CHAPTER 5

IMPLEMENTATION AND TESTING

5.1. Implementation

This study uses the C programming language contained in the Arduino IDE. In this study, the Haversine method is used to convert distances and then send notifications in the form of messages and the minimum distance when the PIR sensor detects movement.

In the program lines 1 - 7, namely to make a configuration connection to connect the ESP8266, PIR sensor with WiFi and connect with telegram.

```
1. const char* ssid = "agusetiawan";
2. const char* password = "123456789";
3. #define CHAT_ID
4. #define BOT_TOKEN
5. WiFiClientSecure client;
6. UniversalTelegramBot bot(BOTtoken, client);
7. const int motionSensor = 14;
```

In this next line to declare Latitude and Longtitude which will be converted to distance using haversine.

```
8. struct LatLong {
9. double latitude;
   double longitude;
10.};
```

In lines 11 – 20 are calculations to convert latitude and longtitude that have been inputted into distances using the haversine method.

```
11.double distance (double lat1, double lng1, double lat2, double lng2) {
12.double dx, dy, dz;
13.lng1 -= lng2;
14.lng1 *= TO_RAD,
15.lat1 *= TO_RAD,
16.lat2 *= TO_RAD;
17.dz = sin(lat1) - sin(lat2);
18.dx = cos(lng1) * cos(lat1) - cos(lat2);
19.dy = sin(lng1) * cos(lat1);
20.return asin(sqrt(dx * dx + dy * dy + dz * dz) / 2) * 2 * R * 1000; //
   *1000 for metres
      }
```

On lines 21 – 29 is an array of codes used to express the converted distance results by calling from the distance variable which has been converted to latitude and longitude so that the minimum distance is calculated to find the nearest location

```
21.double jumlah_posisi=5;
22.double jarak[5];
23.void loop()
24.{
25.jarak[0] = distance(Rumah.latitude, Rumah.longitude, POS1.latitude,
   POS1.longitude);
26.jarak[1] = distance(Rumah1.latitude, Rumah1.longitude, POS2.latitude,
   POS2.longitude);
27.jarak[2] = distance(Rumah2.latitude, Rumah2.longitude, POS3.latitude,
   POS3.longitude);
28.jarak[3] = distance(Rumah3.latitude, Rumah3.longitude, POS4.latitude,
   POS4.longitude);
29.jarak[4] = distance(Rumah4.latitude, Rumah4.longitude, POS5.latitude,
   POS5.longitude);
```

On lines 30 – 34 is a calculation to find the closest route, by calling the array that was created previously, namely the distance and the number of positions to find which distance is the closest.

```
30.double min=10000000000;
31.for (int a=0;a<jumlah_posisi;a++){
32.if (jarak[a]<min){
33.min=jarak[a];
34.}
```

On lines 35– 40 is a code to send output to telegram, if the PIR sensor detects movement it will send a notification along with the closest distance to the telegram bot that is already connected to the ESP8266.

```
35.if (motionDetected){
36.String  protokol = "\nMotion Detected\nJl Mawar No. 10\nJarak Minimum:
   "+String(min)+" M";
37.bot.sendMessage(CHAT_ID, protokol, "");
38.Serial.println("Motion Detected");
39.motionDetected = false;
40.}
```

5.2. Testing

This test uses manual calculations through google maps map scale and displays the conversion results from the haversine method, which is looking for 2 straight line points so that the closest distance can be found, after that looking for the distance to which location is the closest. The test was carried out 10 times with the initial coordinates of the house and the coordinates of the destination point, namely the security post.

The table above is a trial with a total of 10 test scenarios using different coordinate points. From calculations using the haversine method to find the distance between 2 points, it can be seen the location of the post which has the closest distance to send security notifications if motion is detected. With latitude and longitude the starting point is -7.0529222, 110.4226965.

