



# **Smart Water Meter using Radio Frequency and Internet of Things**

**Submitted by: Min Choi**

School of Engineering, Computer and Mathematical Science

A final year project report presented to the Auckland University of Technology  
in partial fulfilment of the requirements of the degree of  
Bachelor of Engineering

**2020**

# Statement of Originality

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

23-October-2020

Date

Signed

Min Choi

# Abstract

This project is a continuation from another project where a proof of feasibility was done for a smart water meter. This project will build upon the previous work and improve on it. Previously a single device containing Wi-Fi enabled microcontroller was produced as a final device. But it was determined that having a Wi-Fi was too power hungry to run on a battery for any meaningful length of time. Therefore, through this project I will produce a battery powered, low power device that can be used to measure the water usage. Then save the data online.

# Acronyms

DC	Direct Current
DNS	Domain Name System
GPIO	General Purpose Input / Output
IP	Internet Protocol
LAMP	Linux, Apache, MySQL, PHP
PCB	Printed Circuit Board
RPi	Raspberry Pi
UART	Universal Asynchronous Receiver/Transmitter
UFW	Uncomplicated Firewall
USB	Universal Serial Bus

# Symbols

Hz	Hertz
mA	milli-Ampere
MHz	Mega-Hertz
uA	micro-Ampere
V	Volts

# Table of Contents

Statement of Originality.....	i
Abstract.....	ii
Acronyms.....	iii
Symbols .....	iv
Table of Contents.....	v
List of Figures.....	vii
List of Tables .....	viii
Chapter 1 Introduction.....	1
1.1 Introduction.....	1
1.2 Conclusion .....	1
Chapter 2 Server.....	2
2.1 Background.....	2
2.2 Raspberry Pi.....	2
2.3 LAMP Stack .....	3
2.4 Website .....	3
2.5 Receiving and Viewing Data .....	4
Chapter 3 Hardware.....	7
3.1 Data Collector Unit.....	7
3.1.1 Design of Data Collector .....	8
3.1.2 Code for Data Collector.....	10
3.2 Data Logger Unit .....	11

3.2.1	Design of Data Logger.....	12
3.2.2	Code for Data Logger .....	14
3.3	Cost of Hardware .....	15
Chapter 4.....		16
Conclusion and Future Work.....		16
4.1	Conclusion .....	16
4.2	Recommendations for Future Work .....	16
4.2.1	Power Management Code .....	16
4.2.2	Data View Page .....	16
4.2.3	More Usage for Data Logger Unit.....	16
References.....		18

# List of Figures

Figure 1 - Raspberry Pi 3 Model B+.....	2
Figure 2 - Home Page of Smart Water Meter Website.....	4
Figure 3 - PHP Script to save data sent by Data Logger Unit to database .....	4
Figure 4 – Log Page to view the data .....	5
Figure 5 - PHP part if Log Page .....	6
Figure 6 - Schematic for Data Collector .....	8
Figure 7 - PCB for Data Collector (Left), Assembled Product (Right) .....	9
Figure 8 - Pinout for ATTINY85.....	9
Figure 9 - Data Collector Code for Demo .....	10
Figure 10 - Pulse Count Code.....	11
Figure 11 – Schematic for Data Logger.....	12
Figure 12 - PCB for Data Logger (Left), Assembled Product (Right) .....	12
Figure 13 - ESP-12E Pinout.....	13
Figure 14 - Code for the Data Logger Unit.....	14



# List of Tables

Table 1 - Power Usage at 5v Supply 1MHz Clock.....	7
Table 2 - Idle Current for Different Modes of HC-12 .....	8
Table 3 - Breakdown of Component Cost .....	15

# Chapter 1

## Introduction

### 1.1 Introduction

This project is based on previous proof of feasibility project. Which was focused on finding out whether or not collecting the usage data from existing water meters in New Zealand is possible. Through this previous project it was determined that modern water meters had the capability of reed-switch based pulse probe. From which the water usage can be acquired through the use of microcontroller. The purpose of this project was to build on the previous work done and improve upon it. The previous project had a few short falls. Which includes that because it was Wi-Fi based it use a lot of power. Second problem was that it relied on third party data logging servers, which imposed lots of restrictions such as logs per minute rules and total retained logs. Also, trouble shooting was difficult due to not having access to the code. In this project I have decided to try and remedy all the afore mentioned issues of the previous project.

