OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
FACULTY OF COMPUTER SCIENCE
Communication and Networked Systems

Communication and Networked Systems

Master Thesis

Li-Fi (Light-Fidelity) in Industry 4.0: Integration of Li-Fi Communication in a Factory Planning Laboratory

Vasu Dev Mukku

Matr. 213509

Supervisor: Prof. Dr. rer. nat. Mesut Güneş
Assisting Supervisor: M.Sc. Marian Buschsieweke

Institute for Intelligent Cooperating Systems, Otto-von-Guericke-University Magdeburg

23. August 2019

Abstract

Abstract

Currently, most of the factory and material flow operations in manufacturing industries are carried out in static factory layouts. Due to the static layout, there is a limitation in flexibility. Flexibility denotes the ability to change the factory layout to optimize the production process for different production programmes. In this context, there is a need for research in the direction of modular, decentralized and distributed process planning for dynamic factory layouts. One major challenge is the design of an appropriate strategy for communication between the factory modules and the identification of appropriate communication technologies. The communication design has two main requirements. First, to make use of wireless communication for changing the factory structure. Second, the communication strategy should be able to quickly identify the current factory layout and the interconnection of factory resources.

According to the current state of knowledge, Light Fidelity (Li-Fi) is one of the adaptable technology to fulfill these requirements. In this thesis, a concept for the integration of Li-Fi communication in the factory planning laboratory is proposed. The distributed factory layouts are demonstrated by applying Li-Fi communication. Specific experiments are conducted to design a robust Li-Fi protocol with a suitable transceiver and reliable communication. RIOT-OS, Arduino-Mega 2560, Li-Fi transceivers and Fischertechink factory modules are used to build modular factory structures.

Acknowledgments

First, I would like to express my sincere thanks to my advisor M. Sc. Marian Buschsieweke, for the constant support of my master thesis and research, for his patience, motivation and tremendous knowledge. His guidance helped me in all the time of implementation and writing of this thesis.

Besides my advisor, I would like to thank Dr. Tobias Reggelin and M. Sc. Sebastian Lang for their encouragement and support to my idea of implementation of Li-Fi communication in the Learning Laboratory.

Finally, I must express my very heartfelt gratitude to my parents and to my brother for providing me with constant support and continuous encouragement throughout my years of study. I'm very thankful to my friends, who helped me through the process of writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

Listings

Acronyms

ADC Analog to Digital Converter. 15, 20, 21, 24, 52
API Application Programmable Interface. 13
ASK Amplitude Shift Keying. 17

CLI Command Line Interpreter. 13
CM Conveyor Motor. 26, 27, 29, 38, 39
CPS Cyber-physical systems. 8
CPU Central Processing Unit. 13

FCS Frame Check Sequence. 10, 11, 18, 19

GIC Global Identification Code. 9
GPIO General Purpose Input and Output. 15, 25

HDLC High-Level Data Link Control. 7, 10, 11, 17, 51

IoT Internet of Things. 7, 12–14
IR Infrared. 13, 14

LDR Light Dependent Resistor. 15, 16, 20, 25, 32–34, 36, 41, 42, 51
LED Light Emitting Diode. 9, 10, 14–17, 20, 24–26, 28, 29, 31–36, 38, 39, 41, 42, 51
Li-Fi Light Fidelity. iii, 3–7, 9, 10, 13–17, 20–28, 30–36, 38, 39, 41–43, 45, 51, 52
LOS Line of Sight. 3
LS Left Switch. 28

MCU Micro Controller Unit. 13
MSB Most Significant Bit. 18, 19

OOK On-Off Keying. 5, 9, 10, 17, 19, 51
OSI Open Systems Interconnection. 9, 18, 52

PDU Protocol Data Unit. 18, 19

PLC Programmable Logic Controller. 3, 14

PLiFi Parallel LiFi. 24

PS Proximity Sensor. 26, 27, 38

PS2 Proximity Sensor 2. 38, 39

RAM Random Access Memory. 17

RF Radio Frequency. 3, 13, 14

RM Rotatory Motor. 27, 38, 39

ROM Read Only Memory. 17

RS Right Switch. 28

SM Slider Motor. 39

SPI Serial Peripheral Interface. 13

UART Universal Asynchronous Receiver Transmitter. 13

USB Universal Serial Bus. 15

VLC Visible Light Communication. 14

Wi-Fi Wireless Fidelity. 3, 13, 14

Glossary

EEPROM permanent memory on the chip. 17, 18, 24, 37

IEEE Institute of Electrical and Electronics Engineers. 9, 12–14

MAC Media Access Control sub layer of data link layer. 9, 10, 14

Operating System Infrastructure software component of a computer system. 12

Radio-Frequency Identification digital data is encoded in RFID tags and captured by reader using radio waves. 4, 5, 8, 9, 15, 26, 28, 30, 35, 36, 38, 39, 51

RIOT-OS Operating system for low end embedded IoT devices. iii, 5–7, 12, 13, 16, 24, 51, 53

CHAPTER 1

Introduction

Nowadays, due to the static factory layouts in manufacturing industries, the material flow for producing different products with several varieties is difficult. Transforming the layout from one layout to another layout is a bottleneck. Consequently, there is considerable scope for research in decentralized and distributed modular structure in factory layouts. The main research field is to build stand-alone factory modules with wireless communication capabilities.

Currently, Radio Frequency (RF) spectrum is used as a wireless communication medium in Industries. The main limitations of RF are interference, high latency, spectrum deficiency and identification of the factory layouts. To overcome these limitations, Li-Fi is a preferred communication technology because of its high bandwidth, speed, immune to interference from electromagnetic sources and Line of Sight (LOS) communication.

1.1 Motivation

In this era of automation, developing concepts related to Industry 4.0 in logistics and material flow is necessary. These concepts focus on decentralized, modular and material handling resources in factory planning. Existing static structure in factory planning laboratory (LogCentre) [1, 2] restricts the flexibility in changing the factory layouts. The problem with the current static structure, as shown in Figure 1.1 is the constraints of the possibilities to customize a production process, which is difficult and time-consuming. The controlling operations are carried out by the central Programmable Logic Controller (PLC). The increasing complexity in the classical centralized material flow control is impeding the development of control systems.

On the other hand, the increasing digitization of processes, development and integration of innovative planning tools leads a path to change the static to dynamic and centralized to decentralized structures. Another problem with static structure is communication medium, i.e., hard-wired, because of which there is an increase in cost, workforce and time. The wireless communications such as RF, Bluetooth, ZigBee, Wireless Fidelity (Wi-Fi), Li-Fi provide a solution for modular factory structures.

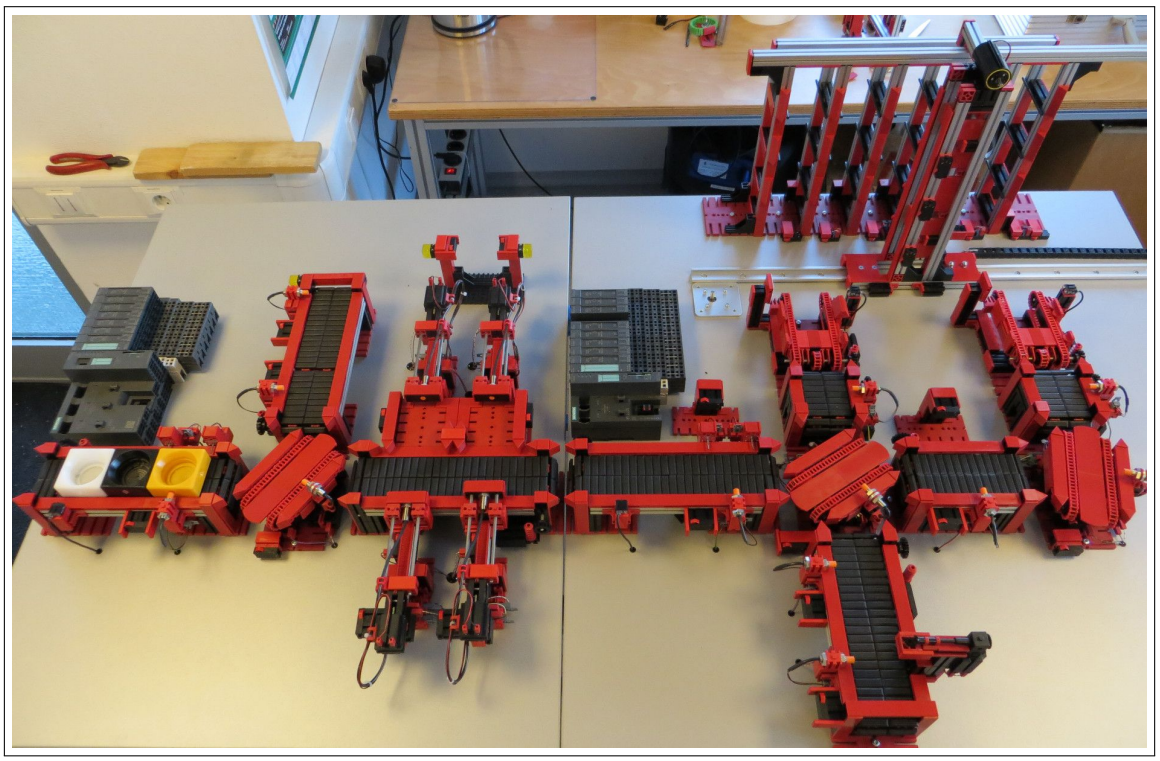


Figure 1.1: Current Factory Structure [1]

The main goal of the thesis is to implement a wireless communication protocol for distributed and decentralized factory planning applications. To summarize, the factory planning laboratory developed in this work shall be an environment to develop cyber-physical systems, modular, decentralized, controlled production and logistics concepts. It allows the researchers to develop, evaluate new concepts and technologies related to Industry 4.0.

1.2 Objectives

Based on the aim discussed in the previous paragraph, the objectives are as follows:

- Conceptual model
- Li-Fi protocol design
- Li-Fi transceiver design
- Modelling stand-alone factory modules
- Fabrication of component holders
- RFID setup
- Integration of the designed protocol in factory planning laboratory

Firstly, a conceptual model, as shown in figure 1.2, is designed to get a basic understanding of the aim of the thesis. For which a master-slave model is referred. In the conceptual model, the modular structures are formed by the master node. Initially, the RFID reader

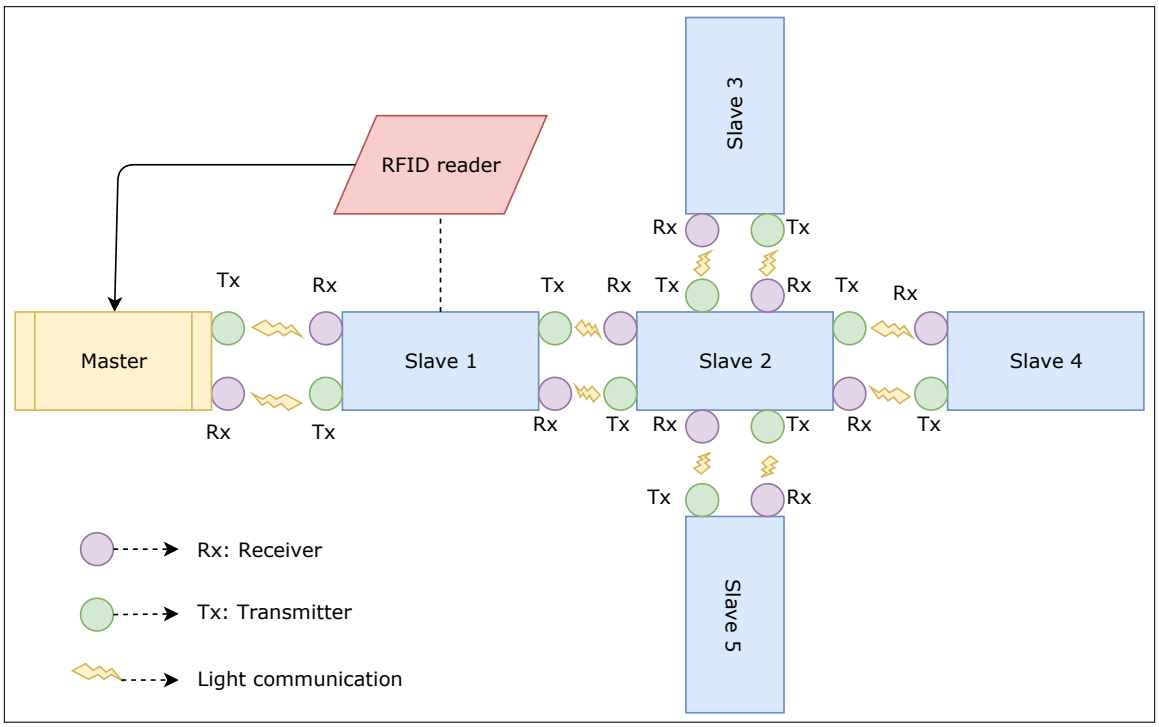


Figure 1.2: Conceptual Model

detects the sink information from the slave 1 and forwards it to the master node. The master node processes the information, communicates with slave 1 and forwards the information through different slaves to the destination. Next step is to design the Li-Fi protocol which has further sub-tasks such as frame format derivation, Manchester encoding and On-Off Keying (OOK) modulation, which is carried out in RIOT-OS firmware. At this point, the firmware is ready.

Now transceiver pairs are to be designed for different factory modules. Next step is to identify the best transceiver pair, finding maximum communication distance and reliability of the protocol. The Li-Fi transceivers are to be tested with the firmware flashed on Arduino-Mega 2560. Further, the factory modules redesigned into stand-alone modules. To attach the hardware components to the module, the component holders are designed and fabricated using 3D printing technology. Finally, to make the setup ready for evaluation, RFID based sink selection master node is to be designed. The detailed description of the hardware, firmware and technical specifications are illustrated in Chapter 3.

1.3 Thesis Structure

The structure of this thesis is illustrated in Figure 1.3.

Chapter 1 gives a brief introduction to the thesis. The motivation and objectives of the thesis are discussed.

In Chapter 2, the related work regarding current research about Li-Fi communication pro-

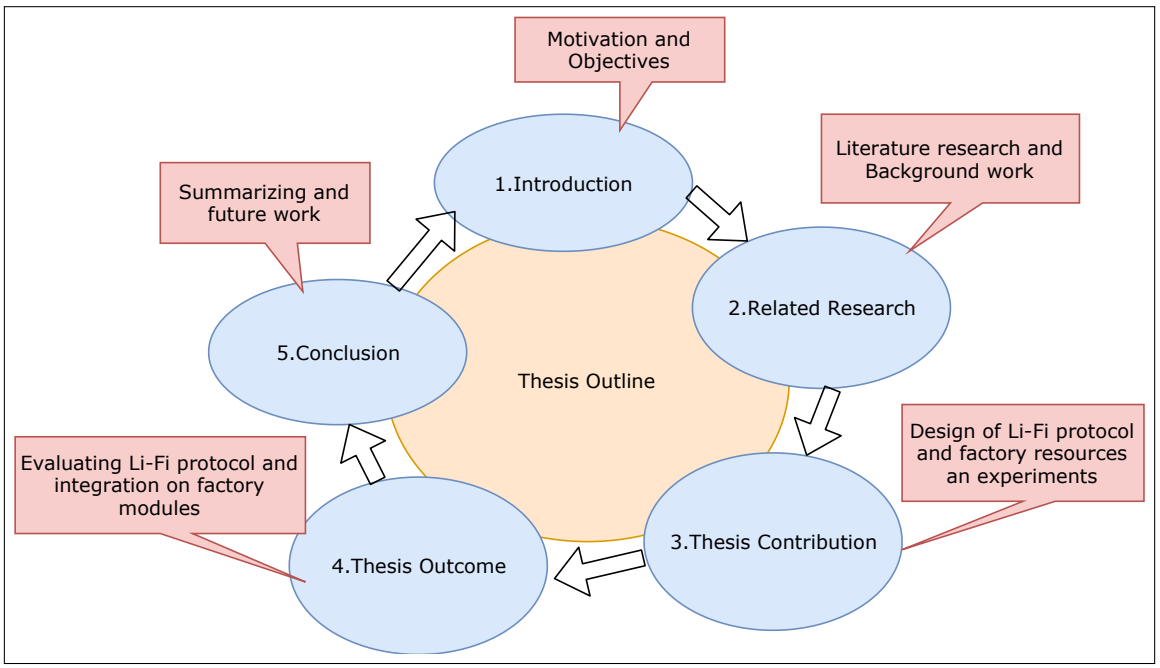


Figure 1.3: Thesis Outline

tocols, factory planning laboratory, the theoretical background of communication protocols and RIOT-OS are discussed.

Chapter 3 describes the software implementation of Li-Fi protocol, design of transceivers, factory modules, design and fabrication of the component holders using a 3D printer. It also deals with the experiments performed to check transmission speed, reliability and maximum communication distance of Li-Fi transceiver.

In Chapter 4, the evaluation of the protocol and integration of the Li-Fi protocol in factory planning laboratory is discussed.

In Chapter 5, summarizing the thesis and aspects of future work is mentioned.

CHAPTER 2

Related Work

2.1 Theoretical Background

In this section, the theoretical background of factory planning laboratory, architecture and modulation techniques of Li-Fi protocol, HDLC protocol, Manchester encoding and the RIOT-OS are described.

2.1.1 Industry 4.0 insights

According to four industrial revolutions so far, the first industrial revolution was the invention of energy generation methods using steam, water, etc., led to the industrial transformation with trains and improvisation of manufacturing. The second industrial revolution was the period where electricity was invented, which enhanced manufacturing with new inventions, such as the assembly lines. This revolution was directed to high productivity with some extent of automation. The third industrial revolution was obviously the birth of the Internet. Computerization, networking, robotics and connectivity, were the significant transformation in a way information is handled and shared with far more automation. Fourth industrial revolution (Industry 4.0) is a shift from the Internet and the client-server model to ubiquitous mobility with additional accelerators such as advanced robotics and artificial intelligence. This revolution enables automation and optimization with entirely new ways that lead to fully automated systems [3].

In general, the primary purpose of Industry 4.0 is the emergence of digital manufacturing, also named as “smart factory” [3], which means smart networking, mobility, Interoperability, flexibility in logistics and supply chain management. The design principles of Industry 4.0 are shown in Figure 2.1. Out of which the main focus of this work is to enhance Interoperability, decentralization and modularity. The modularity deals with the replacement and expansion of the modules and systems as per the changing need, flexibility and dynamic environment. Interoperability and interconnections mainly focus on standard protocols and Internet of Things (IoT). The decentralization and autonomous decisions contribute new innovative ideas in manufacturing and logistics industries [3].