Table 5.1 Test Scenario 1

TEST SCENARIO 1	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.0522627, 110.4222821	120m	86.423m	TRUE POS 4
POS 2 -7.0523033, 110.4225802	150m	70.005m	
POS 3 -7.052238, 110.422411	120m	82.345m	
POS 4 -7.052564, 110.422250	81m	63.358m	
POS 5 -7.052489, 110.422124	93m	79.446m	

Table 5.2: Test Scenario 2

TEST SCENARIO 2	DISTANCE MAP	HAVERSINE DISTANCE	REULT
POS 1 -7.053127, 110.422494	110m	82.037m	TRUE POS 4
POS 2 -7.053227, 110.422733	120m	87.090m	
POS 3 -7.053126, 110.422456	130m	87.096m	
POS 4 -7.052772 , 110.422083	100m	80.242m	
POS 5 -7.052783, 110.422039	100m	88.869m	

Table 5.3: Test Scenario 3

TEST SCENARIO 3	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.0523033 110.4225802	230m	75.455m	FALSE
POS 2 -7.052432, 110.422773	180m	55.158m	
POS 3	210m	47.027m	

-7.052597, 110.422903			
POS 4 -7.052627, 110.423012	210m	47.850m	
POS 5 -7.052593, 110.423156	230m	62.540m	

Table 5.4: Test Scenario 4

TEST SCENARIO 4	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.052678, 110.422173	64m	63.833m	TRUE POS 1
POS 2 -7.052960, 110.421896	120m	88.438m	
POS 3 -7.053120, 110.422572	25m	25.933m	
POS 4 -7.052502, 110.422199	85m	72.092m	
POS 5 -7.052786, 110.422044	90m	73.581m	

Table 5.5: Test Scenario 5

TEST SCENARIO 5	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.0532062, 110.4209491	230m	195.400m	TRUE POS 5
POS 2 -7.052650, 110.421586	190m	126.230m	
POS 3 7.052269, 110.421990	120m	106.555m	
POS 4 -7.051619, 110.422344	220m	150.040m	
POS 5 -7.052786, 110.422044	90m	73.581m	

Table 5.6: Test Scenario 6

TEST SCENARIO 6	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.053395, 110.422715	75m	52.613m	TRUE POS 1
POS 2 -7.053262, 110.422440	140m	95.400m	
POS 3 -7.053538, 110.422282	260m	82.347m	

POS 4 -7.053119, 110.422109	76m	68.426m	
POS 5 -7.052786, 110.422044	90m	73.581m	

Table 5.7: Test Scenario 7

TEST SCENARIO 7	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.052564, 110.422250	81m	63.358m	TRUE POS 1
POS 2 -7.052959, 110.421466	160m	135.852m	
POS 3 -7.053246, 110.421300	190m	158.259m	
POS 4 -7.052334, 110.422145	100m	89.341m	
POS 5 -7.052371, 110.421871	130m	109.796m	

Table 5.8: Test Scenario 8

TEST SCENARIO 8	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.052500, 110.421061	240m	186.489m	TRUE

POS 2 -7.052117, 110.421174	200m	190.381m	POS 5
POS 3 -7.052575, 110.421131	240m	177.020m	
POS 4 -7.052154, 110.421351	190m	171.298m	
POS 5 -7.052245, 110.421549	160m	147.328m	

Table 5.9: Test Scenario 9

TEST SCENARIO 9	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.052718, 110.422299	50m	49.394m	TRUE POS 1
POS 2 -7.053034, 110.422173	71m	59.093m	
POS 3 -7.052716, 110.422216	59m	57.770m	
POS 4 -7.053122, 110.422188	67m	60.353m	
POS 5 -7.053024, 110.422050	85m	72.236m	

Table 5.10: Test Scenario 10

TEST SCENARIO 10	DISTANCE MAP	HAVERSINE DISTANCE	RESULT
POS 1 -7.0582596, 110.4286098	1.6 km	882.074m	TRUE POS 2
POS 2 -7.0557744, 110.4164885	1.0 km	754.923m	
POS 3 -7.0560775, 110.4164389	1.0 km	774.566m	
POS 4 -7.0582786, 110.4163274	1.2 km	921.271m	
POS 5- 7.0558924, 110.4158407	1.1 km	825.507m	

Table 5.11: Final Result

TEST SCENARIO	RESULT
1	TRUE
2	TRUE
3	FALSE
4	TRUE
5	TRUE
6	TRUE
7	TRUE
8	TRUE
9	TRUE
10	TRUE

From the table above, it can be seen that using the haversine method can determine the closest route from the starting point to the destination point, by converting latitude and longitude into distances and calculating the minimum distance by taking the distance from the array that has been created.