ABSTRACT

For the improvement on efficiency in lighting systems' practicality and power consumption, Digital Addressable Lighting Interface (DALI) is known as a solution. This project focuses on studying DALI protocol and the development of a DALI master demonstration board with microcontroller. Tests were conducted to ensure the capability of NUCLEO-F446RE development board to establish DALI Protocol. By conducting tests of demonstrating protocols to communicate with commercial DALI ballasts, the designed DALI master overall resulted in an operable demonstrator of DALI protocol to control the power of LED luminaires. Bi-directional communication testing is conducted to receive statuses of the attached Luminaires. The operation is aided with a Guide User Interface (GUI) program developed in python. Additionally, this project simplifies the final solution by proposing a value-line microcontroller for improvement in cost and power consumption.

ACKNOWLEDGEMENT

Under COVID-19 pandemic, various governance and policies is established to adjust with the pandemic, resulting in limitations to travel. Thus, under these constraints, I decided to travel back to my home country, Indonesia. This project was carried within a home workstation with bare-minimum testing equipment's and solder station, under Professor Scott Roy's supervision. As a result, safety and precautions must be carried solely by myself within the workstation to ensure professional working conditions.

I would dedicate this completion of my project to my family, for their providence both in financial and emotional support during my studies in Indonesia.

Furthermore, I would like to express my gratitude to Professor Scott Roy for all the guidance and help upon the completion of this project. With all his kindness and compassion in teaching, I have found a mentor for my improvement for my academics and engineering life in general. Despite the current difficult times for academics and teaching, I am grateful for his professionalism in arranging times for our weekly meetings and answering my questions. Thank you, professor.

Tears have been shed from sleepless nights, both from hardships and joyous eureka moments.

Effects of Covid-19

This project is affected by the pandemic due to limitations of travel. Hence, access to electronics lab in University of Glasgow is not gained in completion of this project. Some of additional purchases are made to carry the tests of this project:

- RIGOL DS1054Z Digital Oscilloscope
- KOOCU DC Power Supply (0 – 15v)
- LTSYS DALI LED Dimmer

Other peripherals outside the list are personal properties of the author which have been owned before the start of this project, e. g. solder workstation and NUCLEO-F446RE development board. Due to limitations of the solder station, the components used to this project is only Through-Hole components. PCB is manufactured in Spectra in Bandung, Indonesia, a reputable PCB manufacturing company.

Safety precautions are carried individually by wearing goggles during work and careful inspection of electricity isolations within the workstation.

Table of Contents

List of Figures

1. Introduction

Ranging from industrial to home applications, lighting systems are responsible for significant energy consumption in the world. Total domestic power consumption for lighting loads accounts for 25 to 30% globally [1]. Furthermore, based on a recent assessment by the European Commission, the potential to maximise the power consumption efficiency in lighting is significant up to 15% in the EU and 44% solely in China by updating obsolete lighting systems [2]. Replacements of obsolete lighting systems to novel lighting systems predictably could enhance the efficiency of global power consumption.

The demand for installing increasingly efficient lighting systems is indicated by the rise in Light Emitting Diodes (LED) domestic sales. From a report by International Energy Agency (IEA), the potential sales for LED Luminaires in 2030 for domestic purposes will surpass Fluorescent lights (CFL) which predictably accounts for 87%. Dominating usage of LED in the future is predicted to reach sustainable energy future by designating highly efficient lighting systems [3].

To cope with the world's demand for increased lighting usage, Digital Addressable Lighting Interface (DALI) has been developed to establish an efficient lighting control system, to effectively manage lighting networks with enhanced power-saving abilities. Employing the DALI protocol gains control of the intensity of lighting systems. DALI protocol is a duplex 12V logic bus communication system to control and manage luminaires through lighting ballasts, occasionally with real-time monitoring [4]. Moreover, the DALI protocol could be adapted to acquire data from lighting ballasts such as brightness level, dimming status, and power consumption [5].

Alongside the advancement of technology development in smaller and faster microcontrollers, this project updates the compatibility of the DALI protocol to be demonstrated with the latest microcontrollers as the DALI master. The significance of using a microcontroller is to send and receive commands to control luminaires through DALI ballasts. Connection from the central operation PC to the microcontroller is established. Thus, user interface software is developed to establish communication with the microcontroller via USB to ease the operations. By analysing the latest available microcontroller as a benchmark of the project, improvements in the affordability, simplicity, and power consumption are proposed for a DALI master demonstrator device.

1.1 Aim and Objectives

This project aims to study how DALI protocol, and thus a solution for energy-efficient lighting systems, can be demonstrated in practice by utilising currently available microcontrollers. Therefore, the objectives of this project are distributed into the following.

- Design, manufacture, and test a DALI driver, utilising a microcontroller as the processing unit. The baseline design criteria are to propose power consumption and production cost to outstand current market solutions of DALI development boards.
- Develop and test microcontroller embedded software by means to encode and decode basic DALI commands.
- Develop and test GUI software of DALI control with objectives of being user-friendly for end-users.

2. Literature Review

2.1 Energy Consumption Analysis

Technological advancements on smarter lighting systems in support of sustainable energy goals by 2030 appropriates a demand in replacing current systems with LED luminaires. From EU statistics alone, the energy consumption budget of public lighting in European cities accounts for 60%, for which it could be reduced by 70% with the implementation of LED luminaires across public places [6]. The DALI protocol currently supports both LED and CFL lighting systems, which indicates the potential of DALI to be implementable with future lighting systems. To support the aim, a recent assessment of lighting systems energy consumption simulated regulated LED luminous flux controlled with DALI as this results in an 8.5% reduced power consumption [7].

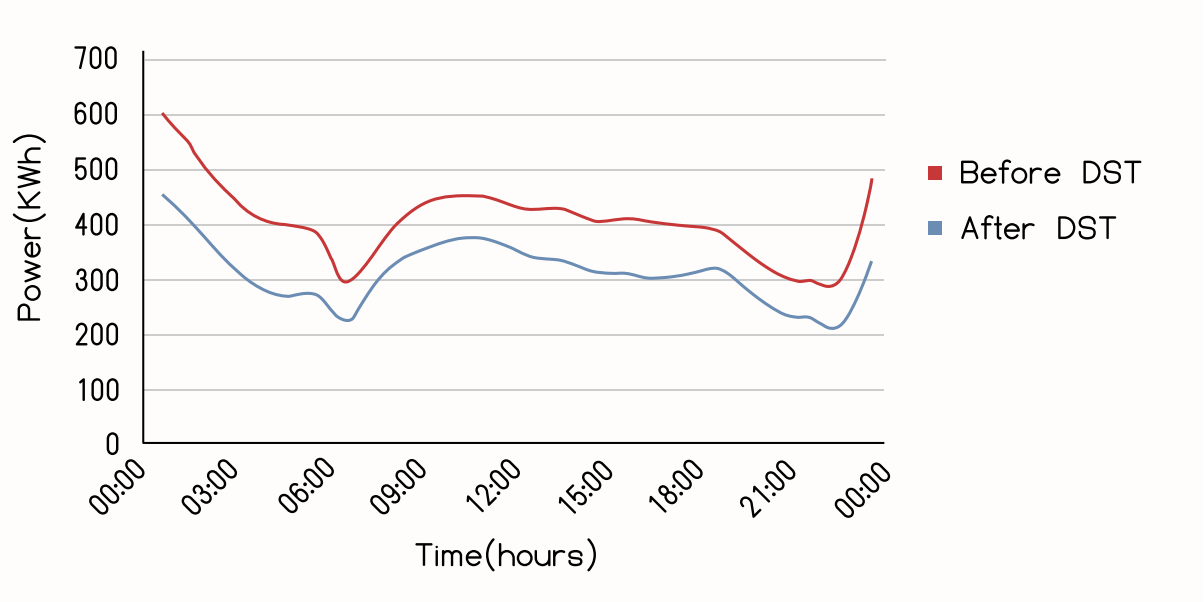


Figure 1. *Power Analysis of Energy Saving in 24 Hour Time Before and After DALI Scheme Transfer (DST). Redrawn and Adapted from: [8].*

Another research has simulated DALI usage. As seen in figure 1, DALI Scheme Transfer is applied in a building with multiple floors, inferring that DALI implementation reduces daily power consumption by around 18% of usage per day. DALI Scheme Transfer optimises lighting intensity at specified hours, which is programmed sequentially to the time in 24 Hours [8].

2.2 DALI Protocol

The DALI protocol under the IEC 62386 standards is a two-way connection that utilizes a 12V half-duplex connection, setting communications between DALI master and slaves. DALI protocol is an advancement of a previous lighting control system, the 1 – 10 V DSI system [9]. The key advantage of DALI lies in its addressable and groupable slaves, advancing the predecessor systems in ease of operations, allowing batches of luminaires to easily be controlled in one command. The possibility of automation of controlling the luminaire sets is also prominent.

Under the standards of DALI, a DALI master can support up to 64 slave units with individual addresses to be allocated. The maximum set-up distance for DALI is 300 m and it is limited to a 2V maximum voltage drop and 250mA across the half-duplex connection. The communication signal format is in the form of the digital communication format of asynchronous Manchester packets, which contain address bits for the first half of the packet

and command bits for the second half [10]. DALI protocol takes advantage of being compatible with central processing units as a programmable lighting system. Logic levels of DALI characterised high logic levels around 9.5 to 22.5, and -6.5V to 6.5V for low logic level [11]. Headroom of 13V for each logic level is possible to give error margin which is caused by overvoltage from components such as inductors or capacitor charges.

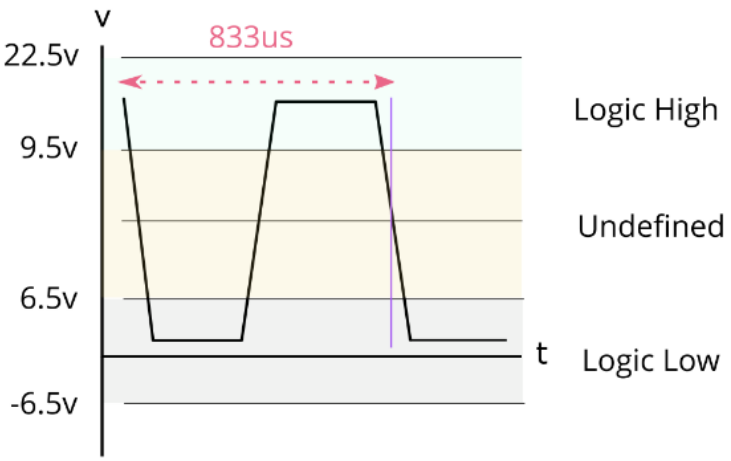


Figure 2. *Time-Voltage Plot of Logic Switching Levels Under DALI Standards. Redrawn From: [11] .*

2.3 DALI Packets and Commands

DALI Protocol employs Manchester encoding as its signal format to ensure error-free communication. Manchester encoding is an asynchronous digital synchronous format that differs the rising edge transition phase as logic "1" and the falling edge transition phase as logic "0". The absence of a transition phase under a designated clock time is considered an error [10]. Manchester encoding is highly attractive due to peak performance under additive noise and greater timing extraction [12]. One clock cycle of DALI standards requires 1200 bits/second to send protocol data, which requires 416,666us for one single bit. The transmission frequency of the DALI packets is designed to withstand external noise, as faster speeds might interfere with noise sources, predictably high-power noise from Power-Line Communication (PLC) [13], which this communication method is commonly installed alongside lighting systems [14]. Bidirectional communication of DALI protocol establishes a forward frame to allocate DALI ballast' address and send control commands. As it could be seen in figure 3, the frame contains one start bit and two stop bits to encapsulate the packet. The backward frame

is a feedback response from DALI standardised ballast, containing information related to the attached luminaire, which could be seen in figure 4.

