CHAPTER 5
IMPLEMENTATION AND RESULTS

5.1. Implementation

```
1. void sensor(){
   //=======================ultrasonik==========================
2.   digitalWrite(trig, LOW);
3.   delayMicroseconds(8);
4.   digitalWrite(trig,HIGH);
5.   delayMicroseconds(8);
6.   digitalWrite(trig,LOW);
7.   delayMicroseconds(8);
8.
9.  durasi  = pulseIn(echo, HIGH); //menerima gelombang ultrasonik
10.    jarak   = (durasi / 2) / 30;  // ubah jarak durasi jadi jarak
11.     // jarak   = 20;
   //=========================TDS================================
12.      gravityTds.update();
13.      nilaiPPM = gravityTds.getTdsValue()  ; //dapatkan nilai PPM
14.    }
```

The source code on the void sensor is used to retrieve data from the input in the form of sensors, namely ultrasonic sensors, and TDS sensors. The ultrasonic sensor itself is used to determine the distance to the water level, while the TDS sensor is used to determine the concentration of nutrients. The ultrasonic sensor implementation is by the formula to determine the distance using an ultrasonic wave system (Line 2-10), while the TDS sensor uses the GravityTDS library (Line 12-13).

```
15.
16.    void tampil(){
17.      //Serial.print(durasi);
18.      lcd.setCursor(0,0);
19.      lcd.print("N  : ");
20.      lcd.print(nilaiPPM);
21.      lcd.setCursor(0,1);
22.      lcd.print("air: ");
23.      lcd.print(jarak);
24.      lcd.print("cm");
25.      lcd.setCursor(0,1);
26.    }
```

Line 17 is used to adjust the position of the text on the LCD, Lines 18 – 23 are used to display the variables called to the LCD screen, Line 24 is used to adjust the location of the LCD.

```
27.   void FuzzyfikasiAir()
28.    {
29.   //============================================== rendah(penuh)
30.     if (jarak <= 10)
31.       {
32.         air[0] = 1;
33.       }
34.       else if (jarak > 10 && jarak <= 15)
35.       {
36.         air[0] = (15 - jarak)/(15 - 10);
37.       }
38.       else if (jarak >= 15)
39.       {
40.         air[0] = 0;
41.       }
42.
43.    //============================================== cukup
44.      if (jarak <= 10)
45.       {
46.         air[1] = 0;
47.       }
48.      else if(jarak > 10 && jarak <= 15)
49.       {
50.         air[1] = (jarak - 10) / (15 - 10);
51.       }
52.      else if(jarak > 15 && jarak <= 20)
53.       {
54.         air[1] = (20 - jarak) / (20 - 15);
55.       }
56.      else
57.       {
58.         air[1] = 0;
59.       }
60.
61.   //=========================================tinggi(rendah)
62.       if(jarak <= 15)
63.       {
64.         air[2] = 0;
65.       }
66.       else if(jarak > 15 && jarak <= 20)
67.       {
68.         air[2] = (jarak - 15)/(20 - 15);
69.       }
70.       else if (jarak >=20)
71.       {
72.         air[2] = 1;
73.       }
74.
75.      Serial.print(" Penuh  = ");
76.      Serial.print(air[0]);
77.      Serial.println("");
78.      Serial.print(" Cukup  = ");
79.      Serial.print(air[1]);
80.      Serial.println("");
81.      Serial.print(" Rendah = ");
```

```
82.      Serial.print(air[2]);
83.      Serial.println("");
84.     }
```

