3rd International Conference on
Robotics Automation and Non-Destructive Evaluation,
Chennai, India, 23 April 2022.
https://doi.org/10.13180/RANE.2022.23.04.06

DYNAMIC DASHBOARD AND MAIL UPDATE FOR ROBOTIC SYSTEM IN INDUSTRY 4.0

R. PRAVEEN, S. GOWTHAM, M. PARTHIBAN and P. SAI CHARAN
Department of Mechatronics Engineering, School of Mechanical Sciences, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India
E-mail: praveen2000rajendran@gmail.com, gowtham.sridher5@gmail.com, parthibanakash140899@gmail.com, saicharancherryroyal@gmail.com

N. SEENU and RM. KUPPAN CHETTY
Centre for Automation and Robotics, School of Mechanical Sciences, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India
E-mail: nseenu@hindustanuniv.ac.in, kuppanc@hindustanuniv.ac.in

A majority of industry based robotic systems are interconnected using IoT (Internet of Things) which has majorly evolved Industry 4.0 applications during the recent decade. The proposed method aims to implement IoT communication to log the details of packages sorted by robotic manipulators. An interactive dashboard is developed using CSS to log the package data and mail alerts are sent to notify the person in-charge. By implementing python request, data is sent to Google Apps Script then concurrently updated in Google Sheets once mail alerts are sent. The logged data is published in a webpage accessed by a HTML based dashboard using AJAX library. Upon completion of sorting, various attributes such as name, quantity, priority and corresponding timestamps of the sorted packages are logged and presented to the end user. In this paper, the functionality of the proposed system is demonstrated in the widely used color-based sorting to perform pick and place tasks. However, the proposed IoT-based approach can be extended to keep track of numerous tasks carried out by several industrial robots within a facility.

Keywords: Internet of Things, Automatic mail, Dashboard, Robotic manipulator, Robotic system, Industry 4.0

1. Introduction

Presently, most production industries are equipped with robots and other machine-based automation systems. However, in various sectors, communication between several robots/machineries are not established within a centralized server. Logging the activities carried out by several automated robots are usually carried out manually. Such conventional approaches are often time-consuming and unreliable due to human error. Therefore, automatic information gathering systems are highly demanded to mitigate this issue [1]. Another notable disadvantage of autonomous robotic systems is the safety concern of people working within the robot's workspace. Commonly, information acquisition and data logging can prove tedious when data is displayed via LCDs of various machineries, as it cannot be accessed outside the industry [2]. Therefore, this research is majorly targeted towards Industry 4.0 applications to make the data readily available for user access, anywhere.

The proposed approach is tested by logging the details of packages sorted by a robot manipulator. The package information is automatically logged into the cloud-based google sheet though app scripting [3]. Once the object is placed in the desired location, request library is utilized by the python script to send the corresponding package data to update the spreadsheet. The logged information displayed in the dashboard is sent via mail to the user using the mailapp library in app script [4]. The sheet is published in a webpage and accessed by the AJAX library

in JavaScript [5]. A dynamic webpage is developed to display the information stored in cloud based google sheet dynamic webpage dashboard designed using CSS. table represents the information is sortable both alphabetical and numerical order [6].

In our approach, communication is established between an articulated robot manipulator and an end user using IoT. The manipulator uses visual input to sort packages based on its color signature. IoT communication is implemented to log the details and timestamps of the sorted packages in a spreadsheet and a webpage embedded interactive dashboard. The users are notified of the product's status and summary via auto-generated emails after the sorting operation. The experimental results prove the efficacy of IoT communication and its simplicity in application for various industrial systems. The dashboard and mail updates used alongside ROS (Robot Operating System) based robotic systems can revolutionize technology in industry 4.0.

2. Methodology

The paper primarily addresses the crucial applications of Industry 4.0 to provide adequate data visualization in intelligent warehouses which is incorporated with robotic systems. It is essential to develop an interactive dashboard to log the details of packages sorted by the manipulator to track numerous operations performed by several industrial robots. In order to communicate information to the user via IoT, the data is in a cloud based platform such as Google sheet. Figure 1 represents the sequence of processes involved in transferring data until it is displayed to the user. Google sheet is embed with google app script which in this research is used to update the values to sheet using HTTP requests. Python script does the manipulation to update the values to google app script using the request library. The package parameters such as sorted status, product name, product ID, timestamps and date. Such information is accessed by app script and consequently updated to the sheet. Dynamic dashboard accesses the information in sheet using AJAX in JavaScript.

