Chapter 5

SYSTEM IMPLEMENTATION

4.1 Hardware Implementation

In this chapter the author will discuss about the hardware implementation. To start with the implementation, first we must look at the hardware specification needed to create the prototype, which will consist of sensors, pins, and design schematics.

The prototype solution the author chose is the water flow sensor, which is able to detect the activity of water flowing through pipes. This prototype will get the data from the sensor and store the data into a local database using phpMyAdmin. With the data stored in the database, those data can be used by the user to do multiple activities such as, monitoring the water flow rate, generating water billing info to the residents, and can be used by the system to be forecasted and produce future data for planning ahead and for maintenance of the hardware.
4.1.1 Hardware Specification

The hardware that are used in the prototype are as follows:

- Arduino Uno R3
- Jumper Wires
- USB Cable
- Water Flowmeter 0.5 Inch YF S201

4.1.2 Pin Description

Water Flow Sensor

<table>
<thead>
<tr>
<th>Pin</th>
<th>Wiring to Arduino Uno R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Line</td>
<td>Digital 2</td>
</tr>
<tr>
<td>VCC</td>
<td>5V</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
</tbody>
</table>

4.1.3 Electronic Diagram

Water Flow Sensor

*Figure 37: Arduino Board connected to a Water Flow Sensor

*Figure 38: Electronic Schematic Diagram*
4.2 Software Implementation

The next in the system implementation is the software implementation. Here, we are going to use the Arduino software as the environment to code and get the data from the sensor. Then, Visual Studio will be use as the environment to get the data from the Arduino code and store it in the localhost database using phpMyAdmin. The reason this prototype is only using a localhost database is because there was no access to the AWS Cloud Service, due to credit card limitation.

4.2.1 Arduino Implementation

The implementation of the Arduino water flow sensor is coded using the Arduino software is as below.

1. Water Flow Sensor

These lines of code are the setup for the water flow sensor by declaring variables and initializing serial connection for reporting values back to the host.

The Hall-effect sensor is connected to pin 2 which uses interrupt 0. This is configured to trigger on a FALLING state change, which is a transition from HIGH state to LOW state.
byte statusLed = 13;
byte sensorInterrupt = 0; // 0 = digital pin 2
byte sensorPin = 2;

float calibrationFactor = 4.5;

volatile byte pulseCount;

float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;

unsigned long oldTime;

void setup()
{
    Serial.begin(9600);

    pinMode(sensorPin, INPUT);
    digitalWrite(sensorPin, HIGH);

    pulseCount = 0;
    flowRate = 0.0;
    flowMilliLitres = 0;
    totalMilliLitres = 0;
    oldTime = 0;

    attachInterrupt(sensorInterrupt, pulseCounter, RISING);
}


Figure 39: Arduino Code

These lines of code are the loop function, where the code will always run in a loop for as long as the Arduino board is online. This code will give an output once every second and calculates the cumulative milliliter of the flowing water using the water flow rate per second.
```cpp
void loop()
{
    // Only process counters once per second
    if((millis() - oldTime) > 1000)
    {
        detachInterrupt(sensorInterrupt);
        flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor;
        oldTime = millis();

        flowMilliLitres = (flowRate / 60) * 1000;
        totalMilliLitres += flowMilliLitres;

        Serial.println(totalMilliLitres);

        pulseCount = 0;
        attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
    }
}
```

Figure 40: Arduino Code

This code is to increment the pulse counter used in the code above. The pulse counter is used to calculate the water flow rate per second and after each loop is finished the pulse counter will reset to zero so that it can start incrementing again.

```cpp
void pulseCounter()
{
    pulseCount++;
}
```

Figure 41: Arduino Code
4.2.2 Database Implementation

The database implementation uses Visual Studio as the environment for getting the data from the Arduino and storing it to the database. The database that we used is a localhost database called phpMyAdmin. The code to store the data into the database is as follows.

These lines of code are written to prepare a connection to the database by declaring a few variables and initializing the variables.

![Visual Studio Code](image1)

*Figure 42: Visual Studio Code*

This function is to establish a connection to the Arduino board in order to get data from the water flow sensor.

![Visual Studio Code](image2)

*Figure 43: Visual Studio Code*
This function is to open and close a connection the database in order to write and store data in the database and closes the connection after writing into the database.

```
//Open DB connection
private static void connectToDB()
{
    try
    {
        connection = new MySqlConnection(connectionString);
        connection.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("error: " + e);
    }
}

//Close DB connection
private static void closeDB()
{
    if (connection != null)
    connection.Close();
}
```

Figure 44: Visual Studio Code

This function is used to insert the data extracted from the Arduino sensor into the database.

```
//Insert a string to our DB's table
private static void insertToDB(String s)
{
    try
    {
        MySqlCommand cmd = new MySqlCommand();
        cmd.Connection = connection;
        cmd.CommandText = "INSERT INTO waterflowsensor(sensorId, outputsignalizer) VALUES('Water Flow Sensor 1', @value)";
        cmd.Parameters.AddWithValue('@value', s);
        cmd.Prepare();
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine("error: " + e);
    }
}
```

Figure 45: Visual Studio Code