Lines 29-40 of the program code contain fuzzification values that have been designed according to the full water level fuzzification formula, where the input contains values between 0-15. Lines 43-58 contain fuzzification values in the range between 10-20 and the maximum value is 15 which will produce a fuzzification value of 0 (lines 47-50). Lines 61 – 72 contain fuzzification values in the range of more than 15 or water in low conditions. Lines 74 – 82 display the results of the fuzzification into a serial monitor.

```
85.    void FuzzyfikasiTDS()
86.     {
87.    // ============================================ sangat kurang
88.      if (nilaiPPM <= 500)
89.        {
90.          nutrisi[0] = 1;
91.        }
92.        else if(nilaiPPM > 500 && nilaiPPM <=750)
93.        {
94.          nutrisi[0]=(750 - nilaiPPM)/(750-500);
95.        }
96.        else if(nilaiPPM >= 750)
97.        {
98.          nutrisi[0]=0;
99.        }
100.
101.    // ============================================== kurang
102.     if (nilaiPPM <= 500)
103.     {
104.       nutrisi[1]= 0;
105.     }
106.     else if(nilaiPPM > 500 && nilaiPPM <= 750)
107.     {
108.      nutrisi[1] = (nilaiPPM - 500)/(750 - 500);
109.     }
110.     else if(nilaiPPM > 750 && nilaiPPM <= 1050)
111.     {
112.       nutrisi[1] = (1050 - nilaiPPM)/(1050 - 750);
113.     }
114.     else
115.     {
116.       nutrisi[1]=0;
117.     }
118.
119.    // ============================================== normal
120.     if (nilaiPPM <= 750)
121.     {
122.       nutrisi[2]=0;
123.     }
124.     else if(nilaiPPM > 750 && nilaiPPM <= 1050)
125.     {
126.       nutrisi[2] = (nilaiPPM - 750)/(1050 - 750);
```

```
127.    }
128.    else if(nilaiPPM > 1050 && nilaiPPM <= 1200)
129.    {
130.      nutrisi[2] = 1;
131.    }
132.    else if(nilaiPPM > 1200 && nilaiPPM <= 1600)
133.    {
134.      nutrisi[2] = (1600 - nilaiPPM)/(1600 - 1200) ;
135.    }
136.
137.   // ================================================ tinggi
138.    if(nilaiPPM <= 1200)
139.    {
140.      nutrisi[3]= 0;
141.    }
142.    else if(nilaiPPM > 1200 && nilaiPPM <= 1600)
143.    {
144.      nutrisi[3]=(nilaiPPM - 1200)/(1600 - 1200);
145.    }
146.    else if(nilaiPPM >= 1600)
147.    {
148.      nutrisi[3]=1;
149.    }
150.    Serial.print(" sangat kurang = ");
151.    Serial.print(nutrisi[0]);
152.    Serial.println("");
153.    Serial.print(" kurang       = ");
154.    Serial.print(nutrisi[1]);
155.    Serial.println("");
156.    Serial.print(" normal       = ");
157.    Serial.print(nutrisi[2]);
158.    Serial.println("");
159.    Serial.print(" tinggi       = ");
160.    Serial.print(nutrisi[3]);
161.    Serial.println("");
162.  }
```

Lines 85-98 contain the value of fuzzification for conditions of very poor nutrition, the range of values that can meet this set is 0 – 750. Lines 101 – 116 for conditions of poor nutrition, the range of values that meet this set is 750 – 1050. Rows 119 – 134 is the fuzzification for normal nutritional conditions, which contains a value of 1050 – 200. Due to the trapezoidal shape of the curve, the normal value of 1050 – 1200 will go to else if on lines 127 – 130 which will produce a value of 1. Lines 137 – 148 contain the value fuzzification for high nutritional conditions, the range of values that meet this set is more than 1200. Lines 149 – 160 display the results of fuzzification to a serial monitor.

```
163.  void Rulenutrisi() //========== POMPA NUTRISI =========
164.  {
165.    int i, j;
166.    int no = 1;
167.    for(i = 0; i <= 3; i = i + 1)
168.    {
169.      for(j = 0; j <= 2; j = j + 1)
170.      {
```

```
171.      rul1 = min(nutrisi[i], air[j]);
172.      rule [i][j] = rul1;
173.      Serial.print("Rules ke ...");
174.      Serial.print(no++);
175.      Serial.print(" ->  ");
176.      Serial.println(rule[i][j]);
177.      }
178.    }
179.  //-------------------------------------------------------
180.    rule1   = rule[0][0]; // (penuh, sangat kurang = cepat)
181.    rule2   = rule[0][1]; // (cukup, sangat kurang = lama)
182.    rule3   = rule[0][2]; // (rendah, sangat kurang = lama)
183.  //-------------------------------------------------------
184.    rule4   = rule[1][0]; // (penuh, kurang = diam)
185.    rule5   = rule[1][1]; // (cukup, kurang = sedang)
186.    rule6   = rule[1][2]; // (rendah, kurang = sedang)
187.  //-------------------------------------------------------
188.    rule7   = rule[2][0]; // (penuh, normal = cepat)
189.    rule8   = rule[2][1]; // (cukup, normal = cepat)
190.    rule9   = rule[2][2]; // (rendah, normal = diam)
191.  //-------------------------------------------------------
192.    rule10  = rule[3][0]; // (penuh, tinggi = diam)
193.    rule11  = rule[3][1]; // (cukup, tinggi = diam)
194.    rule12  = rule[3][2]; // (rendah, tinggi = diam)
195.  //-------------------------------------------------------
196.   }
197.
```