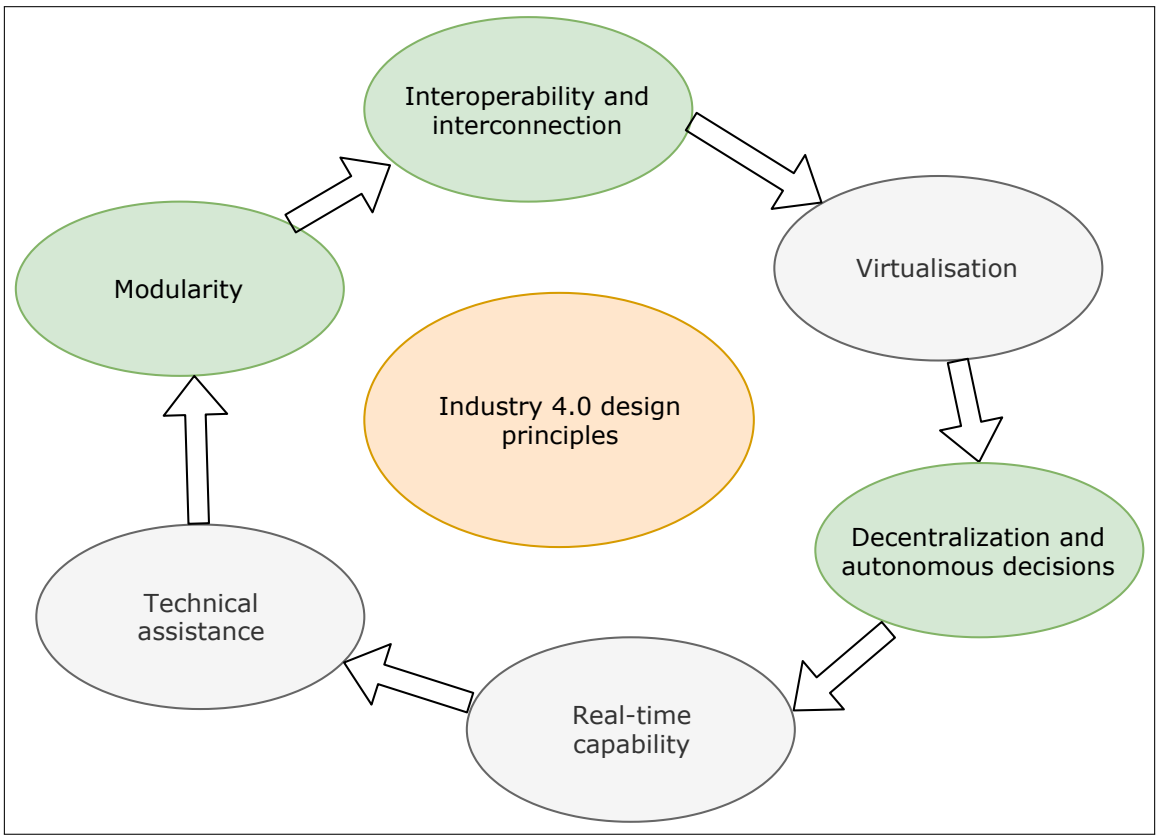


Figure 2.1: Industry 4.0 design principles [3]

According to Barreto et al., the implementation of Cyber-physical systems (CPS) increase the overall productivity, flexibility and agility in the supply chains. The CPS and its components, such as intelligent control systems and embedded software systems, play an essential role in the evolution of manufacturing [4].

2.1.2 Factory Planning Laboratory

Institute of Logistics and Material flow (ILM) department of Otto von Guericke University, Magdeburg, Germany in cooperation with Fraunhofer Magdeburg initiated the factory planning laboratory. As per Hofmann et al., the digitized planning and operating systems replaced with the term “digital factory” [1]. In the context of factory planning and operations, virtual commissioning placed before the physical commissioning [1]. The principal element of the laboratory is a modular, changeable plug-and-play conveying system that built on the idea of the “Flexconveyor” [5]. Modularity extends the flexibility to design and implement dynamic system structures.

Lang et al., introduced [2] modular plug-and-play conveying systems for distributed factory structures. Moor et al., [6] implemented the supervisory control for a flexible manufacturing system(FMS) using discrete event systems.

Currently, in the laboratory, each factory module is equipped with an RFID reader-writer

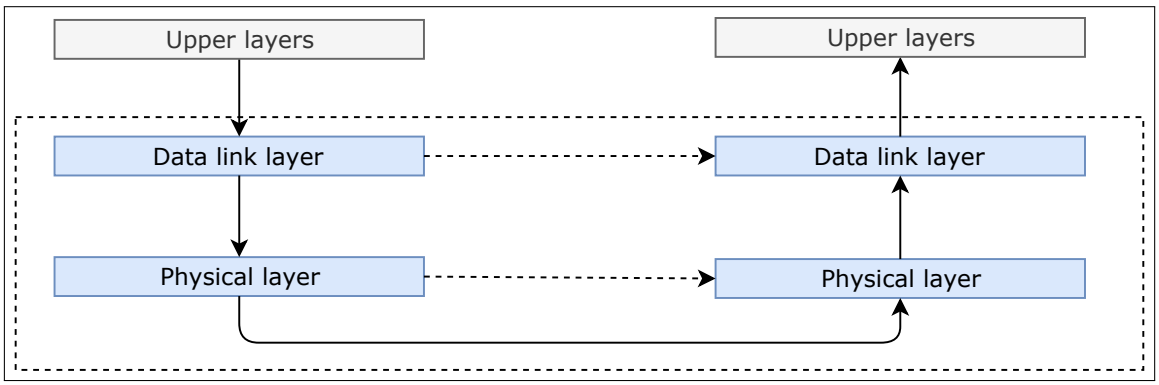


Figure 2.2: Layered architecture of Li-Fi [13]

and controlled by an Arduino-Mega 2560 single-board micro-controller. Each module programmed with a unique ID. The interconnection between the modules and to central bus system is with RS485 [7].

Mukku et al., [8] proposed Li-Fi communication for industry 4.0 learning laboratory. The proposed model helps in building modular factory layouts inside the laboratory.

In [9, 10], the RFID-enabled automation in support of factory integration, using RFID tags that carry Global Identification Code (GIC) information about module identity and connectivity data. The information on the GIC-tagged part can be directly accessed and retrieved from the designated host application. This increases flexibility and responsiveness of the value chains for manufacturing industries [9, 10].

Louw et al., introduced low-cost RFID track and trace system [11], RFID systems comprised of three components such as the tag or transponder, the reader, and the data processing subsystem comprising of middle-ware and internal databases. Tags are divided into two classes, namely active and passive tags. It has shown that using a small amount of data, for instance, the unique Id of the product achieves high passing speeds of up to $4.0\,\mathrm{m\,s^{-1}}$ with 100 successful readings [12].

2.1.3 Layered Architecture and Modulation Techniques for Li-Fi Communication

IEEE 802.15.7 is the standard for short-range optical wireless communications. This standard mainly focuses on the physical layer and MAC sublayer of Open Systems Interconnection (OSI) model [13]. The layered architecture of Li-Fi system is shown in Figure 2.2.

Physical layer: This layer defines the physical configurations of the devices. It specifies the relationship between the device and the physical medium. The physical layer establishes the communication between the transmitter and the receiver. The block diagram of the general physical layer implementation of the Li-Fi system is shown in Figure 2.3. The data frame received from the MAC sublayer is encoded with line encoding technique (e.g., Manchester encoding). The encoded data to proceed for modulation(e.g., OOK). Now the data is ready to transmit. The transmitter can use LED/LD (Laser Diode) for transmission. The data is

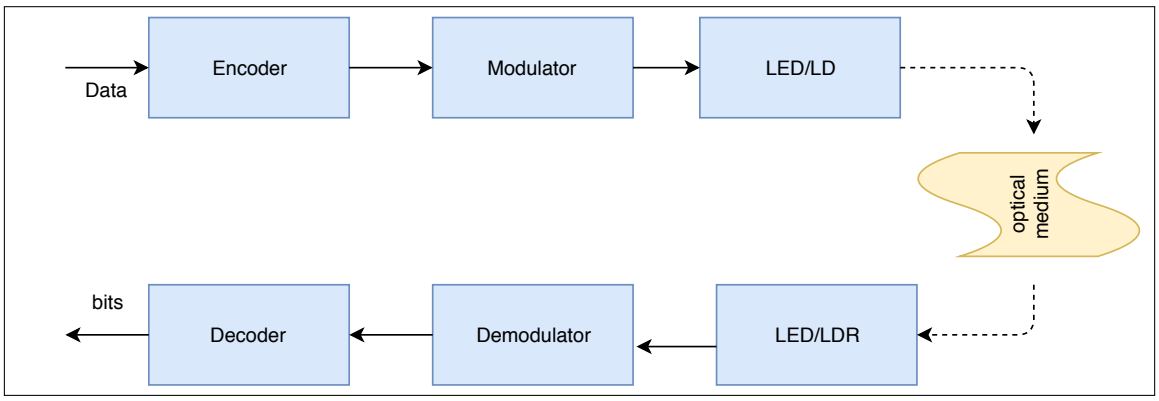


Figure 2.3: Physical layer implementation of Li-Fi system [14]

transmitted as a variation of light intensities. At the receiver end, photo detector or LED is used to receive the data. The receiver synchronizes with the transmitter to establish communication. Next, the data is processed with demodulation and decode with the same technique used at the transmitter [13].

MAC layer: The MAC layer provides error checking with Frame Check Sequence (FCS), addressing and channel access control mechanisms. The data received by the physical layer is processed by the MAC layer [13]. Tasks performed by the MAC layer are Mobility support, Dimming support, Visibility support, Security support, Schemes for mitigation of flickering. MAC layer supports peer-to-peer, broadcast, master-slave and star communication topologies [15].

Modulation Techniques: Li-Fi systems use the following modulation schemes such as OOK, Variable Pulse Position Modulation (VPPM), Colour Shift Keying (CSK), Sub-Carrier Inverse PPM (SCIPPM), SIM-OFDM (Sub-Carrier Index Modulation OFDM) [14]. In OOK, the LED is not switched off completely in the off state, but the reduction in the level of intensity is performed [16]. OOK modulation does not require prior knowledge of transmitter or receiver characteristics [13]. The main advantage of using OOK is its easy implementation and highly responsive with a white LED [16].

2.1.4 HDLC Protocol

HDLC protocol is used for transmitting synchronous data between Point-to-point nodes. It supports half-duplex as well as full-duplex communication [17]. In HDLC, the data is encoded into a frame, as shown in Figure 2.4.

The HDLC frame contains 6 data fields as follows:

- Flag Field: 8-bit delimiter `0b0111 1110` to identify the start and end of a frame. Also provides synchronization between transmitter and receiver.
- Address Field: One or more byte length data field used to represent the source and

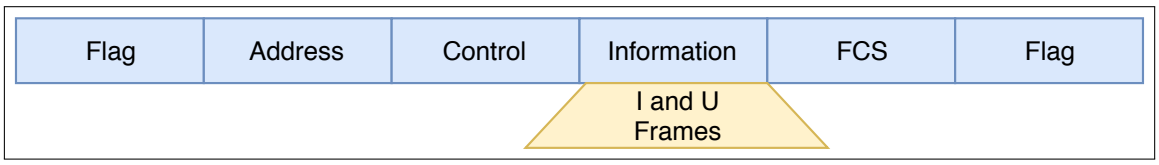


Figure 2.4: HDLC frame format

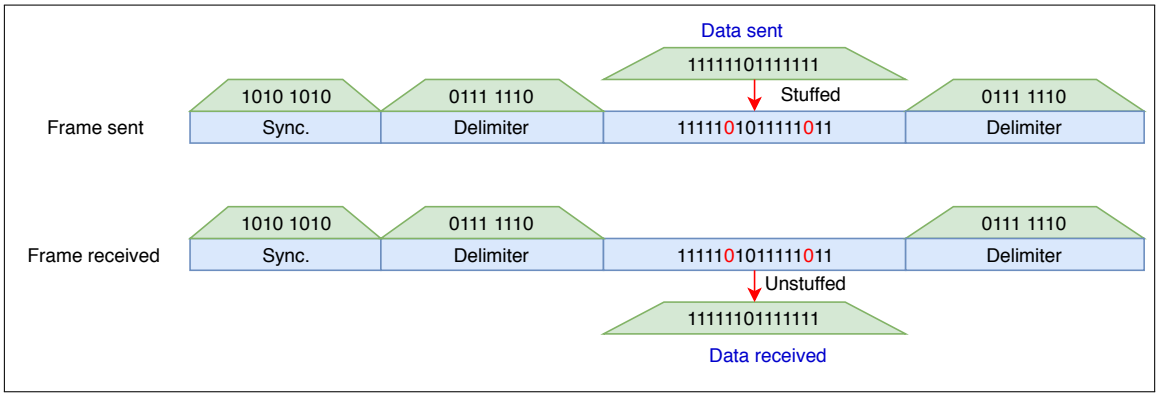


Figure 2.5: Bit-stuffing

destination address. If the transmitter creates the frame, it contains the destination address, and if the receiver creates the frame, it contains source address.

- Control Field: One or two-byte contains flow/error information.
- Information Field: Variable-length field contains user information from the network layer.
- FCS Field: It is an error detection field that contains 2 to 4 byte FCS data.

HDLC uses bit-stuffing to ensure that the bit pattern of the delimiter flag does not repeat in the fields between flags [17]. At the transmitter side, the data field in the frame is stuffed with zero after five ones in a row. The bit stuffed frame is transmitted from the physical layer. At the receiver end, the frame is unstuffed to decode the original data as shown in Figure 2.5. Error detection is carried out in the Data Link layer. At the transmitter side, FCS is calculated for the payload data and added to the HDLC frame. At the receiver end, the FCS is computed and compared with the received frame. If both are equal, then data received was successful. Otherwise, the message frame is to be transmitted again under control of an upper layer. Even though the Data Link layer implements error detection, it does not include a function to perform error recovery. There are three types of frames, such as information frame, supervisory frame, unnumbered frame.

Information Frame: This is also known as I-frame, which transports the user information from the network layer. Also, these fames contain flow and error control information.

Supervisory Frame: This is also named as S-frame. These frames carry flow and error control information and to send the empty frame. These frames do not have information

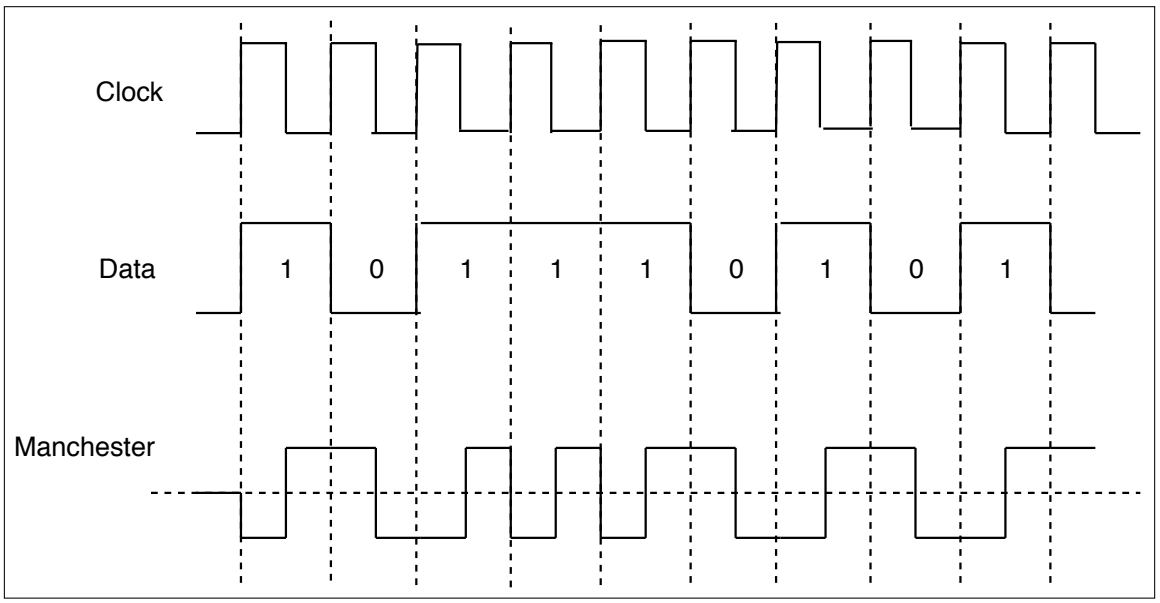


Figure 2.6: IEEE 802.3 standard Manchester encoding

fields.

Unnumbered Frame: This also known as U-frame, these are used for various miscellaneous purposes, including link management. Some U-frames contain an information field, depending on the type of frame.

2.1.5 Manchester Encoding

Manchester encoding is a line coding technique in which each bit represents a transition from low to high or high to low. It is also known as phase encoding. Manchester encoding provides self-clocking, which means the clock signal can be recovered from the encoded data. Because of self-clocking, this encoding technique is often used in many wireless communication technologies. The clock rate is directly proportional to the line voltage transitions. Thus, provides clock recovery. The IEEE 802.3 standard convention of Manchester encoding is shown in Figure 2.6. Each bit is encoded as one clock cycle. Binary **0** is a transition from high to low and **1** is a transition from low to high [18]. These transitions occur at clock midpoint. Initially, overhead bits are transmitted for synchronization.

2.1.6 RIOT-OS

According to Hahm and Baccelli [19, 20], IoT devices can be classified into two categories based on their capability and performance. High-end IoT devices, which includes single-board computers like the Raspberry Pi and Low-end IoT devices, which are very much resource-constrained to run traditional Operating System like Arduino. RIOT-OS is a modular architecture, built on a minimalistic kernel [21]. RIOT-OS is an open source

Operating System for low-end IoT devices [21]. RIOT-OS minimizes resource usage in terms of RAM, ROM and power consumption. It supports 8 bit to 32 bit Micro Controller Unit (MCU) and has developer-friendly Application Programmable Interface (API) which supports C++ and Arduino libraries [20]. The designed features of RIOT-OS are energy-efficient, small memory footprint, modularity and uniform API access for building complex IoT applications [22, 21].

RIOT-OS provides hardware abstraction with Unified APIs [20]. It has generic peripheral API to access peripherals of a microcontroller. It provides specific code for handling interrupts, system clock management, timers and peripheral drivers such as Universal Asynchronous Receiver Transmitter (UART) and Serial Peripheral Interface (SPI). In terms of board abstraction, a board in RIOT-OS is the practical design of real hardware. It provides the Central Processing Unit (CPU) configurations, drivers for components on the board and tools to interact. Driver model is a software module which can access and control the external hardware components connected to the CPU through RIOT-OS peripheral API. The sensor/actuator abstraction can be provided by a generic API for accessing sensors and actuators.

RIOT-OS allows developers to create multithreading applications. Multithreading is used for logical separation between multiple tasks, simple prioritization of tasks. The application can access the threads, based on thread priority. The distributed systems can be easily implemented by using the kernel message API [21]. The synchronization primitives such as mutex, semaphore and messaging (msg) are modelled as sub-modules of the kernel [20].

RIOT-OS allows the integration of third-party software and external libraries as packages. During the compilation time, `pkg` system uses a `Makefile` to build the application [20]. RIOT provides a Command Line Interpreter (CLI) which can be accessed over UART for debugging. RIOT-OS native is an essential hardware virtualizer that allows the compilation and execution of RIOT-OS applications [20].

The source code for RIOT-OS is available in the GitHub repository [23]. The complete documentation of RIOT-OS is available in RIOT Documentation [24].

2.2 Literature Review on Li-Fi Communication

In 2011, Harald Haas demonstrated Li-Fi communication at TED Global. Li-Fi was listed as one of the top 50 innovative technologies [25] in TIME Magazine 2011. Li-Fi is a high speed bi-directional fully connected, visible-light wireless communication system and is complementary to Wi-Fi, which uses radio frequency for communication. Li-Fi also lends support to the IoT. Li-Fi is an innovative technology that is composed to impact many industries. Li-Fi is a fundamental 5G technology. It can unlock the IoT, pilot Industry 4.0 applications, light-as-a-service (LaaS) in the lighting industry [26].

A speed up to 10 Gbps is obtained using Li-Fi, which is 250 times more than the speed of super-fast broadband [15]. The RF spectrum is only a fraction of the entire electromagnetic spectrum. The visible light spectrum and the Infrared (IR) spectrum are unregulated and offer 780 THz of bandwidth. The visible light spectrum extends from 380 to 780 nm in wavelength [26].

Li-Fi has entered in IEEE standardization and can be adopted for several specific industrial use cases beyond its current smart lighting scope [3]. IEEE 802.15.7 standard defines a Physical and MAC layer for short-range optical wireless communications using visible light as the communication medium [27].

According to Isik et al., in the industrial manufacturing process, the process must be completed with high speed and security apart from the product quality. Hence, deploying new communication strategies in the industries are evolving day by day.

