ANTI-FREEZE WATER PIPE SYSTEM

By

Rui Lan

Qinru Li

Zichen Liang

Final Report for ECE 445, Senior Design, Spring 2017

TA: Eric Clark

3 May 2017

Project No. 12

Abstract

Our project provides an automated solution to the frozen water pipe problem in the winter. Specifically, it can monitor the real-time temperature of the pipe, send the data to the user, and heat up the water pipe when necessary. Additionally, the project has a backup battery system which could be used when blackouts happen. By the end of the project, we manage to achieve the features listed above and create a reliable anti-freeze water pipe system.

Contents

1. Introduction

1.1 State of Propose

We notice the fact that during the winter when the temperature drops down, the water pipe will get frozen. The frozen water pipe has a high risk of bursting. This could be a huge problem if no one is at presence to deal with it. We believe it would be helpful if there is a device that can both heat up the water pipe externally and notify customers the current situation of the water pipe. Our device aims to monitor the temperature of the water pipe, heat up the water pipe, send the real-time data to customers and have a backup power source in case of blackouts.

1.2 Objectives

1.2.1 Goals and Benefits

- Reduce the risk of water damage
- Reduce the waste of water
- Increase the longevity of the plumbing utility
- Increase the ease of monitoring the house plumbing system

1.2.2 Functions and Features

- Monitor the temperature of the water pipe
- Provide the real-time temperature data to the user
- Heat up the water pipe when necessary
- Keep the system functioning when blackouts occur

1.3 Block Diagrams

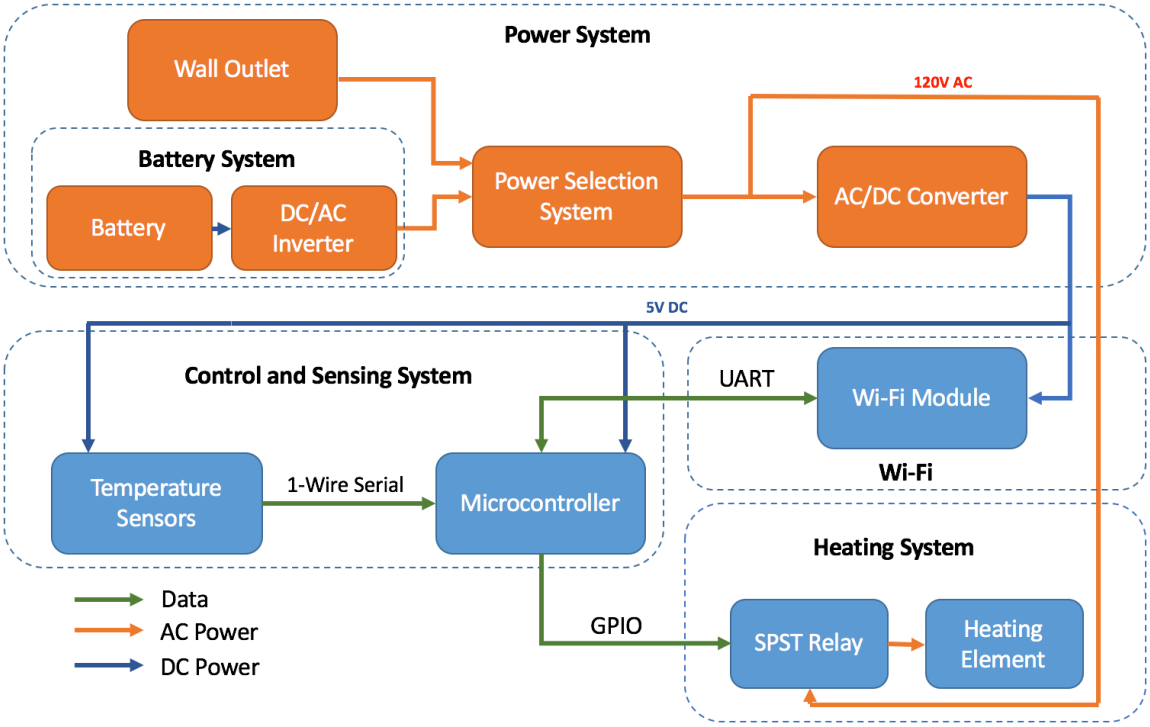

Figure 1 Overview of the Final System Design

2. Design

The following sections will elaborate the detailed design of our subsystems.

2.1 Control and Sensing System

The control and sensing system includes DS18B20 temperature sensors [1] and an ATmega328 microcontroller [2]. The system collects data from the sensors and performs all the decision-makings in our project.

2.1.1 Temperature Sensors

DS18B20 is a digital thermometer which provides a wide range of temperature measurement (-55˚C to 125˚C). It has a user-selectable resolution which provides 9-bit to 12-bit accuracy. The communication to the temperature sensors is supported by One-Wire bus protocol which requires minimal wiring. In our system, we use two sensors to measure the temperature of the pipe for a higher accuracy.

2.1.2 Microcontroller

ATmega382P microcontroller is a low power CMOS 8-bit microcontroller. It allows in-system programming (ISP) and supports multiple communication protocols like One-Wire bus protocol, UART protocol that we require. We connect it to an external 16MHz oscillator as its clock speed to improve the performance of the controller.

The controller collects data from the temperature sensors and sends these data to the Wi-Fi module. If the temperature is too low, it will turn on the heating system. In our final demonstration, we set the lower bound of the temperature as 8˚C and the upper bound as 10˚C. The reason why we set these thresholds is that in the actual experiments when we fill the pipe with ice and water, the temperature of pipe can only drop to around 6 to 8˚C. However, if the device operates during the winter, those thresholds are set to 4˚C and 7˚C as the following flowchart shows.

The software flowchart of the microcontroller is shown as Figure 2.

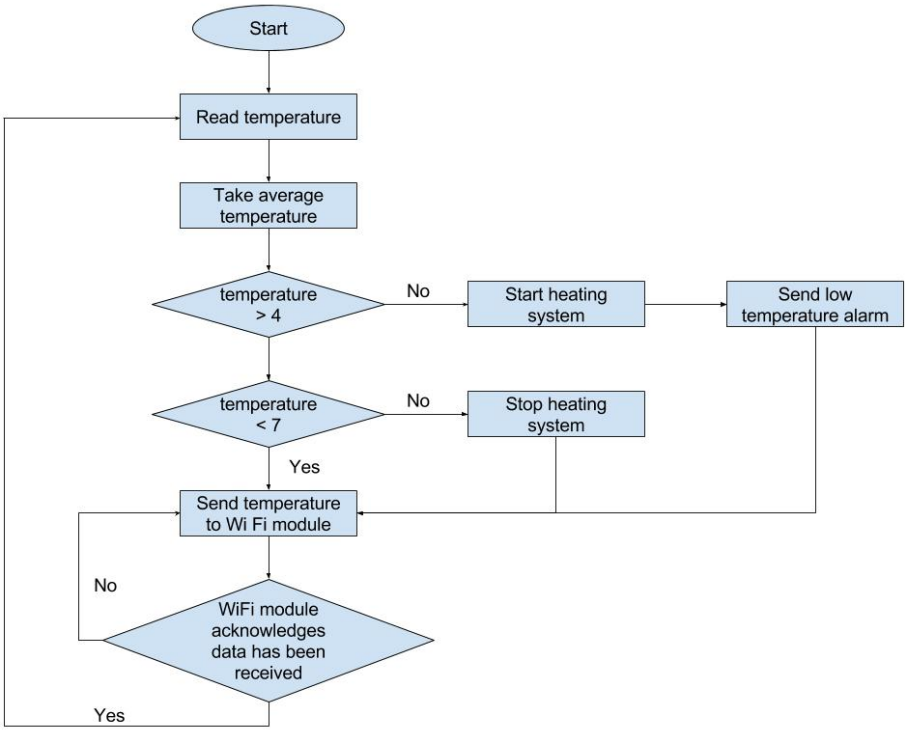

Figure 2 Software Flowchart

2.2 Wi-Fi

The Wi-Fi module is the device that sends data wirelessly to the user. Here we choose Adafruit HUZZAH ESP8266 chip [3] as the module. ESP8266 supports UART protocol which is its primary method of communication to the microcontroller. ESP8266 also supports MQTT protocol which is a software protocol where devices can publish messages to the internet (or subscribe messages from the internet). During the operation, the controller first sends the temperature data to the Wi-Fi module. The Wi-Fi module then converts these data into an MQTT message which is published to the internet. The user on the other side can receive these data from the server. For the demonstration purpose, we develop a Python program that receives messages from the Wi-Fi module and generates a real-time plot in the laptop. For the detailed Python code, please refer to Appendix D.

