AUTOMATIC WASTE SORTING MACHINE

(This project report is submitted in partial fulfilment of the requirements for the Master of Science degree in Physics)

DEPARTMENT OF PHYSICS & ASTROPHYSICS, UNIVERSITY OF DELHI -110007

BATCH: 2022-2024

Submitted By: GROUP-4 (FLUXIFY)

ABHIRAM K ([redacted])

POONAM KUMARI ([redacted])

ASIF AYOUB DAR ([redacted])

MANOJ PRAJAPAT ([redacted])

Supervised By:

Prof. Amarjeet Kaur

Prof. Amita Chandra

Prof. S.A. Hashmi

Dr. Govind Dayal Singh

Dr. Varun Kumar

ACKNOWLEDGMENT

We, the members of "FLUXIFY," extend our heartfelt appreciation to all involved in this project. We recognize that no creation, including this automatic waste sorting machine, is accomplished alone. We express our gratitude to every individual who directly or indirectly contributed to its development.

Special thanks to the Department of Physics and Astrophysics, particularly the Advanced Electronics Lab, for granting us the opportunity to gain practical experience in projects like this.
We owe a debt of gratitude to our esteemed professors, Prof. Amarjeet Kaur , Prof. Amita Chandra, Prof. S.A. Hashmi & Prof. Govind Dayal Singh whose guidance and support helped us shape our ideas into reality.

We also extend our thanks to Mr. Satyendra Saxena & Mr. Mukund Kispotta from Physics Workshop & Mr. Vyas Dev & Mr. Ramchandra & Mr. Shekhar Joshi, the laboratory assistants for their generous assistance throughout this endeavour. Lastly, we appreciate the valuable suggestions, support and enjoyable moments shared with our fellow lab mates.

CERTIFICATE

This document confirms that the project titled "Automatic Waste Sorting Machine," compiled by the members of the "FLUXIFY" group, who are students of the Advanced Electronics Lab at the Department of Physics and Astrophysics, University of Delhi, Group No. 4, has been successfully completed under our supervision and guidance.

We hereby certify that Group 4 (FLUXIFY) is a genuine group of M.Sc. Advanced Electronics Lab, final year students (2022-2024), at the Department of Physics and Astrophysics, University of Delhi. We assert that the conclusions drawn and work done are an outcome of our work. The work has not been submitted to this or any other university/institution for any certificate/degree. Their work reflects original research conducted independently through field and personal investigations.

Submission Date: 3 May, 2024

Group 4 (FLUXIFY)

Department of Physics and Astrophysics

University of Delhi

New Delhi - 110007

TABLE OF CONTENTS:

ABSTRACT:

As urbanization and population growth rise globally, managing municipal solid waste becomes increasingly challenging. In response, we design, and implement an automatic waste sorting machine (AWSM) in this study. Harnessing advanced sensor technology, the AWSM efficiently sorts various waste types like plastic, glass, and metals.

The system follows a multi-stage sorting process, starting with sensor-based identification i.e. capacitive (detect metallic or non -metallic), inductive (for metal detection) and ultrasonic sensor(measure distance and object material properties). Then, the sensor senses the type of waste and segregate the waste into designated compartments. This machine handles large waste volumes, reducing manual labour and boosting overall waste management efficiency. Its implementation promotes recycling and cuts landfill waste, fostering environmental sustainability. By accurately identifying and separating materials, they enable effective recycling and proper disposal and reducing pollution.

AWSMs are crucial components of modern waste management infrastructure. Their continuous operation and adaptability to varying waste compositions offer efficiency, cost-effectiveness, and environmental benefits.

INTRODUCTION

In recent years, the surging global population and rapid urbanization have triggered an unprecedented rise in municipal solid waste (MSW) production, posing substantial challenges for waste management systems worldwide. Traditional manual sorting and disposal methods are not only labour-intensive but also environmentally unsustainable. To confront these issues, the creation of innovative and automated solutions has become essential. Enter the automatic waste sorting machine (AWSM), a concept poised to revolutionize waste management processes.

The AWSM is a big step forward in waste sorting. It uses different sensors like ultrasonic, capacitive, and inductive ones to figure out what kind of waste is – like plastic, glass, or metal. Then, it uses gates and pipes servo motors to move the waste into different sections based on what it is. This makes sorting waste much easier and faster, needing less manual work and reducing mistakes. Plus, it helps separate stuff that can be recycled from stuff that can't, which is good for saving resources and following a circular economy approach.

The implementation of AWSM offering numerous advantages including:

1.Environmental Preservation: By encouraging recycling and minimizing waste sent to landfills, the AWSM aids in conserving natural resources, curbing greenhouse gas emissions, and combating environmental pollution.

2.Sorting of waste. : It helps us to differentiate between the type of waste it is like plastic, glass, and metal. That also helps us to differentiate between recyclable waste or non-recyclable waste.

3.Cost-effectiveness: Despite initial investment costs, the long-term savings from reduced labour and increased recycling rates make the AWSM a financially sound solution for waste management facilities.

4.Scalability: The modular AWSM design allows for scalability and customization to suit the unique needs of various waste management facilities.

METHODOLOGY:

The goal of the automatic waste sorting machine (AWSM) is to sort waste efficiently. We'll pick sensors like capacitive, inductive proximity, and ultrasonic ones, along with two servo motors, to move the waste into different compartments. Then, we will use Arduino to control it all. We'll also figure out the best way to arrange everything to save space The goal and make sure it works well. Integrate Sensors:

Incorporate various sensors such as ultrasonic, Capacitive, and inductive proximity sensor into the AWSM for waste identification and classification. Adjust the ultrasonic sensor on the top of machine and capacitive and inductive proximity sensor in the moving chair to sense the waste type and fine-tune sensor parameters for accurate and consistent detection of different waste materials. Implement algorithms for signal processing to interpret sensor data and make sorting decisions in real-time.

Arduino coding:

After we connect the sensors, we use Arduino IDE to write code that can tell apart glass, metal, and plastic waste. This code makes the gate servo motor and pipe servo motor move, sorting the waste into different bins.

Control System Integration:

Set up a main control system to run everything in the AWSM, like sensors, servo motors, and Arduino IDE. Add safety features and backup plans to stop accidents and keep the AWSM working well.

Testing and Validation:

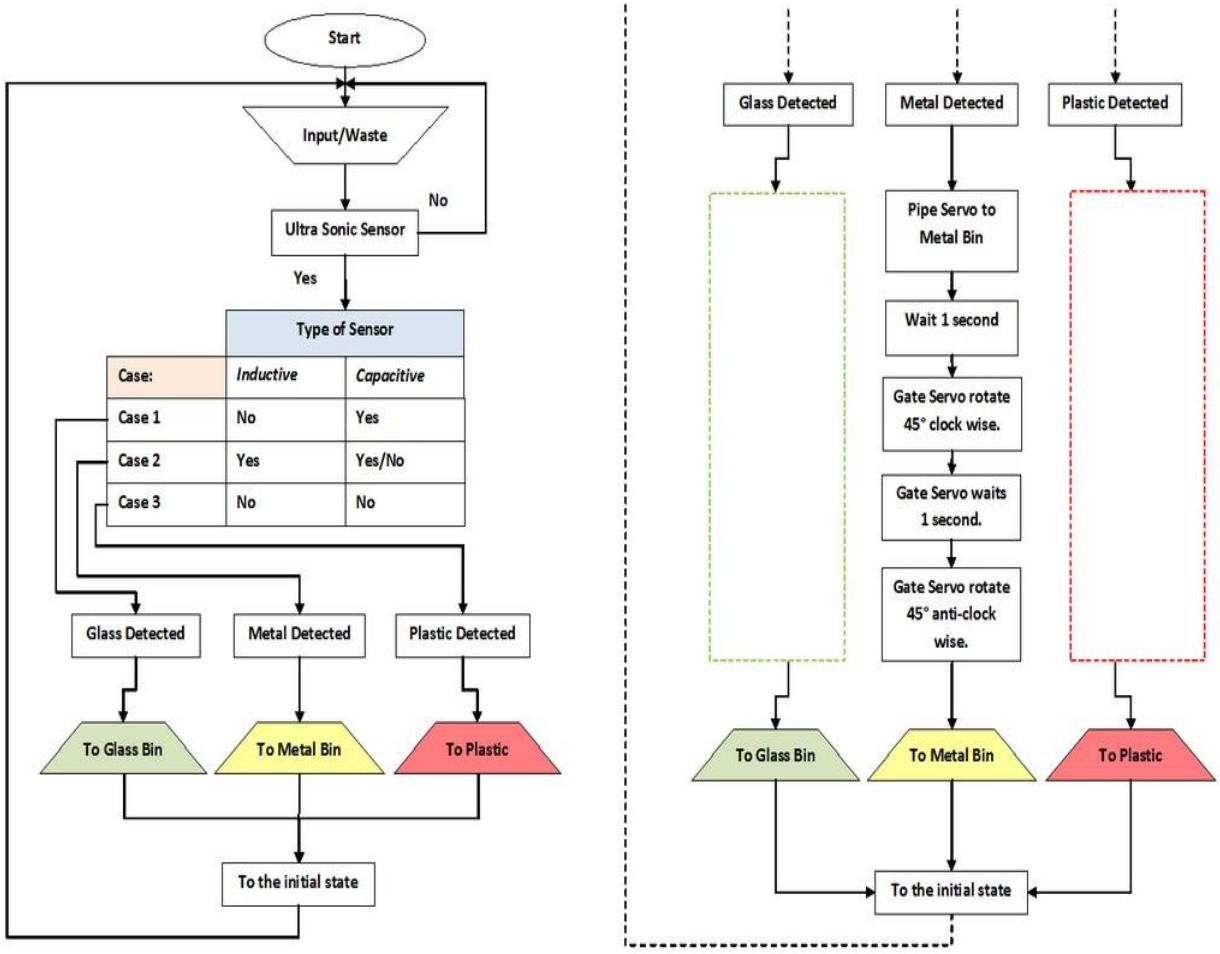
Conduct thorough testing and validation to evaluate AWSM performance under various conditions. Assess sorting accuracy, processing speed, and system reliability to identify areas for improvement. Validate AWSM effectiveness in real-world waste management scenarios, considering factors like waste variability and environmental factors.

