Timing and Control for Distributed Hot Water System

Lance Leber
Electrical Engineering
University of Alaska
Anchorage, AK
lanceleber@gmail.com

Greg Schmidt
Electrical Engineering
University of Alaska
Anchorage, AK
gbschmidt@alaska.edu

Brittany Boring
Electrical Engineering
University of Alaska
Anchorage, AK
baboring@alaska.edu

***Abstract*— This project involves the design and build of a timing and control system for a distributed hot water system. The hot water system is 4 separate electric hot water heaters that share a single 15A breaker. The control system sets a hierarchy, monitors and regulates the heating of all four hot water heaters.**

Keywords—timing; control; heater; Auino

I. INTRODUCTION

Alaska is a large and diverse state. The approximate population of Alaska in 2015 was 737,625 people, which is spread throughout 663,300 square miles [1]. The vast size of the state warrants unique challenges that most other states do not face. Rural Alaska consists of more than 200 communities. 34 of these communities are considered unserved communities, where more than 55% of houses are not connected to a piped septic and well, or a covered haul system [2].

Currently, there are four different water and sewer systems in rural Alaska: washeterias and central watering points, individual wells and septic systems, water and sewer truck and piped water and sewer systems. Water and sewer truck, along with piped water is very expensive to construct and maintain. Individual wells are often impractical because of soil conditions and/or contamination issues. Washeterias and central watering points allows access to fresh water, but people must collect and transport it themselves. This means elementary health benefits such as flushing toilets are not possible.

Professor Dotson of the University of Alaska-Anchorage has decided to take on Alaska's clean water problems. A solution to rural Alaska's water and sewer problems is to design and build an in-house, self contained, water treatment system. This is just what Professor Dotson is doing. This system will allow the user to bring fresh water into the system. The water treatment system will then separate gray water from recyclable water and filter and reuse as much as possible. The design for the arrangement includes four separate hot water heaters. One heater each for the kitchen and bathroom sinks (2.5 gallon heaters), one for laundry and one for a shower (4 gallon heaters). We have been tasked with building a timing and control system for the heaters.

Our objective is the development of a timing and control system that will prioritize, heat and manage all four heaters simultaneously.

II. DESIGN

We were given some specific design requirements to work with. The given requests are as follows:

- At least 3 heaters included
- 2 heaters must be of different gallon size
- All heaters share single 20A or less breaker
- Control system must follow a set priority and manage any interrupts
- Continually cycle through to maintain water temperature when not in use

We chose heaters based on the amount of current draw during use. The heaters we chose run on less than 12A of current. The heaters can share a 20A breaker or even a 15A breaker in accordance with the national electric code. We designed our system around a 15A breaker. The design requirements entail that each of the hot water heaters share a single breaker because of this only one heater will be on at any given time.

We designed a control board, which physically did the power switching of the heaters. We chose solid-state relays (SSR) as our switching device. There are cheaper options than using SSR's, such as a single pole, double throw switch. We felt the cost of SSR's was justified because of the life span of the devices. SSR's generally outlast the equipment they are installed to. SSR's do not have any moving parts making them more durable and have a longer service life. This was

important because switching between each unit can happen up to 4320 times a day which would cause severe degradation on moving parts..

Each heater power cord is attached to an SSR through a power outlet. All power outlets are run to a single power switch. We chose to use a simple light switch as the total power switch rated to 15A. A light switch was chosen because it is easy to use and inexpensive. The power outlets are connected to the SSR's which are controlled by an Ardiuno. Figure 1, shown below is a schematic of the control board.

