Project Report

*PROJ-201802001*

30 May 2018

Compiled By: Cor Cronjé

Contents

[1. Problem Statement: 3](#_Toc515442742)

[2. System Requirement: 3](#_Toc515442743)

[3. System Design: 3](#_Toc515442744)

[4. Development: 5](#_Toc515442745)

[5. Deployment: 8](#_Toc515442746)

[6. Conclusion: 8](#_Toc515442747)

# Problem Statement:

Reduce electrical power consumption through the development of a geyser utility management system based upon the Raspberry Pi platform.

# System Requirement:

A Raspberry Pi single board computer (SBC) is to be employed for monitoring two DS18B20 temperature probes, one installed in an electrically heated residential geyser, the other in a solar geyser. The system’s primary function is to reduce electrical power consumption through, maintaining a regulated water temperature in the electrical geyser as per a predefined temperature schedule, and by means of toggling electrical power to the geyser’s element through a relay.

Secondly, the system is to monitor the temperature difference between the solar geyser and the electrical geyser, and circulate warmer solar heated water into the electrical geyser trough activating an electrical pump, thus potentially reducing electrical power consumption even further.

Lastly, the system is to log the temperature values and also the cumulative element and pump active times to a MySQL database. A PHP web interface must display the data in a visual manner, and further allow for all configurable parameters and the temperature schedule to be set by a user.

# System Design:

The system will employ the Raspberry Pi as the main logic controller, two DS18B20 probes for acquiring temperature values, and a relay breakout board with three relays for activating the element and pump. The following diagram Figure 3.1 provides a high level overview of the solution.

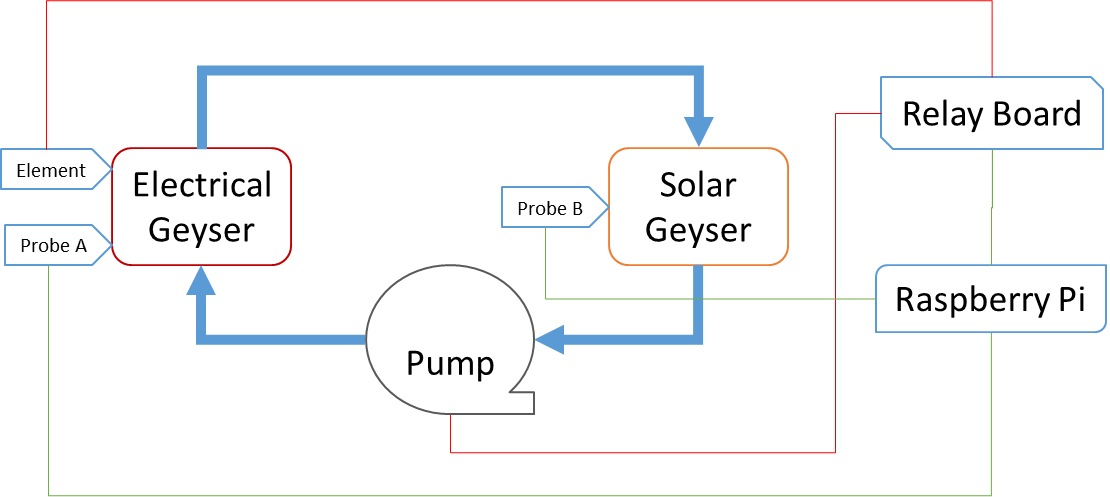


Figure 3.1. An overview of the proposed solution.

The Python scripting language must be employed as the basis for controlling the systems logic, logging both temperature and logical data to, and reading configuration parameters from a MySQL database. The PHP web interface must provide for visual representation of the collected data and user interaction with the system and configuration parameters. Figure 3.2 is a visual representation of the programming logic.

