A Project Stage-I Report On

PESTICIDE DETECTION IN FRUITS AND VEGETBLES

Submitted in partial fulfillment for award of the degree of Bachelor of Technology

in

Electrical and Electronics Engineering Department

by

M.Tejasri	**22321A0231**
R.Veena	**23325A0221**
V.Pranitha	**22321A0214**
Sai Abhignya.J	**22321A0220**

Under the esteemed Guidance of

Dr. S. Asha Kiranmai , ME., Ph.D.(OU)

Associate Professor, EEE Department

Bhoj Reddy Engineering College for Women

Department of Electrical and Electronics Engineering

(Sponsored by Sangam Laxmibai Vidyapeet, Accredited by NAAC with A Grade, Approved by AICTE and Affiliated to JNTUH, Recognized by UGC under section 2(f) of the UGC Act, 1956)

Vinaynagar, IS Sadan Crossroads, Hyderabad – 500 059

Ph: +91-40-2453 7282; Website: www.brecw.ac.in; Email: principal@brecw.ac.in

2022 – 2026

Bhoj Reddy Engineering College for Women
(Sponsored by Sangam Laxmibai Vidyapeet, Accredited by NAAC with A Grade, Approved by AICTE
Affiliated to JNTUH, Recognized by UGC under section 2(f) of the UGC Act, 1956)
Vinaynagar, I S Sadan Crossroads, Hyderabad 500 059, Telangana
Telephone: 040-2453 7282, Website: www.brecw.ac.in, Email: principal@brecw.ac.in

Date:

CERTIFICATE

This is to certify that the Project Stage-I Report entitled **"Pesticide detection in fruits and vegetables "** is a bonafide work carried out by

M.Tejasri	**22321A0231**
R.Veena	**23325A0221**
V.Pranitha	**22321A0214**
Sai Abhignya.J	**22321A0220**

in partial fulfillment for award of the degree of Bachelor of Technology in Electrical and Electronics Engineering department from Bhoj Reddy Engineering College for Women, Hyderabad, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH).

Internal Guide

Dr.S.Asha Kiranmai, M.E., Ph.D. (OU)

Senior Faculty

R. Manju Bhargavi, M.Tech.,(PID)

Project Supervisor
S. Deepti, M.E.,(PS)

Head Of Department
Dr.S.Asha kiranmai, M.E.,Ph.D.(OU)

DECLARATION

We hereby declare that the work presented in this project **"Pesticide Detection in Fruits and Vegetables "** submitted towards completion of Industrial Oriented Major Project in IV year of B.Tech. EEE at '**Bhoj Reddy Engineering College forWomen'**, Hyderabad, is an authentic record of our original work carried out under the guidance of Dr. S. Asha Kiranmai, M.E., Ph.D. (OU), Associate Professor, HOD-EEE, BRECW.

Sign. with date:

M.Tejasri (22321A0231)

Sign. with date:

R.Veena (2325A0221)

Sign. with date:

V. Pranitha (22321A0214)

Sign. With date

Sai Abhignya.J (22321A0220)

ACKNOWLEDGMENTS

The satisfaction and euphoria that accompanies the completion of task would be incomplete without the mention of the people whose constant guidance and encouragement have crowned our efforts in success.

It gives us an immense pleasure to express deep gratitude and respect to our internal guide**, Dr. S. Asha Kiranmai, M.E., Ph.D. (OU),** Associate Professor, Department of EEE, BRECW,for the eminent guidance and suggestions she has provided for successful completion of the Project Stage-I.

We thank **Dr. S. Asha Kiranmai, M.E., Ph.D. (OU)** Head of the Department, EEE, BRECW, forgiving timely cooperation and taking necessary action throughout the course of our project.

We also thank **Dr. G. Shyama Chandra Prasad** Principal, BRECW, for providing us an opportunity to carry out the project.

Last but not the least; we would like to acknowledge all those, whose names cannot be penned here, but who were directly or indirectly are a part of this project.

M. Tejasri(23325A0202)
R. Veena (23325A0219)
V. Pranitha (23325A0206)
Sai Abhignya.J (22321A0222)

CONTENTS

DESCRIPTION **PAGE NO.**

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

LCD	:	Liquid Crystal Display
DHT11 Sensor	:	Digital Humidity and Temperature
LED	:	Light Emitting Diode
P&O	:	Perturb and Observe
SOC	:	State Of Charge
LDR	:	Light Dependent Resistor
INC	:	Incremental Conductance
CV&CC	:	Constant Voltage and Constant Current
PWM	:	Pulse Width Modulation
LCD	:	Liquid Crystal Display
Ah	:	Ampere-hours
PMDC	:	Permanent Magnetic DC Motor

ABSTRACT

Many modern techniques were developed to produce more quantity of food for the growing population. Now a day, fruits and vegetables have become the major source of nutrients and energy. However, many chemicals are used in the production of fruits and vegetables, which are dangerous for consumers.

To identify pesticides in organic fruits and vegetables, it is necessary to build a low-cost, portable, sensitive, and selective biosensing platform.The proposed smart system for organic fruit detection integrates a pH sensor, conductivity sensor, temperature sensor, and humidity sensor with an Arduino Uno controller and ESP Wi-Fi module for real-time monitoring and data transmission. The system is designed to distinguish between organic and non-organic fruits by analyzing key environmental and chemical parameters. The Arduino Uno processes sensor data and transmits it via the ESP Wi-Fi module to a cloud platform or mobile application for further analysis and display.

This system offers a cost-effective, portable, and user-friendly solution for consumers and vendors to assess fruit authenticity, thereby promoting healthier food choices and reducing exposure to harmful substances. The main aim of the project is to calculate the NDVI (Normalized Difference Vegetation Index) using IR sensors, with the software written in Embedded C. Safe pesticide values that can be consumed by humans and animals are pre-programmed in the code, and if a fruit is detected to fall above the threshold level, it is considered contaminated. Through IoT, pesticide content and sensor values are displayed in a mobile application.

If a fruit or vegetable exceeds the threshold value, the sample is flagged as contaminated. In Project Stage 1, the system will detect pesticides in fruits using the proposed sensors and IoT platform, while in Project Stage 2, the same approach will be extended to vegetables, ensuring comprehensive monitoring of both fruits and vegetables.

INTRODUCTION

The demand for organic fruits has increased significantly in recent years due to growing awareness of health benefits and concerns over the harmful effects of chemical pesticides and fertilizers. Organic fruits are cultivated without synthetic inputs, making them healthier and more environmentally sustainable compared to conventionally grown produce. However, differentiating between organic and non-organic fruits is often difficult for consumers and vendors, as visual inspection alone cannot accurately determine the presence of artificial chemicals. This challenge highlights the need for a reliable, affordable, and portable system capable of identifying the authenticity of organic fruits in real-time.

Traditional laboratory-based chemical testing methods for detecting pesticides and artificial residues are accurate but expensive, time-consuming, and require professional expertise. These limitations make them impractical for regular use in local markets, retail stores, and by individual consumers. To address these challenges, integrating sensor-based technology with embedded systems offers a practical alternative. Such systems can provide on-the-spot testing, making fruit quality verification more accessible and efficient.

In this project, a smart organic fruit detection system is developed using multiple sensors, including a pH sensor, conductivity sensor, and temperature and humidity sensors, all controlled by Pico microcontroller. The system utilizes these sensors to capture key chemical and environmental parameters from the fruit's surface. Since organic and non-organic fruits show different pH and conductivity levels due to varying chemical treatments, analyzing these parameters provides an effective way to distinguish between them.
The collected data from the sensors is processed by the pico , which serves as the core processing unit of the system. The processed information is then transmitted through an ESP Wi-Fi module to a cloud platform or mobile application for real-time monitoring and visualization. This connectivity allows users to access and interpret fruit quality data remotely, enabling informed decisions about food safety and authenticity.

Overall, the proposed system represents an innovative, cost-effective, and user-friendly approach to fruit authentication. By combining sensor technology, embedded processing, and IoT-based communication, the system contributes to promoting healthier consumption habits.

1.1 LITERATURE SURVEY

Y. Yuan et al. (2024)Yuan and coauthors applied deep-learning feature extraction (convolutional neural networks) to images and fused these deep features with sensor-derived data (e.g., VOC/gas sensors, temperature) to estimate freshness and classify spoilage stages. Their experiments show that combining visual deep features with simple physicochemical inputs materially improves robustness across lighting and cultivar variation. This hybrid approach is particularly relevant if you envision a cloud-based model where an ESP/Arduino unit provides chemical features (pH, EC, temp/humidity) while a smartphone supplies images for richer features.

IJERT/2024 (project paper)Their workflow standardizes a wash/extraction method, collects multisensor data, performs feature engineering (ratios, normalized EC, pH adjusted by temperature), and trains classifiers (SVM, random forest). Results show promising classification on a small dataset, with ML often outperforming single-threshold rules; however, the paper carefully notes that ML models can overfit to collection sites and stresses the need for diverse, labeled datasets (certified organic vs confirmed conventional). The authors provide a practical stack for Arduino/ESP deployments: on-device preprocessing (rolling averages, outlier rejection), secure MQTT upload, and cloud-side model scoring with user-facing results. They also highlight ethical and regulatory considerations — claims about “organic” status should be probabilistic and presented with confidence measures, and flagged produce should be sent for confirmatory lab testing before legal action. This paper is valuable as it demonstrates an end-to-end prototype and provides implementation tips for integrating pH/EC sensors into an IoT+ML pipeline while pointing out the limitations and necessary validation steps for real-world adoption.

1.2 OBJECTIVE

This project aims to The main objective of this project is to design and develop a Smart Pesticide Detection System for fruits and vegetables using multiple sensors such as pH, conductivity, temperature, humidity, and gas sensors, integrated with an Arduino-based IoT platform.This project provides a portable, low-cost, and user-friendly solution that helps consumers and vendors assess the quality and safety of fruits and vegetables, thereby promoting healthier consumption habits and reducing exposure to harmful chemical contaminants.

1.3 ORGANIZATION OF THE REPORT

Chapter 1 This chapter introduces the organic fruit detection system . This project presents a literature survey and the objective of the project.

Chapter 2 This chapter thoroughly explains about the embedded systems and its applications.

Chapter 3 Presents the hardware description. It deals with the block diagram of the project and explains the purpose of each block.

Chapter 4 This chapter explains the software implementation for detection of pesticides using embedded c and how the code interfaces with the sensors to detect the pesticide levels and how the data is sent to the web page.

Chapter 5 In this chapter, the block diagram of the project is explained.

