AJSE

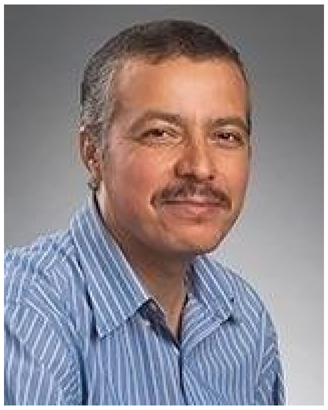
American Journal of Science & Engineering

Volume 4 Issue 1

June 2023

American Journal of Science & Engineering (AJSE)
Society for Makers, Artists, Researchers and Technologists (SMART)
6408 Elizabeth Ave SE, Auburn 98092, Washington, USA
ISSN: 2687-9530 (Print) and 2687-9581 (Online)

Editor-in-Chief



Dr. Izzat Alsmadi
Texas A&M, San Antonio, USA

Research Interest: Cyber Intelligence, Cyber Security, Software Engineering, Social Networks

Bio: Izzat Alsmadi is an Assistant Professor in the department of computing and cyber security at the Texas A&M, San Antonio. He has his master and PhD in Software Engineering from North Dakota State University in 2006 and 2008. He has more than 100 conference and journal publications. His research interests include: Cyber intelligence, Cyber security, Software security, software engineering, software testing, social networks and software defined networking. He is lead author, editor in several books including: Springer The NICE Cyber Security Framework Cyber Security Intelligence and Analytics, 2019, Practical Information Security: A Competency-Based Education Course, 2018, Information Fusion for Cyber-Security Analytics (Studies in Computational Intelligence), 2016. The author is also a member of The National Initiative for Cybersecurity Education (NICE) group, which meets frequently to discuss enhancements on cyber security education at the national level.

Page No.	CONTENT
1-7	**Breaking Down Break-It-Fix-It: An Automated Software Repair Replication** *Software quality is strongly correlated with the quantity and severity of bugs. While there are a variety of tools, techniques, and practices to aid production of robust and resilient code, low quality software is draining trillions of dollars from organizations annually. Meanwhile, debugging and fixing coding errors consumes upwards of half of developer labor. To say this situation is untenable is an understatement. Fortunately, automated software repair offers a possible solution. The literature around automated code fixing has been expanding with a variety of implementations ranging from genetic programming, code translation, and various machine learning algorithms. All report positive results, however there has not yet been a dedicated effort to measure to what extent the various implementations are generalizable. Accordingly, we sought to replicate a prominent study in the field in two parts. The first part consisted of replicating the training of the machine learning model using the source study materials. We found training to be impossible at first due to package dependencies and missing package files. However, we were able to replicate the self-repair evaluation. The results were identical to the source study. Later, using a Docker compose file obtained from the original authors, we were able to replicate BIFI model training and again match outcomes. Overall, based on the replication outcomes, we offer future leaning recommendations and ideas for future work.* **DOI:** Jason M. Pittman (Booz Allen Hamilton, McLean, USA & University of Maryland Global Campus, USA), Kira Cincotta (Booz Allen Hamilton, McLean, USA), and Rebecca Saul (Booz Allen Hamilton, McLean, USA)
8-12	**Microcontroller based Smart Coil Winder System** *The aim of this paper is to develop the control circuit for smart automatic coil winder, which functions with respect to the PIC Microcontroller program. The conventional coil winding machine winds copper wire on a former and the former is attached to the iron rod, which is to be actuated manually. Moreover, the manual coil winding machine does not have any control circuit for the smart functionality. But in this design, the controller has more memory to operate and save the data and also to improve the winder program. Additionally, it also reduces the time of operation and the requirement of Manpower. This coil winder control circuit uses PIC Microcontroller (16F887), which runs the winder system according to main program and also the motor step sequencing program.* **DOI:** Dr. A. Rajamani, Ms. N. Saranya (PSG College of Technology, India)
13-17	**Programming Language Conversion using NLP** *Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do. NLP models language computationally and deals with linguistic features of computation. Once a computer learns to do mathematical calculations it can perform many complex and big calculations much faster than humans. Similarly, once computer starts to understand the human languages it can process all aspects of that language much faster than humans also opening a large number of possibility. So it cuts down on employment as one computer is capable of giving an output 10 times faster than a human can. So it benefits the employer not only financially but also by giving extremely accurate and faster outcome. Here we lay out an overall architecture to explain the overall processing. So now we take a look at the two general classes of systems they are special-purpose system and general-purpose system, explaining how they differ and their relative advantages and disadvantages. After that we point at the few remaining problems that require additional research. Finally, we conclude by discussing when natural language processing technology can be practically used at various levels .We also discuss about when it will become commercially practical, and what will be the cost to practically use this technology.* *The techniques specifically developed for analysing and understanding the inner-workings and representations acquired by neural models of language is EMNLP 2018 BlackboxNLP. The approach includes: investigating the impact on the performance of neural network on systematic manipulation of input and also testing whether the interpretable knowledge can be decoded from intermediate representations to propose modifications to make the knowledge state or generated output more and also to examine the performance of networks on simplified or formal languages.* *In the following report we aim to convert a program of a given language to an equivalent program of another language. For that we have taken help of NLP that is Natural Language Processing. By using Natural Language Tool Kit, we have successfully identified the variables, datatypes, operators, keywords, indentations. We have also discussed various aspects and domains of NLP and some real-world applications of it.* **DOI:** ABHIJIT BANERJEE, MADHUBAN MUKHERJEE, APARAJITA BANERJEE, MD. AALISHAN RAZA, SUCHETA BHOWMICK, SAKSHI BHAGAT, SUDIPTA BASU PAL (University of Engineering and Management, India)

18-24

Smart Waste Bin Monitoring in Municipal Based on IOT for Clean City

Rapid increasing urbanization and increasing population all over the world, there is a dynamic increase in the amount of waste disposal has become a matter of concern, and diseases like malaria, dengue, and cholera are caused due to overflow of garbage which contains rotten things which form foul smell and burning things that cause air pollution to the environment. As a result of this we human beings are sufferers, So, we need to maintain this worst scenario and we need to keep a track of the garbage bin so that it will be cleaned in the proper interval of time. So, we are implementing this project which will identify the waste by checking its humidity and temperature and will also check the level of the garbage bin so that it doesn't overflow and pollute the environment. In this project the garbage level in each bin is monitored using ultrasonic sensors present in every bin, rotten and burning elements or any such abnormal situation that arises will be identified by the Gas sensor and DHT11 sensor. The Gas, ultrasonic sensor, and Humidity sensor will read the data and will send it to the Cloud server and then the Municipal Control room will be able to monitor the information from the Cloud server through a GUI interface. When more than 70 percent of the garbage bin is filled, or any situation mentioned above arises the buzzer will give the indication. The system is driven by a microcontroller-ESP-32 which is working as the brain of the operation, and it is programmed using Embedded C. All the devices and Cloud Server plays a key role to implement the project.

DOI:

Sreya Poddar, Tisa Dutta, Sunando Chowdhury, Soumyadeep Mukherjee, Samprikta Mukherjee
(Institute of Engineering and Management, India)

Breaking Down Break-It-Fix-It: An Automated Software Repair Replication

Jason M. Pittman*†, Kira Cincotta*, and Rebecca Saul*
*Booz Allen Hamilton, McLean, VA 30332 USA
†University of Maryland Global Campus, Adelphi, USA

***Abstract*—Software quality is strongly correlated with the quantity and severity of bugs. While there are a variety of tools, techniques, and practices to aid production of robust and resilient code, low quality software is draining trillions of dollars from organizations annually. Meanwhile, debugging and fixing coding errors consumes upwards of half of developer labor. To say this situation is untenable is an understatement. Fortunately, automated software repair offers a possible solution. The literature around automated code fixing has been expanding with a variety of implementations ranging from genetic programming, code translation, and various machine learning algorithms. All report positive results, however there has not yet been a dedicated effort to measure to what extent the various implementations are generalizable. Accordingly, we sought to replicate a prominent study in the field in two parts. The first part consisted of replicating the training of the machine learning model using the source study materials. We found training to be impossible at first due to package dependencies and missing package files. However, we were able to replicate the self-repair evaluation. The results were identical to the source study. Later, using a Docker compose file obtained from the original authors, we were able to replicate BIFI model training and again match outcomes. Overall, based on the replication outcomes, we offer future leaning recommendations and ideas for future work.**

***Index Terms*—software and engineering, automated software repair, deep learning, replication.**

I. Introduction

SOFTWARE Software quality is strongly correlated with the quantity and severity of bugs. Further, software quality is commonly understood to be a measure of design practices, errors per line of code, and testing acumen Misra & Bhavsar (2003); Abuasad & Alsmadi (2012); Alves et al. (2016). As it stands, poor quality software led to losses of 2.41 trillion USD in 2022 McGuire (2022). In light of these facts, it is natural to wonder why industry simply does not produce more high quality software. In fact, there are a variety of tools, techniques, and practices to aid developers and engineers in product robust and resilient code. However, automation is limited in two critical areas: debugging and generating code repairs (i.e., patches) Tufano et al. (2019); Xia et al. (2022). Bugs and vulnerabilities alike can take days to isolate, especially in large projects Alhefdhi et al. (2020). Manual debugging and patching are notoriously tedious job tasks. Often, software developers spend upwards of 50% of their time debugging Britton et al. (2013). Moreover, such manual interventions have a nonzero probability of regressing source code or introducing new bugs and vulnerabilities.

Manuscript received May 4, 2023; Corresponding author: J. M. Pittman (email: pittman_jason@bah.com).

Automated software repair Harman (2010); Le Goues et al. (2013) is one possible solution to the burden of such manual processes. Indeed, there have been a variety of proposed software self-repair implementations such as GenProg Le Goues et al. (2013), Angelix Mechtaev et al. (2016a), and Neural Machine Translation Tufano et al. (2019) techniques. More recently, Yasunaga and Liang introduced Break-It-Fix-It (BIFI) Yasunaga & Liang (2021). This study is of particular interest for three reasons. The authors offered a detailed description of their self-repair implementation. The work also referenced a full GitHub repository containing source training data, BIFI code, and trained models. As well, Yasunaga and Liang specifically recommend future work explore to what extent BIFI generalizes to other domains. However, Xia et al. Xia et al. (2022) noted existing literature and the demonstrated self-repair paradigms therein may have limited generalizability.

One means to explore the generalizability of existing work is to reproduce the source study. Logically, a replication to verify results should precede reproduction though Plesser (2018). The process of replication involves duplicating a research study using the same techniques, materials, and experimental setup Lindsay & Ehrenberg (1993); Brooks et al. (1996); Gómez et al. (2010); Plesser (2018). The purpose is to verify the results and increase trust in their accuracy and consistency. However, replication is only feasible if the original research provides a thorough description of the materials, setup, and equipment used, enabling another researcher to carry out an identical investigation. In contrast, reproduction encompasses the act of recreating a study using various instruments, data sets, or techniques, with the objective of showing that the results are not specific to one particular implementation and can be generalized Lindsay & Ehrenberg (1993); Brooks et al. (1996); Gómez et al. (2010); Plesser (2018). Additionally, reproducing a study can also uncover any shortcomings or drawbacks in the original methodology. Accordingly, the purpose of this work is to provide a rigorous, scientific replication of BIFI Yasunaga & Liang (2021).

The rest of this work is organized into four sections. First, we offer summaries of seminal and relevant related work. Doing so establishes a conceptual framework for this study. Then, we describe our replication method and how the scientific inquiry present in this work emerged during the replication. Next, we present the qualitative and quantitative results of the replication. This is followed by our recommendations and ideas for future work.