In comparison, Li-Fi technologies preferred over Wi-Fi because of the high speed data transmission rate and security in wireless communication systems [28].

When Wi-Fi systems are replaced with Li-Fi, the receivers must be positioned in specific locations, where light from transmitter is uninterrupted. Thus, the line-of-sight problem is solved. Besides, it can be seen that the production process is faster by using Li-Fi. Therefore, internet connection can be provided without any interruption to the robotic arms or any other devices [28].

Kim et al., [29] proposed a scheme for device management and data transport in IoT networks using Visible Light Communication (VLC). In this concept, uni-directional transmission from the VLC transmitter is used to send the location-based VLC data. From the VLC receiver, the data is forwarded to aggregation agents and a central server in the network.

Mariappan et al., [30] proposed a concept of "Internet of Light"(IoL). Integrating IoT agent on IoL gateway and proposed different communications integrated with VLC such as VLC - PLC, RF - VLC and Ethernet - VLC link to create heterogeneity gateway for IoT devices.

Recently, according to Zhang et al., [31] research on VLC indoor positioning systems with simple system configurations. An indoor positioning system based on VLC was introduced, with no synchronization requirement on the transmitters.

Vinnarasi et al., proposed Li-Fi communication for transmitting audio, text, navigation using text to speech. One-way communication and bi-directional communication is tested with Arduino and IR, LEDs and photodiodes [32].

Chowdhury et al., explains about industrial communications such as device-to-device (D2D), machine- to-machine (M2M), chip-to-chip, device/machine-to-user, user-to-device/machine. These communications can be achieved using VLC technologies [33].

Schmid et al., proposed [34] LED as a photo detector to receive optical messages using the same LED that is used for transmission a setup, which reduces the complexity of the device. The capacitance of LED discharges at different speeds depending on the intensity of incoming light. The discharge speed is directly proportional to the intensity of light [34].

The maximum achievable data rates in Li-Fi are in the ascending order of phosphorous coated LED, red, green and blue (RGB) LEDs, Gallium Nitride (GaN) micro LEDs and laser-based lighting [26].

To summarize, the existing concepts and current work regarding industry 4.0, factory planning laboratory and Li-Fi communications are discussed. After considering all the concepts as a background and ground truth, this thesis proposes the introduction of Li-Fi communication protocol for distributed structures in factory planning laboratory.

CHAPTER 3

Thesis Contribution

This chapter describes the software and hardware implementation of the Li-Fi transceiver and design of the factory modules. The experiments are conducted to identify a suitable Li-Fi transceiver pair, in the context of reliability and maximum communication distance. The integration of Li-Fi communication on the factory modules was performed with two different prototypes.

3.1 Implementation

This section discusses the design choices, requirements, software architecture and implementation of the Li-Fi protocol.

3.1.1 Hardware Setup

An Arduino-Mega 2560 is used as a development board because it is inexpensive, powerful tool to interface with the sensors and supports different development platforms. The Arduino-Mega 2560 has 16 MHz crystal oscillator which is used for the controller clock. The Arduino has both analog and digital General Purpose Input and Output (GPIO) pins and an inbuilt Analog to Digital Converter (ADC). Universal Serial Bus (USB) interface is used to program the Arduino board. For complete details of the Arduino-Mega 2560 controller [35]. The components required to build Li-Fi transceivers and distributed factory resources are given below:

- Controller: Arduino-Mega 2560
- Transmitter: LED
- Receiver: LED, LDR
- Sink selection: RFID reader and tags
- Factory resources: Conveyors, turntable and slider
- Power source: 9 V battery
- Resistors: 270 Ω, 2 MΩ

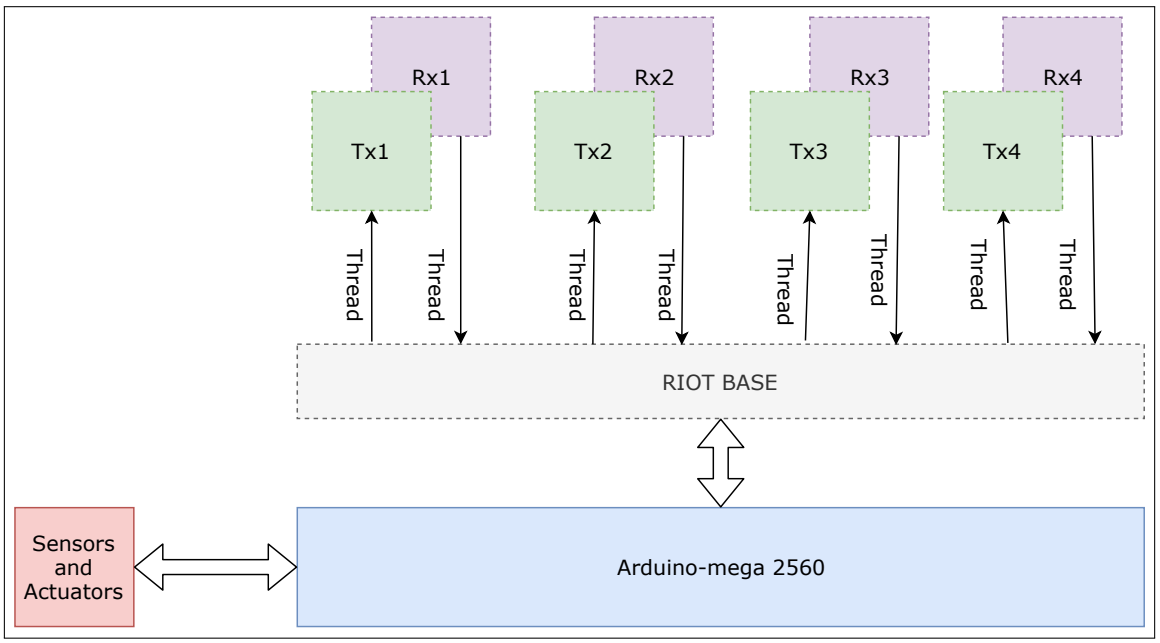


Figure 3.1: Software architecture

- Miscellaneous components: Relay module, connecting wires

3.1.2 Software Tools

RIOT-OS is used as a development platform. RIOT-OS provides a variety of functions, as discussed in Section 2.1.6, to implement the Li-Fi protocol. The operating system provides the multi-threading functionality for designing the multiple transceivers for each node. The software architecture is shown in Figure 3.1. The sensors and actuators are connected to the Arduino and control logics are programmed on RIOT-OS base. The design of a Li-Fi transceiver is explained in the following sections.

The circuit schematics are designed by using Fritzing software tool [36]. The timing diagrams are retrieved by using Pulse view tool [37]. The component holders are shown in Figure 3.15 is designed using an online 3d modelling design tool known as tinkercad [38] and printed using Arduino Materia101 3d printer. The architectures, control algorithms and frame formats are sketched using **draw** online tool [39].

3.1.3 Protocol Design Requirements and Choices

The main design choices and requirements for the protocol are shown in Table 3.1.

The main requirement is to provide clock recovery during communication. This can be achieved by the Manchester encoding technique, which provides self-clocking and clock synchronisation. The alternative option is using two different color LEDs for the clock signal and data signal. But, if the receiver is LDR, which is sensitive to all frequencies of

Design Requirement	Design choice
Clock recovery	Manchester encoding
OOK Modulation	LED as transmitter
Error detection	`CRC-16-CCITT`
Multiple transceivers	Multithreading
Routing	Flooding
Device ID	EEPROM
Communication type	Full duplex
Topology	Point-to-point

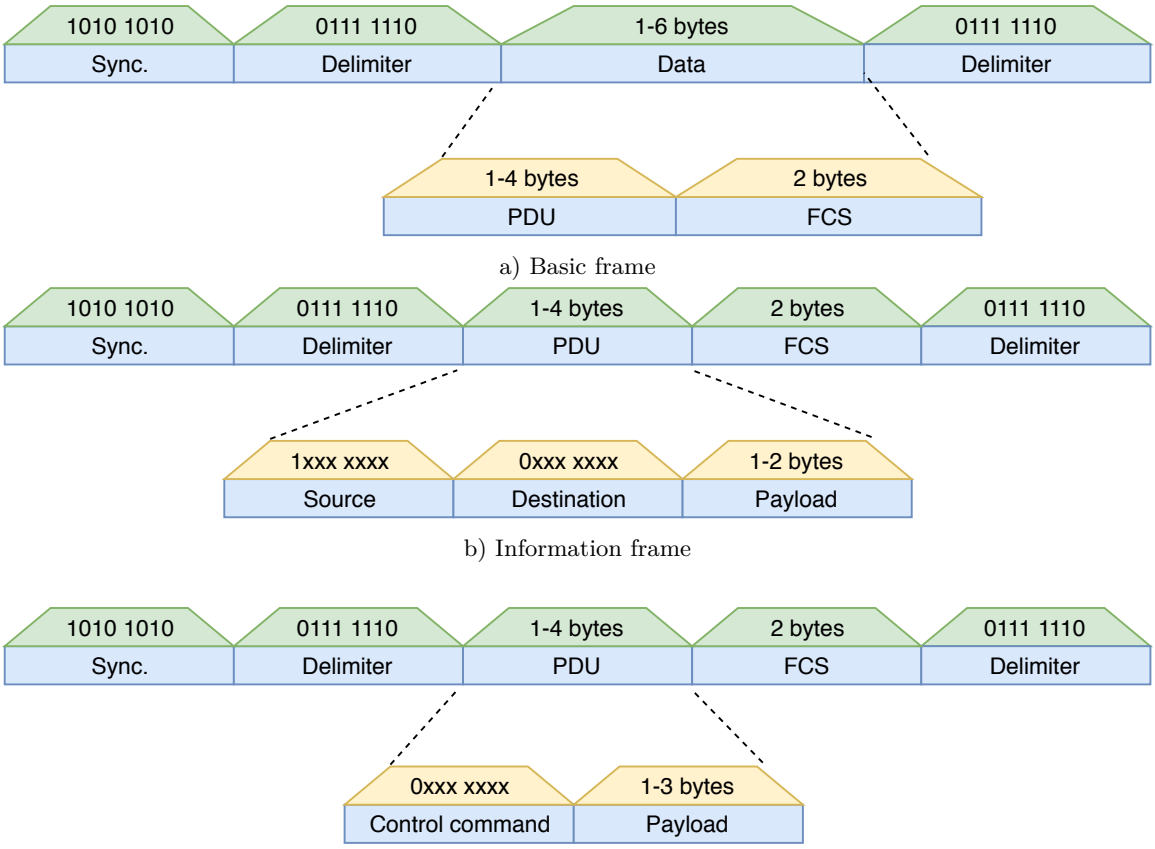
Table 3.1: Design requirements and choices

light and cannot differentiate between the clock and data light. So, the Manchester encoding is chosen as an encoding technique. As, LED is used for transmission, which cannot be used to shift frequency and phase. A modulation technique is required for transmission using LED. The OOK is a simple modulation technique for transmission of data using turning **on** and **off** the LED. The OOK is a binary Amplitude Shift Keying (ASK) technique, high amplitude represents a binary **1**, and low amplitude (not zero amplitude) represents binary **0**. The `CRC-16-CCITT` is chosen for error detection in the frame.

Each module in the factory has to communicate with the neighbouring nodes. So, a single node requires more than one transceiver interface. The full-duplex multiple transceiver interfaces are realised and accessed simultaneously using multi-threading functionality. It can be implemented by the event-based in a single thread, which increases the complexity in code. Since the factory planning laboratory is meant for demonstration of new technologies, implementing explainable and understandable code is more important than efficient code. The factory planning laboratory consists of many modules which indeed required a multi-hop communication. This thesis focuses on the implementation of Li-Fi protocol, not on routing, a simple and robust routing algorithm is chosen. Flooding is such an algorithm and requires little resources in term of Random Access Memory (RAM) and Read Only Memory (ROM). Each module in the factory should have a unique device ID. This can be achieved by storing the device ID on EEPROM of the Arduino controller. The data stored in EEPROM remains, even new program flashed in the controller. This device ID can be used as layer 2 address. Every module in the factory has to communicate with the neighbouring module. This can be realised by the point- point communication topology. Full duplex communication is chosen for bi-direction communication between the factory modules.

3.1.4 Frame Format of Li-Fi Protocol

The frame format for the Li-Fi protocol as shown in Figure 3.2 is derived from the frame format of HDLC protocols, as shown in Figure 2.4.



a) Basic frame

b) Information frame

c) Control frame

Figure 3.2: Frame formats

Basic Frame

The basic frame is shown in Figure 3.2a contains four mandatory fields, such as the sync., delimiter, data, delimiter. The frame synchronisation of the receiver and transmitter can be done using Sync. field. Delimiters can recognize the start and end of the frame. The delimiters contain `0b0111 1110` to identify the start and end of the frame. The data field contains the Protocol Data Unit (PDU) and the FCS. The FCS uses `CRC-16-CCITT` standard to calculate the checksum error of the PDU [40].

Information Frame

The information frame is shown in Figure 3.2b carries data from source to destination. It consists of the PDU field contains source address, a destination address, payload data and FCS from the datalink layer and the synchronisation byte and delimiters from the physical layer. The Information frame is identified by the source address, which always starts the Most Significant Bit (MSB) with 1 and the rest 7 bits are unique identification. The source and destination addresses are acts as the layer 2 address of the OSI model. The source and destination addresses are the device IDs stored in the EEPROM of the node controller. These device IDs are unique within the factory model.

```
Input: Basic Frame
Output: Manchester encoded data frame format
//set the clock to 30 ms;
//set the mid point to i.e. 30/2;
//set the ticks per clock to 10;
//send sync;
//send start delimiter;
//send data;
//send end delimiter ;
while check until input data is transmitted do
    if binary value = 1 then
        //Manchester 1: 01;
        LED Low (Logic 0);
        delay(mid point);
        LED High (Logic 1);
        delay(mid point);
    else
        //Manchester 0: 10;
        LED High (Logic 1);
        delay(mid point);
        LED Low (Logic 0);
        delay(mid point);
    end
end
```

Figure 3.3: Transmission Algorithm

Control Frame

The PDU of the control frame contains the control command, payload data and the FCS. The control frame shown in Figure 3.2c is identified by the control command, which always starts the MSB with 0 and followed 7 bits of the control command. It transmits data to control the actuators of adjacent nodes.

3.1.5 Software Implementation of Li-Fi Protocol

Basic Transmission Algorithm:

The data transmitted using the frame format, as shown in Figure 3.2a The transmission algorithm for the frame is illustrated in Algorithm 3.3. Each frame is encoded with Manchester encoding, as discussed in Section 2.1.5 and transmitted with OOK modulation described in Section 2.1.3. The transmitter uses the controller clock to transmit the data. Each binary **1** is encoded with clock transition from low to high and binary **0** is encoded as clock transition from high to low as shown in Figure 2.6. If the data field in the frame contains a bit pattern like `0b0111 1110`, then the receiver is confused. The algorithm to

avoid sending `0b0111 1110` is to stuff one 0 after every sequence of 5 ones in a row. This is known as bit-stuffing. At the receiver end, bit de-stuffing is carried to retrieve the actual information. More details regarding bit stuffing and de-stuffing are given in Section 2.1.4.

Basic Reception Algorithm

The algorithm for the reception of Li-Fi protocol is illustrated with the flowchart shown in Figure 3.4. LED, LDR is used as receivers and connected to the analog input of the Arduino. The LDR receives analog data based on the intensity of light received. The Arduino-Mega 2560 is used as a controller, which has inbuilt 10 bit ADC, it can detect 1024 discrete analog levels. The digital output value is calculated [41] using Equation 3.5.

$$R = \frac{V_{\text{ref}}}{(2^n) - 1} \tag{3.1}$$

$$V_{\text{ref}} = 5\,\text{V} \tag{3.2}$$

$$n = 10\,\text{bit} \tag{3.3}$$

$$R = 0.00489 \tag{3.4}$$

$$D_{\text{out}} = \frac{V_{\text{in}}}{R} \tag{3.5}$$

$$V_{\text{in}} = 0\,\text{V to } 5\,\text{V} \tag{3.6}$$

$$E.g : V_{\text{in}} = 2.5\,\text{V} \tag{3.7}$$

$$D_{\text{out}} = 511.24 \tag{3.8}$$

$\texttt{V}_{\texttt{ref}}$ is the reference voltage and, it is the maximum value that the ADC can convert, which is set to 5 V. $\texttt{V}_{\texttt{in}}$ is input analog value received at the analog pin. `R` is ADC resolution and `n` is 10 bit. $\texttt{D}_{\texttt{out}}$ is digital output.

The ADC has a maximum value of `1024` and a minimum value of `0`. The ADC provides digital output, which is proportional to the analog input value. The ADC clock (f_{ADC}) uses the controller clock, prescale factor and cycles per clock for conversion. The Arduino-Mega 2560 has 16 MHz crystal oscillator (f_{CPU}) and default prescale factor ($\text{n}_{\text{prescale}}$) of 128. Each sample conversion requires 13 cycles per clock ($\text{n}_{\text{cycles per sample}}$). The sampling of analog data depends on the sampling frequency (f) of the controller.

$$f_{\text{CPU}} = 16\,\text{MHz} \tag{3.9}$$

$$n_{\text{prescale}} = 128 \tag{3.10}$$

$$f_{\text{ADC}} = \frac{f_{\text{CPU}}}{n_{\text{prescale}}} \tag{3.11}$$

$$= 125\,\text{kHz} \tag{3.12}$$

$$n_{\text{cycles per sample}} = 13 \tag{3.13}$$

$$f = \frac{f_{\text{ADC}}}{n_{\text{cycles per sample}}} \tag{3.14}$$

$$= \frac{125\,\text{kHz}}{13} \tag{3.15}$$

$$\approx 9.6154\,\text{kHz} \tag{3.16}$$

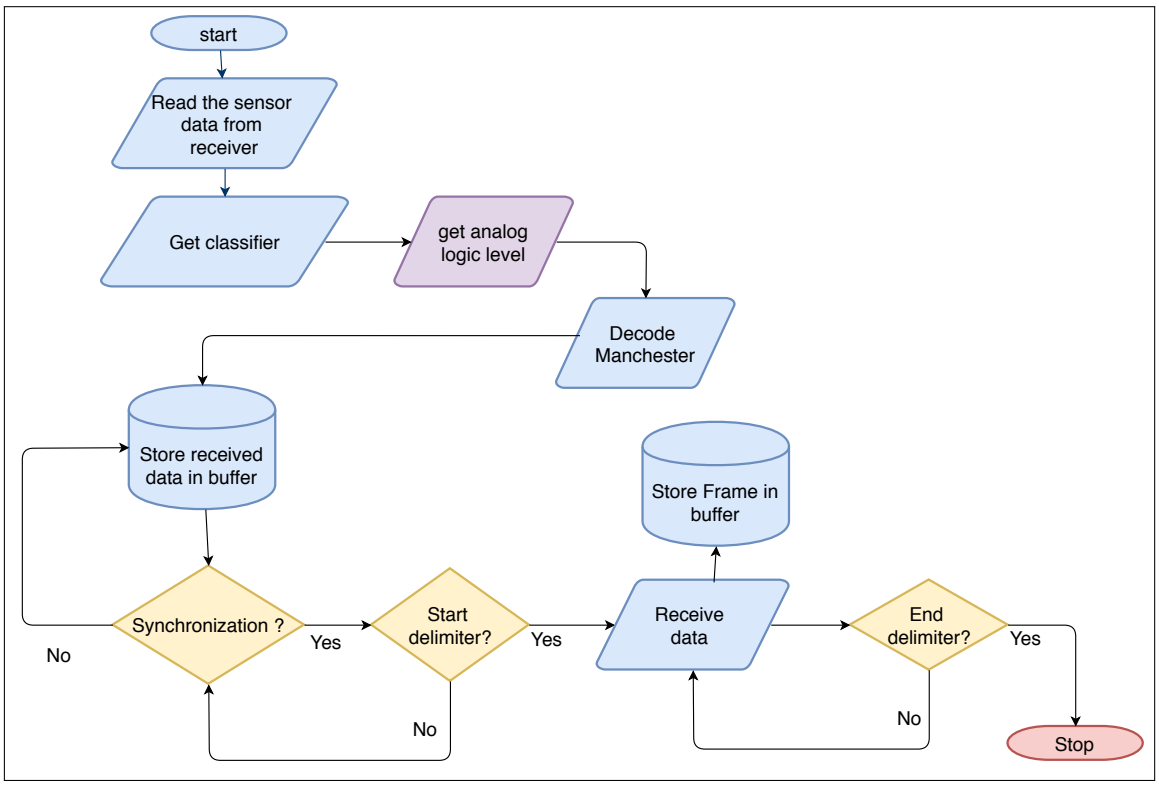


Figure 3.4: Basic reception algorithm

So, the sampling frequency with default parameters is set to 9.6 kHz. The classifier is calculated based on the average of the maximum and minimum ADC values. The classifier differentiates the received data in analog logic levels for decoding. The classifier value is considered to be the threshold value to convert discrete analog data into binary data. This binary data is decoded using Manchester coding, as shown in Figure 2.6. The Manchester coding provides clock synchronisation. If both transmitter and receiver are synchronised, then it checks for the delimiter and then receives the payload data until it gets an end delimiter. The sensitivity of the Li-Fi receiver is calibrated based on the ambient light.

