CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

This sub-section will explain the implementation of the program to get the final results of the problems studied by researchers. There are 3 program implementations that are used to build a prototype of the IoT-based Trash Separation System to produce the final results which are used as problem analysis.

First, a few lines of code are used to detect human objects in front of the bin using the HC-SR04 Ultrasonic sensor and servo as opening the trash can lid.

```
1.  int jarakDepan=0;
2.  jarakDepan=sonar2.ping_cm();
3.  Serial.print("Jarak Objek Manusia: ");
4.  Serial.print(jarakDepan);
5.  Serial.println(" Cm");
6.  delay(500);
7.  servoDepan.write(0);
8.  if(jarakDepan>0&&jarakDepan<30) {
9.  Serial.println("Tempat Sampah Terbuka..");
10. servoDepan.write(120);
11. delay(2000);}
```

Based on the code above, it can be explained as follows. In line 1 we create a variable named jarakDepan of type integer with an initial value of 0. Line 2 fills the jarakDepan variable with an ultrasonic sensor value. Line 3 command used to display "Human Object Distance:" on the Arduino serial monitor. Line 4 command to display JarakDepan data on the serial monitor. Line 5 command to display the word "Cm" on the serial monitor. Line 6 commands provide a delay of half a second. Line 7 Command to set servo at default degree 0. Line 8 command if condition when data range is greater than 0 and distance range is less than 40. Line 9 command displays the word "Trash Open" on the serial monitor. Line 10 command to run a servo at 120 degrees. Line 11 code to give a time delay of 2 seconds.

The second few lines of code are used by the Inductive Proximity sensor to detect metal objects in the stopbox mechanism.

```
1.  proximityData = analogRead(proximityPin);
2.  Serial.print("Data Sensor Inductive: ");
3.  Serial.println(proximityData);
4.  servoLogam.write(0);
5.  servoNonlogam.write(0);
6.  if(proximityData==1){
7.  Serial.println("Metal Detected!");
8.  servoLogam.write(90);
9.  delay(3000);}
10. else {
11. Serial.println("Metal Not Detected!");
12. servoNonlogam.write(90);
13. delay(3000);}
```

Based on the code above, it can be explained as follows. Line 1 variable proximity Data is filled by data from the Metal sensor value. Line 2 displays the words "Inductive Data Sensor:" on the serial monitor. Line 3 displays data from proximityData on the serial monitor. Line 4 command to set servoLogam at default degree 0. Line 5 command to set servoNonlogam at default degree 0. Line 6 code for condition if data from proximityData == 1. Line 7 displays the word "Metal Detected!" on the serial monitor. Line 8 command to set the metal servo at 90 degrees rotation. Line 9 code to provide a time delay of 3 seconds. Line 10 conditions other than proximityData == 1 hence, Line 11 displays the word "Metal Not Detected!" on the serial monitor. Line 12 commands for setting ServoNonlogam at 90 degree rotation. Line 13 provides a time delay of 3 seconds.

Third, a few lines of code are used by the HC-SR04 Ultrasonic sensor to send e-mail through the Ethernet Shield.

```
1. int jarak=0;
2. jarak=sonar.ping_cm();
3. Serial.print("Jarak Volume Sampah: ");
4. Serial.print(jarak);
5. Serial.println(" Cm");
```

```
6. delay(500);
7. if(jarak>0&&jarak<10) {
8. while (!Serial){;}
9. Serial.println("Tempat Sampah Penuh..!");
10. Serial.println("Please Wait While Sending an Email");
11. if (!Ethernet.begin(mac)) {
12. Serial.println("Failed to configure Ethernet using
    DHCP");
13. Ethernet.begin(mac, ip);}
14. delay(1000);
15. if(client.connect(server, 80)) {
16. Serial.println("Connected");
17. client.println("GET /index.php?text=whadap HTTP/1.1");
18. client.println("Host:
    theosendemail.000webhostapp.com");
19. client.println("Connection: close");
20. Serial.println("Email Terkirim...");
21. delay (2000);
22. } else { Serial.println("Connection Failed");}
```

Based on the code above, it can be explained as follows. Line 1 creates a distance variable of type integer with a value of 0. Line 2 fills the distance variable with an Ultrasonic sensor value. Line 3 displays the word "Trash Volume Distance:" on the serial monitor. Line 4 displays distance data on the serial monitor. Line 5 displays the word "CM" on the serial monitor. Line 6 provides a delay of 0.5 seconds. Line 7 codes for conditions if distance> 0 and distance <10. Line 8 commands to wait for the serial port to connect. Line 9 displays the word "Trash Can Full ...!" on the serial monitor. Line 10 displays the word "Please Wait While Sending an Email" on the monitor serial. Line 11 code for conditions if the network settings do not support DHCP. Line 12 displays the word "Failed to Configure Ethernet Using DHCP" on the serial monitor. Line 13 configures Ethernet manually via IP and Mac addresses. Line 14 provides a delay of 1 second. Line 15 codes for conditions if the client connects server and port 80. Line 16 displays the word "Connected" on the serial monitor. Line 17 http request client code to access the index.php file. Line 18 client request to access host: theosendemail.000webhostapp.com. Line 19 client request to close the connection. Line 20 displays the word "Sent Email ..." on the serial monitor. Line 21 provides a delay of 2 seconds.

