International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 11, ISSN No. 2455-2143, Pages 400-405
Published Online March 2020 in IJEAST (http://www.ijeast.com)

OPTICAL DATA COMMUNICATION USING PYTHON AND ARDUINO µC

Fatehsingh Parab, Department of EXTC, Rizvi College of Engineering, Maharashtra, India

Rahil Shaikh, Department of EXTC, Rizvi College of Engineering, Maharashtra, India

Samved Sangle, Department of EXTC, Rizvi College of Engineering, Maharashtra, India

Prof. Nargis Shaikh, Department of ETRX, Rizvi College of Engineering, Maharashtra, India

Abstract – Optical communication is a communication technology that uses light propagating in free space to transmit AV or telecommunication data overseas. There is always a need for a high data rate between two communicating nodes. For this fiber optic communication technology has been massively developed along with the development of new capacity enhancing components. Over years these components became advanced in optical communication and ultimately, they cost more to which only corporates can afford.
However, our project is still in the developing phase and we have designed just the sender system(node). In this paper, we have framed our idea, path to implementation and observation for optically data transmission between two computers using Arduino µC.
Further, we have discussed how our idea can be efficient with parallel computing using python3 and how it would be a cross-platform solution.

Keywords: **Optical communication, fiber optics, python programming, parallel computing, Arduino µC.**

I. INTRODUCTION

The Arduino microcontroller is used in art and design as an open-source programmable tool to create interactive works. It can drive various peripheral devices that support I2C protocol.
Python is a powerful high-level, object-oriented Programming language created by Guido van Rossum. Our python-based project will communicate with Arduino µC via serial communication from standard USB-COM port. Serial sequentially, over a communication channel or computer bus [1-3].

Our idea integrates the above technologies to transmit data optically with cost-efficient hardware/software.

Here the performance or speed of data transmission is directly proportional to the degree of pipelining available in both the sender and receiver.

II. RESEARCH FINDINGS

To analyze various parameters like delay, data rates, the timeout value for USB-COM, laser intensity.

III. DESIGN OF OPTICAL TRANSMITTER

3.1 USB and Serial communication

Serial is used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.

There are two primary forms of serial transmission: Synchronous and Asynchronous [5].

Arduino µC works on UART (Universal asynchronous receiver/transmitter) [1,5].

The below-given figure is the data frame of asynchronous serial communication

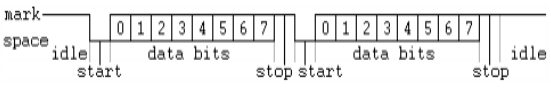


Fig 1 Data frame of UART

3.1.1 Setting Baud rate and necessary USB connections

The baud rate is a measure of the number of bits per second that can be transmitted or received by the UART. Generally, Arduino boards support the baud rate from 300 – 200000. We have kept default baud rate i.e 9600 throughout our experiment.

Arduino works with either USB 2.0 or USB 3.0 but it requires USB type A to B cable to work. We have implemented USB – Serial communication with

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 11, ISSN No. 2455-2143, Pages 400-405
Published Online March 2020 in IJEAST (http://www.ijeast.com)

Arduino µC by installing necessary certified drivers and default serial communication libraries with python3.

3.2 Digital file storage and data accumulation with python3

Data in digital storage is stored either in 1's or 0's with applications of flip-flops in memory chips or with magnetic effects in HDDs [6].

A file with extension as txt, png, jpeg, mp3, etc. All bits are stored at a specific memory address.

Python3 is case sensitive object-oriented language. We are using the inbuilt 'open()' function of python3 with attributes 'rb' to read the file as binary. Without 'rb' attribute python will read the file as default txt file and encodes it, due to this except genuine text file all files will go corrupt. Basically, we're harvesting raw data viz. 1's & 0's.

```
file = open("source.txt","rb")
data = file.read()
```

Fig2 Reading text file as binary with python3

```
file = open("source.png","rb")
data = file.read()
```

Fig3 Reading png file as binary with python3

In Fig2 and Fig2 we have created a variable 'file'. In Fig2 we are reading a text file in binary format (rb) and storing the binary content in variable 'data', similar has been done with png file [7,11].