The overall circuit of the temperature sensors, the microcontroller, and the Wi-Fi module is shown as Figure 3 below.

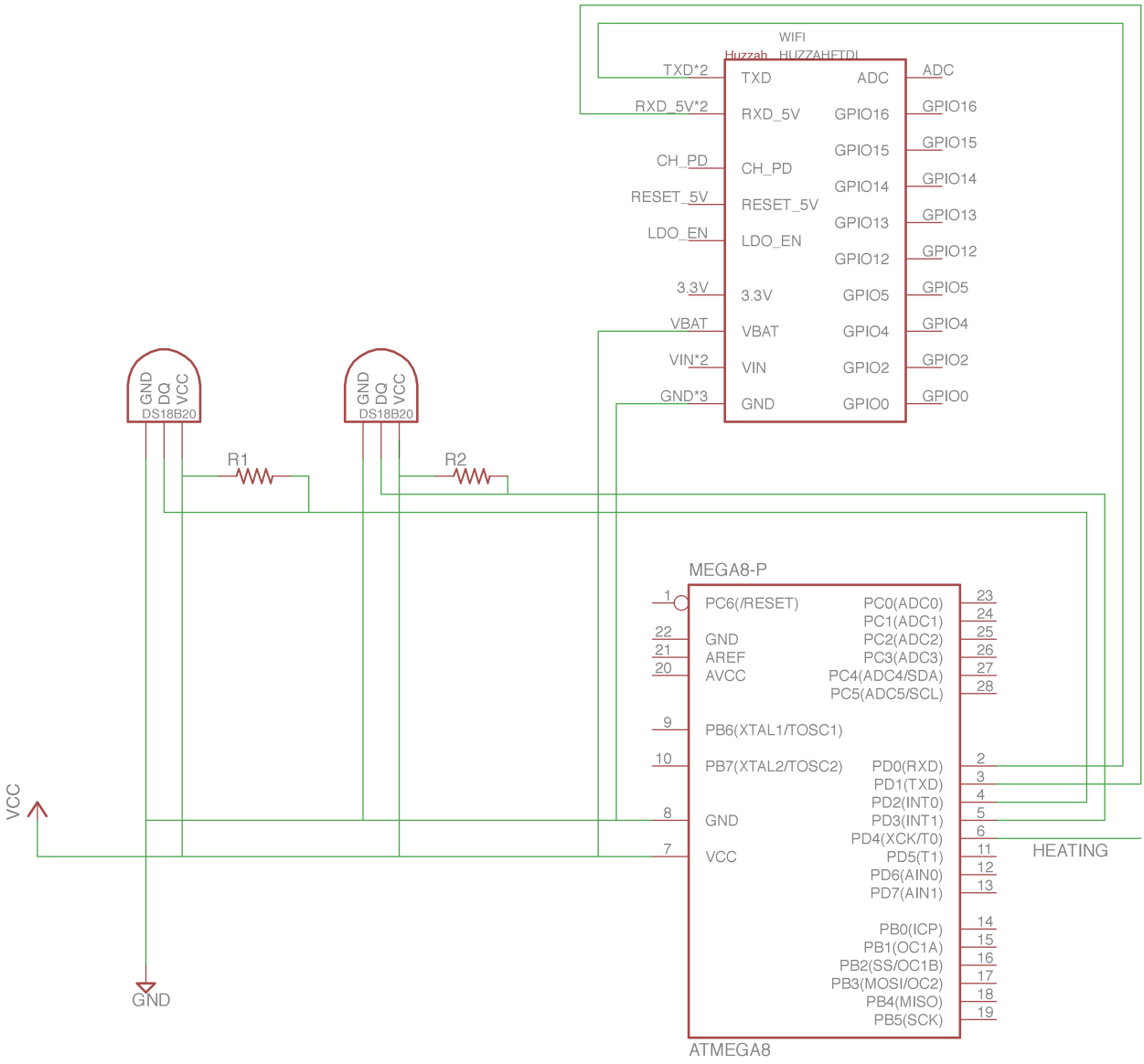

Figure 3 Schematic of the Digital System

2.3 Battery System

The Battery System is to ensure the anti-freeze water pipe system to continue working even there is a blackout (the wall outlet power is no longer available).

We use a 12V 7Ah rechargeable lead acid battery [4] for the backup power source. This battery gives us 12V DC voltage.

The battery is then connected to the voltage inverter [5], which inverts 12V DC from battery to 110V AC voltage. The inverted 110V AC voltage connects to the input of the power selection system.

2.4 Heating System

The heating system aims to provide thermal energy to the frozen water pipe. The source of the thermal energy is silicon rubber bendable heating element [6]. We use single-pole, single-throw (SPST) relay [7] to turn on/off the heating element, based on a certain temperature threshold. The 5V DC control signal coming from the microcontroller controls the operation of the SPST relay. Between the microcontroller and the SPST relay, there exists a driver that can block a larger amount of current to protect the microcontroller. The performance of this system can heat up infused ice and water pipe by 3˚C in 20 minutes. The Figure 4 below shows the connection of this system.

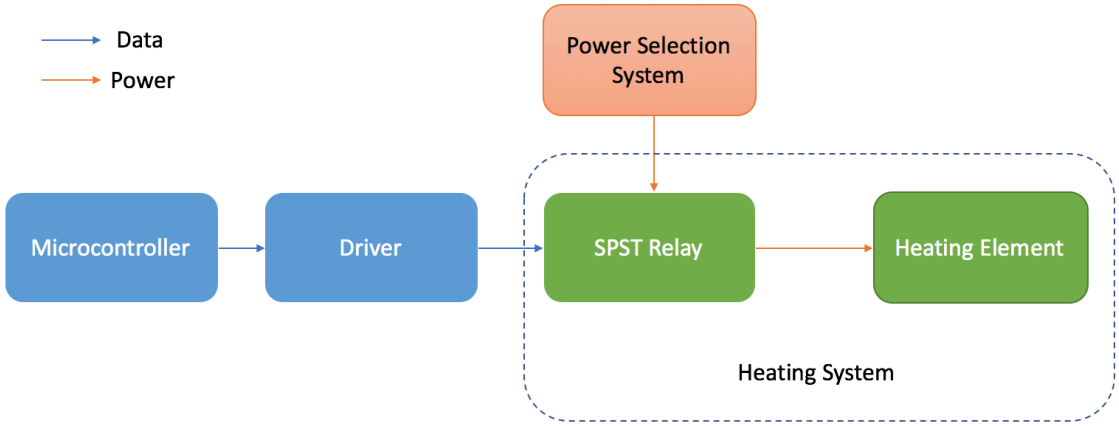

Figure 4 Block Diagram of Heating System

In terms of the physical connection of this system, the microcontroller, the driver, and the SPST relay are soldered on the PCB. Besides, the heating element is placed at the bottom of the water pipe.

2.4.1 SPST Relay

The SPST relay is a switch that controls the heating element. Based on the 5V DC control signal from the microcontroller, the SPST relay controls a high voltage (110V to 120V AC from the power selection), as the heating element requires.

2.4.2 Driver

Because the maximum output current limit of the microcontroller is 40mA whereas the minimum activating current limit of the SPST relay is 40mA, it would be troublesome if we connect them together directly. Therefore, we use a driver as a switch to drive a large amount of current by a small amount of current.

The driver contains a TIP120 transistor. Due to the potential damage that can occur when the SPST relay is de-energized, we have a snubber diode as shown below to prevent the transistor from being overloaded. The overall schematic of the drive is shown in Figure 5 in next page.