Chapter 6 This chapter offers the hardware module and testing methodologies employed in the project are described and the obtained results are presented .

Chapter 7 This chapter offers a summary of the project's key findings and outcomes, drawing conclusions based on the results. It also outlines potential areas for future development and expansion of the project.

`

2. EMBEDDED SYSTEMS

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites. (Each radar probably includes one or more embedded systems of its own.)

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which don't expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded". A modern example of embedded system is shown in fig: 2.1.

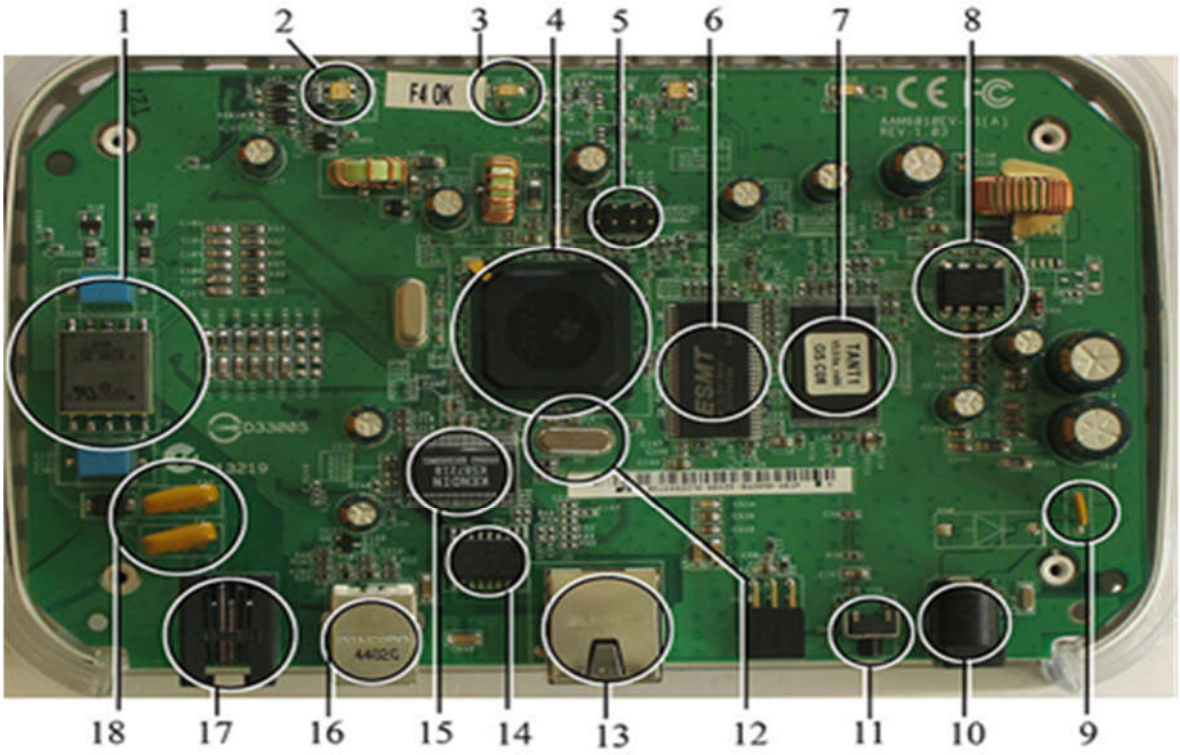


Fig. 2.1 A modern example of embedded system

Labeled parts include microprocessor (4), RAM (6), flash memory (7). Embeded systems programming is not like normal PC programming. In many ways, programming for an embedded system is like programming PC 15 years ago. The hardware for the system is usually chosen to make the device as cheap as possible. Spending an extra dollar a unit in order to make things easier to program can cost millions. Hiring a programmer for an extra month is cheap in comparison. This means the programmer must make do with slow processors and low memory, while at the same time battling a need for efficiency not seen in most PC applications. Below is a list of issues specific to the embedded field.

2.1.1 History

In the earliest years of computers in the 1930–40s, computers were sometimes dedicated to a single task, but were far too large and expensive for most kinds of tasks performed by embedded computers of today. Over time however, the concept of programmable controllers evolved from traditional electromechanical sequencers, via solid state devices, to the use of computer technology.

One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in

the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961. It was built from transistor logic and had a hard disk for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits.

2.1.2 Tools

Embedded development makes up a small fraction of total programming. There's also a large number of embedded architectures, unlike the PC world where 1 instruction set rules, and the Unix world where there's only 3 or 4 major ones. This means that the tools are more expensive. It also means that they're lowering featured, and less developed. On a major embedded project, at some point you will almost always find a compiler bug of some sort.

Debugging tools are another issue. Since you can't always run general programs on your embedded processor, you can't always run a debugger on it. This makes fixing your program difficult. Special hardware such as JTAG ports can overcome this issue in part. However, if you stop on a breakpoint when your system is controlling real world hardware (such as a motor), permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

2.1.3 Resources

To save costs, embedded systems frequently have the cheapest processors that can do the job. This means your programs need to be written as efficiently as possible. When dealing with large data sets, issues like memory cache misses that never matter in PC programming can hurt you. Luckily, this won't happen too often- use reasonably efficient algorithms to start, and optimize only when necessary. Of course, normal profilers won't work well, due to the same reason debuggers don't work well.

Memory is also an issue. For the same cost savings reasons, embedded systems usually have the least memory they can get away with. That means their algorithms must be memory efficient (unlike in PC programs, you will frequently sacrifice processor time for memory, rather than the reverse). It also means you can't afford to leak memory. Embedded applications generally use deterministic memory techniques and avoid the default "new" and "malloc" functions, so that leaks can be found and eliminated more easily. Other resources programmers expect may not even exist. For example, most embedded processors do not have hardware FPUs (Floating-Point Processing Unit).

2.1.4 Real Time Issues

The Embedded systems frequently control hardware, and must be able to respond to them in real time. Failure to do so could cause inaccuracy in measurements, or even damage hardware such as motors. This is made even more difficult by the lack of resources available. Almost all embedded systems need to be able to prioritize some tasks over others, and to be able to put off/skip low priority tasks such as UI in favor of high priority tasks like hardware control.

2.2 Need for Embedded Systems

The uses of embedded systems are virtually limitless, because every day new products are introduced to the market that utilizes embedded computers in novel ways. In recent years, hardware such as microprocessors, microcontrollers, and FPGA chips has become much cheaper. So, when implementing a new form of control, it's wiser to just buy the generic chip and write your own custom software for it. Producing a custom-made chip to handle a particular task or set of tasks costs far more time and money. Many embedded computers even come with extensive libraries, so that "writing your own software" becomes a very trivial task indeed. From an implementation viewpoint, there is a major difference between a computer and an embedded system. Embedded systems are often required to provide Real-Time response. The main elements that make embedded systems unique are its reliability and ease in debugging.

2.2.1 Debugging

The Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most Sophisticate they can be roughly grouped into the following areas:

- Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic).
- External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multi core systems.
- An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
- An in-circuit emulator replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
- A complete emulator provides a simulation of all aspects of the hardware, allowing all of it

to be controlled and modified and allowing debugging on a normal PC.

- Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as assembly code or source-code.

Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software (and microprocessor) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor). An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance.

2.2.2 Reliability

The Embedded systems often reside in machines that are expected to run continuously for years without errors and in some cases recover by them if an error occurs. Therefore, the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

- The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
- The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals, engines on single-engine aircraft.
- The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and also soft errors in the hardware:

- Watchdog timer that resets the computer unless the software periodically notifies the watchdog.

- Subsystems with redundant spares that can be switched over.
- software "limp modes" that provide partial function.
- Designing with a Trusted Computing Base (TCB) Architecture [6] ensures a highly secure & reliable system environment
- An Embedded Hypervisor is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.

  Immunity Aware Programming.

2.3 Explanation of Embedded Systems

2.3.1 Software Architecture

The There are several different types of software architecture in common use.

- **Simple Control Loop**

  In this design, the software simply has a loop. The loop calls subroutines,
  each of which manages a part of the hardware or software.

- **Interrupt Controlled System**

  Some embedded systems are predominantly interrupt controlled .

  Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays. Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

- **Cooperative Multitasking**

  A non-preemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for no operation), etc. The advantages and disadvantages are very similar to the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue-interpreter.

- **Primitive Multitasking**

  In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally

considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non-blocking synchronization scheme.

Because of these complexities, it is common for organizations to buy a real-time operating system, allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a generic real time system, due to limitations regarding memory size, performance, and/or battery life.

Microkernels and Exokernels

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast, and fail when they are slow. Exokernels communicate efficiently by normal subroutine calls. The hardware, and all the software in the system are available to, and extensible by application programmers. Based on performance, functionality, requirement the embedded systems are divided into three categories:

2.3.2 Stand Alone Embedded System

In These systems takes the input in the form of electrical signals from transducers or commands from human beings such as pressing of a button etc.., process them and produces desired output. This entire process of taking input, processing it and giving output is done in standalone mode. Such embedded systems come under standalone embedded systems

Eg: microwave oven, air conditioner etc.

2.3.3 Real-time embedded systems

Embedded systems which are used to perform a specific task or operation in a specific time period those systems are called as real-time embedded systems. There are two types of real-time embedded systems.

- **Hard Real-time embedded systems**

These embedded systems follow an absolute dead line time period i.e.., if the tasking is not

done in a particular time-period then there is a cause of damage to the entire equipment.

Eg: consider a system in which we have to open a valve within 30 milliseconds. If this valve is not opened in 30ms this may cause damage to the entire equipment. So in such cases we use embedded systems for doing automatic operations.

- **Soft Real Time embedded systems**

  These embedded systems follow a relative dead line time period i.e.., if the task is not done in a particular time that will not cause damage to the equipment.

  Eg: Consider a TV remote control system, if the remote control takes a few milliseconds delay it will not cause damage either to the TV or to the remote control. These systems which will not cause damage when they are not operated at considerable time period those systems come under soft real-time embedded systems.

2.3.4 Network communication embedded systems

A wide range network interfacing communication is provided by using embedded systems.

- Consider a web camera that is connected to the computer with the internet can be used to spread communication like sending pictures, images, videos etc.., to another computer with internet connection throughout anywhere in the world.
- Consider a web camera that is connected at the door lock.

  Whenever a person comes near the door, it captures the image of a person and sends to the desktop of your computer which is connected to internet. This gives an alerting message with image on to the desktop of your computer, and then you can open the door lock just by clicking the mouse. Fig: 2.2 show the network communications in embedded systems.

