Acknowledgement:

This project has presented, to me, an objective, a goal, a challenge and a milestone of my pre-final academic year in the engineering department at VIT, Vellore.

This project marks the final hurdle that I tackle, of hopefully what would be one of the many academic challenges I have taken upon and am yet to take. However, I could not have made it without the patient support and guidance from the following people.

Firstly I want to take this opportunity to have special Thanks to My supervisor Mr. Manish Kumar who helped me throughout this project by providing valuable guidance and advice as well as acquiring all components needed for this project to become a success.

And finally thanks to my family and friends who have encouraged and spurred me on through this project.

Project Donation Form

Student Name (SHUBHAM BHATT)
Registration Number (21BEC2350)

03/05/2024

Faculty Name: PROF: MANISH KUMAR

Sub: Donation of project “SMARTT DUSTBIN”

Dear Sir,

I am writing you to your support for representing my mini project “SMART DUSTBIN” , for visual display in School of Electronics Engineering.

I studied and knew all basic concept about Microcontrollers and Embedded System. I utilized my basic knowledge and tried to solve society problems like “GARBAGE”. This project is just a demonstration of how microcontrollers can solve society’s problems. I believe this is a proud moment for me to donate my project for display purposes.

Thank you for taking this project as a donation. If possible, kindly put my project as display purpose in current semester or upcoming semester. If I am available at campus, I may upgrade my current project to the next level and make it a better project for display purposes in the School of Electronics Engineering.

Sincerely,

Student name

SHUBHAM BHATT

21BEC2350

ABSTRACT:

This project aims to design and implement a Smart Dustbin system using the C programming language. The Smart Dustbin is equipped with sensors to detect the presence and level of waste inside the bin. The system utilizes ultrasonic or infrared sensors to accurately measure the waste level and trigger actions accordingly.

In this project, we explore the integration of sensor interfacing, data processing, and control mechanisms in C programming to create an efficient and responsive Smart Dustbin. The program continually monitors the sensor inputs and activates the lid-opening mechanism when the waste level reaches a predefined threshold. Additionally, the system includes error handling and feedback mechanisms to ensure reliable operation under varying conditions.

Through this project, we demonstrate the practical application of C programming in developing embedded systems for real-world scenarios. The Smart Dustbin project serves as an educational tool for learning about sensor integration, data processing, and control algorithms in the context of waste management.

By implementing this project, we aim to promote the understanding and adoption of smart technologies in waste management, contributing to cleaner and more sustainable environments.

CHAPTERS.

1) Introduction:

In the era of smart technology, traditional waste management methods are being revolutionized by the integration of microcontrollers like Arduino into everyday objects. Smart dustbins represent one such innovation, leveraging the power of microcontrollers to enhance waste collection processes and improve urban cleanliness.

Introduction to Arduino and Microcontroller:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of a microcontroller board (usually based on AVR or ARM architecture) and a development environment (Arduino IDE) for writing, compiling, and uploading code. Arduino boards are widely used for prototyping and developing interactive electronic projects due to their simplicity and versatility.

Microcontrollers, on the other hand, are small, self-contained computing devices that contain a processor, memory, and input/output peripherals on a single chip. They are used in a wide range of applications, including embedded systems, robotics, consumer electronics, and industrial automation.

Smart Dustbin Concept:

A smart dustbin is a waste collection bin equipped with sensors, microcontrollers, and communication modules that enable real-time monitoring and management of waste. These sensors, typically ultrasonic or infrared, detect the fill level of the dustbin, while the microcontroller processes this information and triggers alerts or actions as necessary.

Integration with Arduino:

Using Arduino, developers can easily prototype and deploy smart dustbin systems. By connecting sensors to Arduino boards and writing code to process sensor data and control output devices (such as LEDs or communication modules), smart dustbins can be customized to suit specific requirements.

Arduino's user-friendly development environment, extensive community support, and vast ecosystem of libraries and components make it an ideal platform for experimenting with smart dustbin concepts and implementing innovative waste management solutions.

Integration with Microcontroller:

Microcontrollers like the 8051 or PIC series can also be used to implement smart dustbins. These microcontrollers offer greater flexibility and customization options, allowing developers to fine-tune the functionality of the smart dustbin according to specific project requirements.

With the ability to interface with a wide range of sensors, communication modules, and other peripherals, microcontrollers provide the necessary computational power and control capabilities to implement advanced features such as data logging, remote monitoring, and autonomous operation in smart dustbin systems.

