CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Arduino IDE

This Project uses Arduino IDE application. This application used compile and upload the program. The program can be seen as below:

1. Libraries
   
   ```
   #include <ESP8266WiFi.h>
   #include <ThingerESP8266.h>
   #include <Wire.h>
   #include <RtcDS3231.h>
   #include <Servo.h>
   ```

2. Define connection with Broker
   
   ```
   #define USERNAME "sukseskripsi"
   #define DEVICE_ID "konek"
   #define DEVICE_CREDENTIAL "konek_ndug"
   ```

3. Define connection with WIFI
   
   ```
   #define SSID "Buronan"
   #define SSID_PASSWORD "uvuwewewek"
   ```

4. Declaring variable and object
   
   ```
   int trigPin=D0;
   int echoPin=D2;
   double duration, distance;
   ```
char str[15];

5. Setting transfer rate to serial and setting sensor as input / output

Serial.begin(9600);

pinMode(D6, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

6. Define and configure variable with object

RtcDS3231<TwoWire> rtcObject(Wire);

ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

myservo.attach(D5);

rtcObject.Begin();

Wire.begin(D3, D4);

RtcDateTime currentTime = RtcDateTime(__DATE__, __TIME__);

rtcObject.SetDateTime(currentTime);

RtcDateTime currentTime = rtcObject.GetDateTime();

thing.add_wifi(SSID, SSID_PASSWORD);

Servo myservo;

7. Make the device that can connect with Broker

thing["led"] << digitalPin(D6);

thing["SONIC"] >> [] (pson& out){

digitalWrite(trigPin, LOW); // Get Start

delayMicroseconds(2); // stable the line
digitalWrite(trigPin, HIGH); // sending 10 us pulse
delayMicroseconds(10); // delay
digitalWrite(trigPin, LOW); // after sending pulse waiting to receive signals
duration = pulseIn(echoPin, HIGH); // calculating time
distance = (duration/2)/29.1; // single path
out = distance;
}

8. Function from thinger.io to transfer data
   thing.handle();

9. Getting time
   RtcDateTime currentTime = rtcObject.GetDateTime(); // get the time from the RTC
   char str[15]; // declare a string as an array of chars
   sprintf(str, "%d/%d/%d %d:%d:%d", currentTime.Year(), currentTime.Month(), currentTime.Day(), currentTime.Hour(), currentTime.Minute(), currentTime.Second()); // %d allows to print an integer to the string
   currentTime.Year(); // get year method
   currentTime.Month(); // get month method
   currentTime.Day(); // get day method
   currentTime.Hour(); // get hour method
   currentTime.Minute(); // get minute method
   currentTime.Second(); // get second method
);
   Serial.println(str);
10. Make requisite to the machine

    Serial.println(distance);

    if((currentTime.Hour()>=18) || (currentTime.Hour()<=9))
    {
        if(distance>=10)
        {
            myservo.write(180);
            Serial.println("servo 180");
        }
        else
        {
            myservo.write(90);
            Serial.println("servo 90");
        }
    }
    else
    {
        if(distance>=10)
        {
            myservo.write(90);
            Serial.println("servo 90");
        }
        else
        {
            myservo.write(90);
            Serial.println("servo 90");
        }
    }
5.2 Testing

The figure below shows the NodeMcu connected to the wifi and broker, the graph which produced from the distance measurement between feed with ultrasonic sensor, and the result from the distance data which already uploaded to broker.

1. Connecting to the Broker

NodeMcu will print the connection status in monitor serial. If NodeMcu status has success connected with the broker as good, so the NodeMcu will printed in monitor serial like this picture below.

![Figure 7: Connection with wifi and broker](image)
2. measuring the distance

Distance measurement is done after NodeMcu is connected with the broker. NodeMcu gives commands to the HC-SR04 distance sensor for distance reading. The distance measurement data will be uploaded to the broker once every minute.

3. Saving the distance data as a graph in broker

The distance data will be uploaded become a graph in broker every one minute like this picture below.
4. Saving the distance data in bucket broker data.

The result data of distance reading can be saved into bucket data provided by broker. The data can be stored in the broker for a certain period and can be downloaded as a .csv file.

5. Monitoring using smartphone

If the device has integrated with broker, so data who uploaded to broker can be monitoring using smartphone. Broker Thinger.io supplied an application based android who can use to monitor all at once control in your device. This application using internet so that working.

Figure 10: Data bucket

The result data of distance reading can be saved into bucket data provided by broker. The data can be stored in the broker for a certain period and can be downloaded as a .csv file.

Figure 11: Monitoring Device