Fig. 2.2 Network communication embedded systems

The central processing unit (CPU) can be any one of the following Microprocessor, microcontroller, digital signal processing.

- Among these Microcontroller is of low-cost processor and one of the main advantage of microcontrollers is, the components such as memory, serial communication interfaces, analog to digital converters etc.., all these are built on a single chip. The numbers of external components that are connected to it are very less according to the application.
- Microprocessors are more powerful than microcontrollers. They are used in major applications with a number of tasking requirements. But the microprocessor requires many external components like memory, serial communication, hard disk, input output ports etc.., so the power consumption is also very high when compared to microcontrollers.
- Digital signal processing is used mainly for the applications that particularly involved with processing of signals.

2.4 APPLICATIONS OF EMBEDDED SYSTEMS

Consumer applications

At home we use a number of embedded systems which include microwave oven, remote control, VCD players, DVD players, camera etc.

Industrial automation

Today a lot of industries are using embedded systems for process control. In industries we design the embedded systems to perform a specific operation like monitoring temperature, pressure, humidity, voltage, current etc.., and basing on these monitored levels we do control other devices, we can send information to a centralized monitoring station.

3. HARDWARE COMPONENTS

This chapter describes the hardware components that constitute the detection of presticide in fruit and vegetables. It explains the purpose and functionality of each part, such as the microcontroller,conductivity sensor , ph sensor,gas sensor, power supply , and DTH sendor. The selection of components is based on cost-effectiveness, efficiency, and ease of integration for real-time applications. The detailed explanation in this chapter helps in understanding how these components collectively contribute to the proper working of the system.

The Hardware Components are:

1. Ardino Uno.
2. PH sensor.
3. Conductivity sensor.
4. MQ3 gas sensor .
5. DTH11 sensor .
6. L293D motor drive.
7. LED module.
8. Buzzer module.
9. RPS
10. Connecting Wires

3.1 ARDUINO UNO:

The Arduino Uno **i**s one of the most popular and widely used microcontroller boards in the field of electronics, robotics, and embedded systems. It is an open-source platform designed to make it easy for beginners and professionals to create interactive electronic projects. provides a simple and affordable way to design, prototype, and experiment with digital and analog systems and tha Arduino Uno is shown on below figure3.1

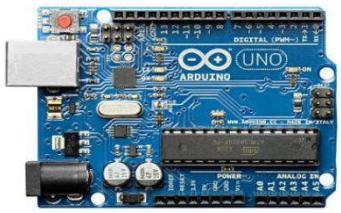


Fig.3.1 Ardino Uno

The Arduino Uno operates at 5 volts and runs at a clock speed of 16 MHz**.** It has 14 digital input/output (I/O) pins, of which 6 can be used as PWM (Pulse Width Modulation) outputs, and 6 analog input pins for reading analog sensors. The board also includes a USB connection for programming and serial communication, a power jack**,** a reset button, and ICSP (In-Circuit Serial Programming) headers for direct programming of the microcontroller.

One of the key reasons for the Arduino Uno's popularity is its ease of use**.** It can be programmed using the Arduino Integrated Development Environment (IDE)**,** which supports a simplified version of the C/C++ programming language. The IDE allows users to write code, compile it, and upload it to the board through a USB cable. The simplicity of the coding environment makes it ideal for students, hobbyists, and educators who want to learn about microcontrollers and electronics.

The Uno can be powered either through the USB connection or an external power supply, typically between 7 to 12 volts. It features a voltage regulator that ensures the microcontroller and connected components receive a stable 5V supply. The board's open-source nature means that its schematics, design files, and firmware are freely available, allowing users to modify or build their own versions.

Another major advantage of the Arduino Uno is its versatility and compatibility with a wide range of shields and modules, such as motor drivers, Wi-Fi and Bluetooth modules, sensors, and display units.

Specifications:

- Microcontroller: ATmega328P
- Operating Voltage:5V
- Input Voltage (recommended):7-12
- Digital /O Pins: 14 (of which 6 provide PWM Output)
- Analog Input Pins: 6· Flash Memory:32KB ·SRAM:2K
- Clock Speed: 16 MHZ

3.1.1 pH SENSOR :

A pH sensor is an analytical device used to measure the acidity or alkalinity (hydrogen ion concentration) of a solution. It typically consists of a glass electrode sensitive to hydrogen

ions and a reference electrode that provides a stable potential. The potential difference between these electrodes is measured and converted into a pH value according to the Nernst equation.the ph sensor is shown in the below figure 3.2

Fig.3.2 PH sensor

In the detection of pesticides in fruits and vegetables, pH sensors are commonly integrated into enzyme-based biosensors, especially those using the enzyme acetylcholinesterase (AChE). This enzyme catalyzes the hydrolysis of acetylcholine into choline and acetic acid, causing a decrease in pH. When pesticides such as organophosphates or carbamates are present, they inhibit AChE activity, reducing or preventing acetic acid formation. Consequently, the change in pH becomes smaller or negligible.By measuring this variation in pH before and after exposure to the sample, the presence and concentration of pesticides can be determined. The smaller the pH change, the higher the pesticide concentration. This method is simple, rapid, and cost-effective, making it suitable for monitoring pesticide residues in fruits and vegetables to ensure food safety and quality control.

Specifications:

- Measurement range: 0 – 14 ph
- Electrode type: Glass electrode (measuring) with Ag/AgCl or calomel electrode
- Sensitivity (slope): ~59 mV per pH unit at 25°C
- Accuracy: ±0.01 to ±0.05 pH units
- Resolution: 0.01 pH
- Response time: Less than 30 seconds
- Operating temperature range: 0°C to 80°C (some sensors up to 100°C)

3.3.CONDUCTIVITY SENSOR:

A conductivity sensor is an analytical device used to measure the electrical conductivity of a solution, which indicates its ability to conduct electric current. Conductivity depends on the concentration of ions present in the solution. The sensor usually consists of two electrodes placed at a fixed distance; when an electric voltage is applied, ions in the solution carry the current between them. The resulting current is proportional to the solution's conductivity it is shown in the below figure 3.3

Fig.3.3 conductivity sensor

In the detection of pesticides in fruits and vegetables, conductivity sensors are often used in electrochemical biosensors. Many pesticides, especially organophosphates and carbamates, inhibit enzyme activity (such as acetylcholinesterase) or undergo chemical reactions that alter the ionic composition of the solution. When the enzyme catalyzes a reaction (e.g., hydrolysis of acetylcholine), ionic products are released, increasing

conductivity. However, in the presence of pesticides, enzyme inhibition reduces ion production, causing a decrease in conductivity.

Specifications:

- Measurement range: 0 to 200 mS/cm
- Electrode type: Typically platinum, stainless steel, or graphite electrodes
- Accuracy: ±1% of full-scale readins
- Resolution: 0.01 µS/cm to 0.1 mS/cm
- Response time: < 10 seconds
- Operating temperature range: 0°C to 80°C
- Power requirement: Low voltage

3.4 MQ3GAS SENSOR:

The MQ3 sensor is a semiconductor gas sensor primarily designed to detect alcohol vapors and other volatile organic compounds (VOCs) in air. It operates based on changes in the electrical resistance of a metal oxide semiconductor (usually tin dioxide, SnO_2) when exposed to certain gases. Under clean air conditions, oxygen molecules adsorb onto the sensor's surface, capturing free electrons and forming a high-resistance layer. When target gases (such as alcohols or pesticide vapors) contact the surface, they react with the adsorbed oxygen, releasing electrons back into the semiconductor. This decreases the sensor's resistance, and the change is measured as an output voltage.

Fig.3.4 MQ3 Gas Sensor

In the detection of pesticides in fruits and vegetables, the MQ3 sensor can identify the presence of volatile pesticide residues that evaporate from contaminated samples. When the fruit or vegetable emits pesticide vapors, these molecules interact with the SnO_2 layer, causing a measurable change in resistance. The magnitude of this change is proportional to

the concentration of pesticide vapors present.

The MQ3 sensor offers advantages such as low cost, fast response, and portability, making it suitable for on-site screening of pesticide contamination in agricultural produce, though it may require calibration and signal processing for accurate, specific detection

Specifications:

- Operating voltage: ~5 V DC
- Heater resistance: ~33 Ω ± 5%
- Current consumption (heater on): ~150–165 mA at 5 V
- Load resistance (recommended RL): ~200 kΩ (for typical use)
- Sensing or sensor resistance range (Rs): ~1 MΩ to 8 MΩ in clean air; ~2 kΩ–20 kΩ
- Digital output: TTL level
- Analog output: ~0.1 V–0.3 V at low contamination, up to ~4 V concentration
- Operating temperature range: ~–10 °C to +50 °C

3.5 DTH11 SENSOR:

The DHT11 sensor, primarily designed for measuring temperature and humidity, is a low-cost digital sensor widely used in environmental monitoring. However, its direct application in detecting pesticides in fruits and vegetables is limited. Pesticide detection typically requires chemical or biosensing approaches, such as enzyme inhibition assays, immunosensors, or advanced spectroscopic techniques, because pesticides are chemical residues, not environmental parameters like temperature or humidity and it’s controls thtemperature .

In general, piezoelectric buzzers that operate on the self-drive method require special handling precautions. These include maintaining the rated voltage range, avoiding resistors in series with the power source, ensuring proper circuit design for stable oscillation, and preventing any obstruction near the sound-emitting hole.

However, in our project, we are using a piezoelectric buzzer with a built-in driver circuit (externally driven type). Therefore, an additional self-drive circuit is not required. The buzzer is directly interfaced through a transistor driver from the microcontroller, ensuring stable operation and consistent sound output without the need for external oscillation or feedback control.

control of the system using a simple and user-friendly mobile interface.

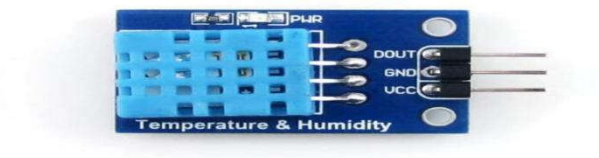


Fig.3.5 DHT11 Sensor

This method allows remote monitoring and control of the system without requiring an internet connection. The use of a standard Android phone makes the setup cost-effective and easy to operate, as messages can be sent using any basic SMS application. Moreover, it enhances the system's flexibility, enabling users to control and monitor the device from any location within the network coverage area.