Optimization and Iterative Improvement:

Gather feedback from testing and validation to identify optimization opportunities. Iteratively refine sensor settings, and mechanical components to enhance sorting accuracy, efficiency, and reliability. Continuously monitor AWSM performance and implement updates or modifications as needed to ensure optimal functionality.

By following this approach, the development and deployment of an automatic waste sorting machine can be systematically managed.

BLOCK DIAGRAM:



Working:

Waste Intake:

Incoming waste is introduced to the sorting machine via a V-shaped basket.

Material Detection:

An array of sensors, including ultrasonic, capacitive, and inductive sensors, detect and categorize various types of waste materials as they traverse the machine. These sensors transmit signals to the control system, providing detailed information about the waste's properties.

Sorting Decision:

Drawing from the sensor data, the control system determines sorting actions. During this process, the inductive sensor identifies metals, the capacitive sensor gauges material dielectric properties, and the ultrasonic sensor confirms material presence. Sorting occurs based on the comparative values obtained from capacitive or inductive readings.

Sorting:

Upon making a sorting decision, if metal is detected, servo motors reposition the chute to direct the waste into the metal waste bin. Similarly, glass and plastic are sorted into their respective bins.

Waste Disposal:

Sorted waste is directed into designated bins according to its classification.

Continuous Operation:

The automatic waste sorting machine operates non-stop, handling a constant flow of waste materials. The control system continually monitors and adjusts the sorting process in real-time to ensure efficient and precise sorting.

Maintenance and Monitoring:

Regular maintenance is conducted to uphold smooth machine operation and prevent potential breakdowns. The machine's performance is closely monitored, and any operational issues or malfunctions are promptly addressed to minimize downtime.

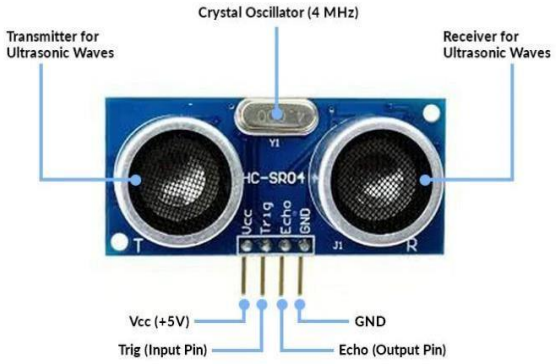
Hardware description:

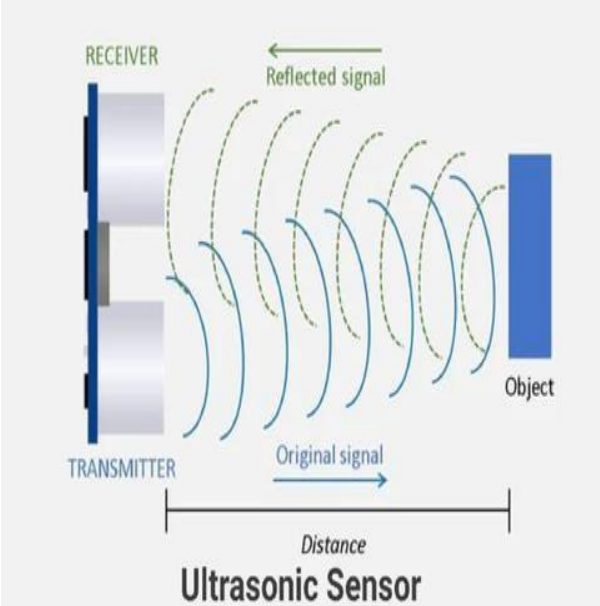
ULTRASONIC SENSOR:

An ultrasonic sensor is an electronic device designed to gauge the distance to a target object by emitting ultrasonic sound waves and converting the reflected sound into an electrical signal. Ultrasonic waves, which travel faster than audible sound, are utilized for this purpose. The sensor operates by emitting sound waves at a frequency beyond human hearing range. Its transducer functions as both a microphone to receive and transmit ultrasonic sound. Unlike some other sensors, ultrasonic sensors use a single transducer to emit a pulse and capture the resulting echo. Distance measurement is achieved by calculating the time elapsed between the emission and reception of the ultrasonic pulse.

WORKING PRINCIPLE OF ULTRASONIC SENSOR

Ultrasonic sensors emit short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they are reflected as echo signals to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo. The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels and when it gets objected by any material it gets reflected toward the sensor this reflected wave is observed by the ultrasonic sensor as shown in the picture below Diagram:





PIN CONFIGURATION OF ULTRASONIC SENSOR

1. Vcc: The Vcc pin powers the sensor, typically with +5V.
2. Trigger: - Trigger pin is an input pin. This pin must be kept high for 10micro sec to initialize measurement by sending ultrasonic signal wave.

3. Echo: - Echo pin is an output pin. This pin goes high for a period which will be equal to the time taken for the ultrasonic wave to return to the sensor.
4. Ground: - This pin is connected to the ground of the system.

How ultrasonic sensor works:

The HC-SR04 sensor is utilized for measuring distances ranging from 2cm to 400cm with an accuracy of 3mm. This sensor module comprises an ultrasonic transmitter, receiver, and control circuitry. The operational principle of the ultrasonic sensor is outlined as follows: - A high-level signal is transmitted for 10 microseconds using the Trigger. The module automatically sends out eight 40 kHz signals and then checks if a pulse is received. Upon receiving a signal, it switches to a high-level state. The duration of this high state represents the time interval between signal transmission and reception.

The distance is calculated using the formula:

Distance = (Time x Speed of Sound in Air (340 m/s)) / 2

The speed of sound in air is known to be approximately 340m/s under standard room conditions. The module's built-in circuitry determines the time taken for the ultrasonic wave to return and activates the echo pin accordingly for the same duration. This enables the calculation of the time taken.

According to the TRD (Time/Rate/Distance) measurement formula, the calculated distance encompasses the round-trip distance travelled by the ultrasonic wave, from the transducer to the object and back. Thus, dividing this distance by 2 yields the actual distance from the transducer to the object. As ultrasonic waves propagate at the speed of sound (approximately 343 m/s at 20°C), the distance between the object and the sensor is half the total distance travelled by the sound wave. The following equation calculates the distance to an object placed in front of an ultrasonic sensor:

Speed of sound is 340m/s v= 0.034cm/s

time = distance/speed t=s/v =

10/0.034 =294 με

distance (s) = t*0.034/2

The module works on the natural phenomenon of ECHO of sound. A pulse is sent for about 10μs to trigger the module. After which the module automatically sends 8 cycles of 40 kHz ultrasound signal and checks its

echo. The signal after striking with an obstacle returns and is captured by the receiver. The echo pin will output the time in microseconds the sound wave travelled.

JUMPER WIRES:

Jumper wires are essentially wires equipped with connector pins at both ends, enabling them to establish connections between two points without requiring soldering Jumper wires are typically made of stranded or solid-core conductive wire, often insulated with PVC or silicone. The connector pins are usually made of metal, such as brass or copper, to ensure good electrical conductivity and durability. They are commonly employed alongside breadboards and other prototyping tools, facilitating convenient modifications to circuits as required. Jumper wires offer a straightforward solution for circuit connectivity, making them an essential and straightforward.

Types of Jumper Wires:

1.Male-to-Male: These jumper wires have male pins on both ends, suitable for connecting two female headers or inserting into breadboard sockets.

2.Male-to-Female: One end of these jumper wires features male pins, while the other end has female . They are commonly used for connecting male headers to components or breadboards.

3.Female-to-Female: Jumper wires with female connectors on both ends are less common but can be useful for specific applications, such as extending connections between components.

Male-to-male jumper wires have pins sticking out that can plug into things, while female ends don't have pins and are used for plugging things in. We'll likely use male-to-male wires the most because they're what we need for connecting two ports on a breadboard

PCB Board:

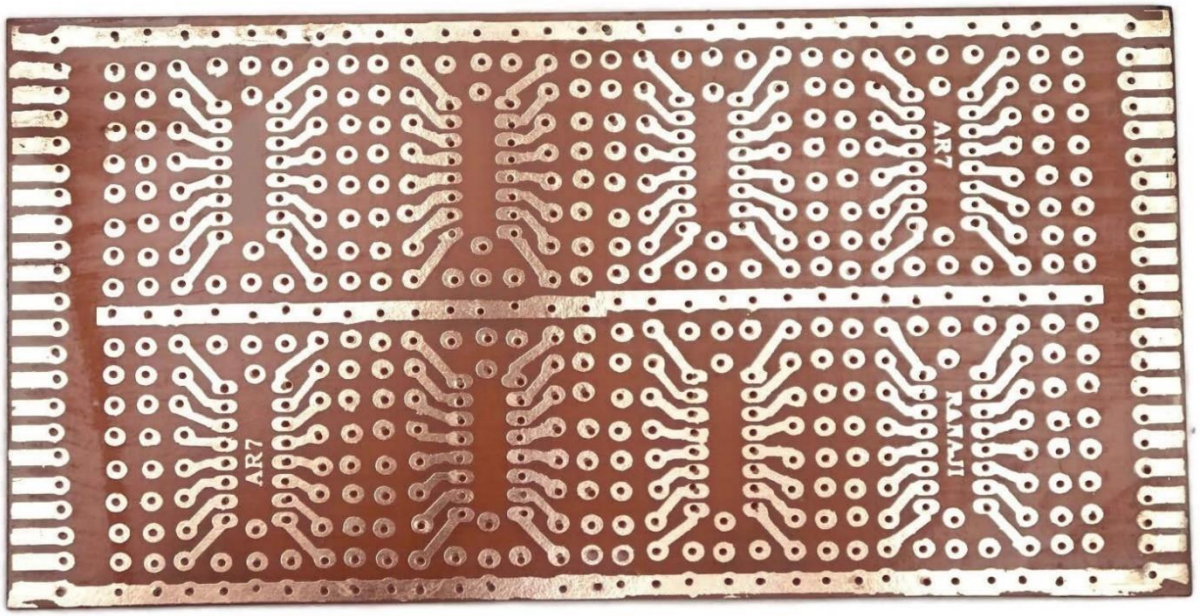
A printed circuit board (PCB) is like a layered sandwich with conducting and non-conducting layers. It's super important in electronics because it helps connect electronic parts, like resistors, capacitors, chips, and connectors, in a tidy and compact way.