In these cases, variable 'data' is of datatype 'Array' or 'list'(an array in python). Larger the data, larger the size of an array.

This array has byte value in it. Python reads a file in binary as by grouping total 8 bits and appending its byte value into variable 'data'. Fig4 shows the byte content of the text file.

```
data = open("source.txt",'r') # reads the text data as TEXT
data = data.read()            # store content in var 'data'
print(data)                   # Print content od 'data'
```

```
This is python3
```

```
file_bin = open("source.txt",'rb') # read the text data as binary
file_bin = file_bin.read()
file_bin = [i for i in file_bin]   # restore all byte values into list
print(file_bin)                    # print content od 'file_bin'
```

```
[84, 104, 105, 115, 32, 105, 115, 32, 112, 121, 116, 104, 111, 110, 51]
```

Fig4 Binary data of a text file

3.3 Programing Arduino (Mega 2560 Rev3) for serial communication

Arduino boards are programmed using 'C embedded' language. Most of the code resembles the traditional C code, but there are some exclusive functions that can be used in Arduino IDE only. Any Arduino *µC* has two basic functions: `void setup()` and `void loop()`.

`Void setup()` can be called an initializing function. This function is called once when Arduino boots from the power-off state. This function contains that code that needs to runs at once. Things like pin initialization, declaring a variable, setting baud rate or running a loop code for once.

`Void loop()` function keeps on looping like a 'while loop' this function executes its code repeatedly and won't stop until the board is reset or power down.

In our optical transmitter we have connected a laser of 650nm 5V at pin 9 of Arduino mega 2560 and also has set a counter variable to count total byte transferred.
Variable 'count' is used to check the total no of bytes that were sent by the transmitter.

```
/* code to initialize laser
pin and counter */
int laser = 9;
int count = 1;
```

Fig5 Code for laser module initialization

3.4 Syncing between Python shell and Arduino µC

Synchronization between both python-shell and Arduino µC is necessary because the host(Operating system) on which python-shell is running is operating at clock speed GHz, whereas Arduino µC is running at a clock speed of MHz (to be precise 16MHz).

This signifies that the job of both these components to be streamed with sync.

To let these components sync we have to tweak two Object-function from each i.e python shell and Arduino µC [8]. Those two are:

1. `Serial.setTimeout`(*args) > Arduino µC
2. `Time.sleep`(*args) >> python3 code

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 11, ISSN No. 2455-2143, Pages 400-405
Published Online March 2020 in IJEAST (http://www.ijeast.com)

`Serial.setTimeout`(*args) – This function takes argument (*args) in decimal format as ms (milliseconds). By default for serial communication in Arduino, this timeout is set at 1000ms ($\therefore$ 1 sec).

`Time.sleep`(*args) – This object code is available in python shell only after importing the standard library 'time'. This library has several other useful objects, one of them is 'sleep()'. The sleep functions take an argument in decimal/floating number as secs.

Both these line-code argument values are decided by calibrating the system repeatedly until a satisfying result is achieved. There is no formula either to calculate these values.

Since there is a difference in clock speed of order 10^3 Hz, we have two options left either sync these components for seconds or milliseconds.

Our motto is to transfer data optically but also with optimum faster speed hence we decided to set time constraints in milliseconds.

$\therefore$ set python shell sleep time in ms.

Below tabulated values are calibrated for data length of 100.

Python shell	Arduino µC	Total length of data received
15ms	0ms	193
16ms	1ms	205
17ms	2ms	104
18ms	3ms	100

Table1 Recorded observations while calibrating timeout

The below Figure is the graphical representation of Table1.