Li-Fi Transceiver

The transceiver uses the basic transmission and reception algorithms. The parameters shown in Listing 3.1 are used for communication.

```
1 #define TICK 3//<-- number of milli seconds per tick (--> use for delay())
2 #define CLOCK_HALF 5 // <-- number of ticks per half clock
3 #define CLOCK (2 * CLOCK_HALF) // <-- number of ticks per clock (1 data bit)
4 #define GET_CLASSIFIER_TICKS (TICK * CLOCK * 3)
5 #define MINIMUM_HIGH_LOW_DIFFERENCE 50 // <-- used in get_classifier()
```

Listing 3.1: Parameters

The `CLOCK` cycle contains 10 `TICK`s and each `TICK` of 3 ms. In one `CLOCK` cycle, one bit of

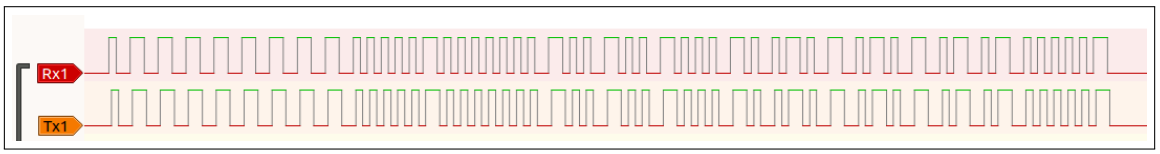


Figure 3.5: Full duplex communication

data is transmitted. The classifier is calibrated using the `MINIMUM HIGH LOW DIFFERENCE`, which is set to 50.

Transceiver Functions: The functions shown in the Listing 3.2 are used to design the Li-Fi transceiver. The main transmission and reception functions are defined in the `LiFi` class. The public function `send_data()` initiates the transmission. The private functions such as `send_sync send_delimiter send_byte` are responsible for creating the frame of data for transmission. The encoding is carried by `man_one()` and `man_zero()`. The public function `receive()` is responsible for the reception of data. The private functions such as `get_classifier()`, `sync_clock()` and `get_delimiter()` are used for identifying incoming data. The `get_bit()` function is used for decoding the Manchester data.

```
1  class LiFi {
2      // private functions
3    private:
4      //transmitter functions
5      void man_one();//<-- sending manchester one
6      void man_zero();//<--sending macherster zero
7      uint8_t send_byte(uint8_t b, uint8_t ones_in_a_row);//<-- sending one uint8_t of
           data
8      void send_sync();
9      void send_delimiter(void);
10     //receiver functions
11     int get_classifier();
12     int get_level();
13     int sync_clock();
14     int get_delimeter();
15     int get_bit();
16     int get_byte(uint8_t *dest);
17     int receive_frame(uint8_t *buf, uint8_t buf_size);
18     //public functions
19   public:
20     //transmitter fuctions
21     LiFi(int sPin, adc_t rPin);
22     void send_data(const uint8_t *data, uint8_t data_len);
23     //receiver functions
24     int receive(uint8_t *buf, uint8_t buf_size);
25 };
```

Listing 3.2: Transceiver functions

Multi-threading: The basic idea is to implement multiple transceivers with full-duplex communication, as shown in Figure 3.5 because each factory module has to send and receive data simultaneously. This can be achieved by applying multi-threading functionality.

The decision nodes in the factory planning laboratory have four neighbouring nodes, each

Figure 3.6: Flooding

node has to access all the neighbouring nodes. The node requires four Li-Fi transceiver interfaces to access the neighbouring nodes simultaneously. The multi-threading functionality allows creating multiple and concurrent threads for transceivers. Each transceiver is assigned with two threads, one for transmission and another one for reception. All threads access the same shared memory, as shown in Listing 3.3. The threads can access based on the thread priority.

```
1 static char tx_stack1[THREAD_STACKSIZE_DEFAULT];
2 static char rx_stack1[THREAD_STACKSIZE_DEFAULT];
3 static char tx_stack3[THREAD_STACKSIZE_DEFAULT];
4 static char rx_stack3[THREAD_STACKSIZE_DEFAULT];
5 static char tx_stack4[THREAD_STACKSIZE_DEFAULT];
6 static char rx_stack4[THREAD_STACKSIZE_DEFAULT];
7 static char tx_stack2[THREAD_STACKSIZE_DEFAULT];
8 static char rx_stack2[THREAD_STACKSIZE_DEFAULT];
```

Listing 3.3: Memory alloacation for threads

Each transceiver interface performs concurrent execution of transmission and reception of data. The transceiver interfaces are defined as shown in the listing 3.4.

```
1 PLiFi lifi1(4, 0, on_rx, NULL);
2 PLiFi lifi2(5, 1, on_rx, NULL);
3 PLiFi lifi3(6, 2, on_rx, NULL);
4 PLiFi lifi4(7, 3, on_rx, NULL);
```

Listing 3.4: Four Li-Fi interfaces

Flooding: The information frame is identified by the source address, which plays an important role in communication between modules. If the destination address is equal to the device ID, then, data is processed. If the destination address is not equal to the device ID, then the data is transmitted to the different Li-Fi interfaces connected to the module controller. Each module contains a maximum of four neighbouring modules. So, each module should have four transceiver interfaces. The data forwarded to all other interfaces except the interface which receives the data is known as flooding, which is shown in Figure 3.6.

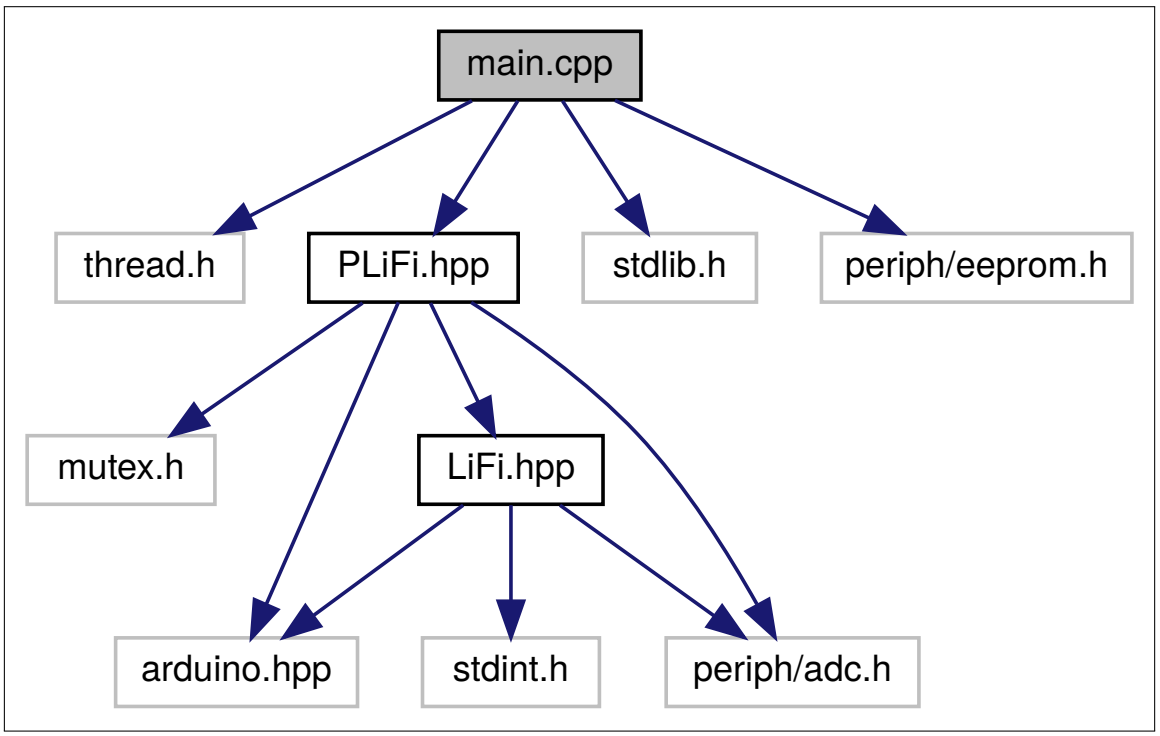


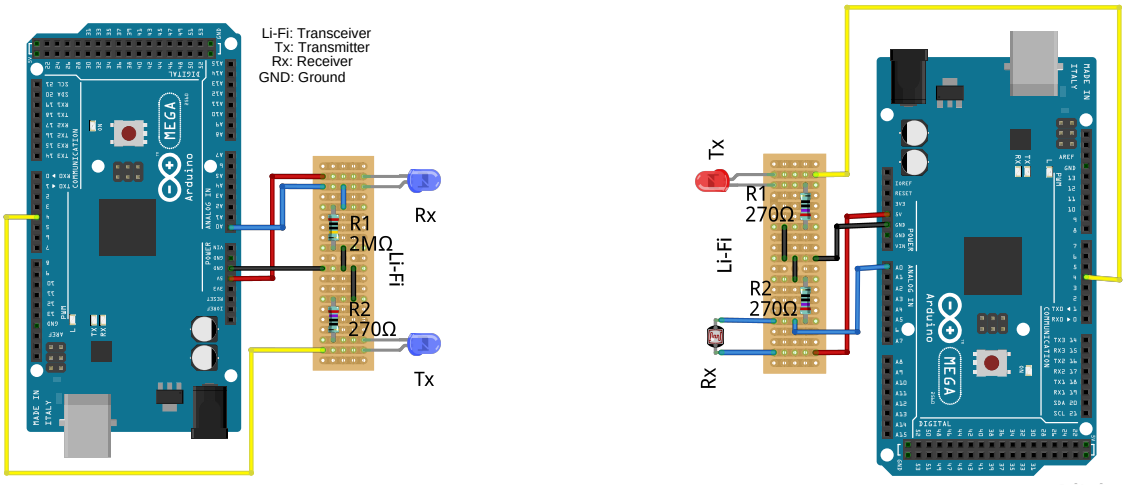
Figure 3.7: Collaboration diagram

Collaboration Diagram of Protocol:

The inbuilt libraries and the classes written for the Li-Fi communication are explained with the collaboration diagram, as shown in Figure 3.7. The dependencies of different libraries are mentioned; the `main` thread is the default thread. It accesses the other default libraries to perform necessary actions. The functionality of the Li-Fi protocol is implemented in the `LiFi` class. The transmission and reception logic is programmed in `LiFi` class. The Parallel LiFi (PLiFi) class access the `LiFi` class to perform parallel execution of the transceiver functionality. The `main` thread calls PLiFi class to perform any communication task, the PLiFi calls `LiFi` class to execute the task. The PLiFi class creates instances of `lifi` interfaces for transmission and reception. The PLiFi library uses `mutex` header file to perform the multi-threading functionality to execute the threads incorrect order and avoids critical thread execution conditions. The RIOT-OS defined `periph/adc` to perform ADC operations and `periph/eeprom` is used to perform the reading and writing of EEPROM of the Arduino-Mega 2560 controller.

3.1.6 Hardware Design of Li-Fi Transceiver:

The hardware design of Li-Fi transceiver can be designed in two different variations. The Arduino-Mega 2560 is used as the controller, and blue color LED is used as transmitter and receiver. A 270 Ω resistor is used at the transmitter side, and 2 MΩ resistors are used at the receiver end. The LED based Li-Fi transceiver, shown in Figure 3.8a. The LDR



a) LED based Li-Fi transceiver

b) LDR based Li-Fi transceiver

based Li-Fi transceiver is shown in Figure 3.8b. LED as transmitter and LDR as the receiver. 270 Ω resistors are used at the transmitter and receiver end. The transmitter is connected to GPIO digital pin `D4` and receiver is connected to GPIO analog pin `A0` of Arduino-Mega 2560 controller. In the factory planning laboratory, some resources have four neighbouring modules. A four-way Li-Fi transceiver is attached to the factory module to make the laboratory as dynamic, as shown in Figure 3.9. The four-way Li-Fi transceiver is achieved using multi transceivers in Section 3.1.5. The Arduino controller is capable of handling the four transceiver interfaces. The receivers are connected to the analog GPIO pins from `A0 to A3`. The transmitters are connected to the digital GPIO pins from `D4 to D7`. These four-way transceivers are used for turntable shown in Figure 3.10b and slider shown in Figure 3.10c.

3.2 Design of Modules in Factory Planning Laboratory

3.2.1 Design of Factory Modules

The factory planning laboratory has different material flow resources such as conveyor tables, turntables, sliders, as shown in Figure 3.10. In any manufacturing industry, there is always a need for flexibility in the plant layout. If the factory resources have to stand-alone functionality, then it is easy to change the structure, and the layout is scalable. The factory modules designed in here are for stand-alone functionality. The factory modules have different nodes, such as master node, source node, decision node and sink nodes.

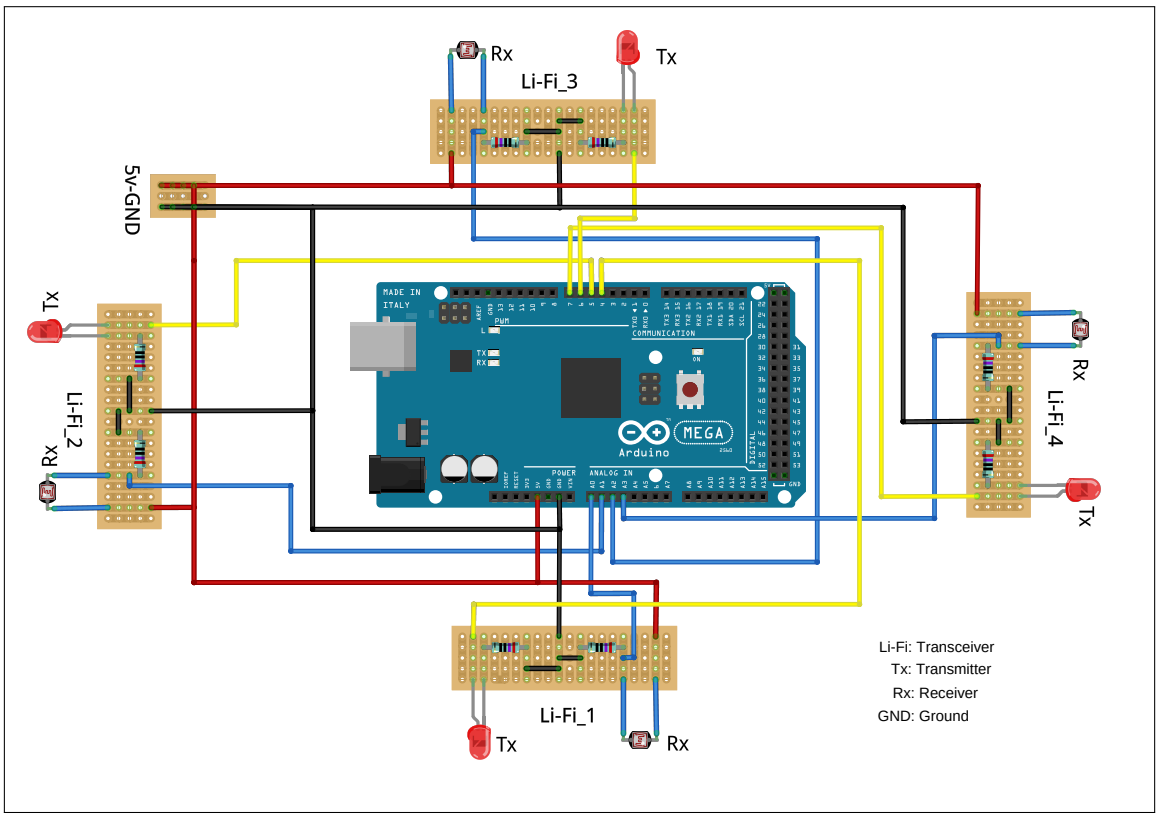


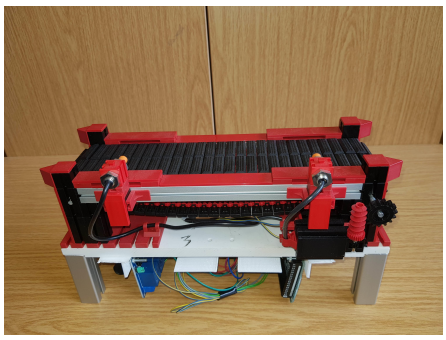
Figure 3.9: Four way Li-Fi Transceiver

Master Node

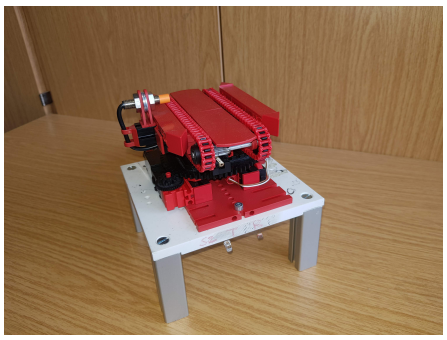
The master node consists of a controller, RFID reader, Li-Fi transceiver module and status LEDs. The master node is the root node and the remaining factory resources are placed around it and communicates with the master. RFID reader connected to the controller of the master node. The RFID tag with sink information is placed on the source node. The RFID reader detects the tag ID and processes the information and selects the sink. The master module provides the control flow of the product to the nodes in the factory layout. This module provides the status of product flow in the factory structure. The status LED glows based on the sink selection.