PCBs have two complementary functions.

I)First, is to affix electronic components in designated locations on the outer layers by means of soldering.

ii)second, is to provide reliable electrical connections (and reliable open circuits) between the component's terminals in a controlled manner often referred to as PCB design.

Diagram:



Structure and Design:

The base material known as the substrate or board, is typically made of fiberglass reinforced with epoxy resin (FR-4). Other materials such as phenolic resin, polyimide, and ceramic are also used depending on the application. Thin layers of copper foil (for conductive pathway) are laminated onto the substrate on one or both sides of the PCB.

Conductive pathways, known as traces, are etched onto the copper layers to connect the various components attachment points, called pads which are typically circular or rectangular areas of exposed copper where components are soldering. Vias are plated-through holes that connect traces on different layers of the PCB, enabling multilayered designs.

Types of PCBs:

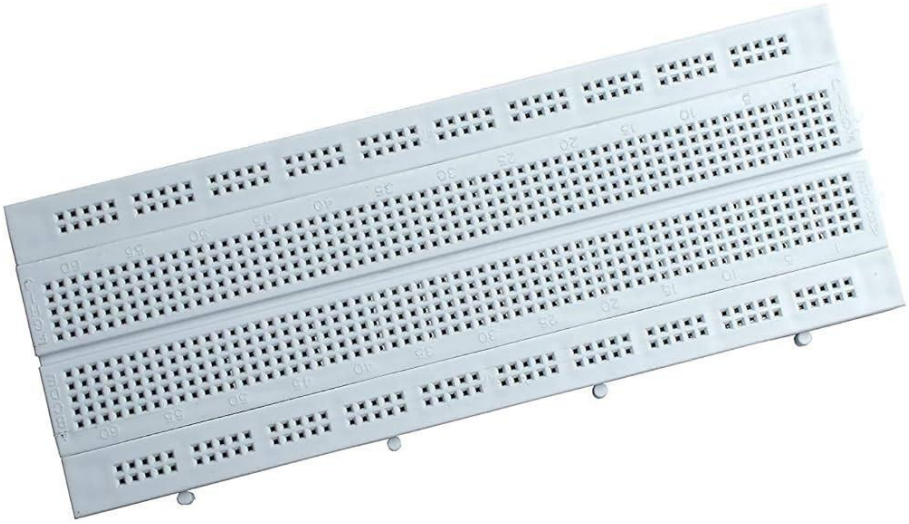
1.Single-Sided PCBs: These PCBs have conductive traces and components on only one side of the substrate.

2.Double-Sided PCBs: Conductive traces and components are present on both sides of the substrate, connected via through-holes or vias.

3.Multilayer PCBs: Multiple layers of conductive traces separated by insulating layers are used to accommodate complex circuits with higher component density and signal integrity requirements.

4.Flexible PCBs (Flex PCBs): These PCBs are made of flexible materials such as polyimide, allowing them to bend or conform to irregular shapes. They are commonly used in applications where space constraints or flexibility are critical.

Breadboard:



A breadboard is a fundamental tool used in electronics prototyping and circuit design. It provides a way to quickly and easily create temporary circuits without the need for soldering. Here is a detailed explanation:

Structure:

Base: The breadboard typically has a plastic base that provides structural support.

Grid of Holes: On top of the base, there's a grid of small holes arranged in rows and columns. These holes are used to insert and connect electronic components.

Metal Clips: Beneath the holes, there are metal clips or springs that create electrical connections when components are inserted.

Components:

Terminals: The holes are usually grouped into rows and columns. Each row typically represents a node of the circuit, while columns are used for connecting components.

Power Rails: There are usually two sets of long rows on the sides of the breadboard called power rails. One rail is often designated for positive voltage (usually labelled as +) and the other for ground (labelled as -).

Distribution Strips: These are sets of short rows or columns used to distribute power or signals throughout the breadboard.

SERVO MOTOR:

A servo motor is a type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. To rotate an object at some specific angles or distance, then use a servo motor. It is made up of a simple motor which runs through a servo mechanism. If motor is powered by a DC power supply, then it is called DC servo motor, and if it is AC then AC servo motor.

Here we are using the DC servo motor. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters planes, Robotics, etc. Servo motors are rated in kg/cm (kilogram per centimetre), this kg/cm tells you how much weight your servo motor can lift at a particular distance.

For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity. The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.

DIAGRAM:

Servo Motor Working Principle: -

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft relates to the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes.

After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse, and a repetition rate.

Servo motor can turn 90 degrees from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds and the length of the pulse will determine how far the motor turns.

For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

Basically, servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of DC motor is converted into torque by Gears. We know that,

WORK= FORCE X DISTANCE

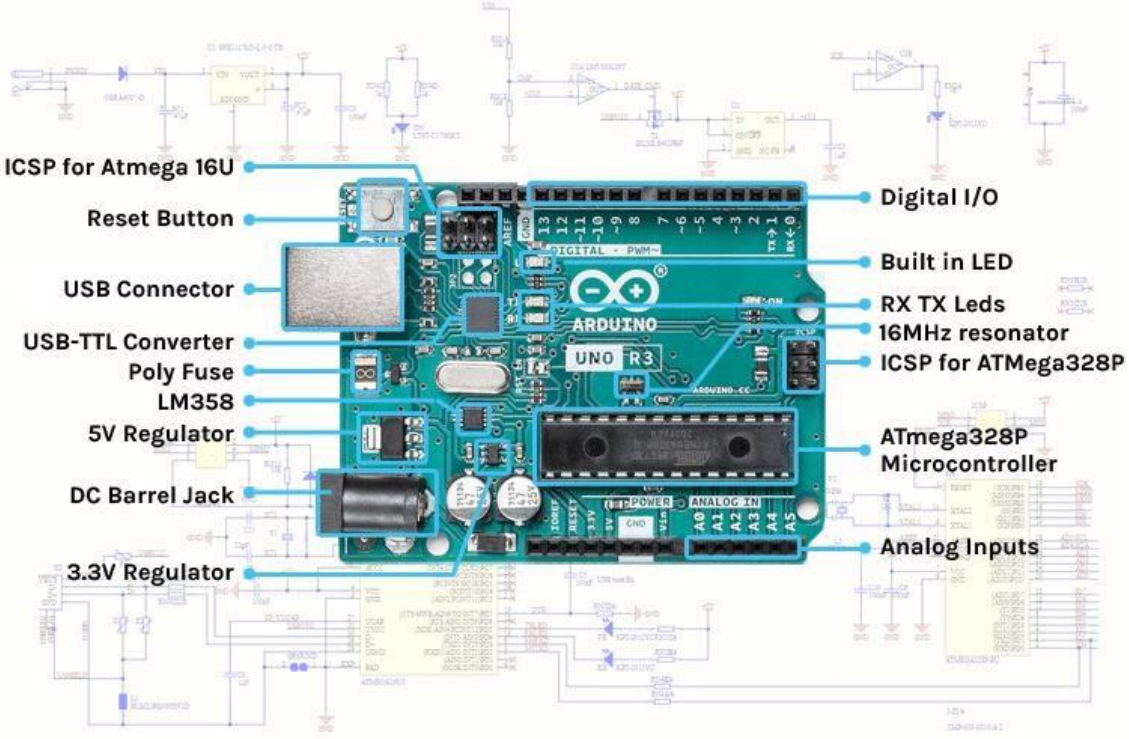
In DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. The potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on the required angle.

Servo motor can be rotated from 0 to 180 degrees, but it can go up to 210 degrees, depending on the manufacturing. This degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. The pulse of 1 millisecond width can rotate the servo to 0 degrees, 1.5ms can rotate to 90 degrees (neutral position) and 2millisecond pulse can rotate it to 180 degree.

At last, servo motor provides precise positioning control, high torque output relative to size, high speeds and have good feedback control.

ARDUINO: -

Arduino is a open-source, so anyone can use and change it. we can use it for all kinds of projects, big or small, especially in places like makerspaces. Inside, there's a microcontroller that can sense and control things in the real world, like lights or motors. It reacts to stuff like buttons or sensors and can make things happen, like turning on LEDs or moving motors. Lots of people like using Arduino because it's so flexible and not too pricey. And because it's open-source, anyone can get creative with it!. Because of its flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects. Here are the components that make up an Arduino board and what each of their functions are.



1. Reset Button - This will restart any code that is loaded to the Arduino board.
2. AREF - Stands for "Analog Reference" and is used to set an external reference voltage for the analog inputs.
3. Ground Pin - There are a few ground pins on the Arduino which is used as the ground reference for the board and connected components.

4. Digital Input/Output - Pins 0-13 can be used for digital input or output. Pins 0 and 1 are also used for serial communication (RX and TX).

5. PWM (Pulse Width Modulation)-The pins marked with the (~) symbol can simulate analog output. Pins 3,5,6,9,10 and 11 support PWM output, which allows for varying the brightness of LEDs, controlling the speed of motors, etc.

6. USB Connection - Used for powering up Arduino and uploading sketches.

7. UART (Universal Asynchronous Receiver /Transmitter- TX/RX used for serial communication with other devices. It Transmit and receive data indication LEDs.

8. ATmega Microcontroller - This is the brains and is where the programs are stored.

9. Power LED Indicator - This LED lights up anytime the board is plugged in a power source.

10. Voltage Regulator - This controls the amount of voltage going into the Arduino board to produce their own board. One of the most popular Arduino boards out there is the Arduino Uno.