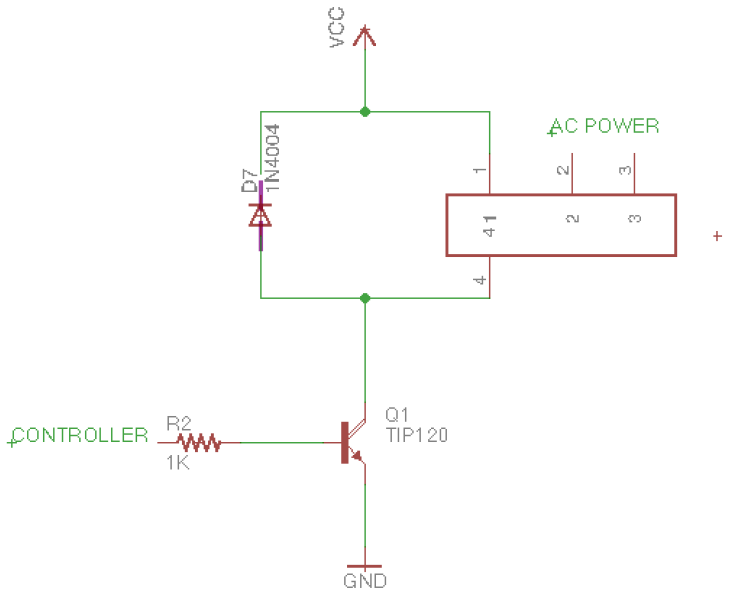

Figure 5 Schematic of Driver in Heating System

2.4.3 Heating Element

We use silicon rubber bendable heating element to heat up the water pipe. The heating element has 150W power, rated 120V AC voltage. In this situation, the current passing through the heating element can be calculated as:

$$I = \frac{W}{V} = \frac{150W}{120V} = 1.25A \quad (1)$$

However, this kind of heating element has certain downsides. It does not quite fit our water pipe. This caused a problem of heating inefficiency. Also, because of the heating inefficiency, we have a longer heating time than we expected.

2.5 Power Selection System

One of the major features of our project is the backup battery system which can be used when blackouts happen. We design this power selection system using the double-pole, double-throw relay (DPDT relay). The DPDT relay acts as a switch of power sources. With two inputs, it can select the power source available based on the existence of the coil voltage. When the coil voltage is activated, the relay chooses wall outlet as the source of power. Otherwise, the relay chooses the power from the battery system. Thus, we connect the voltage from the wall outlet as the coil voltage to fulfill our design requirements stated above. The output of power selection system goes to two parts. One is the AC/DC converter to generate 5V DC, and the other is the heating system to heat up our water pipe. Figure 6 in next page shows the connection of this system. We use the DPDT relay Z5357-ND [8] in our final circuit.

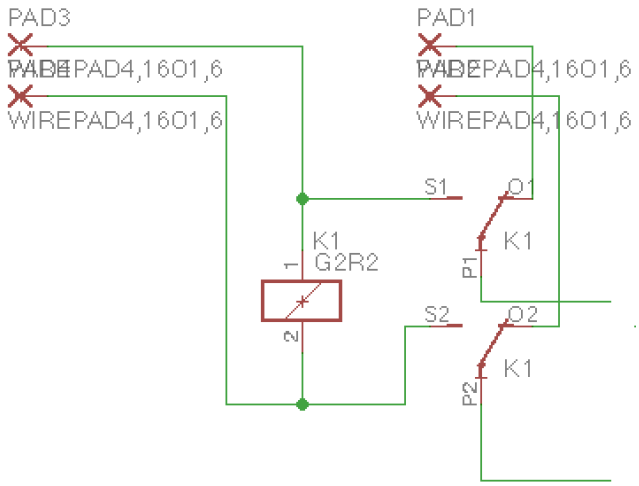

Figure 6 Schematic of Power Selection System

2.6 AC/DC Converter

Since all our digital components require 5V DC voltage, we need an AC/DC converter to generate the DC voltage. The AC/DC converter contains four parts: power transformer, rectifier, filter, and voltage regulator. The high-level block diagram of the AC/DC converter is shown below in Figure 7.

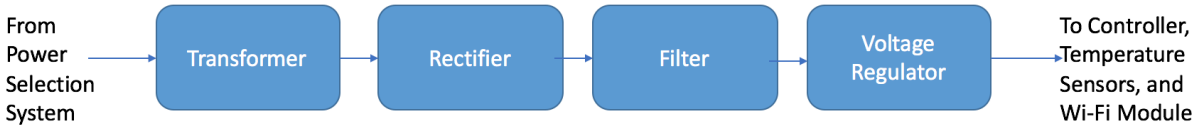

Figure 7 Block Diagram of AC/DC Converter

2.6.1 Transformer

The first part of our AC/DC converter is the transformer which gives us a lower AC voltage on its secondary side. We use the MT2115-ND transformer [9]. If there is 120V AC voltage on its primary side, the secondary side voltage would be 12V AC per the datasheet [9]. We calculate the coil relationship between two sides of the transformer.

$$\frac{N_1}{N_2} = \frac{V_1}{V_2} = \frac{120}{12} = 10 \tag{2}$$

The voltage of the secondary side of the transformer is fed into the rectifier.

2.6.2 Rectifier

The second part of our AC/DC converter is the rectifier. We use a full-wave rectifier because it can be applied to both half-cycles of the incoming AC voltage. Thus, we can achieve a higher accuracy than the

half-wave rectifier does. The full-wave rectifier contains four diodes as the Figure 8 shows. After the rectifier, the signal contains only the positive part and the signal of the whole period is rectified.

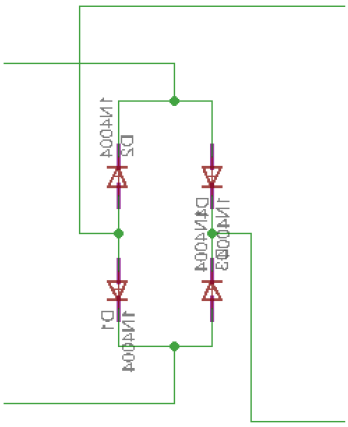

Figure 8 Circuit Diagram of the Full-wave Rectifier

2.6.3 Filter

After the full-wave rectifier, we need a smoothing capacitor to further smooth the voltage signal. The smoothing capacitor acts as a filter. The voltage regulator requires the filter for smoothing the DC voltage. We use a 470μF capacitor in our project.

2.6.4 Voltage Regulator

After the smoothing capacitor, the voltage goes into a voltage regulator to obtain a stable 5V DC voltage. We use LM317 [10] low-dropout (LDO) regulator in our project. As long as the input DC voltage remains in the certain range, we can get stable 5V DC from its output. The range is calculated as following.

$$2.5V + V_{out} = 2.5V + 5V = 7.5V \leq V_{in} \leq 32V \quad (3)$$

This feature of the voltage regulator enables us to get stable 5V for both wall outlet and battery system power supplies. Initially, a 5V linear regulator is used which only supports 100mA output current. After realizing this problem, we change the voltage regulator whose maximum current is as high as 1.5A. This is sufficient for the use of all digital components. After tests, other components of our project require 120mA current in total. The output of the voltage regulator goes to the temperature sensors, the controller, the Wi-Fi module, and the SPST relay.

According to the datasheet of LM317 [10], the circuit diagram of the voltage regulator is shown in Figure 9.

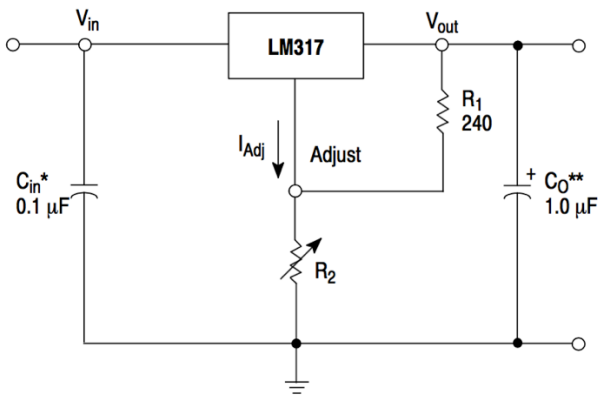

Figure 9 Circuit Diagram of the Voltage Regulator

The output voltage is determined by:

$$V_O = V_{REF}\left(1 + \frac{R_2}{R_1}\right) + I_{ADJ}R_2 \tag{4}$$

where in the equation (4), $V_{REF} = 1.25\text{V}$, $I_{ADJ} = 50\mu F$ (which can be ignored), and $R_1 = 240\Omega$.