II. Background

We need to describe three general areas of related work to properly situate this study in the literature. First, we discuss automated software repair as a field of study. In doing so, we establish a foundation for conceptualizing the dominant lines of inquiry present in the literature. Next, we expand on the specific background related to Yasunaga and Liang Yasunaga & Liang (2021). In this regard, we aim to highlight results and conclusions for comparison after the replication is completed. Last, we provide an overview of scientific replication. The overview will make clear why replication is vital to the field and how software engineering is best replicated.

A. Automated Software Repair

In order to reduce the amount of time developers spend manually identifying and patching bugs, a variety of techniques have been devised to conduct automatic software repair. Here we briefly review the three major approaches in this field: search-based repair, semantics-based repair, and learning-based repair.

1) Search-Based: Search-based approaches to automatic software repair rely on syntactic analyses of the underlying buggy program. Based on the syntax of the original code, these methods first generate large pools of candidate patches for a given bug via processes like source manipulation or modification of abstract syntax trees Ding et al. (2019). Then the search space is traversed to identify the best patch from the pool of candidates, either randomly or using heuristic or other optimization strategies, including genetic programming. A patch is presumed to be successful if it passes a suite of provided test cases. Many search-based algorithms for software repair have been proposed Le Goues et al. (2012); Liu et al. (2018); Mehne et al. (2018); Qi et al. (2014); a common challenge faced by these programs is that as the search space expands, it can become very difficult to navigate effectively Le et al. (2018).

2) Semantics-Based: In contrast to search-based approaches, semantics-based approaches begin with a semantic, rather than syntactic, analysis of the original code. These methods use information derived from evaluation against test suites and symbolic execution to establish semantic constraints, which are then used to guide the generation of possible repairs Le et al. (2018). These repairs are constructed using program synthesis techniques like template-based synthesis and component-based synthesis. While semantics-based methods offer greater precision when compared with their search-based counterparts, they are inherently limited by the power of the underlying semantic analyzers Ding et al. (2019). In addition, like search-based algorithms, semantics-based algorithms consider a patch to be correct if it passes every case in the test suite. This means that both approaches are prone to produce patches that are overfit to the test suite, which is usually incomplete. Some popular semantics-based automatic software repair programs include Angelix Mechtaev et al. (2016b) and SemFix Nguyen et al. (2013).

3) Learning-Based: In recent years, a third approach to automatic software repair has taken root, driven by the entrance of machine learning researchers to this field of study. Deep learning practices have been used both to augment and to replace components of more traditional search-based or semantics-based algorithms, and these techniques have piqued interests in both academia and industry, including inside major companies like Facebook Marginean et al. (2019) and Google Mesbah et al. (2019). Deep learning solutions are admired for their generalizability; unlike older methods, they do not require domain-specific knowledge about programming languages, error types, or common patches Namavar et al. (2021). *Break-It-Fix-It* Yasunaga & Liang (2021), the work we aim to replicate in this paper, is fundamentally a learning-based approach to the automatic program repair problem, and displays some of the flexibility just mentioned, as the original authors apply their algorithm to correct bugs in both C and Python code.

B. Break-It-Fix-It

In their paper Break-It-Fix-It: Unsupervised Learning for Program Repair Yasunaga & Liang (2021), Yasunaga and Liang attempt to address one of the most pervasive problems in the field of software repair - overfitting. Many of the currently available automated repair algorithms were trained on small datasets due to the costly nature of obtaining paired `<bad code, good code>` data, which has traditionally required manual labeling Mesbah et al. (2019). As a result, these algorithms may fail to find patterns or generalize to unseen examples Pu et al. (2016). Recognizing the limitations imposed by a scarcity of true labeled data pairs, previous authors have augmented their datasets with synthetically generated pairs, where broken code examples are produced by applying random or heuristically-guided perturbations to existing examples of good (i.e. error-free) code Gupta et al. (2017). However, the distribution of errors in synthetically-generated bad code often does not align with the distribution of errors seen in real examples of faulty code. Thus, repair algorithms trained on these synthetic datasets fail to perform well when deployed in the real world Yasunaga & Liang (2020).

Yasunaga and Liang address the shortcomings of earlier synthetic datasets in Break-It-Fix-It (BIFI) by training a *breaker* to, given good code, generate realistic examples of bad code. The synthetic dataset produced by the breaker is then used to train a software repair algorithm, or fixer. More specifically, they formulate their task and approach as follows:

<u>Input</u>: An unlabeled dataset of code examples D and a critic (e.g. a code analyzer or compiler) c such that for $x \in D$:

$$c(x) = \begin{cases} 1, & x \text{ has errors} \\ 0, & x \text{ has no errors.} \end{cases}$$

The critic c can be used to partition D into a set of correct code snippets and set of erroneous code snippets; that is,

$D = \{D_{\text{good}}, D_{\text{bad}}\}$. [1]

Goal: Learn a fixer f that takes a code snippet x with $c(x) = 0$ and outputs $f(x)$ such that $c(f(x)) = 1$, while minimizing the edit distance $d(x, f(x))$. [2]

Approach:

0) **[Initialize]** Start with a fixer f_0 from prior work.
1) **[Train]** Repeat the following steps for k rounds. Let i indicate the round number, starting with $i = 1$.
   a) Apply f_{i-1} to $x \in D_{\text{bad}}$. If $c(f_{i-1}(x)) = 1$, save the pair $(x, f_{i-1}(x))$. Let D_i be the dataset of (bad, corrected) code snippets produced in this step.
   b) Train a breaker b_i on D_i.
   c) Apply b_i to $x \in D_{\text{good}}$. If $c(b_i(x)) = 0$, add the pair $(b_i(x), x)$ to D_i.
   d) Train a fixer f_i on D_i
2) **[Evaluate]** Measure the accuracy of f_k on a held out set of real code snippets with errors.

The approach taken in BIFI is similar to the technique of backtranslation, in which one uses a target-to-source model to generate noisy sources, and then uses these noisy sources to train a source-to-target model Lample et al. (2017). BIFI improves on backtranslation in two primary ways: (i) BIFI uses the critic to verify that additions to D_i in steps 1a and 1c are actually$(x_{\text{wrong}}, x_{\text{correct}})$ pairs, whereas this is not guaranteed in backtranslation (ii) BIFI trains the fixer f_i on pairs of (real erroneous code, synthetic fixed good) in addition to pairs of (synthetic erroneous code, real good code), whereas backtranslation only trains the fixer on the latter set of examples.

Yasunaga and Liang assert that BIFI achieves 71.7% accuracy on DeepFix Gupta et al. (2017), a 5.6% improvement over the current state-of-the-art algorithm, as well as 90.5% accuracy on Github-Python, a dataset of three million python code snippets introduced in the original BIFI paper. Our study aims to replicate their methods and results.

C. *Scientific Replication*

Generally, in the scientific community, there is a low regard for scientific replications Lindsay & Ehrenberg (1993). As researchers often seek out new discoveries, they are inclined to wonder where the innovation lies in replicating a study exactly how it was originally conducted. However, if this were the case, if replications were truly identical to the original study, all conditions (e.g., time, testing environment) would need to be the same. Thus, all replications must involve some level of variation in the conditions. Once we accept that replication isn't merely repeating the *exact* same study, we can take advantage of the differences in the study conditions and note that despite these differences, the same results were obtained Lindsay & Ehrenberg (1993). Replication not only validates the original findings but establishes an increased range of conditions for which the findings hold, thereby extending the scope of the work Lindsay & Ehrenberg (1993); Brooks et al. (1996).

According to Gómez, Juristo, and Vegas there are five notable elements in software engineering experimentation that form the structure of an experiment and may vary in replication: Site, Experimenters, Apparatus, Operationalizations, and Population Properties. Site and Experimenters account for the experiment location and who is conducting the experiment, respectively Gómez et al. (2010). Apparatus is defined by the "experimental design, instruments, forms, materials, experimental objects and procedures used to run an experiment" Gómez et al. (2010). Operationalizations describe the independent and dependent variables that are used to measure the effects of the experimentGómez et al. (2010). Population Properties refers to the subjects and experimental objects, where subject properties are subject type and experience and experimental objects are "specifications, design documents, source codes, programs or any other artefact related to the software development" Gómez et al. (2010). We use these elements as a framework to ensure our BIFI replication research aligns with the scientific method.

III. Method

While we are interested in validating the results of the original study, the goal of our work is to establish an increased range of different conditions in which the findings of BIFI will hold. Simply put, we wish to know if it is possible to replicate the results of BIFI using the same data in a different testing environment. In seeking to replicate these results, we will also be evaluating whether the researcher's repository contains explicit enough instructions and links to source code that aid the clear replication of their study. Successful replication of BIFI will help us to determine if the results can be generalized and may create new avenues for potential work and innovation.

Literature suggests Lindsay & Ehrenberg (1993) that it is best to start with closer replications in the initial stages of replication because the more differentiated ones may not replicate successfully. Therefore, we first opted to complete a close replication in which we kept relatively all the known conditions of the study the same. However, due to issues with the dependencies we were unable to follow the documented protocol from the original experimenters Yasunaga & Liang (2021). Consequently, the replication design pivoted from one where little variance existed to one that has variance but follows the overall method of the reference experiment Gómez et al. (2010).

Overall, we sought to conduct a replication of the BIFI study to assess the generalizability of the research and the potential for further use of the self-repair function. While we came across some dependency issues at first, we were able to produce results that corresponded to the ones reported in the BIFI paper by using their trained models. However, based on the author's provided documentation, we were unable to train

[1]Note: the code snippets in D_{good} and D_{bad} have no relation to each other. We do NOT have pairs of snippets $(x_{\text{bad}}, x_{\text{good}})$ where x_{bad} is a piece of code with errors and x_{good} is a corrected, error-free version of x_{bad}.

[2]In software repair, we want the fixer f to be semantics-preserving, but since this is very difficult to check, Yasunaga and Liang use edit distance as a proxy measure, under the assumption that if the edit distance between two code snippets is small, they are semantically similar.

our own models and therefore the scope to which BIFI can be expanded may be limited. Details are provided in the next section.

IV. Results

We present the results in two sections. The first covers the replication of the BIFI Yasunaga & Liang (2021) initial fixer training. Then, in the second section, we reveal the results of evaluating the fixer with the models we trained as well as the trained models Yasunaga and Liang provided as part of the materials. Along the way, we outline critical findings and important details that contribute to answer the research question.

A. Training the BIFI fixer

We followed the steps as outlined in the BIFI GitHub repository. However, we encountered numerous errors. Some errors were correctable and we advanced to the next step. Ultimately, however, some errors were terminal. The details are as follows.

We cloned the BIFI repository Yasunaga & Liang (2021). Next, we followed the BIFI environment creation commands indicated in the repository up to `pip install -e`. Installing in editable mode produced errors because of a missing file. We corrected the error by back-tracing to the most likely version of the `fairseq` package in Facebook's GitHub and manually insert the missing files (Table I) in the `fairseq\data` directory. To back-trace, we triangulated the `fairseq` release based on the time of BIFI publication, the BIFI repository last commit timestamp, and the `fairseq` version release dates. We were able to build `fairseq` successfully after inserting these files from the Facebook repository into the local BIFI environment.

