

A Barcode-Based Prototype Authentication System Using Python Programming and Database Applications

Soham Chakraborty, Dedipya Datta, Aritra Mondal, Barsha Saha, Koushik Pal

Department of Electronics and Communication Engineering
Guru Nanak Institute of Technology, Kolkata
Sodepur, Kolkata - 700140, India

Abstract - Authentication is necessary for situations where limited or selected entries are accepted. In such cases, the products are authenticated based on the amounts to make a count. So, I tried to do the job using barcodes but in a different way. In the conventional level of barcode authentication, we use a barcode reader, which consists of an infrared light source and some necessary components inside a gun-like structure. Nowadays, we can see barcode scanners in robots automating the whole process. Now, what we are trying to do is, make the system work using Python codes. From the data entry process to the end of a transaction, the whole process is automatic in our system. As a barcode scanner, there is a camera, which we all have on our mobile phones, laptops and as separate webcams. So, there will be no need for the extra effort in the process of the fruitful authentication system. We have already done research work on this topic earlier. Now, this is the whole project that we were thinking about.

Key Words: Barcode, QR code, Authentication, Python, Pandas, Segno, smtplib, Security, Camera Sensor, Scanner, Reader, Database

1. INTRODUCTION

In older days, authentications were done with bare hands using pen and paper. The process was safe and secure but was a time-consuming one. Now we are blessed with technological authentication processes like RFID and Barcode authentication systems. These processes take less time and are also reliable in terms of authentication. In the conventional way of barcode authentication, we use a barcode scanner which consists of infrared light and some necessary hardware components inside a gun-shaped structure. We wanted to change the process of the scanning or authentication a bit and make the whole process automated using some accessories which are simply available in our daily life. We created 1D and 2D barcodes or QR codes using data from the physical entity. These codes can be read by the camera sensor using Python intelligence. In the below explanation, we are going through the process of the whole system.



Fig -1: Barcode Authentication

1.1 Idea of barcode

Barcode is a combination of black and white bars which are rectangular. It is the converted form of data about an object or a physical entity into a unique barcode. The Barcode was invented conceptually in the year of 1948. Two students from Drexel University named "Norman Joseph Woodland" and "Bernard Silver" created this project. It was patented in 1952. In the starting, it was a combination of black and white vertical parallel bars. These barcodes we can see in food items and grocery items. We use 2D barcodes or QR codes, a collection of vertical and horizontal bars, to get a simple and secure authentication system. Using a QR code, it is even possible to store biometric data.

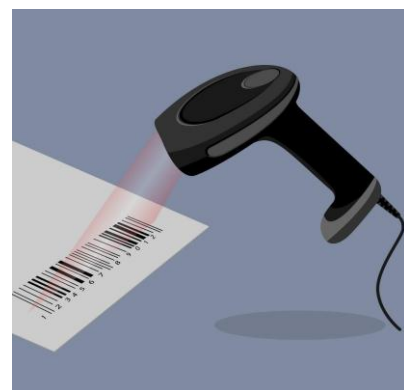


Fig -2: Barcode Scanning using Scanner

1.2 Barcode in Authentication

The information, which can describe a physical entity uniquely, can be used to make a barcode of the corresponding physical entity. A barcode is an encoded visual form of a physical entity, so using the visible form, we can authenticate that entity as per the requirements. In our daily life, many goods and products go through the markets. It would have been hectic to count all those products manually. Barcode scanning and authentication make it much easier and faster. Nowadays, in some functions, paid entry purposes, airport entries, and railway entries also use a QR code authentication system for public entries to validate their purchases. In all these cases, they use a typical barcode scanner. We have been thinking about changes to the process of "scanning" and "validating" to a Python-based system that is fast, reliable and hustle-free.