R_2 can be calculated based on the output voltage $V_O = 5V$ as our expectation.

$$\frac{5}{1.25} = 1 + \frac{R_2}{240} \tag{5}$$

$$R_2 = 3\times 240 = 720\Omega$$

As the voltage regulator can generate a large current up to 1.5A, a diode is inserted between the output side of the regulator to the input side. This diode can protect the voltage regulator if there is a short circuit in our digital system. Also, the two external capacitors are required to increase the output stability. The capacitor on the input side is 0.1µF and the capacitor on the output side is 1µF.

The schematic of AC/DC converter is shown as Figure 10.

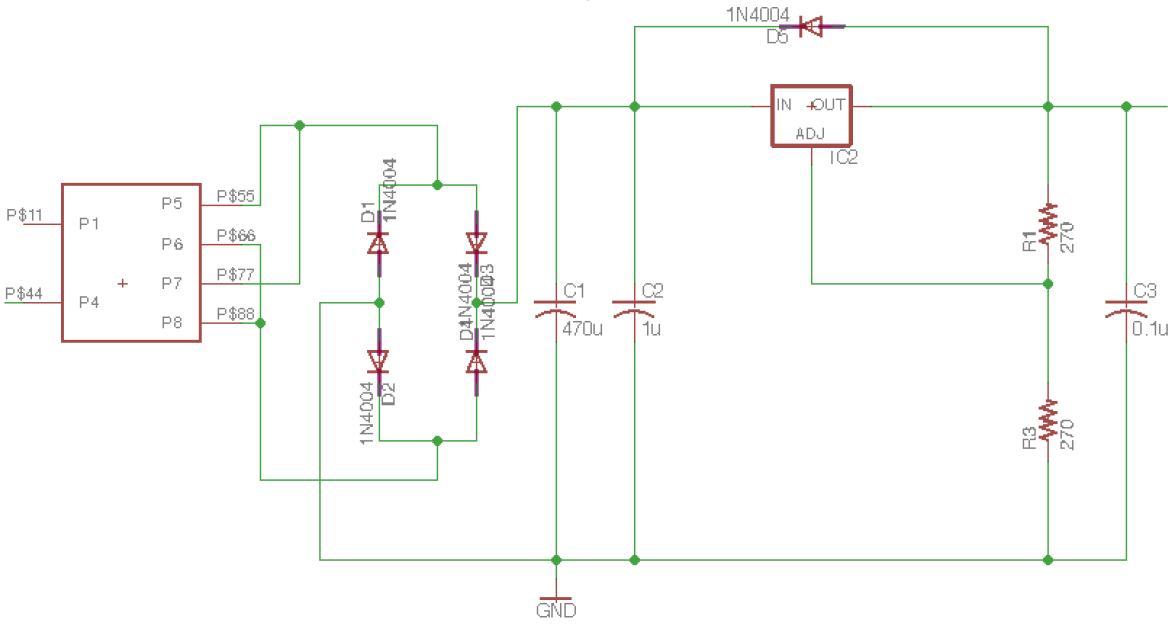

Figure 10 Schematic of the AC/DC Converter

2.7 Physical Design

We configure a water pipe on which the anti-freeze water pipe system can be easy assembled and dissembled. The water pipe used in the demonstration is a 5-feet-long copper water pipe. This can transfer the thermal energy generated from the heating element to the overall water pipe in a rapid and efficient way. At each end of the water pipe, we install a steel cap. These caps not only can stop the leakage of the water coming out of the water pipe, but also can provide a more realistic environment to simulate the condition of the water pipe when the ice and water are flowing through. Between two steel caps and the water pipe, there is a female adaptor on each side. These female adaptors build a connection between the water pipe and steel caps to facilitate the assembling of the caps.

Besides all the mentioned above, there are several specific considerations we take in the physical design of sensors.

Firstly, to prevent the external thermal influence from the environment, the sensor is wrapped in a bubble wrap for insulation purpose. Such wrapping also ensures a firmer attachment between the sensors and the pipe, which improves the measurement accuracy. In our experiment, the lowest temperature we can get from sensors without wrapping is 15˚C whereas we can get as low as to 5˚C with wrapping applied.

Secondly, sensors are placed in various positions for better measurement accuracy. Different locations in the pipe have different temperature. For example, ice and water are gathered in the bottom of the water pipe so the temperature there is close to zero degree. On top of the pipe, however, usually is air. The thermal conductivity of air is 0.024W/mK while that of water is 0.58W/mK [11]. Therefore, water is more sensitive to the change of temperature. Due to this reason, we place our sensors on multiple

positions close to the bottom of the water pipe. Taking an average of measurement data in multiple positions also reduces the probability of measurement noise and hence improves the measurement accuracy.

Finally, we introduce the water-proof design on the sensors to prevent potential electrical hazards. We apply two layers to all electrical connections on the sensors. The first layer is the heat shrink. It isolates all the electrical connections to avoid short circuits. The second layer is the water-proof glue which covers outside the heat shrink. It prevents any contact between the heat shrink and water. These two layers ensure that the sensors can still function properly even in the wet condition.

3. Design Verification

We create a specific requirements and verifications (R&V) table for individual systems in the project. During the demonstration, all the subsystems can function as the requirements of the R&V, except the battery system. All the subsystems can be integrated into a project which fulfills our expectation described in the Objective (1.2) part. Our detailed descriptions and verifications are in the following sections. Our Appendix A includes our full R&V table.

3.1 Control and Sensing System

3.1.1 Temperature Sensors

A working temperature sensor can provide an accurate real-time temperature data consistently and reliably to the controller. The temperature it measures must be close or equal to the temperature of the water pipe. For details on the specific requirements and testing procedures of the temperature sensors, please see Appendix A.

3.1.2 Microcontroller

A working controller can reliably and robustly control all sensors, the heating system, and the Wi-Fi with a small power consumption. It can process data and decide whether to turn on or off the heating system. To ensure the reliable operation, we divide the functionality of the controller into portions and test them individually. These potions include the digital output and input of the One-Wire bus protocol and UART bus protocol, and the control signal of the heating system. For details on the specific requirements and testing procedures of the controller, please see Appendix A.

3.2 Wi-Fi

A working Wi-Fi module can send real-time data to the server consistently and reliably. It publishes messages to the server with MQTT protocol. These messages will then be available to the user end as soon as possible. For details on the specific requirements and testing procedures of Wi-Fi, please see Appendix A.

3.3 Battery System

The output voltage of the battery system should be 110V AC with an error range of $\pm$5V. From the performance of the final demonstration, the result satisfies the requirement of the battery system. For the detailed R&V for this section, please refer to Appendix A.

3.4 Heating System

The goals of the heating system are to provide the thermal energy to the water pipe and to turn on/off the heating process based on the control signal from the microcontroller.

3.4.1 SPST Relay

The SPST relay must be able to turn on the heating element when the control signal is 5V DC and turn off the heating element when the control signal is around 0V. For the detailed R&V for this section, please refer to Appendix A.

3.4.2 Heating Element

Our heating element aims to provide sufficient thermal energy to the water pipe. The heating element must increase the temperature of the water pipe in 3˚C by 20 minutes. For the detailed R&V for this section, please refer to Appendix A.

3.5 Power Selection System

The aim of our power selection system is to choose from two power sources: wall outlet and battery system. Thus, the project should work when blackouts happen. According to the requirement, when the coil voltage is 120V AC, the DPDT relay selects power from the wall outlet. Otherwise, it chooses the power from the battery system. Our test results show our power selection system fulfills its requirements. For the detailed R&V, please refer to Appendix A.

3.6 AC/DC Converter

The overall aim of the AC/DC converter is to provide 5V stable DC at least 120mA for whole system to use. The result shows the AC/DC converter fulfills the requirements for both the wall outlet and the battery system power supplies. For the detailed R&V for this section, please refer to Appendix A.

3.6.1 Transformer

We expect that the transformer can step 120V AC voltage down to 12V AC on its secondary side. As the voltage supplied by the wall outlet is nearly 120V AC and the voltage provided by the battery system is 110V AC, we tolerate the error within the range of $\pm$2V. Based on tests performed, both power sources give us a voltage within the tolerance range.

3.6.2 Rectifier

We require the voltage signal after the full-wave rectifier to be fully rectified. We did the unit test on the rectifier by giving a 10V, 60Hz sinusoid signal to its input. And from the output, we found the voltage signal only contains the positive part and the signal of the whole period has been rectified. Figure 11 and 12 show the waveform of the test.