TABLE I
Fairseq file replacements and the versions

	Versions	
File	BIFI	Replaced
data_utils_fast.pyx	0.10.2	0.10.2
token_block_utils_fast..pyx	0.10.2	0.10.2
dictionary.py	0.10.2	0.10.2

Following the install of `numpy` and `editstance`, we downloaded the minimal dataset from the BIFI repository. We then created the set of *round* directories per the layout diagram Yasunaga and Liang provided. However, executing the python statements from `run-round0.sh` produced errors. In this phase of the replication, we encountered import errors for various `fairseq` subordinate packages. We traced the errors to additional missing files in the same directory as before. At this point, we opted to simply copy the Facebook `\data` subdirectory for version 0.10.2 into the BIFI local environment. While this resolved missing file issues, we then encountered a series of package import obstacles. Fixing these, while conceptually possible, would require editing BIFI source code which we elected to not do given the intended goal and methodology of this replication.

For completeness, we also tried to implement the BIFI training procedure in an updated local environment. The environment reflected a current software stack (i.e., python 3.10.6, fairseq 0.12.2, numpy 1.24.1). This was not successful due to significant differences between package versions and the BIFI source code architecture. Therefore, getting the BIFI system to a working state based on replicating training the models was not possible with the materials available in the BIFI GitHub repository.

However, one of the authors (Yasunaga) responded to an email inquiry regarding the above. The response included a Docker compose file and a `fairseq` package build from March 2020. Upon inspection, we observed six differences between the Docker compose directives, the BIFI repository instructions and contents, as well as the details in the original paper. We indexed all the components and range of versions (Table II) before attempting further replication of model training.

TABLE II
Breakdown of BIFI environmental components

	Versions	
Component	Docker	GitHub & Paper
Operating System	Ubuntu 16.04	NA[1]
CUDA	10.1	NA[1]
cuDNN	7	NA[1]
miniconda	latest	NA[1]
Python	3.7.7	3.7.7
pytorch	1.4.0	1.4.0
torchvision	0.5.0	0.5.0
tqdm	4.53.0	latest
numpy	1.20.1	1.20.1
editdistance	latest	latest
fairseq[2]	0.9.x	NA[1]

[1] NA: information is not available in the BIFI GitHub repository materials.
[2] Version information had to be inferred from file date stamps and other file object clues contrasted against package release history.

We then decided to pursue two model training paths given the updated BIFI component information. First, we used the Docker compose file to deploy a container environment as recommended by Yasunaga. Second, we took the `fairseq` package from Yasunaga and overwrote the same directory in the local GitHub repository clone. In both cases, we injected the training data as detailed by the authors. Training completed without error for both the Docker container and the local clone environments. Further, both training paths yielded models identical in file size and object metadata to the trained models provided in the BIFI repository.

B. Evaluating the BIFI fixer

We were able to run the BIFI fixer evaluation without error using all three trained models. We used the full data download from the BIFI repository. Then, we ran Yasunaga and Liang's final BIFI evaluation step against the models trained in our Docker and local repository clone environment. More specifically, we ran the `python src/c005__eval_fixer.py` routine using the `--round_name round2-BIFI-part2` option. Our intent was to replicate Yasunaga and Liang's *round-2* accuracy in Total (90.5%), for Unbalanced Parentheses (94.2%), Indentation Error (85.9%), and Invalid Syntax (93.5%). The results from our replication trials are as follows (Table III).

TABLE III
REPLICATION RESULTS

	Accuracy		
Category	Docker	Local	Yasunaga & Liang (2021)
Total	90.5%	90.5%	90.5%
Unbalanced Parentheses	94.2%	94.2%	94.2%
Indentation Error	85.9%	85.9%	85.9%
Invalid Syntax	93.5%	93.5%	93.5%

Considering the outcomes of this replication, we reached several conclusions. These are discussed in the next and final section of this work. We also detail where our assumptions and limitations differ from those offered by Yasunaga and Liang. As well, as a replication of existing work, we have unique recommendations and ideas for future research in software self-repair. In any case, we begin by offering a brief summary as a grounding mechanism for conceptualizing our results specifically and software self-repair in general.

V. CONCLUSION

Today, software developers spend upwards of 50% of their time finding and patching bugs and vulnerabilities Britton et al. (2013); Alhefdhi et al. (2020). Automated software repair Harman (2010); Le Goues et al. (2013) is intended to reduce or eliminate such a labor burden, thus freeing trapped development capacity. One current attempt at self-repair was Yasunaga and Liang's Break-It-Fix-It (BIFI) Yasunaga & Liang (2021). The authors demonstrated a self-repair function capable of repairing errors with greater than 90% accuracy. However, Yasunaga and Liang specifically recommend future work explore to what extent BIFI generalizes to other domains. Echoing such a sentiment, Xia et al. Xia et al. (2022) noted existing literature, and the demonstrated self-repair paradigms therein, may have limited generalizability.

For that reason, the purpose of this work was to conduct a rigorous, scientific replication of BIFI. We were largely successful in this effort, with one major caveat. On one hand, our replication succeeded in validating the BIFI evaluation results using just the code, documentation, and pre-trained models available in the BIFI repository published by the authors. We take such an outcome as one step completed towards assessing the generalizability of software self-repair in general and BIFI in specific. On the other hand, we were initially unable to replicate the model training protocol for BIFI due to missing files and package import errors in the public version of the code. These issues were only resolved once we contacted the BIFI authors and received a copy of the Docker compose file used to generate the environment used in their experiments. Using this compose file, we were able to construct a viable model training environment and build the models using directions in the BIFI repository [3].

[3]We did not receive permission to share the Docker compose file.

Though we eventually succeeded in training the BIFI model, and achieved the same performance the authors claim, the successful implementation of BIFI used significantly outdated versions of Python, numpy and fairseq. Furthermore, attempts to reimplement BIFI with updated versions of these packages proved fruitless. Based on this result, we surmise BIFI is not actively maintained in the software engineering sense. The last repository commit was on August 31, 2021. The technology stack (i.e., Python, fairseq, and so forth) have evolved since that commit and without the Docker compose file it is not possible to use BIFI independent of its pre-trained models.

A. *Assumptions and Limitations*

Of course, we have assumed the GitHub repository provided by Yasunaga and Liang Yasunaga & Liang (2021) contains the correct codebase and training data. The assumption is reasonable because the commit history is aligned with the publication date. Further, a thorough search did not uncover any additional code repositories. We also assume the errors encountered during the BIFI training procedure are unresolvable given publicly available documentation on BIFI. While it might be possible to engineer a Python environment suitable for BIFI training, the source paper lacks complete information particularly as it relates to the fairseq package, and the Docker compose file we used to resolve issues with fairseq is not published at this time. Therefore, potential resolutions for the errors are limited. On a related point, we recognize the limitation of our results in establishing generalizability for BIFI. Achieving identical results, while positive, do not fully establish external, general application of the automatic software self-repair tool.

B. *Recommendations*

Given the challenges that emerged during the replication effort, it is important that researchers consider the pace at which technology evolves. Research that provides clear, executable documentation and timeless code will help to ensure that future technology can build on existing work. Therefore, our BIFI replication work should be extended to explore the technology debt (tech debt) that exists in current research. Briefly, tech debt speaks to the measurable value associated with short term technology design or implementation decisions in exchange for increased maintenance costs long term. At an extreme, tech debt is highest when no maintenance takes place or technology is developed and abandoned. Thus, if the aim of self-repair functions is to alleviate the time spent by developers on identifying and patching bugs, we must first look at the sustainability of the research in this area.

A first step to doing this would be to find the percentage of relevant papers (with code) that are in a tech debt state. From here, future work should explore which of this work has been updated and if the patches stay true to the original functionality of the code. Similar to the Docker file shared by Yansunaga, practitioners and researchers alike should consider using a container environment to better preserve their work and increase the likelihood of successful replication. Utilizing a container environment enables developers to share code and its dependencies with others, thereby reducing the potential for errors and hopefully slowing down the pace at which the research enters a state of technical debt. On this note, it may be worthwhile to examine how many self-repair function papers utilize micro-server architectures. Once we have an understanding for the ubiquity of tech debt in recent self-repair work, it can be extended into future research such as self-repair functions that automatically review code and are built into the memory environment of the workspace. Such innovation will only evolve if we write code with tomorrow in mind so that our systems of today can be used in the future. Finally, it may be of value to the field if future work is able to reproduce the BIFI accuracy measures using an updated technology stack. Doing so will invariably require changes to the BIFI source code but will add to the body of evidence for generalizability of the self-repair function.

References

Abuasad, A., & Alsmadi, I. M. (2012). Evaluating the correlation between software defect and design coupling metrics. In *2012 international conference on computer, information and telecommunication systems (cits)* (pp. 1–5).

Alhefdhi, A., Dam, H. K., Le, X.-B. D., & Ghose, A. (2020). Adversarial patch generation for automatic program repair. *arXiv preprint arXiv:2012.11060*.

Alves, H., Fonseca, B., & Antunes, N. (2016). Software metrics and security vulnerabilities: dataset and exploratory study. In *2016 12th european dependable computing conference (edcc)* (pp. 37–44).

Britton, T., Jeng, L., Carver, G., & Cheak, P. (2013). Reversible debugging software "quantify the time and cost saved using reversible debuggers". *University Cambridge: Cambridge, UK*.

Brooks, A., Daly, J., Miller, J., Roper, M., & Wood, M. (1996). Replication of experimental results in software engineering. *International Software Engineering Research Network (ISERN) Technical Report ISERN-96-10, University of Strathclyde*, *2*.

Ding, Z. Y., Lyu, Y., Timperley, C., & Le Goues, C. (2019). Leveraging program invariants to promote population diversity in search-based automatic program repair. In *2019 ieee/acm international workshop on genetic improvement (gi)* (p. 2-9). doi: 10.1109/GI.2019.00011

Gómez, O. S., Juristo, N., & Vegas, S. (2010). Replications types in experimental disciplines. In *Proceedings of the 2010 acm-ieee international symposium on empirical software engineering and measurement* (pp. 1–10).

Gupta, R., Pal, S., Kanade, A., & Shevade, S. (2017, Feb.). Deepfix: Fixing common c language errors by deep learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). Retrieved from `https://ojs.aaai.org/index.php/AAAI/article/view/10742` doi: 10.1609/aaai.v31i1.10742

Harman, M. (2010). Automated patching techniques: the fix is in: technical perspective. *Communications of the ACM*, *53*(5), 108–108.

Lample, G., Conneau, A., Denoyer, L., & Ranzato, M. (2017). *Unsupervised machine translation using monolingual corpora only.* arXiv. Retrieved from `https://arxiv.org/abs/1711.00043` doi: 10.48550/ARXIV.1711.00043

Le, X.-B. D., Thung, F., Lo, D., & Goues, C. L. (2018). Overfitting in semantics-based automated program repair. In *Proceedings of the 40th international conference on software engineering* (p. 163). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3180155.3182536` doi: 10.1145/3180155.3182536

Le Goues, C., Forrest, S., & Weimer, W. (2013). Current challenges in automatic software repair. *Software quality journal*, *21*(3), 421–443.

Le Goues, C., Nguyen, T., Forrest, S., & Weimer, W. (2012). Genprog: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, *38*(1), 54-72. doi: 10.1109/TSE.2011.104

Lindsay, R. M., & Ehrenberg, A. S. (1993). The design of replicated studies. *The American Statistician*, *47*(3), 217–228.