11.DC Power Barrel Jack - This is used for powering Arduino with a power supply. here we connect an external power source, such as battery pack or ac adapter.

12. 3.3V Pin - This pin supplies 3.3 volts of power to projects.

13. 5V Pin-This pin supplies 5 volts of power to projects.

14. Ground Pins - There are a few ground pins on the Arduino and they all work the same.

15. Analog Pins - These pins can read the signal from an analog sensor and convert it too digital. These pins can read analog voltage values ranging from 0 to 5 Volts.

16.External Interrupts -pin 2 and 3 can be used as external interrupt pins, allowing the uno to respond to external events.

17.ICSP Header-This header allows to connect to SPI (Serial peripheral Interface) Pins of the microcontroller for programming or for using the uno as an SPI peripheral.

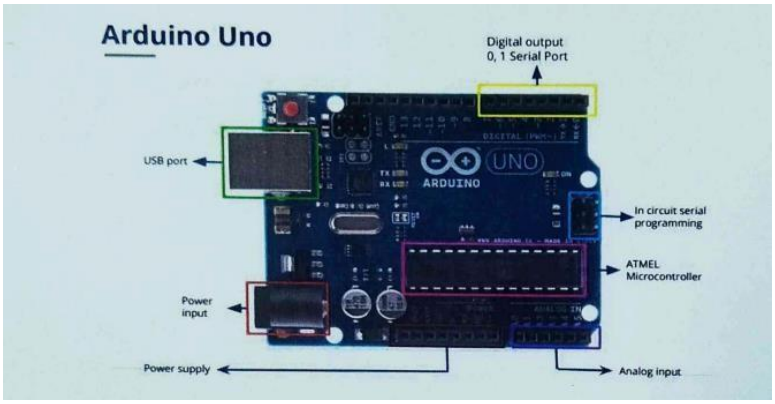
Arduino Power Supply

The Arduino Uno needs a power source for it to operate and can be powered in a variety of ways. connect the board directly to computer via a USB cable. If using a mobile, consider using a 9V battery.

SOFTWARE USED:

ARDUINO IDE- Arduino integrated development environment (IDE):

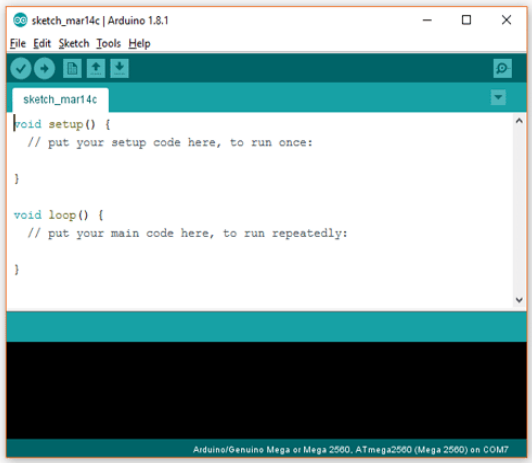
The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

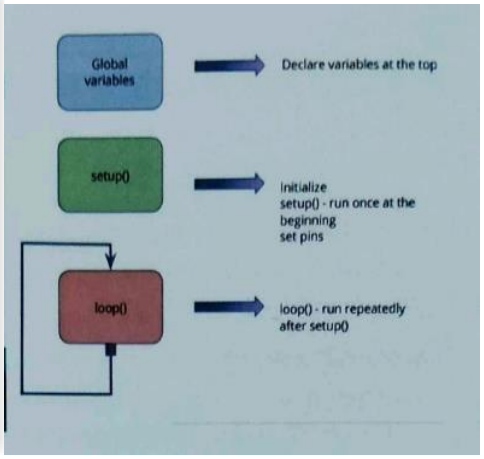


Sketches:

Programs written using Arduino software (IDE) are called sketches. The sketches are written in the text editor and are saved with the file extension. Ino

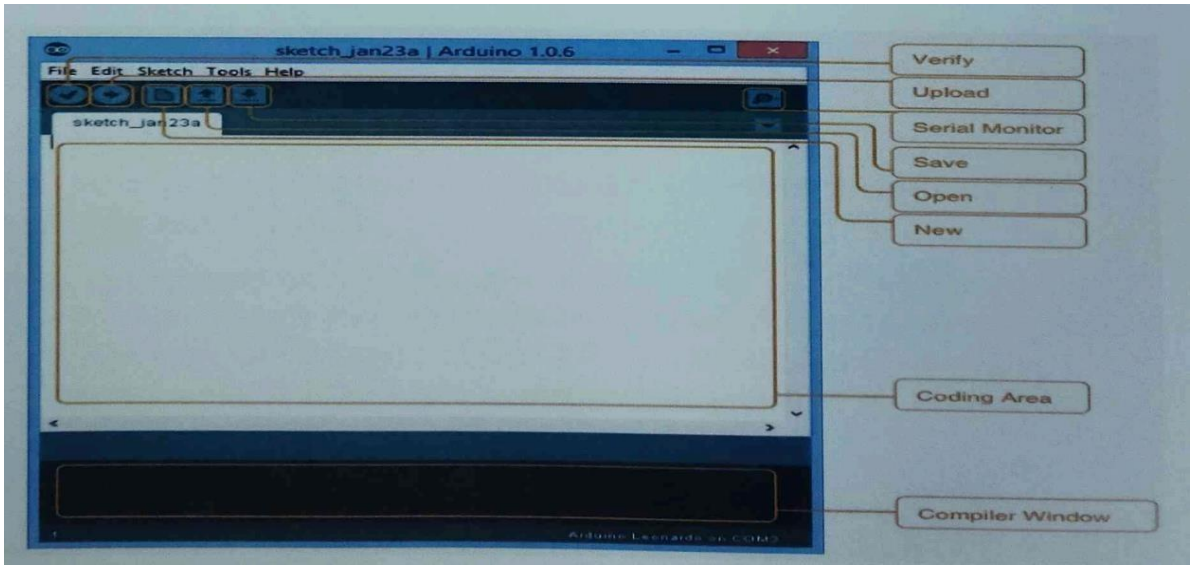
The message area gives feedback while saving and exporting and displays error. The console display text output by the Arduino software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port.





Useful functions:

pinMode()	set pin as input or output
digitalWrite()	set a digital pin high/low
digitalRead()	read a digital pin's state
analogRead()	read an analog pin
analogWrite()	write an "analog" PWM value
delay()	wait an amount of time
millis()	get the current time



Toolbar button: The icons displayed on the toolbar are new, open, save, upload, and verify.

Uploading code to Arduino:

The upload button compiles and runs out code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the upload button present on the toolbar. If the uploading is failed, it will display the message in the error window.

Opening a file:

The open button is used to open the already created file. The selected file will be opened in the current window.

Save:

The save button is used to save the current sketch or code.

New:

It is used to create a new sketch or opens a new window.

Verify:

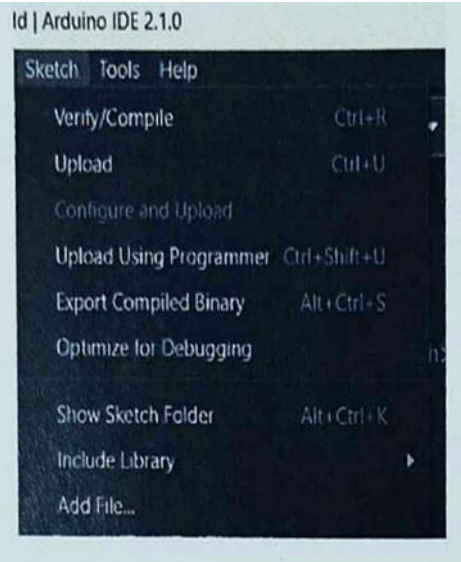
The verify button is used to check the compilation error of the sketch or the written code.

Serial monitor:

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor. When we connect the serial monitor, the board will reset on the operating system. If we want to process the control characters in out sketch, we need to use an external terminal program. The terminal program should be connected to the COM part, which will be assigned when we connect the board to the computer.

Menu bar

Sketch When we click on the sketch button on the menu bar, a drop-down list appears.



Show sketch folder:

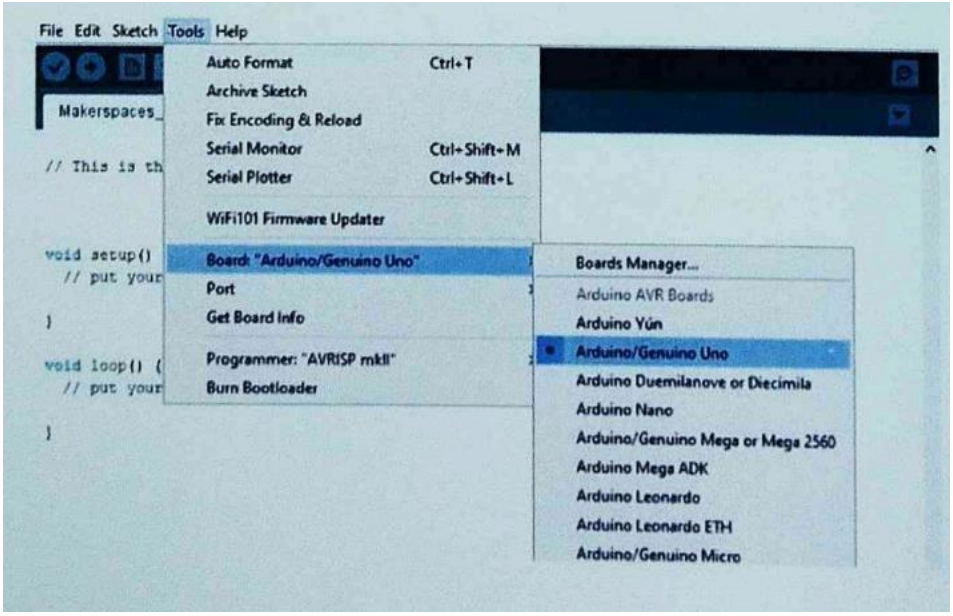
It opens the folder of the current code written or sketch.