2) Aim and Objectives:

2.1. Aim:

The aim of this project is to design a system that is the smart dustbin by an ultrasonic sensor and Arduino and microcontroller.

2.2. Objectives:

The objective of a smart dustbin is to enhance waste management practices through the integration of technology. These bins are equipped with sensors, IoT (Internet of Things) technology, and sometimes even AI capabilities. The main objectives include:

Efficient Waste Collection: Smart dustbins can optimize waste collection routes by providing real-time data on bin fill levels. This ensures that collection trucks are dispatched only when bins are nearing capacity, saving time and resources.

Reduced Overflow and Littering: By alerting authorities or waste management companies when bins are reaching full capacity, smart dustbins help prevent overflow and littering, thereby maintaining cleanliness in public spaces.

Resource Optimization: They help in optimizing the allocation of resources by ensuring that waste collection services are provided precisely where and when they are needed, reducing unnecessary trips and fuel consumption.

Data Collection and Analysis: Smart dustbins gather data on waste generation patterns, which can be analyzed to identify trends, improve waste management strategies, and potentially even encourage behavioral changes to reduce waste production.

Environmental Impact: By improving waste management efficiency, smart dustbins contribute to reducing the environmental impact of waste disposal, such as reducing greenhouse gas emissions from unnecessary collection trips and minimizing pollution from overflowing bins.

2.3. Specifications:

Capacity: Specify the capacity of the dustbin, whether it's in liters or cubic meters. This determines how much waste it can hold before needing to be emptied.

Sensor Technology: Describe the type of sensors used to detect the fill level of the dustbin. Common options include ultrasonic sensors, infrared sensors, or weight sensors.

Connectivity: Specify how the dustbin will communicate its fill level data. This could be through Wi-Fi, cellular network, or other IoT connectivity options.

Data Transmission Frequency: Determine how often the dustbin will transmit data about its fill level to the central monitoring system. This could be in real-time or at regular intervals.

Power Source: Specify the power source for the smart dustbin. It could be battery-powered, solar-powered, or connected to an external power source.

Alert Mechanism: Describe how alerts will be generated when the dustbin reaches a certain fill level threshold. This could be through email notifications, SMS alerts, or a dashboard interface.

Data Analytics: Outline how the data collected from the smart dustbins will be analyzed to identify waste generation patterns and optimize collection routes.

User Interface: If applicable, describe any user interface elements such as touchscreens or mobile apps that allow users to interact with the smart dustbin system.

Security Measures: Specify any security measures implemented to protect the data transmitted by the smart dustbins, such as encryption protocols or access controls.

Environmental Considerations: Consider any environmental factors, such as weatherproofing or durability, that may affect the performance and lifespan of the smart dustbin system.

2.4. Deliverable:

The project outcome will be as follows:

Efficient Waste Management

Reduced Overflow and Littering

Improved Resource Allocation:

Enhanced Public Awareness

Technological Innovation

Cost Savings

Environmental Impact

3) Technical Background

3.1. Hardware:

1. Bread Board
2. Ultrasonic sensor
3. Battery
4. Servo Motor
5. Wires

3.2. Software:

1. Arduino Ide
2. Keil u vision

4) Technical Approach:

Hardware Setup:

Arduino Board: Choose an Arduino board suitable for your project, such as Arduino Uno, Arduino Nano, or Arduino Mega.

Ultrasonic Sensor: Use an ultrasonic sensor to measure the distance from the top of the dustbin to the surface of the waste.

Indicator LEDs: Optional LEDs to indicate the fill level of the dustbin locally.

Power Supply: Provide a power source for the Arduino board and other components, which can be a battery or a DC power adapter.

Sensor Integration:

Connect the ultrasonic sensor to the Arduino board according to its specifications.

Write Arduino code to read the distance measured by the ultrasonic sensor, which correlates to the fill level of the dustbin.

Algorithm Development:

Implement algorithms to manage the data received from the dustbin.

Set up thresholds for fill levels to trigger alerts or notifications.

Optimize waste collection routes based on real-time fill level data to minimize resource usage.

Power Management:

Implement power-saving techniques to conserve energy, especially if using battery power.

Consider using sleep modes to reduce power consumption when the dustbin is not in use.

Testing and Calibration:

Test the system thoroughly to ensure accurate fill level detection and reliable data transmission.

Calibrate the sensors if necessary to account for environmental factors or variations in waste density.