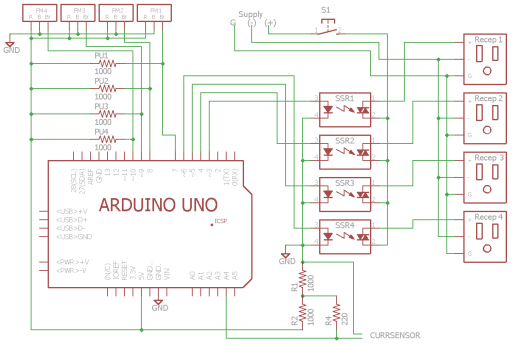


Fig. 1. Arduino Control Flow Chart

We chose an Arduino to be the central control unit of the entire system. The Arduino was chosen based on the low cost, previous experience, customizability, and the ease of use. The outlets are labeled 1-4. The Arduino is set up so that it follows the same hierarchy as the marked outlets. Outlet 1 will take first precedence followed by outlet 2 and so on. Heaters 1 and 2 are 4-gallon heaters, assuming that the larger heaters will be of more importance to the user. The user can select which heater is plugged into which outlet, thereby customizing priority.

Attached to each heater is a flow meter, every 5 seconds the Arduino checks for flow from each of the heaters. One 4 gallon heater and one 2 gallon heater has a thermistor attached to the inside of the power dial. The thermistor allows us to collect data on the heaters temperature for testing purposes only.

When the power is first switched on, the Arduino will switch power to Heater 1. If there is no flow detected it will chronologically power heaters 1-4 by switching the power to each heater through the relays. If flow is detected in heater 1, but heater 2 requests to come on the Arduino will determine that heater 1 has priority and keep the system stable. However, if heater 2 requests to come on and there is not flow in heater 1, then the Arduino gives priority to heater 2. The power will be switched, through the relays, from its current location to heater 2 allowing that hot water heater to heat.

The flow chart, shown below, in Figure 1, symbolizes the code which the Arduino follows in determining priority and switching of the power. Appendix A contains the hard code, which is written to the Ardiuno.

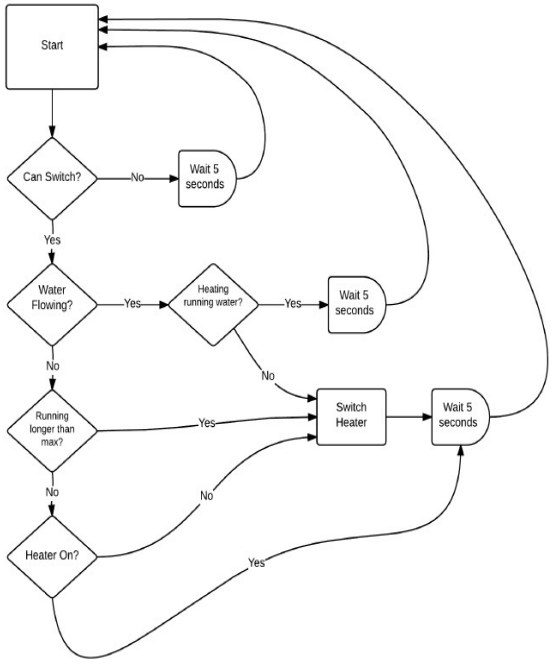


Fig. 2. Arduino Control Flow Chart

We designed and milled a circuit board to ease connectivity of the flow meters, current clamps, and outputs of the relays to the Ardiuno. The board was designed using Eagle. The design schematic is shown below in Figures 3 and 4.