### 1.2 Conclusion

In this project I have decided to build a low power outdoor device which will be powered by battery to collect the pulse data from the water meter. Then the outdoor device will send the acquired data to an indoor unit using radio frequency device. Finally, the indoor device will upload the data to a server where users can view the usage.

# Chapter 2

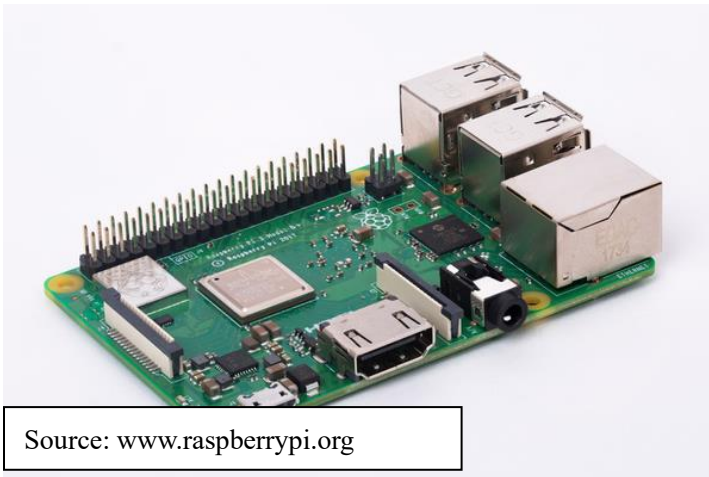
## Server

### 2.1 Background

The previous project relied on a third-party data logging server for saving and viewing the water usage data. For this project I have decided to build my own server to host the database and the website. I have decided to use Raspberry Pi (RPi) as my server device as it was low cost and low power device. The RPi will host Linux, Apache, MySQL, PHP (LAMP) server stack.

### 2.2 Raspberry Pi

Raspberry Pi is a single-board computer developed by Raspberry Pi Foundation. The device I have chosen was the Pi 3 model B+ variant which was the latest device at the time of purchase.



Source: [www.raspberrypi.org](http://www.raspberrypi.org)

Figure 1 - Raspberry Pi 3 Model B+

I have chosen this device because I have determined that it was powerful enough to perform the tasks that I require from it for this project. The RPi was chosen rather than my PC because it uses less energy when I need to let it run for a long time. Also, because this is my first time working on a server, I did not want my personal computer to be fully

exposed to the internet as I am not knowledgeable in securing the server computer.

## 2.3 LAMP Stack

I have decided to use the Linux, Apache, MySQL, PHP server. Commonly known as LAMP stack. The Raspberry Pi OS is a Linux operating system based on the Debian distro. This is the most versatile operating system available for the RPi. Apache HTTP Server (Apache) is a free and open-source webserver software. Apache is one of the most widely used Linux server software. MySQL is a relational database management system (RDBMS). A relational database is a database that is based on the relational model of data. It uses the Structure Query Language (SQL). SQL is a domain-specific language. It is used to communicate with the database. PHP is a general-purpose scripting language generally used for web development.

## 2.4 Website

After setting up all the LAMP stack I needed to open my server to be accessible through online as a website. First, I have bought the domain name “[www.minchoi.xyz](http://www.minchoi.xyz)” then I used Cloudflare as my name server. I have chosen Cloudflare because it gave me lots of security options and the ability to update my IP address automatically. This was necessary because I am using my home network for the server and the internet service provider doesn’t provide static IP for home users. Therefore, setting up a dynamic DNS was my only option. Which Cloudflare did provide among other useful security features. For better security I have set up the server to only be accessible through the domain name through Cloudflare using the UFW. I have also blocked access from outside New Zealand using Cloudflare.