Specifications:

- Operating (supply) voltage: 3.3 V to 5.5 V DC.
- Measurement ranges:
- Humidity: ~20 % to 90 % RH
- Temperature: 0 °C to 50 °C.
- Accuracy:
- Humidity: about ±5 % RH typical.
- Temperature: about ±2 °C typical.
- Resolution: 1 % RH for humidity; 1 °C for temperature

Typical current consumption: ~0.3 mA when measurements

3.6 L239D MOTOR DRIVE:

The L293D motor driver is a widely used integrated circuit for controlling DC motors and stepper motors, allowing precise movement in electronic projects. In the context of pesticide detection in fruits and vegetables, it plays a crucial role in automating sample handling and sensor positioning. Typically, a pesticide detection system involves sensors, microcontrollers, and mechanical actuators. TheL293D motor driver interfaces between the

microcontroller and the motors, enabling controlled rotation or linear movement of robotic arms or conveyor belts that transport the produce.

Fig.3.6 L293D motor drive

Using the L293 motor driver, the system can rotate or move samples with precise speed and direction control, ensuring consistent exposure to detection sensors. This reduces human error, improves throughput, and enhances the reproducibility of measurements.

Overall, while the L293 does not directly detect pesticides, it is essential for the mechanical automation of the detection process. Its reliability, ability to drive multiple motors, and compatibility with microcontrollers make it an effective component in modern automated pesticide detection systems for fruits and vegetables

Specifications:

- Motor supply voltage (Vs) range: 4.5 V up to 36 V absolute maximum.
- Logic supply (Vcc1) typically ~5 V
- Continuous current output per channel: 600 mA (typical) under normal conditions.
- Peak (non-repetitive) current per channel: up to 12volts
- Operating temperature range: 0 °C to 70 °C

3.7 LCD MODULE:

The LCD module plays na crucial role in automated pesticide detection systems for fruits and vegetables by serving as a controlled light source for optical or photometric sensing techniques. Many modern detection methods rely on the interaction between light and the chemical residues present on the surface of produce. LCDs provide stable, monochromatic light at specific wavelengths, which can be absorbed, reflected, or fluoresced by pesticide molecules. This optical response is then captured by sensors, such as photodiodes, spectrometers, or camera-based systems, enabling rapid and non-destructive detection.

The LCD module is shown in the below figure 3.7

Fig.3.7 LCD Module

In an automated setup, the LED module is often synchronized with conveyor belts or robotic arms to illuminate fruits and vegetables uniformly during scanning. While LEDs do not detect pesticides directly, they provide the controlled illumination essential for optical sensing techniques, improving sensitivity, repeatability, and throughput in pesticide detection systems. This integration enables safer and more efficient monitoring of food quality.

Specifications

- Supply voltage: around 3.3 V to 5 V DC
- Output luminous flux: e.g., ~200 lm under rated current.
- Rated power consumption: ~3 W
- Forward voltage (typical): e.g., ~11.2 V
- Operating temperature range: e.g., -40 °C to +85 °C
- Lifetime: typical LED modules are rated for ~3 to 5 years

3.8 . BUZZER MODULE:

The buzzer module serves as an alert mechanism in automated pesticide detection systems for fruits and vegetables, providing immediate feedback to users when pesticide residues

exceed safe thresholds. In such systems, sensors—whether chemical, optical, or electrochemical—analyze produce for pesticide contamination. Once the microcontroller processes the sensor data and identifies unsafe levels, it activates the buzzer module to emit an audible alarm. This rapid notification allows operators to take prompt action, such as segregating contaminated items, thereby improving food safety and preventing human exposure to harmful chemicals the buzzer module is shown in the below figure 3.8

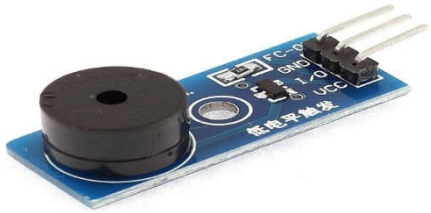


Fig.3.8 Buzzer module

Typically, the buzzer module is integrated with microcontrollers or control boards and can be either piezoelectric or magnetic. Piezo buzzers are preferred in many systems due to their low power consumption, compact size, and reliable sound output. The module is easy to interface: the microcontroller sends a digital signal to trigger the buzzer, which produces a consistent tone or a variable alert pattern. Some advanced systems also synchronize the buzzer with LED indicators for visual confirmation, making the detection system more intuitive and user-friendly.

While the buzzer does not detect pesticides directly, it is essential for real-time alerts and system interactivity. Its integration ensures that users can quickly identify and respond to contaminated fruits and vegetables, enhancing the efficiency, safety, and usability of automated pesticide detection setups.

Specifications:

- Operating voltage: ~3.3 V to 5 V DC
- Current consumption: less than ~20-30 mA
- Sound frequency: about 1.5 kHz to 2.5 kHz
- Sound pressure level: typically ≥ ~80-85 dB at ~10 cm
- Operating temperature range (varies by model): e.g., –20 °C to +70 C

3.9. REGULETED POWER SUPPLY (RPS):

A regulated power supply (RPS) provides a stable and consistent voltage or current to analytical instruments used in detecting pesticides in fruits and vegetables. Stable power is essential for maintaining the accuracy, reliability, and sensitivity of electronic detection systems such as a biosensors. Fluctuations in power can cause noise, signal drift, or measurement errors, affecting the precision of pesticide analysis. By delivering constant electrical output, an RPS ensures consistent instrument performance and reliable data during pesticide residue testing, thus supporting accurate monitoring of food safety and compliance with regulatory standards.

3.10 CABLES:

Cables play an important role in the detection of pesticides in fruits and vegetables by ensuring reliable electrical connections between sensors, instruments, and power sources. They transmit signals, data, and power with minimal loss or interference, which is crucial for accurate and stable measurements. High-quality, shielded cables prevent noise and electromagnetic interference that could affect sensitive analytical equipment such as chromatographs, spectrometers, or biosensors. Proper cable selection and management enhance system performance, safety, and durability in laboratory setups, ensuring consistent and precise results during pesticide residue detection and analysis in food safety testing.

Fig.3.10 cables

4. SOFTWARE REQUIREMENT

This project is implemented using following software's:

- Embedded C language
- Arduino IDE

4.1.EMBEDDED C LANGUAGE:

Embedded C is a crucial programming language in the field of embedded systems. These systems are at the heart of numerous modern devices, from smartphones and home appliances to automotive control systems and industrial machinery.

Embedded C is specifically designed for embedded systems, and its importance lies in its efficiency, portability, and close-to-the-hardware capabilities. Embedded systems often have limited processing power and memory, making efficiency paramount. Embedded C allows developers to write compact and optimized code, ensuring these systems operate smoothly while conserving resources.

Portability is another key aspect. Embedded C code can be written to be platform-independent, enabling the same code to run on various micro-controllers and processors, enhancing re-usability and minimizing development time.

Moreover, Embedded C provides low-level access to hardware components, making it ideal for tasks like controlling sensors, motors, and communication interfaces. Its ability to work at a hardware level ensures precise control and real-time responsiveness, crucial for safety-critical applications.

In safety-critical industries like healthcare and automotive, Embedded C's reliability and determinism are invaluable. It allows for the development of robust and predictable systems, reducing the risk of failures.

4.2 ARDUINO IDE:

Arduino IDE (Integrated Development Environment) is an open-source software platform used for programming and developing applications for Arduino microcontroller detection of the presticides in fruit and vegetables. It provides a user-friendly interface for writing, compiling, and uploading code to Arduino devices, making it accessible to both beginners and experienced developers.

The IDE supports the Arduino programming language, which is based on C/C++, and offers a range of libraries and example code to simplify the development process. It allows users to interact with various sensors, actuators, and other hardware components, enabling the creation of a wide array of projects, from simple LED blinkers to complex robotics and IOT applications.

Arduino IDE is cross-platform, compatible with Windows, and Linux, making it widely adopted in the maker and electronics enthusiast communities. Its simplicity and extensive community support make it a popular choice for prototyping and experimenting with hardware projects.