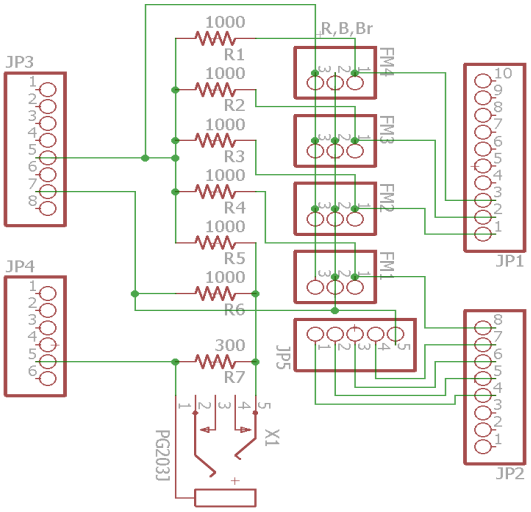


Fig. 3. Circuit Board Schematic

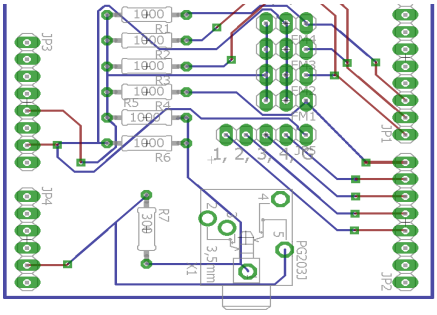


Fig. 4. Circuit Board Schematic

III. BUILD

The build of the circuitry was straightforward and went fairly well. Four power outlets, one for each heater is wired to a single switch just as we designed. The power outlets then go to the SSR's. The SSR's are connected to the circuit board and Ardiuno. The Ardiuno monitors the flow in each heater through flow rate sensors. Figure 5, shown below, is our final build of the timing and control board. See Appendix B for entire parts list of complete build.

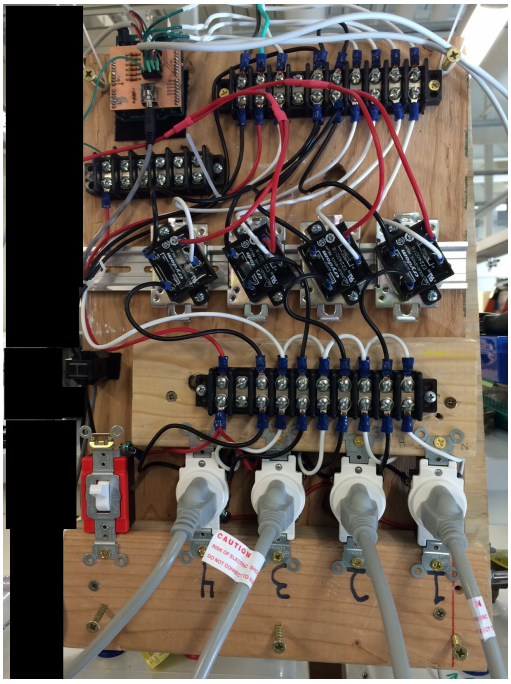


Fig. 5. Timing and Contol Board

In Figure 5 it can be seen that the light switch, outlets and numbering are upside down. Orginally the power rail was on the very top of our board. But due to space constraints the board ended up being flipped upside down to better fit the power cords.

The first challenge we encountered was finding a room within UAA that had a water source and a power source close enough for our system. We ended up finding space in EIB Room 103. The water spigot and power outlet for our project space in this room is located along the ceiling. We did not plan for our control board to be so far away from a power source.

The large distance from the water spigot meant we needed flexible hosing not only to connect to it, but to easily connect to the heaters. Flexible hosing was also chosen for ease of use and adaptability.

We chose vinyl tubing to plumb the water heaters to each other and to the water spigout. The tubing was difficult to clamp down enough to stop all water leaks. Once we began testing we realized hot water would travel up the cold water line made of vinyl tubing. When this happened, the tubing would begin to expand like a balloon. After it was expanded the tub would then lose its shape and pull on other parts of the system, see Figure 6.

Fig. 6. Vinyl Tubing Issues

An issue was realized when we milled the circuit board. Power for the sensors was run to Vin instead of to 5V. Instead of starting over we cut the pin to Vin and soldered a jumper between the Vin and 5V. In Figure 7, shown below, the yellow jump is easily seen on the left side of the board. Another issue was that the pins of the current clamp connector were miswired. Another jumper quickly resolved this problem.