Liu, K., Koyuncu, A., Kim, K., Kim, D., & F. Bissyandé, T. (2018). Lsrepair: Live search of fix ingredients for automated program repair. In *2018 25th asia-pacific software engineering conference (apsec)* (p. 658-662). doi: 10.1109/APSEC.2018.00085

Marginean, A., Bader, J., Chandra, S., Harman, M., Jia, Y., Mao, K., … Scott, A. (2019). Sapfix: Automated end-to-end repair at scale. In *2019 ieee/acm 41st international conference on software engineering: Software engineering in practice (icse-seip)* (p. 269-278). doi: 10.1109/ICSE-SEIP.2019.00039

McGuire, M. (2022, Dec). *What is the cost of poor software quality in the u.s.?* `https://www.synopsys.com/blogs/software-security/poor-software-quality-costs-us/`. (Accessed: 11-01-2023)

Mechtaev, S., Yi, J., & Roychoudhury, A. (2016a). Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *Proceedings of the 38th international conference on software engineering* (pp. 691–701).

Mechtaev, S., Yi, J., & Roychoudhury, A. (2016b). Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *2016 ieee/acm 38th international conference on software engineering (icse)* (p. 691-701). doi: 10.1145/2884781.2884807

Mehne, B., Yoshida, H., Prasad, M. R., Sen, K., Gopinath, D., & Khurshid, S. (2018). Accelerating search-based program repair. *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, 227-238.

Mesbah, A., Rice, A., Johnston, E., Glorioso, N., & Aftandilian, E. (2019). Deepdelta: Learning to repair compilation errors. In *Proceedings of the 2019 27th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering* (p. 925–936). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3338906.3340455` doi: 10.1145/3338906.3340455

Misra, S. C., & Bhavsar, V. C. (2003). Relationships between selected software measures and latent bug-density: Guidelines for improving quality. In *International conference on computational science and its applications* (pp. 724–732).

Namavar, M., Nashid, N., & Mesbah, A. (2021). *A controlled experiment of different code representations for learning-based bug repair.* arXiv. Retrieved from `https://arxiv.org/abs/2110.14081` doi: 10.48550/ARXIV.2110.14081

Nguyen, H. D. T., Qi, D., Roychoudhury, A., & Chandra, S. (2013). Semfix: Program repair via semantic analysis. In *2013 35th international conference on software engineering (icse)* (p. 772-781). doi: 10.1109/ICSE.2013.6606623

Plesser, H. E. (2018). Reproducibility vs. replicability: a brief history of a confused terminology. *Frontiers in neuroinformatics*, *11*, 76.

Pu, Y., Narasimhan, K., Solar-Lezama, A., & Barzilay, R. (2016). *sk_p: a neural program corrector for moocs.* arXiv. Retrieved from `https://arxiv.org/abs/1607.02902` doi: 10.48550/ARXIV.1607.02902

Qi, Y., Mao, X., Lei, Y., Dai, Z., & Wang, C. (2014). The strength of random search on automated program repair. In *Proceedings of the 36th international conference on software engineering* (p. 254–265). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2568225.2568254` doi: 10.1145/2568225.2568254

Tufano, M., Pantiuchina, J., Watson, C., Bavota, G., & Poshyvanyk, D. (2019). On learning meaningful code changes via neural machine translation. In *2019 ieee/acm 41st international conference on software engineering (icse)* (pp. 25–36).

Xia, C. S., Wei, Y., & Zhang, L. (2022). Practical program repair in the era of large pre-trained language models. *arXiv preprint arXiv:2210.14179*.

Yasunaga, M., & Liang, P. (2020). *Graph-based, self-supervised program repair from diagnostic feedback.* arXiv. Retrieved from `https://arxiv.org/abs/2005.10636` doi: 10.48550/ARXIV.2005.10636

Yasunaga, M., & Liang, P. (2021). Break-it-fix-it: Unsupervised learning for program repair. In *International conference on machine learning* (pp. 11941–11952).

Microcontroller based Smart Coil Winder System

Dr.A.Rajamani[1] Ms.N.Saranya[2]

[1]HoD, EEE Department, PSG polytechnic College

[2] Student, EEE Department, PSG College of Technology

Abstract - The aim of this paper is to develop the control circuit for smart automatic coil winder, which functions with respect to the PIC Microcontroller program. The conventional coil winding machine winds copper wire on a former and the former is attached to the iron rod, which is to be actuated manually. Moreover, the manual coil winding machine does not have any control circuit for the smart functionality. But in this design, the controller has more memory to operate and save the data and also to improve the winder program. Additionally, it also reduces the time of operation and the requirement of Manpower. This coil winder control circuit uses PIC Microcontroller (16F887), which runs the winder system according to main program and also the motor step sequencing program.

Key words: PIC Microcontroller, Stepper Motor, Stepper motor driver module, 16*2 LCD.

1.INTRODUCTION

In Electrical and Electronics Engineering, coil winding process is mandatory and it is used to wind electromagnetic coils and transformer coils. Coils are used as essential components in various circuits and also to provide the magnetic field for motors, transformers, generators, and in the manufacture of loudspeakers and microphones. The shape and dimensions of a winding are designed to fulfil the particular purpose. Parameters such as inductance, Q factor, insulation strength, and strength of the desired magnetic field greatly influence the design of coil windings. Coil winding can be structured into several groups regarding the type and geometry of the wound coil. Mass production of electromagnetic coils relies on automated machinery.Efficient coils minimize the materials and volume required for a given purpose. The ratio of the area of electrical conductors, to the provided winding space is called "fill factor". Since round wires will always have some gap, and wires also have some space required for insulation between turns and between layers, the fill factor is always smaller than one. To achieve higher fill factors, rectangular or flat wire can be used.

2.PROPOSED SYSTEM

In the speed running world everyone is considering the time factor as an important issue. To reduce this time or managing this time, reducing labour cost and reducing the labours, a small implementation this is mainly used in transformer producing industries and training institutions. We have proposed simple, low cost, low power consumption components. The system is PIC Microcontroller based automatic carriage movement of the coil winder. The carriage is placed on the coil winder it is moved by the help of a stepper motor is interfaced with the PIC Microcontroller. In the proposed project the carriage is moved by an automated way instead of using manual way. Its vision is used to reduce human effort and at the same time increase the productivity & accuracy levels that cannot be achieved with manual operations. The automatic coil winder in this article uses stepper motors to position the wire. The machine winds the coil within the size of the bobbin. The machine is controlled by the PIC16F877A microcontroller.

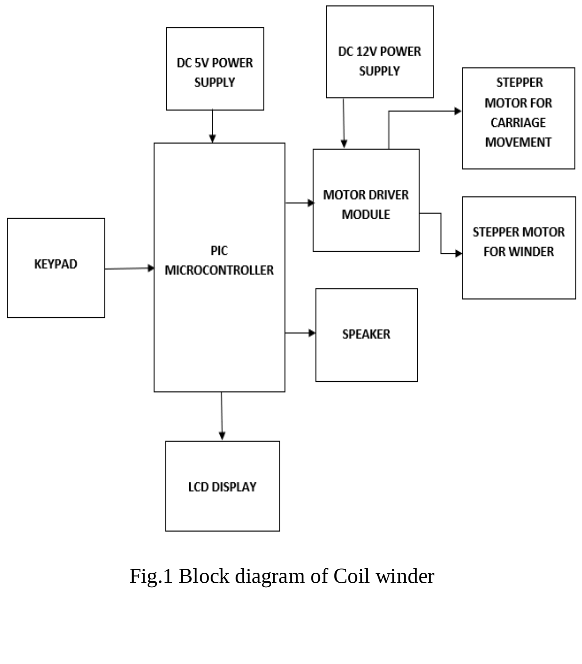


Fig.1 Block diagram of Coil winder

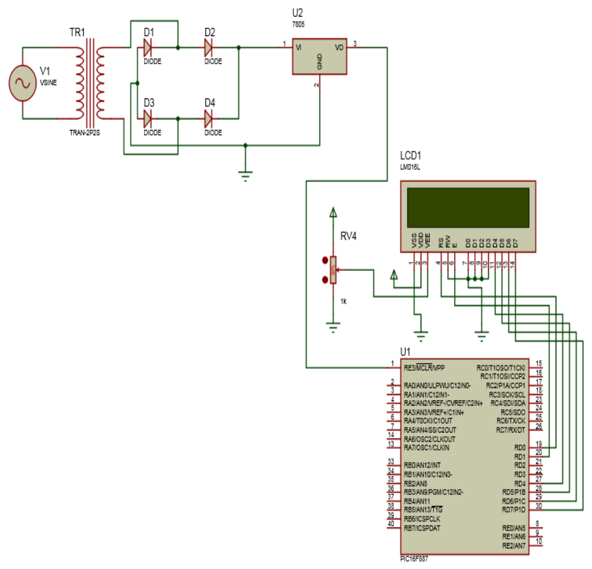


Fig.2 Circuit diagram of Interfacing LCD

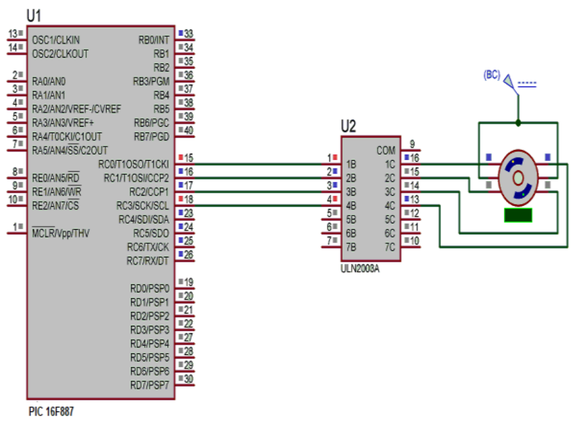


Fig.3 Circuit diagram of Interfacing Stepper motor

3.HARDWARE DESCRIPTION

This project consists of the following components.

a) *PIC Microcontroller (PIC16F887)*
b) *Stepper Motor (28BYJ-48)*
c) *Stepper Motor Driver Module (ULN2003)*
d) *16*2 LCD display*
e) *PIC Microcontroller programming kit*

a)PIC Microcontroller (PIC16F887)

PIC microcontrollers are a family of specialized microcontroller chips produced by Microchip Technology in Chandler, Arizona. The acronym PIC stands for "peripheral interface controller," although that term is rarely used nowadays. A microcontroller is a compact microcomputer designed to govern the operation of embedded systems in motor vehicles, robots, office machines, medical devices, mobile radios, vending machines, home appliances, and various other devices. A typical microcontroller includes a processor, memory, and peripherals.

The PIC microcontrollers appeal to hobbyists and experimenters, especially in the fields of electronics and robotics. Key features include wide availability, low cost, ease of reprogramming with built-in EEPROM (electrically erasable programmable read-only memory), an extensive collection of free application notes, abundant development tools, and a great deal of information available on the Internet. The PIC microcontrollers often appear under the brand name PIC Microcontroller.

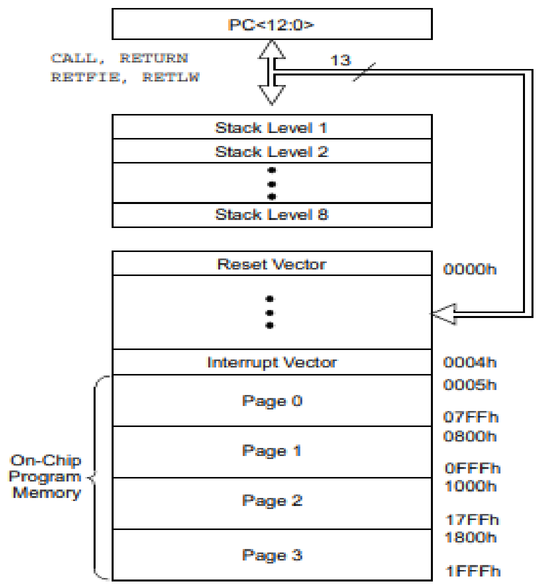


Fig.4 Program memory map and stack for the PIC16F887

b)Stepper Motor (28BYJ-48)

Stepper motor, also known as step motor or stepping motor, is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed. Switched reluctance motors are very large stepping motors with a reduced pole count, and generally are closed-loop commutated.