Conveyor Table

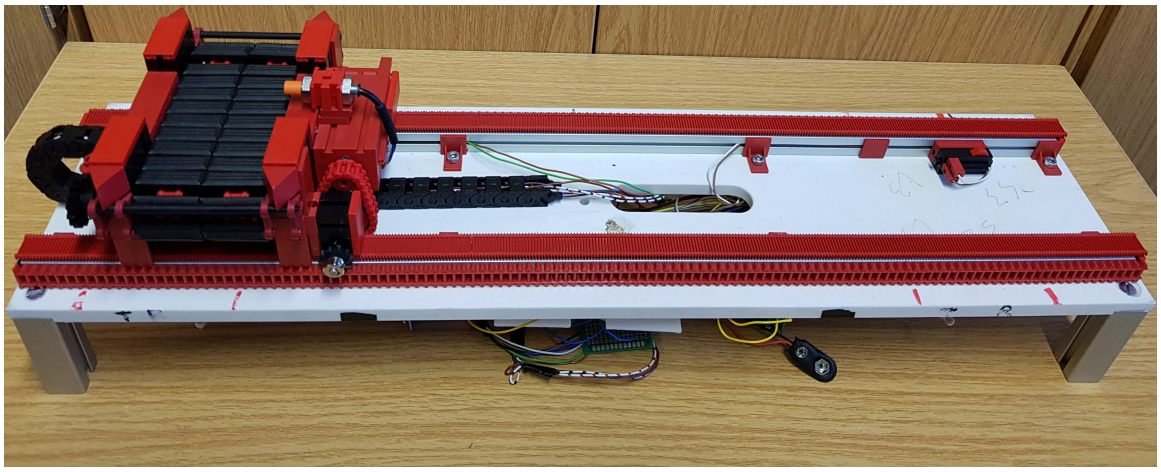
The conveyor tables are the basic building blocks of the material flow and logistics industry, as shown in Figure 3.10a. In this context, the conveyor tables are used as source and sink nodes. A source node contains two Li-Fi transceivers, one is directed to the master node, and another one is directed to the decision node such as turntable and sliders. A sink node contains one Li-Fi transceiver, which is directed to the decision node such as turntable and sliders. The connection diagram of the conveyor as a stand-alone factory module is as shown in Figure 3.12. The conveyor has two Proximity Sensor (PS) and one Conveyor



a) Conveyor table



b) Turntable



c) Slider

Figure 3.10: Factory resources

Motor (CM) for driving the conveyor forward and reverse. The PS as the sensor control the actions of the CM for logical transportation of products. The sensors and actuators are programmed based on the control strategy of individual processes. The CM is connected to the relay and 9 V battery. The Arduino board is used as node controller and is powered up with 9 V battery.

The conveyor has three states of operation. It can move forward, reverse and stop. The control logic for these states depends on the process flow.

Turntable and Slider

The turntable is the main building block of logistics shown in Figure 3.10b. This is an important resource to make the factory layouts scalable and flexible. The turntable is the decision node where it has more probabilities of control actions. The connection diagram for the turntable is shown in Figure 3.13. It has Arduino-Mega 2560 as node controller, four-way Li-Fi transceiver module connected to the controller, one CM to control the conveyor belt of the turntable, one PS to sense the product flow, one Rotatory Motor (RM) for the

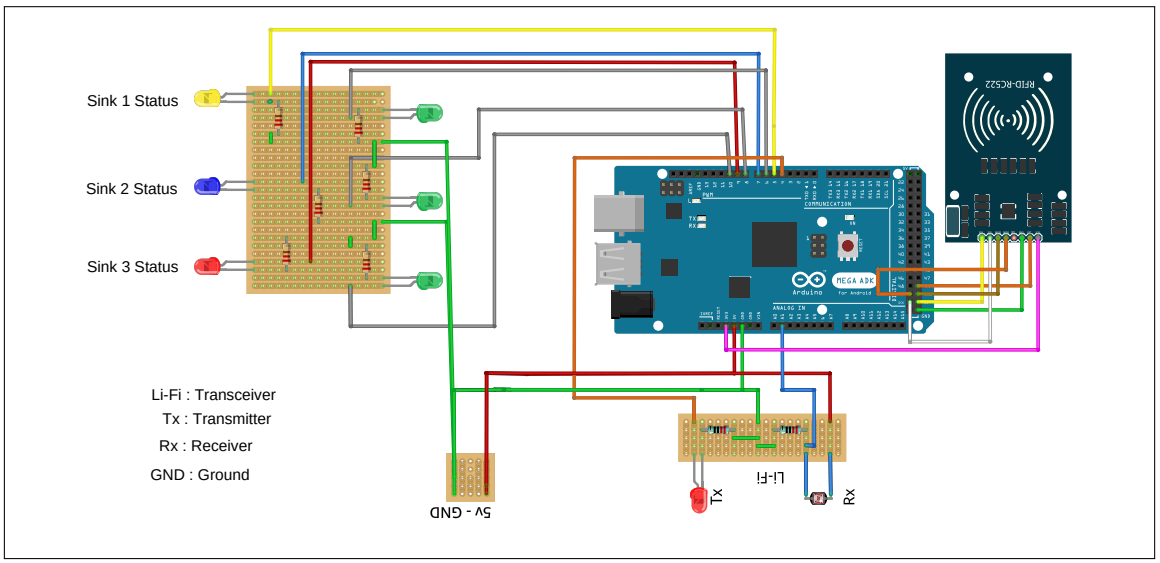


Figure 3.11: Master node

turning actions and two switches Left Switch (LS) and Right Switch (RS) to control the turning actions of the turntable. A relay module is used to control two motors and two 9 V batteries to power up the board and motors.

The slider has almost the same circuit connections, but the functionality is a bit different. It is also known as railing contact. This is also a major factory module which can change the factory structure. This can be used to add many streams of material flow by adding new structures to the slider. This consists of the Arduino controller and four-way Li-Fi transceiver module. The turntable and slider have different states of operation. Based on the process flow, these states are logically defined.

Design of Control Strategy

The control strategy for the prototypes that are discussed in Section 3.3.3 is explained with the control algorithm shown in Figure 3.14. The product flows through different intermediate nodes to reach the destination.

The control flow is illustrated as follows:

1. Sink information of the product is stored in the RFID tag or card
2. RFID module reads the information from the tag placed on the source node.
3. Master selects the sink based on the RFID information.
4. Based on the sink selected, status LED glows as discussed in the Section 3.2.1
5. The sink information is forwarded to the source node to activate the conveyor belt.
6. When the product reaches the second sensor of the conveyor belt, the source node forwards the sink information to the turntable/slider

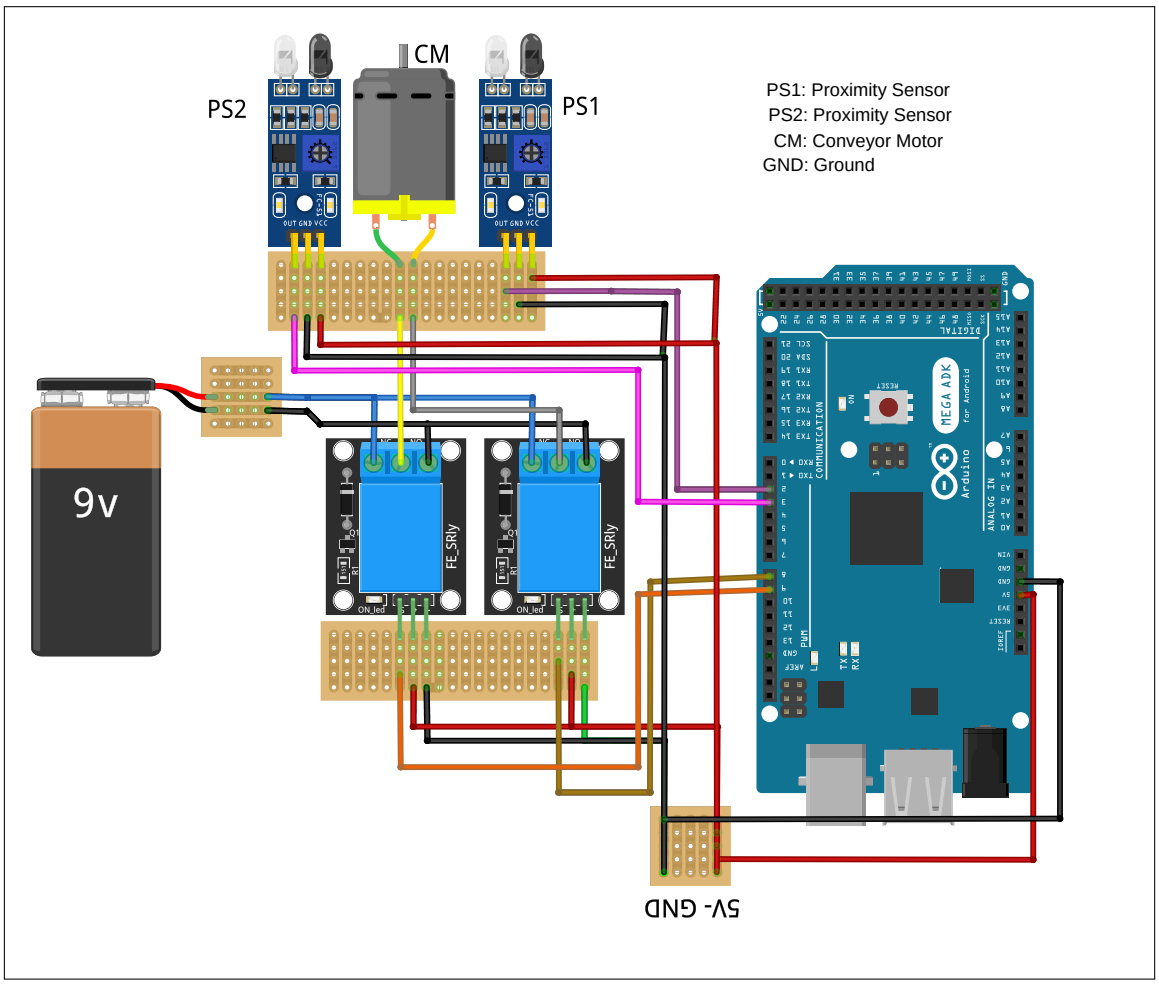


Figure 3.12: Connection diagram of conveyor table

7. Intermediate node activates the CM.
8. When a product reaches the sensor on the belt, the intermediate node forwards information to the particular sink.
9. The particular sink conveyor receives the information and activates CM.
10. When the product reaches the second sensor of the sink node, the CM stops.
11. The sink node sends acknowledgement information to the master using information frame 3.2b.
12. This data passes through the intermediate nodes to the master.
13. If the master receives the acknowledgement information, the corresponding green LED glows.
14. The status LEDs represent the status of the product from source to sink.

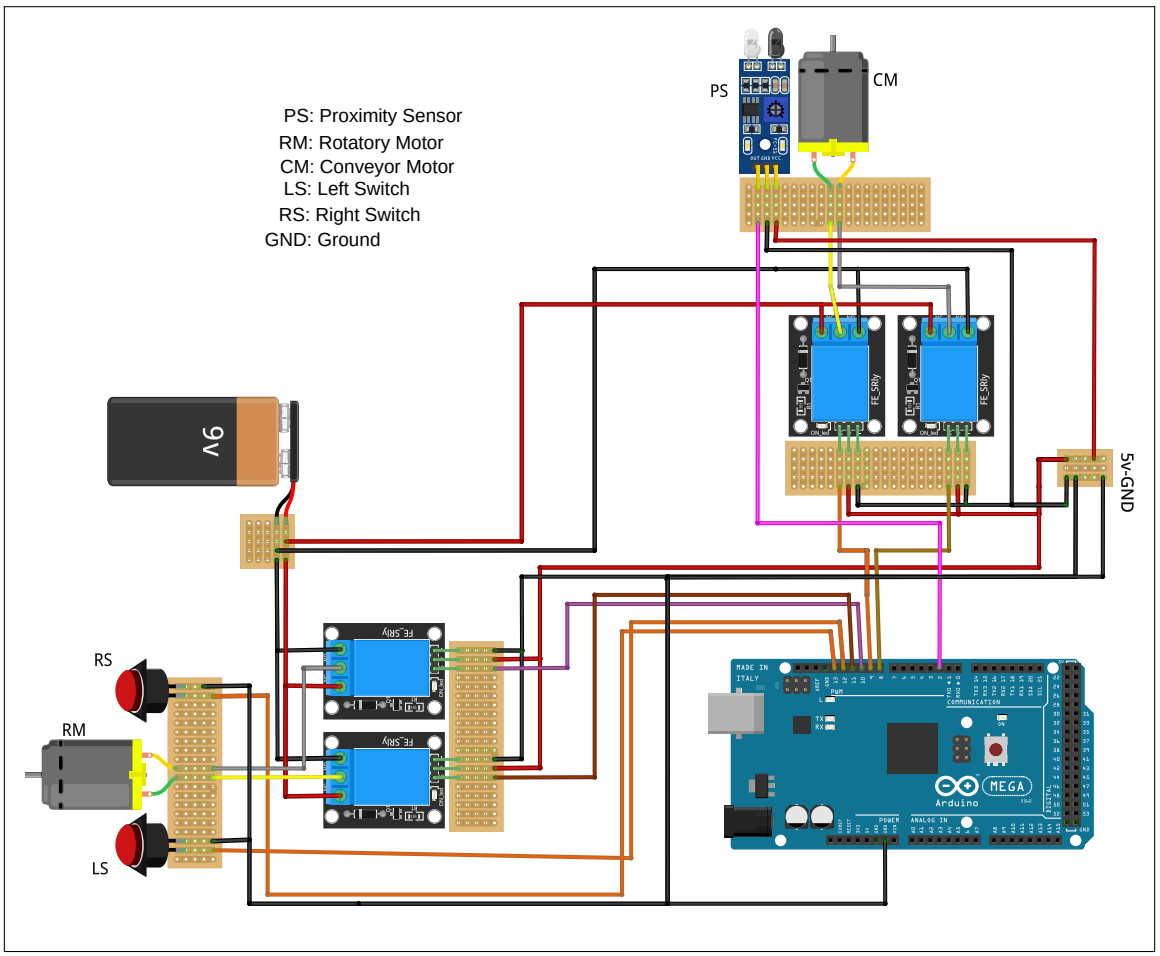


Figure 3.13: Connection diagram of turntable and slider

3d Design of Component Holder

All the components required to control the factory resource as stand-alone are integrated on the module. To attach the hardware components such as Arduino controller, relay module, RFID, Li-Fi transceiver module and batteries are requires component holders. The components holders are shown in Figure 3.15, modelled with an online 3d designing tool and fabricated using Arduino Materia101 3d printer.

3.3 Experiments

This section describes the experiments conducted to identify the protocol parameters, which give reliability, maximum communication distance. The integration of Li-Fi communication in factory planning laboratory is tested with two prototypes of distributed factory layouts.

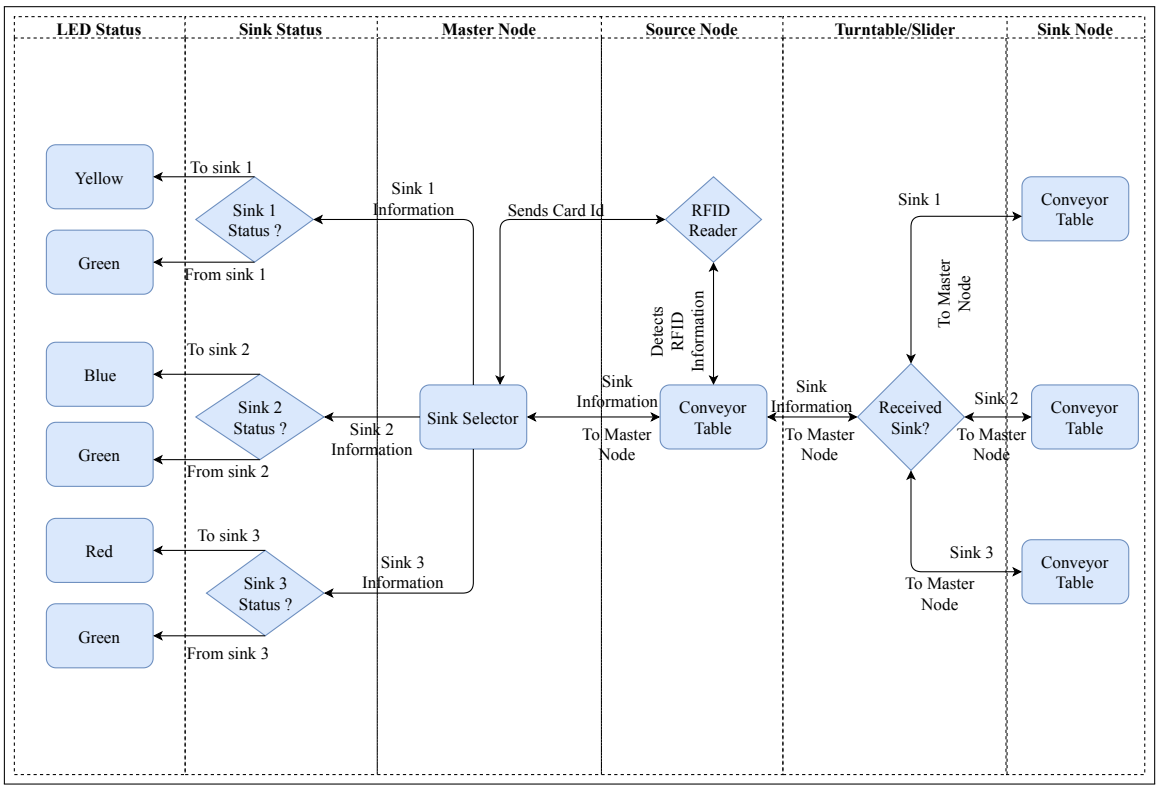


Figure 3.14: Control strategy for the factory prototype with turntable and slider

3.3.1 First Experiment: Maximum Communication Distance

Aim

The objectives of this experiment are to find the maximum communication distance and to identify a suitable transceiver pair.

Experimental Setup

LED as a receiver: The LED based Li-Fi transceiver shown in Figure 3.8a is used in this experiment. The experimental setup shown in Figure 3.16 is verified with different color LEDs as transceivers and with different `TICK` sizes. The parameters of Li-Fi protocol used for building the LED based transceiver are:

- Payload = 1 B
- `TICK` =3 ms and 4 ms // number of milliseconds per Tick
- `CLOCK_HALF` = 5 // number of ticks per half clock
- `MINIMUM_HIGH_LOW_DIFFERENCE` = 3 // to get the classifier at the receiver

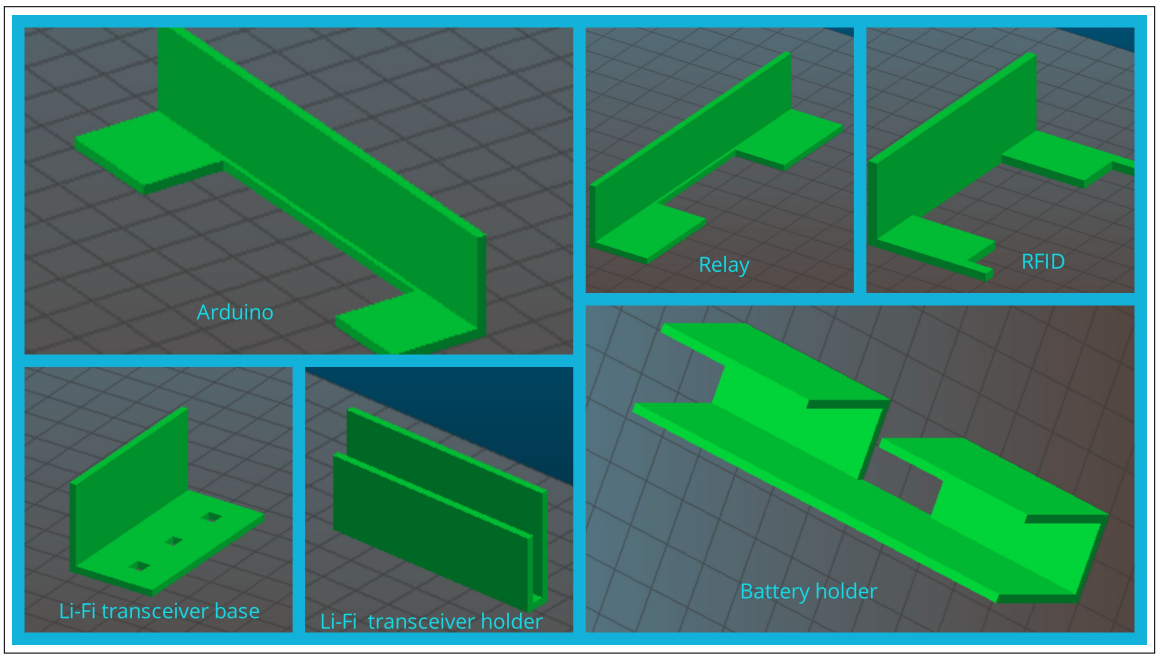


Figure 3.15: Component holders

LDR as a receiver: The LDR based Li-Fi transceiver is designed, as shown in Figure 3.8b. The experimental setup with LDR based transceiver is shown in Figure 3.17. This setup is tested with different color transmitting LED and LDR as the receiver. The hardware components required for this experimental setup are:

- Arduino-Mega 2560.
- LEDs: Red, Blue, Yellow, Green, White at the transmitter side.
- LDR at the receiver end.
- 270 Ω resistor at the transmitter side and 270 Ω resistor at the receiver end.