4.3 Program

```
#include <LiquidCrystal.h>
#include <stdio.h>

//LiquidCrystal lcd(6, 7, 5, 4, 3, 2);
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

#include <Wire.h>
#include "dht.h"

int phv=0;
int condv=0;

char smoke_string[20];
```

```
char fruit_string[30];

unsigned char rcv,gchr,gchr1;
int cntlmk=0;

int smoke     = 16;
int red_led   = 13;
int green_led = 12;

int buzzer    = 22;

#define dht_apin 11
dht DHT;



float tempc=0,humc=0;

char res[130];

void serial1Flush()
{
  while(Serial1.available() > 0)
  {
    char t = Serial1.read();
  }
}

char check(char* ex,int timeout)
{
  int i=0;
  int j = 0,k=0;
  while (1)
  {
    sl:
```

```
 if(Serial1.available() > 0)
 {
  res[i] = Serial1.read();
  if(res[i] == 0x0a || res[i]=='>' || i == 100)
  {
   i++;
   res[i] = 0;break;
  }
  i++;
 }
 j++;
 if(j == 30000)
 {
  k++;
  // Serial2.println("kk");
  j = 0;
 }
 if(k > timeout)
 {
  //Serial2.println("timeout");
  return 1;
  }
}//while 1
if(!strncmp(ex,res,strlen(ex)))
{
 // Serial2.println("ok..");
 return 0;
 }
else
{
 // Serial2.print("Wrong  ");
 // Serial2.println(res);
 i=0;
 goto sl;
```

```
  }
}

char buff[200],k=0;
void upload(unsigned int s1,unsigned int s2);
char readserver(void);
void clearserver(void);

const char* ssid = "iotserver";
const char* password = "iotserver123";

int sti=0;
String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

void beep()
{
 digitalWrite(buzzer, HIGH);delay(2500);digitalWrite(buzzer, LOW);delay(500);
}
void setup()
{
 Serial.begin(9600);//serialEvent();
 Serial1.begin(9600);

 pinMode(buzzer, OUTPUT);
 pinMode(red_led, OUTPUT);pinMode(green_led, OUTPUT);
 pinMode(smoke, INPUT);

 digitalWrite(buzzer, LOW);
 digitalWrite(red_led, HIGH);digitalWrite(green_led, HIGH);

 lcd.begin(16, 2);
 lcd.print(" Normal Fruit");
 lcd.setCursor(0,1);
```

```
  lcd.print(" Detection IOT");
    delay(2500);

  wifiinit();
    delay(2500);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:"); //2-3-4,0
  lcd.setCursor(5, 0);
  lcd.print("H:");//7-8-9,0
  lcd.setCursor(10,0);
  lcd.print("P:");  //12-13-14,1

  lcd.setCursor(0,1);
  lcd.print("S:"); //2-3-4,1
  lcd.setCursor(5,1);
  lcd.print("C:"); //7-8-9,1
  //serialEvent();
}

char bf3[50];
int g=0,f=0,count=0,lc=0;

void loop()
{
   DHT.read11(dht_apin);

   tempc = DHT.temperature;
   humc  = DHT.humidity;

   lcd.setCursor(2,0);convertl(tempc);
   lcd.setCursor(7,0);convertl(humc);
```

```
phv = analogRead(26);
phv = ((1024 - phv)/103);

lcd.setCursor(12,0);convertl(phv);

memset(smoke_string,'\0',strlen(smoke_string));
if(digitalRead(smoke) == LOW)
  {
   lcd.setCursor(2,1);lcd.print("Det");
   strcpy(smoke_string,"Detection");
  }
if(digitalRead(smoke) == HIGH)
  {
   lcd.setCursor(2,1);lcd.print("---");
   strcpy(smoke_string,"---");
  }

condv = analogRead(27);
condv = (1025 - condv);
condv = (condv/10);
lcd.setCursor(7,1);convertl(condv);

memset(fruit_string,'\0',strlen(fruit_string));
if(tempc >= 40 || phv >= 10)
  {
   strcpy(fruit_string,"Non_Organic");
   digitalWrite(red_led, LOW);digitalWrite(green_led, HIGH);
  }
else if(digitalRead(smoke) == LOW || condv > 50)
  {
   strcpy(fruit_string,"Damage");
   digitalWrite(red_led, LOW);digitalWrite(green_led, HIGH);
   beep();
  }
```

```
    else
      {
       strcpy(fruit_string,"Normal");
       digitalWrite(red_led, HIGH);digitalWrite(green_led, LOW);
      }

    delay(1000);
    cntlmk++;
    if(cntlmk >= 40)
      {cntlmk=0;
       upload(tempc,humc,phv,smoke_string,condv,fruit_string);
      }
}



char bf2[50];
void upload(int s1,int s2,int s3,const char *s4,int s5,const char *s6)
{
  delay(2000);
  lcd.setCursor(15, 1);lcd.print("U");
  serial1Flush();
  Serial1.println("AT+CIPSTART=4,\"TCP\",\"projectsfactoryserver.in\",80");
     delay(8000);

  memset(buff,0,strlen(buff));
  sprintf(buff,"GET
http://projectsfactoryserver.in/storedata.php?name=iot1539&s1=%u&s2=%u&s3=%u&s4
=%s&s5=%u&s6=%s\r\n\r\n",s1,s2,s3,s4,s5,s6);

     serial1Flush();
     sprintf(bf2,"AT+CIPSEND=4,%u",strlen(buff));
     Serial1.println(bf2);
       delay(5000);
```

```
        serial1Flush();
        Serial1.print(buff);

           delay(2000);

        Serial1.println("AT+CIPCLOSE");
     lcd.setCursor(15, 1);lcd.print(" ");
}

char readserver(void)
{
 char t;
 delay(2000);
 lcd.setCursor(15, 1);lcd.print("R");
 serial1Flush();
 Serial1.println("AT+CIPSTART=4,\"TCP\",\"projectsfactoryserver.in\",80");

 //http://projectsfactoryserver.in/last.php?name=amvi001L

    delay(8000);
    memset(buff,0,strlen(buff));
    sprintf(buff,"GET http://projectsfactoryserver.in/last.php?name=iot4L\r\n\r\n");
    serial1Flush();
    sprintf(bf2,"AT+CIPSEND=4,%u",strlen(buff));
    Serial1.println(bf2);

      delay(5000);


        serial1Flush();
        Serial1.print(buff);

     //read status
```

```
        while(1)
        {
          while(!Serial1.available());
           t = Serial1.read();
          // Serial2.print(t);
           if(t == '*' || t == '#')
           {
            if(t == '#')return 0;
            while(!Serial1.available());
             t = Serial1.read();
           //  Serial.print(t);
             delay(1000);
              serial1Flush();
             return t;
           }
        }
             delay(2000);

      Serial1.println("AT+CIPCLOSE");
      lcd.setCursor(15, 1);lcd.print(" ");
      delay(2000);
return t;
}

void clearserver(void)
{
  delay(2000);
  lcd.setCursor(15, 1);lcd.print("C");
  serial1Flush();
  Serial1.println("AT+CIPSTART=4,\"TCP\",\"projectsfactoryserver.in\",80");

  //sprintf(buff,"GET
http://projectsfactoryserver.in/storedata.php?name=iot1&s10=0\r\n\r\n");
     delay(8000);
```

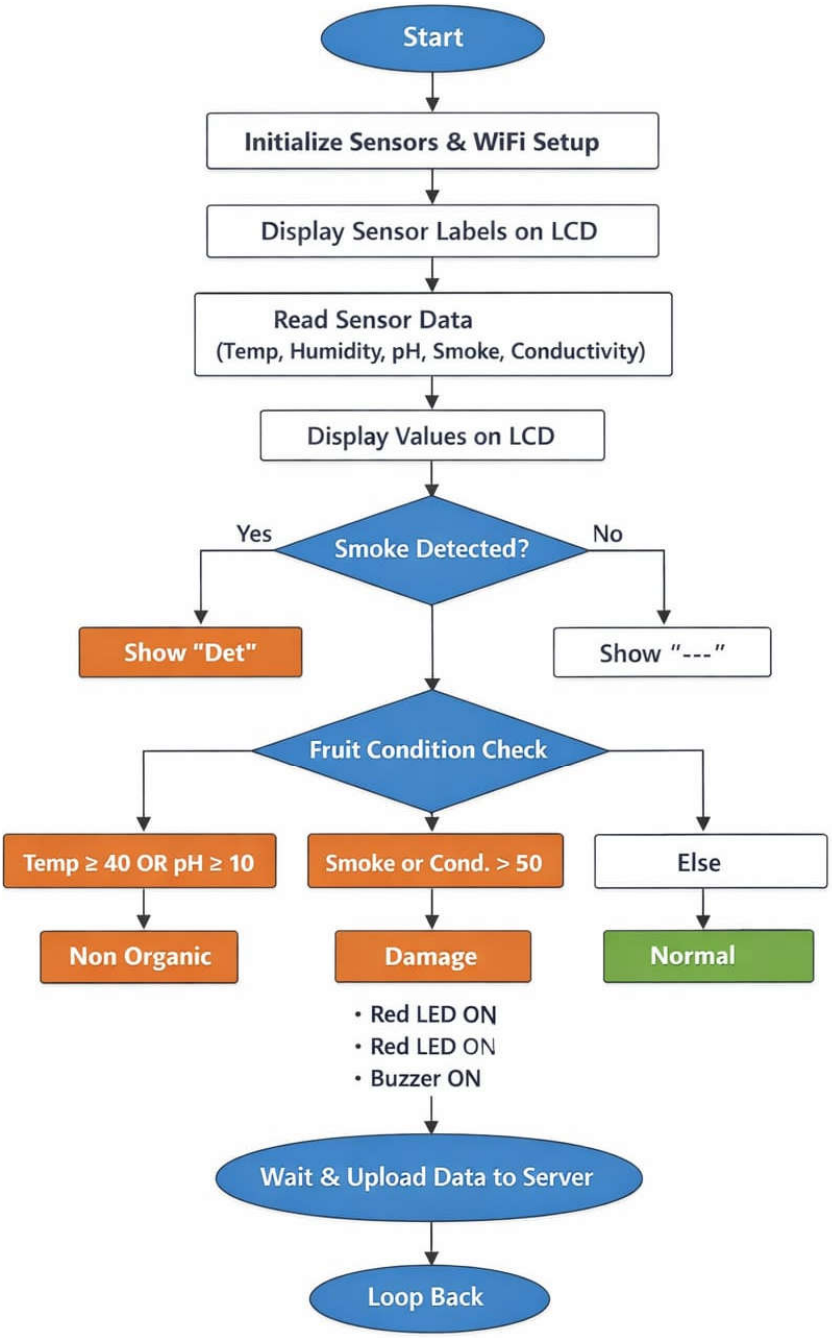
```
    memset(buff,0,strlen(buff));
    sprintf(buff,"GET
http://projectsfactoryserver.in/storedata.php?name=iot4&s10=0\r\n\r\n");
    serial1Flush();
    sprintf(bf2,"AT+CIPSEND=4,%u",strlen(buff));
    Serial1.println(bf2);

      delay(5000);



       serial1Flush();
       Serial1.print(buff);

       delay(2000);
       serial1Flush();



    Serial1.println("AT+CIPCLOSE");
    lcd.setCursor(15, 1);lcd.print(" ");
    delay(2000);
}


void wifiinit()
{
 char ret;
  st:
  Serial1.println("ATE0");
  ret  = check((char*)"OK",50);
  Serial1.println("AT");
  ret  = check((char*)"OK",50);
  if(ret != 0)
  {
```

```
   delay(1000);
  goto st;
 }

    lcd.clear();lcd.setCursor(0, 0);lcd.print("CONNECTING");
 Serial1.println("AT+CWMODE=1");
  ret  = check((char*)"OK",50);
cagain:

 serial1Flush();
 Serial1.print("AT+CWJAP=\"");
 Serial1.print(ssid);
 Serial1.print("\",\"");
 Serial1.print(password);
 Serial1.println("\"");
 if(check((char*)"OK",300))goto cagain;
 Serial1.println("AT+CIPMUX=1");
 delay(1000);


 lcd.clear();lcd.setCursor(0, 0);lcd.print("WIFI READY");
}


void convertl(unsigned int value)
{
 unsigned int a,b,c,d,e,f,g,h;

     a=value/10000;
     b=value%10000;
     c=b/1000;
     d=b%1000;
     e=d/100;
     f=d%100;
```

```
    g=f/10;
    h=f%10;


    a=a|0x30;
    c=c|0x30;
    e=e|0x30;
    g=g|0x30;
    h=h|0x30;


  //lcd.write(a);
  //lcd.write(c);
  lcd.write(e);
  lcd.write(g);
  lcd.write(h);
}

void converts(unsigned int value)
{
 unsigned int a,b,c,d,e,f,g,h;

    a=value/10000;
    b=value%10000;
    c=b/1000;
    d=b%1000;
    e=d/100;
    f=d%100;
    g=f/10;
    h=f%10;


    a=a|0x30;
    c=c|0x30;
```

```
    e=e|0x30;
    g=g|0x30;
    h=h|0x30;


  Serial1.write(a);
  Serial1.write(c);
  Serial1.write(e);
  Serial1.write(g);
  Serial1.write(h);
}
```