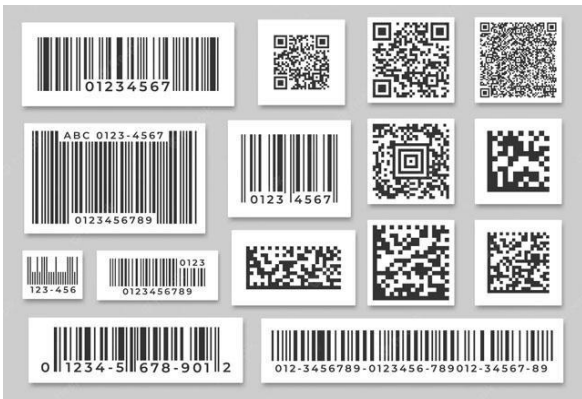


Fig -3: Barcodes

1.3 What We Have Tried

After observing different places, we have got to understand that the conventional way of barcode scanning and authentication is a relatively slow process compared to what we are trying. Retail stores, industries and warehouses use IR technology to scan a barcode printed on a product. In security entries like railway stations and airports, we see fixed IR sensors which allow us to make our barcodes scan. This process takes more time. There is a need for dedicated instruments for the purpose. So, using a simple camera which works fine and take pictures, we have tried to make a system for scanning the QR code. We all have this type of camera on our mobile phones and laptops, which we have used for this purpose. Python makes the QR code or barcode from the source data stored of a particular physical entity in the database to scan and validate it. Python has a rich library of built-in packages, which blesses us to do a lot of stuff in real. So, we have

tried to make a low-cost, fast, efficient Python-driven authentication system.



Fig -4: Python Based System using Computer Vision

2. Older Process vs New Acceptance

In the earlier days, in offices, retail stores, bookstores and even warehouses, people used to authenticate physical entities with paper and pen. Now, this scenario has changed a lot. Organisations started automation on every process and also in the authentication processes. Below we are going to discuss the growing use of automatic authentication processes.

2.1 Procedural Changes and Acceptance

In previous days, we used to authenticate things using pen and paper. But nowadays, every organisation has a database to store all the data about the products or the physical entities with their unique identities. The modern system is better and less time-consuming than the previously manual processes. Using the data from the database, we can count all the physical entities and authenticate them as per requirements. And we are working more on the modern system to make it more robust, fast and reliable.

2.1 Usage of Modern Technology

The QR or barcode technology got introduced in 1948. Two students from Drexel University named "Norman J Woodland" and "Bernard Silver" made an encoded pictorial representation of data that looked like a bullseye. After this, over the years, technology has developed a lot, and farms have started to take barcodes into everyday usage. Now it is being used in warehouses, retail stores and many more sectors where authenticity is an issue.

2.3 Technological Growth Roadmap

1948: Barcode was first introduced in 1984 by two fellows named "Norman J Woodland" and "Bernard Silver". At that time, it featured a bullseye design.

1952: The technology got patented in 1952, but it failed when technological hurdles came.

1960: The association of American Railroads sponsored the KarTrak Barcode System, the size of a refrigerator featuring 13 horizontal labels.

1972: The Committee on Uniform Grocery Products Code recommended barcode usage on most products in the "United States".

1974: On June 26, 1974, a "Wrigley's Gum" was scanned. It became the first commercially-scanned UPC barcode.

1974: The first alphanumeric barcode technology was invented. That was "Code-29".

1975: Almost all US rail cars carried KarTrak labels, although the program got abandoned three years later.

1982: The first charge-coupled device (CCD) scanner got introduced. It paved the way for the widespread adoption of barcode scanning technology.

1986: The first handheld fixed-beam laser scanner is patented.

1999: The two-dimensional QR code got unveiled in Japan.

2005: Airlines begin printing barcodes on passenger boarding passes to speed up the process.

2008: Mobile phones gain the required technology to display barcodes.

2016: The Digimarc-Barcode is a code read by machines and nearly invisible to the human eye, and is adopted by the GS1.

2.4 Usage Statistics of QR-Code

Barcode technology is much older than its subdivision, "QR-code". Now everyone has smartphones, and people use barcode or QR-Code scanning for various purposes. Only the barcode technology is 65-70 years old, and QR-Code technology has grown up in 26 years. After the covid-19 breakout, the usage of contactless transactions and all got hiked up suddenly. In the meantime, the use of QR Codes reached its top user count in between 2020 to 2022.

The survey from "QRCode Tiger" shows that in 2022 the scan count of QR codes in the USA is 2,880,960. The number of counts in India is 1,101,723. And five Asian countries come under the top 10 list of QR-Code usage count. 75% of the total users want to use QR codes also in future for their easy-to-use work.

So, looking at all those facts, we tried to make a system that can stand over society and create a valuable impact.