The most commonly used stepper motor is the **28-BYJ48 Stepper Motors**. It can be mostly used in DVD drives, Motion camera and many more places. The motor has a 4coil unipolar arrangement and each coil is rated for +5V hence it is relatively easy to control with any basic microcontrollers. These motors have a stride angle of 5.625°/64, this means that the motor will have to make 64 steps to complete one rotation and for every step it will cover a 5.625° hence the level of control is also high. However, these motors run only on 5V and hence cannot provide high torque, for high torque application you should consider the **Nema17 motors**.

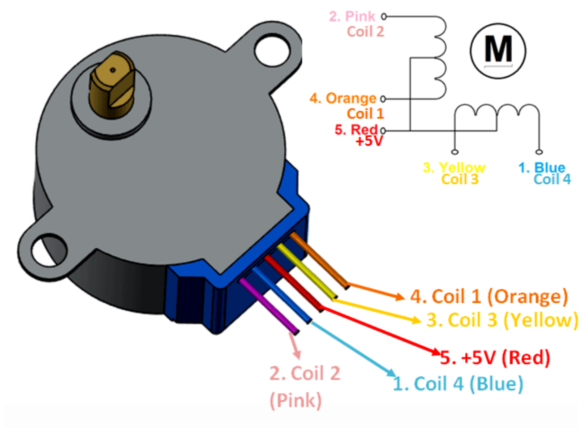


Fig.5 Pin out of 28BYJ-48 Stepper motor

c)Stepper Motor Driver Module (ULN2003)

Stepper motor drivers are specifically designed to drive stepper motors, which are capable of continuous rotation with precise position control, even without a feedback system. Our stepper motor drivers offer adjustable current control and multiple step resolutions, and they feature built-in translators that allow a stepper motor to be controlled with simple step and direction inputs. These modules are generally basic carrier boards for a variety of stepper motor driver Ics that offer low-level interfaces like inputs for directly initiating each step. An external microcontroller is typically required for generating these low-level signals.

The ULN2003 is high voltage, high current 3Darlington arrays each containing seven open collectors3Darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout. Fig. 5.1 shows the ULN2003 stepper motor driver board. These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high-power buffers.

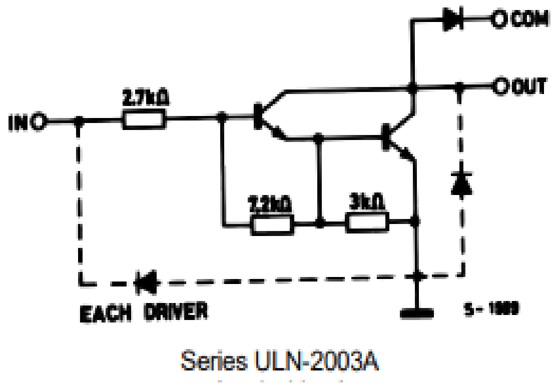


Fig.6 Schematic diagram of ULN2003 driver module

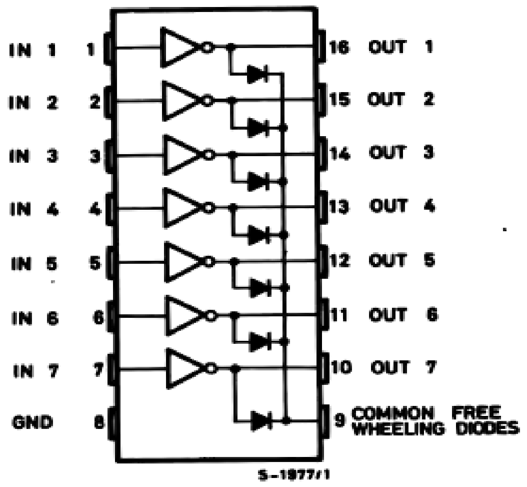


Fig.7 Pin diagram of ULN2003

d)16*2 LCD Display

A liquid crystal display or LCD draws its definition from its name itself. It is combination of two states of matter, the solid and the liquid. LCD uses a liquid crystal to produce a visible image. Liquid crystal displays are super-thin technology display screen that are generally used in laptop computer screen, TVs, cell phones and portable video games. LCD's technologies allow displays to be much thinner when compared to cathode ray tube (CRT) technology. Liquid crystal display is composed of several layers which include two polarized panel filters and electrodes. Light is projected from a lens on a layer of liquid crystal. This combination of coloured light with the grayscale image of the crystal (formed as electric current flows through the crystal) forms the coloured image. The LCD is either made up of an active matrix display grid or a passive display grid. Most of the Smartphone's with LCD display technology uses active matrix display, but some of the older displays still make use of the passive display grid designs. Most of the electronic devices mainly depend on liquid crystal display technology for their display.

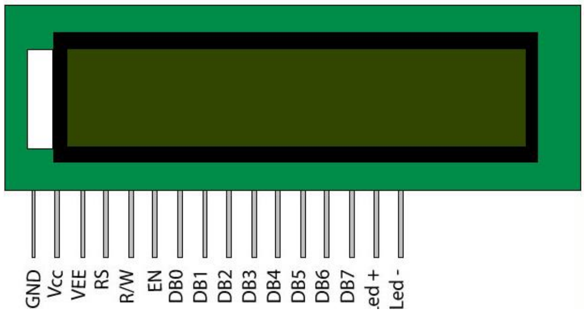
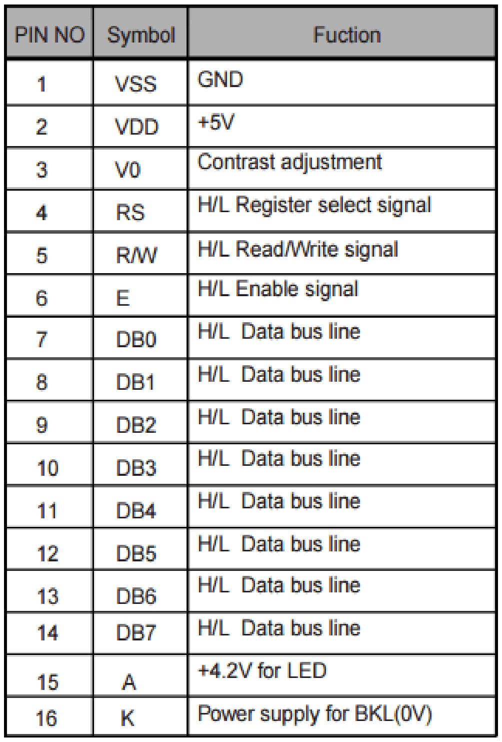


Fig.8 Pin out of LCD

The liquid crystals are the organic compound which is in liquid form and shows the property of optical crystals. The layer of liquid crystals is deposited on the inner surface of glass electrodes for the scattering of light. The liquid crystal cell is of two types; they are Transmittive Type and the Reflective Type.

TABLE 1 PIN DETAILS OF LCD



PIN NO	Symbol	Fuction
1	VSS	GND
2	VDD	+5V
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	R/W	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	+4.2V for LED
16	K	Power supply for BKL(0V)

e)PIC Microcontroller programming kit

PIC kit is a family of programmers for PIC microcontrollers made by Microchip Technology. They are used to program and debug microcontrollers, as well as program EEPROM. Some models also feature logic analyser and serial communications (UART) tool. The people who develop open-source software for the PIC kit use a mailing list for collaboration.

The PIC kit 2 introduced in May 2005replaced the PIC kit 1. Fig. 3.2 shows the PIC kit- 2 programmer kit. The most notable difference between the two is that the PIC kit 2 has a separate programmer/debugger unit which plugs into the board carrying the chip to be programmed, whereas the PIC kit 1 was a single unit. This makes it possible to use the programmer with a custom circuit board via an in-circuit serial programming (ICSP) header. This feature is not intended[3] for so-called "production" programming, however.

The PIC kit 2 uses an internal PIC18F2550 with Full Speed USB. The latest PIC kit 2 firmware allows the user to program and debug most of the 8- and 16-bit PIC micro and ds PIC members of the Microchip product line. The PIC kit 2 is open to the public, including its hardware schematic, firmware source code (in C language) and application programs (in C# language). End users and third parties can easily modify both the hardware and software for enhanced features. e.g. Linux version of PIC kit 2 application software, DOS style CMD support, etc.

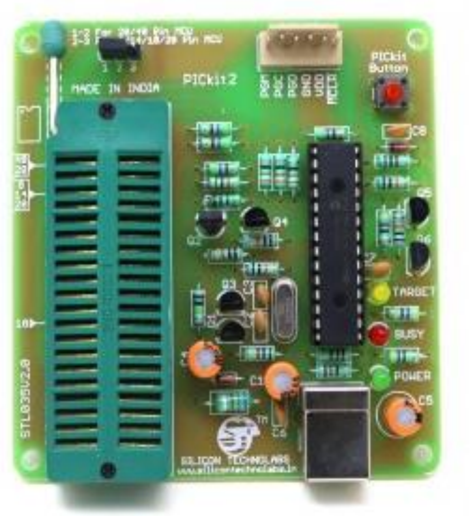


Fig.9 PIC kit

4.ADVANTAGES

- Reduction of material cost.
- Reduction of overall cost.
- Increased production.
- Increased storage capacity.
- Increased safety.
- Reduce in fatigue.
- Improved personnel comfort
- Efficiency is improved.
- Fully automatic winding machine saves energy.

5.APPLICATIONS

- It is very useful in Transformer manufacturing, to wind the transformer quickly.
- To wind the stator or rotor in motor or submersible pump.
- To wind the condensers coil and fan coils quickly.
- It is very useful in small scale industries where ever winding coils are used.
- It used to train students for wind the small transformers & relay coils.

6.CONCLUSION

The main objective of this machine is to replace the manual labour and optimize the process. The inference is that, this automated system has increased the production and also provided solution for lack of human labour for such hectic jobs is compensated. In general, it needs one worker for one machine but by implementing this automation it needs one worker for four machines. All electrical components such as stepper motor, motor driver module and the microcontroller were assembled and the machine was tested. The manual carriage movement of coil winder is automated for high reliability. The project can be extended by using another one stepper motor for the winder and a buzzer is provided for intimating the worker that the process is over.

REFERENCES

1. Benbouzid MEH. A review of stepper motor signature analysis as a medium for faults detection. IEEE Trans Ind Electron 2000; 47: 984-93.
2. Krishnamurthy TN. Fabrication of Low-Cost Filament Winding Machine. International Journal of Recent Trends in Electrical and Electronics Engineering 2014; 4: 30-9.
3. Hong H, Chao-Ming C. Design Fabrication and Failure Analysis of Stretchable Electrical Routings. Sensors 2014; 14: 11855-77.
4. Mulik P, Kamble RK. Development of Automatic Transformer Winding Machine. International Journal of Innovations in Engineering Research and Technology 2015; 2: 1-8.
5. Joshi NS, Bulbule CB, Domale SD. Automatic Transformer Winding Machine International Journal for Research in Applied Science and Engineering Technology 2015: 3: 942-7.
6. Ikhankar P, Golhar R. Automation in Manufacturing of Winding. International Journal for Scientific Research and Development 2016; 4:453 – 6.
7. Good, JK; Roisum, David R. (1 January 2008). Winding: Machines, Mechanics & Measurements (1ST ed.). Lancaster, PA: DEStech Publications. p. 478. ISBN 978-1-932078-69-5. Retrieved 9 January 2015.
8. Querfurth, William (1954). Coil Winding: A Description of Coil Winding Procedures, Winding Machines and Associated Equipment. University of Michigan: G. Stevens Mfg. Company.
9. Tarun, Agarwal. "Stepper Motor – Types, Advantages & Applications".