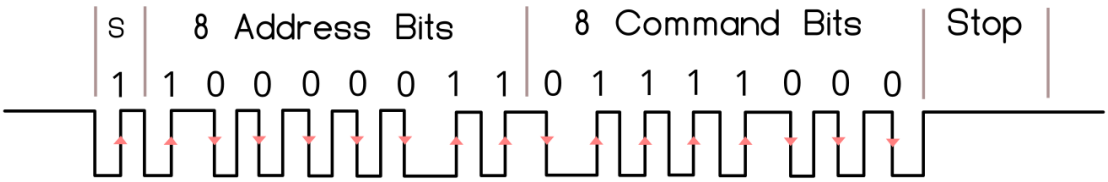


***Figure 3**. Manchester Encoded DALI Forward Frame of "203" and "170"*

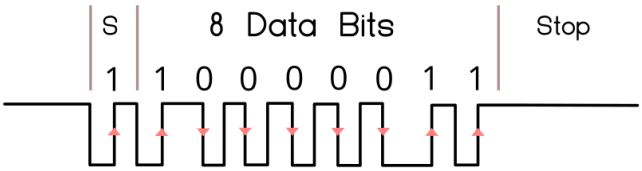


***Figure 4.** Manchester Encoded DALI Backward Frame Example*

DALI protocol supports up to 64 assignable ballast addresses and 16 assignable groups of ballasts to control multiple ballasts in a group simultaneously. From Table 1, "A" is the corresponding bit of the addresses. The least significant bit of "S" on the address corresponds for "0" if the command is for DIRECT ARC POWER CONTROL to assign the power control for the luminaires, or "1" for other commands. Direct arc power commands are used to control lighting with the value of the bits to control logarithmic light levels.

***Table 1**. Assignable Addresses with Corresponding Bytes [15].*

Type	Address Code	Address Byte
Short Addresses	0 - 63	0AAA AAAS
Group Addresses	0 - 15	100A AAAS

A special case of broadcast addressing of 255 (1111 111S) is to target all ballasts attached to the DALI bus, as any following command would affect all the corresponding attached ballast. Commands of DALI protocols are from the DALI master.

Query commands are sent to request the status of the current state of the luminaire attached to the ballast, such as communication availability or power arc level. As a form of response, backward frames are sent from the ballast to answer the statuses of the luminaire attached with each bit from bit 0 to bit 7 corresponding to a Boolean 'yes' or 'no' answer.

Table 2 shows important commands as examples of the DALI forward frame. Note that Y infers the value of "0" for short addresses (e. g single DALI ballast) and "1" to target a group address.

Table 2. *Important Commands of DALI Protocol Commands as Examples [15]*

Command Type	Address and Command Code	Command Number
DIRECT ARC POWER CONTROL	YAAA AAA0 XXXX XXXX	n/a
OFF	YAAA AAA1 0000 0000	0
ON AND STEP UP	YAAA AAA1 0000 1000	8
QUERY STATUS	YAAA AAA1 1001 0000	144

2.4 DALI Logarithmic Dimming

To control the lighting intensity or dimming value, ballasts should regulate the power level to control the luminaire consumption with minimum arc power of 0.1%. Equation 1 represents the relationship of the logarithmic dimming level of 0.1 % to 100% with corresponding 8-bit value interpolation, which is 0 to 255 in integer values. The binary bits value of the dimming level is replaced by "X" in table 2.

$$\boldsymbol{X(n) = 10^{\frac{\frac{n-1}{253}-1}{3}}} \quad \boldsymbol{(1)}$$

Equation 1. *DALI Logarithmic Dimming Mapping Equation [16]*

The correspondent DALI ballast would interpret the dimming level to set PWM values that supply power to the attached luminaire. Logarithmic dimming is implemented due to the appeal for end-users as scene environment setting aesthetics. This dimming method is also suitable for the human eye when perceiving light intensity, as the nature of its perception is logarithmic itself [17]. However, another dimming curve method, the Square Law curve is reportedly more favourable by end-users in an environment setting [18]. This result could be beneficial for the evaluation of DALI standards for dimming curves choice.

2.5 DALI Network Connection

As mentioned previously, the configuration of the lighting system under the DALI compliance allows the master to drive multiple ballasts. Figure 5 shows an example of the networking of DALI protocol, adapted from the DALI guide [9]. Notably, the connections are established with two wires of D+ and D-.

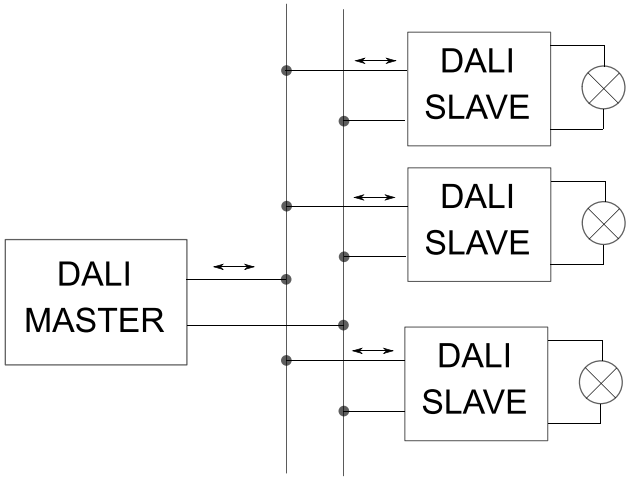


Figure 5. *DALI Network Connection Example*

2.6 Current Available Solutions

By reviewing previous works of DALI evaluation boards, the most predominant solutions are taken into consideration. The analysis was conducted to observe baseline criteria to develop a DALI demonstrator. Breakdowns of worth-mentioning evaluation boards are analysed below.

AN11175 - NXP Semiconductors

Produced by NXP semiconductors, the AN11175 is a DALI master intended for building lighting control use in the form of an evaluation board. This device features USB Human Interface Device (HID), which operates without dedicated software installed on the host PC. The system is built to operate the working of LPC1343, which is cored with ARM Cortex-M3 with a clock speed of 72MHz. It is included with 32kB on-chip Flash, 8kB SRAM, and UART [19]. The physical layer contains the DALI circuitry which utilizes optocouplers to create isolation between the microcontroller side to the DALI bus as logic level conversion.

AC160214-1 - Microchip

Widely available in the market, the AC160214 is a DALI master & slave support prototyping communication board. It is cored with PIC16F1947 to support the user interface and communication of the peripherals. Similar to the AN11175, optocouplers are utilised by means for logic isolations. However, this prototype board requires an adapter of DM160214-2 to operate with an RJ45 connection, and other peripherals such as a USB connection are separated [20].

DALI CLICK- MikroElektronika

The DALI Click is an innovative adapter board that is populated with DALI interface circuits to adapt raspberry PI SOC with DALI Bus [21]. Therefore, this solution requires an operable raspberry PI to control lighting system with DALI, which relatively consumes larger power for constant usage compared to the utilization of microcontrollers.

Table 3 details the summary of the current and additional available solutions alongside the market price obtained from distributors' websites, such as Farnell, and Digikey.

Table 3. *Noteworthy Available Market Solutions of DALI Master Evaluation Boards*

Device	Microcontroller	Price (USD)
DALI Control Gear Infineon	XMC1200	129.90
Microchip AC160214	PIC16F1947	40.80
OM13046, 598	LPC134x	68.75
DALI CLICK (MIKROE-1297)	Raspberry PI	25.00 (Without Raspberry Pi)

To summarize, from the analysis of current solutions, a design baseline is concluded to create the DALI Demonstrator evaluation board with the following preliminaries.

Microcontroller: To utilize a working microcontroller that is capable to process DALI protocol. Evaluation on microcontroller selection is discussed further.

Cost: To reduce the bill of materials of the populated PCB of the DALI Demonstrator circuit without the degradation of DALI protocol standard operation. This is appealing for further research to create a price-competitive solution.

3. Methodology

The mentioned design criteria from available designs are selected to fulfil the baseline requirements of a standard DALI master demonstrator, namely, to send DALI packets and receiving with potentially multiple DALI standardised ballast. Overall, the microcontroller is required to be programmed to read serial communications in the form of integer input by the PC connection and convert it into DALI packets. Furthermore, DALI protocol logic levels are generated with DALI master circuitry, steady to be transmitted to commercial DALI ballasts. To ensure the bi-directional function of the protocol, DALI standards require the master to be tested with a receiving circuit to read a backward frame from DALI ballasts. Figure 6 exhibit the overview of the designed system in this project.

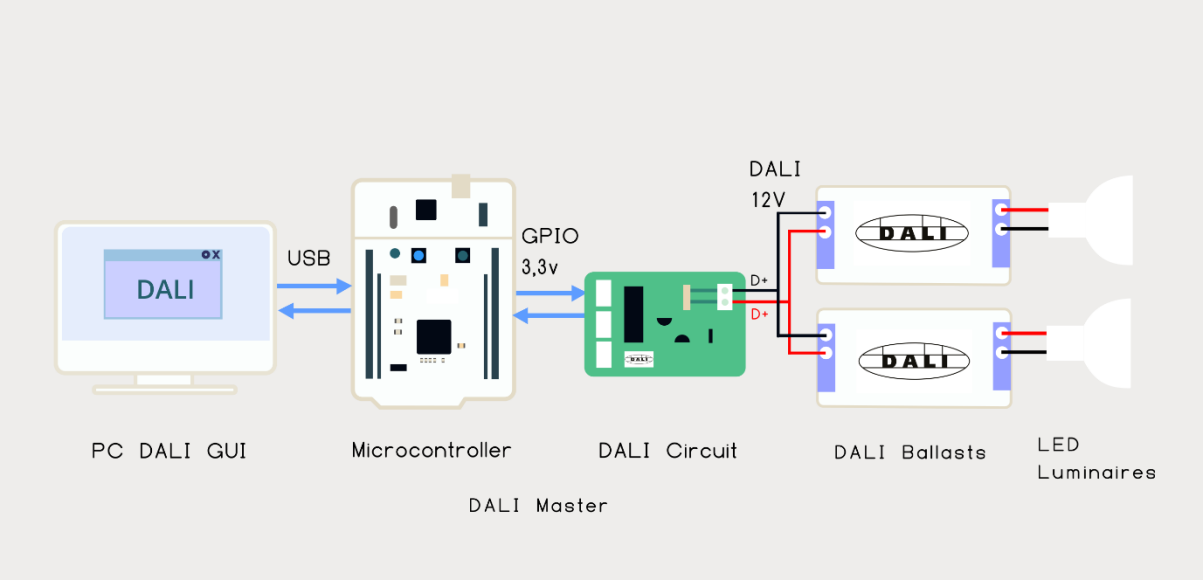


Figure 6. *DALI Protocol Demonstrator System Overview*

3.1 Hardware Overview

3.1.1 DALI Master

The proposed DALI master comprises a microcontroller and a DALI circuitry. The designed DALI Interface circuitry is adapted from the NXP DRM0004 DALI demo board, limitations of equipment in personal lab limits the ability to populate PCB with Surface Mount Device (SMD) components, note that this version of the designed DALI circuitry consists of Through-Hole Technology (THT) components, which contrasts to the use of SMD components in the NXP DRM0004. The interface circuitry is then adapted with through-hole components by the following project of DALI implementation in homes [22]. After studying this circuit, adjustments are made to compensate for the availability of components from local distributors.

The objective of the circuitry is to convert Manchester encoded packets in the form of 3.3V logic levels to 12V DALI logic levels. The circuit of the following schematic in Figure 7 is the DALI TX. The Manchester encoded packet is inverted with a 74HC04 module, since the nature of the DALI protocol signal is normally high, an inverter is required to saturate the transistor Q3. For the case of Q3 being activated, 12V of the supply voltage is shorted, therefore it enables a logic low to the DALI BUS. Therefore, through the combination of 12V logic low and high, it establishes a series of logic, encoded in a packet.

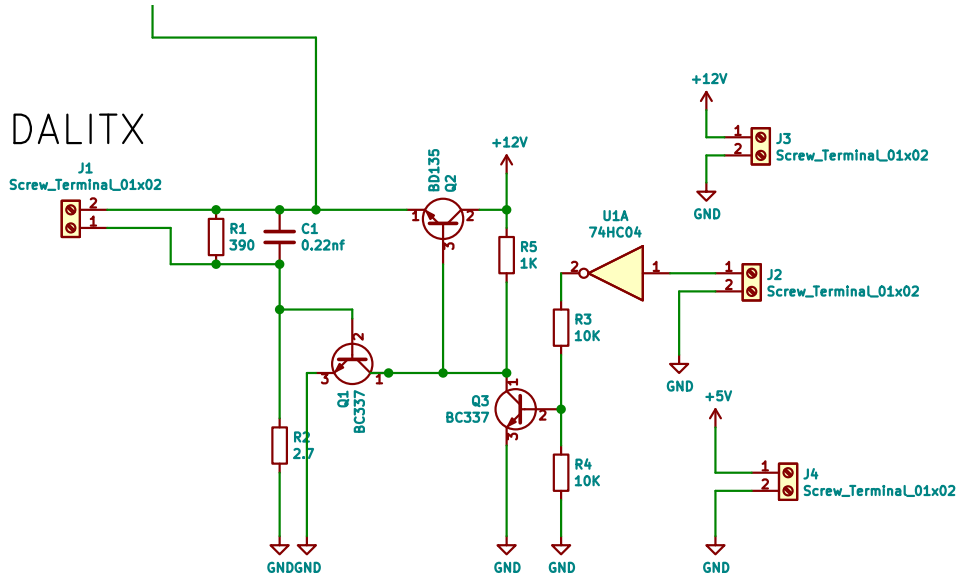


Figure 7. *DALI Transfer Interface Schematic*

As seen in figure 8, the receiving part of the circuit consists of a comparator of LM393 which of a substitute of LMC7221 from the board referenced in [4], enabled in a Schmitt-trigger configuration which operates on 5V power. The voltage reference input to the non-inverting pin of the comparator is set around 10v, hence it would compare logic from the DALI bus, which 12V is enabled to logic high, and below 10v is enabled to logic low. The output pin of the comparator is supplied directly to the input pin of the microcontroller. In summary, the circuitry creates a logic isolation method of 12V logic levels to 5v logic levels. A circuit protection method is realised by Q1, when unwanted signal is conducted to D-, Q1 shorts the 12V to ground.

