function of the DC motor:

$$\frac{0.01}{0.005s^2 + 0.06s + 0.1001}$$

Table 8.1 Parameters of the DC Motor:

Sr. No	Physical Parameters(Units)	Values
1.	Moment of inertia of the rotor (kg.m^2)	0.01
2.	Motor viscous friction constant (Nms)	0.1
3.	Electromotive force constant (V/rad/Amp)	0.01
4.	Motor Torque Constant (Nm/Amp)	0.01

5.	Electric Resistance (Ohm)	1
6.	Electric Inductance (H)	0.5

In the system, we are using Linear MPC to handle the latency in the network and it parameters are as follows:

Table 8.2 Parameters of the Model Predictive Control:

Sr. No	**MPC parameters**	**Value**
1.	Sample time	0.1
2.	Prediction Horizon	10
2.	Control Horizon	2
4.	Input Scale u(1)	1
5.	Output Scale y(1)	1
6.	Input weight	0
7.	Input Rate weight	0.0201896517
7.	Output weights	4.95303242439511

(Data Source [Online]: https://www.mathworks.com/help/mpc/controller-design.html)

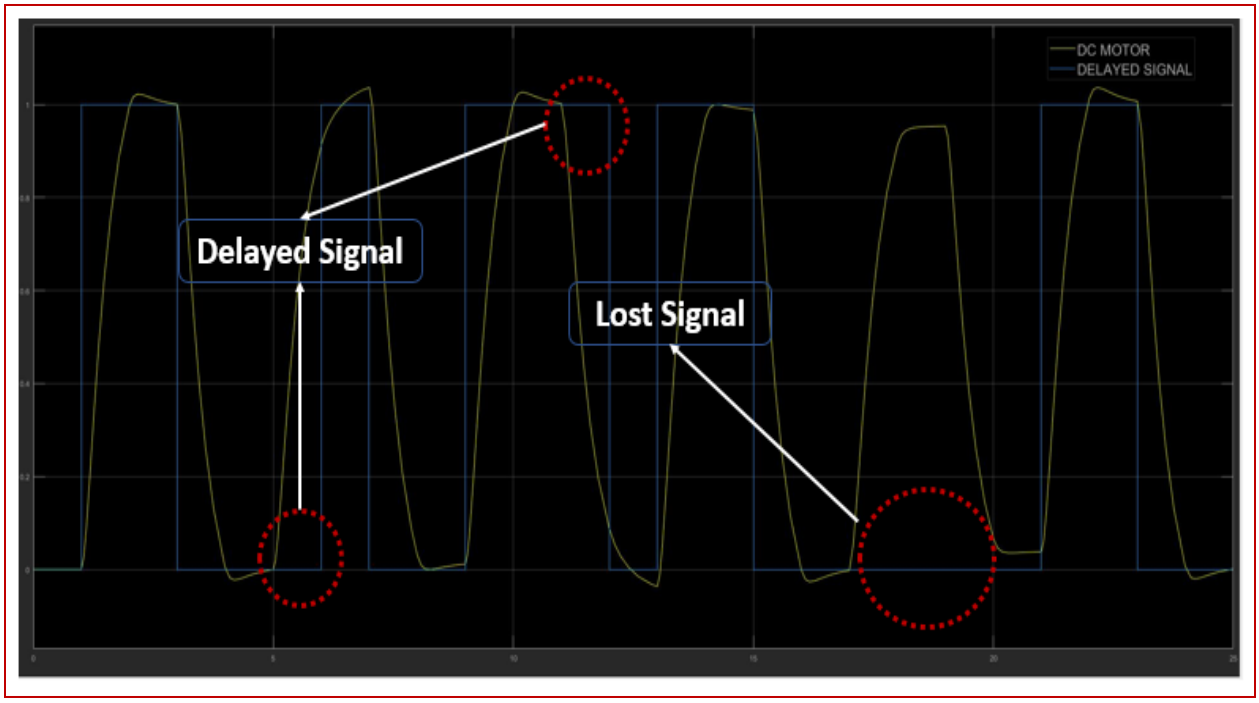

Figure 8.1 Simulation result for network compensation using MPC

The simulation results in Figure 8.1, show the compensation of the network delays and losses. The results we obtained describes that the model predictive control can predict the delays and the losses vividly and hence the output of the DC motor is unaffected by the network interruptions.

Time delays always have the significant effect on the system which makes the system erroneous. By introducing a compensator as per the system requirements the difference between the two signals can be reduced to the reference value. The obtained simulation results show the effectiveness of the MPC for the case of constant delays in the network, improving the system performance.