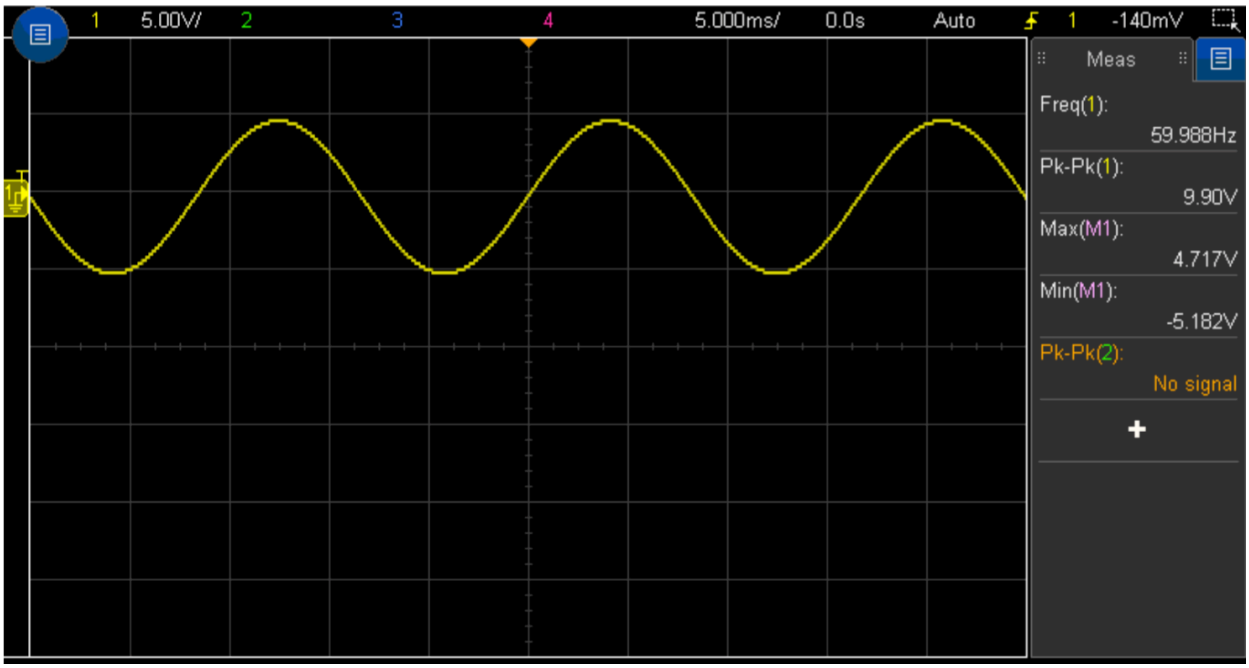

Figure 11 Input of Rectifier Test

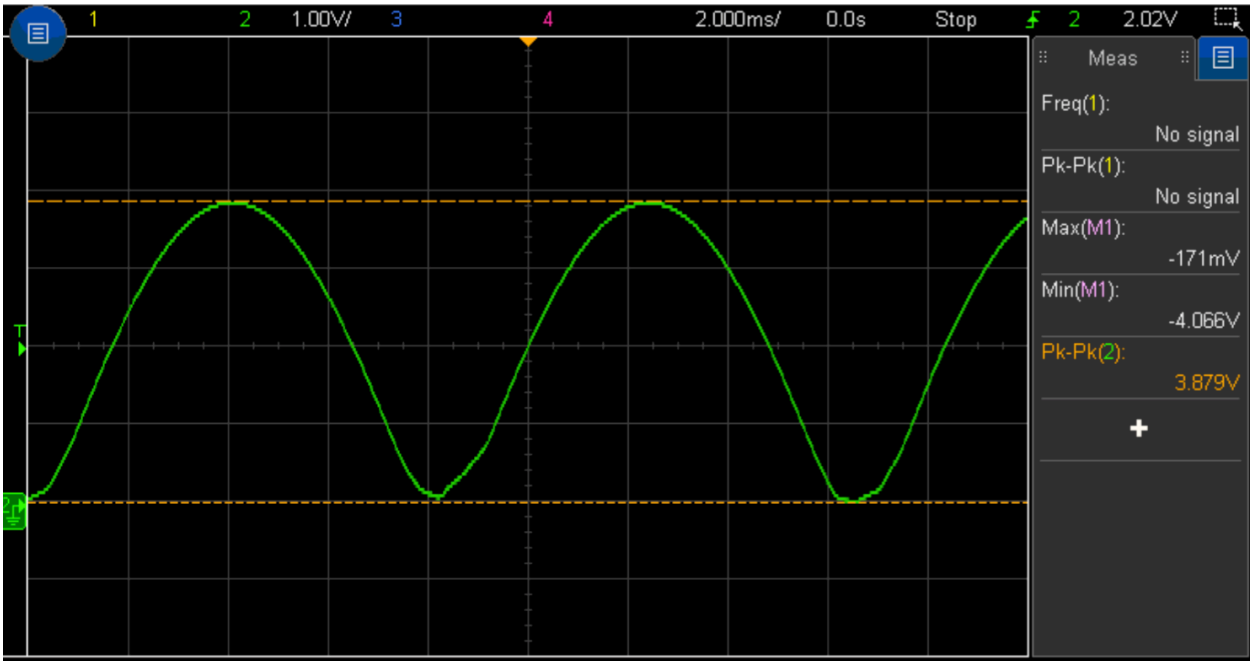

Figure 12 Output of Rectifier Test

3.6.3 Filter

The filter provides a smoothed DC signal for the LDO to regulate. This is important in the AC/DC converter design. Experiments show, after the filter, the voltage signal gets further smoothed.

3.6.4 Voltage Regulator

The voltage regulator is the final part of the AC/DC converter. With a certain range of the input, it can provide 5V DC voltage. Also, it offers enough current up to 1.5A, which fulfills the requirement that the AC/DC converter should provide at least 120mA. For the detailed R&V for this section, please refer to Appendix A.

4. Costs

The followings are the parts and labor costs of the project.

4.1 Parts

Table 1 Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
1-1/2" x 5' Type L Copper Pipe	Menards	36.48	36.48	36.48
1-1/2" PVC Male Adapter	Carlon	1.54	1.54	1.54
DS18B20 Temperature Sensor	Maxim	5.98	5.98	5.98
Z5357-ND DPDT Relay	Omron	11.81	11.81	11.81
MT2201-ND Transformer	Tamura	5.08	5.08	5.08
LM317 Voltage Regulator	Texas Instruments	0.62	0.62	0.62
DPCS10 Heating Element	Briskheat	129.95	129.95	129.25
EXP1270 Rechargeable Lead Acid Battery	ExpertPower	16.99	16.99	16.99
300W Inverter DC 12V to 110V AC	SNAN	24.99	24.99	24.99
G5NB-E SPST POWER RELAY	Omron	1.98	1.98	1.98
ATmega328P Controller	Atmel	1.96	1.96	1.96
HUZZAH ESP8266 Wi-Fi	Adafruit	9.95	9.95	9.95
Total		**247.33**	**247.33**	**247.33**

4.2 Labor

Table 2 Labor Costs

Engineer	Rate ($/hour)	Hours	Total ($)
Rui Lan	25	300	18750
Qinru Li	25	300	18750
Zichen Liang	25	300	18750
Total		**900**	**56250**

4.3 Total Cost

Total Cost = Parts + Labor = $247.33 + $56250 = $56497.33

5. Conclusion

5.1 Accomplishments

In the end, our project is capable of adequately heating up the water pipe, consistently sending messages to the customer, and promptly using backup battery system when necessary. The device can prevent the water pipe from freezing and being burst. We are able to design a power system that supplies constant power whatever happens. We also accomplish the interaction between physical sensing tools (temperature sensors, heating element) and electrical digital components (Wi-Fi module, control system). We put our efforts, knowledge, and diligence from past four years of undergraduate studies into this project, and we are pleased with the outcome of this project.

5.2 Uncertainties

We have several uncertain issues about the project. Firstly, the heating time is out of our expectation. We underestimate the heating efficiency of the heating element. This effect is due to the unmatched heating element to the water pipe.