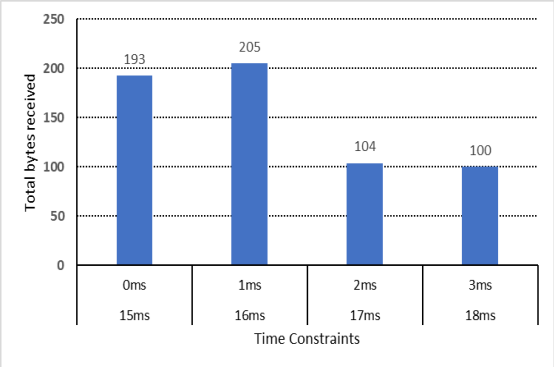


Fig6 Graphical representation for calibrating timeout

Finally, we have set the timeout value for python shell as 18ms and for Arduino µC as 3ms because at these time constraints, there was no loss of byte.

3.5 Pseudo Code of the transmitter

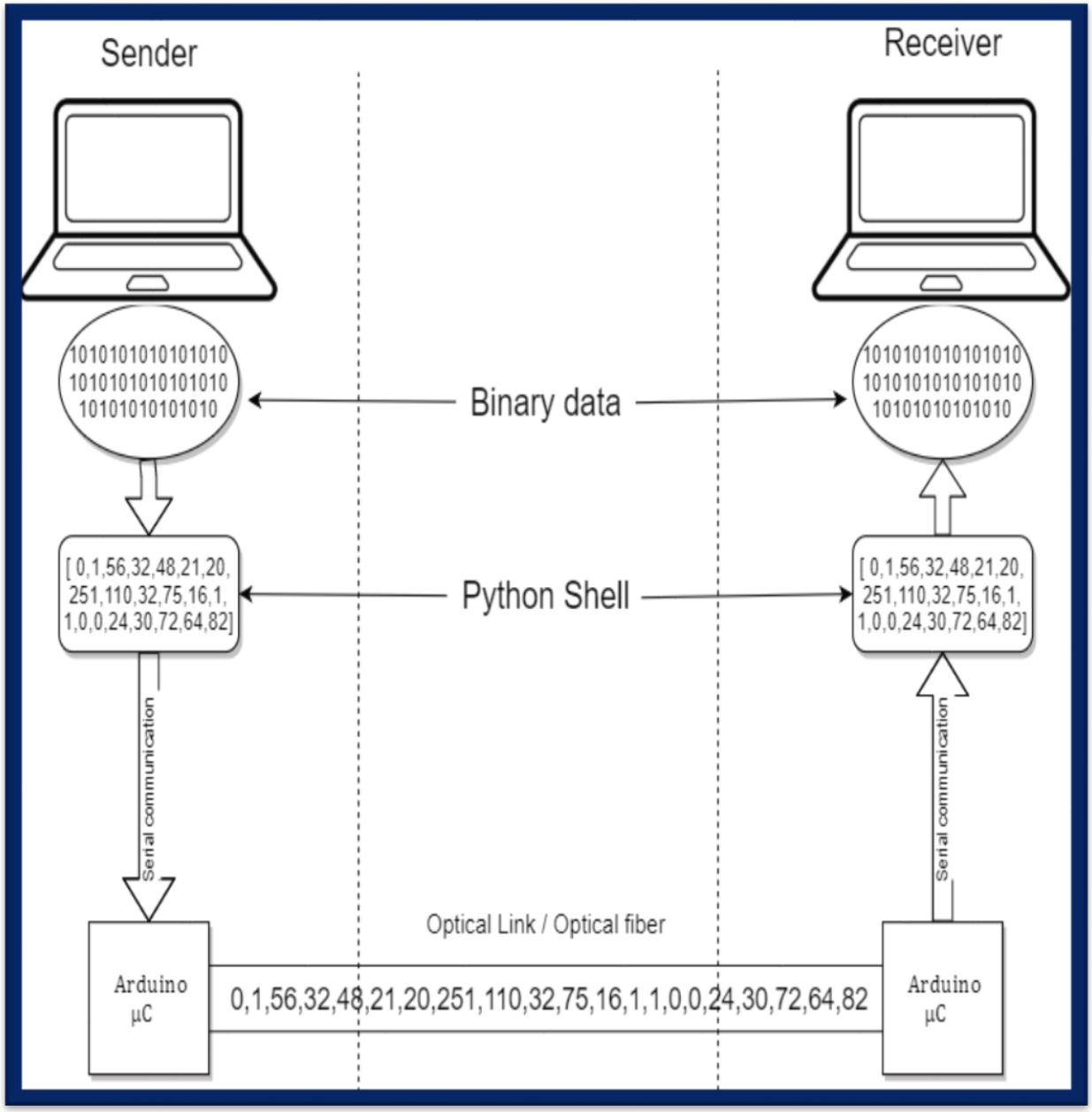
```
1. Initialize Arduino board
connection with python shell

2. Read the file as binary with
python3.

3. Load file data into a list
datatype variable.

4. Run 'for loop' up to the length
of list variable.

   1. At each incremental loop
   send serial data of byte to
   Arduino µC.
   2. Wait for 18ms
   3. Repeat
```

```
1. read serial data from the python
shell
   1. wait for 3ms

   2. store this data into
   Integer datatype.

   3. write analog at the
   desired pin with the stored
   integer value.

   4. Modulate laser intensity

2. goto line 1 and repeat
```