Programming Language Conversion using NLP

ABHIJIT BANERJEE[1], MADHUBAN MUKHERJEE[1], APARAJITA BANERJEE[1], MD. AALISHAN RAZA[2], SUCHETA BHOWMICK[1], SAKSHI BHAGAT[1], SUDIPTA BASU PAL[3]
[1]Department of CST
University of Engineering and Management, Kolkata India
[2]Department of CSIT
University of Engineering and Management, Kolkata India
[3]Department of CST and CSIT University of Engineering and Management, Kolkata India
banerjeeabhijit111@gmail.com, madhubanmukherjee77@gmail.com, aparajitab535@gmail.com, aalishan69raza@gmail.com, suchetabhowmick99@gmail.com, sakshibhagat9873@gmail.com, sudipta_basu68@yahoo.com

Abstract- Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do. NLP models language computationally and deals with linguistic features of computation. Once a computer learns to do mathematical calculations it can perform manycomplex and big calculations much faster than humans. Similarly, once computer starts to understand the human languages it can process all aspects of that language much faster than humans also opening a large number of possibility. So it cuts down on employment as one computer is capable of giving an output 10 times faster than a human can. So it benefits the employer not only financially but also by giving extremely accurate and faster outcome. Here we lay out an overall architecture to explain the overall processing. So now we take a look at the two general classes of systems they are special-purpose system and general-purpose system, explaining how they differ and their relative advantages and disadvantages. After that we point at the few remaining problems that require additional research. Finally, we conclude by discussing when natural language processing technology can be practically used at various levels .We also discuss about when it will become commercially practical, and what will be the cost to practically use this technology.

The techniques specifically developed for analysing and understanding the inner-workings and representations acquired by neural models of language is EMNLP 2018 BlackboxNLP. The approach includes: investigating the impact on the performance of neural network on systematic manipulation of input and also testing whether the interpretable knowledge can be decoded from intermediate representations to propose modifications to make the knowledge state or generated output more and also to examine the performance of networks on simplified or formal languages.

In the following report we aim to convert a program of a given language to an equivalent program of another language. For that we have taken help of NLP that is Natural Language Processing. By using Natural Language Tool Kit, we have successfully identified the variables, datatypes, operators, keywords, indentations. We have also discussed various aspects and domains of NLP and some real-world applications of it.

Keywords: Language Conversion, NLP, Keyword detection, valid Identifier checking, operators' identification,checking constants, Datatypes identification, NLTK, tokens.

I. INTRODUCTION

NLP orNatural Language Processingis a branch of AI that gives the machine the ability to read, understand and derive meaning from human languages. Every day we exchange data via social media or other devices. These data is extremely useful and data experts implement this data to machine so that they can mimic human linguistic behaviour and it saves so much time and effort. It involves programming techniques to create a model that can understand the language just like normal human beings. It can even classify the contents and even generate and create new composition in human based language.

we don't even realize the wide use of NLP. we basically use it daily, like while using autocorrect method in mobile phones or checking if any document is going to be plagiarised or not. The Prolong language11 was originally invented for NLP applications. Its syntax is especially suited for writing grammars, although, in the easiest implementation mode rules must be phrased differently from those intended for a yacc-style parser. Top-down parsers are easier to implement than bottom-up parsers but are much slower.

A. Application of NLP

Talking about the use of Natural Language Processing involves how it has evolved in the era of technology. The basic aim of NLP is not only to understand the single word to word but also to have the capability to understand the context based on syntax, grammar, etc of those words, sentences, and paragraphs to give the desired result out of it.

In short, NLP gives the machines the ability to read, understand, drive meaning from the text and often generate the text from various languages.

NLP-enabled software assists us in our daily lives in various ways, for example:

- Personal Assistants (Speech recognition): Siri, Cortana, and Google Assistant
- Machine Translation: Google Translator
- Grammar check: Grammarly app
- Autosuggestion (Sentiment analysis): In search engines, Gmail, Developer's IDE
- Making Chat bots

B. Working Process of NLP

The working process of NLP not as easy as it may seem. Basically, it works in 3 steps, Speech recognition → NLU → NLG
| |

| (Natural Language Generation)
(Natural Language Understanding)

Computers do not directly understand the words and sentences which belong to human languages. The computer only understands binary numbers as 0s and 1s. So, we had to initially develop a way for computers to understand the words. Word representation is a widely used implementation for this problem. Word representation is a technique to represent a word with a vector and each word has its unique vector representation. Text and word representation are essential for making computers understand words and thus we need to encode words into a format understandable by the computers. One-hot encoding is one such technique used to convert categorical data into numerical data. The numerical data is then used by the algorithms to learn and predict.

In short, we encode the input into numerical form and train our neural network model with that. And the we decode the output of the NN to get our desired result.

C. Encoding language into numbers

We can encode the language into numbers in many ways. The most common is to encode by letters. we generally use ASCII or Unicode value for that. But due to the presence of antigram the same number represents two words in a different order, which might make building a model to understand the text a little difficult. A better alternative might be to use numbers to encode entire word instead of the letters within them.

II. TASKS AND TECNIQUES OF NLP

There are different NLP techniques that helps us to convert the human language into machine understandable language such as

- Stemming
- Lemmatization
- Tokenization
- Stop words removal.
- Word Sense Disambiguation
- Part of Speech Tagging

A. Stemming

Stemming is a process of reducing similar words to their stem word. E.g.: - History, Historical – Histori

B. Lemmatization

Lemmatization is the process of mapping words into their meaningful base structure. E.g.: - Reach, Reaching, Reached, Reaches: - Reach

C. Tokenization

Tokenization breaks a sentence into words and turn them into tokens.

D. Stop words Removal.

Removal of words from sentences which do not contain any valuable information is known as Stop words removal. E.g.: - a, the, is, are etc.

Word Sense Disambiguation: It is used to determine if the Same words can have different meanings in different sentences.

Part of Speech Tagging: Part of Speech (POS) tagging is well-known in NLP, which is used to label each word in a sentence or it can be a paragraph with its appropriate part of speech. Part of speech includes verbs, adverbs, adjectives, pronouns, etc.

III. PROPOSED IDEA

The main goal of this project is Conversion of Different programming languages.

In recent days, learning multiple programming languages

is really time consuming. If there were a system that could easily convert a general programming language into another one, then working in different fields would've been much easier.
Suppose a person only knows python and he need to work in Java for a certain project. Now this kind of compiler would make it easier. The person will enter his code in python and the compiler will turn it into Java.

Now this concept was proposed earlier but making it work is not that easy. As we know that different programming languages have their own syntax and executing process, so converting them is not only time consuming but also it needs a lot of skills.

IV. APPLICATION OF NLP IN PROGRAMMING LANGUAGE CONVERSION of NLP

Here we will try to change a block of code from one language to another by the use of NLP. For simplicity we are choosing two Object Oriented Programming languages, otherwise it would be difficult to change a Object Oriented Programming language to a procedural language and vice versa.so for this report we are choosing python and Kotlin. As of now we are approaching the problem as mentioned below:

Steps to be followed: -

1. Just like part of speech tagging we must come up with a method that can tag key words, variables, constants, different types of operators (conditional, logical, mathematical).

2. If we can tag and tokenize the block of code like this it would be a lot easier for the computer to understand.

3. Then we will feed this tokenized encoded input to our trained neural network model.
4. After getting the output from the NN we have to decode it to get our desired result.

5. The NN will only change the words that has been tagged as keywords or operators.

6. The variable and constants will remain as it is. while the key words, operators and syntax will be changed as necessary.

7. We have to follow sequence to sequence conversion so that the main structure of the code does not get changed.

A. *Experimental Setup and Result Analysis:*

Theoretical approach for solving this kind of problem: -

Let us consider one simple python code of adding two integers-

Input program file:-
```
 a=10
b=2 x=a+b if(x>=25):
print(x)

else:

print(x+10)
```

First, we will tag the variables and constants that is a, b, x ,10,2,25. these variables and constants value won't change after conversion.

Then we will tag the key words i.e. print, if, else. we will tokenize it and feed it to the NN which will give us the output as the version of this keywords in the desired language.

After this by using neural machine translation we have to correctly change the syntax of the code if necessary. And then after decoding we will get our desired output.
Expected Kotlin code: -

B. Keyword detection

Output: -
```
a is an identifier
= is an operator
10 is a constant
b is an identifier
2 is a constant
x is an identifier

+ is an operator
if is a keyword
bracket
>= is an operator
25 is a constant
bracket
: is used for indentation
print is a keyword
else is a keyword
```

C. *Challenges:*
Implementing this Program is not that easy as we don't know if it'll actually work practically or not. The challenges that we have faced while implementing the process are quite a few:

1. Removing ambiguity- Ambiguity is an intrinsic characteristic of human conversations. Ambiguity is one of the biggest challenges in NLP. When trying to understand the meaning of a word we consider several different aspects, such as the context in which it is used.
2. Improvement of the performance of individual analysers, specially at the semantic/pragmatic level.
3. Definition of new tasks, such as the detection of exclusivity, parallelism/concurrency, decision points, or iteration of tasks.
4. Detecting comment lines and text lines separately.

V. PROPOSED SOLUTION

Solving these problems at once is not a piece of cake. Therefore, our goal is to find different solutions for the problems that are rising throughout the implementation process.

1. segmenting text into meaningful groups.
2. identifying individual tokens within a sentence.
3. Word/phrase order variation.
4. The identification of problem-specific information and its transformation into structured form.
5. Determining relationships between entities or events.

VI. Conclusion and Future scope

Natural Language processing is one of the fastest growing technologies in todays world. It is like a blessing to the mankind as it makes every single task which required human involvement solvable in a very minimum time. But again every good thing has a dark side too so this natural language processing has also cut down on employment as it benefits the employer by doing the task in an accurate and faster manner. As its processing speed is ten times faster than human so an employer always prefer this technology. Innumerous number of research and development in this technology makes it a tremendously strong upcoming field which has a dire need of skilled professionals. With the exponential growth of multi-channel data like social or mobile data, businesses need solid technologies to assess and evaluate customer sentiments. So far, businesses have been happy analyzing customer actions, but in the current competitive climate, that type of customer analytics is outdated.

Natural language processing is an AI-complete problem. It is same as solving central artificial intelligence problem whose main aim is to make computers as intelligent as people so that they can think and solve problems like humans. They can perform all the activities that humans can perform at a much accurate and faster level and makes it more efficient than humans. Also it can perform tasks that humans cannot. Growth of Artificial intelligence basically predicts the growth of natural language processing in future. Through natural language computers or machines or devices understanding of human language will increase and they will be able to collect understand and the information online and apply what they learned in the real world. Combined with natural language generation, computers will become more and more capable of receiving and giving useful and resourceful information or data.

NLP can be used in areas where technology is not customer or human-facing like voice assistants and chatbots. NLP is one of the largest growing technologies in the field of data science. It can be also used to decipher meaning from unstructured data. NLP will find more and more applications in everyday technology as and when Ai and need for applications and humans to inter-communicate increases.

Till now we only have been able to detect the keywords, identifiers, constants, datatypes, indentation, operators etc. In future we aim to convert these tokenized strings into its respective code form to get the equivalent code in target programming language.