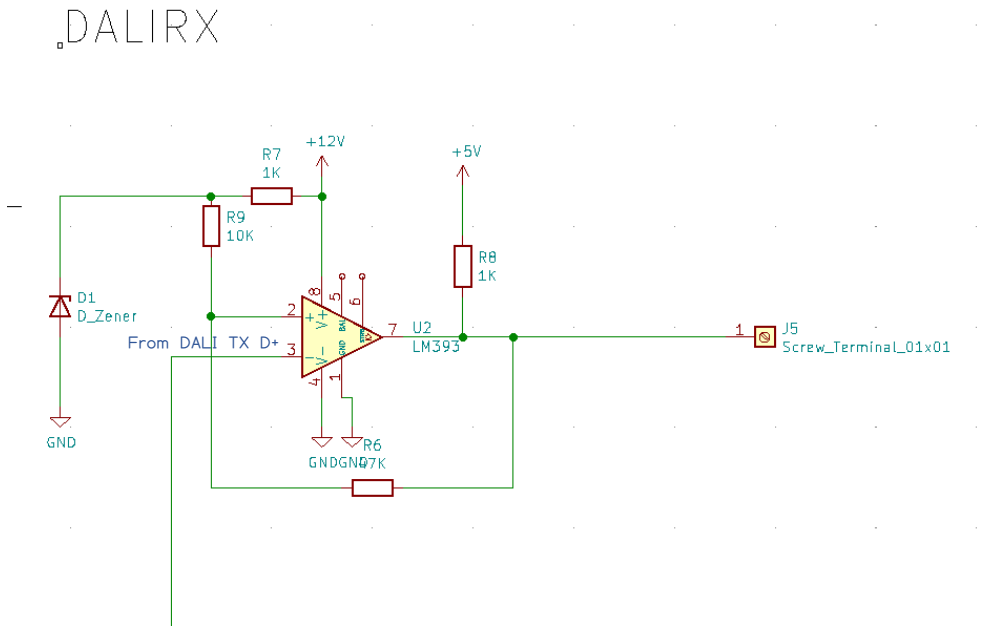


Figure 8. *DALI Receiver Interface Schematic*

3.1.2 Microcontroller Selection

Predicted baseline criteria to select an evaluation board for bare minimum testing demonstration are summarized into several following aspects.

- A minimum clock speed of 24MHz or possibly lower to compensate for the bit transmission speed of 1200 bit/s.
- 128Kbytes of memory storage, as this is a typical size that would suffice.
- Two GPIO pins of transfer and receive pins, with interrupt support.
- USB or UART support to establish serial communications with GUI.

The rise of ARM cortex processors' popularity has been an additional decision selector for the microcontroller board. Narrowing to the selection of MBED-supported microcontrollers familiarity with the supported Integrated Development Environment (IDE) is of great importance for the time effectivity of the development process, which is a crucial design determinator. Subsequently, the selected operating system for this project is the MBED-OS.

The available device which suffices the criteria is the NUCLEO-F446RE by STMicroelectronics, which employs an ARM Cortex M4 as the core. The NUCLEO-F446RE features 180Mhz maximum Clock Speed, with additional highly sufficient memory of 512KB. Although this selection decision is overly adequate, it is not the optimum solution through power consumption and cost. By designating for alternatives within the ARM Cortex family,

optimum costs and power reduces would be the next design criteria for future revisions. And note that the utilisation of the development board is limited to testing stage only.

3.1.3 PCB Design

To increase space effectivity to the design, given that the DALI is established to be applicable in industrial and household settings, PCB for the DALI circuitry was designed. Two attempts of designs were made, as an approach of fail test before commencing further milestone of the design process. The first master board is the DALI TX circuitry to demonstrate the data transfer protocol. Featured aspects of this PCB contain block terminals for ease of connections to other attached peripherals. As a practice, 14 Pin DIP sockets are installed to accommodate the 74HC04 IC packet from STM Electronics, as this installation would be a replaceable mount to ease the debugging process.

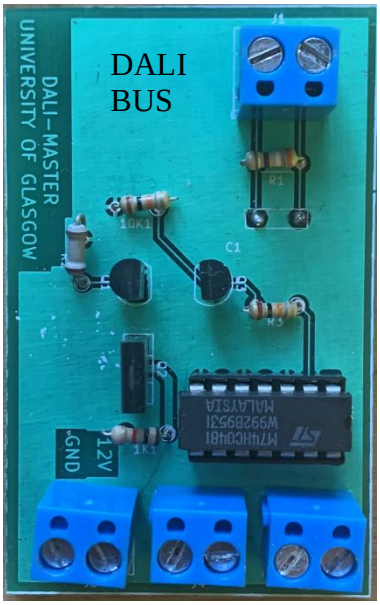


Figure 9. *First Iteration of DALI Interface PCB*

After the successful task is completed by the current PCB, the creation of a second design proceeds. The second attempt of the PCB master design is the implementation of both the DALI TX and DALI RX circuitry. Noise prevention caused by sources such as power rails was attempted by the separation of ground routes and power routes from the signals. Troubleshooting and debugging of this PCB version have resulted in a successful DALI protocol demonstration with both features of transmitting and receive.

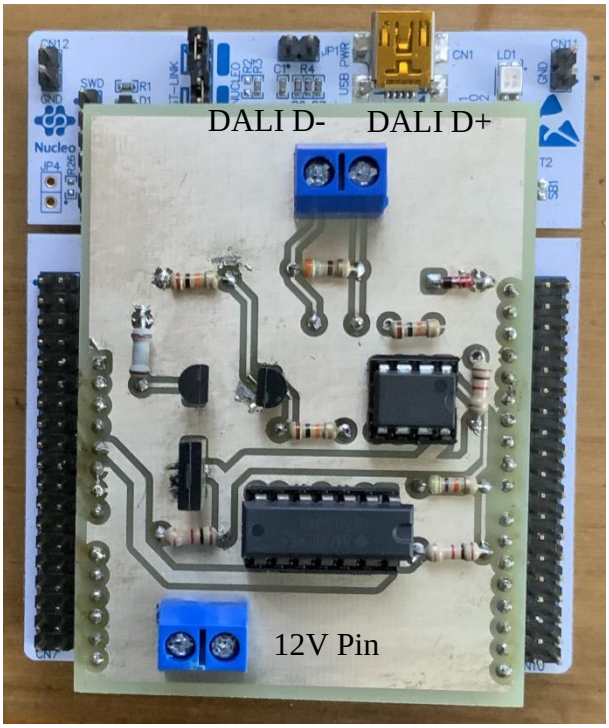


Figure 10. *Second Iteration of DALI Interface PCB*

After the testing and debugging process, the second iteration of the DALI Interface PCB was improved with the "shield" feature to the NUCLEO-F446RE pins. As can be seen in figure 10, this design decision was made to ensure the practicality of the installation to the design, thus additionally increased the effectiveness of the space consumed. It also reduces the cable connections.

All the iterations of the designed PCB board file are created in KiCAD PCB, an open-source electronic design software. PCB design was manufactured from the company of Spectra PCB in Bandung, Indonesia. The process of masking the solder is coated with epoxy, and the PCB is CNC-milled. Finally, the components are populated by the method of soldering from home workstation.

3.1.4 DALI Ballast and LED

Smart lighting systems are implemented with luminaires attached to a controllable ballast. For this experiment, it was designated with LED attached to a commercial, DALI-supported ballast, as this provides a reliable power source to ensure constant illumination. The chosen DALI ballast was the LTSYS LED Driver, due to availability and ease of access from a local distributor. The LED driver was ensured to operate under DALI compliances. DALI ballasts provide power regulation from 220V mains to supply LED luminaires, typically supporting 12V LED.

Testing was carried with a 0.25 W signal LED chosen to simplify the testing process. However, it does not hindrance the usage of power LEDs as the LTSYS Driver has a maximum output power of up to 75 Watts.

3.2 Software Overview

Before the discussion of the software, a summary for the whole system of the DALI protocol demonstrator is important to be emphasized. Each layer of subsystem contains peripherals that serve as the building blocks for the system. The layers of the software specifically designed for the DALI protocol demonstrator are modelled with the Open Systems Interconnection (OSI) model, a framework of description of the constitution of heterogeneous networks, for the definition of standard protocols [23]. Thus, the OSI framework is suitable to constitute the layers of the proposed DALI Protocol Demonstrator. Note that not all the seven layers of OSI are describable in this protocol.

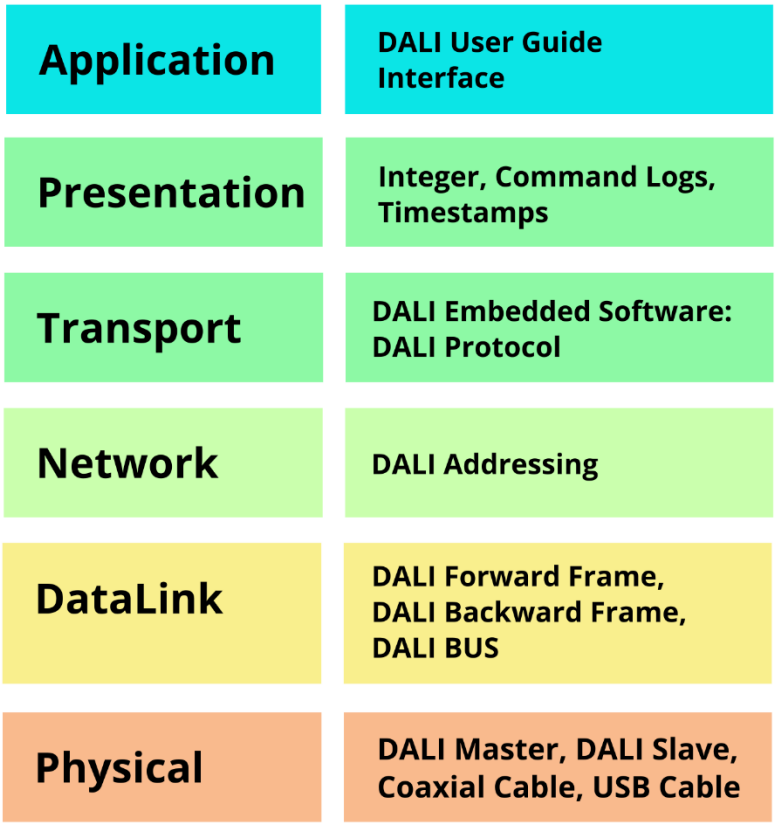


Figure 11. *OSI Architecture of DALI Lighting Control System*

Each of the architecture's layers of DALI lighting control system is described according to the definitions of each layer, referencing from a journal by Zimmermann [23].

Application - The application layer is the topmost layer in the architecture, which directly faces end users by providing distributed information service. The designated application layer of the DALI Protocol is the control command centre, which is a designed Guide User Interface (GUI).

Presentation – The presentation layer contains any method of data conversion to a presentable layer. The DALI Protocol transmits and receives data in the form of serial-readable integers, which represent the command and the address for the forward frame, and binary representation for the backward frame received.

Transport – The transport layer contains methods of conversion of data to packets. The embedded software converts integer values of selected command, inputted from the GUI, and convert them to array-stored binary.

Network Layer – The network layer comprises a subsystem of logical addressing of designated destinations of the packets. The assignment of addressing the ballasts falls into the category of this layer.

Data Link – Data Link layer establishes the functional and procedural purposes to maintain physical connections between network entities. DALI frames of forwarding and backward frames are the supporting data links of the protocol.

Physical – The bottom layer of the physical layer provides mechanical, electrical, functional, and procedural characteristics by means to establish physical connections. DALI physical layers are established through DALI master, DALI slaves, and connection cables.