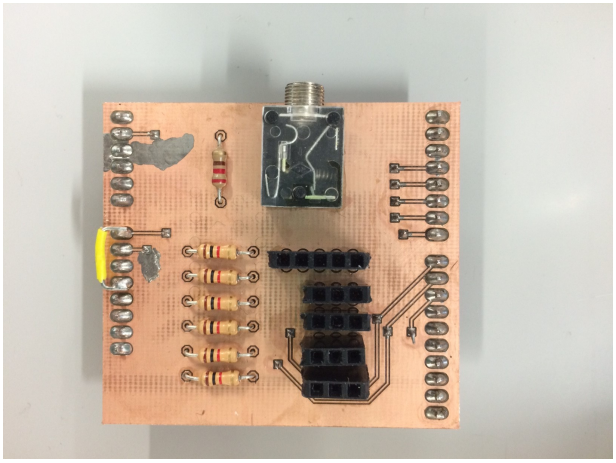


Fig. 7. Milled Circuit Board with Jumper Cable

IV. TESTING

The testing phase was a bit time consuming, but effective. The hot water heaters came with no specific heating information. To set a cycle time we needed to know the time it takes the heaters to heat up from ambient water temperature. The first testing phase involved finding the heating times of both sized heaters. Both the 2-gallon and the 4-gallon heaters have an ideal heat setting, along with a max. We monitored the length of heating time for each heater by setting the heater to ideal, turning the heater on, and timing how long the indicator light took to turn off.

After finding out how long the heaters took to heat, we needed to know how long the water could sit in the tank, unused, before it will cool off enough to need reheating. Originally, we started this test by connecting a multi-meter to the thermistor on the 2.5-gallon tank and manually writing down the resistance every 30 seconds while the tank's heat was set to ideal. After a couple minutes of doing this we realized the resistance was dropping very slowly and it would be a very long process. The Arduino was coded to collect the data used to calculate the temperature every 5 seconds and record it.

Once the data was collected for the 2.5-gallon warm up/cool off cycle the process was repeated on the same tank, but set on max temp. The procedure was repeated for the 4-gallon tank on ideal and then on max temp. The data was then loaded into Matlab and plotted in temperature vs. time graphs.

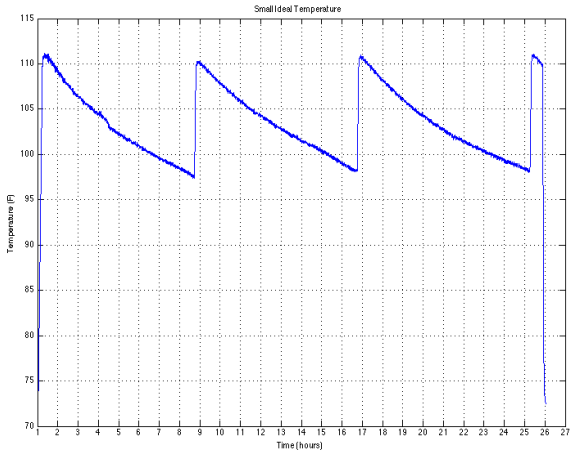
V. RESULTS

The results for the heat up cooldown time of both sized heaters at both temperature settings are in Table 1.

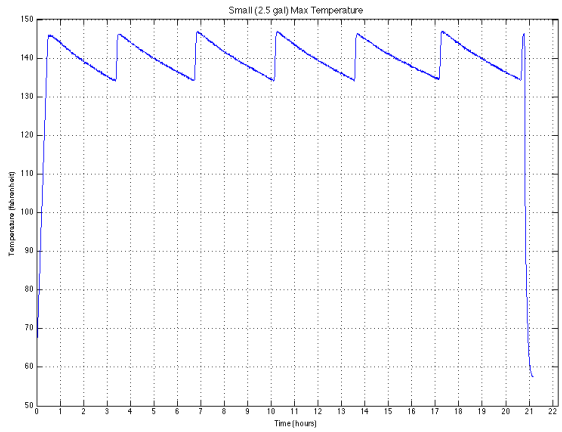
TABLE I.