The parameters of Li-Fi protocol used for this experimental setup are:

- Payload = 1 B
- `TICK` = 3 ms and 4 ms // number of milliseconds per Tick
- `CLOCK_HALF` = 5 // number of ticks per half clock
- `MINIMUM_HIGH_LOW_DIFFERENCE` = 50 // to get the classifier at the receiver

Procedure

Procedure for LED as a receiver: In the first iteration, the communication distance is measured with `TICK` size of 3 ms. A series of iterations are performed to find the maximum communication distance. In the next instance, the `TICK` size is changed to 4 ms and continued with the same procedure to find the communication distance. The maximum communication is verified with different color LEDs as transceivers.

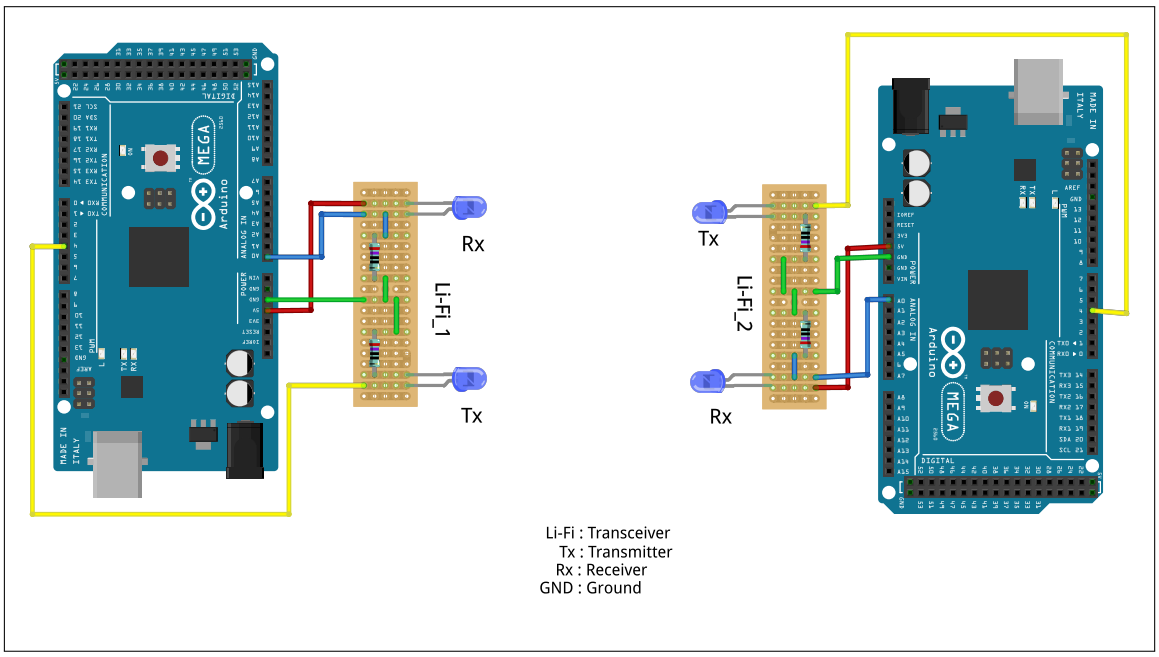


Figure 3.16: Li-Fi Transceiver : LED as a receiver

Color of LED	Communication Distance in cm with different `TICK` sizes 3 ms `TICK`	 4 ms `TICK`
RED	6 cm	17 cm
BLUE	8 cm	30 cm
YELLOW	3 cm	13 cm
GREEN	4.4 cm	23 cm
WHITE	30 cm	68 cm

Table 3.2: Maximum communication distance with different LEDs

Procedure for LDR as a receiver: Initially, a red LED is used as a transmitter and the LDR as a receiver. The communication distance is measured with a `TICK` size of 3 ms. A series of iterations performed to find the maximum communication distance. The same procedure is applied for different color transmitting LEDs such as Blue, Yellow, Green, White and LDR as a receiver to find the effective transceiver pair. In the next instance, the `TICK` size is changed to 4 ms and performed the same process to verify maximum communication distance. The experimental results listed in Table 3.2 are evaluated in Section 4.1.1.

3.3.2 Second Experiment: Reliability of the Li-Fi protocol

Aim

The aim of this experiment is to find the reliability of Li-Fi communication between two nodes, based on different communication distances, payloads and `TICK` sizes.

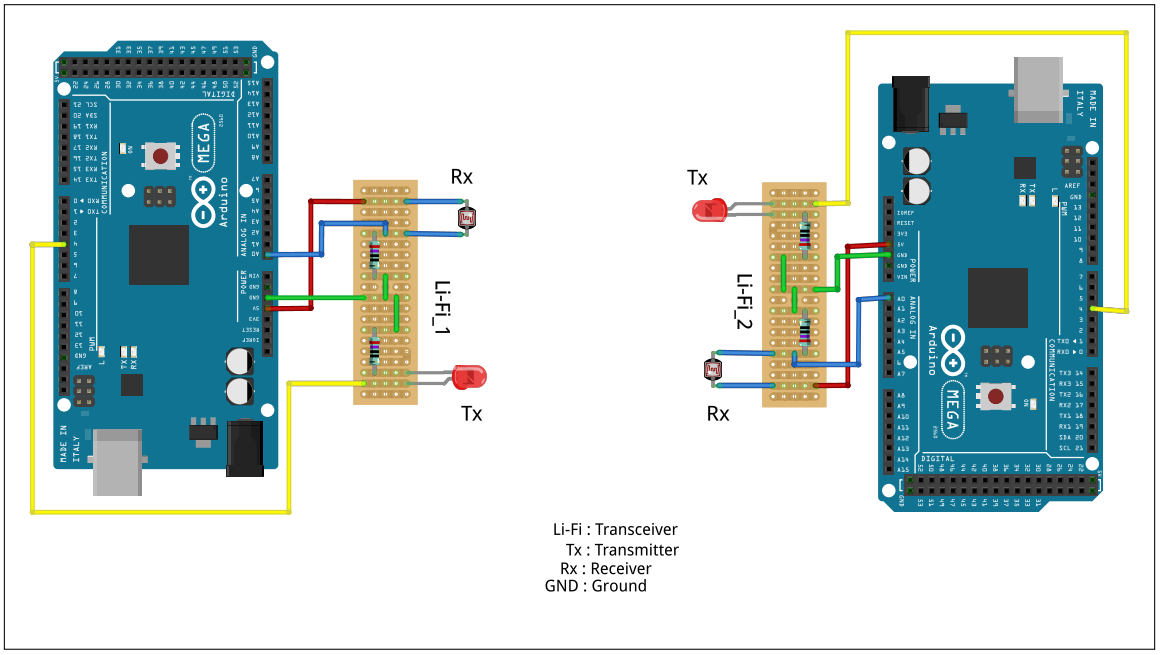


Figure 3.17: Li-Fi Transceiver : LDR as a receiver

Experimental Setup

The hardware setup for this experiment is shown in Figure 3.17. The reliability of protocol is verified with a white LED as a transmitter and LDR as a receiver. The frame is shown in Figure 3.2b is transmitted 100 times with different payloads and verified how many times the data is successfully received. The communication distance is varied from 5 cm to 20 cm. The protocol parameters are as follows:

- `TICK`=3 ms and 4 ms // number of milliseconds per Tick
- `CLOCK_HALF` = 5 // number of ticks per half clock
- `MINIMUM_HIGH_LOW_DIFFERENCE` = 50 // to get the classifier at the receiver

Procedure

Initially, both transceivers are placed at 5 cm communication distance with `TICK` size of 3 ms. The transceiver sends 1 B payload using the information frame illustrated in Figure 3.2b. The transceiver in the other end receives data. This transmission is repeated 100 times, and the second transceiver counts on successful data reception. This procedure applied for different payloads from 1 B to 5 B. In the next instance, the `TICK` size is changed to 4 ms and the communication distance is set to 5 cm. A series of iterations performed to compute the reliability of Li-Fi protocol with different parameters. The reliability of Li-Fi protocol with 3 ms `TICK` is listed in Table 3.3. The comparison of the reliability with 4 ms `TICK` is listed in Table 3.4. These results are evaluated in Section 4.1.1.

	Reliability with different communication distances			
Payload	5 cm	10 cm	15 cm	20 cm
1 B	99%	97%	96%	96%
2 B	99%	98%	96%	96 %
3 B	98%	96%	95%	94%
4 B	97%	96%	95%	93 %
5 B	97%	95%	93%	91%

Table 3.3: Reliability with 3 ms `TICK`

	Communication distance			
Payload	5 cm	10 cm	15 cm	20 cm
1 B	99%	98%	97%	95%
2 B	97%	96%	96%	95%
3 B	97%	96%	95%	95%
4 B	97%	96%	95%	94%
5 B	96%	95%	94%	93%

Table 3.4: Reliability with 4 ms `TICK`

3.3.3 Third Experiment: Integration of Li-Fi Communication in a Factory Planning Laboratory

Aim

The main objectives of this experiment are to test the Li-Fi protocol in the factory planning laboratory and finding the reliability for multi-hop communication.

Experimental Setup

This experiment is conducted with two prototypes built on factory modules. The layout of factory planning laboratory can be changed with the stand-alone factory resources. The construction of each module was discussed in Section 3.2.1. These prototypes are designed based on the conceptual model discussed in Section 1.1. The hardware required for the experimental setup of prototypes are:

- An Arduino-Mega 2560 for every module as a controller
- RFID reader and tags for sink selection
- Master with status LEDs
- Conveyor tables as source and sinks

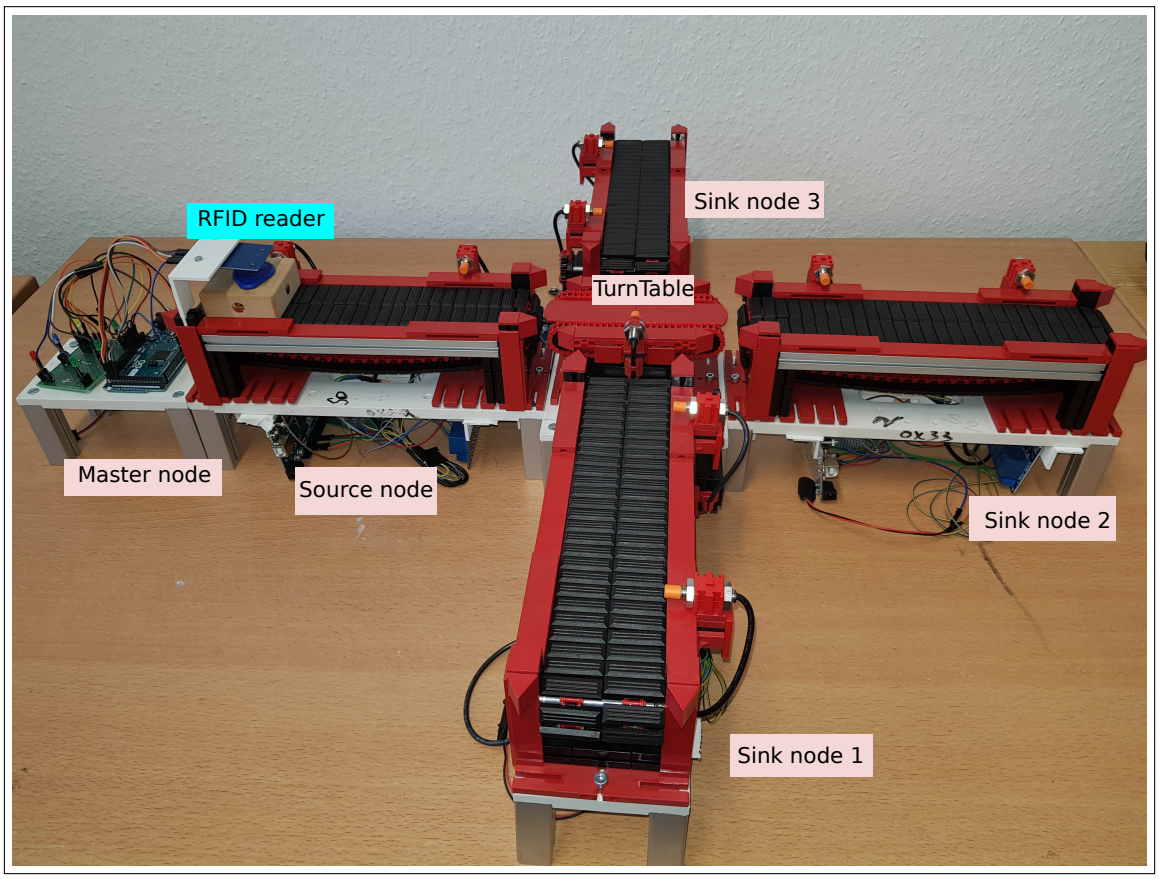


Figure 3.18: Factory prototype with turntable module

- Turntable, slider as the intermediate nodes
- Li-Fi Transceiver module
- Relay module
- 9 V batteries

The parameters used for Li-Fi protocol are:

- `Tick` = 3 ms and 4 ms // number of milliseconds per Tick
- `CLOCK_HALF` = 5 // number of ticks per half clock
- `CLOCK` = (2* `CLOCK_HALF`)
- `MINIMUM_HIGH_LOW_DIFFERENCE` = 50 // to get the classifier at the receiver
- White LED as transmitter and LDR as receiver for Li-Fi transceiver module

Prototype setup with the turntable: In this experimental setup a distributed structure is created with the turntable as a decision node. The master node with the RFID reader and source and three sink nodes. The setup is shown in Figure 3.18.

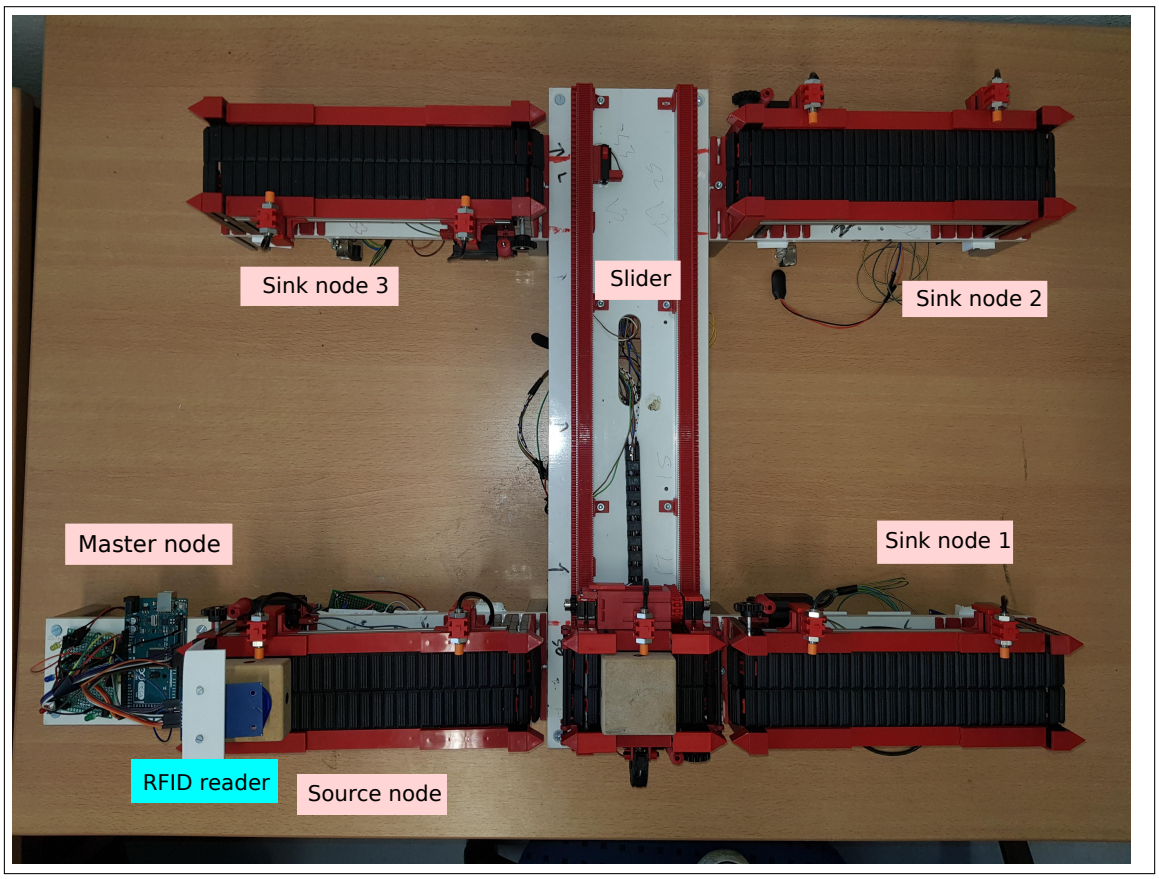


Figure 3.19: Factory prototype with slider module

Prototype setup with the slider: In this setup slider is used as decision node. The master node and source nodes are placed as shown in Figure 3.19. The sinks are placed at the exits of slider.

Data frames used in communication: There are two types of data frames, as discussed in Section 3.1.4 is used for communication between modules. The information frame 3.2b, which is used to send data from one node to a particular destination node. The control frame shown in Figure 3.2c sends the actuator control information to the neighbouring node. The device Ids are stored in the EEPROM of the Arduino controller. This device Id is used to recognize the type of module. Table 3.5 lists the type of factory modules with their device IDs. The device IDs are unique in the laboratory. The modules communicate with these device IDs. The control data and sink status data for different sinks is shown in Table 3.6. The control data drives the product from source to destination. The sink status data provides product status information to the master.

Module Type	Device ID
Master	0xFF
Source	0xAA
Slider	0xBB
Turntable	0xCC
Sink 1	0xA1
Sink 2	0xA2
Sink 3	0xA3

Table 3.5: Module type and device Id

Module type	Control data	Sink status
Sink 1	0xD1	0xF1
Sink 2	0xD2	0xF2
Sink 3	0xD3	0xF3

Table 3.6: Different types of information

Procedure

Communication reliability is tested for multi-hop communication. There are different module connected to each other to create a distributed layout. The information has to pass between different modules. These prototypes are evaluated by with reliability of multi-hop communication between the modules The designed prototypes follow the control strategy discussed in Section 3.2.1. The control and sink status information is listed in Table 3.6.

Prototype with Turntable The control strategy of this model is shown in Figure 3.14. The process flow to sink 1 is explained as follows:

1. The master node with the RFID reader detects the sink information from the RFID tag placed on the source node.
2. The master node process RFID tag information and selects the sink 1.
3. The master node forwards sink 1 control information as 0xD1 to the source node.
4. The send status LED on master node glows.
5. The source conveyor receives control frame from the master and CM gets activated.
6. When the product reaches at Proximity Sensor 2 (PS2), then source conveyor sends control information to the turntable module.
7. The CM of turntable gets activate and product moves forward to PS.
8. Then the Li-Fi interface of turntable, which is directed to sink 1 transmits control information.
9. Then RM rotates the turntable to the right and activates forward action of belt with CM to pass the product to sink1 node.