Deployment and Maintenance:

Install the smart dustbin in the desired location and ensure proper connectivity to the communication network.

Monitor the system periodically for any issues and perform maintenance as needed.

8051 Microcontroller:

Features:

Architecture: The 8051 microcontroller architecture is based on the Harvard architecture, which separates program and data memory. It has an 8-bit CPU core with a single accumulator and several general-purpose registers.

8-Bit CPU: The CPU of the 8051 is an 8-bit processor, meaning it processes data in 8-bit chunks. This makes it suitable for handling small-scale tasks efficiently.

Instruction Set: The 8051 instruction set is composed of a relatively small number of instructions, making it easy to learn and program. Instructions are typically one or two bytes long.

Memory: It has separate address spaces for program memory (ROM) and data memory (RAM). The original 8051 microcontroller had 4 KB of on-chip ROM and 128 bytes of RAM, although variations with expanded memory sizes are available.

I/O Ports: The 8051 microcontroller typically has four I/O ports, labeled P0, P1, P2, and P3, each consisting of 8 bits. These ports can be used for interfacing with external devices such as sensors, actuators, and communication modules.

Timers/Counters: The 8051 usually includes two or more timers/counters, which can be used for tasks such as generating precise time delays, measuring external events, or generating PWM signals.

Serial Communication: It supports serial communication protocols such as UART (Universal Asynchronous Receiver/Transmitter) for asynchronous serial communication and SPI (Serial Peripheral Interface) or I2C (Inter-Integrated Circuit) for synchronous serial communication.

Interrupts: The 8051 microcontroller supports both hardware and software interrupts. It typically has multiple interrupt sources, allowing external events to interrupt the normal program execution flow.

Clock and Power Management: It operates at various clock frequencies, typically ranging from a few kilohertz to tens of megahertz, depending on the specific model and application requirements. It also features power-saving modes to reduce power consumption during idle periods.

Development Tools: A wide range of development tools, including assemblers, compilers, simulators, and in-circuit emulators, are available for programming and debugging 8051-based systems.

Arduino IDE:

Code Editor: The Arduino IDE provides a simple yet powerful code editor where users can write and edit Arduino sketches (programs). It supports syntax highlighting, auto-indentation, and code completion, making it easier to write and manage code.

Sketch Management: Arduino sketches are the programs written to control the behavior of Arduino boards. The IDE allows users to create, open, save, and organize sketches in a project folder structure.

Integrated Libraries: The Arduino IDE comes with a collection of built-in libraries that provide pre-written code for common tasks, such as interfacing with sensors, displays, communication modules, and more. Users can easily include these libraries in their sketches to extend functionality without writing code from scratch.

Serial Monitor: The IDE includes a serial monitor tool that allows users to communicate with Arduino boards via the serial port. It displays incoming and outgoing data sent between the Arduino board and the computer, making it useful for debugging and monitoring sensor readings, output values, and other data.

Board Manager: Arduino supports a wide range of microcontroller boards, each with its own specifications and capabilities. The IDE includes a board manager tool that simplifies the process of installing board support packages (cores) for different Arduino-compatible boards. Users can select their board type from a list and configure settings such as the processor, clock frequency, and port.

Code Compilation and Upload: The Arduino IDE compiles Arduino sketches into machine code (binary) that can be understood by the microcontroller on the Arduino board. It also provides a one-click upload feature that allows users to upload compiled code to the target board via USB or other communication interfaces.

Error Checking and Debugging: The IDE performs basic syntax checking and error detection while users are writing code, highlighting syntax errors, typos, and other issues in real-time. It also provides error messages and hints to help users troubleshoot and fix code problems.

Version Control Integration: The Arduino IDE integrates with version control systems like Git, allowing users to manage and track changes to their Arduino projects using version control features such as commit, push, pull, and branch.

Cross-Platform Compatibility: The Arduino IDE is available for multiple operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of users regardless of their preferred platform.

Ultrasonic sensors:

Features:

Ultrasonic Waves: Ultrasonic sensors emit high-frequency sound waves (typically above 20 kHz) and then measure the time it takes for the waves to bounce back after hitting an object.

Transmitter and Receiver: Ultrasonic sensors consist of both a transmitter and a receiver. The transmitter emits ultrasonic pulses, while the receiver detects the reflected waves.

Distance Measurement: Ultrasonic sensors are commonly used for non-contact distance measurement. By measuring the time delay between the emission and reception of ultrasonic pulses, the sensor can calculate the distance to the object.