Heater Warm Up Time		
Gallon Size	***Temp Setting***	***Time (Minutes)***
2.5	Ideal	
2.5	Max	
4	Ideal	
4	Max	

Using the results from Table 1 we were able to set a cycle time for the Arduino to cycle between heaters. The cycle time is currently set to 15 minutes because that is approximately how both heaters take to heat up on the ideal setting. For Max setting a cycle time of 30 minutes is recommended.

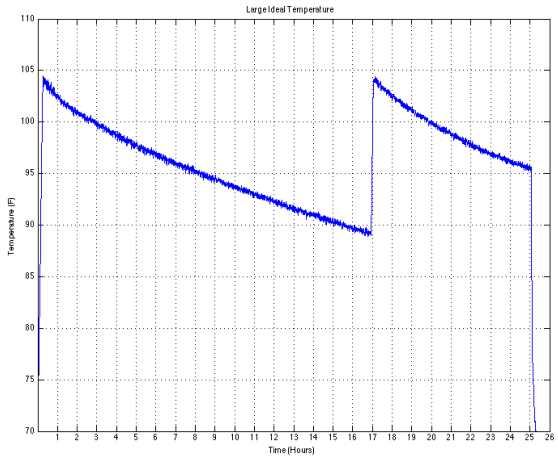


Graph 1. 2.5-Gallon Ideal Heat/Cool Cycle

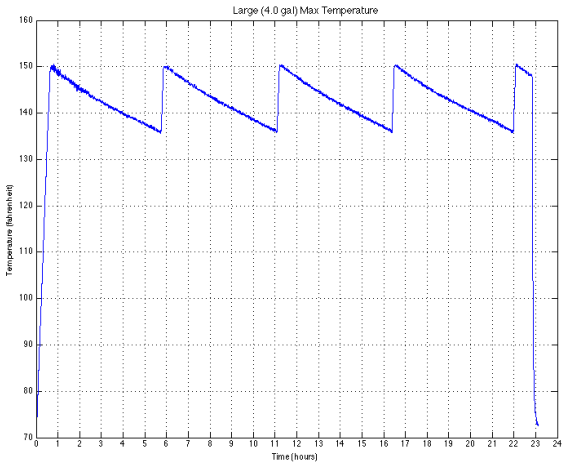


Graph 2. 2.5-Gallon Max Heat/Cool Cycle

As Graphs 1 and 2 above show, if the heater is not in use it will stay warm for a long period of time. The ideal setting takes almost 8 hours for the water to cool off enough for the heater to kick back on. The cool off period is a lot greater for the 4-gallon heater. On the ideal setting the cool off period is about 17 hours. This is shown below in Graphs 3 and 4.



Graph 3. 4-Gallon Ideal Heat/Cool Cycle



Graph 4. 4-Gallon Max Heat/Cool Cycle

VI. Cost Analysis

The total cost of our timing and control system was $2052.52. The cost of our project was separated into two parts: control costs and heating/plumbing costs. We decided to separate the costs because a lot of the plumbing costs were for testing purposes only and these costs will not be carried onto further phases of this water treatment system project.

The total control costs were $789.37. The two main costs were the flow sensors that were chosen and the SSR's. We picked the flow sensors based on recommendations from Professor Dotson due to longevity. Longevity of the sensors outweighs the upfront cost. The SSR's were also picked due to stability and longevity. The control cost break down is shown in Table 2.

TABLE II.