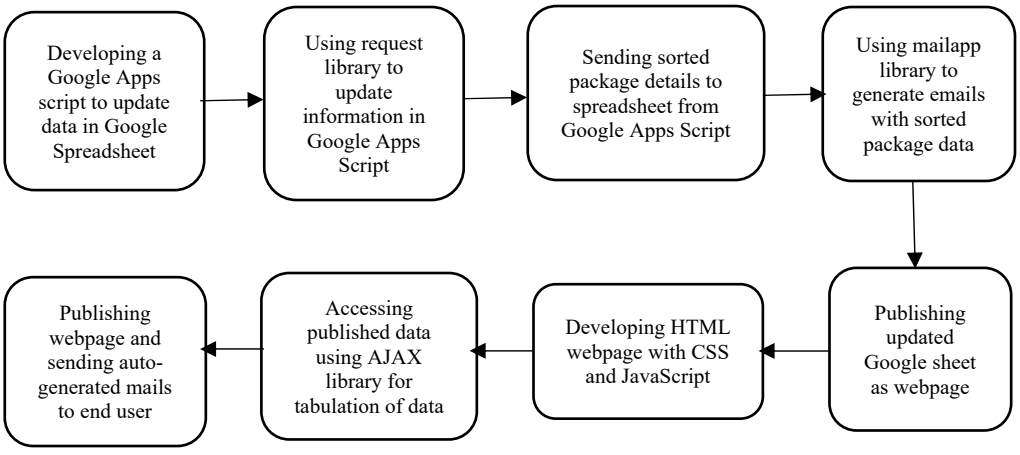

Figure 1. Process flowchart.

2.1. *Implementation of IoT through App Scripting*

The Google Apps Script platform acts as an intermediate between the webpage and Python to establish IoT communication. Without explicitly defining the contents and formatting changes into the spreadsheet, specific cells are targeted by creating custom functions and macros using the script editor. While defining a spread sheet, the respective column headers to log data of

sorted packages are labelled as product name, product ID, quantity, sorted status, and timestamp. The python request library sends an HTTP request to the web application by passing package details in a conditional structure within the doGet function [7]. The GET method indicates data retrieval from a specified resource, in this case, the Google spreadsheet. Dependencies such as 'requests' and 'datetime' are imported in the script to send HTTP requests and log the current date and time under the timestamp column, respectively. The spreadsheet is deployed as a web application, and the URL is passed into the 'get' function to log the detected parameters of the package into the spreadsheet.

2.2. *Developing an Auto-generated E-mail Alert System*

In the previous section, the developed python script serves as a software interface between Python and Google Apps Script. The request library retrieves the package parameters from the apps script and sends auto-generated e-mails to the user's Google mail ID. The mail contains information on parameters such as product ID, quantity, sorted status, and sorted time. The mail app class sends an auto-generated mail containing the package's parameters through an HTTP request [8]. After sending the mail, sorted data is automatically updated in the spreadsheet. The user's Gmail ID is stored in the 'to' variable indicating the recipient similarly; the subject is stored in the message variable. The program implementation in Google Apps Script performs the spreadsheet's data logging activity using the package parameters retrieved from the conditional structure of the python script.

2.3. *Creating a Webpage Embedded Interactive Dashboard*

The AJAX (Asynchronous JavaScript and XML) library is used in JavaScript to retrieve and update values from the spreadsheet without refreshing the web page, which is a technique to make fast dynamic web pages [9]. CSS is used to describe the pad- ding around images and other objects and the design and thickness of the table's border. CSS allows web designers more precise control over the appearance of pages than HTML does. Therefore, the former is implemented to develop an interactive dashboard using gradient colors. The AJAX library is a web development method utilized to send requests to the server asynchronously. The user sends a JavaScript call request to the XML HTTP request block containing the callback function in the browser block. The HTTP request is then sent to the webserver, and the server does database interacting using a scripting language, PHP, to retrieve the data back to the callback function [10]. This way, the AJAX technique automatically sends and retrieves information in the browser background without refreshing the web page for every 5 seconds. The HTML table can sort the entries alphabetically, numerically, and based on sorted time. In the next section, experimental results obtained after applying the above methodologies are discussed.

3. Experimental Results

The experimental setup consists of a USB camera and Kinova JACO2 robot manipulator to demonstrate IoT communication after the pick and place operation. USB camera detects the various packages in its field of view and updates the package information to sheet accordingly. IoT communication is established in ROS (Robot Operating System) and Python so updating information to sheet is carried out using python request and manipulation is accomplished using ROS MoveIt and Python API for MoveIt.

