CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

The server runs on port 50000. The server is running wait for connections from the client. When the connection is received, the server will notify IP address and the origin port from the client. If the client sends a message, the server will also notify.

All received data is handled by RequestHandler Class. The RequestHandler Class separates the data received based on whitespaces. Each data is given the date and time when it is received by the server. The sensor value is converted to double data type.

Data received is handled by the RequestHandler Class, stored into the database. The data stored into database are:

1. Access Key
2. Sensor Identity
3. Date and Time
4. Sensor Value

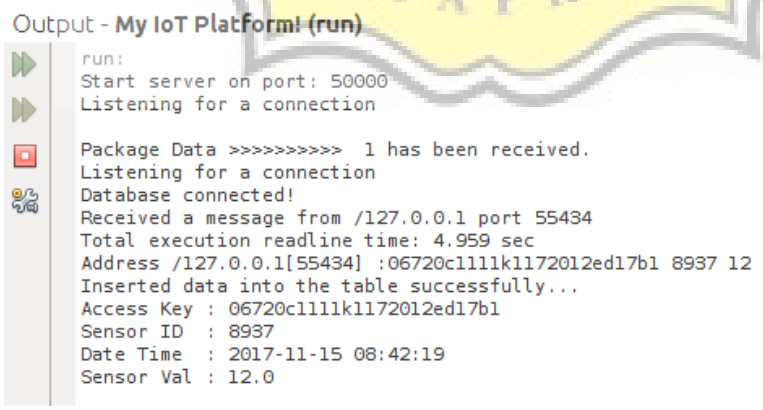

Illustration 5.1: Implementation My IoT Platform!

5.2 Testing

The devices that used in all testing scenario are :

1. PC 1

   Processor : AMD A4-3330MX 2,4 GHz APU

   Graphics : Radeon(tm) HD Graphics

   Memory : 4 GB

   OS : Ubuntu 16.04 64-bit

2. PC 2

   Processor : Intel® Core™ i3-3217U (3M Cache, 1.8 GHz)

   Graphics : Nvidia GeForce GT 920M 2GB DDR3

   Memory : 4 GB

   OS : Windows 8.1 64-bit

There are two testing in this project :

5.2.1 Server Endurance

Table 5.1: Single PC (Using PC 1) in Localhost, MySQL limit 50

No.	Attack	MySQL Max Connection	Success	Fail	Execution Time (s)
1.	500	50	500	0	8,593
2.	2500	50	2493	7	40,681
3.	7500	50	7500	68	127,028
4.	15000	50	14732	268	253,203
5.	30000	50	29342	658	510,715

Default MySQL variable maximum connection is 151. By using mysql maximum connection to 50, from the result table above we can conclude that the attack does not work 100%.

Table 5.2: Single PC (Using PC1) in Localhost, MySQL limit is set to default

No.	Attack	MySQL Max Connection	Success	Fail	Execution Time (s)
1.	500	151	500	0	7,673
2.	2500	151	2500	0	42,842
3.	7500	151	7500	0	126,007
4.	15000	151	15000	0	254,193
5.	30000	151	30000	0	509,024

The same test is done like the table 5.1, the maximum mysql connection is set to default, from the result table above we can conclude that the attack does work 100%.

Table 5.3: Single PC (Using PC 2) in Localhost, MySQL limit 151

No.	Attack	MySQL Max Connection	Success	Fail	Execution Time (s)
1.	500	151	195	305	3
2.	2500	151	1936	564	15
3.	7500	151	6307	1193	37,918
4.	15000	151	13678	1322	55,418
5.	30000	151	28726	1274	96,618

The same test is done like the table 5.2, but the computer specifications are better. From the result table above we can conclude theat the attack does not work 100%. The time needed to attack is shorter. And the failure rate is higher than second table.

Table 5.4: Single PC (Using PC 2) in Localhost, MySQL Limit 251

No.	Attack	MySQL Max Connection	Success	Fail	Execution Time (s)
1.	500	251	500	0	2,93
2.	2500	251	2500	0	14,1
3.	7500	251	7500	0	39,198
4.	15000	251	15000	0	53,81
5.	30000	251	30000	0	99,019

The same test is done like the table 5.3, but the maximum mysql connection is set to 251. From the result table above we can conclude the attack does work 100%. The time needed to attack is relatively the same.

Table 5.5: Multi PC (Using PC 1) in Wireless Network, Attacker (Using PC 2)

No.	Attack	MySQL Max Connection	Success	Fail	Execution Time (s)
1.	500	151	492	8	12,523
2.	2500	151	2454	46	32,955
3.	7500	151	7039	461	105,225
4.	15000	151	14404	596	149,556
5.	30000	151	29506	494	371,287

This test is done over a wireless network. Maximum MySQL connection in the default set. Attacks are done via PC 2. From the result table above we can conclude the attack does not work 100%.

Table 5.6: Multi PC (Using PC 2) in Wireless Network, Attacker (Using PC 1)

No.	Attack	MySQL Max Connection	Success	Fail	Execution Time (s)
1.	500	151	500	0	8,843
2.	2500	151	2500	0	26,316
3.	7500	151	7500	0	75,617
4.	15000	151	15000	0	138,197
5.	30000	151	30000	0	230,88

This test is done over a wireless network. Maximum connection of MySQL in default set. Attacks done via PC 1. From the result table above we can conclude the attack does work 100%.

All of the test proved that hardware is a major component in server endurance. The better the quality of server hardware, the better the robustness of the server. By knowing the resilience of the server, we can set the maximum mysql connection efficiently so as not to extravagant memory. With the maximum efficiency of mysql connection will balance the attack of various speeds of client attacks

5.2.2 Packet Size

Since no program can monitor the quantity of packets sent / received efficiently, then to calculate the amount of packets by manually counting the number of characters sent by the client.

- Thingspeak code to update value

```
String url = "/update?key=";
url += writeAPIKey;
url += "&field1=";
url += String(temperature);
url += "&field2=";
url += String(humadity);
url += "\r\n";
client.print(String("GET       ")+url+"       HTTP/1.1\r\n")+"Host:
"+host+"\r\n"+"Connection: close\r\n\r\n");
```

```
The result of the code above are :
GET /update?key=QN9RMDDN47AKAMFJ&field1=10&field2=20
HTTP/1.1
Host: 192.168.1.3
Connection: close
```

One character is one byte, for once data transmission to transmits two sensors data with two digits the number of thingspeak takes 94 bytes

- My IoT Platform! Code to update value

```
read_sensor_1 = digitalRead(sensor1);
Serial.println(read_sensor_1);
  if (sensor_1.connect(server, 50000)) {
    sensor_1.print(ACCESS_KEY);
    sensor_1.print(" ");
    sensor_1.print(sensor_ID_1);
    sensor_1.print(" ");
    sensor_1.println(read_sensor_1);
    sensor_1.stop();
    delay(5000);
1.   }else if (sensor_2.connect(server, 50000)) {
2.     sensor_2.print(ACCESS_KEY);
3.     sensor_2.print(" ");
4.     sensor_2.print(sensor_ID_1);
5.     sensor_2.print(" ");
6.     sensor_2.println(read_sensor_1);
7.     sensor_2.stop();
8.     delay(5000);
9.   }

The result of the code above:
06720c1111k1172012ed17b1 8937 10
```

One character is one byte, then for once data transmission transmits two sensor data with two digits the number of MyIoT Platform! takes 64 bytes.