3.2.1 DALI Master Software

DALI master is embedded with software written in "C" programming language. The chosen embedded operation system is MBED OS, due to familiarity with working with this operating system. A library of DALI.h was created to ensure the modularity and transferability of a program; hence, this practice is also beneficial to the future distribution of the software as open-source software.

Although better explained with the code directly, this section outlines the general operation of the software. The transfer code of the software encodes inputs of integer values and converts the integer to binary. Each of the binary bits is represented with two bits, where 0 is represented ‘1’ and ‘0’ and the value of 1 is represented ‘0’ and ‘1’ respectively. An array buffer stores the entire data packet of an address or command. Additionally, start bits and stop bits are appended to the packets to fulfil the requirements of a DALI packet. The process of sending the transmitted data is sent within intervals of 433 us for every bit change, as this is the length of one bit to ensure 1200 bit/s.

The receive algorithm of the software decodes the backward frame from the DALI circuitry after a delay of the DALI TX frame. After the query command is executed, the received part of the code is initiated. The interrupt is set to capture falling and rising edges of the data stream, as each of the intervals will be timed to ensure accurate timing for each bit measurements. The timing values are converted to binary values consecutively after each bit transitions. For timing around 433us, the value of the bit is single, and timing 833us corresponds to the double value of the bits. Now, the Manchester encoded bits are obtained, and the data stream will be converted into an 8-bit value.

3.2.2 DALI Guide User Interface (GUI)

As the name suggests, Guide User Interface (GUI) was designed to guide end-users to operate DALI protocol within an operable PC, allowing end-users to set the power to luminaires. PC serial to a microcontroller is utilised as the communication line to send and read data of DALI protocol. The DALI GUI was made in python programming language, utilising a practical library of Tkinter which aids the development process. Hence, python-dependent libraries of Tkinter, Pandas, and Openpyxl are required to be installed beforehand. Some of the features of the GUI software can be seen in figure 12.

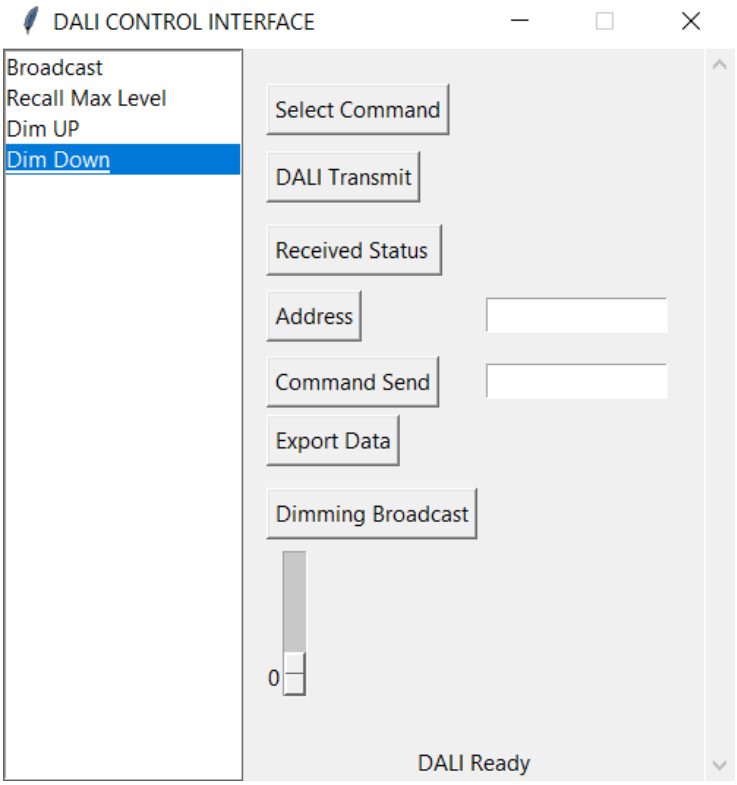


Figure 12. *DALI Control Guide User Interface.*

To operate the GUI, one must connect the USB port of the microcontroller to the operating PC and designate the USB communication port (e. g. COM5) within the python file of DALIGUI.py. The GUI software implements a look-up table to select and send commands, which stores up to 255 commands of DALI, providing ease to users for not having to read the manual. Each command label represents the corresponding integer of the value of the command according to DALI standards. The dimming slider to set the illumination power of the attached LED is operable within the GUI, allowing the user to choose the percentage of the lamination power. The “Received Status” button is to read serial any response from the status bar on the bottom of the GUI updates upon any changes of status or received decoded packets from the DALI ballast.

	A	B	C	D	E
1	**Time**	**Command**	**Address**	**Query**	**Power**
2	06:26:33	144	255		
3	06:26:33	144	255		
4	06:26:33	200	255		
5	06:26:33	008	255		
6	06:26:33	004	255		
7	06:26:33	008	255		
8	06:26:33	004	255		
9	06:26:33	008	255		
10	06:26:33	004	255		

Figure 13. *DALI Control Data Logger Xlsx File Example*

Necessary lighting control data to be kept on record is featured with an additional data export to the excel table, able to be executed by the "Export Data" command. The data is presented in an excel table of recorded timestamps labelled to a log of executed commands. This feature is favourable upon further data analysis of power consumption and security aspects of lighting control.

3.3 Workflow

Research is done step by step from the software development to the hardware design. Upon looking at the literature review, this project was initiated by creating the flow of the program of the microcontroller that encodes and decodes Manchester packets. Alongside the development of the program, it was tested with an available development board. The research was done to analyse various circuitry of the DALI interface; hence the most implementable circuitry was prototyped to a breadboard. The breadboarding process accelerated debugging and board improvements. Debugging process was conducted with an oscilloscope alongside interfacing the GPIO pins. Before the development of GUI, a terminal application was used to send and debug DALI commands. The first PCB iteration was designed after the breadboard prototype has successfully sent basic commands to the DALI ballast and luminaire. Furthermore, the second iteration of the PCB was manufactured. The developed GUI was established and tested with serial communications to the PCB prior to testing the whole system. DALI protocol was tested by controlling commercial LED ballast with important commands reviewed from table 2, which are the following experiments,

- Embedded DALI software tests
- Broadcast power on and off test
- Logarithmic dimming Test
- Query test

4 Results

The testing method is to demonstrate DALI as a commercially working system to be operable with DALI commercial ballasts. The subsystems of DALI are tested to ensure its operation integrated system. As a result, each of the subsystems was gradually tested in each successful state of operation, with progressive improvements for each stage of development before proceeding to further stages.

4.1 Embedded DALI Software Test

Before demonstrating the whole interconnected components to set up DALI Protocol, the capability of the microcontroller is necessary to be tested to ensure the NUCLEO-F446RE operations to transmit and receive Manchester-encoded data packets. The following code snippet is an example of an input command to test the DALI library, as the input was operated within a terminal software.

```
/// Main Function starts here //////////////
int main()
{
    ///Initialise DALI Pins////
    DALITX DALI(D7, D8);
    while (true)
    {
        int cmd;
        printf("Enter an integer: ");
        scanf("%d", &cmd);
        DALI_RECEIVE();
        DALI.DALI_SENT(255,cmd);
    }
}
```

Furthermore, this test confirms the feasibility of the complete development board to transmit the DALI protocol. An oscilloscope was used to measure the transmit data of the transmitting pins, probed from the GPIO D7 pin of the NUCLEO-F446RE.

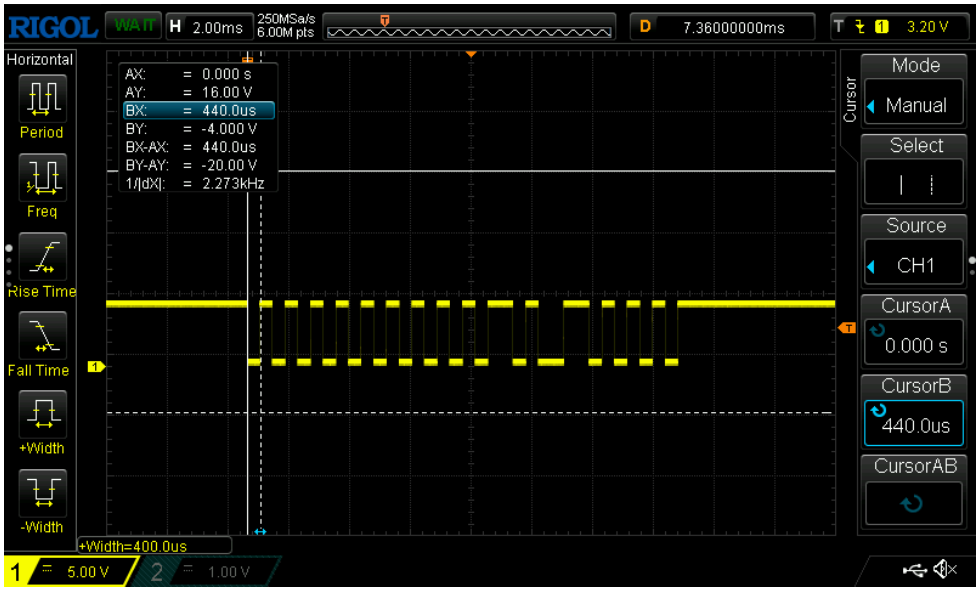


Figure 14 *DALI Forward Frame Simulation Oscilloscope Trace*

From the oscilloscope trace in the figure, by measuring the timing with cursor, it is confirmed that the timing for each bit is correct for the fashion of DALI Protocol, which was around 417 µs. The command tested above is the "144" and the broadcast addressing of "255". Furthermore, various sets of commands were tested to ensure the repeatability of the experiment with different variables.

4.2 Embedded DALI Software Data Receive Test

Beforehand, to decode a backward frame from DALI ballast, a simulation to create a pseudo-response was created. A microcontroller of NUCLEO-F411RE was selected to generate a DALI backward frame from GPIO pin, which sends the data of the pseudo-response of a DALI ballast. The NUCLEO-F466RE serves as the receiver of the data directly from GPIO pin; subsequently encoding it to integer values. Testing operations to communicate with the microcontroller are established by a serial terminal application within the PC.

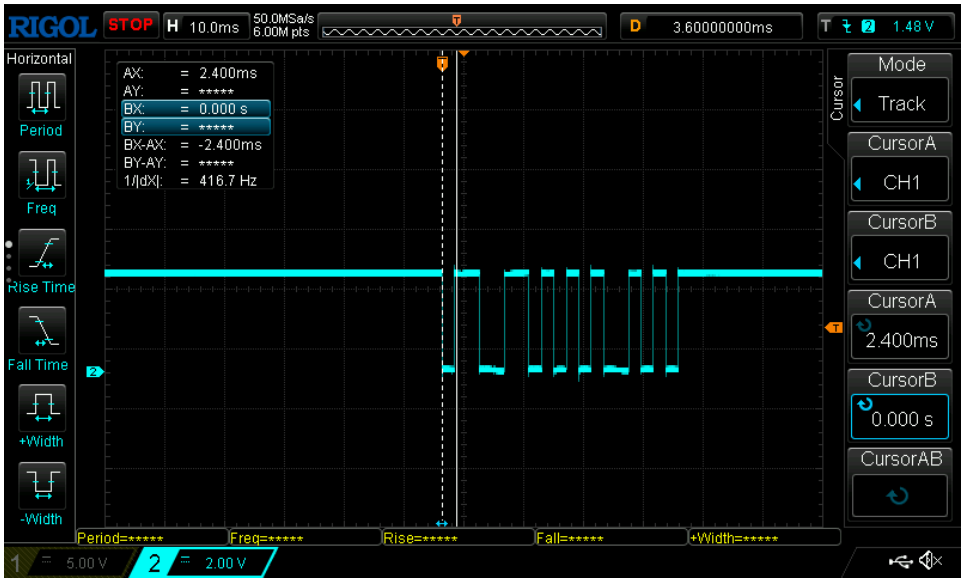


Figure 15. *DALI Backward Frame Simulation Oscilloscope Trace*

From the instances of figure 15, the transmitter microcontroller has successfully sent an example backward frame of 101000011, and the correctness of the timing is observed with the cursor function of the oscilloscope. Observed from the testing, the embedded receive software microcontroller has successfully decoded backward frames. Conversion from binary to integer is done within the embedded software.

4.3 DALI Protocol Demonstration Tests

To demonstrate the whole subsystems to a functional level, communication tests with DALI ballast are conducted. After each of the subsystems of the DALI master was tested and debugged to ensure the operation-ready status, the master is connected using DALI bus with the LTSYS DALI ballast. Prior to testing, the following set-up configuration is shown in figure 16.