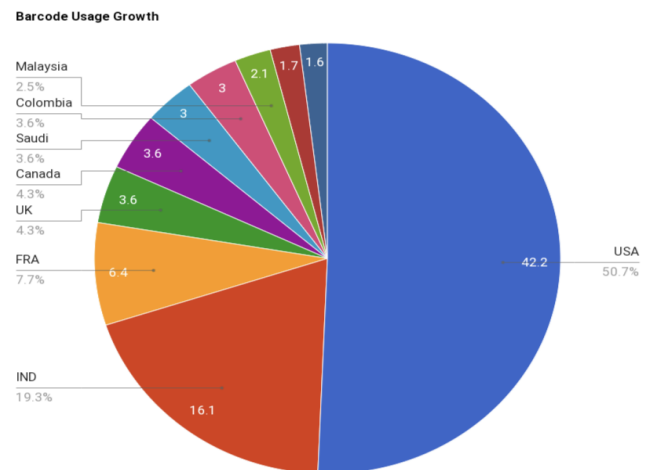


Fig -5: Barcode Usage Growth Statistics

3. What We Created

We created a complete Python Programming based system that also has some database applications in it. There we get lots of pre-installed Python packages that make every work easy. We used some of them, which are "Pandas", "Segno", "OS", "Shutil", "OpenCV", and many more. We created this project into two parts. In the first part, we made a QR code generator that takes several data of an entity from the database and makes a QR code from those data. After that, it also has an email section that sends those codes with more data to specific contacts. And in the second section, there is a camera vision-based QR code scanning section which evaluates those QR codes and checks the corresponding data with the data stored in the database earlier.

3.1 QR-Code Generation Section

We made an authentication system using the barcode or QR code technology. It is a system where people register their names with unique personal details, and we make the QR codes using those details. After creating the QR

codes, we made an email-generation section that automatically sends those QR codes to the corresponding person with some additional text.

In this QR Code Generation section, the data comes from a CSV file. People who registered their names through Google Forms have stored their data in our database. Then we prepared QR codes using those data.



Fig -6: Pandas and Numpy library logo

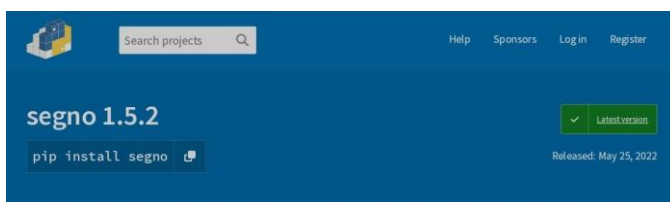


Fig -7: Segno library function

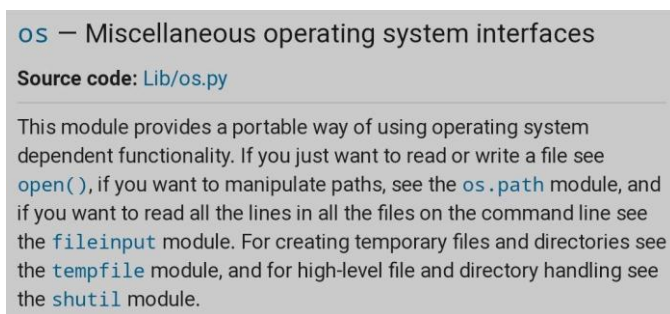


Fig -8: OS library function

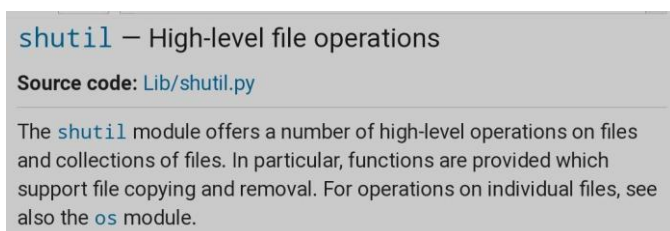


Fig -9: Shutil library function

3.2 Sending QR to People

After making unique QR codes, sending those codes to the corresponding people was the next priority. So, I imported "smtplib" from the built-in Python packages. "SMTP" stands for "Simple Mail Transfer Protocol", and I used this module to send emails to an internet machine with an SMTP or ESMTP listener.

