CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Arduino Code

This program was created using the Arduino programming language, below will explain the program code.

```
1. #include <DHT.h>
2. #include <HCSR04.h>
3. #define DHT_PIN A0
4. #define DHTTYPE DHT11
5. DHT dht(DHT_PIN, DHTTYPE);
6. int Temp;
7. int y;
8. int x;
9. int vib = A5;
10.String life;
11.const int TriggerPin = 2;
12.const int TriggerPinx = 4;
13.const int EchoPin = 6;
14.const int EchoPinx = 5;
15.long duration, distance;
16.long durationx, distancex;
```

Lines 1-2 to add the library to be used, lines 3-5 to set DHT11 reading on pin A0 and determining the type of DHT sensor used, line 10 to set Piezoelectric Vibration Ceramic sensor reading at pin A5, lines 12-15 to set 2 The HC-SR04 reads on pins D2, D4, D6, D5, lines 7-9 and 16-17 are used to add variables.

```
1. void setup() {
2.   Serial.begin(9600);
3.   dht.begin();
4.   pinMode(TriggerPin,OUTPUT);
5.   pinMode(EchoPin,INPUT);
6.   pinMode(TriggerPinx,OUTPUT);
7.   pinMode(EchoPinx,INPUT);
8. }
```

Line 2 to start Serial with 9600 baudrate, line 3 to start sensor

DHT, lines 4-7 to set TriggerPin, TriggerPinx as OUTPUT and EchoPin, EchoPinx as INPUT.

```
1. void loop() {
2.   String req = "";
```

```
3.   while(Serial.available()>0){
4.     req += char(Serial.read());
5.   }
6.   req.trim();
7.   if(req=="Yes"){
8.     sendata();
9.   }
10.        req = "";
11.        delay(1000);
12.      }
```

Lines 3-5 to receive the serial communication process from NodeMCU and make the req variable into a String filled with character data, line 6 to delete the received data space, lines 7-9 when Arduino receives serial data from NodeMCU in the form of "Yes" then Arduino will sending data to NodeMCU.

```
1. if ((Temp <= 30) && (VibFix <= 50) && (x <= 250) && (y <= 150))
2.     {
3.       life=">5Tahun";
4.     }
5.      else if ((Temp <= 30) && (VibFix <= 50) && (x <= 250) && (y >
   150))
6.     {
7.       life=">5Tahun";
8.     }
9. .....
10.      else
11.          {
12.            life="<5Tahun";
13.          }
```

Lines 1-13 make a statement to fill in the value of life, where the constraints used in the sample code are personal assumptions. For application to the machine, calibration needs to be carried out according to the standard machine used.

```
1. String    data    =    String(Temp)+"#"+String(VibFix)+"#"+String(x)
   +"#"+String(y)+"#"+String(life);
2.   Serial.println(data);
```

Line 1 is used to accommodate all sensor data on 1 string variable 'data' because in serial communication each sending process can only send 1 data, line 2 is used to display String data on a serial monitor.

5.1.2 NodeMCU Code

```
1. #include <SoftwareSerial.h>
2. #include <ESP8266WiFi.h>
3. #include <ESP8266HTTPClient.h>
```

Line 1 is used to accommodate all sensor data on 1 string variable 'data' because in serial communication each sending process can only send 1 data, line 2 is used to display String data on a serial monitor.

```
1. SoftwareSerial DataSerial(12,13);
2. unsigned long previousMillis = 0;
3. const long interval = 3000;
4. String arrData[5];
```

Line 1 digunakan untuk membuat variable Software serial (Rx, Tx), Line 2-3 untuk menggantikan delay karena NodeMCU akan restart dengan sendirinya apabila terlalu lama delay, Line

```
1. const char *ssid = "Bhaskara";
2. const char *password = "cobatanyampos";
3. const char *host = "192.168.1.12";
```

Line 1-3 to declare the ssid, password, ip used.

```
1. WiFi.mode(WIFI_STA);
2.   WiFi.begin(ssid, password);
3.   while (WiFi.status() != WL_CONNECTED){
4.     delay(500);
5.   }
```