Include Library:

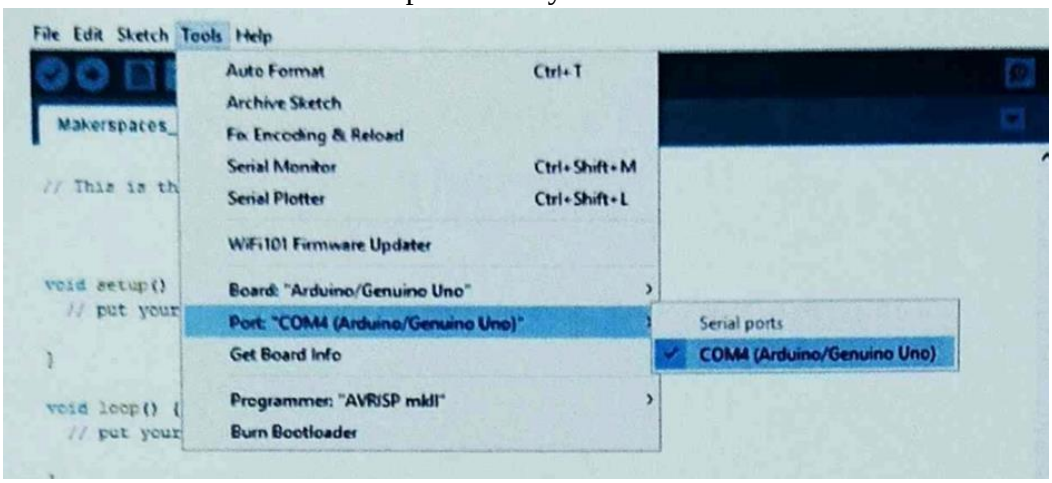
Include library includes various Arduino libraries. the libraries are inserted into code at the beginning of the code starting with the #. we can also import libraries from .zip file.

Connect Your Arduino Uno

At this point, connect Arduino to computer. Plug one end of the USB cable to the Arduino Uno and then the other end of the USB to USB port. Once the board is connected, go to Tools then Board then finally select Arduino Uno .



Next, tell the Arduino which port are using on computer. To select the port, go to Tools then Port then selects the port that says Arduino.



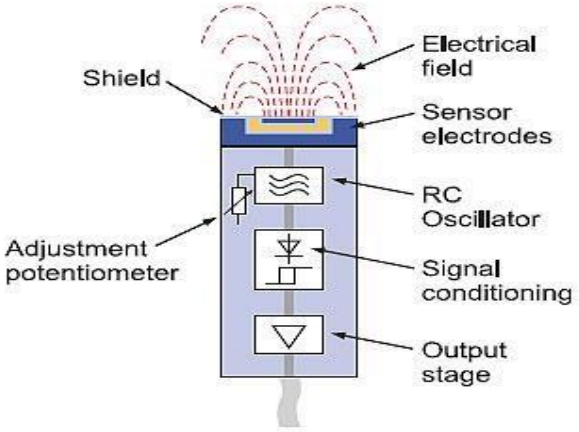
After uploading the required program/code for our project and then the Arduino board reacts according to the compilation of coding.

Capacitive and inductive sensor internal working:

1. Capacitive sensor:

Capacitive sensor: A capacitive sensor is a type of proximity sensor that detects the presence of an object near the sensing electrode that alters the electric field between the electrodes, which in turn changes the capacitance of the sensor. This change in capacitance is detected by the sensor's circuitry, which then generates an output signal indicating the presence or absence of the object.

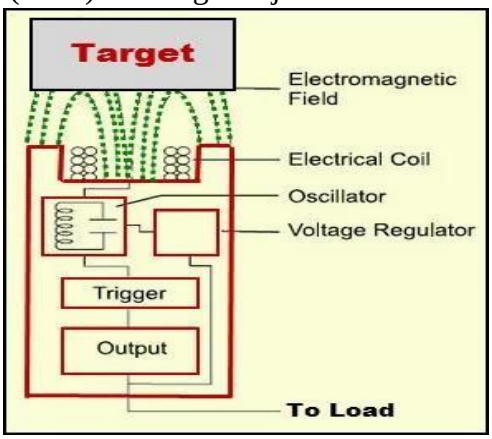
Diagram:



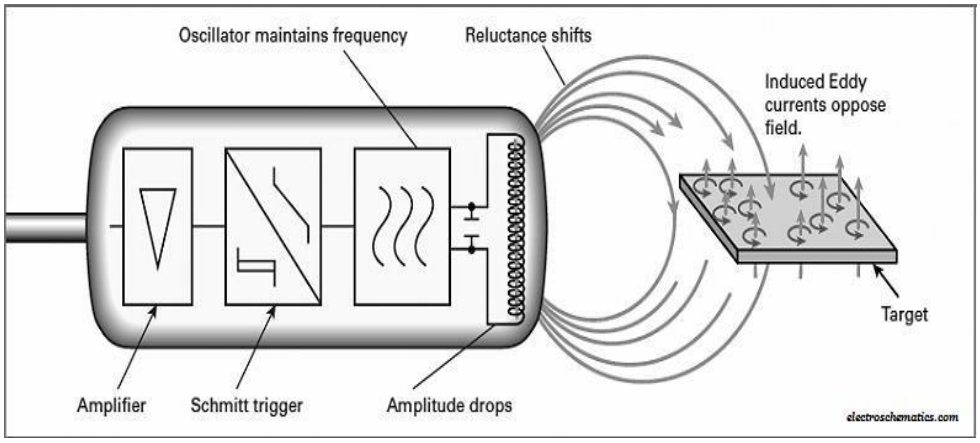
- Sensing Electrode: The primary component responsible for detecting changes in capacitance. Made up of a conductive material, such as metal or conductive polymer. It interacts with nearby objects and measures the capacitance changes caused by their presence.
- Ground Electrode: It serves as a reference point for the capacitance measurement. The ground electrode helps establish the electric field between itself and the sensing electrode.
- Dielectric Material: The dielectric material fills the space between the sensing electrode and the ground electrode. The dielectric constant of the material influences the sensitivity and detection range of the sensor. It affects the capacitance.
  Dielectric materials can be air, plastic, glass, or any non-conductive substance.
- Signal Processing Circuitry: Signal processing circuitry measures changes in capacitance and converts them into usable electrical signals. This circuitry includes amplifiers, oscillators, filters, and analog-to-digital converters (ADCs).It processes capacitance changes and generates digital or analog output signals.
- Interface: The interface provides connections for power, ground, and signal output. It allows the sensor to be connected to external devices, such as microcontrollers or data acquisition systems.
- The interface may include pins, connectors, or solder pads for easy integration into electronic systems.
- Shielding: Shielding layers or materials may be added to reduce interference from external electric fields or noise sources. Shielding improves the sensor's accuracy and reliability, especially in environments with high levels of electromagnetic interference (EMI) or electrical noise.
- Enclosure: Enclosures or packages protect the sensor from environmental factors such as dust, moisture, and physical damage. They provide mechanical support and facilitate mounting or installation of the sensor in various applications.

Applications: Common applications include proximity sensing, touch sensing (e.g., touchscreens and touch buttons), object detection, liquid level sensing, and displacement sensing.

2. Inductive Sensor: This sensor operates under the electrical principle of inductance, where a fluctuating current induces an electromotive force (EMF) in a target object.

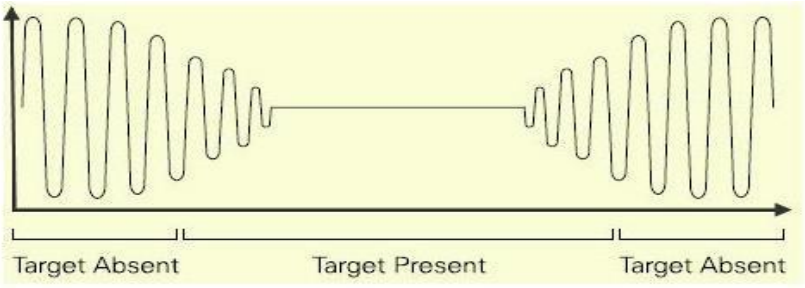


Working Principle



The oscillator creates a symmetrical, oscillating magnetic field that radiates from the ferrite core and coil array at the sensing face. When a ferrous target enters this magnetic field, small, independent electrical currents (eddy currents) are induced in metal's surface. Due to this, load will be caused on the sensor that decreases the electromagnetic field amplitude. If the metal object moves towards the proximity sensor, the eddy current will increase accordingly. Thus, the load on the oscillator will increase, which decreases the field amplitude.

The Schmitt trigger block monitors the amplitude of the oscillator and at a particular level (predetermined level) the trigger circuit switches on or off the sensor. If the metal object or target is moved away from the proximity sensor, then the amplitude of the oscillator will increase.



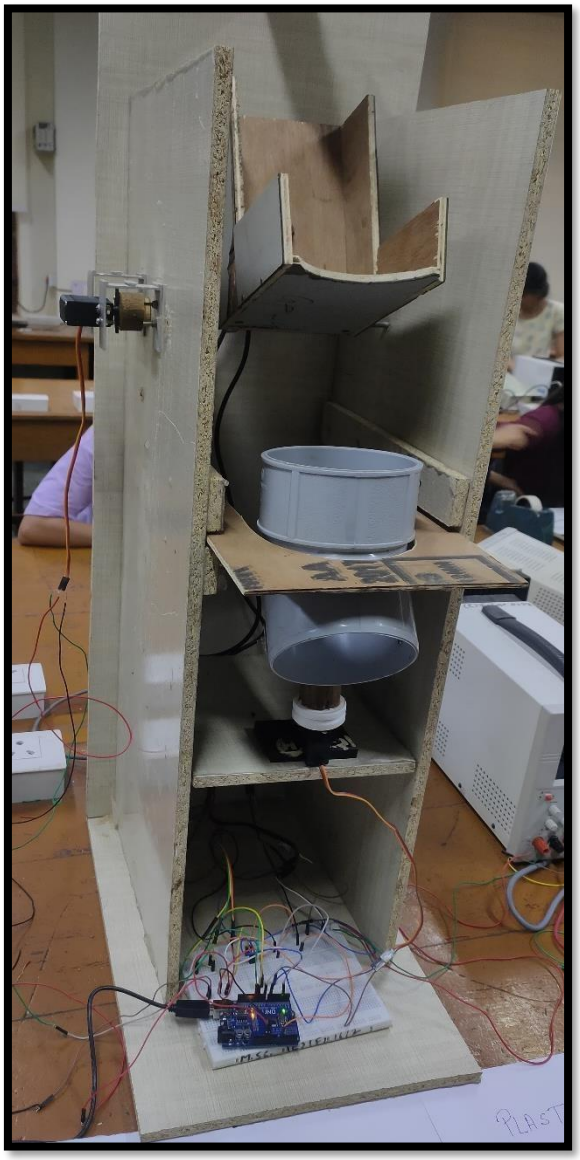
The above image shows the waveform of the inductive proximity sensor oscillator in the presence of the target and in the absence of the target.