There is a Python function for attaching various data to an email named "email.mime". Using this, I created multiple sections in the email and added some text about the validation of the QR code and the image of the QR code. I did those activities using "email.mime.multipart", "email.mime.text", and "email.mime.image". After this, I created a login instance using "smtplib.SMTP()", which takes the host and port as parameters. Then, using object-oriented programming concepts, I wrote the texts in the mail as the sender, subject, body, receiver, and all other details.

After running all these codes together, the system creates QR codes and sends them to the appropriate person via email.

3.3 Database Usage

I have used my computer hard-drive memory as the database of the project. For larger datasets, we can use databases like Oracle Database or MySQL. In our case, the dataset we used was too small, so we took a google form to collect the data from people and managed that using google sheets. Google sheet provides a good feature by which we can export the sheet as a CSV file. After getting the CSV file, I fetched all the data using a module of Python named Pandas and used it for the project.

If we need to use a database in this project, we have to create a database using MySQL or similar tools creating databases. After that, we need to insert all the data in the database in a tabular form. When the database creation is complete, we need one connector that helps to connect MySQL databases with Python. After the connection establishment, we can fetch the database using SQL commands from Python.

3.4 Scanning QR and Authentication

At the time of authentication, people need to bring those QR codes to us to authenticate themselves. For this purpose, I made another section that can take the visual of the QR code and fetch the data from it. Then the system checks the fetched data with the stored data in the "strings.text" file. After that, on the window, the result of the authentication appears. If these two data are from people and the text file matches, it tells that the person got authority. And if the data does not match or appears twice, it seems that the person got unauthorized or any outsider is trying to enter.

Let's talk about the tools and processes. Our system works based on visuals. Therefore it is necessary to get visuals of the target. For this purpose, I imported "cv2", a

Python package to operate the camera connected to the system. Then I used a command as "cv2.VideoCapture(0)" and made a variable of it. From this variable, I made the capture size of the QR code scanning window using pixel representation. I also imported "decode" from "pyzbar.pyzbar" to decode the QR code. I used "with open()" function to read and write from the text files. I created every automatic step and movement using "for" and "while" loops.

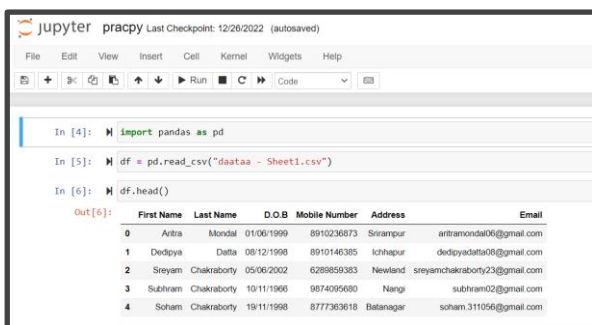
For the prevention of unauthorised entries, I created two text files. On them, the "strings.text" file was for initial data for every person, while another text file named "reentry.text" was empty. Whenever a person comes and authenticates himself, corresponding data of him moves to the "reentry.text" file. If a person comes with a fake QR code or a used QR code, the person gets marked because the authenticator does not accept QR's from outside or reentry.

On the camera visual, a text shows the result of the QR code. At the top of the QR code, it tells whether the QR code is Authorised, Reentry or Unauthorised. If the QR is valid, a green text appears on the top of the visual of the QR code as "Authorised". The result appears as blue text as "reentry" when someone enters the same code twice and as red text as "unauthorised" when an invalid QR comes.

In this way, the system authenticates physical entities fast and efficiently.

4. Process Discussion With Screen Snippets

At the start of the process, I created a Google Form to collect the data of all the people registered through our programme. I made a table with their unique details like first name, last name, date of birth, mobile number, address and email.



```

In [4]: import pandas as pd
In [5]: df = pd.read_csv("daataa - Sheet1.csv")
In [6]: df.head()
Out[6]:

```

	First Name	Last Name	D.O.B	Mobile Number	Address	Email
0	Artra	Mondal	01/06/1999	8910236873	Srisampur	artramondal06@gmail.com
1	Dedpya	Delta	08/12/1998	8910146385	Ichhapur	dedpyadelta06@gmail.com
2	Sreyam	Chakraborty	05/06/2002	6289859383	Newland	sreyamchakraborty23@gmail.com
3	Subhram	Chakraborty	10/11/1986	9874095680	Nangi	subhram02@gmail.com
4	Soham	Chakraborty	19/11/1998	8777363618	Batanagar	soham.311056@gmail.com