Accuracy: Ultrasonic sensors can provide accurate distance measurements, typically with resolutions down to millimeters, depending on the sensor's specifications and application requirements.

Range: Ultrasonic sensors can detect objects at varying distances, ranging from a few centimeters to several meters, depending on the sensor's design and capabilities.

Robustness: Ultrasonic sensors are robust and can operate effectively in various environmental conditions, including dust, humidity, and ambient light.

Versatility: Ultrasonic sensors can detect a wide range of materials, including solid objects, liquids, and powders, making them suitable for diverse applications.

Cost-Effectiveness: Compared to other distance sensing technologies like laser-based sensors, ultrasonic sensors are often more cost-effective, making them a popular choice for many applications.

LED FEATURES:

Efficiency: LEDs are highly energy-efficient compared to traditional incandescent and fluorescent lights. They convert a higher percentage of electrical energy into light, resulting in lower power consumption and reduced energy costs.

Longevity: LEDs have a much longer lifespan than traditional lighting sources. They can last tens of thousands of hours, which translates to several years of continuous operation under normal usage conditions.

Instantaneous On/Off: LEDs illuminate instantly when powered on and do not require warm-up time like some other types of lighting. They can also be turned on and off repeatedly without affecting their lifespan or performance.

Compact Size: LEDs are small and compact, allowing for flexible design options and integration into various applications where space is limited.

Color Options: LEDs are available in a wide range of colors, including red, green, blue, yellow, white, and various shades in between. This color versatility enables designers to create custom lighting solutions for different environments and applications.

Color Temperature Control: Some LEDs offer color temperature control, allowing users to adjust the color output from warm (yellowish) to cool (bluish) light. This feature is useful for creating different ambiances or mimicking natural daylight.

Low Heat Emission: LEDs generate significantly less heat compared to traditional lighting sources. This makes them safer to handle and reduces the risk of fire hazards, particularly in enclosed or poorly ventilated spaces.

Directional Light Output: LEDs emit light in a specific direction, resulting in more efficient light distribution compared to omnidirectional sources like incandescent bulbs. This directional characteristic can be advantageous for applications where precise illumination is required.

Durability: LEDs are solid-state devices with no fragile filaments or glass bulbs, making them more resistant to shock, vibration, and environmental factors. This durability makes LEDs suitable for use in rugged environments and outdoor applications.

Dimmability: Many LEDs are dimmable, allowing users to adjust the brightness level to suit their preferences or specific lighting requirements. Dimmable LEDs offer flexibility in creating different atmospheres and saving energy when full brightness is not needed.

Software:

Keil u vision:

Code Editor: µVision provides a comprehensive code editor with features such as syntax highlighting, code folding, auto-indentation, and code completion. It supports various programming languages, including C, C++, and Assembly.

Project Management: µVision allows users to create, manage, and organize projects for embedded software development. Users can add source files, libraries, and configuration files to projects and configure project settings such as compiler options, linker settings, and target device specifications.

Compiler and Debugger Integration: µVision integrates the Keil C/C++ Compiler and Debugger, providing a seamless development experience. Users can compile their code directly within the IDE and debug it using the built-in debugger with features like step-by-step execution, breakpoints, watch variables, and memory inspection.

Simulator: µVision includes a built-in simulator that allows users to simulate their embedded software without the need for physical hardware. The simulator accurately models the behavior of the target microcontroller or processor, enabling developers to test and debug their code in a virtual environment.

Device Database: µVision includes a comprehensive device database with support for a wide range of microcontrollers and processors from various manufacturers, including ARM, STMicroelectronics, NXP, Texas Instruments, and others. Users can select their target device from the database and access device-specific peripherals and registers directly from within the IDE.

Peripheral Configuration: µVision provides tools for configuring and interfacing with device peripherals such as GPIO pins, timers, UARTs, SPI, I2C, ADC, DAC, and more. Users can configure peripheral settings and generate initialization code automatically, saving time and effort in device setup.

RTOS Support: µVision offers support for real-time operating systems (RTOS) such as CMSIS-RTOS, FreeRTOS, and RTX. Users can create and manage RTOS tasks, semaphores, mutexes, and other synchronization primitives within the IDE.

Version Control Integration: µVision integrates with version control systems like Git, Subversion (SVN), and Mercurial, allowing users to manage and track changes to their projects using version control features such as commit, push, pull, and branch.

Cross-Platform Compatibility: µVision is available for Windows operating systems, making it accessible to a wide range of users regardless of their preferred platform.