4.4 Flow Chart



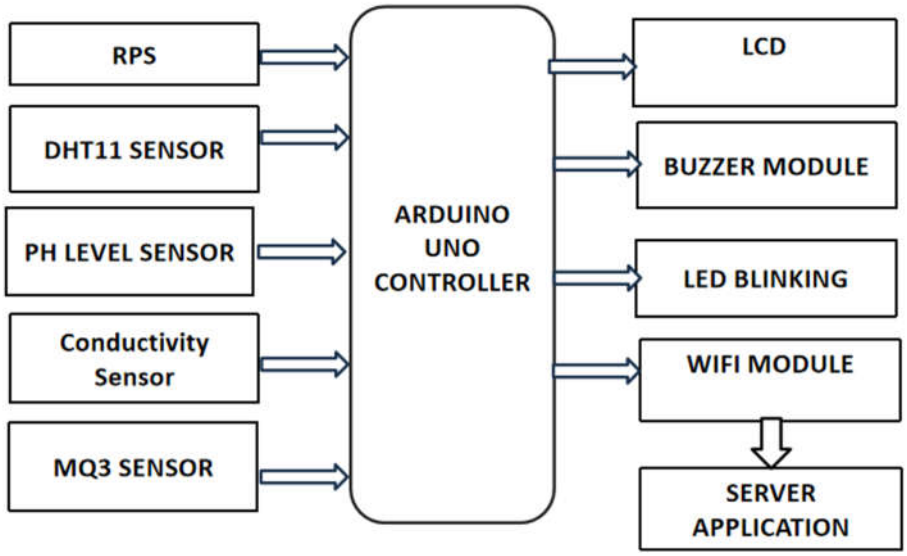
The flowchart represents the working process of an IoT-based Pesticide Detection System, which monitors environmental parameters and determines the condition of fruits using various sensors.The process begins with the Start

block, where the system is powered on. Initially, all sensors and the Wi-Fi module are initialized, ensuring proper communication and data acquisition. After initialization, the system displays the sensor labels on the LCD for user reference.Next, the system proceeds to read sensor data, including parameters such as temperature, humidity, pH level, smoke, and conductivity. These values are then displayed on the LCD screen for real-time monitoring.The system then checks whether smoke is detected. If smoke is present, it displays "Det" (Detected) on the LCD otherwise, it shows a default indication ("---").Following this, the system performs a fruit condition check based on predefined thresholds:

- If the temperature is greater than or equal to 40°C or pH is greater than or equal to 10, the fruit is classified as Non-Organic.
- If smoke is detected or conductivity exceeds 50, the fruit is considered Damaged.
- If none of these conditions are met, the fruit is categorized as Normal.
- In case of damaged fruit, alert mechanisms are activated, including turning on LEDs and a buzzer to notify the user.
- After processing the data, the system enters a stage where it waits and uploads the collected data to the server via Wi-Fi for remote monitoring and analysis.Finally, the system performs a loop back, continuously repeating the process to ensure real-time monitoring and detection.

5. TECHNICAL ARCHITECTURE

5.1 BLOCK DIAGRAM



1. Regulated Power Supply (RPS – 12V, 1A)

- Provides stable 12V/5V power to all modules.
- Protects components from voltage fluctuations.
- Ensures reliable and continuous operation of sensors.

The RPS supplies regulated 12V and 5V to the microcontroller, sensors, and Wi-Fi module, ensuring every component receives a safe and constant power source.

2. Arduino Uno (Microcontroller – 5V)

- Acts as the brain of the entire system.
- Reads, processes, and analyzes sensor data.
- Controls display, alerts, and communication activities.

Arduino collects data from all sensors, processes the readings, compares them with preset limits, and then triggers the LCD, buzzer, LED, and sends values to the ESP8266.

3. Raspberry Pi Pico

- Provides faster dual-core processing if additional computation is required.
- Can serve as an alternative to Arduino.

- SupportsMicroPython/C for flexible programming.

  If included, it handles additional computation such as data filtering or advanced sensor processing and then communicates results to the Arduino or cloud

4. 16×2 LCD Display (5V)

- Shows real-time sensor readings.
- Helps users see immediate contamination status.
- Works even without internet connection.

  The LCD displays the pH value, MQ3 readings, conductivity, temperature, humidity, and the system's safety status like "Safe" or "Pesticide Detected."

5. ESP8266 Wi-Fi Module

- Enables IoT-based monitoring.
- Sends sensor data to the cloud or mobile app.
- Allows long-distance alerts and data logging.

  The ESP8266 receives processed values from Arduino and transmits them wirelessly to a server or dashboard for remote monitoring.

6. MQ3 Gas Sensor

- Detects pesticide vapors and chemical gases.
- Sensitive to harmful volatile organic compounds (VOCs).
- Provides quick contamination detection.

  The MQ3 senses hazardous pesticide gases and sends an analog signal to Arduino, where the system decides if the gas level is within the safe range or not.

7. DHT11 Temperature & Humidity Sensor

- Monitors environmental conditions near produce.
- Helps understand how temperature/humidity affect sensor accuracy.
- Indicates storage quality of fruits/vegetables.

8. Conductivity Sensor (0–12V)

- Measures electrical conductivity of fruit/vegetable extracts.
- Detects dissolved chemicals or pesticide residues.

- Helps determine contamination or freshness.

  The sensor outputs a voltage depending on the conductivity of the sample. Arduino reads this value to check if chemicals or residues are present beyond normal levels.

9. pH Sensor (0–14 Range)

- Indicates acidity/alkalinity of produce.
- Helps detect chemical contamination.
- Acts as a freshness indicator of fruits/vegetables.

  The pH sensor produces a voltage based on the hydrogen ion concentration in the sample. Arduino converts this into pH value and checks whether it falls within the safe range.

10. Buzzer (30 dB, 5V)

- Provides instant audio alert for unsafe conditions.
- Enhances safety by notifying users immediately.
- Helps identify contamination without looking at the screen.

  When any sensor reading crosses the unsafe limit, Arduino activates the buzzer to alert the user of pesticide presence or abnormal values.

11. LED Indicator (5V)

- Gives quick visual status of the system.
- Easy to understand: green = safe, red = danger.
- Useful in noisy environments where buzzer may not be heard.

  Arduino controls LEDs based on sensor analysis. Safe readings turn on green LED; contaminated readings activate the red LED.

5.2 OVERVIEW OF BLOCK DIAGRAM

The block diagram represents the overall structure and workflow of the pesticide detection system. It shows how each component is interconnected and how data flows from the

sensors to the microcontroller and finally to the output devices. The system begins with the Regulated Power Supply (RPS), which provides stable 12V and 5V power to all modules, ensuring safe and reliable operation. Multiple sensors such as the MQ3 gas sensor, pH sensor, conductivity sensor, and DHT11 temperature–humidity sensor are connected to the Arduino controller. These sensors continuously measure different parameters related to chemical contamination and environmental conditions.

The Arduino acts as the central processing unit, receiving analog and digital signals from all sensors. It processes these values, compares them with predefined thresholds, and determines whether the fruit or vegetable sample is safe or contaminated. The processed information is then displayed on the 16×2 LCD for real-time monitoring. If any reading crosses the safe limit, the Arduino triggers the buzzer and LED indicators to alert the user immediately.

Additionally, the ESP8266 Wi-Fi module is connected to the Arduino to enable IoT-based monitoring. It sends the processed data wirelessly to a server or mobile application, allowing users to view contamination status remotely. Overall, the block diagram illustrates a complete smart pesticide detection system where sensing, processing, alerting, and communication components work together to ensure food safety

6.HARDWARE MODULE TESTING AND RESULTS

6.1 HARDWARE MODULE

The Hardware Module of Detection of pesticides in fruits and vegetables is shown in Fig 6.1

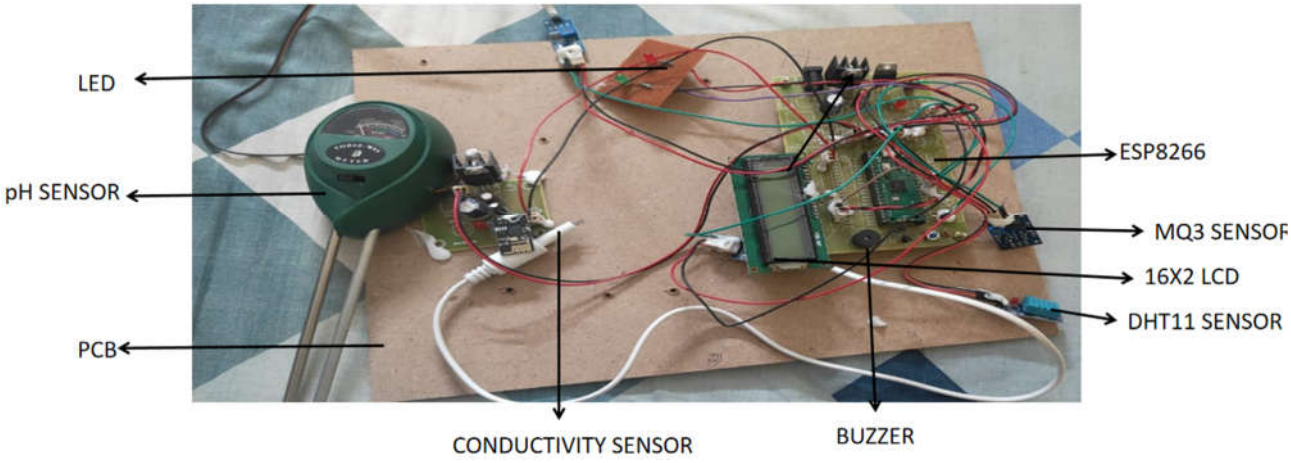


Fig 6.1 Hardware Module

WORKING

The working principle of the system is based on collecting multiple sensor readings from fruits and vegetables, processing those values using the ESP8266 controller, and displaying the final condition of the sample both locally and remotely through IoT. The project uses sensors like the pH sensor, MQ3 gas sensor, conductivity sensor and DHT11 to measure key parameters that reveal the chemical condition, freshness and possible pesticide contamination in fruits and vegetables.

When the system is powered ON, all sensors begin collecting readings.

The pH sensor measures acidity levels.

The MQ3 sensor detects harmful chemical vapors or pesticide gases.

The conductivity sensor checks the amount of dissolved chemicals in the fruit extract.

The DHT11 sensor records temperature and humidity around the sample.

All these raw sensor values are continuously sent to the ESP8266 which acts as the central processor. The ESP8266 compares each value with predefined safe limits. If the readings stay within the normal range, the system marks the status as “Normal” and displays it on the 16×2 LCD. The LED glows green to indicate safe produce.