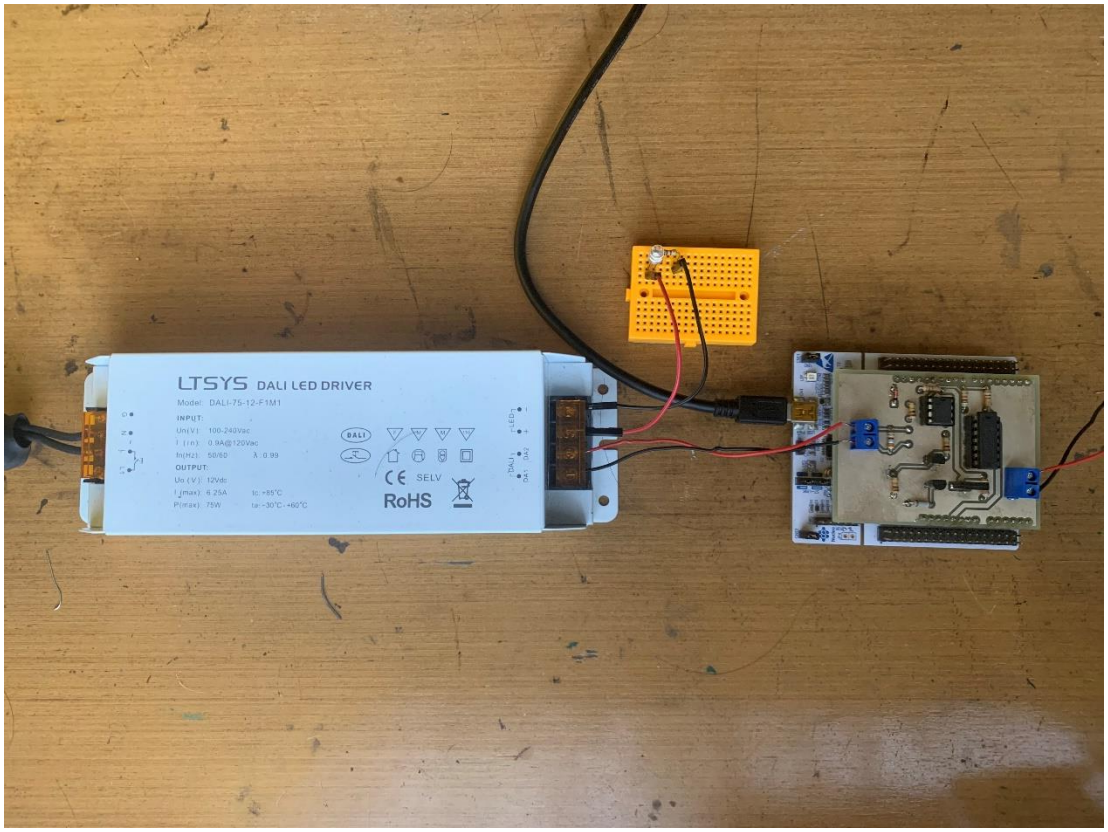


Figure 16. *DALI Demonstrator Testing Setup with DALI-Supported Ballast*

4.3.1 Broadcast Power Command Test

The experiment is carried out by operating the DALI GUI and sending a forward frame within a functional PC. Address of "255" corresponds to the broadcast addressing, targeting all ballast attached to the DALI master, bypassing all addresses assigned. Command of "008" corresponds to, turning on the attached luminaire to the maximum power provided. Followed by the command of "000" which is the command for extinguishing the luminaire. Measured with a voltmeter, the fluctuations of voltage occur when power commands are executed. By this, ballast can control the supply power to the LED.

4.3.2 Logarithmic Dimming Test

Slider control of the GUI was piloted to set the dimming value of the luminaire attached. The test is conducted to broadcast the DIRECT ARC POWER CONTROL command and uses the slider to control the luminaire from 0 to 255 value incrementally. The luminaire lighting intensity was observed, and it could be seen that there was an incremental rise in lighting intensity. This test is significant to control the luminaire for the purpose of power-saving and scene conditioning.

4.3.3 Query Test

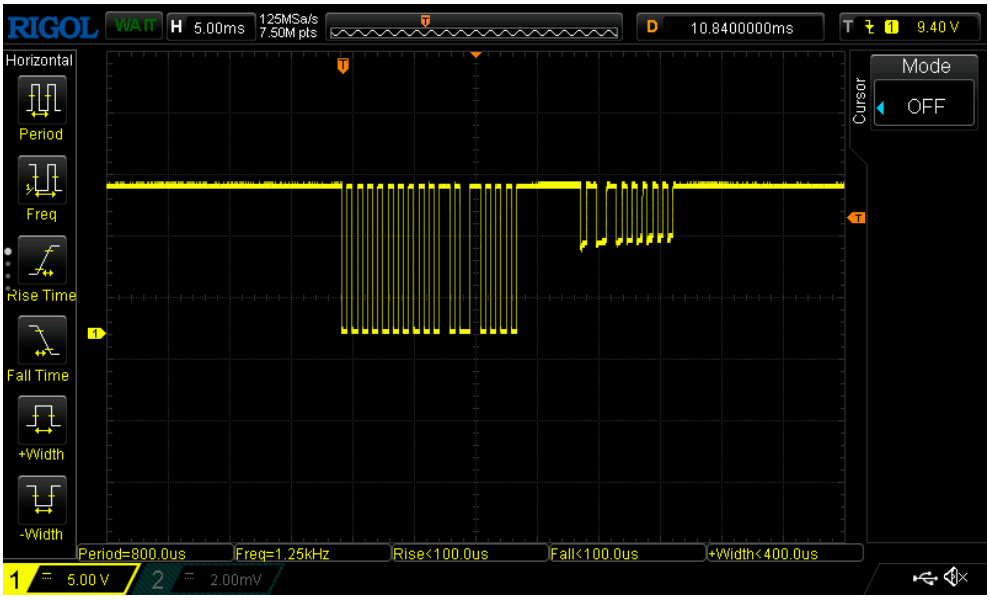


Figure 17. *DALI Forward Frame Query Followed by Backward Frame.*

DALI bi-directional communication must be ensured to follow the established standard. QUERY STATUS command of “144” is transmitted with broadcast addressing to enquire STATUS INFORMATION of the luminaire, with each byte corresponding to Status of control gear, Lamp failure, Lamp arc power on, Query: Limit error, Fade Running, Query Reset State, Query Missing short address, and Query: Power Failure. From the test, the answer in the form of a backward frame was received from the DALI ballast to the terminal application in pc.

A hardware-related problem is found. The results of this test show inaccuracy of the encoded packets from the ballast. Timing from the commercial DALI ballast is shown to have length inaccuracies deviating from 370 µs to 700 µs. Furthermore, margins of error in timing are required to compensate the timing as a workaround solution within the interrupt-driven codes.

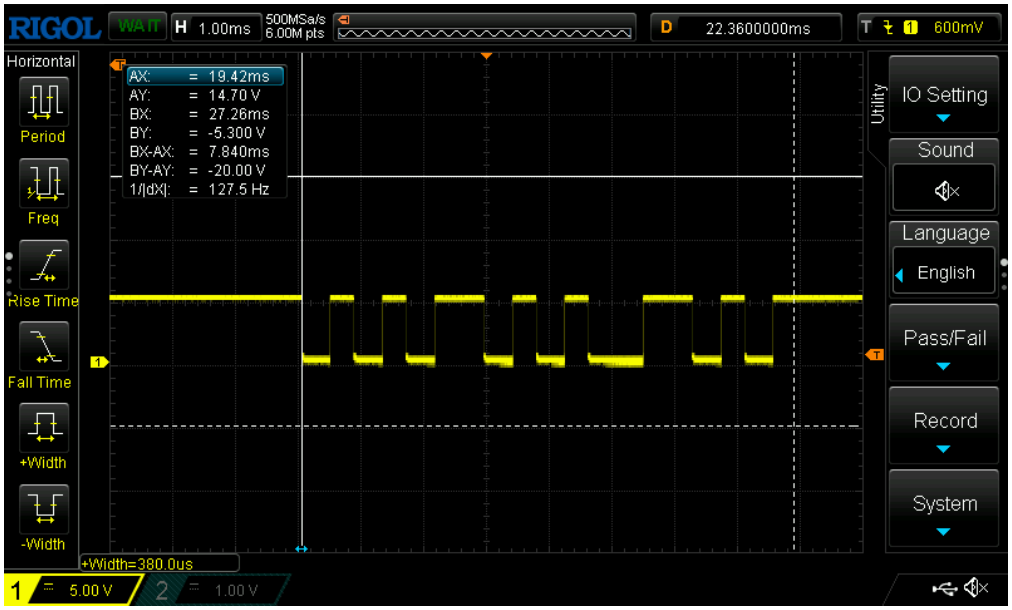


Figure 18. *DALI Backward Frame*

Decoded by the microcontroller, the following binary value is the response of the backward frame when there was no mounted luminaire.

Table 4*. DALI Backward Frame Corresponding Query Answers Decoded from Figure 18.*

Order	Bit Response	Query of Status Information	Answer
1st	1	Status of control gear	not OK
2nd	1	Lamp failure	not OK
3rd	0	Lamp arc power	OFF
4th	0	Query: Limit Error	OFF
5th	0	Fade Running	OFF
6th	1	Query: “RESET STATE”?	Yes
7th	0	Missing short address?	No
8th	0	Query: “POWER FAILURE”?	No

5. Discussion and Evaluation

The experimental results of this project have proven the bare minimum capability of using a microcontroller to communicate with DALI Ballasts, under the compliance of DALI Protocol. It has successfully conjoined the bi-directional transfer and receive function of DALI, proposing a lighting control system solution with deployable features of GUI for end users. However, the GUI has not yet been able to read the backward frame from the serial connection as the readable backward frame is only at the terminal software.

The capability of demonstrating DALI protocol within a data-logging system exposes the potentiality to deliver parameters on the statuses of luminaire and power consumption, making DALI a feasible solution for power saving and maintenance, resorting to sustainability and longevity of lighting systems. This shows DALI protocol's prospects to integrate lighting systems with Smart Grid approaches. Additionally, the capability of operating DALI protocol within a centralised computing system suggests the idea of automation in lighting systems. Ideas such as timed commands can be automated within the python GUI. Light intensity measurements could be implemented as a feedback control system, resulting in unmanned operations of lighting systems. The customisable time-dimming feature operation of DALI is the pilot of energy savings, as it is suggested within this paper [5].

However, on the microcontroller level, the infancy of the embedded program is apparent due to the lack of power-saving features, such as the sleep function. Sleep states of the microcontroller possibly eliminate the redundant tasks and processes, focusing only to execute the given command when other processes are idle. It is recommended for further iteration of the embedded software to implement sleep functionality. Moreover, the program embedded in the microcontroller is still missing connection links, such as decoding to their corresponding queries and converting them into readable data acceptable by the GUI.

The current proposed solution for the DALI adapter circuitry might not be the penultimate solution for the DALI interface circuit. Observing from current DALI boards in the market, e. g. AN11175 is implemented with a bridge rectifier on the input. A possible function of the bridge rectifier is possibly to ensure that DALI is not affected by polarity changes of the bus. Protection from overvoltage or noise occurrence to the DALI setup is required to be experimented with to show the reliance of bus voltage of 12V and the designated transmission bit. This would be an advantageous testing method to be carried out to present the durability of DALI, which uptakes this protocol as a feasible solution in industries and buildings. Hence it marks the proceeding step, which is to select a better microcontroller in terms of price and power efficiency.

The microcontroller of the STM32L072 series is proposed as a feasible solution to replace the NUCLEO-F446RE development board, narrowing the selection to STM32 microcontrollers for adaptability with CUBEMX IDE, a more advanced IDE compared to MBED. The proposed microcontroller has 48MHz of max frequency and a memory size of 128Kb [24]. Obtained from the memory consumption of the embedded software of 47Kb, this memory size would

suffice. To conclude, this microcontroller has the prerequisites necessary for the system, such as UART and USB connection for serial communications, and interrupt-enabled pins.

In terms of installation in complex configurations of lighting systems, addressing wizards is necessary for the system to auto-assign the address for each DALI ballast, as the protocol is supported with commands to assign addresses. This algorithm to sort and assign addresses should be implemented within the embedded level of the software. The implementation of the addressing wizard algorithm is to benefit assigning of numerous ballast addresses if it were to be implemented in a building with complex lighting systems.