Fig8 Psuedo code for Arduino

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 11, ISSN No. 2455-2143, Pages 400-405
Published Online March 2020 in IJEAST (http://www.ijeast.com)

3.6 Block Diagram of transmitter and receiver



International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 11, ISSN No. 2455-2143, Pages 400-405
Published Online March 2020 in IJEAST (http://www.ijeast.com)

IV. OBSERVATIONS

Every type of file can be transmitted, whether to be a PNG, JPEG, TXT, PDF, etc. can be transmitted

Such files have a byte value between 0 to255.

Laser beam intensity can be modulated for value from 0 – 255. For a file of size 5KB, it took 1 minute and 10 seconds. No byte was lost during transmission.

V. CONCLUSION

Fig7 is a collaged image of different bytes value and Fig8 is image negative of Fig7. The negative of the image lets us see the difference in the light pattern at every byte value.

Note: Observe the black shaded region for every byte.

For a 5KB file, it took 1 minute and 10 seconds then,

5KB = 70 sec
1KB = 14 sec
∴ 1000 byte = 14 sec
∴ 1 byte = 0.014 sec
∴ 8 bits = 0.014 sec

Hence it is efficient to send 8 bits in 0.014 secs instead of single bit (1's or 0's) in 0.014 secs.

∴ The observed data transfer speed or bandwidth is approx.

71Bps (Bytes per second) or 568 bps (bits per second)