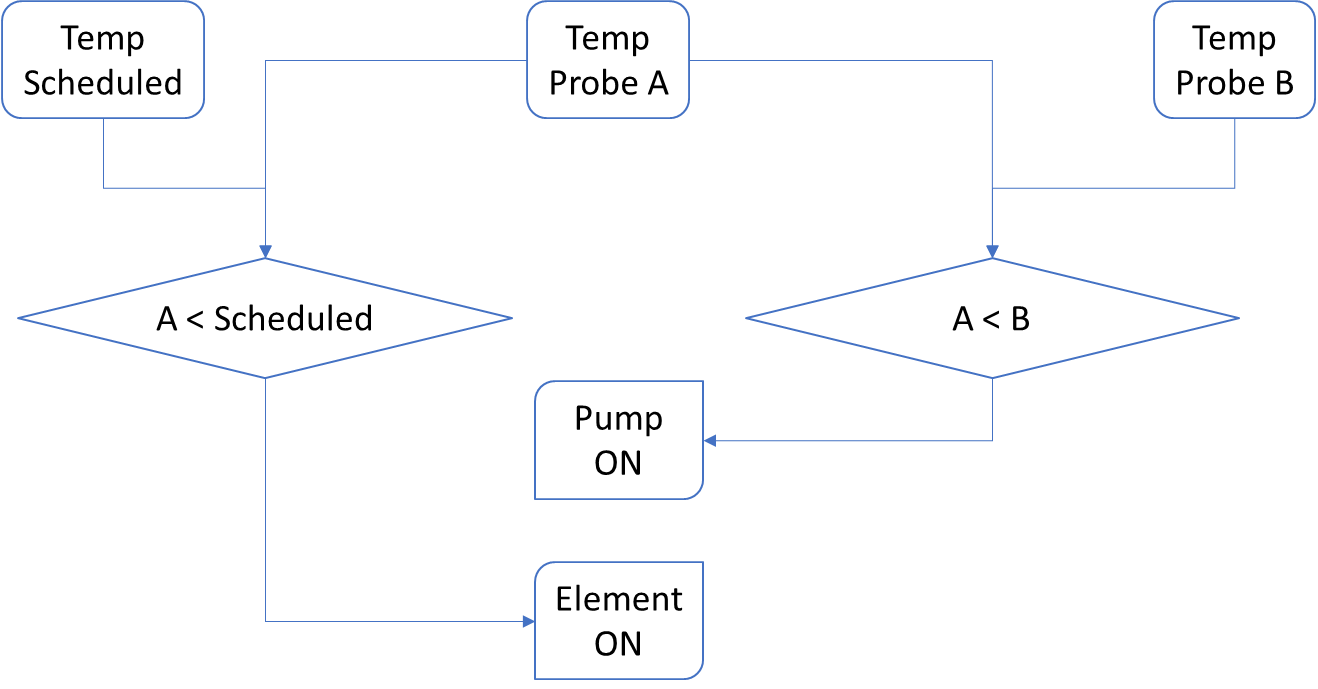


Figure 3.2. A visual representation of the programming logic.

# Development:

Several Python, PHP and MySQL routines where developed to meet the requirement, with the following Table 4.1 describing the purpose and function of the various scripts and routines employed.

Table 4.1. A tabular description of the various scripts and routines.

|  |  |
| --- | --- |
| Python | |
| **Name** | **Description / Purpose** |
| Main.py | This is the main logic of the solution, reading configuration parameters from the MySQL database, acquiring environmental data, and updating the database with the collected data. |
| getGeyserTemp.py | This script is a subroutine included by Main.py for acquiring the digital temperature value from the DS18B20 temperature probe to be installed in the electrical geyser. |
| getReservoirTemp.py | This script is a subroutine included by Main.py for acquiring the digital temperature value from the DS18B20 temperature probe to be installed in the reservoir of the solar geyser. |
| logTemperatureValues.py | This script runs independently from the main logic, and log the temperature values from both probes to the MySQL database. |
| logControlValues.py | This script runs independently from the main logic, and log the relay output values from both relays to the MySQL database. |
| logRuntimeValues.py | This script runs independently from the main logic, and trigger a MySQL routine to log the cumulative values for both relays. |

|  |  |
| --- | --- |
| PHP | |
| **Name** | **Description / Purpose** |
| Database.php | This script is included by the other PHP scripts and provides the global values for accessing the MySQL database. |
| index.php | Provides the user with a visual overview of the temperature values, system status, and manual controls for the system. |
| Schedule.php | This web interface provides the user with a visual matrix for setting the geyser temperature schedule. |
| History.php | This scripts renders historical temperature and runtime data to the user in two charts. |
| Setup.php | Provides an interface where the user can adjust and setup the configurable parameters of the system. |
| Api/element.php | Provides the API interface for the electrical geyser element. |
| Api/pump.php | Provides the API interface for the electrical circulation pump. |
| API/holiday.php | Provide the API interface for the holiday mode setting. |
| API/runtime.php | Provides an API for retrieving the historical cumulative runtime data of the pump and element. |
| API/schedule.php | Provides an API for reading and setting the temperature schedule matrix. |
| API/temperature.php | Provides an API to retrieve the historical data for both the temperature probes. |

|  |  |
| --- | --- |
| MySQL | |
| **Routine** | **Description / Purpose** |
| getRuntimeForDate | Retrieve the runtime values for a custom specified date as called by the respective API. |
| getRuntimeForToday | Retrieve the runtime values for specified date as called by the respective API. |
| getRuntimeForYesterday | Retrieve the runtime values for specified date as called by the respective API. |
| getRuntimeOverview | Retrieve the runtime values for the last 12 hours as called by the respective API. |
| getTemperatureForDate | Retrieve the temperature values for a custom specified date as called by the respective API. |
| getTemperatureForToday | Retrieve the temperature values for specified date as called by the respective API. |
| getTemperatureForYesterday | Retrieve the temperature values for specified date as called by the respective API. |
| getTemperatureOverview | Retrieve the temperature values for the last 12 hours. |
| incrementElementRuntime | Increment the cumulative pump runtime parameter as triggered by the logRuntimeValues routine. |
| incrementPumpRuntime | Increment the cumulative pump runtime parameter as triggered by the logRuntimeValues routine. |
| logRuntimeValues | Increment the cumulative element and pump runtime parameter as triggered by the logControlValues.py script. |
| setElementOff | Toggle the element status parameter to off. |
| setElementOn | Toggle the element status parameter to on. |
| setPumpOff | Toggle the pump status parameter to off. |
| setPumpOn | Toggle the pump status parameter to on. |

# Deployment:

The solution is deployed to the pi user’s home directory and initiated on system start-up through the pi user’s “.profile” file.

# Conclusion:

The aforementioned sections provide an overview of the requirement and the solution, and also describe the various functions and purpose of each component. The system was not tested in the real world, and only simulated tests was performed to evaluate the system’s operation. It is recommended the installation and maintenance of the system is done by a reputable and certified electrician, and that the electrical element’s maximum temperature is set below boiling point, and just above the maximum temperature configuration parameter of the solution. It must be noted that no security is included in the web interface nor is the API secured, hence allowing for easy integration with other systems, but greater security risk.