Lines 163 – 177 loop for 12 fuzzy nutrition rules by taking the minimum value from the two inputs, then in lines 179 – 193 create a rule container that will contain the values according to the rules that have been created.

```
198.  void Ruleair()
199.  {
200.    int a, b;
201.    int nom = 1; //mulai dari 1
202.    for(a = 0; a <= 3; a = a+1)
203.    {
204.      for(b = 0; b <= 2; b = b+1)
205.      {
206.        rul2 = min(nutrisi[a], air[b]);
207.        ruleb [a][b] = rul2;
208.        Serial.print("Rules ke ...");
209.        Serial.print(nom++);
210.        Serial.print(" ->  ");
211.        Serial.println(ruleb[a][b]);
212.      }
213.    }
214.  //-------------------------------------------------------
215.    ruleb01   = ruleb[0][0]; // (penuh, sangat kurang = diam)
216.    ruleb02   = ruleb[0][1]; // (cukup, sangat kurang = cepat)
217.    ruleb03   = ruleb[0][2]; // (rendah, sangat kurang = lama)
218.  //-------------------------------------------------------
219.    ruleb04   = ruleb[1][0]; // (penuh, kurang = diam)
220.    ruleb05   = ruleb[1][1]; // (cukup, kurang = cepat)
221.    ruleb06   = ruleb[1][2]; // (redah, kurang = lama)
222.  //-------------------------------------------------------
223.    ruleb07   = ruleb[2][0]; // (penuh, normal = diam)
```

```
224.    ruleb08   = ruleb[2][1]; // (cukup, normal = cepat)
225.    ruleb09   = ruleb[2][2]; // (rendah, normal = lama)
226.  //------------------------------------------------------------
227.    ruleb010  = ruleb[3][0]; // (penuh, tinggi = cepat)
228.    ruleb011  = ruleb[3][1]; // (cukup, tinggi = cepat)
229.    ruleb012  = ruleb[3][2]; // (rendah, tinggi = lama)
230.  //------------------------------------------------------------
231.  }
```

Lines 198 – 212 loop for 12 water rules by taking the minimum value of the two inputs, then in lines 214 – 228 create a container which will later contain the values according to the rules that have been created.

```
232.   void Defusifikasi_nutrisi ()
233.   {
234.    float Diam = 0;
235.    float Cepat = 5;
236.    float Sedang = 10;
237.    float Lama  = 15;
238.    output1 = (rule1*Cepat)+(rule2*Lama)+(rule3*Lama)+
            (rule4*Diam)+(rule5*Sedang)+(rule6*Sedang)+
      (rule7*Diam)+(rule8*Diam)+(rule9*Diam)+
      (rule10*Diam)+(rule11*Diam)+(rule12*Diam);
239.  output2   = rule1+rule2+rule3+rule4+rule5+rule6+rule7+rule8+
                  rule9+rule10+rule11+rule12;
240.    hasil_defu = output1/output2;
241.    delayPompaNutrisi = (hasil_defu * 1000);
242.    Serial.print(output1);
243.    Serial.println("");
244.    Serial.print(output2);
245.    Serial.println("");
246.
247.    if (hasil_defu >= 0.00 && hasil_defu <5.00 ){
248.      digitalWrite(pompa01, LOW);
249.      //delay(delayPompaNutrisi);
250.      digitalWrite(pompa01, LOW);
251.      Serial.print("Result  : ");
252.      Serial.print(hasil_defu);
253.      Serial.println(" -> pompa mati");
254.    }
255.    else if (hasil_defu >= 5.00 && hasil_defu <10.00){
256.      digitalWrite(pompa01,HIGH);
257.      delay(delayPompaNutrisi);
258.      digitalWrite(pompa01, LOW);
259.      Serial.print("Result  : ");
260.      Serial.print(hasil_defu);
261.      Serial.print(" -> pompa cepat");
262.    }
263.    else if (hasil_defu >= 10.00 && hasil_defu < 15.00){
264.      digitalWrite(pompa01,HIGH);
265.      delay(delayPompaNutrisi);
266.      digitalWrite(pompa01, LOW);
267.      Serial.print("Result  : ");
268.      Serial.print(hasil_defu);
269.      Serial.print(" -> pompa sedang");
270.    }
271.    else if (hasil_defu >= 15.00 ){
272.      digitalWrite(pompa01,HIGH);
```

```
273.       delay(delayPompaNutrisi);
274.       digitalWrite(pompa01, LOW);
275.       Serial.print("Result  : ");
276.       Serial.print(hasil_defu);
277.       Serial.print(" -> pompa lama");
278.     }
279.
280.   }
```