Although this project has demonstrated a working GUI, it is still on the level of a prototype, not yet on fully executable stand-alone software. Purposefully, the GUI must be compiled on an executable without the help of IDE. Evaluation on the interface indicated room for improvements for the software to a more user-friendly level; to increase appeal to the market.

5.1 Power Consumption Analysis

Accessing the power consumption during the operation of the device is necessary as a measurement for a power-saving solution, as this device would have continuous operation when installed. Each of the DALI master subsystems of the DALI circuitry current draw is measured with the utilisation of readings from the variable power supply and a multimeter.

Table 5. *Power Consumption Approximation of Each Subsystems of DALI Master*

Subsystem	Current (mA)	Voltage Supply
DALI Circuit PCB	30.0	12V
NUCLEO F446RE	46.1	5V

From current measurements in table 5, the calculated power consumption during operation is around 0.6 Watts. Note that the PC from the system is not included in the power calculation. Obtaining from the datasheet of the STM32F446RET6 from the development board, the microcontroller's typical current consumption (with all peripherals enabled and external clock cycle of 180 MHz) is around 72mA in run mode, and when entering sleep mode, it reduces to 51.2 mA [24]. In contrast to the proposed microcontroller of the STM32L072 series, power consumption is greatly reduced to around 7mA (with 32MHz run mode), and 2.1mA during

sleep mode [25]. For the proposed final design, this could reduce power consumption in processing by approximately 90%.

One of the predecessors of this prototype, an example of power consumption taken from the NXP DRM0004's design, draws a current of 500mA for supplying approximately 12V the power for both master and slave [4]. The resulting power is 6 W, with our DALI master consuming only 3 Watts. Therefore, it is safe to estimate that the whole system power consumption is comparable, thus sufficiently being more efficient, albeit the final decision to improve the selection of microcontroller has yet to be suggested. However, the power consumption calculation of the slaves is not included, thus making the comparison not on par, but noteworthy regardless. Furthermore, a suggestion to compare this prototype's power consumption with other available solutions is an intriguing topic to be discussed.

5.2 Cost Analysis

Minimum cost production is appealing in the current market of lighting systems, specifically in terms of cost reduction for mass production and upfront costs. As an analysis to compare affordability with currently available solutions, each subsystem of the DALI master is taken into consideration cost-wise. Note that the final calculations would include cost services from PCB manufacturing, and price per components are taken from bulk order price of 1000 per batch. Moreover, the price summation on USB cable and copper wires is excluded. The following list is the bill of materials required to build DALI master.

Table 6. *DALI Master Demonstrator Bill of Materials, Price Acquired from Octoparts*

Item	Quantity	Price (USD)	Total price in 1000-unit order quantity (USD)
NUCLEO-F466RE	1 Unit	14.430	14.43
LM393P	1 Unit	0.920	0.441
74HC04	1 Unit	0.650	0.252
Resistors 0.25W	9 Units	1.080	1.080
BD135 Transistor	1 Unit	0.302	0.170
BC337 Transistor	2 Units	0.400	0.085
Capacitor 0.22nF	1 Unit	0.960	0.368
BZX55 Zener Diode	1 Unit	0.210	0.043
Block Terminal 2 Pins	2 Unit	1.120	0.379
6DIP 8 Pin Socket	1 Unit	0.097	0.081
DIP 14 Pin Socket	1 Unit	0.240	0.108
Male Headers	1 Set	0.685	0.600
	Total Price	**21.094 USD**	**18.037 USD**

With the summation of PCB manufacturing service, which is around USD 4.00 per PCB, the total cost to produce one DALI master in bulk production is about USD 22.00. This price is considerably lower than the proposed contenders from section 2.6, resulting in this as a relatively low-cost solution for prototyping lighting systems. Further cost reductions are possible through the implementation of the proposed microcontroller of the STM32LO72 series, which costs about USD 3.10 per chip, alongside the onboard implementation of the microcontroller to eliminate redundancy and unnecessary components.

6. Conclusion

This project has delivered a functional DALI protocol demonstrator prototype by utilising a microcontroller. DALI protocol is studied and successfully implemented in a DALI driver master. Testing is done to operate various DALI protocol commands to establish communication with commercial ballasts and controlling luminaires. Observed results from the evaluation of power and cost of the device shown that this solution is a feasible competitor to current solutions of DALI evaluation boards.

The demonstration process is operated with a designated GUI specifically for the demonstrator. Data logging features are implemented within the software to ease assessment on power consumption analysis. Therefore, the whole system approach is intended as an application of smart grid in lighting systems, alongside maintaining efficiency in power consumptions.

To improve the usability aspect of the DALI master, this project offered an integrated solution of the DALI interface and accessed value-line microcontroller selection within one PCB system. This is to ensure that the solution is to increase effectiveness for further progress in lighting system control.

7. Further Work

The final design is still prematurely manufacturing in contrast to claiming that the prototype of this DALI demonstrator is industrial-standardized. Iterations to create a better prototype are yet to be done upon reaching the industrial level for design usage. With extra additional time is given, further testing of all the commands and multiple address is a complementary step to ensure the standards increasing the reliance of the demonstrator. testing related to safety and noise immunity from industrial or household should be conducted to ensure the DALI master's feasibility. Moreover, to implement data read of DALI backward frame within the GUI software has yet to be implemented, as it is necessary.

The end deliverables of this project have possibilities to be continued by future works, and the following further implementations are proposed.

7.1 DALI GUI Evaluation

Further research on clean user interface design is thus recommended to increase the design aesthetics and ease of operation. Features such as indicators on each status of the ballasts are yet to be implemented to ease the practicality of operations in complex lighting systems.

7.2 DALI Control Automation

The implementation of the GUI in python appends possibilities of studies related to lighting automation control and smart grids implementation. Time-based commands could be executed and stored on the PC. By implementing automation to the lighting control, the power analysis could be obtained. This will contrast the power consumption reduction when DALI is installed. Thus, data analytics and method to analyse power consumption is a lucrative topic to be discussed further, although, the analytics are beyond the scope of this project.

7.3 DALI – Microcontroller Board

A prototype design of the selected microcontroller and DALI circuitry interface combined is proposed with aid of diagram shown in figure 19.

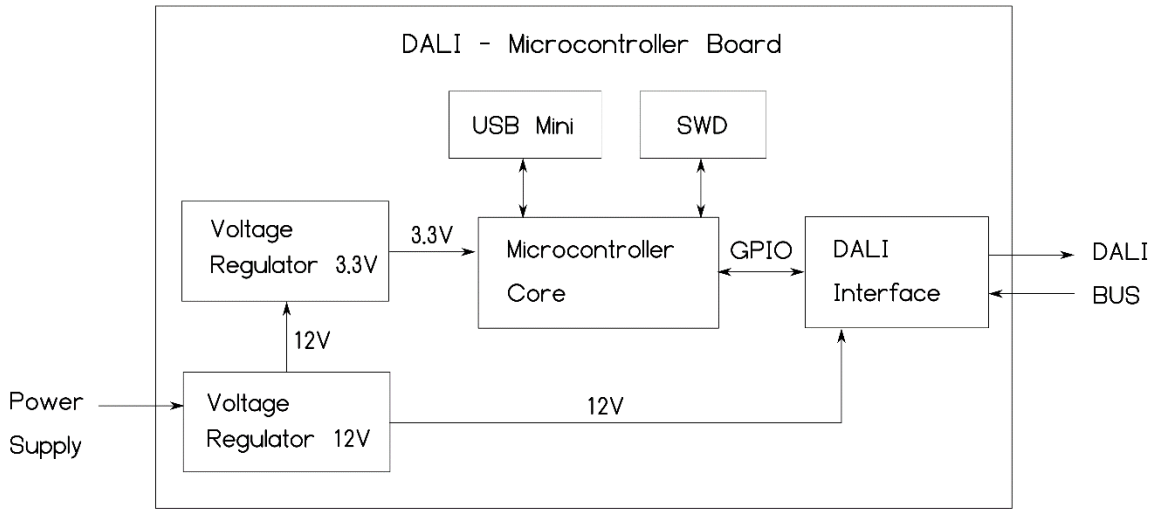


***Figure 19**. Proposed DALI Master Evaluation Board Block Diagram*

The final PCB design of the integrated DALI interface is proposed with peripherals of the board, taken from the requirements based from the results of this report. The utilisation of ARM processor upon the final design board specifically requires a debugging interface of Serial Wire Interface [26]. Therefore, an interface of ARM's Serial Wire Debugging (SWD) component is required on the PCB. Power supply conversion is conducted with voltage cascading voltage regulators, as the DALI and microcontroller require 12V and 3.6V operation voltage, respectively. Further research on the DALI interface with a selection of SMT components should be conducted. Thus, with the support of a professional manufacturing process, the final design purposedly should consume less power with less footprint.

References

[1] Khan N, Abas N. Comparative study of energy saving light sources. Renewable and Sustainable Energy Reviews. 2011;15(1):296-309. DOI: 10.1016/j.rser.2010.07.072

[2] European Commission. Savings and benefits of global regulations for energy efficient products [Internet]. 2021. Available from: https://ec.europa.eu/energy/sites/ener/files/documents/Cost%20of%20Non-World%20-%20Final%20Report.pdf

[3] Lighting – Analysis - IEA [Internet]. IEA. 2021 [cited 18 January 2021]. Available from: https://www.iea.org/reports/lighting.

[4] NXP Semiconductors, "DRM004, Digitally Addressable Lighting Interface (DALI) Unit Using the MC68HC908KX8 Designer" [Online]. Available: https://www.nxp.com/files-static/microcontrollers/doc/ref_manual/DRM004.pdf

[5] Bellido-Outeirino F, Flores-Arias J, Domingo-Perez F, Gil-de-Castro A, Moreno-Munoz A. Building lighting automation through the integration of DALI with wireless sensor networks. IEEE Transactions on Consumer Electronics. 2012;58(1):47-52. DOI: 10.1109/tce.2012.6170054

[6] Ding Q, Liang X, Chen H, Liu M, Yang J. Study on Policy and Standard System of LED Lighting Industry in EU. E3S Web of Conferences. 2020; 194:02016. DOI: 10.1051/e3sconf/202019402016

[7] Wisniewski A. The Calculation of Energy Saving in use Light Management Systems. 2018 VII Lighting Conference of the Visegrad Countries (Lumen V4). 2018;:1 - 4. DOI: 10.1109/lumenv.2018.8521043.

[8] Zhang Y, Zhou P, Wu M. Research on DALI and Development of Master-Slave module. 2006 IEEE International Conference on Networking, Sensing and Control. 2006;:1106-1110. DOI: 10.1109/icnsc.2006.1673307.

[9] ZVEI-Division Luminaires, "DALI Manual," [Online]. Available: https://sitelec.org/download.php?filename=cours/abati/dali/pdf/manual_gb.pdf

[10] Maheshwari S, Kale I. Adiabatic Implementation of Manchester Encoding for Passive NFC System. 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2019;:1615-1618. DOI: 10.23919/date.2019.8714838.

[11] Visconti P, Primiceri P, Cavalera G. Wireless Monitoring System of Household Electrical Consumption with DALY-based Control Unit of Lighting Facilities Remotely

Controlled by Internet. Journal of Communications Software and Systems. 2016;12(1):4. DOI: 10.24138/jcomss.v12i1.86.

[12] Roy J, Rakshit J. Manchester code generation scheme using micro-ring resonator based all optical switch. 2014 9th International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP). 2014;:1118-1122. DOI: 10.1109/csndsp.2014.6923997.

[13] Hsia S, Sheu M, Ciou J. Cost-Effective LED Dimming Driver with Single Chip Design for Smart Lighting System. IEEE Access. 2020;8:141025-141032. DOI: 10.1109/access.2020.3013302

[14] Janse van Rensburg P, Snyders A, Ferreira H. Modeling of Coupling Diversity for Extra-Low-Voltage Power-Line Communication Networked LED Lighting in Smart Buildings. IEEE Journal of Emerging and Selected Topics in Power Electronics. 2018;6(3):1224-1234. DOI: 10.1109/jestpe.2018.2836356

[15] British Standards Institution. BS EN 62386-102:2009. Digital Addressable Lighting Interface. London: BSI; 2009.

[16] Atmel, "Atmel AT01244: DALI Slave Reference Design" [Online]. Available: http://ww1.microchip.com/downloads/en/appnotes/atmel-42071-dali-slave-reference_design_application-note_at01244.pdf

[17] Raggiunto S, Belli A, Palma L, Ceregioli P, Gattari M, Pierleoni P. An Efficient Method for LED Light Sources Characterization. Electronics. 2019;8(10):1089. DOI: 10.3390/electronics8101089.