PROJECT REPORT:

Aim:

Main aim of this project is to sort minimum three different types of wastes:

Plastic, Metal and Glass.The waste is directed by the pipe into their respective waste bins.



Automatic waste sorting machine

Equipment in this project that we have used:

- One Arduino Uno
- Bread Board
- Soldering material + equipment
- Wires, jump wires

• One Inductive Sensor

• One Capacitive Sensor

• One Ultrasonic Sensor

• Two Servo motors • One Power Supply

Each component as listed below is described, detailing its specification, connections, and method of operation.

Arduino Uno

As a microcontroller board we have used the Arduino UNO which is based on the ATmega328P. The Arduino Uno has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.

Inductive Sensor

Inductive proximity sensors have been used to detect metal objects. We have used a larger sensor, based on our requirement of having a larger sensing range.

Type of sensor

• Inductive Proximity Sensor Switch Model: JG 8/12/18/30 TYPE---JG GI 18-08N1

• Wire Type: DC 3 Wire Type

•Output Type: NPN-NO (Normally Open)

• Detecting Distance: 15mm/ 0.59'

• Supply Voltage: DC 6-36V

•Current Output: 300mA

Capacitive Sensor

Capacitive sensors can sense any object within their sensing range. When the target approaches the face of the sensor, the capacitance increases, resulting in an increase in amplitude of the oscillator. Then the solid-state output switch detects the increase in amplitude and based on that it is turned on or off.

There are different types of Capacitive Sensors. In our project we have used Capacitive Sensors with a compensation adjustment. The capacitive sensor sensitivity depends on the material dielectric constant. Plastic, in fact, has a lower dielectric constant than glass. We have adjusted our sensor in such a way that it can "see through" the objects of plastic nature (to not detect plastic), and glass is detected by the capacitive sensor alone.

Larger sensors have a larger range of detection.

Capacitive sensor specifications

- Product Name: Capacitive Proximity Sensor PCMN-3020-N3-80
- Wire Type: DC 3 Wire Type (Black, Brown, Blue)
- Output Type: NPN NO+NC (Normal Open/ Closed)
- Diameter of Column Sensor: 50mm

Detecting distance: 20mm

- Supply Voltage: DC 10-35V

Current Output: 220mA

Motors

We have used servo motors for both gate and pipe, specifically,

we have used TowerPro MG995 servo motors.

Specifications of the TowerPro MG995 servo motor

- Weight: 55 gm
- Operating voltage: 4.8V~ 7.2V
- Servo Plug: JR
- Stall torque @4.8V : 13kg-cm
- Stall torque @6.6V : 15kg-cm

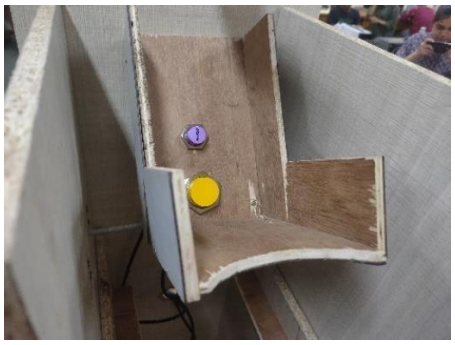
PARTS USED FOR ASSEMBLY:

Step 1

Step2

Step 3

Step 4



Step 5

Arduino Code:

#include <Servo.h> //code includes servo library which allows control of servo motors.

Servo Pipe_Servo; //declares servo motor object name Pipe_ Servo.

Servo Gate_Servo; // declares another servo motor object name Gate_Servo.

// define the variables

int sensorInd = A0; // assigns the analog Pin A0 to variable sensorInd used for analog sensor readings.

int sensorpin = 2; //assigns the digital Pin 2 to the variable sensorpin used for digital sensor readings. int indValue; //declares a variable to store sensor readings.

const int trigPin = 7; //declares a constant integer trigPin and assigns pin 7 to it. This pin is used as the trigger pin for the ultrasonic sensor.

const int echoPin = 8; //declares a constant integer 'echoPin 'and assigns pin 8 to it. This pin is used as the echo pin for the ultrasonic sensor. int i=0; //declares a loop control variable i and initializes it to 0.

float Pipe_Pos=0.0; //declares a floating-point variable 'Pipe_Pos' to store the position of Pipe servo motor.

float Gate_Pos=0.0; //declares a floating-point variable ' Gate_Pos' store the position of Gate servo motor. int n=0; // declares a loop control variable n and initializes it to 0.

long duration; //declares a variable 'duration' of type long to store the duration of the pulse from the ultrasonic sensor.

int distance; //declares a variable 'distance' to store the calculated distance from the ultrasonic sensor.

float metalDetected; // declares floating point variable 'metalDetected ' to store the reading of metal detection.

***//These lines initialize variables and set up pin assignments for sensors and servos used in the code. They establish the groundwork for the functionality of the rest of the code.

void setup() { //set up function

pinMode (trigPin, OUTPUT); // sets the pin mode of the trigPin (pin 7) to OUTPUT mode. This pin is used to trigger the ultrasonic sensor. (trigPin=output)

pinMode (echoPin, INPUT); // sets the pin mode of the echoPin (pin 8) to INPUT mode. This pin receives the echo signal from the ultrasonic sensor. (echoPin = input)

Serial.begin(9600); // This line initializes serial communication with a baud rate of 9600 bits per second. This allows communication between the Arduino board and a connected computer via the USB cable.

// Codes that run only once[

//--System Initialization--

Pipe_Pos=Pipe_Servo.read(); //Read the last value given to Pipe servo

Gate_Pos=Gate_Servo.read(); //Read the last value given to Pipe servo

Serial.println("Motor_PIPE Position"); // This line prints the string "Motor_PIPE Position" to the serial monitor. used as a label to indicate that the following value represents the position of the Pipe servo motor.

Serial.println(Pipe_Pos); // prints the value of the variable Pipe_Pos to the serial monitor. It represents the current position of the Pipe servo motor.

Serial.println("Motor_GATE Position"); // This line prints the string "Motor_GATE Position" to the serial monitor. It is used as a label to indicate that the following value represents the position of the Gate servo motor.

Serial.println(Gate_Pos); // prints the value of the variable Gate_Pos to the serial monitor. It represents the current position of the Gate servo motor.

delay(5000); // introduces a delay of 5000 milliseconds (or 5 seconds) in the program execution before continuing to the next line of code. used for allowing time for sensors to stabilize or for a certain action to complete.

Pipe_Servo.attach(11); // this line attaches the Pipe_Servo object to pin 11 of the Arduino board. The attach() function is used to specify which pin the servo motor is connected to. Once attached, the Arduino board can send signals to control the servo motor's position based on the desired angle.

if (Pipe_Pos<90){ // This line checks if the current position of the Pipe_Servo (stored in the variable Pipe_Pos) is less than 90 degrees. If this condition is true, it means the servo needs to move towards 90 degrees.

for (i=Pipe_Pos; i<=90; i=i+1) { // This line initializes a for loop with the loop control variable i starting from the current position of the Pipe_Servo (Pipe_Pos) and iterating until it reaches 90 degrees. Within the loop, i is incremented by 1 in each iteration.

Pipe_Servo.write(i); // inside the loop, this line writes the value of i to the Pipe_Servo. As i increments from the current position towards 90 degrees, the servo gradually moves to the desired position.

delay (15); // delay of 15 milliseconds in each iteration of the loop. This delay controls the speed at which the servo moves. Smaller delays result in faster movement, while larger delays result in slower movement.

}

} // Once the loop reaches 90 degrees, it exits, and the program continues to the next line of code.

else { // If the condition in the if statement (Pipe_Pos < 90) is false, indicating that the current position of the Pipe_Servo is greater than or equal to 90 degrees, this line initiates an else block.

for (i=Pipe_Pos; i>=90; i=i-1) { // inside the else block, this line initializes another for loop. This time, the loop starts from the current position of the Pipe_Servo (Pipe_Pos) and iterates until it reaches 90 degrees. Within the loop, i is decremented by 1 in each iteration

Pipe_Servo.write(i); // Inside the loop, this line writes the value of i to the Pipe_Servo. As i decrements from the current position towards 90 degrees, the servo gradually moves to the desired position.

delay (15);

}

} // end of the else block.

delay (1000); //Delay (wait) for servo pipe to go to the bin.

//Next rotate the gate servo.

//And control the speed of Gate Servo.

Gate_Servo.attach(10); // attaches the Gate_Servo object to pin 10 of the Arduino board. This line essentially initializes the servo motor connection to the designated pin, allowing the Arduino to control its movements.

for (n=0; n<=45; n=n+1) { // This line initiates a loop that iterates from n being 0 to 45, incrementing n by 1 each time (n=n+1). This loop is used to gradually increase the angle of the servo motor attached to the gate, likely to open it to different degrees.