REFERENCES

[1] AI and Machine Learning for coders By Laurence Moroney

[2] https://www.qblocks.cloud/blog/natural-language-processing-machine-translation#:~:text=Natural%20Language%20Processing%20(NLP)%20is,the%20same%20way%20humans%20do.

[3] https://academic.oup.com/jamia/search-results?cqb=[{%22terms%22:[{%22filter%22:%22Keywords%22,%22input%22:%22Natural%20language%20processing%22}]}]&qb={%22Keywords1%22:%22Natural%20language%20processing%22}&page=1&searchType=advanced&adv=true

[4] https://dl.acm.org/doi/abs/10.5555/3340

[5] https://apps.dtic.mil/sti/pdfs/ADA567972.pdf

[6] https://arxiv.org/pdf/1904.04063

[7]https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=eeace1d14e266a5cd44fe781a874c662928602fd

[8] https://www.ibm.com/in-en/topics/natural-language-processing#:~:text=IBM%20Watson%20Discovery-,What%20is%20natural%20language%20processing%3F,same%20way%20human%20beings%20can.

[9] https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

[10]https://realpython.com/python-keywords/

[11]https://www.digitalocean.com/community/tutorials/python-keywords-identifiers

[12]https://www.w3schools.com/python/python_datatypes.asp

[13]https://www.geeksforgeeks.org/python-operators/

[14]https://www.nltk.org/

Smart Waste Bin Monitoring in Municipal Based on IOT for Clean City

[1]Sreya Poddar, [2]Tisa Dutta, [3]Sunando Chowdhury, [4]Soumyadeep Mukherjee, [5]Samprikta Mukherjee
{poddarsreya8, tisadutta78, 650sunando, soumyadeepmukherjee8236, mukherjeesamprikta0 }@gmail.com
BCA 3RD YEAR, IEM KOLKATA

[6]Prof. Manab Kumar Das manab.das@iem.edu.in
Asst. Prof, BCA & M.SC, IEM KOLKATA

Abstract — **Rapid increasing urbanization and increasing population all over the world, there is a dynamic increase in the amount of waste disposal has become a matter of concern, and diseases like malaria, dengue, and cholera are caused due to overflow of garbage which contains rotten things which form foul smell and burning things that cause air pollution to the environment. As a result of this we human beings are sufferers, So, we need to maintainthis worst scenario and we need to keep a track of the garbage bin so that it will be cleaned in the proper interval of time. So, we are implementing this project which willidentify the waste by checking its humidity and temperature and will also check the level of the garbage bin so that it doesn't overflow and pollute the environment. In this project the garbage level in each bin is monitored using ultrasonic sensors present in every bin, rotten and burning elements or any such abnormal situation that arises willbe identified by the Gas sensor and DHT11 sensor. The Gas, ultrasonic sensor, and Humidity sensor will read the data and will send it to the Cloud server and then the Municipal Control room will be able to monitor the information from the Cloud server through a GUI interface. When more than 70 percent of the garbage bin is filled, or any situation mentioned above arises thebuzzer will give the indication. The system is driven by a microcontroller- ESP-32 which is working as the brain of the operation, andit is programmed using Embedded C. All the devices and Cloud Server plays a key role toimplement the project.**

Keywords- Waste Bin, IoT Sensors anddevices, Cloud Server.

I. INTRODUCTION

One of the famous technologies in this world is the Internet of Things (IoT). The term IoT was first used by Ashton in 1999. IoT works with devices like bulbs, cameras, sensors, relays, and some internet-connected microcontrollers among other things. Various applications of IoT are smart health, smart city, environment threatening diseases that in turn harm the lives of a whole city and country as well. Our generation's main problems are the prevention, tracking, and treatment of these wastes . The current waste disposal schemes are not effective enough to dispose of the huge amount of waste generated from cities, leading to the spread of diseases, and prevention of harmful situations. So, we propose an alternative waste disposal strategy, consisting of a smart waste bin with three sensors for real- time monitoring of the garbage bin.

Different sensors are used for checking the temperature and humidity with level detection. In this innovative system, smart bins are installed in urban areas at different places that store garbage. The labour work, time, and cost will be less required than the traditional garbage collection system. Municipalities and corporations struggle to keep up with the outdoor bins to determine when to clean them or whether they are completely filled or not, that is why the level of the garbage or any harmful situation that arises in the bins is monitored continuously and is emptied timely.
The advantages of this technique are as follows:
i) The above dustbin also sends a mobile notification when the dustbin is almost filled.
ii) In this process, various electronic components are used to make this dustbin smart.
As per the report published by World Bank, approximately 1.3 billion tons of municipal waste is generated every year and it is expected to rise to approximately 2.2 billion tons per year by 2025. Due to a lack of proper cleaning of waste, a large amount of untreated waste is dumped into landfills. Implementing our project at the regional level will reduce the expenditure on waste disposal, and make people aware that how much cleaning is important for us and for the environment too.

II. LITERATURE SURVEY

Aniqa Bano,1 Ikram Ud Din, and Asma A. Al-Huqail. Presented "A IoT-Based Smart Bin for

Real-Time Monitoring and Management of Solid Waste" in which they aim to keep the environment green and clean, monitoring and disposing of the waste is very important these days. Improper disposal and poor monitoring of collected waste and waste bins can cause serious damage to human lives. Therefore, a waste management mechanism is proposed for smart cities, named SBM (smart bin mechanism) in order to sanitize and clean the environment intelligently.[1] V R Ravi1*, M Hema2 , S SreePrashanthini3 and V Sruthi4. Designed an IoT-integrated smart bin for Smarter Waste Disposal System is devised. In the proposed work an alternative efficient and economical

waste disposal strategy is developed. A newer waste bin is designed in the proposed work and is attached with four sensors for effective real-time monitoring of the smart bin conditions. Whenever the garbage level in the smart bin reaches a programmed threshold level, an alert message is sent to the cleaning authority to empty the smart bin. Thus, the proposed waste disposal scheme using smart bins can effectively assist as a benchmark for waste disposal schemes used in smart cities [2] S. Vishnu , S. R. Jino Ramson* , Samson Senith, Theodoros Anagnostopoulos, Adnan M. Abu-Mahfouz , Xiaozhe Fan, S. Srinivasan and A. Alfred Kirubaraj proposed "IoT-Enabled Solid Waste Management in Smart Cities" in which they say as most homes are equipped with a wireless internet connection, it is inferred that the Wi-Fi-based solution is well suited for monitoring the household bins. This will minimize the additional infrastructure expense. Therefore, this work proposes an IoT-based solid waste management system for smart cities. The main contributions of this work in contrast to the existing solutions are as follows –

i) Hybrid Network Architecture to monitor the household and public trash bins.
ii) Solar energy harvesting facility to extend the lifetime of the end nodes.
iii) A GPS module is embedded to evaluate the Geo-location of the trash bins.
iv) An intelligent GUI is employed to view the status of every trash bin [3]. Tariq Ali Muhammad Irfan1, Abdullah Saeed Alwadie1 & Adam Glowacz "IoT-Based Smart Waste Bin Monitoring and Municipal Solid Waste Management System for Smart Cities" where Numerous IoT-based smart technologies have been developed to deal with different issues associated with trash management systems in smart cities. From the literature, it is recognized that the most significant issue is solid waste management for the smart city. Scholars have used a variety of strategies and procedures to overcome these challenges. In this system, sensors sense the level of waste in the bins and send alerts to the controller. A microcontroller encodes these alerts and forwards them to the main central processing unit [4]. B.Balaji Naik, T.Sai Kiran, B.K.N.Harish, J.Hermes Sujit ,D.Sai & Kiran written "IOT Based Waste Monitoring System for Smart Cities." where they describe The Internet of

Things (IoT) technology is transforming society in a variety of fields, including healthcare, industrial automation, transportation, and smart cities, in the age of interconnected devices. In this study, we present an internet of things (IoT)-based smart waste monitoring system that enables waste management authorities to continually monitor the status of trash cans located at various places and, as per the status, take suitable actions to collect it quickly and effectively.[5]Himadri Nath Saha, Supratim Auddy, Subrata Pal, Shubham Kumar, Shivesh Pandey, Rakhee Singh, Amrendra Kumar Singh, Swarnadeep Banerjee, Debmalya Ghosh, Sanhita Saha, "Waste Management using Internet of Things (IoT)", IEEE 2017.in which The trash can is battery- or solar-powered and functions as a Wi-Fi hotspot. It measures the volume of waste present inside the compartment and wirelessly sends information about the fill level to a cloud server. Time is used more effectively as a result, and the roads are cleaner. So, in this paper, we have proposed a system that can be deployed in general-purpose dust bins placed in public places. This system allows us to monitor its status remotely over the internet for efficient waste management.[6] A Vanitha proposed that the present day it has been seen that the dustbin is overflown with garbage, so the proposed system will help to avoid the overflow of dustbin. This system will give the real time information about the status of the dustbin [7]. S Vinod Kumar proposed that, with the help of Ultrasonic sensor, the level of waste in the dustbins is detected. To measure the weight of the dust bin force sensor is used [8].

III. METHODOLOGY

A. Working Algorithm of Smart Waste Bin:

STEP 1: Start (Initialize the process).

STEP 2: Gather all the information on the waste by reading the sensors.

STEP 3: Temperature, the humidity will check through the DHT-11 sensor and level detection will check through the ultrasonic sensor.

STEP 4: Now it will check whether the bin is full or not.

STEP 5: If it is above the limit, then a message will be sent to the cloud server.

STEP 6: After receiving the message, a garbage vehicle will be sent for the waste collection.
STEP 7: After collection, it will show the current status of the dustbin.

STEP 8: Otherwise, the scanning process of the garbage will repeat in a loop.

STEP 9: Stop (Process Terminated).

B. Flowchart:

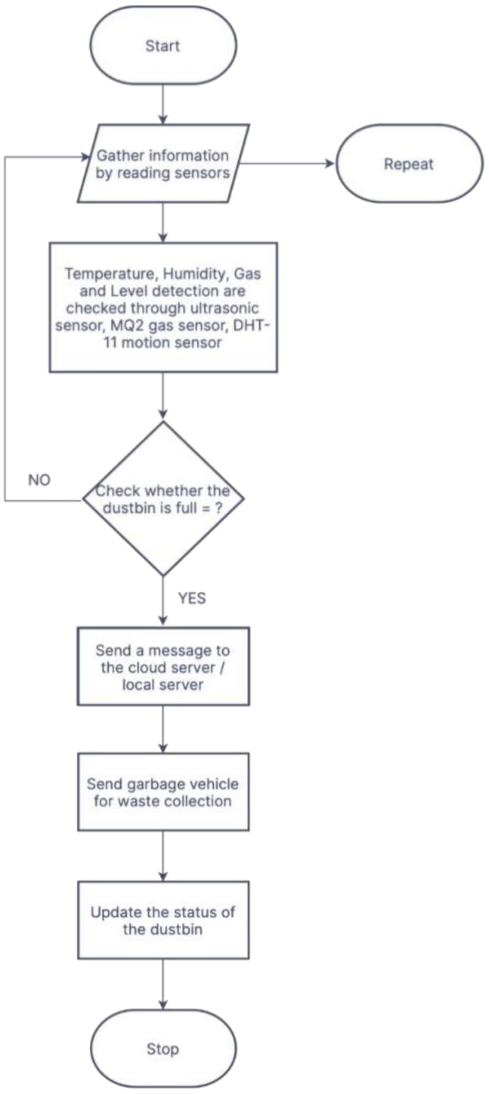

Start
Gather information by reading sensors
Repeat
Temperature, Humidity, Gas and Level detection are checked through ultrasonic sensor, MQ2 gas sensor, DHT-11 motion sensor
NO
Check whether the dustbin is full = ?
YES
Send a message to the cloud server / local server
Send garbage vehicle for waste collection
Update the status of the dustbin
Stop