Figure 2 - Home Page of Smart Water Meter Website

Above Figure 2 shows the home page of my server. It is very simple with just the name and 3 tabs. First is the homepage, second is where logged data are shown, and third page shows the author of the website. The website is working correctly, and I have verified that it can be accessed from external networks. Such as from AUT and on mobile data.

## 2.5 Receiving and Viewing Data

The data can be logged to the server using the “minchoi.xyz/post.php” the data logger unit is set up to send the data over Wi-Fi to that address.

```
<?php
//Set up login for phpmyadmin database
$hostName = "localhost";
$username = "USERNAME";
$password = "PASSWORD";
$dbName = "DATABASE";

$conn = new mysqli($hostName, $username, $password, $dbName);

if ($conn->connect_error) {
    die("Database Connection failed: " . $conn->connect_error);
}

//Get Date and Time
date_default_timezone_set('Pacific/Auckland');
$d = date("Y-m-d");
$t = date("H:i:s");

//Post Received data to phpmyadmin database
if(!empty($_POST['status']) && !empty($_POST['station']))
{
    $status = $_POST['status'];
    $station = $_POST['station'];

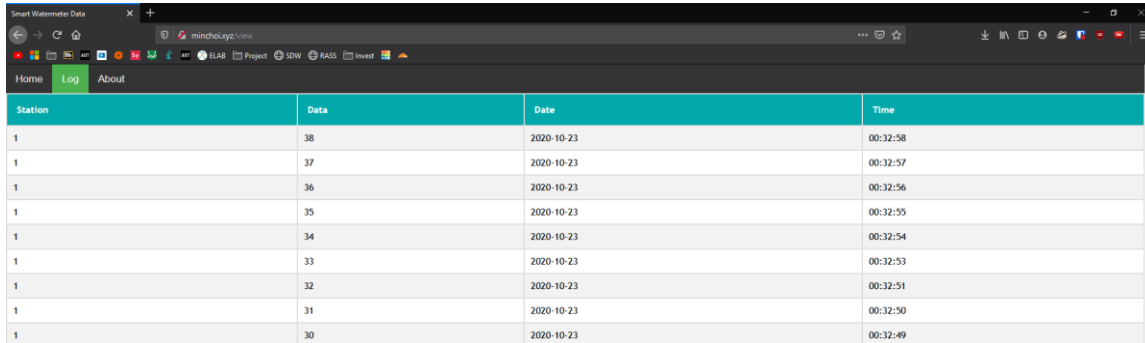
    $sql = "INSERT INTO logs (station, status, Date, Time)
VALUES ('".$station."', '>".$status."', '>".$d."', '>".$t."')";

    if ($conn->query($sql) === TRUE) {
        echo "OK";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}

$conn->close();
?>
```

Figure 3 - PHP Script to save data sent by Data Logger Unit to database

The logged data can be view from “minchoi.xyz/view”. This page shows a table of station, pulse data, date and time it was logged on.



Station	Data	Date	Time
1	38	2020-10-23	00:32:58
1	37	2020-10-23	00:32:57
1	36	2020-10-23	00:32:56
1	35	2020-10-23	00:32:55
1	34	2020-10-23	00:32:54
1	33	2020-10-23	00:32:53
1	32	2020-10-23	00:32:51
1	31	2020-10-23	00:32:50
1	30	2020-10-23	00:32:49