Control Costs			
Item	Price	Quantity	Total
Flow Rate Sesnor	$115.00	4	$460.00
SSR Relay	$57.92	4	$236.68
Current Clamp Senor	$9.95	1	$9.95
Arduino Uno	$24.95	1	$24.96
Heatsink	$13.98	4	$55.92
Light Switch	$2.97	1	$2.97
Mounting Rail	$3.90	1	$3.90
Total			$789.37

Table 3 shows our heating and plumbing costs, which totaled about 62% of the entire cost. The plumbing and heating costs were vital for testing purposes, but if this system were to be implemented in a home these costs would mostly be unnecessary. For a home build, the plumbing would probably be included in the price of the home. A user would not be buying vinyl tubing, or brass fittings for the hot water heaters as we did. The heaters chosen might also be different. The thermistors are no longer needed in the build. They were a fundamental part of building and testing the initial system, however now that the heating and cooling cycles are known they are no longer needed in the overall design.

TABLE III.

Heating and Plumbing Costs			
Item	Price	Quantity	Total
4 Gal Heater	$197.00	2	$394.00
2 Gal Heater	$187.00	2	$374.00
Thermistor	$0.67	2	$1.34
Miscellaneous Plumbing Parts	$493.81	1	$493.81
Total			$1263.15

The miscellaneous portion of Table 3 includes all the vinyl tubing used and all the fittings. It also includes the brass water valves, the brass overflow fitting and any tools we needed such as wire cutters. The miscellaneous section is the most expensive for the plumbing costs. As stated previously it was almost all testing costs and if this project were industrialized it would not be user cost.

VII. Recommendations

For further testing and in home the vinyl tubing should be replaced with high temperature hose or PVC pipe or some other form of plumbing. The vinyl tubing used in this project will explode.

We recommend building or purchasing waterproof housing for the control board. The control board should be encased in waterproof or water resistant housing for safety purposes.

To decrease the control cost while preserving the integrity of the system we recommend replacing the four flow sensors with current clamps. The four flow sensors cost a total of $460.00. Current clamps cost $9.95 a piece, with a total of 4 needed. If the four flow sensors were replaced with current clamps the total cost would decrease by roughly $420.00. The current clamps would be placed on each water pump. If the water pump is on and current is detected, it can be deduced that water is flowing into that heater.

APPENDIX A-PARTS LIST

TABLE IV.

Timing and Control Parts List				
Part Name	***Part SKU***	***Quantity***	***Price Per Unit***	***Website***
Water Flow Sensor	314150005	4	$115.00	Grainger.com
Relay	HS501DR-D2425-ND	4	57.92	Digikey.com
4 Gal Water Heater	ES 4	2	197.00	Grainger.com
2 Gal Water Heater	ES 2.5	2	187.00	Grainger.com
Current Sensor Clamp	SEN-11005	1	9.95	Sparkfun.com
10k Thermistor	Ntcl 428	2	0.67	Digikey.com
Arduino Uno	DEV-11021	1	24.95	Sparkfun.com
Kill A Watt	P4460	1	31.87	Homedepot.com
Heatsinks	Cc1794-nd	4	13.98	Digikey.com
Light switch	CSB115STW-SP	1	2.97	Homedepot.com
Mounting Rail	18Z759	1	3.90	Grainger.com
Miscellaneous (this includes all the tubing, fittings, values and wiring for the heaters)	---------	--------	493.81	----------