Lines 223 – 236 are the declaration of the output value for the nutrition pump, then lines 237 – 239 are the implementation of defuzzification (WA) the number of rules is multiplied by the output value then divided by the value of all the rules. Line 240 creates a variable delayPumpNutrition which contains the result of Defuzzyfication multiplied by 1000 to convert it to seconds. Lines 241 – 244 display the output values of the numerator and denominator. Lines 246 – 253 are used to enter the value of the nutrient pump delay if the value is less than 5 with the result that the pump is off. Lines 254 – 261 for condition values between 5 – 9 with fast pump results. Lines 262 – 269 for conditions of values between 10 – 14 with medium pump results and the last line 270 – 277 for conditions of values more than 15 with old pump results.

```
281.   void Defuzifikasi_air()
282.   {
283.     float Diam1 = 0;
284.     float Cepat1 = 10;
285.     float Lama1  = 20;
286.     output1_air  =  (ruleb01*Diam1)+(ruleb02*Cepat1)+
                  (ruleb03*Lama1)+(ruleb04*Diam1)+
          (ruleb05*Cepat1)+(ruleb06*Lama1)+
       (ruleb07*Diam1)+(ruleb08*Cepat1)+
       (ruleb09*Lama1)+(ruleb010*Cepat1)+
       (ruleb011*Cepat1)+(ruleb012*Lama1);
287.   output2_air=ruleb01+ruleb02+ruleb03+ruleb04+ruleb05+ruleb06+
       ruleb07+ruleb08+ruleb09+ruleb010+ruleb011+ruleb012;
288.     hasil_defu_air = output1_air/output2_air;
289.     delayPompaAir = (hasil_defu_air * 1000);
290.     Serial.print(output1_air);
291.     Serial.println("");
292.     Serial.print(output2_air);
293.     Serial.println("");
294.
295.     if (hasil_defu_air >= 0.00 && hasil_defu_air <10 ){
296.       digitalWrite(pompa02, LOW);
297.       //delay(delayPompaAir);
298.       digitalWrite(pompa02, LOW);
299.       Serial.print("Result  : ");
300.       Serial.print(hasil_defu_air);
301.       Serial.println(" -> pompa mati");
302.     }
303.     else if(hasil_defu_air >= 10.00 && hasil_defu_air <20.00){
304.       digitalWrite(pompa02,HIGH);
305.       delay(delayPompaAir);
306.       digitalWrite(pompa02, LOW);
```

```
307.       Serial.print("Result  : ");
308.       Serial.print(hasil_defu_air);
309.       Serial.print(" -> pompa cepat");
310.     }
311.     else if (hasil_defu_air >= 20.00 ){
312.       digitalWrite(pompa02,HIGH);
313.       delay(delayPompaAir);
314.       digitalWrite(pompa02, LOW);
315.       Serial.print("Result  : ");
316.       Serial.print(hasil_defu_air);
317.       Serial.print(" -> pompa lama");
318.     }
319.   }
```

Lines 282 – 284 declare the value of the water pump output and then lines 285 -287 are the implementation of the Defuzzyfication model (WA) by multiplying the number of rules by the output value then dividing by the value of all the rules. Line 288 to convert the defuzzification value to a seconds scale. Lines 294 – 301 to enter a value with a result of less than 10 with a dead pump result, lines 302 – 309 to enter a defuzzification value 10 – 19 with a fast pump result then on lines 310 – 317 to enter a defuzzification value of 20 and above with an old pump result.

5.2. Result

At this stage, testing is carried out with the initial nutrient conditions of 777.88 ppm and the height of the water is 19.33 cm. The microcontroller will work according to the fuzzy logic that has been implemented into it. To determine the pump delay is done with the following logic:

- Output Nutrient pump:

if (hasil_defu >= 0.00 && hasil_defu < 5.00) = Pompa off

if (hasil_defu >= 5.00 && hasil_defu < 10.00) = Pompa cepat

if (hasil_defu >= 10.00 && hasil_defu < 15.00) = Pompa sedang

if (hasil_defu >= 15.00) = Pompa lama

- Output Water pump:

if (hasil_defu_air >= 0.00 && hasil_defu_air <10) = Pompa off

if (hasil_defu_air >= 10.00 && hasil_defu_air <20.00) = Pompa cepat

if (hasil_defu_air >= 20.00) = Pompa lama

Time	Nutrient	Water Level	Nutrient Pump	Water Pump
20:13:25	777.88 ppm	19.33 cm	8.43	17.85
20:14:52	793.26 ppm	18.59 cm	7.29	16.14
20:15:10	805.76 ppm	18.28 cm	6.87	15.74
20:15:34	818.45 ppm	18.04 cm	6.87	15.34
20:15:48	818.45 ppm	17.75 cm	6.67	15.08
20:16:11	824.86 ppm	17.56 cm	6.31	14.71
20:16:37	837.82 ppm	17.27 cm	6.97	14.92
20:17:00	815.26 ppm	17.44 cm	7.18	13.43
20:17:22	808.92 ppm	16.41 cm	6.97	14.25
20:18:44	815.26 ppm	16.00 cm	6.72	12.48
20:18:55	831.31 ppm	15.82 cm	6.94	11.89
20:19:10	828.08 ppm	15.58 cm	7.15	9.86*
20:19:51	841.09 ppm	15.33 cm	6.99	10.40
20:20:10	837.82 ppm	15.10 cm	6.69	9.67*
20:20:33	847.66 ppm	14.91 cm	7.12	9.24*
20:20:35	963.83 ppm	14.73 cm	3.08*	9.01*
20:20:38	1159.05 ppm	14.30 cm	0.00*	8.59*
20:20:39	1205.00 ppm	14.67 cm	0.00*	8.49*
20:20:40	1195.99 ppm	14.24 cm	0.00*	9.36*
20:20:41	1205.00 ppm	14.67 cm	0.00*	9.39*
20:20:42	1214.06 ppm	14.67 cm	0.00*	9.41*
20:20:43	1223.18 ppm	14.67 cm	0.00*	9.41*

Table 2: Table Testing

From Table 2 it can be seen that the movement of nutrients and water level to the optimal condition is where the value of defuzzification of water and nutrients is off. based on data from where the initial condition is the nutrition of 777.88 ppm and the height of the nutrient reservoir is 19.23 cm it takes 7 minutes 18 seconds to reach the optimal condition in the table with code *. In other words, the designed tool can work according to the ordered program to mix nutrients with water properly.

Furthermore, testing is carried out on a nutrient condition of 122.10 PPM (very low) and a water level of 14.30 cm (full), this test will prove that the nutrient pump will still turn off even though the nutrients in the reservoir are not optimal. This is done so that the water in the reservoir is not full and overflows everywhere which causes wastage of nutrients. The data can be seen in the following table

Time	Nutrient	Water Level	Nutrient Pump	Water Pump
15:37:10	122.10 ppm	14.30 cm	13.32	8.32*
15:37:27	122.10 ppm	14.16 cm	12.64	7.64*
15:37:43	199.52 ppm	13.82 cm	11.93	6.93*
15:37:58	280.68 ppm	13.46 cm	12.03	7.03*
15:38:13	331.97 ppm	13.52 cm	12.17	7.17*
15:38:29	393.94 ppm	13.58 cm	12.61	7.61*
15:38:44	446.10 ppm	13.80 cm	10.98	5.98*
15:38:59	478.19 ppm	12.99 cm	8.44	3.75*
15:39:10	508.19 ppm	11.83 cm	7.53	4.11*
15:39:21	557.52 ppm	11.85 cm	7.82	4.60*
15:39:32	568.81 ppm	12.19 cm	5.92	2.96*
15:39:41	592.03 ppm	11.05 cm	6.06	3.34*
15:39:50	608.80 ppm	11.25 cm	5.41	2.89*
15:39:59	623.53 ppm	11.02 cm	4.19*	1.91*
15:40:02	638.59 ppm	10.59 cm	5.23	2.86*
15:40:11	633.53 ppm	11.00 cm	4.42*	2.25*
15:40:14	646.24 ppm	10.73 cm	4.09*	2.09*
15:40:17	656.58 ppm	10.66 cm	3.90*	1.91*
15:40:21	656.58 ppm	10.59 cm	3.06*	0.86*
15:40:24	641.13 ppm	10.23 cm	3.33*	0.80*
15:40:27	623.53 ppm	10.22 cm	2.33*	0.00*
15:40:31	633.53 ppm	9.74 cm	2.13*	0.25*
15:40:34	656.58 ppm	10.06 cm	3.42*	1.59*
15:40:37	664.44 ppm	10.47 cm	1.55*	0.00*
15:40:40	672.38 ppm	10.00 cm	2.93*	1.59*
15:40:44	694.00 ppm	10.47 cm	3.06*	1.54*

Table 3: Tabel Testing

In table 3, it can be concluded that the nutrient pump will continue to stop if the water in the reservoir is full in the table with code *, in the data the nutrient pump stops at a value of 633.53 PPM or the nutrients are in the less category, this is because the distance of the water in the nutrient reservoir is full or close. Which is less than 10 cm from the ultrasonic sensor.