TICK size	Reliability of multi-hop communication (3 hops)
3 ms	95%
4 ms	92%

Table 3.7: Reliability of multi-hop communication with different TICK sizes

10. The sink 1 conveyor starts moving forward and the product reaches to PS2.
11. RM rotates the turntable to the default position.
12. This node generates sink status information 0xF1 and transmits the data to the master node with frame format described in Figure 3.2.
13. If the master node receives 0xF1, then the process flow is succeeded and sink 1 receive status LED glows.

This procedure is tested for 60 products to find the reliability of multi-hop communication with the TICK size of 3 ms. The experimental result is listed in Table 3.7

Prototype with the Slider This prototype also uses the control algorithm described in Figure 3.14. The process flow to sink 3 is explained as follows:

1. The product with sink 3 RFID information is placed on the source node.
2. The master receives information from the RFID reader and selects the sink.
3. The master node sends sink 3 control information to the source node as 0xD3.
4. The sink 3 send status LED glows at the master node.
5. The CM of the source gets to activate the conveyor belt.
6. The product moves forward and reaches to PS2.
7. The source node sends control information to the slider module.
8. The CM activates the conveyor belt on slider.
9. The product moves forward and reaches to PS2.
10. Li-Fi interface directed to sink3 transmits control information to sink 3 node.
11. TheSlider Motor (SM) activates to move the slider to the left and CM activates the reverse action of belt.
12. The sink 3 conveyor starts moving and the product reaches to PS2.
13. The SM moves right to the initial position.
14. This sink node generates sink status data 0xF1 and transmits the data to the master node with frame format described in Figure 3.2.
15. If the master receives 0xF3, then the process is succeeded and sink 3 receive status LED glows.

This procedure is repeated for 60 times to find the reliability of multi-hop Li-Fi communication with the TICK size of 4 ms. The result is listed in Table 3.7.

CHAPTER 4

Thesis Outcome

In this section, the experimental results and the parameters of the Li-Fi protocol are evaluated to identify the good transceiver pair and communication reliability. Integration of Li-Fi communication in factory planning laboratory is evaluated.

4.1 Evaluation

4.1.1 Discussion of Experimental Results

Maximum Communication Distance

The maximum communication experiment discussed in Section 3.3.1 provides the experimental results of both LED based Li-Fi transceiver and LDR based Li-Fi transceiver.

The LED based Li-Fi transceiver gives a maximum communication distance of 2 cm with 4 ms `TICK` and 1 cm with 3 ms `TICK`. This experiment also states that only blue color LED provides reliable communication, and other color LEDs failed to communicate. As, this experiment results in very smaller communication distances, one more experiment is conducted with LDR based Li-Fi transceiver. This results in higher communication distances, LDR is used as a receiver which is sensitive to different light intensities. So, different color LEDs used as transmitters and verified the maximum communication distances. The graph is shown in Figure 4.1 illustrates the comparison of maximum communication distance with different transmitting LEDs and `TICK` sizes. The protocol parameter `TICK` size also has a major impact on the maximum communication distance. As shown in Table 3.2, the white LED as a transmitter and LDR as a receiver gives a distance of 30 cm with a `TICK` size of 3 ms. The maximum communication distance achieved was 68 cm with a `TICK` size of 4 ms. From this experiment, it is clear that higher `TICK` size results in longer communication distances.

From the graph, the communication distances of other LEDs show that the white LED as a transmitter and LDR as a receiver can be used for the design of Li-Fi transceiver in the factory planning laboratory.

Reliability of the Li-Fi protocol

Following the first experiment, this experiment gives the reliability of the protocol, which plays a major role in communication between factory modules. The reliability is verified between two nodes with the rate of successful reception of data explained in Section 3.3.2. The comparison of reliability with different parameters are listed in Table 3.3 and Table 3.4. The graphs are plotted from the tables. The graph shown in Figure 4.2 illustrates the increase in payload results in loss of data. The reliability of 5 B payload is less compared to 5 B payload with a communication distance of 5 cm.

The graph shown in Figure 4.3 explains the increase in distance with constant payload decreases the reliability. The reliability of 3 B payload with different communication distances is explained in the graph. The graph illustrates the influence of `TICK` size on reliability. If the communication distance is more,then 4 ms `TICK` size gives slightly better reliability.

The graph is shown in Figure 4.4 provides the comparison data of reliability with different payloads, communication distances and `TICK` sizes. It is shown that the `TICK` size does not have any significant effect on reliability for smaller communication distances. The reliability is less with more payload and longer communication distances.

After carefully examining all graphs discussed in this section, it is concluded that 5 cm communication distance results in 96% to 99% of reliability with different payloads and `TICK` sizes. The communication distance effects the reliability, but the protocol provides slightly better reliability for longer communication distances with 4 ms `TICK` size. The distance between factory modules as shown in Figure 3.18 and Figure 3.19 is very small. The `TICK` size impacts more on communication distance not significantly on reliability. So, 3 ms `TICK` size can be used as `CLOCK` parameter for faster communication.

Integration of Li-Fi Communication in a Factory Planning Laboratory

From the first two experiments, the suitable Li-Fi transceiver is identified which provides longer communication distance and reliable communication. The white LED as a transmitter and LDR as a receiver are chosen for designing the Li-Fi transceiver in factory planning laboratory.

In this experiment, the Li-Fi protocol was integrated into factory planning laboratory. This experiment is conducted on two different prototypes and verified the reliability of multi-hop communication. The two prototypes require three hops to communicate from source to sink. The reliability is calculated with two `TICK` sizes. The reliability was calculated by sending the product from source to destination explained in Section 3.3.3 for 60 times. The results are listed in Table 3.7. This experiment differentiates the reliability with single-hop and multi-hop. If the reliability (r_1) is 99% for single-hop, the theoretical reliability (r_3), in

this case three hop is calculated as follows:

$$l = n - 1 \tag{4.1}$$
$$r_l = {r_1}^l \tag{4.2}$$
$$r_3 = 0.99^3 \tag{4.3}$$
$$\approx 0.97 \tag{4.4}$$

Here `l` is number of hops and `n` is number of nodes. The theoretical reliability for three hops is 97%. The practical reliability with the `TICK` size of 3 ms is 95%, which is slightly lesser then theoretical reliability. Similarly, the reliability with 4 ms is also lesser then the theoretical reliability. AS shown in Table 3.7, there is only small difference in reliability with different `TICK` sizes. So, lesser `TICK` size is chosen for faster communication. From this experiment, the Li-Fi communication is significantly reliable within the factory planning laboratory.

4.1.2 Performance Evaluation

Transmission Time

In this section, the performance of Li-Fi protocol is evaluated based on the transmission time of frame, throughput and Goodput. Initially, the transmission time is calculated with different parameters and information frame as shown in Figure 3.2b is used for transmission. The parameters of Li-Fi protocol to compute the time required to transmit 1 B of payload with overhead are as follows:

- `TICK(T)` = 3 ms and 4 ms and 5 ms // number of milliseconds per Tick
- `CLOCK_HALF(c) = 5` // number of ticks per half clock
- `CLOCK(C) = (2 * CLOCK_HALF)` //number of ticks per clock (1 bit of data)

The calculation of transmission time for 1 B payload is shown below:

$$T \stackrel{\wedge}{=} 3\,\text{ms} \tag{4.5}$$
$$c \stackrel{\wedge}{=} 5 \times 3\,\text{ms} \tag{4.6}$$
$$\stackrel{\wedge}{=} 15\,\text{ms} \tag{4.7}$$
$$C \stackrel{\wedge}{=} 2 \times c \tag{4.8}$$
$$\stackrel{\wedge}{=} 30\,\text{ms} \tag{4.9}$$
$$1\,\text{bit} \stackrel{\wedge}{=} 30\,\text{ms} \tag{4.10}$$
$$\Rightarrow 1\,\text{B} \stackrel{\wedge}{=} 8 \times 30\,\text{ms} \tag{4.11}$$
$$\stackrel{\wedge}{=} 240\,\text{ms} \tag{4.12}$$
$$8\,\text{B} \stackrel{\wedge}{=} 8 \times 240\,\text{ms} \tag{4.13}$$
$$\stackrel{\wedge}{=} 1920\,\text{ms} \tag{4.14}$$

The payload data is encoded in the information frame. The frame is composed of 8 B to transmit 1 B of payload data. It requires 1920 ms from Equation 4.14 to transmit 1 B

	Transmission time with different TICK sizes		
Payload	3 ms	4 ms	5 ms
1 B	1920 ms	2560 ms	3200 ms
2 B	2160 ms	2880 ms	3600 ms
3 B	2400 ms	3200 ms	4000 ms
4 B	2640 ms	3520 ms	4400 ms
5 B	2880 ms	3840 ms	4800 ms

Table 4.1: Transmission time for different payloads and TICK sizes

payload including the overhead of information frame. Similarly, the transmission time for different payloads and TICK sizes are calculated and listed in Table 4.1. From the table, the graph shown in Figure 4.5 is plotted.

It can be seen that with smaller TICK size the transmission time is less compared higher TICK size for the same amount of data transmitted. The throughput is calculated from the time taken to transmit one bit of binary data. From Equation 4.11 time taken to transmit one bit data is 30 ms. The calculation of throughput is shown in below:

$$1\,\text{bit} \stackrel{\wedge}{=} 30\,\text{ms} \tag{4.15}$$

$$T_\text{d} \approx \frac{1 \times 1\,\text{bit}}{30\,\text{ms}} \tag{4.16}$$

$$\approx 33.333\,\text{bps} \tag{4.17}$$

$$\tag{4.18}$$

The throughput(T_d) shown in Equation 4.17 is 33.333 bps. Similarly, the throughput with different TICK sizes are calculated as 25 bps with 4 ms TICK and 20 bps for the 5 ms TICK.

The Goodput is the actual information transmitted excluding overhead. The Goodput for different transmission times listed in Table 4.1 is calculated as follows:

$$G_{1\,\text{B}} = \frac{1\,\text{B} \times \frac{8\,\text{bit}}{1\,\text{B}}}{1920\,\text{ms}} \tag{4.19}$$

$$\approx 4.166\,\text{bps} \tag{4.20}$$

Similarly, The Goodput for different payloads are calculated and listed in Table 4.2. The graph shown in Figure 4.6 illustrates the comparison of Throughput with Goodput. The Goodput is lesser as the amount of payload data transmitted is lesser then the frame overhead. The Goodput reaches Throughput when the payload is more.

	Goodput with different TICK sizes		
Payload	3 ms	4 ms	5 ms
1 B	4.166 bps	3.125 bps	2.5 bps
2 B	7.407 bps	5.555 bps	4.444 bps
3 B	10 bps	7.5 bps	6.0 bps
4 B	12.121 bps	9.090 bps	7.2727 bps
5 B	13.888 bps	10.416 bps	8.333 bps

Table 4.2: Goodput for different payload with different tick sizes

One of the performance factor of Li-Fi protocol is throughput which can be increased with decreasing the TICK size based on this context of implementation. In this thesis the data transmitted over Li-Fi is smaller then frame overhead. So, the Goodput is significantly less. In the context, of higher data transmissions the Goodput can reach the Throughput.

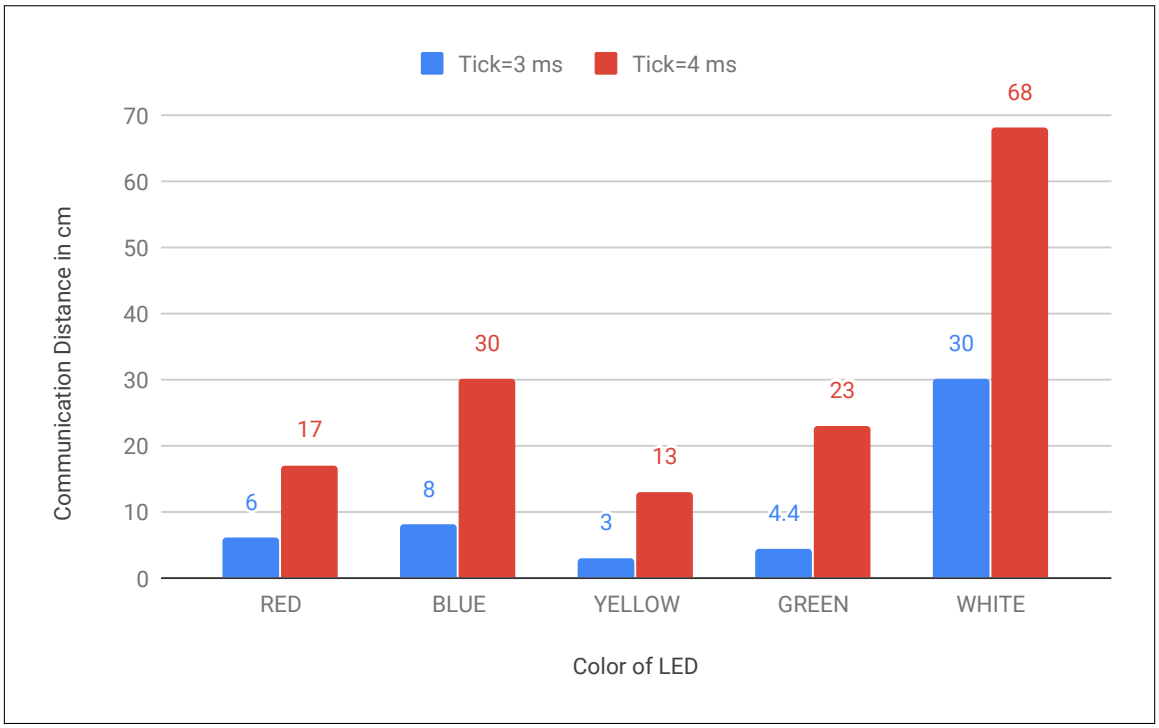


Figure 4.1: Maximum communication distance

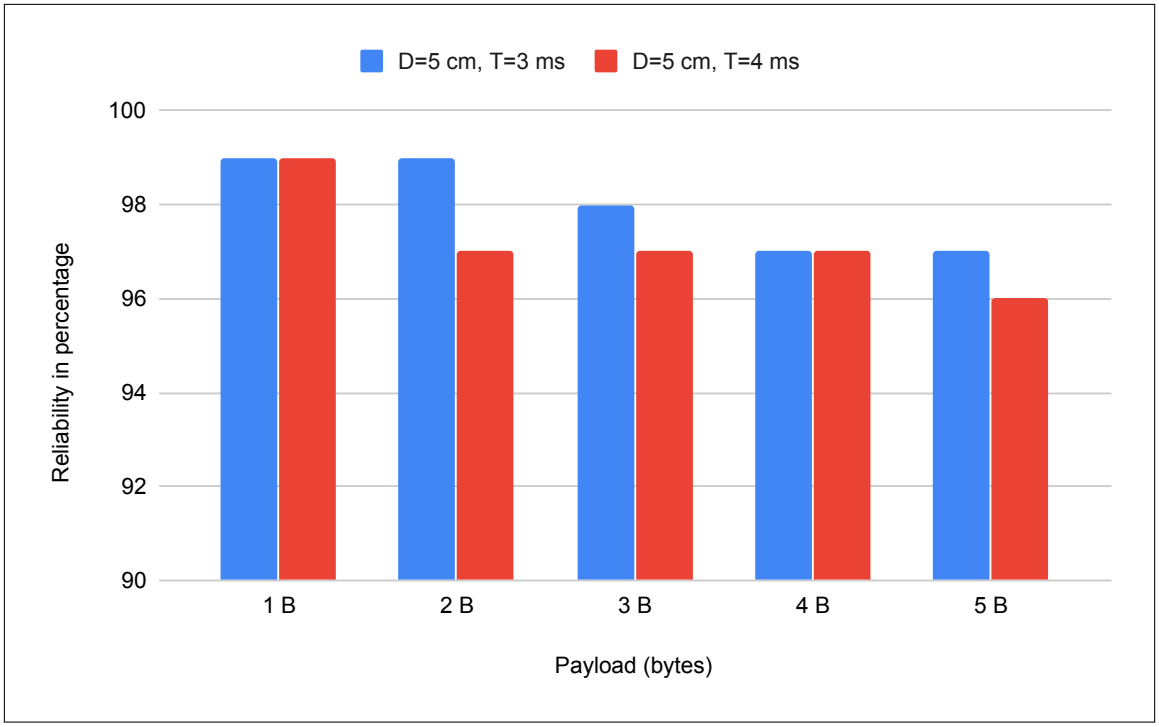


Figure 4.2: Reliability with different payloads and constant communication distance

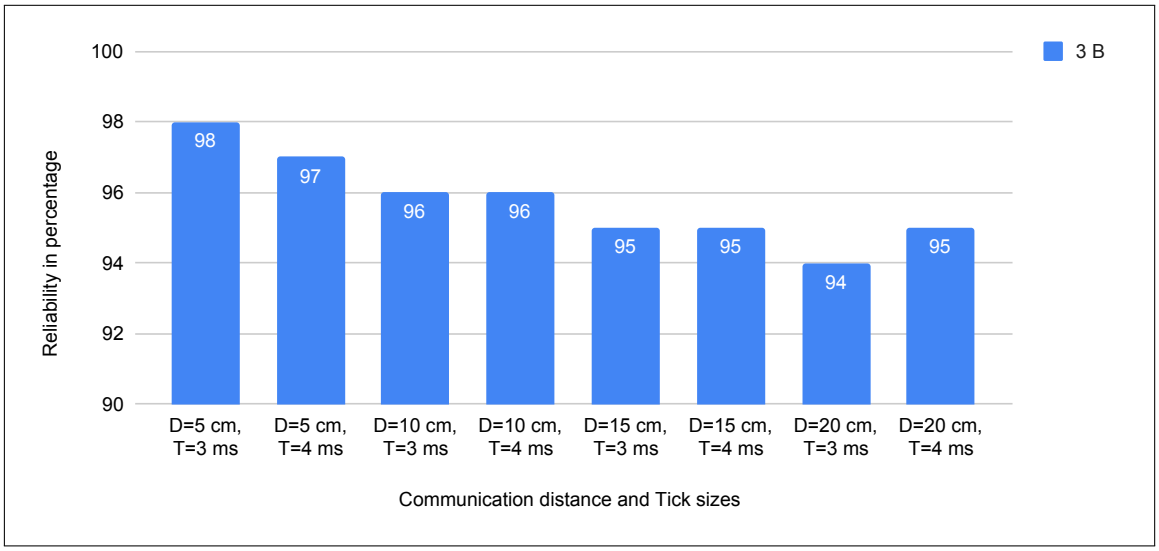


Figure 4.3: Reliability with different communication distance and constant payloads