Figure 4 – Log Page to view the data

Figure 4 above shows screenshot of the log page where the user can view the logged data. It shows which station sent the data, what the data was and the date and time it was logged on. I have added the station number to the log because a single Data Logger Unit can be used for multiple Data Collector Units. If this project were to be commercialized only the Data Collector Units will need to be install at each water meter and a single Data Logger Unit can serve the whole street (within the working distance of the RF transceiver). Each Data Collector Unit will be assigned different station number. For example, the house number on the street.

```

<?php
    //Connect to database and create table
    $servername = "localhost";
    $username = "USERNAME";
    $password = "PASSWORD";
    $dbname = "DBNAME";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
?>

<div id="cards" class="cards">

<?php
    $sql = "SELECT * FROM TABLENAME ORDER BY id DESC";
    if ($result=mysqli_query($conn,$sql))
    {
        // Fetch one and one row
        echo "<TABLE id='c4ytable'>";
        echo "<TR><TH>Station</TH><TH>Data</TH><TH>Date</TH><TH>Time</TH></TR>";
        while ($row=mysqli_fetch_row($result))
        {
            echo "<TR>";
            echo "<TD>".$row[1]."</TD>";
            echo "<TD>".$row[2]."</TD>";
            echo "<TD>".$row[4]."</TD>";
            echo "<TD>".$row[5]."</TD>";

            echo "</TR>";
        }
        echo "</TABLE>";
        // Free result set
        mysqli_free_result($result);
    }

    mysqli_close($conn);
?>

```

Figure 5 - PHP part if Log Page

Above shows the PHP section of the Log Page. PHP is used to pull the data from the database so it can be displayed on a table.

# Chapter 3

## Hardware

### 3.1 Data Collector Unit

The design features for the data collector unit was that it needed to be able to count the number of pulse output from the probe then send it to the data logger unit via radio frequency (RF) communication. The most important design requirement for the data collector was that it needed to be efficient enough to run on a battery.

I have decided to use the ATTINY85-20PU(Tiny85) microcontroller as the brain of my data collector. I have chosen the Tiny85 because it had enough pins for my use case, it has a small form factor and it has low power consumption.

Table 1 - Power Usage at 5v Supply 1MHz Clock

Mode	Active	Idle	Power-down
Current	0.9mA	0.21mA	>1uA

For the RF device I have used HC-12. HC-12 is a transceiver based on the Si4463 low current transceiver chip made by Silicon Labs. I have chosen this device because of its low price and because it uses serial data, which is very simple to program.

HC-12 has what is called AT Command Mode. Where the user can send command lines to change Baud rate, select different operation mode, put the device to sleep etcetera.



Table 2 - Idle Current for Different Modes of HC-12

Mode	FU1	FU2	FU3
Idle Current	3.6mA	80uA	16mA

Table above shows the different idle current for the 3 selectable working modes. The HC-12 uses up to 200mA when transmitting and can use as low as 22uA when in sleep mode.

### 3.1.1 Design of Data Collector

All components in the Data Collector Unit can work safely at 5 Volts. Therefore, it only needs a single linear regulator which outputs 5 Volts DC. The input source can be any battery(s) above 6.2 Volts up to 20 Volts.