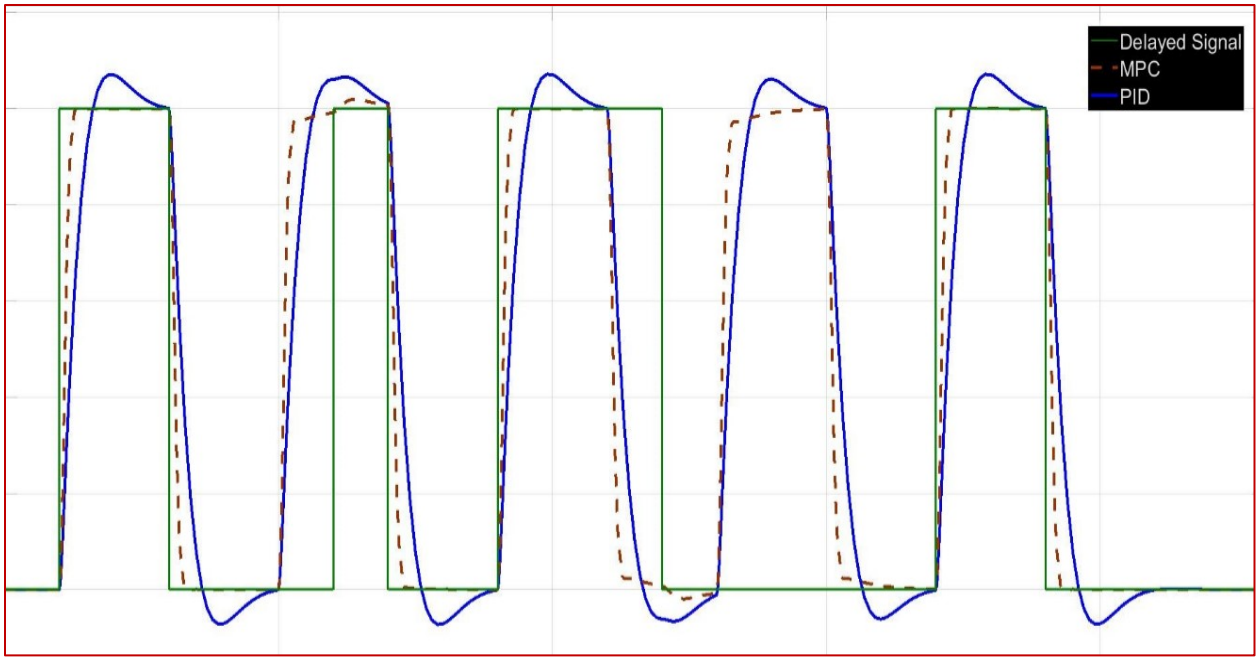

Figure 8.2 Simulation results comparison for network compensation using PID and MPC

As we compare the results of PID and MPC we got with simulation shown in Figure 8.2, it shows PID having more overshoot and undershoot as compared with the MPC. MPC has an advantage over the PID as it's a robust controller and gives the more stable results in comparison with PID controller, making it more reliable.

Chapter 9

Conclusion and future work

Wireless network control system depends heavily on the network protocol. Because of the time delays NCS gets unstable, in order to stabilize the system a controller must be used for the network delay and loss compensation. In the proposed study, the model predictive controller has been used in order to counter the effects of the time delay. The system has been built by using simulation environment on MATLAB/Simulink software. The proposed scheme gives us the better results as compared to the other literatures.

In future work, the practical experimentation and evaluation with the constant and variable time delays will be done. Other delay compensation schemes can be applied and compared with the proposed ones.

The experimentally recorded data shows the randomness in the time delay and the data losses. The future work will help present the intended experimental study of a WSN for a velocity remote control of a mobile robot and a block diagram like system 1 and system 2, but for the robot velocity command in the form of a sinusoidal velocity command and robot velocity estimation subject to random cycle time. [26] This study will help to operate remotely based sensor nodes and eventually record the sensor data accounting for the time delay and data loss problems.

References

[1] Munindar P. Singh and Amit K. Chopra The Internet of Things and Multiagent Systems. [Online]. Available: https://www.csc2.ncsu.edu/faculty/mpsingh/papers/tmp/notes-IoT.pdf

[2] Munindar P. Singh and Amit K. Chopra. Internet of Things. Munindar P. Singh [Online]. Available: https://www.csc2.ncsu.edu/faculty/mpsingh/tutorials/IoT.

[3] Yinbiao, Shu, P. Lanctot, and Fan Jianbin. "Internet of things: wireless sensor networks." White Paper, International Electrotechnical Commission, http://www.iec.ch (2014).