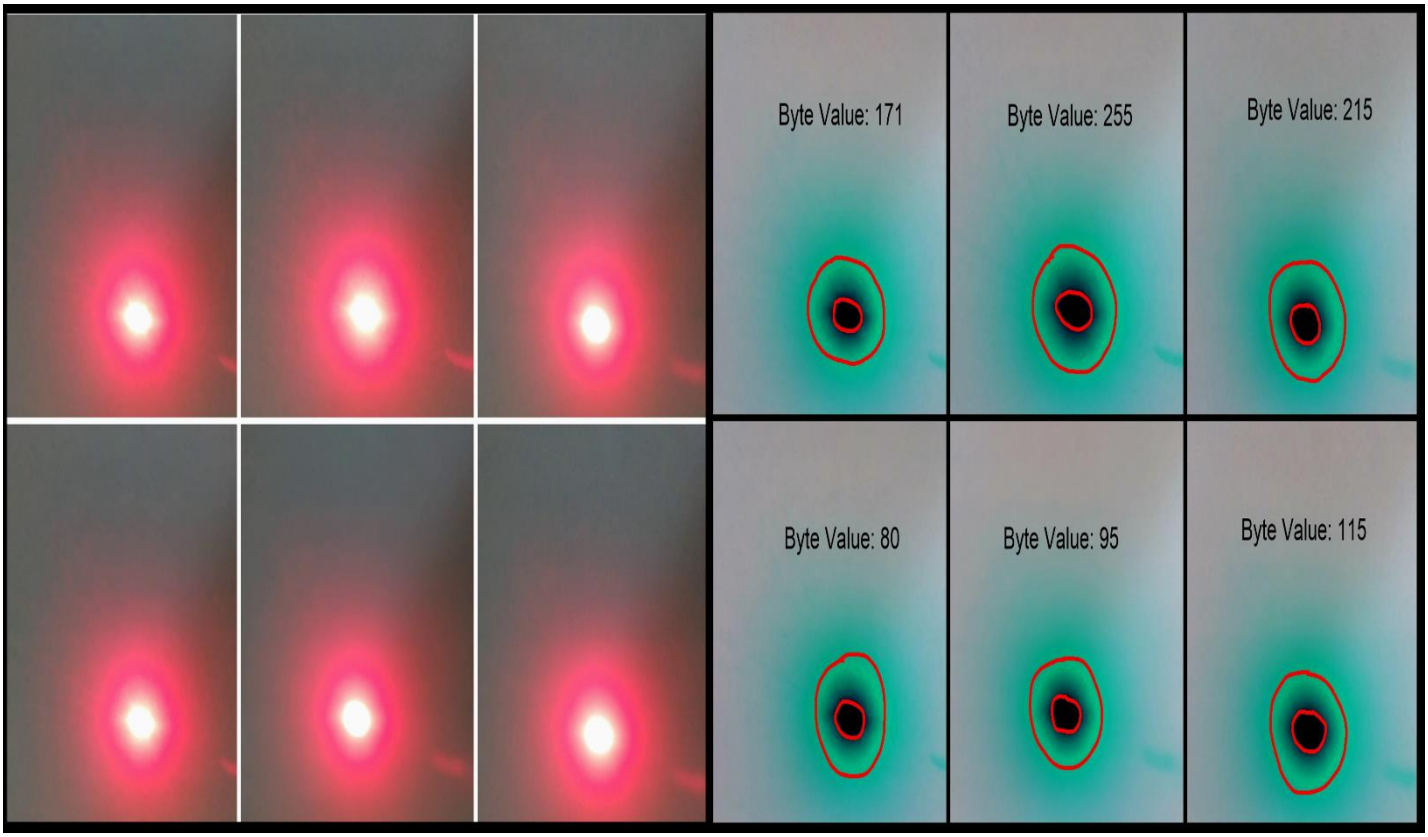


Fig7 Image of light intensity at different byte value

Fig8 Image negative of Fig7

VI. REFERENCES

[1] McRoberts Michael (2013), "Beginning Arduino". *DOI: 10.1007/978-1-4302-5017-3.*

[2] Ghelot Anita, Singh Rajesh, Gupta Raj, Singh Bhupendra, and Swain Mahendra, (2019) "Basics of Arduino". *DOI: 10.1201/9780429284564-4.*

[3] Kleveno Clasrisaa, (2020), "Arduino code explained". *DOI:10.1007/978-1-4842-5671-8_4.*

[4] Blum Jeremy, (2019), "The I2C Bus". *DOI: 10.1002/9781119405320.ch10*

[5] Blum Jeremy, (2019), "USB Serial Communication".*DOI:10.1002/9781119405320.ch.*

[6] Fry T.F, (1977), "Computer Storage". *DOI: 10.1016/B978-0-408-00239-4.50010-X.*

[7] Pilgrim Mark, (2009), "Dive into Python 3". *DOI: 10.1007/978-1-4302-2416-7.*

[8] Liechti Chris, (2017), "Welcome to pySerial's documentation". *Website: http://pyserial.readthedocs.io/en/latest/.*

[9] Klotzkin David, (2020), "Introduction: The Basics of Optical Communications". *DOI: 10.1007/978-3-030-24501-6_1* .

[10] Hui Rongqing, (2020), "Introduction to Fiber-Optic Communications". *DOI: 10.1016/B978-0-12-805345-4.00002-0.*

[11] Documentation on python 'open' function. *Website:https://docs.python.org/3/library/functions .html#open.*

[12] Rolt Stephen, (2020), "Laser and Laser applications".*DOI:10.1002/9781119302773.ch12.*

[13] Sills J.A, Wood Jr J.F, (1996), "Optimal baud-rate estimation ".
DOI: 10.1109/MILCOM.1996.571435.