Secondly, we are uncertain about the working time of the battery system. We cannot have an exact working time that guarantees the battery can work sufficiently. This is because the 12V DC battery is connected to the voltage inverter such that the 12V DC voltage is inverted to 110V AC voltage, and the inverted voltage is applied to other electrical components (AC/DC converter, control and sensing system, Wi-Fi module, and heating system).

5.3 Ethical considerations

The aim of this project is to develop an anti-freeze water pipe system that can reduce the waste of water and the rate of plumbing utility being busted. This device can increase the safety and welfare of the human races, and reduce the damage that may threaten the public. This follows the first code of the IEEE Code of the Ethics [12]:

"to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment."

Moreover, since the project is related to the infusion of ice and water into the pipe, this sort of behavior may disobey the lab code, we always infuse electric conducting materials into the pipe outside the lab and then conduct experimentations inside the lab. This action follows the ninth code of the IEEE Code of the Ethics:

"to avoid injuring others, their property, reputation, or employment by false or malicious action."

5.4 Future work

We believe that our product has the potential to protect water pipe from frozen and people from horrendous water damage. In the future, we plan to improve the heating element so that it fits better to the water pipe and supplies higher heating efficiency. We may also amend the battery system correspondingly to ensure that the entire system can operate normally in a sustainable amount of time during a blackout. Since the system involves high voltage, we will improve the electrical safety of the

entire system and guarantee that all the electrical connections are isolated from the outside. Together with these design improvements, we will work on reducing the costs and power consumption of the system and investigate the marketability of our product.

References

[1] Programmable Resolution 1-Wire Digital Thermometer, datasheet, Maxim, 2016. Available at: https://cdn-shop.adafruit.com/datasheets/DS18B20.pdf

[2] ATMEL 8-BIT MICROCONTROLLERWITH 4/8/16/32 KBYTES IN-SYSTEM PROGRAMMABLE FLASH, datasheet, Atmel, 2015. Available at: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A- 88PA-168A-168PA-328-328P_datasheet_Complete.pdf

[3] Adafruit HUZZAH ESP8266 breakout, datasheet, adafruit.com, 2016. Available at: http://www.mouser.com/ds/2/737/adafruit-huzzah-esp8266-breakout-932845.pdf

[4] ExpertPower 12V 7 Amp EXP1270 Rechargeable Lead Acid Battery, web page. Available at: https://www.amazon.com/ExpertPower-EXP1270-Rechargeable-Lead-Battery/dp/B003S1RQ2S/ref=sr_1_1?ie=UTF8&qid=1487726717&sr=8-1&keywords=12vdc+battery). Accessed May 2017

[5] SNAN 300W Car Power Inverter DC 12V to AC 110V with Dual AC Outlet and 4.8A Dual USB Charging Port, web page. Available at: https://www.amazon.com/dp/B019IFYMQU?psc=1). Accessed May 2017.

[6] 5 gallon plastic bucket heater- DPCS10, web page. Available at: http://www.gordosales.com/store/pc/5-gallon-plastic-bucket-heater-DPCS10-p3707.htm. Accessed May 2017.

[7] The Best Seller G2R, datasheet, Omron, 2016. Available at: http://www.omron.com/ecb/products/pdf/en-g2r.pdf

[8] Omron Electronics Inc-EMC Div G2R-2-AC110, datasheet, Omron, 2016. Available at: http://www.digikey.com/products/en?keywords=Z5357-ND%20

[9] 3FS-2XX series, datasheet, tamuracorp.com, 2009. Available at: http://www.tamuracorp.com/clientuploads/pdfs/engineeringdocs/3FS-2XX.pdf

[10] LM317L 3-Terminal Adjustable Regulator, datasheet, Texas Instrument, 2014. Available at: http://www.ti.com/lit/ds/symlink/lm317l.pdf

[11] Thermal Conductivity of common Materials and Gases, web page. Available at: http://www.engineeringtoolbox.com/thermal-conductivity-d_429.html. Accessed May 2017.

[12] IEEE Code of Ethics, web page. Available at: http://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A Requirement and Verification Table

Table 3 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
Temperature Sensor 1. The controller can use 1-wire protocol to communicate with the sensor and acquire the temperature data.	1.Verification of Time for Adjustment a. Attach the temperature sensor to a desk. b. Follow the transaction sequence as describe above to communicate with the temperature sensor. c. In the first 1-wire bus communication, use SKIP ROM Rom command and Convert T function command to instruct sensor to initialize the temperature conversion. d. In the second communication, use SKIP ROM Rom command and Read Scratchpad function command to read the temperature data. e. Connect the microcontroller to PC with Arduino board, once the microcontroller receives the data, print out the temperature data in Arduino console. f. Use a thermometer to measure the desk temperature. Compare its result with the printout in the Arduino console, the printout data should be close to the infrared thermometer data.	1. Y
Controller 1. The controller should sample the temperature data at least 1Hz. 2. Microcontroller can send temperature data to Wi-Fi module at least every 5 seconds. 3. The controller can turn on the heating system when the temperature is below 4°C±0.5°C and can turn it off when the temperature is above 7°C±0.5°C.	1. Verification of Sampling Data a. Microcontroller uses 1-wire bus protocol to read the temperature data from temperature from the sensor. Once it receives the 2-byte temperature from sensor, it turns on an I/O pin in the controller. Here we pick the I/O pin to be pin 8. b. Microcontroller uses 1-wire bus again to read the temperature. Once it receives it, it turns off the pin 8. c. Microcontroller keeps repeating the step a, b above. d. Measure the pin 8 output signal in the oscilloscope, the frequency should be at least 0.5Hz. 2. Verification of Sending Temperature Message to Wi-Fi module a. Use oscilloscope to probe the TX pin of controller and we should see the controller send the data packet in a period less than 5	1. Y 2. Y 3. Y

	seconds. 3. Verification of Correct Timing of Switching Heating System a. Emulating the 1-WIRE protocol by using another Arduino act as the DS18B20 slave b. Master issues the reset bit. Slave response with an acknowledge bit. c. Master issues the data read command. Slave doesn't acknowledge. d. Then Slave send 9-byte pre-programmed data to Master and. e. Repeat above steps for 3 scenarios: when the temperature data sent is below 4 degree, when the temperature data sent is between 4 to 7 degree, when the temperature data sent is above 7 degree. f. Connect a LED in series with the heating system. Observe its behavior, when the LED is off, it means the heating system is off. When the LED is on, the heating system is on. g. From the first to the second scenario, the LED should be on. From the third to the second scenario, the LED should be off.	
Wi-Fi Module 1. The Wi-Fi module must be able to send a real-time temperature data to the online server at least every 10 seconds. 2. The software in the PC can be able to receive the real-time temperature data sent from Wi-Fi module	1. Verification of Wi-Fi Module Frequency a. Use the temperature sensor to measure the room temperature. Use a thermometer to measure the room temperature as a reference. b. Use microcontroller to send the sensor's data to the Wi-Fi module. c. Use a stopwatch app designed in Python on a PC to read the data Wi-Fi module sent to the MQTT online server. d. The stopwatch app print out the data it receives and the time duration between it receives every data. e. The data should be close to the data measured by the thermometer in Step 1 and the time duration should be less than 10 seconds. 2. Verification of Software Client a. The controller will keep sending a counter to the Wi-Fi module which starts from 0. b. Create a MQTT software in python that subscribes to the topic Wi-Fi module	1. Y 2. Y

	publishes to. c. Once the software receives the data, it prints it out immediately in the console. An array of consecutive digital number should be displayed in the console.	
Heating System 1. The heater should increase the temperature of the water pipe at least 3℃ within 20 minutes. 2. The current of heating system should be less than 2A while operating.	1. Verification Process for Item 1: a. Use thermometer to measure the temperature of water pipe b. Power up the heating element, and start to time using the timer c. Ensure the temperature of the water pipe to increase at least 3℃ within 20 minutes 2. Verification Process for Item 2: a. Attach multi-meter to measure the current when passes the heating system b. Power up the heating element c. Ensure the current is always less than 2A through the heating process	1. Y 2. Y
Battery System 1. The battery system must work without power from wall outlet for at least 20 minutes.	1. Verification Process for Item 1: a. start the timer when the battery system starts b. stop the timer when the system stops c. make sure the operation time is at least 20 minutes	1. N
Power Selection 1. The relay should always choose source one (PIN2 and PIN7) when the coil voltage within the range from 110V AC to 120V AC.	1. Verification Process for Item 1: a. Connect source one (the one relay will choose if the coil voltage exists) of the relay to 120V (amplitude), 60Hz (frequency) sine wave (the power from wall outlet), and source two to ground. b. Connect the output of the relay to the multimeter in AC voltage mode. c. Connect the coil of the relay to the same power supply from wall outlet. d. Repeat step c for different coil voltage: 0V (ground), 110V (power from battery system). e. Ensure the multimeter will always reads nearly 120V AC voltage which is the power from wall outlet when the coil voltage is above 110V.	1. Y
AC/DC Converter 1. The output of AC/DC converter should be within the range of 5V±0.5V at the 220mA (the current our whole system needs)	1. Verification Process for Item 1: a. Attach 22Ω resistor as the load b. Attach a multimeter in DC voltage mode across the load c. Ensure the output voltage remains in 5V±0.5V	1. Y 2. Y