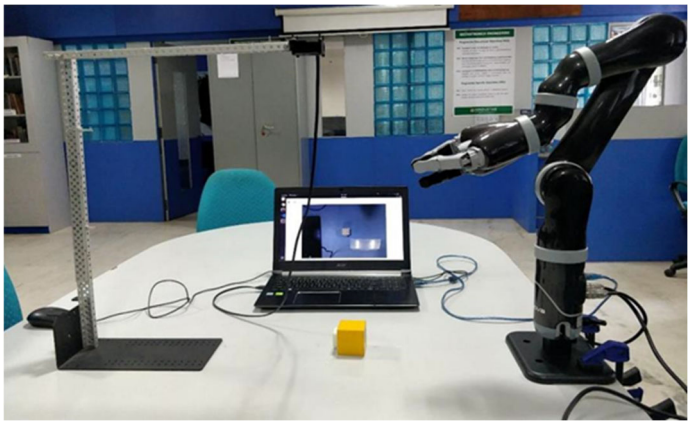

Figure 2. Experimental setup of real-time robot

3.1. *Updating the Dashboard through AJAX Requests*

The Google Apps Script web application is utilized to log the sorting information into a Google spreadsheet such that it indicates the product summary and sorting timestamps of each package. After dropping a package to the goal destination, the request library from python sends the google apps script's required parameters and updating the Google spreadsheet values. Figure 3 represents the updated values sent by HTTP request in the python script. An additional python script is developed to load new data entries in the web page during sorting operation after a given timeout. Instead of manually reloading the webpage URL to view the updated data, the python script uses HTTP requests to extract data from the spreadsheet automatically in the background.

	A	B	C	D	E	F
1	Timestamp	productid	Product	quantity	sorted	sortedTime
2	Wed Mar 24 2021, 05:55:25	CL01	Controller	1	YES	2021-03-24 15:25:24
3	Wed Mar 24 2021, 05:55:30	PS01	PCB	1	YES	2021-03-24 15:25:29
4	Wed Mar 24 2021, 05:55:37	CL01	Controller	1	YES	2021-03-24 15:25:35
5	Wed Mar 24 2021, 05:55:42	PS01	PCB	1	YES	2021-03-24 15:25:41
6	Wed Mar 24 2021, 05:55:48	CL01	Controller	1	YES	2021-03-24 15:25:47
7	Wed Mar 24 2021, 05:55:53	CL01	Controller	1	YES	2021-03-24 15:25:52
8	Wed Mar 24 2021, 05:55:58	LM00	Motor	1	YES	2021-03-24 15:25:58
9	Wed Mar 24 2021, 05:56:04	CL01	Controller	1	YES	2021-03-24 15:26:03
10	Wed Mar 24 2021, 05:56:09	LM00	Motor	1	YES	2021-03-24 15:26:08
11	Wed Mar 24 2021, 05:56:15	CL01	Controller	1	YES	2021-03-24 15:26:14
12	Wed Mar 24 2021, 05:56:20	PS01	PCB	1	YES	2021-03-24 15:26:19
13	Wed Mar 24 2021, 05:56:26	PS01	PCB	1	YES	2021-03-24 15:26:25

Figure 3. Updating spreadsheet with google apps script

The data retrieval node is unsubscribed from the topic at a given instant using the 'unregister' function. It ensures that once a color signature is retrieved and an email is sent, the node unsubscribes from the topic until a new color signature is detected to avoid notification spam in the inbox. Using JavaScript, the HTML table is made sortable in alphabetical and numerical order, and CSS provides an interactive background, as illustrated in Figure 4. The following google drive link consists of videos of the above experimental results https://drive.google.com/drive/folders/1xypKQLm6wyqZq9y4wf5Mha4491wSLvBp?usp=sharing

Figure 4. Updating the dynamic dashboard using AJAX requests

3.2. *Auto-generating E-mails through App Scripting*

The Google spreadsheet is updated using the "doGet" function in the apps script editor to access the python script contents using a GET request. Subsequently, an additional conditional code snippet is developed inside the 'doGet' function to check for data log and retrieve data entries from the spreadsheet. The package's colour information is retrieved from the '/pixel data' topic using the subscriber node's callback function. Upon completion of the sorting operation, the subscriber node checks for the gripper status, i.e., open/close, to initiate the e-mail generation process. If the gripper status condition reads open, the node is initiated, and the callback function is called by passing the color signature as an argument.

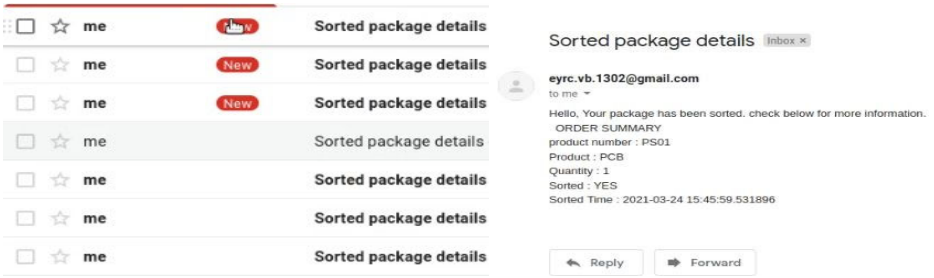

Figure 5. Auto-generated e-mail as user notification

After retrieving the package's respective parameters, the 'send e-mail' function is called within the class method's callback function. The parameters of the sorted package are sent to the web application URL to log the spreadsheet automatically. For instance, if the published color signature is '12', the PCB parameters are logged in the respective spreadsheet columns to send an auto-generated e-mail, as shown in Figure 5. In the spreadsheet and user e-mail, the packages are named "PCB," "Motor," and "Controller," representing that sorting operation was carried out on dual-colored boxes. After sorting each package, parameters such as product number, product ID, Product name, quantity, and timestamps are displayed under the order summary in the body of the e-mail. The data is updated either to the customer or the managing person based on the e-mail ID provided in the apps script.