Line 1 to set wifi mode as STA, line 2 to start connection on stated SSID and password, line 3-5 to check if NodeMCU is connected to wifi.

```
1. unsigned long currentMillis = millis();
2.   if(currentMillis - previousMillis >= interval){
3.     previousMillis = currentMillis;
4.     String data = "";
5.     while(DataSerial.available()>0){
6.     data += char(DataSerial.read());
7.     }
8.     data.trim();
9.     if(data!=""){
10.          int index = 0;
11.            for(int i=0; i<data.length(); i++){
12.            char delimiter = '#';
13.            if(data[i] != delimiter)
14.              arrData[index] += data[i];
15.            else
16.              index++;
```

```
17.             }
18.             if(index == 4){
19.             Serial.println("Temperature : " + arrData[0]);
20.             Serial.println("Vibration : " + arrData[1]);
21.             Serial.println("X : " + arrData[2]);
22.             Serial.println("Y : " + arrData[3]);
23.             Serial.println("Life : " + arrData[4]);
24.             Serial.println();
25.             }
26.             Temperature = arrData[0].toInt();
27.             Vibration = arrData[1].toInt();
28.             X = arrData[2].toInt();
29.             Y = arrData[3].toInt();
30.             Life = arrData[4];
31.             arrData[0]="";
32.             arrData[1]="";
33.             arrData[2]="";
34.             arrData[3]="";
35.             arrData[4]="";
36.           }
37.           DataSerial.println("Yes");
38.         }
```

Line 1 to read the current mailing list time, line 2 to reread data, lines 11-17 to receive data from Arduino Uno, lines 18-25 to display data values to serial monitor, lines 26-29 to convert String data to INT, lines 31-35 to empty the arrData[] variable is used to hold new data, line 37 sends the data string “Yes” to Arduino Uno.

```
1. WiFiClient client;
2.   const int httpPort = 80;
3.   if (!client.connect(host, httpPort)){
4.     Serial.println("Failed");
5.     return;
6.   }
7.   String url = "/Project/write-data.php?Temperature=";
8.   url += Temperature;
9.   url += "&Vibration=";
10.         url += Vibration;
11.         url += "&X=";
12.         url += X;
13.         url += "&Y=";
14.         url += Y;
15.         url += "&Life=";
16.         url += Life;
17.         client.print(String("GET ") + url + " HTTP/1.1\r\n" +
18.                      "Host: " + host + "\r\n" +
19.                      "Connection: close\r\n\r\n");
20.         unsigned long timeout = millis();
21.         while (client.available() == 0) {
22.           if (millis() - timeout > 1000) {
23.             Serial.println(">>> Client Timeout !");
24.             client.stop();
25.             return;
26.           }
```

```
27.         }
28.          while (client.available()) {
29.           String line = client.readStringUntil('\r');
30.           Serial.print(line);
31.         }
```

Lines 3-6 to connect NodeMCU on port 80, lines 7-25 to send data that has been received from Arduino Uno to localhost database.

5.1.3 C45 Algorithm Code

This program was created using the MySql programming language, below will explain the program code.

```
1. CREATE TABLE tblData(
2. id INT,
3. Temperature INT,
4. Vibration INT,
5. X INT,
6. Y INT,
7. Life varchar(20)
8. );
9.
10.       LOAD DATA LOCAL INFILE 'myFile0.csv'
11.       INTO TABLE tblData
12.       FIELDS TERMINATED BY ','
13.       ENCLOSED BY '"'
14.       LINES TERMINATED BY '\n'
15.       IGNORE 1 ROWS;
```

Lines 1-8 to create a table to store csv data, lines 10-15 to import csv data to tblData.

```
1. select @amountdata:=count(*)
2. from tblData;
3.
4. select @kurang5tahun:=count(*)
5. from tblData
6. where Life LIKE ('%<5Tahun%');
7.
8. select @lebih5tahun:=count(*)
9. from tblData
10.       where Life LIKE ('%>5Tahun%');
```