2. The output of the transformer should be AC voltage with in 12V±2V	2. Verification Process for Item 2: a. Attach a multimeter in AC voltage mode across the secondary side of the transformer b. Supply the primary side of the transformer with 115V AC c. Ensure the voltage on the secondary side remains in 10V to 14V	

Appendix B Program for the Controller

```
#include <SoftwareSerial.h>

#include <OneWire.h>


//-----------------Temperature Sensor Macro-----------------

#define SEARCH_ROM  0xF0

#define READ_ROM    0x33

#define MATCH_ROM   0x55

#define SKIP_ROM    0xCC

#define ALARM_SEARCH  0xEC


#define CONV_T      0x44   // Temperature conversion command

#define WRITE_SCRA  0x4E

#define READ_SCRA   0xBE

#define COPY_SCRA   0x48

//----------------------------------------------------------

#define CNTL_PIN 4

#define T_UP_BOUNCE   10  //28.5

#define T_LOW_BOUNCE  8  //27

#define LED_PIN 8
```

```
int led_state = 0;
const byte rxPin = 2;   //pin 2 = digital pin 2 = pin 4 in ATmega328
const byte txPin = 3;   //pin 3 = digital pin 3 = pin 5 in ATmega328

SoftwareSerial mySerial(rxPin, txPin);

//Temperature pin
OneWire ds1(10);   // on pin 10
OneWire ds2(9);
byte addr1[8], addr2[8];       //64-bit ROM addr for each temperature sensor

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  //Controller pin
  pinMode(CNTL_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  led_state = 0;
  Serial.print("----Start----\n");
}

void loop() {
  //search temperature sensor
  searchTempSensor();
```

```
/*
 * Create a byte array to hold the result of the data.
 * The first 2 bytes are from sensor 1. ret[0] = LowByte, ret[1] = HighByte
 * The second 2 bytes are from sensor 2. ret[2] = LowByte, ret[3] = HighByte
 */
int len = 4;
byte ret[len];
readSensor(ret);

// Send data to wifi
sendToWifi(ret, len);

// Control the heating system
int reso = 12;
double t1 = readTemperature(ret, reso);
double t2 = readTemperature(ret+2, reso);

double t_ave = (t1 + t2) / 2;

Serial.print("t1 = ");
Serial.print(t1);
Serial.print(" t2 = ");
Serial.print(t2);
Serial.print(" t_ave = ");
```

```
  Serial.println(t_ave);


  if(t_ave > T_UP_BOUNCE){
    Serial.println("Turn off heating system");
    digitalWrite(CNTL_PIN, LOW);
  }
  if(t_ave < T_LOW_BOUNCE){
    Serial.println("Turn on heating system");
    digitalWrite(CNTL_PIN, HIGH);
  }


  blink_led();
}

void sendToWifi(byte *data, int len){
  mySerial.write(data, len);
  //mySerial.flush();
}

void blink_led(){
  if(led_state == 0){
    digitalWrite(LED_PIN, HIGH);
    led_state = 1;
  }
  else{
```

```
    digitalWrite(LED_PIN, LOW);
    led_state = 0;
  }
}

//-----------------Temperature Sensor Function-----------------
/*
 * This function search the devices (temperature sensors) in our two OneWire bus.
 * Input:
 *    Nothing
 * Output:
 *    Nothing. But after the function called, the addr1 and addr2 will be updated.
 *    ds1 and ds2 will be ready to communicate with the sensors.
 */
void searchTempSensor(){
  int success = 0;
  while(!success){
    //Find available devices in the bus
    ds1.reset_search();  //Clear up all search memory so that we can re-search everything again
    if(!ds1.search(addr1)){
      Serial.print("No more addresses.\n");
      ds1.reset_search();
      continue;
    }
    ds2.reset_search();
```

```
  if(!ds2.search(addr2)){
    Serial.print("No more addresses.\n");
    ds2.reset_search();
    continue;
  }

  //Check the CRC
  if(OneWire::crc8(addr1, 7)!=addr1[7]){
    Serial.print("sensor_1 CRC is not valid!\n");
    continue;
  }
  if(OneWire::crc8(addr2, 7)!=addr2[7]){
    Serial.print("sensor_2 CRC is not valid!\n");
    continue;
  }

  success = 1;       // If reach here, we must have found the 2 sensors
 }
}

/*
 * The toplevel function that read temperature data from sensors. It will handle all the details of One-Wire protocal
 * The read temperature will be converted to double number and stores in the input *ret
 * Input:
 *    ret: A pointer that points to the beginning of a double array which is assumed preallocated.
```

```
 * Output:
 *    Nothing. But after the function called, temperature data will be stores in the ret
 */
void readSensor(byte *ret){
  byte i;
  byte present1 = 0, present2 = 0;
  byte data1[12], data2[12];

  //Request temperature conversion (Write-Wait-Read Pattern)
  sendInst(ds1, CONV_T, addr1);
  sendInst(ds2, CONV_T, addr2);
  delay(750);

  //Read temperature
  present1 = sendInst(ds1, READ_SCRA, addr1);   //Return 1 is a device responds with a presence pulse
  present2 = sendInst(ds2, READ_SCRA, addr2);

  for(i=0; i<9; i++){
    data1[i] = ds1.read();
    data2[i] = ds2.read();
  }

  assignData(ret, data1);
  assignData(ret+2, data2);
```

```
//  int reso = 12;
//  ret[0] = readTemperature(data1, reso);
//  ret[1] = readTemperature(data2, reso);
}

/*
 * Helper function for readSensor. It will assign the lowByte of temperature to the first byte of des
 * and the HighByte to the second byte of des
 */
void assignData(byte *des, byte *data){
  des[0] = data[0];   //LowByte
  des[1] = data[1];   //HighByte
}

/*
 * This function is the general function of sending a instruction to the temperature sensor.
 * Based on the 1-wire protocol, it will return 1 if device is present, 0 otherwise.
 * Input:
 *    dev: the one-wire bus where the data is sending through
 *    inst: the instruction we want to send (it is one byte long)
 *    addr: the address of device we want to send the instruction to.
 * Output:
 *    present: present bit received from device. 1 if device is present, 0 otherwise.
 */
int sendInst(OneWire dev, char inst, byte addr[]){
```

```
  int present = dev.reset();
  dev.select(addr);
  dev.write(inst);
  return present;
}

/*
 * This function will convert the input byte array 'data'
 * into temperature in degree Celsius
 * Input:
 *    data: a pointer points to a 2-byte array which stores lowerByte + higherByte (in sequence) of the temperature data.
 *    resolution: the resolution of the temperature data. We have 4 options: 9-12 bit
 * Output:
 *    a double number which is the temperature in degree Celsius.
 */
double readTemperature(byte *data, int resolution){
  int LowByte, HighByte, SignBit;
  int T;      // The temperature, it has to be the int type not the double b/c we want to do bitwise operation
  LowByte = data[0];
  HighByte = data[1];

  T = (HighByte << 8) + LowByte;
  SignBit = T & 0x8000;   //Read the Most siginficant bit
  if(SignBit){      // Negative temperature
```

```
  T = (T ^ 0xffff) + 1; // 2's compliment
}

double result;
switch(resolution){
  case 9:
    T = T & (~0x0007);    // Clear 0~2 bit to zero
    T = T >> 3;
    result = 0.5 * T;
    break;
  case 10:
    T = T & (~0x0003);    // Clear 0~1 bit to zero
    T = T >> 2;
    result = 0.25 * T;
    break;
  case 11:
    T = T & (~0x0001);    // Clear 0 bit
    T = T >> 1;
    result = 0.125 * T;
    break;
  default:
    result = 0.0625 * T;
    break;
}
```

```
 //If temperature is negative
 if(SignBit){
   result = -result;
 }
 return result;
}
```