[18] Hu W, Davis W. Dimming curve based on the detectability and acceptability of illuminance differences. Optics Express. 2016;24(10):A885. DOI: 10.1364/oe.24.00a885.

[19] NXP Semiconductors, "DALI Master Using LPC134x" [Online]. Available: https://www.mouser.com/datasheet/2/302/nxp_OM13046_AN-1189016.pdf

[20] Microchip, "Lighting Communications Development Platform User's Guide" [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/40001717A.pdf

[21] MikroElektronika, "DALI Click". [Online]. Available: https://www.digikey.be/htmldatasheets/production/1398394/0/0/1/DALI-Click.pdf

[22] Millar R, Rowan P, Rowan S, Easdale C, Kakaiya A. Use of 'Digital Addressable Lighting Interface' (DALI) and xAP open protocols to create a cost-effective automated lighting system to improve lighting efficiency in the home [Masters]. University of Glasgow; 2012.

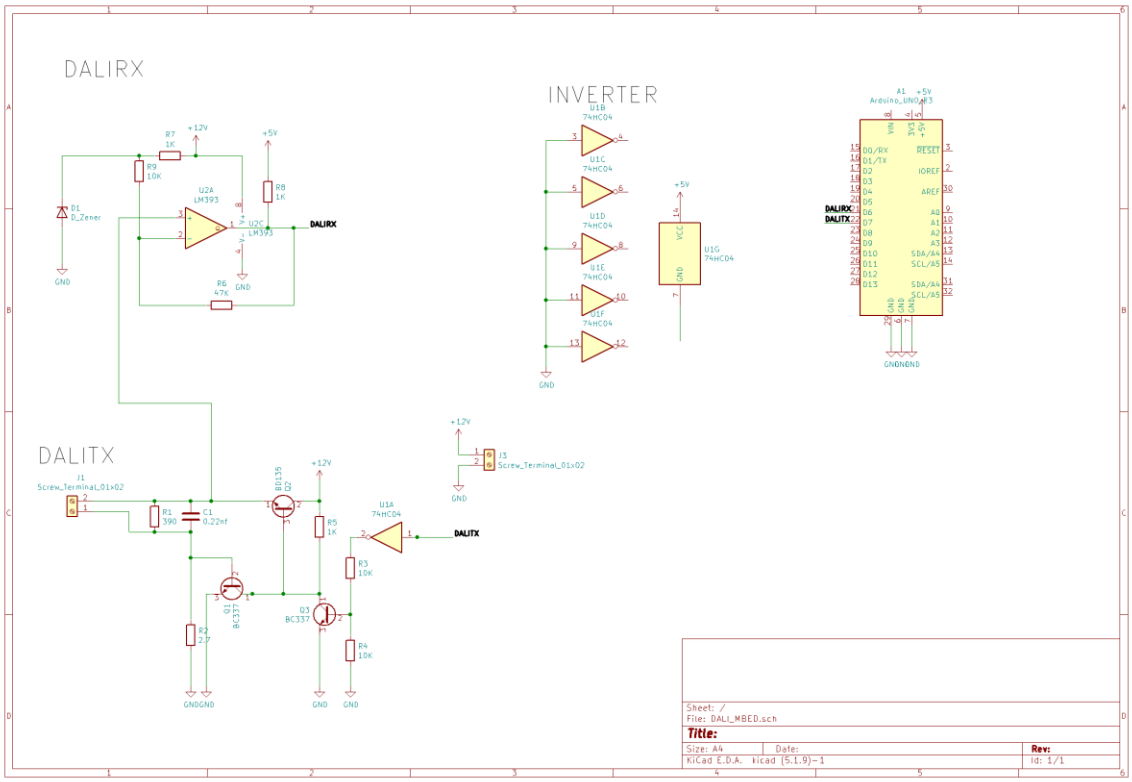
[23] Zimmermann H. OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications. 1980;28(4):425-432. DOI: 10.1109/tcom.1980.1094702.

[24] STMicroelectronics, “STM32F446xC/E Datasheet”, [Online]. Available: https://datasheet.octopart.com/STM32L072CBT6-STMicroelectronics-datasheet-49624919.pdf

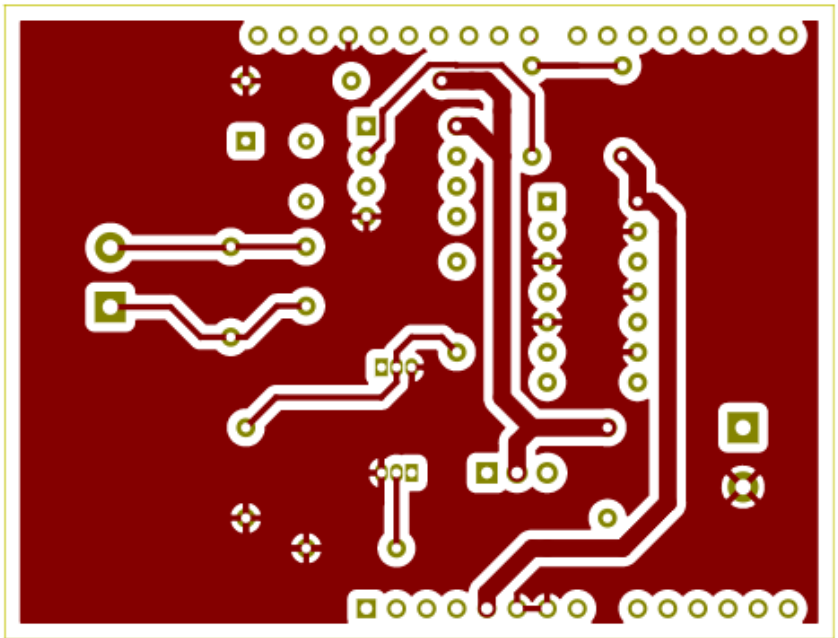
[25] STMicroelectronics, “STM32L072x8 STM32L072xB STM32L072xZ Datasheet”, [Online]. Available: https://datasheet.octopart.com/STM32F446RET6-STMicroelectronics-datasheet-58295083.pdf

[26] Bogdanov L. Multiple Microcontroller Programming Using the SWD Interface. 2020 XXIX International Scientific Conference Electronics (ET). 2020;:1-4. DOI: 10.1109/et50336.2020.9238282.

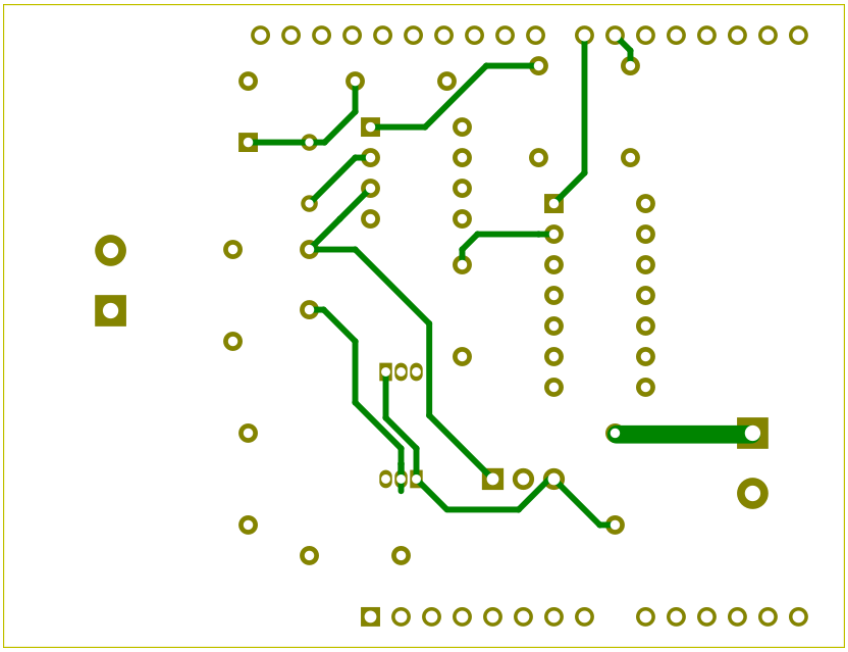
Appendices



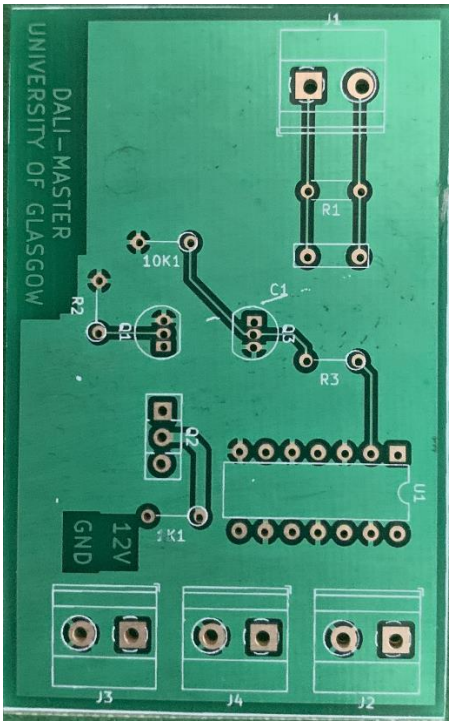
Appendix 1. *DALI Interface Schematic with Arduino Shield Pins*



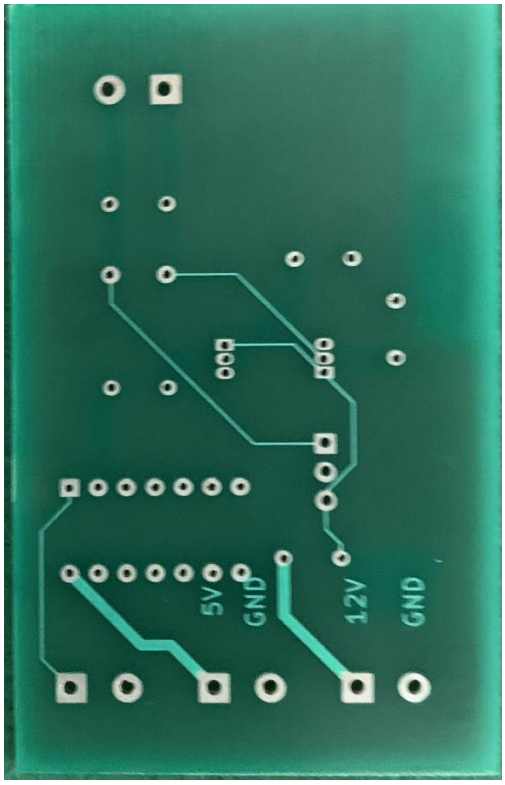
Appendix 2. *DALI Master Second Iteration PCB Top Layer (Not In Scale)*



Appendix 3. *DALI Master Second Iteration PCB Bottom Layer (Not In Scale)*



Appendix 4. *DALI Master Second Iteration PCB Top Layer (Not In Scale)*



Appendix 5. *DALI Master First Iteration PCB Bottom Layer (Not In Scale)*

```
#include "mbed.h"
#include "DALITX.h"

DALITX::DALITX(PinName DALI_TX, PinName DALI_RX) : TX(DALI_TX), RX(DALI_RX) {
    /** configure the rising edge to start the timer */
    TX = 1;
    RX.mode(PullUp);
    i = 0;      //Bitcount
    m = 0;      //Timecount
    j = 0;
}

///write DALI timing
void DALITX::DALIPRINT(){
    for (j = 1; j < 17; j++) {
        printf("%i ", time[j]);
    }
    j = 0;
}


///DALI Forward Frame Manchester Packets command, input in the main//////////
void DALITX::DALI_PACKET(int address, int command){
    int p;
    int ones;
    //Start Bits 1
    man_packets[0] = 1;
    man_packets[1] = 0;
    man_packets[2] = 1;
    //Stop Bits

    for(p=0; p<8; p++){
        //For Address Conversion
        ones = address >> 7 - p;   //Shift the bit to the left
        ones &= 0x01;
        if(ones == 1){
            man_packets[(p*2)+3] = 0;
            man_packets[(p*2)+4] = 1;
        } else {
            man_packets[(p*2)+3] = 1;
            man_packets[(p*2)+4] = 0;
        }
    }

    for(p=0; p<8; p++){
        //For Comman Conversion
        ones = command >> 7 - p;   //Shift the bit to the left
        ones &= 0x01;
        if(ones == 1){
            man_packets[(p*2)+19] = 0;
            man_packets[(p*2)+20] = 1;
        } else {
            man_packets[(p*2)+19] = 1;
            man_packets[(p*2)+20] = 0;
        }
    }
}
```

```
///DALI SENT frame //////////
void DALITX::DALI_SENT(int address, int command){
    int bit = 0;
    DALITX::DALI_PACKET(address, command);
    for(bit=0; bit < 36; bit ++){
        TX = man_packets[bit];
        wait_us(417);
    }
    TX = 1;
    TX = 1;
    }
```