[4] Ashton, K. That 'Internet of Things' Thing. In the real world, things matter more than ideas. RFID Journal, 22 June 2009. Available from: http://www.rfidjournal.com/articles/view?4986.

[5] Arne Bröring. New generation sensor web enablement. Sensors, 11, 2011, pp. 26522699. ISSN1424-8220. [Online]. Available from: doi:10.3390/s110302652.

[6] Sensei. Integrating the physical with the digital world of the network of the future. [Online]. Available: http://www.sensei-project.eu.

[7] Kang G. Shin and Xianzhong Cui. "Computing Time Delay and Its Effects on Real-Time Control Systems". IEEE transactions on control systems technology, vol. 3, no. 2, June 1995.

[8] Thomas Strohmer and Jared Tanner. "Implementations of Shannon's sampling theorem, a time-frequency approach". Vol. 4, No. 1, Jan. 2005, pp. 1-17 ISSN: 1530-6429.

[9] P. K. Khosla, "Choosing Sampling Rates for Robot Control," Proceedings. IEEE Int. Con5 Robotics Automat., 1987, pp. 169-174.

[10] T. Mita, "Optimal digital feedback control systems counting computation time of control laws," IEEE Trans. AuromatContr., vol. AC-30, no. 6, pp. 542-547, June 1985.

[11] Luigi Atzori, Antonio Iera, Giacomo Morabito. The Internet of Things: A survey. Computer Networks 54 (2010) 2787–2805, 1 June 2010. [Online]. Available: www.elsevier.com/locate/comnet.

[12] Luigi Atzori, Antonio Iera, Giacomo Morabito. The Internet of Things: A survey. Computer Networks 54 (2010) 2787–2805, 1 June 2010. [Online]. Available: www.elsevier.com/locate/comnet.

[13] "Cloud and Mobile Network Traffic Forecast - Visual Networking Index (VNI)." Cisco, 2015. [Online]. Available: http://cisco.com/c/en/us/solutions/serviceprovider/visual-networking-index-vni/index.html

[14] Danova, Tony. "Morgan Stanley: 75 Billion Devices Will Be Connected to The Internet of Things By 2020." Business Insider, October 2, 2013. [Online]. Available: http://www.businessinsider.com/75-billion-devices-will-be-connected-to-the-internet-by-2020-2013-10

[15] "Global Connectivity Index." Huawei Technologies Co., Ltd., 2015. Web. 6 Sept. 2015.

[Online]. Available: http://www.huawei.com/minisite/gci/en/index.html

[16] Manyika, James, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. “The Internet of Things: Mapping the Value Beyond the Hype.” McKinsey Global Institute, June 2015.

[17] Dave Thaler, Hannes Tschofenig, Mary Barnes. “Architectural Considerations in Smart Object Networking” Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. [Online]. Available: https://www.rfceditor.org/rfc/rfc7452.txt.

[18] Bluetooth [Online]. Available: http://www.bluetooth.com and http://www.bluetooth.org.

[19] Z-Wave [Online]. Available: http://www.z-wave.com.

[20] Zigbee [Online]. Available: http://www.zigbee.org.

[21] Duffy Marsan, Carolyn. "IAB Releases Guidelines for Internet-of-Things Developers." IETF Journal 11.1 (2015): 6-8. Internet Engineering Task Force, July 2015. [Online]. Available: https://www.internetsociety.org/sites/default/files/Journal_11.1.pdf

[22] “Meet the Nest Thermostat | Nest.” Nest Labs. 31 Aug. 2015. [Online]. Available: https://nest.com/thermostat/meet-nest-thermostat.

[23] “Samsung Privacy Policy--SmartTV Supplement.” Samsung Corp. 29 Sept. 2015.

[Online]. Available: http://www.samsung.com/sg/info/privacy/smarttv.html.

[24] Duffy Marsan, Carolyn. "IAB Releases Guidelines for Internet-of-Things Developers." IETF Journal 11.1 (2015): 6-8. Internet Engineering Task Force, July 2015. [Online]. Available: https://www.internetsociety.org/sites/default/files/Journal_11.1.pdf.

[25] “How It Works.” SmartThings, 2015. [Online]. Available: http://www.smartthings.com/how-it-works

[26] Arpit Ainchwar, Dan Necsulescu. “Determination of cycle time constraints in case of link failure in closed loop control in Internet of Things”. Proceedings of the 4th International Conference of Control, Dynamic Systems, and Robotics (CDSR'17) Toronto, Canada – August 21 – 23, 2017 Paper No. 122 DOI: 10.11159/cdsr17.122.