Line 22 conditions other than if then display the word "Connection: Failed" on the serial monitor.

5.2 Testing

In this sub-chapter, researchers conduct testing programs with several datasets. In this test, the Inductive Proximity sensor is tested to get the optimum sensor distance in reading objects. In addition, the Inductive Proximity sensor is also tested to obtain the accuracy of the sensor in reading metal waste that is not fully metal.

For testing the optimum distance of the Inductive Proximity sensor, researchers conducted experiments 6 times with different distances to get the optimum distance sensor results in reading objects. The following is the optimum distance sensor Inductive Proximity sensor testing table.

Table 5.1 Testing Table of Inductive Proximity Sensor Optimum Distance

No	Distance	Results
1	2 Cm	No Sensor Detected
2	1.5 Cm	No Sensor Detected
3	1 Cm	No Sensor Detected
4	0.5 Cm	No Sensor Detected
5	0.3 Cm	Detected by sensor
6	0.1 Cm	Detected by sensor

From the above test it produces the output in the image below, the LED on the sensor lights up red which means that metal objects are detected.

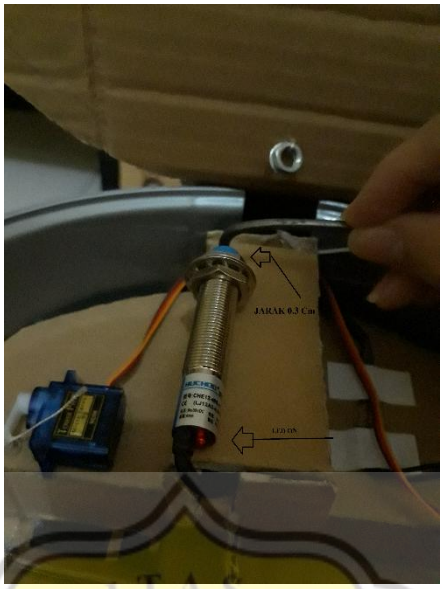

Illustration 5.1 Inductive Proximity Sensor Optimum Distance Output

Then for testing the accuracy of the sensor in reading metal waste that is not fully metal, the researchers conducted experiments 10 times with 5 datasets of metal waste, each dataset has a different metal composition and 5 datasets of non-metal waste. The sensor reading time in the table below is not constant, this is influenced by metal waste that has a different metal composition so that the sensor reading time can be faster or longer. The following is a testing table for the accuracy level of Inductive Proximity sensors in reading metal waste that is not fully metal.

Table 5.2 Testing Table of Inductive Proximity Sensor Accuracy Level

No	Trash	Metal Waste Composition	The Top Side of Metal Trash	The Bottom Side of Metal Trash	The Right Side of Metal Trash	The Left Side of Metal Trash	Time	Results
1	Drink Cans	100 %	Detected by Sensor	Detected by Sensor	Detected by Sensor	Detected by Sensor	3 Seconds	Fully Detected by sensor
2	Spike	100 %	Detected by Sensor	Detected by Sensor	Detected by Sensor	Detected by Sensor	2 Seconds	Fully Detected by sensor

3	Perfume Cans	70 %	Not Detected by the Sensor	Detected by Sensor	Detected by Sensor	Detected by Sensor	6 Seconds	75% Detected by sensor
4	Scissors	50 %	Detected by Sensor	Not Detected by the Sensor	Detected by Sensor	Detected by Sensor	7 Seconds	75% Detected by sensor
5	Used Cable	20 %	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	0 Seconds	Fully Not Detected by the Sensor
6	Paper	0 %	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	0 Seconds	Fully Not Detected by the Sensor
7	Used Cloth	0 %	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	0 Seconds	Fully Not Detected by the Sensor
8	Plastic Bag	0 %	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	0 Seconds	Fully Not Detected by the Sensor
9	Plastic Snacks	0 %	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	0 Seconds	Fully Not Detected by the Sensor
10	Styrofoam	0 %	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	Not Detected by the Sensor	0 Seconds	Fully Not Detected by the Sensor

From the above test it produces the output in the picture below, namely the value of sensor data 1 if the garbage is metal waste and sensor data 0 if the garbage is non-metal waste.