REFERENCES

[1] http://labor.alaska.gov/research/pop/popest.htm

[2] http://watersewerchallenge.alaska.gov/ruralCommunities.html

APPENDIX B-ARDUINO CODE

```
#include <PinChangeInt.h>
#define WHCOUNT 4
#define MIN_FLOW_RATE 100 // In Hertz;
#define CURRENTTHRESH 660 // About 3.5 volts
#define MINSWITCHTIME 5 * 1000 // 5 Seconds
#define MAXRUNTIME  (15*60*1000)

#define bitRead(value, bit) (((value) >> (bit)) & 0x01)
#define bitSet(value, bit) ((value) |= (1UL << (bit)))
#define bitClear(value, bit) ((value) &= ~(1UL << (bit)))



//Priority goes from index 0 to 3.
int flowMeterPins[WHCOUNT] = { 7, 8, 9, 10 };
int currentXfmrPin = A4;
int relayControlPins[WHCOUNT] = { 3, 4, 5, 6 };

//Priority goes from Least significant bit to most.
byte waterRunIndication = 0;
byte waterHeatRun = 0;

int intCount; // Varible used in interrupt to calculate Flow rate.

unsigned long lastSwitch;
unsigned long lastWaterRun;

bool WaterRun;


void setup() {
  // Set the input and output of each device. \
  Serial.begin(9600);
  for(int cnt = 0; cnt < WHCOUNT; cnt++){
      pinMode( relayControlPins[cnt], OUTPUT);
      pinMode( flowMeterPins[cnt], INPUT);
  }

  WaterRun = false;
  lastSwitch = 0;
  lastWaterRun = 0;
  switchWaterHeater(-1);
  Serial.println("Setup complete, beginning");

}

void interruptCount(){
    intCount++;
}

void ReadFlowMeters(){
    waterRunIndication = 0;
  for(int cnt = 0; cnt < WHCOUNT; cnt++)
  {
```

```
      intCount = 0;
      PCintPort::attachInterrupt(flowMeterPins[cnt], interruptCount, RISING);
      delay(500);
      PCintPort::detachInterrupt(flowMeterPins[cnt]);

        /*Serial.print("WH ");
        Serial.print(cnt);
        Serial.print("is flowing ");
      double flowRate = (intCount * 2) * (.007229);
        Serial.println(flowRate);*/
      if((intCount * 2) > MIN_FLOW_RATE)
      {
              Serial.print("WH ");
              Serial.print(cnt);
              Serial.print("is flowing ");
              Serial.println(intCount*2);
        bitSet(waterRunIndication, cnt);

      }

  }
}
bool currentFlow(){
    int maxc = 0;
    int temp =0;
    for(int i = 0; i < 32; i++){
        temp = analogRead(currentXfmrPin);
        if(temp > maxc){
            maxc = temp;
}
        delay(1);
    }
    if(maxc > CURRENTTHRESH)
        return true;
         Serial.println("Water heater is not on");
    return false;
}

void loop() {

    if((unsigned long)(millis() - lastSwitch) >= MINSWITCHTIME)
    {
        ReadFlowMeters();
        if(waterRunIndication == 0){
            if(WaterRun == true){
                lastWaterRun = millis();
                WaterRun = false;
            }

            if((unsigned long)(millis() - lastWaterRun) >= MAXRUNTIME){
                switchWaterHeater(-1);
            }
            else if(currentFlow() == false){
                switchWaterHeater(-1);
            }
        }
        else{
```

```
        if(WaterRun == false)
          WaterRun = true;

        if(waterRunIndication != waterHeatRun){
          for(int cnt = 0; cnt < WHCOUNT; cnt++){
            if(bitRead(waterRunIndication, cnt) == 1){
              switchWaterHeater(cnt);
              break;
            }
          }
        }
      }
        lastSwitch = millis();
  }
  delay(500);
}

//return the index of the water heater that is currently on.
int currentHeaterOn(){
  for(int cnt = 0; cnt < WHCOUNT; cnt++){
    if(bitRead(waterHeatRun, cnt) == 1){
      return cnt;
    }
  }
  return 0;
}

void switchWaterHeater(int toHeater){
    int stoHeater = toHeater;//Temp variable
  if(stoHeater == -1){
    int curHeater = currentHeaterOn();
    stoHeater = curHeater + 1;
    if(stoHeater >= WHCOUNT)
      stoHeater = 0;
  }
    Serial.print("Switching to relay: ");
    Serial.println(stoHeater);
  for(int cnt = 0; cnt < WHCOUNT; cnt++){
    digitalWrite(relayControlPins[cnt], LOW);
  }
  waterHeatRun = 0;
  digitalWrite(relayControlPins[stoHeater], HIGH);
  bitSet(waterHeatRun, stoHeater);

}
```