Appendix A: Codes

1.Code for connecting LED in Wi-Fi network.

```
#include <ESP8266WiFi.h>
int LED = 13;
char ssid[] = "Sher-E Hindustan"; //  your network SSID (name)
char pass[] = "vandematram";    // your network password
int keyIndex = 0;                // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
WiFiServer server(8080);
void printWifiStatus();

void setup() {
 pinMode(LED,OUTPUT);

  //Initialize serial and wait for port to open:
  Serial.begin(115200);

  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
Serial.println("Yes...Code is running");
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop() {

  // listen for incoming clients
  WiFiClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);

        if (c == 'T')
```

```
{
 c = client.read();
 if (c == 'N')
 {
  Serial.print("LED ON");
  digitalWrite(LED,1);
 } else if (c == 'F')
 {
  Serial.print("LED OFF");
  digitalWrite(LED,0);


 }
}
if (c == '\n' && currentLineIsBlank) {
 // send a standard http response header
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/html");
 client.println("Connection: close");  // the connection will be closed after completion of the response
 client.println("</html>");
 break;
}
if (c == '\n') {
 // you're starting a new line
 currentLineIsBlank = true;
} else if (c != '\r') {
 // you've gotten a character on the current line
 currentLineIsBlank = false;
}
```

```
      }
    }
    // give the web browser time to receive the data
    delay(1);

    // close the connection:
    client.stop();
    Serial.println("client disonnected");
  }
}
void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

2. Code for uploading photocell data to the website.

```
#include <SPI.h>
#include <WiFi101.h>

// Local Network Settings
char ssid[] = "Pakke Canada wale";  // your network SSID (name)
char pass[] = "Vandematram";    // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;

WiFiServer server(80);

// ThingSpeak Settings
char thingSpeakAddress[] = "http://datalogger.comeze.com";
//String APIKey = "JSVKEBSOFPUUJ3H4";             // enter your channel's Write API Key
const int updateThingSpeakInterval = 5 * 1000; // 20 second interval at which to update ThingSpeak

// Variable Setup
long lastConnectionTime = 0;
boolean lastConnected = false;

// Initialize Arduino Ethernet Client
WiFiClient client;

void setup() {
  // Start Serial for debugging on the Serial Monitor
  Serial.begin(9600);
```

```
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while (true);
  }

  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  // you're connected now, so print out the status:
  printWifiStatus();
}
int Photocell_Old = 0;
int bDataChanged = 0;
int Photocell_New = 0;
int difference = 0;
```

```
int value = 0;
int tVariable = 0;
int tlast = 0;
void loop() {
  Photocell_New = analogRead(A0);                 // read Photocell value
  if (Photocell_New > 200 )
  {
    tVariable = 1;
  }
  else {
    tVariable = 0;
  }
  if (tlast != tVariable)
  {
    tlast = tVariable;
    Serial.print("_____ Changed _____");
    Serial.print(tVariable);
    Serial.print("]");
    Serial.println(Photocell_New);

    int temp;// = String(Photocell_New);//String(Photocell_New);
    if (tVariable == 0) {
      temp = random(0, 5);
      Serial.println("Value = ");
      Serial.print(temp);
    }
    else {
      temp = random(200, 230);
```

```
      Serial.println("Value = ");
      Serial.print(temp);
    }
    updateThingSpeak(String(temp));
  }
}

void updateThingSpeak(String tsData) {
  Serial.println("Now Sending");
  if (client.connect("www.datalogger.pe.hu", 80)) {
    Serial.println("Connected");
    client.print("GET /add.php?temp1=");
    client.print(tsData);
    client.println(" HTTP/1.1");
    client.println("Host: www.datalogger.pe.hu");
    client.println();
    Serial.println();
    delay(1000);
    Serial.println("Data Sent");

    client.stop();
  }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
```

```
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

3. Code for operating DC motor remotely.

```
#include <SPI.h>
#include <WiFi101.h>

int dir1PinA = 2;  // in2
int dir2PinA = 3;  // in3
int speedPinA = 9; // EN2  Needs to be a PWM pin to be able to control motor speed

int c = 0;

char ssid[] = "Sher-E Hindustan"; //  your network SSID (name)
char pass[] = "vandematram";    // your network password

int status = WL_IDLE_STATUS;
WiFiServer server(8080);
```

```
void setup() {

  pinMode(dir1PinA,OUTPUT);
  pinMode(dir2PinA,OUTPUT);
  pinMode(speedPinA,OUTPUT);

  //Initialize serial and wait for port to open:
  Serial.begin(9600);

  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while (true);
  }

  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  server.begin();
  // you're connected now, so print out the status:
  printWifiStatus();
}
```

```
void loop() {
  WiFiClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);

        if (c == 'T')
        {
          c = client.read();
          if (c == 'N')
          {
            Serial.print("MOTOR ON");
            analogWrite(speedPinA, 150);//Sets speed variable via PWM
            digitalWrite(dir1PinA, LOW);
            digitalWrite(dir2PinA, HIGH);

          } else if (c == 'F')
          {
            Serial.print("MOTOR OFF");
            analogWrite(speedPinA, 0);//Stop motor by giving 0 PWM
            digitalWrite(dir1PinA, LOW);
            digitalWrite(dir2PinA, LOW);
```

```
        }
       }
      }
    }
    client.stop();
    Serial.println("client disonnected");
  }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

4. Code for posting DC motor velocity.