Appendix 6. *DALITX.cpp*

```
#include "mbed.h"
#ifndef _DALI_H_
#define _DALI_H_
class DALITX{
public:
    DALITX(PinName DALI_TX, PinName DALI_RX);
    void DALI_PACKET(int address, int command);
    void DALI_SENT(int address, int command);

    void DALI_PACK(int address, int command);

    void DALI_RECV();

    void DALI_TEST();

    void DALI_ARR_TO_INT();
    void DALIPRINT();

private:
    DigitalOut TX;
    InterruptIn RX;
    int i;
    int highcount;
    int lowcount;
    int arr[17];                //Create an array of 8 se
    int bit[8];
    int time[17];
    int m;
    int j;
    int k;
    int n;

    int man_packets[20];
    int address;
    int command;
    int dali;
    int dali_tx;

    volatile int startbit;
    int addbit;                 // Int address to send bit to array
    int addbit_low;
    int addbit_high;
    int packet_fin;
};

#endif
```

Appendix 7. *DALITX.h*

```
#include "mbed.h"
#include "DALITX.h"

Serial pc(USBTX, USBRX);

volatile int addr[3];
volatile int cmdd[3];
volatile int address;
volatile int command;

//using namespace std::chrono;

// Initialise the digital pin LED1 as an output
DigitalOut led(LED1);

//Timer to capture high and low time
Timer high;
Timer low;
Timer timeout;
Timer t_skip;


volatile int highs = 0;
volatile int lows = 0;
//Store bits, bit, and temp
volatile int bits[54];
volatile int bit[18];
volatile int temp[54];
volatile int data[9];

volatile int query[9];
volatile int i;
volatile int k;
volatile int l;
volatile int lock = 0;
volatile int init = 0;
volatile int elapsed = 0;

//DALI RECEIVE Pin declaration
InterruptIn RECV(D6);

//Interrupt Declaration
void fall_irq(void);
void rise_irq(void);
void fall_end_irq(void);

//Init detect first interrupt, add lock flag
void fall_irq(){
    low.start();
```

```
    lock = 1;
    RECV.rise(&rise_irq);
}

//Interrupt for rise detection, store captured time in bits[]
void rise_irq(){
    low.stop();
    high.start();
    if (low.read_us() > 1) {
        bits[i] = low.read_us();
    }
    i++;
    low.reset();
    RECV.fall(&fall_end_irq);
    timeout.start();
}


/// Interrupt for fall detection, store captured time in bits[]
void fall_end_irq(){
    low.start();
    high.stop();
    if (high.read_us() > 1) {
        bits[i] = high.read_us();
    }
    i++;
    high.reset();
    RECV.rise(&rise_irq);
}

void delay_irq(){
    t_skip.start();
    init = 1;
}

/// DALI Receive, time conversion to bit values
void DALI_RECEIVE(){
    if(lock == 0){
        RECV.fall(&fall_irq);
    }

    //while (true) {
        if (timeout.read_us() > 27200){

            k = 0;
            for(int j=0;j<54;j++) {
                if(bits[j] < 5000) {
                    if (bits[j] > 100) {
                        temp[k] = bits[j];
```

```
                    ///For Debugging Purposes
                    ///printf("%d ",temp[k]);
                    k++;
                }
            }
        }

        // Convert time to bits, if larger than 750 us, it will be convert
ed to two values, if it is less than 500, it is converted to 0
        k = 0;
        for(int j=0;j<50;j++) {
            if ((j % 2) == 1) {
                if (temp[j] > 0) {
                    if (temp[j] < 900){
                        if (temp[j] > 750){
                            bit[k] = 0;
                            bit[k+1] =0;
                            k++;
                            k++;
                        }
                        else if (temp[j] < 500){
                            bit[k] = 0;
                            k++;
                        }
                    }
                }
            }
            else if ((j % 2) == 0) {
                if (temp[j] > 0) {
                    if (temp[j] < 900){
                        if (temp[j] > 750){
                            bit[k]   = 1;
                            bit[k+1] = 1;
                            k++;
                            k++;
                        }
                        else if (temp[j] < 500){
                            bit[k] = 1;
                            k++;
                        }
                    }
                }
            }
        }

        ///For Debugging Purposes
        for(int j = 0; j < 54; j ++) {
            ///printf("%d ",bit[j]);
```

```
            }


            k = 0;
            printf("DALI Response: ");
            for(int j=34; j<52; j++) {
                if (bit [j] == 1 && bit[j + 1] == 0) {
                    data[k] = 0;
                    printf("%d ", data[k]);
                    k++;
                    j++;
                    }
                else if(bit [j] == 0 && bit[j + 1] == 1) {
                    data[k] = 1;
                    printf("%d ", data[k]);
                    k++;
                    j++;
                }
            }

            printf("\n");
            // Reset Counters
            k = 0;
            i = 0;
            l = 0;

            for(int j=0;j<54;j++) {
                temp[j] = 0;
                bit[j] = 0;
            }

            lock = 0;
            timeout.stop();
            timeout.reset();
            elapsed = 0;
    }

}


/// Main Function starts here //////////////
int main()
{
    DALITX DALI(D7, D8);
    while (true)
    {
        int cmd;
        printf("Enter an integer: ");
        scanf("%d", &cmd);
```

```
        DALI_RECEIVE();
        DALI.DALI_SENT(255,cmd);
    }
}
```

Appendix 8 . *Main.cpp*

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 23 10:07:34 2021

@author: Wikara Gunawan
"""
import serial
from tkinter import Tk, Label, Button, Scrollbar, Listbox, Entry, Scale
import tkinter
import pandas as pd
import time
from datetime import datetime

#Configure Baudrate for serial communication, usually baudrate = 9600
#ser = serial.Serial( port = "COM5", baudrate = 9600, )

dalidata = {'Time': [],
        'Command': [],
        'Address': [],
        'Query': [],
        }


class DALI_GUI:
    def __init__(self, master):

        self.now = datetime.now()
        #Current Time Obtain
        self.current_time = self.now.strftime("%H:%M:%S")


        #Create Pandas DataFrame
        self.dali_df = pd.DataFrame(dalidata, columns = ['Time', 'Command', 'A
ddress', 'Query', 'Power'])

        global leftside
        self.should_stop = False

        self.address = 0
        self.command = 0
        self.cmd = 0
        self.addr = 0

        self.command_list = ["Broadcast", "Recall Max Level","Dim UP", "Dim Do
wn"]
        self.command_enum = ["255", "005", "004", "003"]
        self.master = master
        master.geometry("500x500")
```

```
        master.resizable(0, 0)
        master.title("DALI CONTROL INTERFACE")

        #self.label = Label(master, text="DALI Software Interface")
        #self.label.pack()

         #send button
        self.transmit_button = tkinter.Button(master, text="DALI Transmit", co
mmand=self.DALI_Send)
        self.transmit_button.pack()
        self.transmit_button.place(x=180, y=70)
        #receive button
        self.receive_button =tkinter.Button(master, text="Received Status ", c
ommand=self.DALI_Receive)
        self.receive_button.pack()
        self.receive_button.place(x=180, y=120)
        #custom command
        self.cmd_custom = tkinter.Button(master, text="Address Input", command
=self.DALI_Custom_Addr)
        self.cmd_custom.pack()
        self.cmd_custom.place(x=180, y=165)
        #input command
        self.input_cmd = Entry(master,width=15)
        self.input_cmd.pack()
        self.input_cmd.place(x=330, y=215)
        #address Custom
        self.addr_custom = tkinter.Button(master, text="Command Send", command
=self.DALI_Custom_Cmd)
        self.addr_custom.pack()
        self.addr_custom.place(x=180, y=210)
        #Input address
        self.input_addr = Entry(master,width=15)
        self.input_addr.pack()
        self.input_addr.place(x=330, y=170)
        #Select
        self.selectbutton=Button(master, text="Select Command", command=self.g
et_command)
        self.selectbutton.pack()
        self.selectbutton.place(x=180, y=24)
        #Quit Button
        self.quit_button=Button(master, text="Quit", command=self.quit_status)
        self.quit_button.pack()
        self.quit_button.place(x=180, y=250)
        #Dimmer Slider
        self.dimmer = Scale(master, from_=255, to=0)
        self.dimmer.pack()
        self.dimmer.place(x=160, y=340)
```

```
        self.dim_select=Button(master, text="Dimming Broadcast", command=self.
dim_value)
        self.dim_select.pack()
        self.dim_select.place(x=180, y=300)
        #Side Scroll Bar
        self.scroll=Scrollbar(root)
        self.scroll.pack(side='right', fill='y')

        #Export Time to Excel
        self.quit_button=Button(master, text="Export Data", command=self.expor
t_xlsx)
        self.quit_button.pack()
        self.quit_button.place(x=180, y=250)

        self.leftside = Listbox(master, yscrollcommand = self.scroll.set)
        self.cmdside = Listbox(master, yscrollcommand = self.scroll.set)

        for line in range(len(self.command_list)):
            self.leftside.insert('end', self.command_list[line])


        self.leftside.pack(side='left', fill='both')
        self.scroll.config(command=self.leftside.yview)

        #Side Status Bar
        self.current_cmd = "DALI Ready"
        self.status = Label(master, text = "Status")
        self.status.pack(side='bottom')

    """ Send data to DALI via USB Serial Connection """
    def DALI_Send(self):
        print("DALI ADDRESS + COMMAND SENT")
        self.current_cmd = "DATA SENT"

        message = self.address + self.command
        print(message)
        #ser.write(message.encode())
        #line = ser.read(size=6)
        #print(line)

    def DALI_Receive(self):
        print("DALI RECEIVED STATUS")
        self.current_cmd = "Received Data"
        #line = ser.read(size=18)
        #print(line)

    """ Get Address"""
    def DALI_Custom_Cmd(self):
```

```
        print("DALI CUSTOM ADDRESS STORED")
        self.current_cmd = "Custom Address Stored"
        self.cmd = self.input_cmd.get()
        #ser.write(cmd.encode())
        #line = ser.readline().decode('ascii')
        #print(line)

        self.current_time = self.now.strftime("%H:%M:%S")

        timelog = {'Time':self.current_time, 'Command':self.cmd, 'Address':sel
f.addr}
        self.dali_df = self.dali_df.append(timelog, ignore_index=True)
        print(self.dali_df)

    """ Get Command"""
    def DALI_Custom_Addr(self):
        print("DALI CUSTOM COMMAND TRANSMIT")
        self.current_cmd = "Custom Command Transmitted"
        self.addr = self.input_addr.get()
        #ser.write(addr.encode())
        #line = ser.readline()
        #(line)


    def get_command(self):
        userline=self.leftside.get('active')
        indexed = ["Broadcast", "Recall Max Level","Dim UP", "Dim Down"].index
(userline)
        cmd = self.command_enum[indexed]
        cmdhex = hex(int(cmd))
        self.current_cmd = "Command Selected: " + userline + "   " +  cmdhex

    """ Select Value For Light Dimmer with slider """
    def dim_value(self):
        value = self.dimmer.get()
        light = self.dimmer.get()
        count = 0
        while value > 0:
            count = count + 1
            value = value//10

        if count == 1:
            dim = '0' + '0' + str(light)
        elif count == 2:
            dim = '0'  + str(light)
        else:
            dim = str(light)
```

```
        self.current_cmd = "Lighting Level:" + dim
        cmd = dim
        ##Uncomment to write data dim value
        #ser.write(cmd.encode())
        #line = ser.read(size=10)


    def update_status(self):
        if not self.should_stop:
            self.status["text"] = self.current_cmd
            self.stop_id = self.master.after(1000, self.update_status)

    """ Quit Button """
    def quit_status(self):
        self.should_stop = True
        self.master.after_cancel(self.stop_id)
        self.master.destroy()


    """ Export XLSX of DATA Log, currently only support function from custom c
ommand and address"""
    def export_xlsx(self):
        self.current_cmd = "Exported to Filepath"
        self.dali_df.to_excel(r'C:\Users\Visitor\Desktop\DALI.xlsx', index = F
alse)



#commands initialise starts here
root = Tk()
DALI = DALI_GUI(root)
root.after(500, DALI.update_status)
root.mainloop()
root.after_cancel(DALI.update_status)
root.destroy()
```

Appendix 9. *DALI_GUI.py*