Gate_Servo.write(n); // this sets the angle of the servo to the current value of n, effectively controlling the position of the gate.

delay(20); //Control the speed of Gate servo

}

delay(1000); //Delay (wait) and then bring back the Gate_Servo to its initial position.So from actual position 2-> n=45; we go backwards up to n=0.

for(n=45; n>=0; n=n-1){ // This line initializes a for loop with the variable n starting at 45. The loop continues if n is greater than or equal to 0, decrementing n by 1 each iteration (n=n-1). Gate_Servo.write(n);

delay(25); //Control the speed of Gate servo

}

delay (5000);

//--System Initialization--//

// Codes that run only once]

}

void loop () { // Code inside this function runs repeatedly as long as the Arduino is powered on. //delay (100); // It was likely commented out to disable it temporarily.

delay (3000); // It causes the program to pause for 3 seconds before continuing to the next instruction.

// Clears the trigPin

digitalWrite (trigPin, LOW); // This line sets the digital output pin trigPin to a LOW state. This is typically done to ensure the pin starts at a known state before sending a trigger signal.

delayMicroseconds (10) ; // this line adds a small delay of 10 microseconds. Microsecond delays are very short and are often used in timing-sensitive operations.

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite (trigPin, HIGH); delayMicroseconds (30) ;

digitalWrite (trigPin, LOW); // This line sets the trigPin back to a LOW state, ending the trigger signal.

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn (echoPin, HIGH); // This line reads the duration of the pulse on the echoPin. It waits for the pin to go HIGH, measures the duration of the pulse in microseconds, and stores this duration in the variable duration. This is typically used in ultrasonic distance sensors to measure the time it takes for a sound wave to travel to an object and back.

// Calculating the distance

distance= duration*0.034/2; // This line calculates the distance based on the duration of the pulse (measured earlier) and the speed of sound. It divides the duration by 2 to account for the round trip of the sound wave. The speed of sound (in air) is approximately 0.034 centimeters per microsecond (or 340 meters per second). Hence, this calculation derives the distance from the object in centimetres.

// Prints the distance on the Serial Monitor

Serial. Print ("Distance: ");

Serial.println(distance);

// These lines print the calculated distance to the Serial Monitor for debugging or monitoring purposes.

indValue=analogRead(sensorInd); //Save value that is read from the analog pin A0 to the variable indValue

delay (10); // small delay of 10 milliseconds. It might be used to stabilize the readings from the analog sensor.

int sensorstate = digitalRead(sensorpin); // This line reads the digital state (HIGH or LOW) from the digital pin sensorpin and stores it in the variable sensorstate. This is likely used to read the state of a digital sensor, possibly a capacitive sensor.

delay(500);

metalDetected = (float)indValue*100/1024.0; // This line converts the analog value indValue to a percentage value (metalDetected) by scaling it from the range 0-1023 (which is the range of values returned by analogRead()') to the range 0-100.

delay (50);

Pipe_Pos=Pipe_Servo.read(); //Read the actual position of Pipe servo

Gate_Pos=Gate_Servo.read(); //Read the actual position of Gate servo

//**METAL DETECTED**/

if(indValue>=250 && sensorstate == 1 && distance<=14) // line begins

an if statement that checks three conditions:

If the value read from the analog sensor (indValue) is greater than or equal to 250.

If the digital state of the sensor (sensorstate) is HIGH (equal to 1). If the distance

calculated (distance) is less than or equal to 14

if(indValue>=250){ //if statement. It checks if the value read from the analog sensor (indValue) is greater than or equal to 250.

Serial.println("Metal Detected"); // line prints "Metal Detected" to the Serial Monitor, indicating that a metal object has been detected.

Pipe_Pos=Pipe_Servo.read(); //Read the last value given to Pipe servo

Gate_Pos=Gate_Servo.read(); //Read the last value given to Pipe servo

Serial.println("Motor_PIPE Position");

Serial.println(Pipe_Pos);

Serial.println("Motor_GATE Position");

Serial.println(Gate_Pos);

Serial.println("Motor to metal bin"); delay (100);

//Wait for some time after the measurements and move servo to the corresponding bin for initial position of servo_gate/PIPE in this case the read function will give us the last position of servo in degree.

//Go to the second position of the PIPE servo @90 degree.To control the speed of the servo we do this for loop.We have 2 cases 1; when pipe is somewhere in position greater then 90, in this particular case we want to go to 90 so we have to loop back.The other case is to loop forward i=i+1, (meaning to increase the angle in incremental manner).

if (Pipe_Pos<90){ // This line begins another if statement, checking if the position of the pipe servo (Pipe_Pos) is less than 90 degrees.

for(i=Pipe_Pos; i<=90; i=i+1){ // if the condition in the if statement is true, this line initiates a for loop. It starts from the current position of the pipe servo (Pipe_Pos) and iterates until the servo reaches 90 degrees, increasing the angle by 1 degree in each iteration Pipe_Servo.write(i); delay(15);

} //inside the loop, this line sets the angle of the pipe servo to the current value of i, effectively moving the servo to the desired position. The delay (15) adds a small delay of 15 milliseconds between each iteration, controlling the speed of the servo movement.

}

else {

for (i=Pipe_Pos; i>=90; i=i-1) { // for loop control variable i with the current position of the pipe servo (Pipe_Pos). The loop continues as long as i is greater than or equal to 90, decrementing i by 1 in each iteration (i=i-1). This loop is used to gradually decrease the angle of the pipe servo from its current position down to 90 degrees.

Pipe_Servo.write(i); // sets the angle of the pipe servo to the current value of i. As the loop progresses, i decreases, thereby decreasing the angle of the servo gradually.

delay (15);

}

}

delay (1000); //Delay (wait) for servo pipe to go to the bin.Next rotate the gate servo,and control the speed of Gate Servo. for (n=0; n<=45; n=n+1) { Gate_Servo.write(n); delay (20);

}

```
delay (1000); //Delay (wait) and then bring back the Gate_Servo to its initial position.
for(n=45; n>=0; n=n-1){
Gate_Servo.write(n); delay(25);
}

//**METAL DETECTED**//
//**PLASTIC DETECTED**//


}
else if(indValue<=250 && sensorstate == 1 && distance<=15){ //  NO  NO state delay(100);
Serial.println("Plastic Detected");
Pipe_Pos=Pipe_Servo.read();          //    Read the last value given to Pipe servo
Gate_Pos=Gate_Servo.read();          //Read the last value given to Pipe servo
Serial.println("Motor_PIPE Position");
Serial.println(Pipe_Pos);
Serial.println("Motor_GATE Position");
Serial.println(Gate_Pos);
Serial.println("Motor to Plastic Bin"); delay(100);
```

//Wait for some time after the measurements and move servo to the corresponding bin. For initial position of servo_gate/PIPE in this case the read function will give us the last position of servo in degree,Go to the second position of the gate Pipe_Servo in degree (145) for Plastic. To control the speed of the servo we do this for loop.We have 2 cases 1; when pipe is somewhere in position greater then 145, in this particular case we want to go to 145 so we have to loop back.The other case is to loop forward i=i+1, (meaning to increase the angle in incremental manner)until we reach the desired angle.

```
if (Pipe_Pos<145){
for(i=Pipe_Pos; i<=145; i=i+1){
Pipe_Servo.write(i); delay(15);
}
}

else {
for(i=Pipe_Pos; i>=145; i=i-1){
Pipe_Servo.write(i); delay(15);
```

}

}

delay (1000); //Delay (wait) for servo pipe to go to the bin.Next rotate the gate servo. This remains the same for all other cases. And control the speed of Gate Servo.

for (n=0; n<=45; n=n+1) {

Gate_Servo.write(n); delay (20);

}

delay (1000); //Delay (wait) and then bring back the Gate_Servo to its initial position. Bring Back Gate_Servo to initial position,So from actual position 2--> n=45; we go backwards up to n=0. for(n=45; n>=0; n=n-1){ //Choose the right angle for Gate servo Gate_Servo.write(n); delay (25); //Control the speed of Gate servo (it can be a little faster now)

}

}

//**PLASTIC DETECTED**// //**GLASS DETECTED**//

else if(indValue<=250 && sensorstate != 1 && distance<=14){

Serial.println("Glass Detected");

Pipe_Pos=Pipe_Servo.read(); //Read the last value given to Pipe servo

Gate_Pos=Gate_Servo.read(); //Read the last value given to Pipe servo

Serial.println("Motor_PIPE Position");

Serial.println(Pipe_Pos);

Serial.println("Motor_GATE Position");

Serial.println(Gate_Pos);

Serial.println("Motor to Glass Bin"); delay (1000);

//Wait for some time after the measurements and move servo to the corresponding bin. For initial position of servo_gate/PIPE in this case the read function will give us the last position of servo in degree, Go to the second position of the Pipe_Servo @25 degree for Glass. To control the speed of the servo we do this for loop. We have 2 cases; when pipe is somewhere in position greater then
90, in this case we want to go to 25 so we must loop back. The other case is to loop forward i=i+1, (meaning to increase the angle in incremental manner). In this case we can have only one case when angle is greater then 25, but however!

if (Pipe_Pos<25) {

```
for (i=Pipe_Pos; i<=25; i=i+1) {
Pipe_Servo.write(i); delay (15);
}
}

else {
for (i=Pipe_Pos; i>=25; i=i-1) {
Pipe_Servo.write(i); delay (15);

}
}
```