Fig -10: Data collected and fetched

I used "Pandas" to fetch and use the data of all the registered persons in the project. After getting all the data, I exported them as a CSV(comma-separated values) file and used it as the data center. I showed these steps in Fig-10.



```

In [41]: with open("strings.text", "w") as output:
         for i in range(len(df)):
             res = df["First Name"] + str(df["Mobile Number"]) + df["Last Name"] + df["Email"] + df["Address"]
             output.write(str(res) + '\n')
In [42]: import os
In [43]: import shutil
In [34]: dest = 'C:\QR Code Scanner\strings.text'
         sour = 'C:\Src\strings.text'
         shutil.move(sour, dest)
Out[34]: 'C:\QR Code Scanner\strings.text'
In [42]: f = open("strings.text", "r")
         file_con = f.read()
         print(file_con)

```

Fig -11: Creating QR codes and Saving as .jpg

In the fig-11, I showed how I created a formula to generate unique QR codes and save them as '.jpg' images. I fetched all the personal data of people using a for-loop to make a combination of those data, which is different for everyone as the mobile number and email ids are unique for everyone. After that, I saved all those strings into a file named 'strings.text' and used 'shutil' to make a copy of that file in the QR scanner section. This text file in the scanner section will check with the scanned data from the QR codes at the time of authentication.

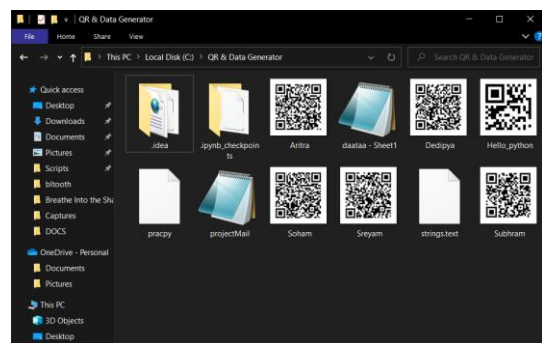


Fig -12: Made QR using Segno and Saved as .jpg

After making strings from all the personal data, I used 'Segno' to create the QR codes and saved them into the same file. I showed this in Fig-12.

```

jupyter pracky Last Checkpoint: 12/06/2022 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3.8
In [9]: In [9]: M Import smtplib
In [10]: M from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from pathlib import Path
from email.mime.image import MIMEImage

with smtplib.SMTP(host="smtp.gmail.com", port=587) as smtp:
    smtp.starttls()
    smtp.login("sujata94@gmail.com", "sqoqilbksqscvf")
    for i in range(0, len(emails)):
        message = MIMEMultipart()
        message["from"] = "Projectforun Authority"
        message["to"] = email[i][1]
        message["subject"] = "Do not show it to someone"
        message.attach(MIMEText("Dear customer, \n\n This mail contains an authentication QR code for the security purpose o
name = email[i][0]
        message.attach(MIMEImage(Path(f'{name}.png').read_bytes()))

    smtp.send_message(message)
    print("Sent...")
Sent...
Sent...
Sent...
Sent...
    
```

Fig -13: Designed the email and sent with QR codes

Then I used 'smtplib' to send those QR codes to the people accordingly. For that purpose, I used another email id as the host and created a temporary password for the mail id login. Then I designed the mail subject, body and all the necessary things using MIMEMultipart. Using for-loop, I fetched all the email ids and sent the email to them with the QR codes.

I used Pycharm IDE to write and run the QR scanner section code and verify the QR codes. Using openCV, time, and many other packages, I created an efficient code that checks whether the QR code is valid or invalid. Pre-created 'strings' text file contains all the QR code data. When someone shows their QR code in front of the camera vision, the algorithm checks it with the 'strings' file and gives the authentication result in a GUI window named "Authentication Result".

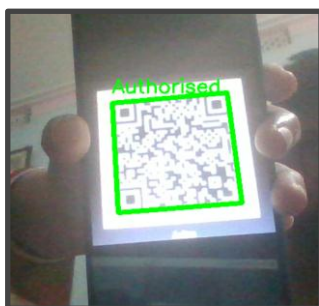


Fig -14: Authenticated Message in GUI Window

When someone shows a QR code to the camera vision and the data matches the 'strings' file data, the GUI shows an "Authenticated" message with a green colour in the window. It also sends the data from the 'strings' file to another file named 're-entry', and the space where it was in 'strings' remains null. Fig-14.