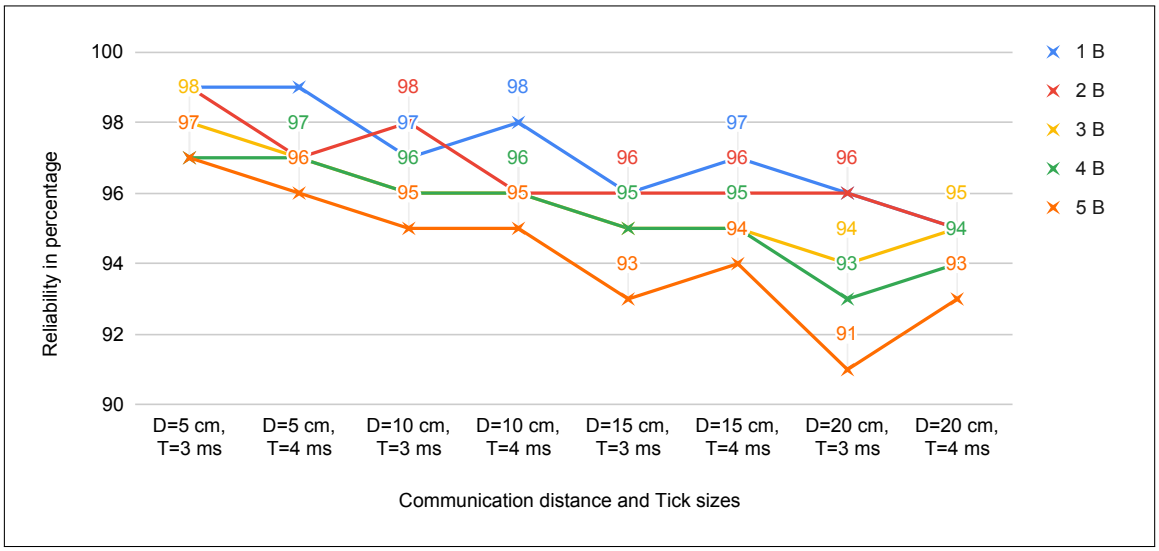


Figure 4.4: Reliability with payload, communication distance and `TICK` sizes

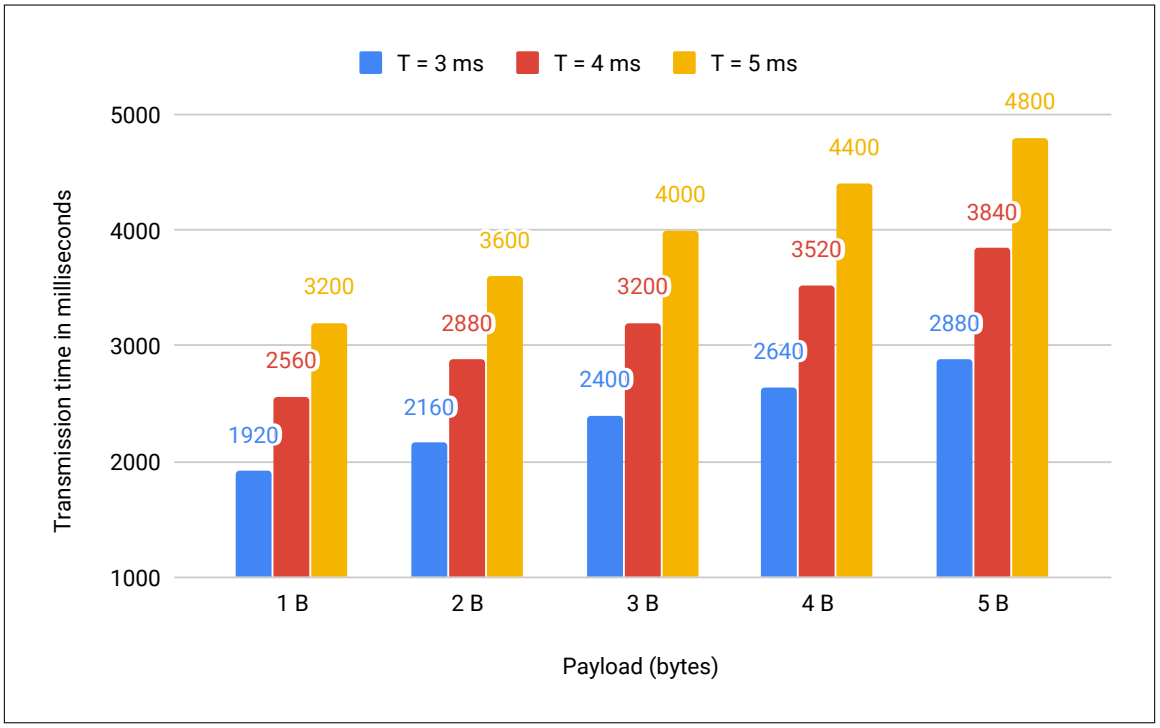


Figure 4.5: Transmission time with different `TICK` sizes

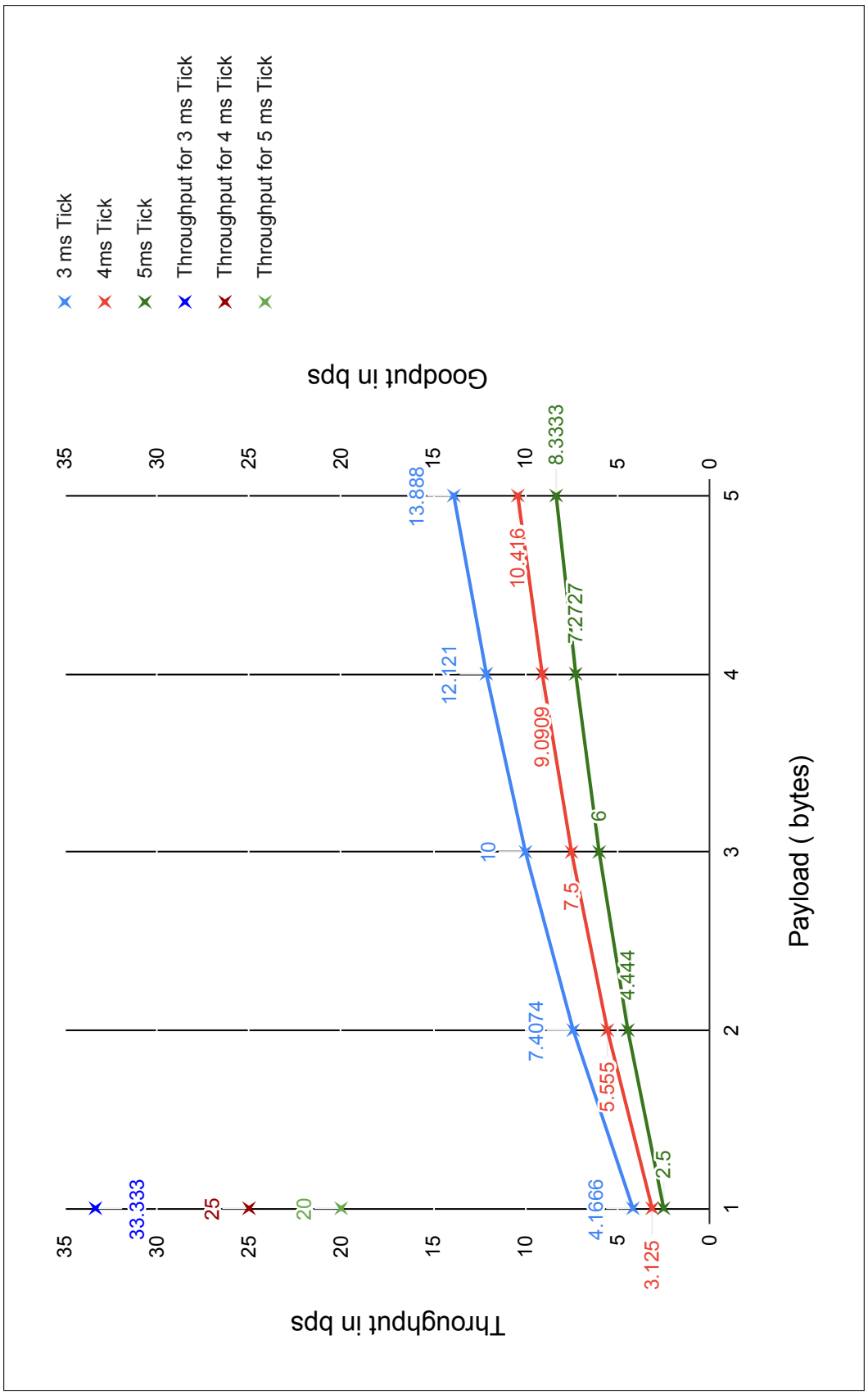


Figure 4.6: Comparison of Throughput and Goodput

CHAPTER 5

Conclusion

5.1 Summary

The goal of this thesis work was to design a robust wireless communication protocol for the distributed factory structures in the factory planning laboratory. This was achieved by setting various objectives. The objectives are the conceptual model of distributed factory layouts, Li-Fi protocol design, Li-Fi transceiver design, stand-alone factory modules from scratch, fabrication of component holders and RFID setup.

The conceptual model was designed based on the master-slave principle with modular structures, as shown in Figure 1.2. Next, the Li-Fi protocol frame format was derived from the HDLC protocol, Manchester encoding and OOK modulation technique. This protocol was designed using various functionalities of RIOT-OS. The software implementation was performed using C++ on Arduino-mega 2560 controller board. The Li-Fi transceiver was designed with LED as a transmitter and LDR as a receiver. The factory modules from Fischertechink [42] are redesigned to convert as stand-alone. 9 V batteries are used to power the modules. RFID based sink selection master node was designed using Arduino-mega 2560 controller and RFID reader. The component holders are designed and printed with the 3d modelling and Arduino Materia101 3d printer.

Once the entire setup was made ready with all the software and hardware components, experiments were performed. Based on the experimental results, the Li-Fi communication protocol was evaluated. The maximum communication distance achieved was 68 cm with a white LED as transmitter and LDR as a receiver. The maximum reliability is achieved with the optimal parameters of Li-Fi protocol. The reliability of the communication was achieved as 99% with the single-hop and 95% with the three hop communication. The maximum throughput achieved was 33.333 bps with a `TICK` size of 3 ms. The evaluation of this thesis gives the best practical experience for setting up the distributed factory layouts in factory planning laboratory.

In summing-up, Li-Fi communication can be adaptable for the factory planning laboratory. This communication can be implemented in real-time applications. Li-Fi is inexpensive and the hardware implementation and setup is straightforward. In this thesis the communication speed achieved is less compared to the actual speed of Li-Fi, which is in terms of $\mathrm{GB\,s^{-1}}$.

The speed can be increased with high speed processors. The Arduino-Mega 2560 provides an ADC clock with 9.6154 kHz which shows the impact on communication speed. But, the communication speed is sufficient for the factory planning laboratory.

5.2 Future Work

As of now the Li-Fi communication protocol was successfully integrated in factory planning laboratory. For further research, one can explore in various areas of the proposed field. For instance, The routing algorithm can be improved by implementing dynamic routing techniques, the communication speed can be increased by choosing a controller which provides high ADC clock. The error correction and re-transmission of data to make the communication more robust and efficient, the designed protocol can be extended in the other layers of OSI model.

Bibliography

[1] Wladimir Hofmann, Sebastian Langer, Sebastian Lang, and Tobias Reggelin. Integrating virtual commissioning based on high level emulation into logistics education. *Procedia Engineering*, 178:24–32, 2017.

[2] Sebastian Lang, Tobias Reggelin, Motasem Jobran, and Wladimir Hofmann. Towards a modular, decentralized and digital industry 4.0 learning factory. In *2018 Sixth International Conference on Enterprise Systems (ES)*, pages 123–128. IEEE, 2018.

[3] i SCOOP. Industry 4.0: the fourth industrial revolution - guide to industrie 4.0. `https://www.i-scoop.eu/industry-4-0/`, 2013. Accessed on 22nd April, 2019.

[4] Luis Barreto, Antonio Amaral, and Teresa Pereira. Industry 4.0 implications in logistics: an overview. *Procedia Manufacturing*, 13:1245–1252, 2017.

[5] Stephan H Mayer. *Development of a completely decentralized control system for modular continuous conveyors*, volume 73. Verlag nicht ermittelbar, 2009.

[6] Thomas Moor, Klaus Schmidt, and Sebastian Perk. Applied supervisory control for a flexible manufacturing system. *IFAC Proceedings Volumes*, 43(12):253–258, 2010.

[7] Wladimir Hofmann, Jan Hendrik Ulrich, Sebastian Lang, Tobias Reggelin, and Juri Tolujew. Simulation and virtual commissioning of modules for a plug-and-play conveying system. In *INCOM 2018-16th IFA C Symposium on Information Control Problems in Manufacturing*, 2018.

[8] Vasu Dev Mukku, Sebastian Lang, and Tobias Reggelin. Integration of lifi technology in an industry 4.0 learning factory. *Procedia Manufacturing*, 31:232–238, 2019.

[9] Robin G Qiu. RFID-enabled automation in support of factory integration. *Robotics and Computer-Integrated Manufacturing*, 23(6):677–683, 2007.

[10] SX Ye and Robin G Qiu. Global identification code scheme for promptly retrieving the pertinent information of a worldwide uniquely identifiable object. In *2003 4th International Conference on Control and Automation Proceedings*, pages 1000–1004. IEEE, 2003.

[11] Louis Louw and Mark Walker. Design and implementation of a low cost RFID track and trace system in a learning factory. *Procedia Manufacturing*, 23:255–260, 2018.

[12] Marko Mladineo, Ivica Veza, Nikola Gjeldum, Marina Crnjac, Amanda Aljinovic, and Andrej Basic. Integration and testing of the RFID-enabled Smart Factory concept within the Learning Factory. *Procedia Manufacturing*, 31:384–389, 2019.

[13] Sridhar Rajagopal et al. IEEE 802.15. 7 vlc phy/mac proposal samsung/etri. *IEEE P802*, pages 15–09, 2009.

[14] Anurag Sarkar, Shalabh Agarwal, and Asoke Nath. Li-fi technology: Data transmission through visible light. *International Journal of Advance Research in Computer Science and Management Studies*, 3(6), 2015.

[15] Latif Ullah Khan. Visible light communication: Applications, architecture, standardization and research challenges. *Digital Communications and Networks*, 3(2):78–88, 2017.

[16] Jelena Grubor, Sian Chong Jeffrey Lee, Klaus-Dieter Langer, Ton Koonen, and Joachim W Walewski. Wireless high-speed data transmission with phosphorescent white-light LEDs. In *33rd European Conference and Exhibition of Optical Communication-Post-Deadline Papers (published 2008)*, pages 1–2. VDE, 2007.

[17] Neeraj Kumar Mishra. Xilinx HDLC bit stuffed algorithm for insertion and deletion and checking 32bit CRC for 16 bit address. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.300.4842`.

[18] Roger Forster. Manchester encoding: opposing definitions resolved. *Engineering Science and Education Journal*, 9(6):278–280, 2000.

[19] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. Operating systems for low-end devices in the internet of things: a survey. *IEEE Internet of Things Journal*, 3(5):720–734, 2015.

[20] Emmanuel Baccelli, Cenk Gündoğan, Oliver Hahm, Peter Kietzmann, Martine Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C Schmidt, and Matthias Wählisch. RIOT: An open source operating system for low-end embedded devices in the IoT. *IEEE Internet of Things Journal*, 5(6):4428–4440, 2018.

[21] Emmanuel Baccelli, Oliver Hahm, Mesut Güneş, Matthias Wahlisch, and Thomas C Schmidt. RIOT OS: Towards an OS for the internet of things. In *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 79–80. IEEE, 2013.

[22] Aleksandar Milinković, Stevan Milinković, and Ljubomir Lazić. Choosing the right RTOS for IoT platform. *Infoteh-Jahorina*, 14:504–509, 2015.

[23] RIOT GitHub. RIOT github repository. `https://github.com/RIOT-OS/`, 2015.

[24] RIOT Developers. Riot documentation. `http://api.riot-os.org/`, 2013. Accessed on 26$^{\text{th}}$ February, 2019.

[25] Harald Hass. purelifi. `https://purelifi.com/harald-haas-on-inspired-edinburgh-podcast/`, 2011.

[26] Harald Haas. LiFi is a paradigm-shifting 5g technology. *Reviews in Physics*, 3:26–31, 2018.

[27] IEEE Standard. IEEE approved draft standard for short-range wireless optical communication using visible light. *IEEE P802.15.7/D8, April 2011*, pages 1–306, Feb 2011.

[28] Mehmet Fatih Isik, Busra Yartasi, and Mustafa Resit Haboglu. Applicability of Li-Fi

technology for industrial automation systems. *International Journal of Electronics and Electrical Engineering*, 5(1):21–25, 2017.

[29] Cheol-Min Kim and Seok-Joo Koh. Device management and data transport in IoT networks based on visible light communication. *Sensors*, 18(8):2741, 2018.

[30] Vinayagam Mariappan, Soonho Jung, Sangwoon Lee, and Jaesang Cha. IoL field gateway: An integrated IoT agent using networked smartLED lighting controller. 34(2):12–19, 2017.

[31] Weizhi Zhang, MI Sakib Chowdhury, and Mohsen Kavehrad. Asynchronous indoor positioning system based on visible light communications. *Optical Engineering*, 53(4):045105, 2014.

[32] Liwei Ding Vinnarasi and ST Aarthy. Transmission of data, audio signal and text using Li-Fi. *International Journal of Pure and Applied Mathematics*, 117(17):179–186, 2017.

[33] Mostafa Zaman Chowdhury, Md Tanvir Hossan, Amirul Islam, and Yeong Min Jang. A comparative survey of optical wireless technologies: Architectures and applications. *IEEE Access*, 6:9819–9840, 2018.

[34] Stefan Schmid, Giorgio Corbellini, Stefan Mangold, and Thomas R Gross. An LED-to-LED visible light communication system with software-based synchronization. In *2012 IEEE Globecom Workshops*, pages 1264–1268. IEEE, 2012.

[35] Arduino devlopers. Arduino-mega2560. `https://www.arduino.cc/en/Guide/ArduinoMega2560`, 2015. Accessed on 26$^{\text{th}}$ May, 2019.

[36] Friends-Of-Fritzing. Fritzing. `http://fritzing.org/learning/`, 2016. Accessed on 12$^{\text{th}}$ May, 2019.

[37] PulseViewSigrok. PulseView - sigrok. `https://sigrok.org/wiki/PulseView`, 2012. Accessed on 10$^{\text{th}}$ May, 2019.

[38] Tinkercad. Tinkercad. `https://www.tinkercad.com/`, 2015. Accessed on 26$^{\text{th}}$ May, 2019.

[39] David Benson. Flowchart maker and online diagram software. `https://www.draw.io/`, 2015. Accessed on 26$^{\text{th}}$ May, 2019.

[40] AutomationWiki. CRC-16-CCITT. `http://automationwiki.com/index.php/CRC-16-CCITT`, 2017. Accessed on 26$^{\text{th}}$ May, 2019.

[41] Arduino devlopers. Analog to digital conversion. `https://learn.sparkfun.com/tutorials/analog-to-digital-conversion/all`, 2015. Accessed on 26$^{\text{th}}$ May, 2019.

[42] Fishertechnik. Fishertechnik. `https://www.fischertechnik.de/de-de/produkte/simulieren`, 1965. Accessed on 26$^{\text{th}}$ May, 2019.

Appendix

A.1 RIOT-OS Setup in Linux Environment

Clone the source repository of RIOT-OS from GitHub repository

Instructions to setup Linux environment with RIOT-OS for Arduino-mega2560 controller:

Install avr-gcc toolchain:

```
sudo apt-get update
sudo apt-get upgrade all
sudo apt-get install gcc-avr binutils-avr avr-libc
sudo apt-get install gdb-avr
```

Install avrdude:

```
sudo apt-get install avrdude
```

Install directly from the shell:

```
sudo apt-get install gcc-avr binutils-avr gdb-avr avr-libc avrdude
```

Include board in bash:

```
export BOARD=arduino-mega2560
```

A.2 Building an Application using Makefile

```
APPLICATION = name of the application
# If no BOARD is found in the environment, use this default:
BOARD ?= arduino-mega2560
# This has to be the absolute path to the RIOT base directory:
RIOTBASE ?= $(CURDIR)/../RIOT-spi
# Comment this out to disable code in RIOT that does safety checking
# which is not needed in a production environment but helps in the
# development process:
DEVELHELP ?= 1
```

```
# Change this to 0 show compiler invocation lines by default:
QUIET ?= 1
USEMODULE += arduino
FEATURES_REQUIRED += periph_spi
FEATURES_REQUIRED += periph_adc
FEATURES_REQUIRED += periph_eeprom
include $(RIOTBASE)/Makefile.include
CXXEXFLAGS += -std=c++11
CFLAGS += -DDEBUG_ASSERT_VERBOSE
```

I herewith assure that I wrote the present thesis titled *Li-Fi (Light-Fidelity) in Industry 4.0: Integration of Li-Fi Communication in a Factory Planning Laboratory* independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, 23rd August, 2019

(Vasu Dev Mukku)