If any sensor reading crosses the threshold—for example, high gas detection, abnormal pH, or high conductivity—the ESP8266 marks the sample as “Damage” or “Pesticide Detected.” In such cases, the red LED glows and the buzzer produces an alert sound, warning the user instantly.

Along with this, the ESP8266 Wi-Fi module sends all processed data to the cloud server. Users can view this information on their mobile phone as shown in the screenshot. The uploaded data includes temperature, humidity, pH, gas level, conductivity, and final status. This allows users to remotely monitor the condition of fruits and vegetables from anywhere. Thus, the working principle combines sensor measurement, microcontroller processing, alert generation, and IoT-based remote monitoring to create a complete pesticide detection system.

6.2 TESTING AND RESULTS:

CASE -1: Undamaged Fruit (Orange)

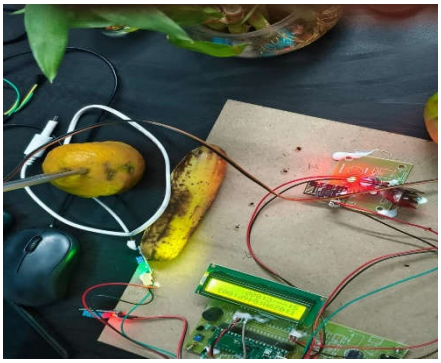


Fig 6.2.1 Undamaged Fruit (Orange)

In this case, naturally grown, chemical-free orange sample is tested. The pH sensor readings remained within the normal biological range for each fruit and the MQ-3 sensor showed very low or zero VOC levels, indicating the absence of artificial ripening agents or pesticide vapors. Conductivity remained minimal, showing no presence of chemical ions. Across all output interfaces, the system classified these samples as Organic. The webpage displayed the status as Normal, the green LED turned ON indicating safe fruit condition, and the LCD displayed readings such as “pH Normal, MQ3 Low” followed by STATUS: ORGANIC**.** These consistent results confirm that the system accurately identifies organic fruits with stable sensor readings.

Case-2: Pesticide Contaminated Fruit (Orange)

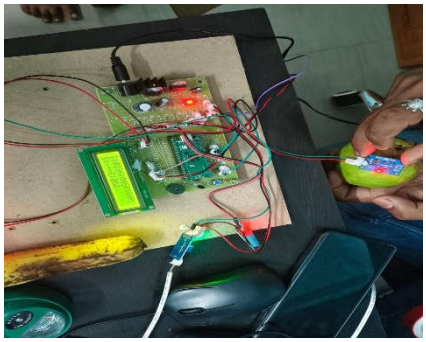


Fig 6.2.2 Pesticide Contaminated Fruit (Orange)

In this case, market-purchased fruits suspected to be chemically ripened or containing pesticide residues were tested. The pH sensor showed irregular or shifted values compared to natural levels, suggesting chemical alteration. The MQ-3 gas sensor detected higher VOC concentrations, indicating the presence of ripening chemicals or pesticide vapors. Conductivity values also showed noticeable spikes, reflecting chemical ion presence. As a result, the system classified these fruits as damaged/pesticide detected. On the webpage, the status appeared as Non organic, the red LED illuminated to warn of contamination, and the LCD displayed messages such as “Abnormal pH, High MQ3” followed by STATUS: PESTICIDE DETECTED**.** These results demonstrate that the system successfully detects chemically treated or pesticide-affected fruits through measurable sensor deviations.

CASE: 3 – Pesticide Contaminated Guava

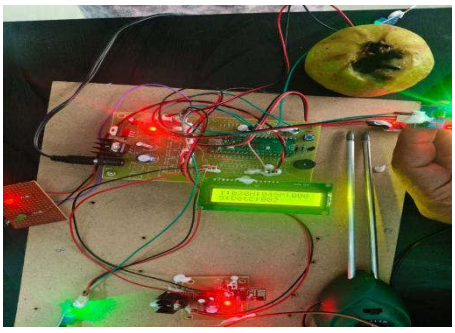


Fig 6.2.3 Pesticide Contaminated Guava

For organic guava, the MQ-3 sensor displayed consistently low and stable gas readings throughout the testing process, which indicates that the fruit did not emit any volatile organic compounds (VOCs) associated with pesticides, artificial ripening agents, or chemical preservatives. Since the guava was naturally grown without the use of chemicals, the sensor detected only the fruit’s natural aroma levels, which remained within the safe threshold. Based on this clean and steady MQ-3 response, the system correctly categorized the guava as Organic, and the results were reflected across all output interfaces with the

webpage showing a “Normal” status, the green LED turning ON, and the LCD displaying a message confirming that the fruit is free from chemical contamination.

CASE – 4: Undamged Guava

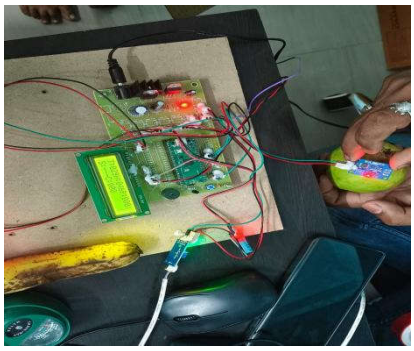


Fig 6.2.4 Undamaged Guava

For organic guava, the MQ-3 sensor displayed consistently low and stable gas readings throughout the testing process, which indicates that the fruit did not emit any volatile organic compounds (VOCs) associated with pesticides, artificial ripening agents, or chemical preservatives. Since the guava was naturally grown without the use of chemicals, the sensor detected only the fruit’s natural aroma levels, which remained within the safe threshold. Based on this clean and steady MQ-3 response, the system correctly categorized the guava as Organic, and the results were reflected across all output interfaces with the webpage showing a “Normal” status, the green LED turning ON, and the LCD displaying a message confirming that the fruit is free from chemical contamination.

CASE: 5 - Undamaged Potato (Without Chemicals)

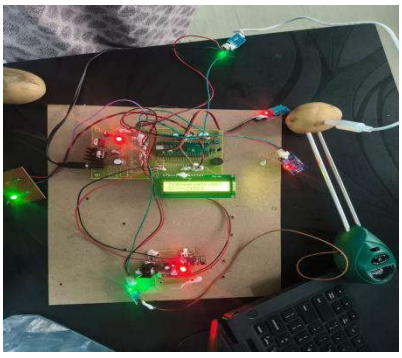


Fig 6.2.5 Undamaged Potato

For organically grown potatoes, the pH sensor displayed stable readings within the normal biological range of the vegetable, indicating that the potato’s internal acidity was natural and unaffected by any external chemicals. At the same time, the conductivity sensor showed very low ion activity, confirming the absence of chemical fertilizers, preservation agents, or surface treatments. Since organic potatoes do not contain synthetic residues, both sensors maintained consistent and predictable values throughout the analysis. Based on this

combination of normal pH and low conductivity, the system accurately classified the potato sample as Organic. This classification was reflected across all outputs—the webpage displayed a “Normal” or “Organic” status, the green LED illuminated to indicate safety, and the LCD clearly showed messages such as “Stable pH & Low Conductivity – ORGANIC,” confirming that the potato was free from harmful chemical contamination

CASE 6: Pesticide contaminated Potato (With Chemicals / Treated)

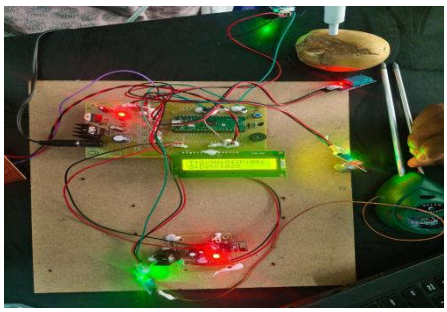


Fig 6.2.6 Pesticide Contaminated Potato

For inorganic or chemically treated potatoes, the pH sensor recorded noticeable variations from the expected natural range due to the presence of chemical residues, synthetic fertilizers, or artificial sprouting inhibitors often applied during storage. Additionally, the conductivity sensor measured significantly higher ionic activity, indicating the presence of chemical ions absorbed or retained on the potato surface or interior. These elevated conductivity and irregular pH values clearly signaled contamination that organic potatoes would not normally exhibit. Based on these abnormal readings, the system correctly identified the potato as Non-Organic. This detection was shown through all output interfaces, where the webpage displayed the status as “Non_Organic,” the red LED turned ON as a warning, and the LCD presented a message such as “Abnormal pH & High Conductivity – NON-ORGANIC,” confirming that the potato contained chemical residues or treatment compounds.

CASE – 7: Undamaged Tomato (Without Chemicals)

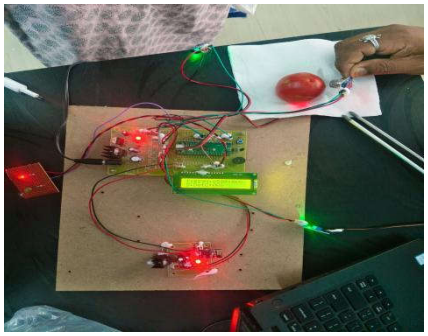


Fig 6.2.7 Undamaged Tomato

For organically grown tomatoes, the pH sensor showed stable acidity values within the natural tomato range, indicating that the ripening process occurred naturally without the influence of artificial chemicals or external agents. Along with this, the MQ-3 sensor recorded very low levels of volatile organic compounds (VOCs), confirming that no chemical vapors, artificial ripening sprays, or pesticide residues were present on the fruit. Organic tomatoes release only natural aroma gases, which fall well below the MQ-3 detection threshold, and this stability across both sensors clearly indicated purity. Based on these normal readings, the system classified the tomato as Organic. This result appeared across all outputs—the webpage displayed a "Normal / Organic" status, the green LED lit up, and the LCD showed a message such as "pH Normal, MQ3 Low – ORGANIC," confirming that the tomato is free from chemical contamination.

CASE -8: Pesticide Contaminated Tomato

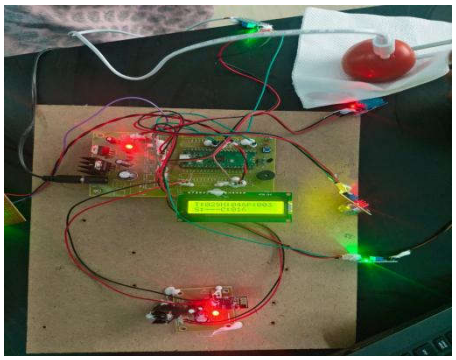


Fig 6.2.8 Pesticide Contaminated Tomato

For chemically treated or pesticide-exposed tomatoes, the pH sensor showed noticeable deviation from the natural pH range due to the presence of synthetic ripening agents, pesticide residues, or chemical coatings used to enhance shelf life. At the same time, the MQ-3 sensor detected significantly higher VOC levels because chemically treated tomatoes release gases from pesticides and artificial ripening substances such as ethylene-based chemicals. These elevated MQ-3 readings, combined with the irregular pH values, clearly indicated contamination. The system therefore classified the tomato as Non-Organic. This

classification was reflected across all interfaces—the webpage displayed a "Non_Organic" status, the red LED glowed to indicate chemical presence, and the LCD showed a warning message such as "Abnormal pH, High MQ3 – NON-ORGANIC," confirming that the tomato was affected by pesticide or chemical treatment.