Fig -15: Re-Entering Message in GUI Window

The algorithm checks two text files consecutively. If the QR data matches the 're-entry' text file data, it implies someone is entering their QR code again. In this situation, the GUI window will show a blue-coloured message as "Re-Entering". Fig-15.

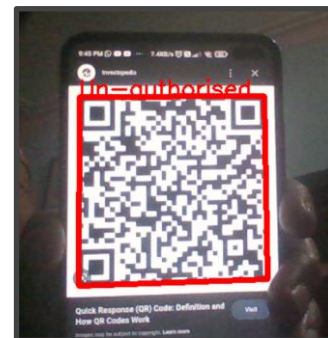


Fig -16: "Un-authorized" Message in GUI Window

The GUI window will show a red-coloured "Un-Authorised" message when the QR data is neither in the 'strings' nor the 're-entry' text file.

4. Where can be used

4.1 Selective People Entries: We can use this system in urban clubs and parties where restricted or invited people are allowed. At the time of data enlistment, we can store the data of each person in the database or a sheet, like Google form or MS Excel. After that, we can send the invitations via their unique email id and put the unique QR code in the email. For authentication purposes, they will show those QR codes to us on the event date. They will place their QR codes under our specified place at the entrance, and our system will ensure whether the person is allowed to enter. If the system shows a green signal, the person is allowed to enter. Otherwise, we will ask the person to authenticate him or herself with other entities.

4.2 Railways and Bus Entries: We can use this system in railway stations and bus entries to buy travel tickets. Whenever a person buys a token, a unique QR code is generated and printed on it for them so they can use this code to authenticate themselves in the departure section. By using such systems in public places, we can get more information about the everyday activities in stations or bus-travels. Information like this can be crucial for the government to track people.

4.3 Other Usage: We can use this system in places where authentication of any physical entity is needed. In public sectors like government service sectors, offices, schools, taxis, heritage locations and many others, we can use this system as the authenticator. We can implement face recognition in this system to make it nimbler and more reliable.

5. Advantages of The System

5.1 The system will be effortless, robust and reliable. Developers and the system itself will manage all the background processes of it.

5.2 QR code authentication system is a fast one we all know. After implementing the camera sensor as the computer vision and Python-based system, it gets quicker and effortless.

5.3 In the system, the database stores all the data. The system works based on those data, so there is no risk of losing any data.

5.4 All the processes of the system as data collection, processing, QR-code generation, sending QR codes and authentication, are automated. So, making the system automated makes it more robust and effortless.

6. Drawbacks of The System

6.1 I have used Python as the process control language. Development time is lesser in the case of Python, but it is 3 to 5 times slower than Java in runtime. Python's built-in high-level data types and some other things are responsible for it.

6.2 I haven't developed any publicly usable version of the system. So, it is only a developer prototype version till now. Everyone cannot understand it and use it seamlessly.

7. Future Improvement Scopes

7.1 We are working on it to improve the runtime of the system code. Though the system is fast enough, we are trying to make it more efficient.

7.2 We are working on it to improve the runtime of the system code. Though the system is fast enough, we are trying to make it more efficient.

7.3 We are looking forward to making a publicly usable version of the system so that people can use our system by themselves.

8. Conclusion

The authentication process is a crucial step in every sector nowadays. We have made a system that can serve the purpose of authentication effortlessly and in an automated way. Starting from data collection to the end of the authentication process, it's a seamless process. Undoubtedly our system is better than the conventional authentication process, but we have to give it a user-friendly user interface that everyone can use. Soon, our system can be grown to a new height and serve society with good value.

REFERENCES

- [1] Barcode Recognition System, N. M. Z. Hashim, N. A. Ibrahim, N. M. Saad, F. Sakaguchi, Z. Zakaria
- [2] Barcode Based Student Recognition System, Samira Nigrel, Akshay Kumar Prajapati, Kunal Lad, Sachin Jhaveri,
- [3] IoT(Internet of things) and Its Application_0 LEVEL Book M4:R5_BILINGUAL BOOK(ENGLISH-HINDI) by T BALAJI PUBLICATION
- [4] Data Structures and Algorithms in Python, 1st Edition - 5 July 2013 Michael T. Goodrich, Roberto Tmassia, Michael H. Goldwasser