Fig. 1: Process flow diagram of proposed methodology

C. Proposed Model

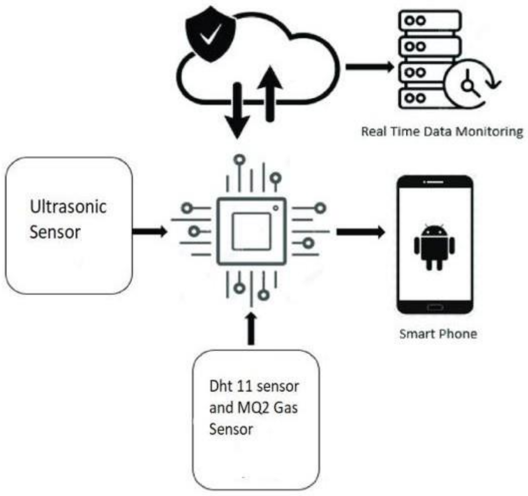


Fig. 2: Block Diagram of Proposed Model

The Smart Waste Bin system is driven by the Microcontroller ESP 32. All the components that are connected to ESP 32 are programmed in C++

language and it reads the input/output pins of the components. The temperature and humidity are monitored by using the DHT11 sensor and it will be displayed on the dashboard. Whenever the DHT11 sensor detects unusual temperatures in the bin which can hamper the system, a notification will be sent to the clearing authorities to remove it. The measure of the dustbin level is calculated by the Ultrasonic sensor connected at the edge of the dustbin. When the dustbin is full, the message- "BIN IS FULL" is sent to the cleaning authorities. The message is sent using the WI-FI that provides communication between the bin and the authority. The sensor sends the data to the Microcontroller which is connected to the Cloud and then it will be displayed on the dashboard. As the location of the bins is already mentioned in the code, when the bins are overloaded, they will be displayed on the dashboard with their location.

D. Components Requirement:

(a) Software Requirement:

1. **Arduino IDE:** The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

2. **Arduino IOT Cloud:** Arduino IoT Cloud is an application that helps makers build connected objects quickly, easily, and securely. You can connect multiple devices and allow them to exchange real-time data. You can also monitor themfrom anywhere using a simple user interface.

(b) Hardware Requirement:

ESP-32: ESP-32 is a series of low-cost, low-power systems on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 seriesemploys either a Ten silica Extensa LX6 microprocessor in both dual-core and single-core variations, an Extensa LX7 dual-core microprocessor, or a single RISC V-core. RISC- microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Express if Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

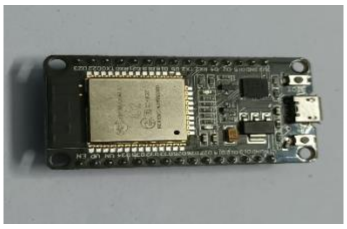


Fig. 3(a): Microcontroller ESP-32

Power Supply: A power supply is an electrical device that supplies electricpower to an electrical load. The main purpose of a power supply is to convert electric current from a source to the correct voltage, current, and frequency to power the load. As a result, power supplies are sometimes referred to as electric power converters. Some power supplies are separate standalone pieces of equipment, while others are built into the load appliances that they power. Examples of the latter include power supplies found in desktop computers and consumer electronics devices. Other functions that power supplies may perform include limiting the current drawn by the load to safe levels, shutting off the current in the event of an electrical fault, power conditioning to prevent electronic noise orvoltage surges on the input from reaching the load, power-factor correction, and storing energy so it can continue to power the load in the event of a temporary interruption in the source power (uninterruptible power supply).

DHT-11 Sensor: The digital temperature and humidity sensor DHT11 is a compositesensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature andhumidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resistive sense of wet components and an NTC temperature measurement device and is connected to a high-performance 8-bit microcontroller.

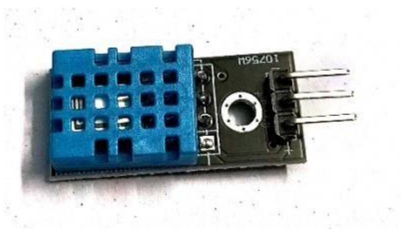
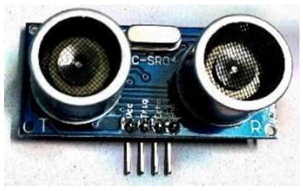


Fig. 3(b): DHT-11 Sensor

4. **Ultrasonic Sensor:** Ultrasonic transducers and ultrasonic sensors are devices that generate or sense ultrasound energy. They can be divided into three broad categories: transmitters, receivers, and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.



5. **Breadboard:** A breadboard, solderless breadboard, or protoboard is a construction base used to build semi-permanent prototypes of electronic circuits. Unlike a perf board or stripboard, breadboards do not require soldering or destruction of tracks and are hence reusable. For this reason, breadboards are also popular with students and in technological education.

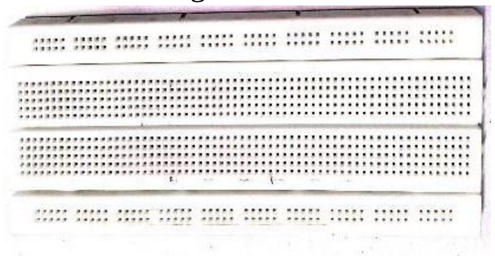


Fig. 3(d): Breadboard

6. **Jumper Wires:** A jump wire is an electricalwire, or group of them in a cable, with a connector or pins at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

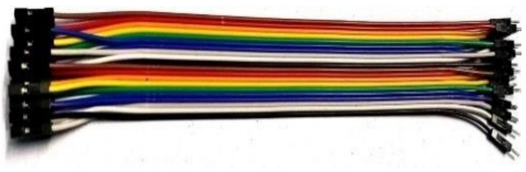


Fig. 3(e): Jumper Wires

7. **Gas Sensor:** Gas sensors are devices that candetect the presence and concentration of various hazardous gases and vapors, such as toxic or explosive gases, volatile organic compounds (VOCs), humidity, and odours.

Fig. 3(f): Gas Sensor

8. **Buzzer:** An audio signalling's device like a beeper or buzzer may be electromechanics or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarms, music, bell & siren.

Fig. 3(g): Buzzer

IV. RESULT & DISCUSSIONS

As we do not have any proper waste management system to date, the hazardous impact of this unrestrained condition is continuously affecting the environment. So, we need to take the initiative to maintain the deteriorating condition we live in to make our city smart and clean. In order to make this happen we have used IoT technology, by using which we have made this Smart Waste Bin. Here we have used the Microcontroller ESP-32, DHT-11 Sensor, Gas Sensor, and other components mentioned above.

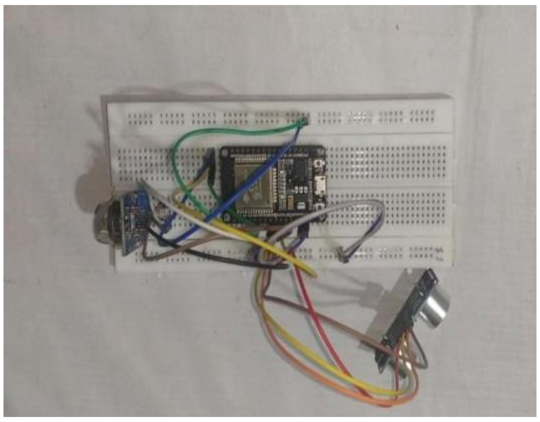


Fig. 4: Circuit Design of proposed model

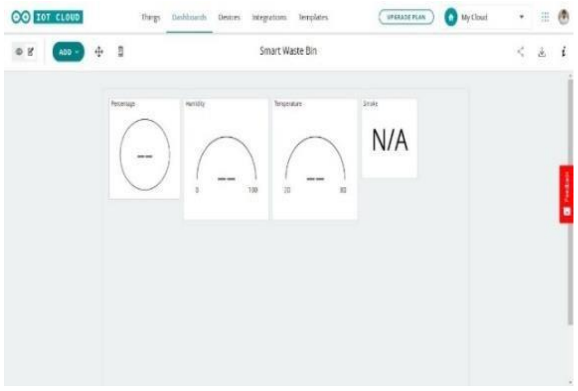


Fig. 5: Dashboard Image to display real timedata.

V. CONCLUSION

Due to the rise in urbanization, waste is increasing very fast. Therefore, waste management is an important need to protect the environment. Any discarded object that hasbeen passed to a party, a crowded room, a social structure, a school, or an apartment is considered to be waste This method of implementation saves time in level detection by humans and affordability in domestic applications. This Smart Dustbin prototype will contribute a lot to society to provide a clean and hygienic environment. The project focuses on "IoT technology" and how it can be used in "Smart City applications" (IoT). The initiative will aim to minimize the use of trashcans in the future. The main purpose is to cleanthe dustbin and better clean the environment.

By continuously using this system to find the maximum height of rubbish in a dustbin that isplaced in it. If a dustbin is nearly 70 percent, a mail notification can be sent immediately. Thiswould decrease the stray trash on the streets.

REFERENCES

[1] Aniqa Bano, Ikram Ud Din and Asma A. Al-Huqail "AIoT-Based Smart Bin for Real-Time Monitoring and Management of Solid Waste" Hindawi Scientific Programming Volume 2020, Article ID 6613263, 13 pages

[2] V R Ravi, M Hema, S SreePrashanthini and V Sruthi " Smart bins for garbage monitoring in smart cities using IoT system" Published under licenceby IOP Publishing Ltd IOP Conference Series: Materials Science and Engineering, Volume 1055, International Virtual Conference on Robotics, Automation, Intelligent Systems and Energy (IVC RAISE 2020) 15th December 2020, Erode, India.

[3] S. Vishnu, S. R. Jino Ramson , Samson Senith, Theodoros Anagnostopoulos, Adnan M. Abu-Mahfouz,Xiaozhe Fan, S. Srinivasan and A. Alfred Kirubaraj "IoT-Enabled Solid Waste Management in SmartCities"SmartCities2021,4,10041017.10.3390/sm artcities4030053, Received: 25 May 2021
Accepted: 5 July 2021
Published: 14 July 2021

[4]] Integrated Sensing Systems and Algorithms for SolidWaste Bin StateManagement Automation Md. Abdulla Al Mamun, Mahammad A. Hannan, Member, IEEE, Aini Hussain, Member, IEEE, and Hassan Basri VOL. 15,NO. 1, JANUARY 2015

[5] P. H. Brunner and J. Fellner, “Setting priorities for waste management strategies in developing countries,” Waste Manage. Res., vol. 25, no. 3, pp.234–240, Jun. 2007

[6] Himadri Nath Saha, Supratim Auddy, Subrata Pal, Shubham Kumar, Shivesh Pandey, Rakhee Singh, Amrendra Kumar Singh, Swarnadeep Banerjee, Debmalya Ghosh, Sanhita Saha, “Waste Management using Internet of Things (IoT)”, IEEE 2017.

[7] A.Vanitha,.PadmaPriya ,S Maheshwari,"Waste Management System Using Iot MrAnuradha",May2018

DOI:10.22214/ijraset.2018.5477.

[8] S. Vinoth Kumar1, T Senthil Kumaran2, A Krishna Kumar and Mahantesh Mathapati4,"SmartGarbage Monitoring and Clearance System using Internet of Things", August 2017 DOI:10.1109/ICSTM.2017.8089148 Conference: 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy andMaterials (ICSTM).