4. Discussion

IoT based dashboard and mail update has been implemented using google app script by updating using python script with request library. The updated data is gathered by google app script to

log the data in its respective columns. Once the robotic manipulator opens its gripper and drops the package. The python script sends the values with the type of object is sorted. The type of package sorted is recognized vision sensor and then stored in list to be updated by google sheet. By using mailapp library in google app script the values are sent to the customer. Updated google sheet is published as webpage which then accessed by HTML dynamic dashboard using AJAX library to refresh the information from the sheet for every 5 seconds. Dynamic dashboard consists of a sortable table used to display the data updated in sheet. Dynamic dashboard and mail update is incorporated with robotic system in industry 4.0 for acquiring better view of information and can be accessed worldwide with use of internet.

5. Conclusion and Future work

Cloud based google sheet is successfully used to update the values in dynamic dashboard and send mail updates to the customer. This type of dashboard and mail update can be used with any kind of robotic system in Industry 4.0. Information is updated to google sheet by using request library in python. Google sheet is updated with data shared to google app script in respective columns. Information from the google spreadsheet has been successfully transferred to dashboard and mail alerts are received by the user. Dashboard developed with HTML show the data which is stored in the spreadsheet using AJAX in JavaScript. Background of the dashboard is made with dynamic gradient background to make the dashboard more attractive to customers. This IoT system is tested with Kinova Jaco2 robotic arm which gives successful results once the package is sorted by robotic arm. IoT based dashboard and mail update is majorly suited in the field of Industry 4.0 and can function with most of the robotic systems with python support.

The future work to be incorporated to yield the performance and applicability of IoT based dashboard and mail update falls into the following perspectives:

- To make use of this type of dashboard and mail update in warehouses.
- To implementing IoT communication in all robotic systems within a facility.

References

1. Azizi A, *Applications of control engineering in industry 4.0: utilizing internet of things to design an agent based control architecture for smart material handling system.* International Robotics & Automation Journal, 4(4), 253–257 (2018).
2. Gao Z, Wanyama T, Singh I, Gadhrri A and Schmidt R, *From industry 4.0 to robotics 4.0 - A conceptual framework for collaborative and intelligent robotic systems.* Procedia Manufacturing, 591–599 (2020).
3. Lawton J, *Software, Robots, IoT and Real Steps to Industry 4.0. https://www.automateshow.com/filesDownload.cfm?dl=Lawton-SoftwareRobotsIoTandRealStepstoIndustry4_0.pdf*
4. Ruano P, Delgado L.L, Picco S, Villegas L, Tonelli F, Merlo M and Masuelli M, *Interlinking Industry 4.0 and Academia through Robotics and Automation: An Indian Perspective* (2016).
5. Ruano P, Delgado L.L, Picco S, Villegas L, Tonelli F, Merlo M, Rigau J, Diaz D and Masuelli M, *We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists TOP 1 %.* Intech, tourism, 13. https://www.intechopen.com/books/advanced-biometric-technologies/liveness-detection-in-biometrics (2016).
6. Dhanabalan T and Sathish A, *Transforming Indian industries through artificial intelligence and robotics in industry 4.0.* International Journal of Mechanical Engineering and Technology, 9(10), 835–845 (2018).

7. Gounder M. S, Iyer V. V and Mazyad A. Al, *A survey on business intelligence tools for university dashboard development.* 3rd MEC International Conference on Big Data and Smart City, ICBDSC 2016, 85–91 (2016).
8. Kao K. C, Chieng W. H and Jeng S. L, *Design and development of an IoT-based web application for an intelligent remote SCADA system*. IOP Conference Series: Materi- als Science and Engineering, 323(1) (2018).
9. Karnouskos S, Gaertner N, Verzano N, Beck F, Becker A, Bianchino S, Kuntze D, Perez M, Roy R, Saelens S and Schmut M, *Experiences in integrating Internet of Things and cloud services with the robot operating system.* Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017, 1084–1089 (2017).
10. Al-Taee M A, Al-Nuaimy W, Muhsin Z. J and Al-Ataby A, *Robot Assistant in Management of Diabetes in Children Based on the Internet of Things*. IEEE Internet of Things Journal, 4(2), 437–445 (2017).