Appendix C Program for the Wi-Fi Module

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>

#define BLUELED 2
#define REDLED 0

const byte rxPin = 12;
const byte txPin = 14;
SoftwareSerial mySerial(rxPin, txPin);

//-----------------WiFi Variable-----------------
const char* ssid     = "IllinoisNet_Guest";
const char* password = "password";
//MQTT Setting
const char* mqtt_server = "m13.cloudmqtt.com";//"mqtt.thingspeak.com";//"broker.hivemq.com";
const char* clientName = "ESP8266Client";
char* mqtt_user = "txzacqlp";
```

```
char* mqtt_password = "yXkhY-YQdmkS";


WiFiClient espClient;
PubSubClient client(espClient);


// track the last connection time
unsigned long lastConnectionTime = 0;
//// post data every 1 seconds
//const unsigned long postingInterval = 1L * 1000L;
//-------------------------------------------------


void setup(){
  Serial.begin(9600);
  //mySerial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 18568);
  //client.setServer(mqtt_server, 1883);


  pinMode(BLUELED, OUTPUT);      // Pin 2 control the blue LED
}


void loop(){


  // Read from controller
  double tempData[2];
```

```
  int alarm;
  readFromController(tempData, &alarm);

  // Ensure the network connection
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  Serial.print("Sending data\n...");
  Serial.print("t1 = ");
  Serial.print(tempData[0]);
  Serial.print(" t2 = ");
  Serial.println(tempData[1]);

  double ave_t = (tempData[0] + tempData[1])/2;
  if(mqttpublish(ave_t)){
    digitalWrite(BLUELED, HIGH);
  }

  delay(900);   // read a bit faster than controller
  digitalWrite(BLUELED, LOW);   // Close the Blue LED
}

/*
```

```
 * This function will read data from controller who is sending the temperature and alarm data
 * through the UART protocol. This function will also convert these data from byte to digital number
 * and store the values into the input point *tempData and alarm respectively.
 * Input
 *    tempData: a pointer that points to an array of double. It assumes that the data is preallocated.
 *    alarm: a pointer points to a int. It assumes that it is preallocated.
 * Output
 *    Return nothing. But after function called, *tempData and alarm will be updated.
 */
void readFromController(double *tempData, int *alarm){
  int len = 4;
  byte data[len];
  int flag = 0;

//  while(!Serial.available()){
//  }
//  Serial.readBytes(data, len);
//  flag = 1;

  if(Serial.available()){
    Serial.readBytes(data, len);
    flag = 1;
  }

//
```

```
//  if(mySerial.available()){
//    mySerial.readBytes(data, len);
//    flag = 1;
//  }


  if(flag){
    int reso = 12;
    tempData[0] = readTemperature(data, reso);
    tempData[1] = readTemperature(data+2, reso);
  }
}


/*
 * This function will convert the input byte array 'data'
 * into temperature in degree Celsius
 * Input:
 *    data: a pointer points to a 2-byte array which stores lowerByte + higherByte (in sequence) of the temperature data.
 *    resolution: the resolution of the temperature data. We have 4 options: 9-12 bit
 * Output:
 *    a double number which is the temperature in degree Celsius.
 */
double readTemperature(byte *data, int resolution){
  int LowByte, HighByte, SignBit;
  int T;      // The temperature, it has to be the int type not the double b/c we want to do bitwise operation
```

```
LowByte = data[0];
HighByte = data[1];

T = (HighByte << 8) + LowByte;
SignBit = T & 0x8000;  //Read the Most siginficant bit
if(SignBit){     // Negative temperature
 T = (T ^ 0xffff) + 1; // 2's compliment
}

double result;
switch(resolution){
 case 9:
   T = T & (~0x0007);    // Clear 0~2 bit to zero
   T = T >> 3;
   result = 0.5 * T;
   break;
 case 10:
   T = T & (~0x0003);    // Clear 0~1 bit to zero
   T = T >> 2;
   result = 0.25 * T;
   break;
 case 11:
   T = T & (~0x0001);    // Clear 0 bit
   T = T >> 1;
   result = 0.125 * T;
```

```
    break;
   default:
    result = 0.0625 * T;
    break;
  }

  //If temperature is negative
  if(SignBit){
   result = -result;
  }
  return result;
}

//-----------------WiFi Function-----------------
/*
 * This function will set up the WiFi connection in ESP8266
 * WiFi will be set up after function called
 * Input: None
 * Output: None
 */
void setup_wifi(){
  delay(10);
  Serial.print("\nConnecting to ");
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

/*
 * This function will try to reconnect ESP8266 to the WiFi
 * If it continuously fails to connect to the WiFi, it will delay for a while before trying again.
 */
void reconnect() {
  // Loop until we're reconnected
  int retryDelayTime = 2000;

  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(clientName, mqtt_user, mqtt_password)) {
```

```
    Serial.println("connected");
   } else {
    Serial.print("failed, rc=");
    Serial.println(client.state());
    Serial.println("sleep...try again later");
    delay(retryDelayTime);
   }
  }
}

/*
 * This function handle the MQTT publishment. It takes a temperature and alarm
 * as input and will publish these data onto ThingSpeak broker.
 * Input:
 *    temperature: the temperature data we want to publish
 *    alarm: the alarm data we want to publish
 * Output:
 *    Return 1 if pulish succeeded, 0 otherwise.
 */
int mqttpublish(double temperature){
  char msgBuffer[1024];
  // Convert data from
  dtostrf(temperature, 4, 3, msgBuffer);    // dtostrf(double #, the width of string, precision, buffer)
assume buffer has large enough memory
  Serial.println(msgBuffer);
```

```
 // Construct topic

 char* topic = "ece445/waterpipe/esp8266";



 // Publish data to HiveHQ

 int ret = client.publish(topic, msgBuffer);

 if(ret){

  Serial.println("MQTT: publish msg succeeded.");

 }



 // Note the last connection time

 lastConnectionTime = millis();



 return ret;

}
```

Appendix D Program in User-end (Python)

```python
import paho.mqtt.client as paho
import matplotlib.pyplot as plt
import numpy as np

numData = 100

def on_connect(client, userdata, flags, rc):
    print(""CONNACK received with code %d." % (rc))

# This function is called once the broker has responded to a subscription request
def on_subscribe(client, userdata, mid, granted_qos):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

def on_message(client, userdata, msg):
    # print("Receive from Topic:" + msg.topic + "\nQoS:"+str(msg.qos) + "\nMsg:" +
str(msg.payload))
    global temp
    global cnt
    if cnt == numData:
        cnt = 0

    upperBound = 10
    lowerBound = 8
    temp[cnt] = float(msg.payload)
```

```
    print("Water pipe temperature: ", temp[cnt])
    if temp[cnt] < lowerBound:
        print("Alert! Water pipe temperature is too low!")
    if temp[cnt] > upperBound:
        print("Water pipe temperature is heated above the required threshold.")
    cnt += 1

mqtt_user = "txzacqlp"
mqtt_password = "yXkhY-YQdmkS"

client = paho.Client()
client.on_subscribe = on_subscribe
client.on_message = on_message

client.username_pw_set(mqtt_user, mqtt_password)
client.connect("m13.cloudmqtt.com", 18568)
# client.connect("broker.hivemq.com", 1883)

subTopic = "ece445/waterpipe/esp8266"
client.subscribe(subTopic)

ranlow = 5; ranup = 14
time = np.arange(numData)
temp = np.zeros(numData)
cnt = 0
plt.axis([0,numData,ranlow,ranup])
plt.ion()
# plt.scatter(time, temp, c='lightgreen')
plt.xlabel("Time Frame (per second)")
plt.ylabel("Water Pipe Temperature")
plt.title("Real Time Water Pipe Temperature")

# client.loop_forever()
client.loop_start()
while True:
    plt.scatter(time, temp, c='lightgreen')     # you can pick color at here
https://matplotlib.org/users/colors.html#cn-color-selection
    plt.pause(0.5)
```

Appendix E PCB Layout