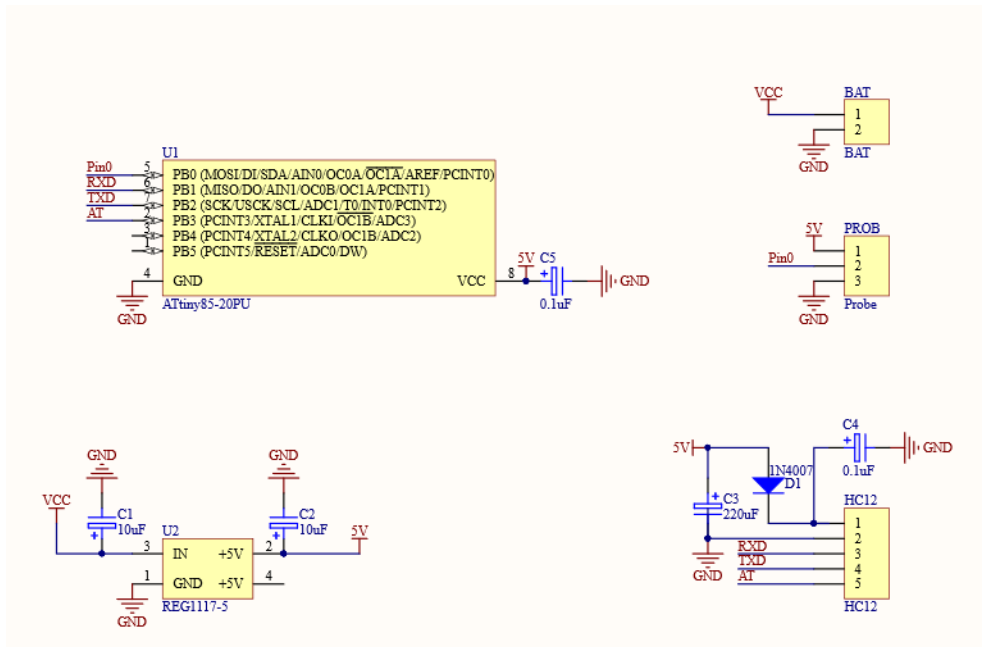


Figure 6 - Schematic for Data Collector

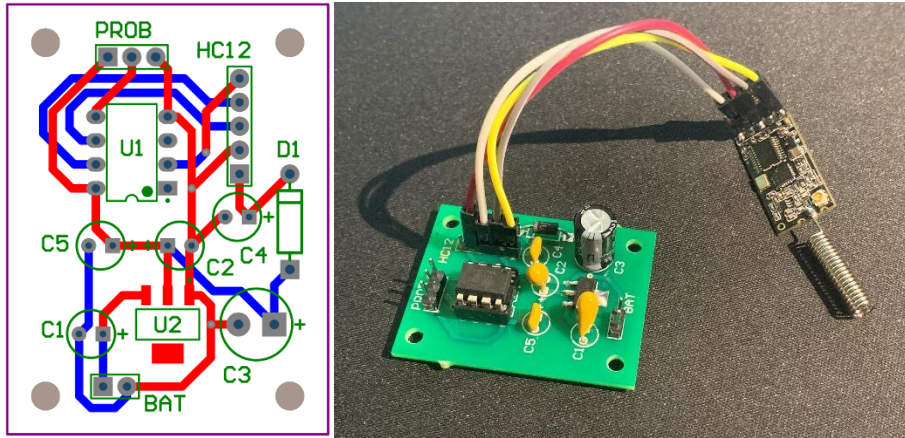


Figure 7 - PCB for Data Collector (Left), Assembled Product (Right)

The Linear regulator needed to handle a minimum of 201mA (200mA for HC-12, 1mA for ATTINY85). Due to COVID-19 I did not have much choice for components. I just chose the cheapest components that were available the fastest which met the current requirement.

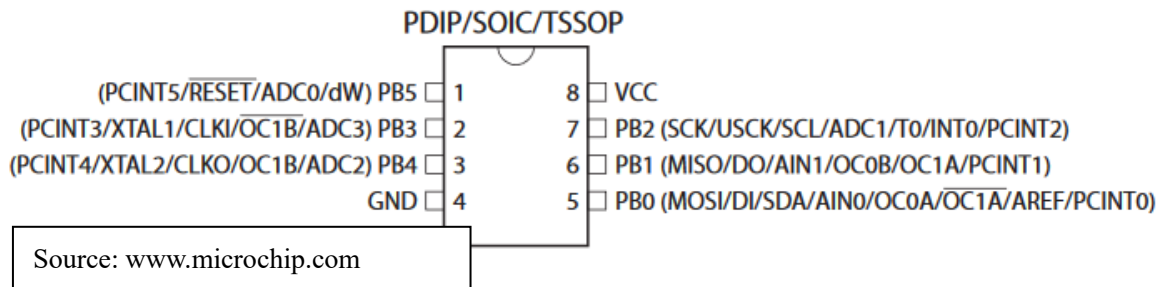


Figure 8 - Pinout for ATTINY85

PB0 is used as input from the probe. PB0 was chosen for this purpose because it is the interrupt pin. In the final code, when there is pulse from the probe detected on PB0 interrupt the microcontroller will count the falling edge of the pulse detected. PB1 is used as Rx pin (connects to Tx of HC-12) and PB2 is used as Tx pin (connects to Rx pin of HC-12) using the software serial library. Finally, the PB3 pin will be connected to the SET pin of the HC-12. When SET pin is set to low the HC-12 enters AT command mode where we can put the transceiver to sleep. This is very

useful feature as we want to keep the transceiver in sleep mode as much as we possibly can to maximize our battery life.