```
#include <SPI.h>
#include <WiFi101.h>

// Local Network Settings
char ssid[] = "Sher-E Hindustan";                  //N etwork SSID (name)
char pass[] = "vandematram";                       // your network password

#define MainPeriod 100

// Variable Setup
long previousMillis = 0;                           // will store last time of the cycle end
volatile unsigned long duration=0;                 // accumulates pulse width
volatile unsigned int pulsecount=0;
volatile unsigned long previousMicros=0;

int Speed_Old = 0;
int bDataChanged = 0;
int Speed_New = 0;
int difference = 0;
int value=0;
int Speed;

long lastConnectionTime = 0;
boolean lastConnected = false;

int status = WL_IDLE_STATUS;
```

```
WiFiServer server(80);
WiFiClient client;

void setup() {
  attachInterrupt(0, myinthandler, RISING);     // Initialize External Interrupt
  Serial.begin(9600);                    // Start Serial for debugging on the Serial Monitor with baudrate 9600
  while (!Serial) {
    ;                                    // wait for serial port to connect.
  }

 if (WiFi.status() == WL_NO_SHIELD) {          // check for the presence of the shield:
    Serial.println("WiFi shield not present");
    while (true);                        // don't continue:
  }

  while ( status != WL_CONNECTED) {            // attempt to connect to Wifi network:
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);            // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    delay(5000);                         // wait 5 seconds for connection:
  }
  printWifiStatus();                      // you're connected now, so print out the status:
}

void loop() {
  unsigned long currentMillis = millis();
```

```
  if (currentMillis - previousMillis >= MainPeriod)
  {
    previousMillis = currentMillis;
    unsigned long _duration = duration;     // need to bufferize to avoid glitches
    unsigned long _pulsecount = pulsecount;
    duration = 0;                           // clear counters
    pulsecount = 0;
    float Freq = 1e6 / float(_duration);    //Duration is in uSecond so it is 1e6 / T


    Freq *= _pulsecount;                    // calculate Frequency
    Serial.print("Frequency: ");
    Serial.print(Freq);
    Serial.println("Hz");
    Speed = Freq;
  }


  Speed_New = Speed ;
  difference = Speed_New - Speed_Old;       // Find out the difference between two consitutive
speed
  if (difference < 0) {
    difference = -difference;
  }
  if (difference > 200) {                   // Check the differnece betwen two consitutive difference
    bDataChanged = 1;                       // Set the flag to upload the value
    value ^=1;                            // Toggle flag
    Serial.println("Data Changed");         // Serially display message
  }
  else {
    bDataChanged = 0;                       // Clear the flag
```

```
  }
  Speed_Old = Speed_New;
  String temp = String(value*550);
  Serial.println(temp);

  if (client.available()) {
    char c = client.read();
    Serial.print(c);                    // Print Update Response to Serial Monitor
  }
  if (!client.connected() && lastConnected) { // check the status of client connection
    Serial.println("...disconnected");
    Serial.println();
    client.stop();
  }
  if (!client.connected() && (bDataChanged == 1)) {
    UpdateOnWeb(temp);                  // Update the value on Web
    Serial.println("Data Changed");     // Serially display message
  }
  lastConnected = client.connected();
}
void myinthandler()                     // interrupt handler
{
  unsigned long currentMicros = micros();
  duration += currentMicros - previousMicros;
  previousMicros = currentMicros;
  pulsecount++;
}
```

```
// Function to send the php commands to send the value on server
void UpdateOnWeb(String tsData) {
  if (client.connect("www.datalogger.pe.hu", 80)) {
    Serial.println("Connected");
    client.print("GET /add.php?temp1=");
    client.print(tsData);
    client.println(" HTTP/1.1");
    client.println("Host: www.datalogger.pe.hu");
    client.println();
    Serial.println();
    delay(1000);
    lastConnectionTime = millis();

    if (client.connected()) {
      Serial.println();
    }
  }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
```

```
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```