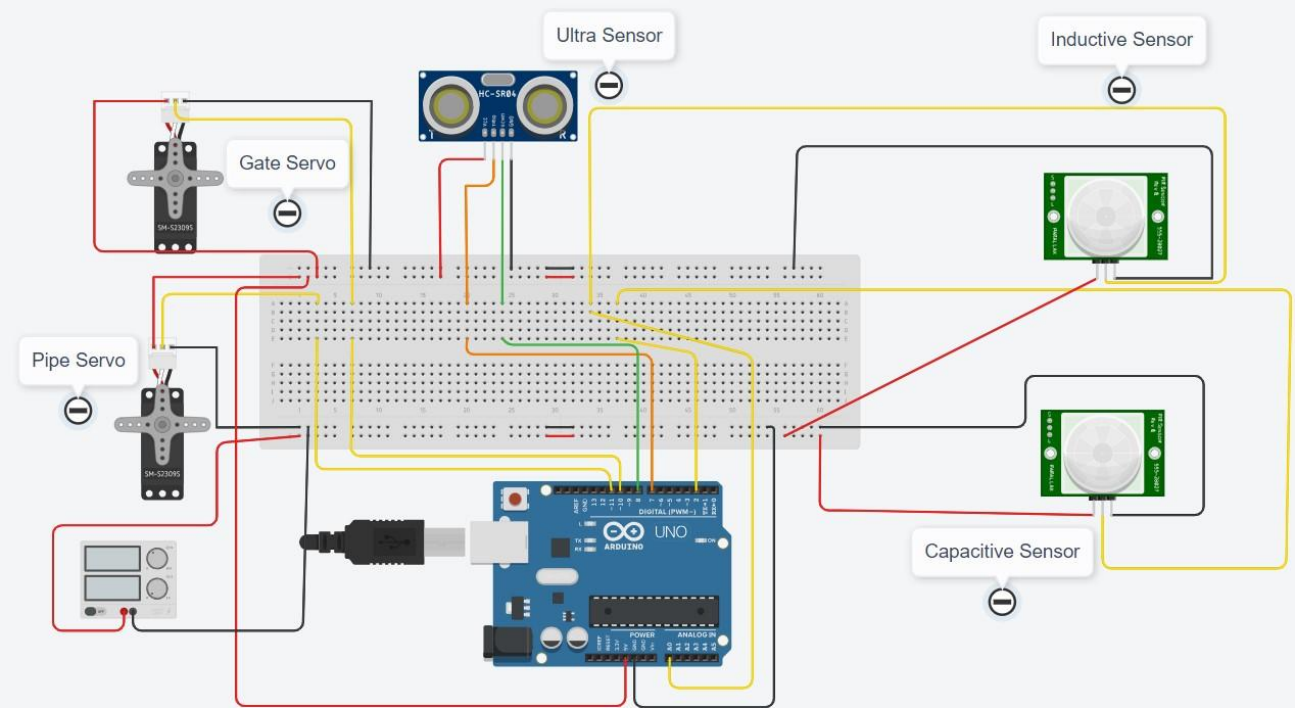
delay (1000); //Delay (wait) for servo pipe to go to the bin. Next rotate the gate servo. This remains the same for all other cases. And control the speed of Gate Servo.

```
for (n=0; n<=45; n=n+1) {
Gate_Servo.write(n); delay (20);
}
```

delay(1000); //Delay (wait) and then bring back the Gate_Servo to its initial position.So from actual position n=45; we go backwards up to n=0.

```
for(n=45; n>=0; n=n-1){
Gate_Servo.write(n); delay(25);
}
Serial.println("Motor_PIPE Position");
Serial.println(Pipe_Pos);
Serial.println("Motor_GATE Position");
Serial.println(Gate_Pos);
}}
```

Connection:



As seen in the figure above, the electronic components have been connected in a bread board using jumper wires. The general schema is the following: We utilized one 6V power supply which drives all the components. With this 6-volt provision, we powered the two servos, inductive sensor, and the capacitive sensor. If we use PCB board, then While soldering one must be careful and connect all the components to the same ground.

Table 1. Pins Connections to Arduino

Component	Connections	Pin
Ultrasonic Sensor	Trigger (output)	7
	Echo (input)	8
Inductive Sensor	Input	A0
Capacitive Sensor	Input	A2
Pipe Servomotor	Output	11
Gate servomotor	Output	10

Operation:

1. System Initialization

At startup, the Gate and Pipe servos will move to their prescribed positions. This procedure will be repeated with each startup or reset. This part of the code executes only once during startup.

Read the current positions of the two Servo motors.

If the Pipe position is not at the zero position:

Move the Pipe servo to the zero position.

Move the Gate servo and then return it to the initial position. (This is intentionally done at every system startup/reset, regardless of whether it is at the zero position or not).

If the Pipe position is at the zero position:

Move the Gate servo and then return it to the initial position. (This is intentionally done at every system startup/reset, regardless of whether it is at the zero position or not).

Initialize sensors.

2. Object and Material Detection

If (Inductive Sensor reading is less than or equal to 250 && Capacitive Sensor reading equals 1 && Ultrasonic Sensor distance is less than or equal to 14 cm):

Plastic Detected: Move to Plastic Bin (step 3)

If (Inductive Sensor reading is less than or equal to 250 && Capacitive Sensor reading does not equal 1 && Ultrasonic Sensor distance is less than or equal to 14 cm):

Glass Detected: Move to Glass Bin (step 3).

If (Inductive Sensor reading is greater than or equal to 250):

Metal Detected: Move to Metal Bin (step 3).

3. Servo Movements

If Plastic Detected:

Move Pipe Servo to Plastic Bin.

Delay.

Move Gate Servo from position 0 to 1.

Delay.

Move Gate Servo from position 1 to 0.

If Glass Detected:

Move Pipe Servo to Glass Bin.

Delay.

Move Gate Servo from position 0 to 1.

Delay.

Move Gate Servo from position 1 to 0.

If Metal Detected:

Move Pipe Servo to Metal Bin.

Delay.

Move Gate Servo from position 0 to 1.

Delay.

Move Gate Servo from position 1 to 0.

RESULT:

With the help of Capacitive and Inductive & Ultrasonic sensors the sorting of Metal, Plastic & Glass using appropriate conditions was Successful.

Conclusion

This project has been done during the academic year 2022-2024 as part of the Advanced Electronics (Lab) course, in Department of Physics & Astrophysics, Delhi University. The requirement was to do an automatic machine to be able to sort minimum two different types of waste. As a team we managed to meet these requirements and developed a machine that sorts three types of waste.

Project budget and progress of our project:

1.Project Budget: The hardware used in this project are

S. No	Name of components	No. of components	Price(Rs)
1.	Arduino UNO	1	250
2.	Capacitive sensor	1	815
3.	Inductive sensor	1	190
4	Ultrasonic sensor	1	60
5.	Jumper wires	3 set	120
6.	Servo motors	2	900
7.	PVC Pipe	1	320
8.	Wooden board	4 parts	workshop
9.	Breadboard	2	lab

Progress of project:

- January: In the last week of the month, discussion of various projects among group members in the lab. then consult with lab teachers and finalize this project 'Automatic waste sorting Machine'.

- February: We learned about online resources and had discussions with group members. We also learned about the equipment used in projects and how to use them, including the Arduino Uno and its programming. After that, we began gathering the equipment needed for our project.

- March: We started working on the workshop to create the parts needed for the automatic waste sorting machine. Then, we connected everything (connection between Arduino, servo motors and sensors) on a breadboard using jumper wires and placed the servo motor and sensors in their respective positions in the machine.

- April: Once all the connections were done, we focused on writing the code to make the machine successfully sort the waste. We also addressed some problems that arose during the testing of our model.

  Then at the end of April month, we have successfully completed our project. But due to some sensitivity problem it is not working smoothly.

Problems faced on doing the projects:

1) Rotating the Pipe: It was highly difficult to make the pipe rotate considering the friction and the region the pipe took for free rotation.

2) Designing a Gate: Designing a gate where the sensor detects the type of waste and directs it to the pipe, while the pipe rotates in the meantime to the respective garbage can was very tricky. We had several ideas from having an Objection plate while the garbage is held on the plate for detection and having a tilting bucket etc. Finally, we went for an idea of Vgate to avoid the movement of sensors and have less mechanical moving parts.

3) Using bearings: The original Idea involved using bearing both for V-gate shaft and Pipe support. The rotating pipe shaft was to be supported by a thrust bearing, eventually we opted out of this idea as the wooden frame constructed for the Pipe was more than sufficient to support it. This saved a lot of time and prevented further any future mechanical complications in the project.

4) Using only one power supply: It was a challenge to restrict ourselves in using only one power supply and having least electronic components as possible. Therefore, in our design phase, we have considered all the possible ways to use only one power supply efficiently.

5) Accuracy of sensors: Sensors are not always accurate, having limitation in their distance of measurement. This affects its accuracy in detection (particularly when the object is small). We intended to modify the gate design by adding a printed part but it effected the performance of the sensor and so for future implications of the system for better detection, shielded sensors can be used.

6) Position of Sensors: There are many factors that came in to the picture while deciding the position and location of our sensors. We considered making a fixed structure for the sensors because we did not want the sensors to be attached to the gate. Also, the distance between the two sensors had to be considered so that the sensors do not influence each other. This has been done by considering the recommended distance in the datasheet and by some trial and error we found the optimal locations to place our sensors.

7) Initialization of the system: In the start-up of the system, we wanted everything to go to the prescribed zero position. However, servos were moving without a command given to them (fast and not as commanded). We "fixed" this by considering the actual position of the motors on the system start-up. Furthermore, one can consider here other ways of solving this issue, for instance, using different servos, other software solutions like using EEPROM; or hardware solutions.

8) Tuning and adjusting different parameters of the system: Tuning and adjusting different parameters of the system, and making everything work together was rather challenging. We opted for values that are optimal, however, the system response can be increased by changing these parameters (ex. speed of

servos, the delays added into the code, etc.). Also, we observed that it will detect more efficiently metals if we only consider the data from the inductive sensor, this can be seen in the codes given in the programming section.

9) Complexity of Materials: Waste materials come in various shapes, sizes, and materials, making it challenging for sorting machines to accurately identify and separate them. Materials like plastics, which come in numerous types and forms, can be particularly difficult to sort effectively.

Future applications:

Our waste sorting machine is only a prototype. With modifications and improvements, a similar concept of our waste sorting machine can be implemented in house hold, offices and industrial applications.

References:

https://www.elprocus.com/atmega328-arduino-uno-board-working-and-itsapplications/https://www.youtube.com/watch?v=gyvJ_8_jWf0 https://robu.in/inductive-proximity-sensor-working-principle/ https://byjus.com/physics/capacitive-sensors/

https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/ https://www.javatpoint.com/arduino-ide

https://robocraze.com/blogs/post/what-is-ultrasonic-sensor