### 3.1.2 Code for Data Collector

Currently I do not have access to the water meter and the probe. So, I have set the collector unit to count every second, then transmit the count to the data collector.

```
#include <SoftwareSerial.h>

int count;
SoftwareSerial myserial(1,2);

void setup() {
    pinMode(3,OUTPUT);
    digitalWrite(3,HIGH);
    myserial.begin(4800);
}

void loop() {
    count++;
    delay(100);
    myserial.print(count);
    delay(1000);
}
```

Figure 9 - Data Collector Code for Demo

The ATTINY85 uses software serial library to communicate with the HC-12 using 2 digital pins. Software serial allows the microcontroller to use any of its digital pins as if they were a UART pins. This is very useful as the ATTINY85 does not have any UART. I did not want to get bigger microcontrollers just for UART and end up having wasted pins.

```

#include <SoftwareSerial.h>

const int pulsePin = 0;
volatile byte Count = 0;
int numInt = 0;
SoftwareSerial myserial(1,2);

void setup() {
    Serial.begin(4800);
    pinMode(pulsePin, INPUT_PULLUP);
    pinMode(3, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(pulsePin), handleInterrupt, FALLING);
}
void handleInterrupt() {
    Count++;
}
void loop() {
    if(Count>0)
    {
        Count--;
        numInt++;
        myserial.print(numInt);
    }
}
}

```

Figure 10 - Pulse Count Code

Figure 10 above shows the pulse count code without any power saving codes applied. I have yet to write the power saving code.

## 3.2 Data Logger Unit

The core of the data logger unit is ESP-12E based. ESP-12E is a variant of ESP8266, which is a very popular low-cost microcontroller with Wi-Fi made by Espressif Systems. I have chosen this device because I have previous experience using it. Unlike the data collector, the data logger did not have the constraint of needing to have low power usage. This is because the data logger will be an indoor device and be powered from the wall plug using a 5V USB charger through micro USB cable. The data logger unit will also consist of HC-12 so that the 2 units can communicate with each other via RF.

### 3.2.1 Design of Data Logger

The data logger unit requires 2 different voltages. 3.3 Volts for the ESP-12E and 5 Volts for the HC-12. Because this unit will be power by USB, I only needed to implement a linear regulator that drops 5 Volts input from USB to 3.3 Volts for the ESP microcontroller and directly feed the HC-12 transceiver form the USB input.