Lines 1-2 to calculate the amount of data in tblData, lines 4-6 to calculate the age of the data with the value '<5Tahun', lines 8-10 to calculate the age of the data with the value '>5Tahun'.

```
1. insert into tblCount
2.   (info, amountdata, kurang5tahun, lebih5tahun)
3.   select distinct(A.Temperature) as TEMPERATURE, count(A.Temperature)
   as JUMLAHDATA,
4.   (
5.     select COUNT(*)
```

```
6.     from tblData as B
7.     where B.Life LIKE ('%<5Tahun%') and
8.     B.Temperature = A.Temperature
9.   )AS RATINGLOW,
10.        (
11.          select COUNT(*)
12.          from tblData as C
13.          where C.Life LIKE ('%>5Tahun%') and
14.          C.Temperature = A.Temperature
15.        )as RATINGHIGH
16.      from tblData as A
17.      group by A.Temperature;
18.      ....
19.      insert into tblCount
20.          (info, amountdata, kurang5tahun, lebih5tahun)
21.        select distinct(A.Y) as Y, count(A.Y) as JUMLAHDATA,
22.          (
23.            select COUNT(*)
24.            from tblData as B
25.            where B.Life LIKE ('%<5Tahun%') and
26.            B.Y = A.Y
27.          )AS RATINGLOW,
28.          (
29.            select COUNT(*)
30.            from tblData as C
31.            where C.Life LIKE ('%>5Tahun%') and
32.            C.Y = A.Y
33.          )as RATINGHIGH
34.        from tblData as A
35.        group by A.Y;
```

Lines 1-35 are used to calculate the sum of '<5Tahun' and '>5Tahun' for each attribute, then put in tblCount.

```
1. update tblCount set entropy =
2. (-(kurang5tahun/amountdata) *log2(kurang5tahun/amountdata))
3. +
4. (-(lebih5tahun/amountdata) *log2(lebih5tahun/amountdata));
5.
6. insert into tblTampung(atribut, gain)
7. select atribut, @entropy - SUM((amountdata/@amountdata)*entropy) as
   COUNTGAIN
8. from tblCount
9. group by atribut;
```

Lines 1-4 to calculate the entropy value, lines 6-9 to calculate the gain value.

5.2 Testing

In the C-45 algorithm, after calculating all the data and getting the results from each iteration, then get the final result. Confusion matrix used to calculate the accuracy of the data.

Table 1.4 : The results of the iteration process with 700 data

Iteration	Attribute
1	Y
2	X
3	Vibration

Table 1.5 : Condition training results after iteration process with 700 data

Atribute	Info	Damage	GoodCondition
Vibration	1	0	1
Vibration	9	1	0

Table 1.6 : Condition training master data

Atribute	Info	Damage	GoodCondition	Amount Data
Vibration	1	14	24	38
Vibration	9	15	21	36

Table 1.7 : Confusion matrix data with 700 dataset

	Predicted GoodCondition	Predicted Damage
Reality GoodCondition	24	14
Reality Damage	15	21

$$Accuracy = \frac{21+24}{74} = 0.608\ x\ 100 = 60.8\,\%$$

After calculating the accuracy using 700 Training data, then compare the accuracy with 300 Testing data to show whether the results of calculating the accuracy of the data are correct.

Table 1.8 : Condition testing results after iteration process with 300 data

Atribute	Info	Damage	GoodCondition
Y	219	1	0
Y	239	0	1
Y	242	1	0

Table 1.9 : Condition testing master data

Atribute	Info	Damage	GoodCondition	Amount Data
Y	219	2	3	5
Y	239	0	6	6
Y	242	2	4	6

Table 2.0 : Confusion matrix data with 300 dataset

	Predicted GoodCondition	Predicted Damage
Reality GoodCondition	6	0
Reality Damage	4	7

$$Accuracy = \frac{6+7}{17} = 0.764\,x\,100 = 76.4\,\%$$

After calculating the accuracy of the Training data and Testing data, there is a difference in accuracy. And the accuracy of the final result is the accuracy of the Testing data.