Smart dustbin code in Arduino IDE:

```
// Include the Ultrasonic library
#include <Ultrasonic.h>
// Define the pins for the ultrasonic sensor
#define TRIGGER_PIN  9
#define ECHO_PIN     10
// Define the threshold for the fill level (in centimeters)
#define FILL_THRESHOLD  20
// Define the pin for the alert LED
#define ALERT_LED_PIN  13
Ultrasonic ultrasonic(TRIGGER_PIN, ECHO_PIN);
void setup() {
  Serial.begin(9600);
  // Initialize the alert LED pin as an output
  pinMode(ALERT_LED_PIN, OUTPUT);
}
void loop() {
  float distance = ultrasonic.distanceRead();
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  if (distance <= FILL_THRESHOLD) {
    digitalWrite(ALERT_LED_PIN, HIGH);
    Serial.println("Dustbin is full! Please empty it.");
  } else {
    // If the fill level is below the threshold, turn off the alert LED
```

```
    digitalWrite(ALERT_LED_PIN, LOW);
  }
  delay(1000); // Adjust delay as needed
}
```

Smart dustbin code in C:

```
#include <8051.h>
// Define GPIO pins for ultrasonic sensor
#define TRIGGER_PIN P1_0
#define ECHO_PIN P1_1
// Define threshold for fill level (in centimeters)
#define FILL_THRESHOLD 20
// Define LED pin
#define ALERT_LED P1_2
// Function to generate delay in microseconds
void delay_us(unsigned int us) {
    unsigned int i, j;
    for (i = 0; i < us; i++)
        for (j = 0; j < 5; j++);
}
// Function to measure distance using ultrasonic sensor
unsigned int measure_distance() {
    unsigned int distance;
    // Send trigger pulse
    TRIGGER_PIN = 1;
    delay_us(10);
    TRIGGER_PIN = 0;
    // Measure pulse width
    while (!ECHO_PIN);
    TH0 = TL0 = 0;  // Clear timer
    while (ECHO_PIN);
    distance = TL0; // Store pulse width
    // Calculate distance (in centimeters)
    distance = (distance * 0.017); // Speed of sound in air = 343 m/s
```

```
    return distance;
}
// Main function
void main() {
    unsigned int distance;
    // Initialize LED pin as output
    ALERT_LED = 0;
    // Main loop
    while (1) {
        // Measure distance
        distance = measure_distance();
        // Check if fill level is above threshold
        if (distance <= FILL_THRESHOLD) {
            // If fill level is above threshold, turn on alert LED
            ALERT_LED = 1;
        } else {
            // If fill level is below threshold, turn off alert LED
            ALERT_LED = 0;
        }
    }
}
```

Results and Conclusions:

In conclusion, the development and implementation of a smart dustbin offer numerous benefits for waste management systems. By leveraging technologies such as sensors, microcontrollers, and communication networks, smart dustbins can revolutionize traditional waste collection processes. Here's a summary of the key points:

1. Efficiency: Smart dustbins optimize waste collection routes by providing real-time data on fill levels. This reduces unnecessary trips and resource wastage, leading to cost savings and improved operational efficiency.

2. Environmental Impact: By reducing overflow and littering, smart dustbins help maintain cleanliness in public spaces and minimize environmental pollution. Additionally, optimized waste collection routes contribute to lower carbon emissions and energy consumption.

3. Data-driven Insights: Smart dustbins gather valuable data on waste generation patterns, which can be analyzed to identify trends, optimize collection schedules, and make informed decisions for waste management strategies.

4. User Convenience: With features such as automatic alerts when bins are full and mobile applications for monitoring, smart dustbins provide convenience to both waste management authorities and residents.

5. Technological Integration: The integration of sensors, microcontrollers, and communication networks enables real-time monitoring and control of dustbins, paving the way for a more interconnected and efficient waste management infrastructure.

6. Scalability and Adaptability: Smart dustbins can be deployed in various settings, including urban areas, commercial complexes, and industrial facilities. They can also be tailored to specific requirements and scaled up or down as needed.

7. Public Awareness and Participation: Smart dustbins can raise public awareness about waste management practices and encourage behavioral changes to reduce waste generation and promote recycling.

Overall, smart dustbins represent a promising solution for addressing the challenges of traditional waste management systems. By harnessing the power of technology, they offer a more sustainable, efficient, and environmentally friendly approach to managing waste in modern urban environments.