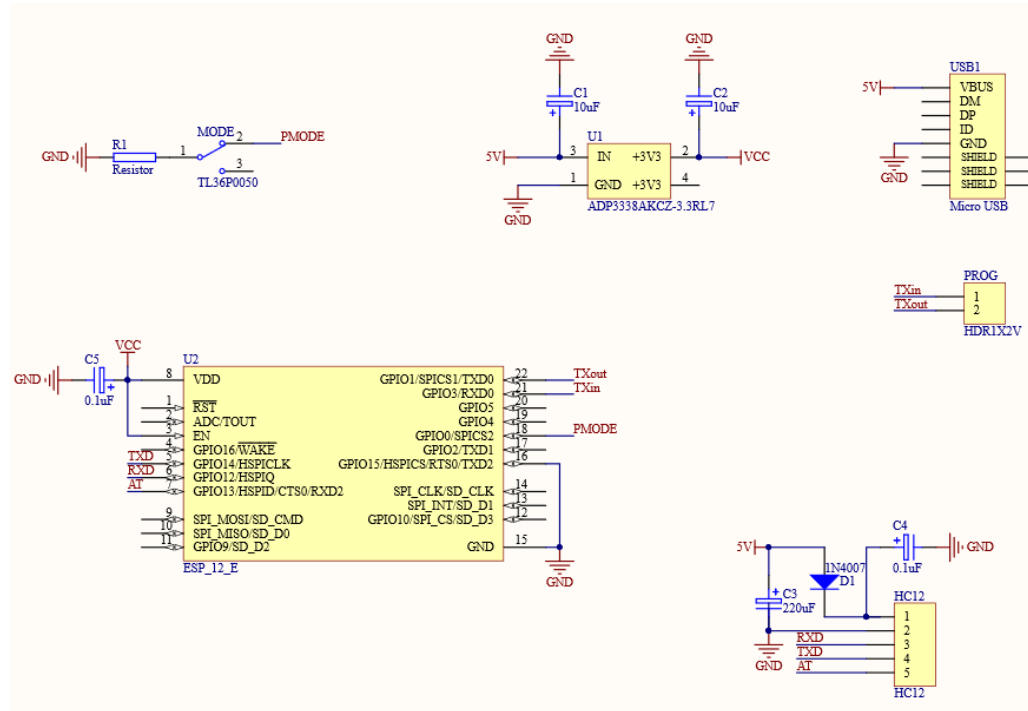


Figure 11 – Schematic for Data Logger

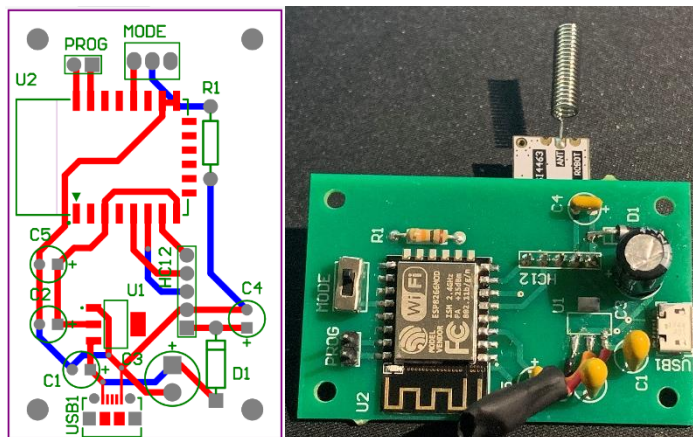


Figure 12 - PCB for Data Logger (Left), Assembled Product (Right)

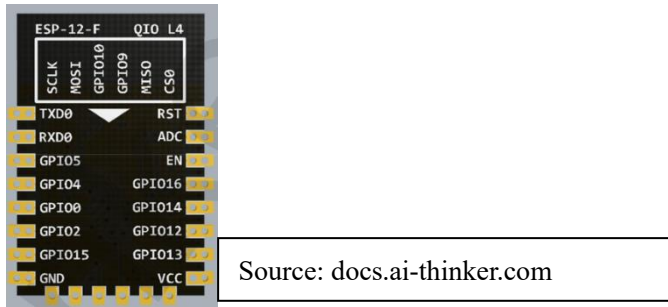


Figure 13 - ESP-12E Pinout

First, there are a few pins that we need to set to either low (ground) or high (3.3v). EN and VCC needs to be connected to 3.3 Volts and GND and GPIO15 needs to be connected to ground. Next, I have routed TXD0 and RXD0 pins to a pin header so I can use them to program the ESP-12E. to be able to program the ESP microcontroller you need to set the GPIO0 to low when booting. I have implemented a switch into the circuit so I can switch between programming mode and normal boot mode easily. Pins I have used to connect to HC-12 are GPIO12, 13 and 14. GPIO13 is connected to SET pin of the transceiver. GPIO12 and GPIO14 to RX and TX pins of the transceiver respectively.

### 3.2.2 Code for Data Logger

```

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <SoftwareSerial.h>

SoftwareSerial HC12(D5, D6);

const char *ssid = "SSID";
const char *password = "PASSWORD";
const char *host = "EXAMPLE.COM";
String Data="", station, postData;
byte incomingByte;

void setup() {
  pinMode(D7, OUTPUT);
  digitalWrite(D7, HIGH);
  delay(1000);
  Serial.begin(4800);
  HC12.begin(4800);
  WiFi.mode(WIFI_OFF);
  delay(1000);
  WiFi.mode(WIFI_STA);

  WiFi.begin(ssid, password);
  Serial.println("");

  Serial.print("Connecting");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  while (HC12.available()) {
    incomingByte = HC12.read();
    Data += char(incomingByte);
  }
  while (Serial.available()) {
    HC12.write(Serial.read());
  }
  delay(100);
  if (Data != ""){
    upload();
  }
}

void upload(){
  Serial.print(Data);
  HTTPClient http;

  station = "1";
  postData = "status=" + Data + "&station=" + station ;

  http.begin("http://EXAMPLE.COM/post");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int httpCode = http.POST(postData);
  String payload = http.getString();

  Serial.println(httpCode);
  Serial.println(payload);

  http.end();
  Data="";
}