CASE- 9: Undamaged Brinjal (Without Chemicals)

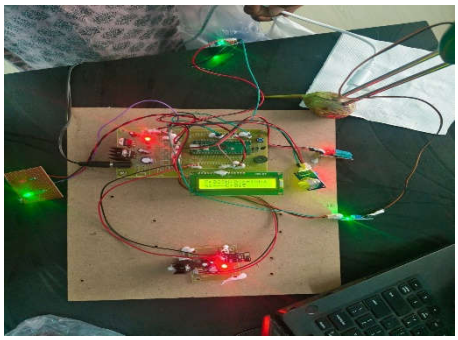


Fig 6.2.9 Undamaged Brinjal

For organically grown brinjal, the pH sensor displayed stable readings within the natural pH range of fresh brinjal extract, clearly indicating that the vegetable matured naturally without exposure to synthetic chemicals or harmful preservatives. Along with this, the conductivity sensor recorded low and steady conductivity values, showing that the internal ion concentration remained normal and free from chemical residues, artificial enhancers, or surface treatments. Organic brinjal contains only natural moisture and mineral content, which keeps conductivity at a minimal level, and this consistency across both sensors strongly confirmed its purity. Based on these natural and stable readings, the system classified the brinjal as Organic. This result was reflected across all output modules—the webpage displayed a "Normal / Organic" status, the green LED turned on, and the LCD showed a message such as "pH Normal, Conductivity Low – ORGANIC," confirming that the brinjal is free from any chemical contamination.

CASE – 10: Pesticide Contaminated Brinjal

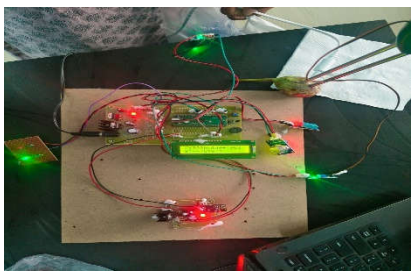
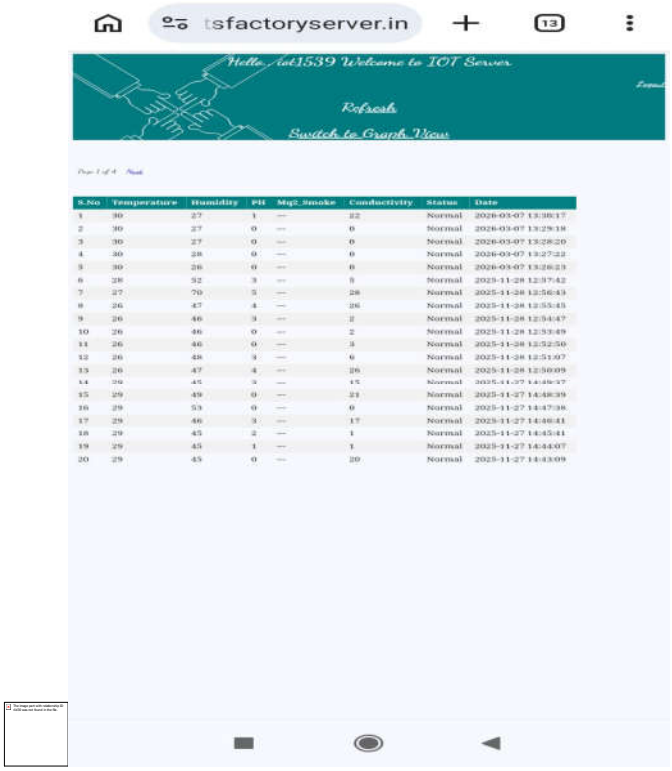


Fig 6.2.10 Pesticide Contaminated Brinjal

For chemically treated or pesticide-exposed brinjal, the pH sensor showed a significant deviation from the natural pH range due to the presence of chemical sprays, wax coatings, or synthetic preservatives commonly used to enhance shelf life. Simultaneously, the

conductivity sensor detected noticeably higher conductivity levels, caused by increased ion concentration resulting from pesticide residues, chemical seepage, or internal spoilage triggered by chemical reactions. These elevated conductivity readings, combined with abnormal pH values, provided clear evidence of chemical contamination or artificial treatment. As a result, the system classified the brinjal as Non-Organic. This classification appeared across all interfaces—the webpage displayed a "Non_Organic" warning, the red LED illuminated to indicate contamination, and the LCD presented a message such as "Abnormal pH, High Conductivity – NON-ORGANIC," confirming that the brinjal was affected by pesticides or chemical agents.

Result Table:



The system collected real-time sensor data from tomato samples using temperature, humidity, pH, MQ-3 (VOC), and conductivity sensors. Based on these parameters, each sample was classified into Normal**,** Non-Organic**,** or Damaged categories.

1. Normal Tomato Samples

Most of the collected readings fall under the Normal category. These samples showed consistent patterns across all sensors:

- Temperature values remained within 28–35°C, which is typical for stored tomatoes.
- Humidity levels varied between 39–53%**,** indicating natural moisture retention.
- pH levels were within the expected acidic range for tomatoes.
- The MQ-3 sensor showed no VOC detection, confirming the absence of chemical ripening agents.
- Conductivity values were very low, indicating the fruit's internal structure was stable and not chemically altered.

These readings confirm that the tomatoes in this category underwent natural ripening without artificial chemicals or degradation.

2. Non-Organic Tomato Samples

Two samples were classified as Non-Organic based on sensor deviations:

- The MQ-3 sensor detected volatile organic compounds (VOCs)**,** indicating exposure to artificial ripening chemicals such as ethylene spray or carbide.
- pH readings were unusually low (pH = 0), suggesting altered acidity due to chemical treatment.
- Temperature and humidity values also showed slight fluctuations compared to normal readings.The VOC detection is the primary indicator confirming that these tomatoes were chemically ripened rather than organically grown.

3. Damaged Tomato Sample

One sample was categorized as Damaged. This classification was based on:

- High humidity level (48%), which can be associated with internal spoilage.
- MQ-3 sensor detection of VOC**s**, indicating microbial activity or fruit decomposition.
- Conductivity value = 0, suggesting loss of internal fluid consistency.
- Inconsistent sensor patterns compared to normal samples.
- These characteristics reflect internal decay or fungal damage inside the fruit.

4. Sensor Data Interpretation

Each sensor contributed to reliable classification:

1. **Temperature and Humidity**

Provided insights into environmental conditions. Consistent values indicated stable storage, while deviations reflected chemical treatment or spoilage.

2. **pH Sensor**

Measured acidity levels. Natural tomatoes showed stable pH values, while non-organic samples exhibited unnaturally low readings.

3. **MQ-3 VOC Sensor**

Played the most critical role:"---" → No VOCs detected → Normal/organic samples "Detection" → VOCs present → Non-organic or damaged samples

4. **Conductivity Sensor**

Used to analyze internal fruit quality.Low values indicated firmness and freshness,higher values or zero indicated over-ripeness, chemical stress, or internal damage.

6. CONCLUSION AND FUTURE PLAN

6.1 CONCLUSION

The project "Pesticide Detection in Fruits and Vegetables" successfully demonstrates a smart, cost-effective, and portable system for identifying pesticide contamination using multiple integrated sensors. The prototype uses pH, conductivity, MQ3 gas, and DHT11 temperature humidity sensors to analyze chemical and environmental parameters of fruits and vegetables. These sensor readings are processed by the ESP8266 controller, which compares them with predefined safe thresholds and identifies whether the sample is Organic, Normal, or Pesticide-Contaminated.

The IoT-enabled ESP8266 module further enhances the system by sending real-time data to web dashboard, ensuring remote monitoring. Testing on various fruit samples showed clear, measurable differences in sensor outputs, validating the reliability of the system.

Overall, the project provides a simple, efficient, and user-friendly solution for assessing food safety. It supports consumers, vendors, and quality inspectors by offering immediate alerts through LEDs, buzzers, LCD displays, and online platforms. This system helps promote healthier food choices and reduces exposure to harmful chemicals present in conventionally grown produce.

6.2 FUTURE PLAN

The developed system shows strong potential for expansion and real-world application. Future improvements can include:

1. Sensor Accuracy Improvement

- Calibrate the sensors more precisely to improve measurement accuracy
- Use better-quality sensors for reliable real-time readings
- Reduce environmental noise and interference during detection

2. Compact and Portable Design

- Convert the prototype into a more compact and user-friendly device

- Design a handheld model for easy usage by vendors or consumers
- Improve power efficiency for longer operation

3. Expansion to More Vegetables in Stage–2

- Extend detection to: Leafy vegetables, Root vegetables, High-water-content produce.

REFERENCES

[1] Thorat, T. Advancements in techniques used for identification of pesticide residues: A review. Journal of Food Safety and Technology, 12(2), 45–52. (2023).

[2] Natarajan, S., Ramesh, K., & Kumar, P. Classification of organic and conventional vegetables using portable multispectral sensors. Sensors, 23(7), 3542. https://doi.org/10.3390/ s23073542 (2023).

[3] Jiang, N., Li, X., & Zhang, Y.Differentiation between organic and non-organic apples using computer vision and machine learning techniques. Sensors, 18(2), 422. https://doi.org/10.3390/s18020422 (2018).

[4] Miglione, A., et al. Electrochemical inhibition biosensor for direct detection of organophosphate pesticides on fruit peels. Sensors and Actuators B: Chemical, 420, 145635. (2025).

[5] Deen, A. A. J. Arduino-based smart IoT food quality monitoring system. International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering,11(4), 25–30. (2023).

[6] Yuan, Y., Chen, L., & Xu,J. Fruit freshness detection based on deep feature fusion of image and sensor data. Journal of Food Engineering, 345, 111522. (2024).

[7] IJERT Detection of pesticides in organic fruits and vegetables using IoT and machine learning. International Journal of Engineering Research & Technology, 13(5), 234–239. (2024).