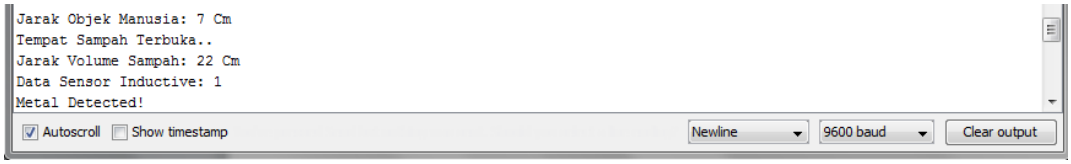

Illustration 5.2 Output Data Sensor 1

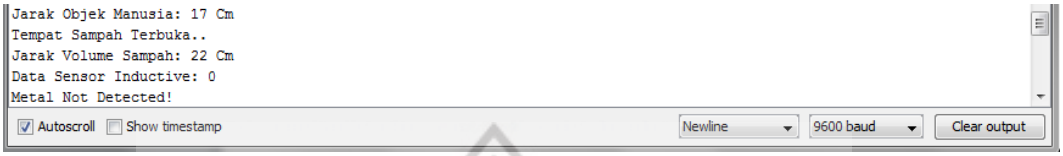

Illustration 5.3 Output Data Sensor 0

Testing the e-mail notification volume of the trash can is full, the system will detect the trash can is full if the trash volume is less than 10 cm. Sending e-mails using the Gmail PHPMailer and SMTP libraries. Arduino only accesses the php file via HTTP Request. Following are the outputs and images when the e-mail was successfully sent.

Illustration 5.4 Arduino Output of Email Sent

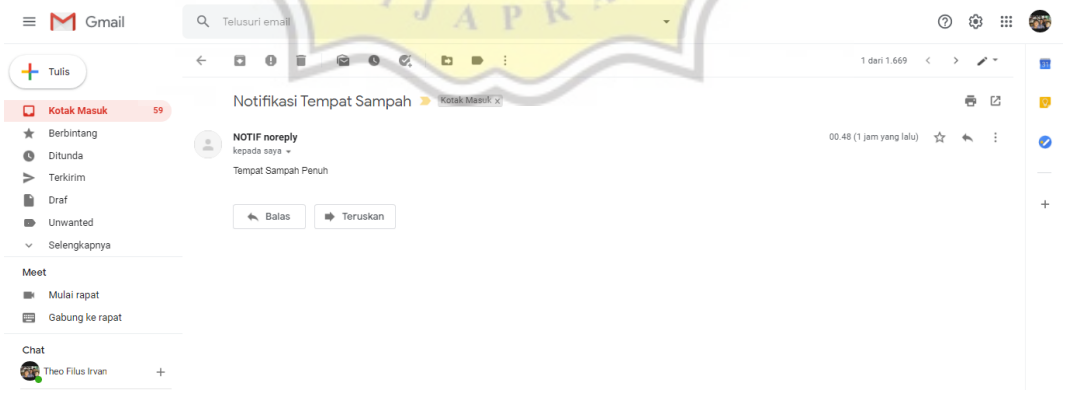

Illustration 5.5 Image of Email Sent to the User

Problem testing response time on Inductive Proximity sensors when sensors read metal junk. Testing is done with 2 scenarios, each scenario has 10x experiments. The first scenario testing the Inductive sensor uses the whole trash system, the second scenario tests the Inductive sensor using only the sensor and Arduino Uno. The following is the response time testing table on the Inductive Proximity sensor.

Table 5.3 Sensor Testing with the System as a whole

No	Response Time Sensor	Detected by sensor
1	12 Seconds	Yes
2	8 Seconds	Yes
3	2 Seconds	Yes
4	1 Seconds	Yes
5	1 Seconds	Yes
6	3 Seconds	Yes
7	2 Seconds	Yes
8	10 Seconds	Yes
9	2 Seconds	Yes
10	1 Seconds	Yes

From the above test, the Inductive Proximity sensor with the trash system as a whole produces an average response time of 4.2 seconds.

Table 5.4 Sensor Testing with Arduino Uno

No	Response Time Sensor	Detected by sensor
1	1 Seconds	Yes
2	7 Seconds	Yes
3	5 Seconds	Yes
4	7 Seconds	Yes
5	6 Seconds	Yes
6	5 Seconds	Yes
7	3 Seconds	Yes
8	2 Seconds	Yes
9	4 Seconds	Yes
10	2 Seconds	Yes

From the above test, the Inductive Proximity sensor using Arduino Uno alone produces an average value of response time of 4.2 seconds.