```

Figure 14 - Code for the Data Logger Unit

Figure 14 above shows the code I used for the data logger unit. Because I am using the UART pins of the ESP-12E for programming I used software serial to communicate with the HC-12 transceiver.

### 3.3 Cost of Hardware

Table 3 - Breakdown of Component Cost

Part	Units (EA)	Price (\$)	Total (\$)
ATTINY85-20PU	1	2.16	2.16
ESP-12E	1	7.28	7.28
HC-12	2	5.56	11.12
TC1262-3.3VDBTR	1	0.88	0.88
NCP1117ST50T3G	1	0.83	0.83
MICRO USB CONN	1	1.60	1.60
10uF TANTALUM	4	1.45	5.8
100nF CERAMIC	4	0.16	0.64
220uF ELECTROLY	2	0.63	1.26
SWITCH	1	0.7	0.7
		TOTAL	\$32.27

Table 3 above shows the breakdown of all the components used for the 2 devices I have manufactured. This table does not include the PCB and shipping cost but does include 15% GST. I could not include the cost of the PCB because I used the 1 day turn over service. Which was horrendously expensive. But for manufacturing it will be below \$1 per device even for just 100s of units. And shipping can be ignored for bulk orders. So, I estimate that even at single unit price the devices will cost less than \$35 for both the Data Collector and Data Logger.



# Chapter 4

## Conclusion and Future Work

### 4.1 Conclusion

I have achieved the initial goal of manufacturing low cost low power outdoor data collector. I have also completed the indoor data logger device which receives the data from the collector via RF transceiver. I have also managed to set up a data logging server using Raspberry Pi and LAMP stack. The server is very stable currently. The longest time I have let it run is just over a week. I don't have any reason to suspect that the server will have any problem going forward.

### 4.2 Recommendations for Future Work

Future works that needs to be done are as follows.

#### 4.2.1 Power Management Code

Currently I only have the concept of power management for the battery powered data collector unit and have yet to implement the power management code into its program. So, this is the first future work that needs to be done.

#### 4.2.2 Data View Page

The data view page needs to be improved to be more user friendly. As of now it only displays the table of all data collected. I want to improve this by adding charts that can show the water usage by day, week, month etc.

#### 4.2.3 More Usage for Data Logger Unit

The Data Logger device has many potential usages with minimal changes. Most

cases only need the change of code for new use case. Any device that is built using the HC-12 can be interfaced with this Data Logger and be connected to the internet. Even for non-HC-12 RF device. You only have to change out to a compatible device.

# References

Figure 1 Image acquired from <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>

Figure 8 Image acquired from [http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf)

Figure 13 Image acquired from [https://docs.ai-thinker.com/\\_media/esp8266/docs/esp-12f\\_product\\_specification\\_en.pdf](https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf)

Websites used for research:

<https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/>

<https://randomnerdtutorials.com/esp32-esp8266-raspberry-pi-lamp-server/>

<https://circuits4you.com/2018/03/10/esp8266-nodemcu-post-request-data-to-website/>

<https://pimylifeup.com/raspberry-pi-port-forwarding/>

<https://www.electronics-lab.com/project/raspberry-pi-web-based-data-logger-using-mysql-php/>

<https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/>

<http://qqtrading.com.my/rf-433mhz-serial-communication-module-hc-12>

<https://www.esp8266.com/viewtopic.php?f=160&t=14367>

HTML and PHP sample codes from <https://www.w3schools.com/>