The goal of this project was to create a prototype thermostat. The thermostat must be able to have an adjustable setpoint, read the temperature, send a signal to activate a heater when appropriate, and send data to an external server. For this project, the UART peripheral was used in place of an external server, and a LED was used in place of a heater; for prototyping and testing purposes. This project made use of a task scheduler and interrupt services to achieve the desired functionality. Per the specifications, the setpoint should be checked and updated every 200 ms, the temperature should be read and compared to the setpoint every 500 ms, and the unit should provide the status of the system every 1 second (temperature, setpoint, heater status, seconds active). The tasks were divided into three categories based on their timing requirements. To meet these specifications, the greatest common denominator was determined to be 100 ms, which was set as the period for the timer interrupt. Each 100 ms, the timer would update the flag and the system will check each of the tasks and lower the timer flag. When the time aligned with task interval, that task would be executed. In this manner the setpoint would be updated at 200, 400, 600, 800, and 1000 ms. The temperature would be read and the heater activated or deactivated at 500 and 1000 ms. The output would be provided ever 1000 ms, and the timer would be reset to 0. This loop will continue indefinitely. To update the setpoint, two buttons on the device were used. These interrupts would increase or decrease a ‘setpoint change’ variable. This variable would be applied to the setpoint on the 200 ms intervals, and the setpoint change variable would be reset. This allows for the changes to be input and accumulated at any time, but only affect the setpoint when that task is called. During the temperature read task, the current temperature would be compared to the setpoint. If it is found to be less than the setpoint, the heater (LED) would be activated. When the temperature is at or above the setpoint, the heater would deactivated.

The next step in prototyping would be to make a fully functional device beyond the proof of concept that has been developed. A review of hardware architectures from TI, Microchip, and Freescale has been performed. The prototype has been developed on a TI board that has WiFi capabilities, making it a strong contender. This architecture fully supports the code and desired functionality of the device. Microchip is an alternative and has several similarities to the TI architecture in use. Both can support the peripherals used in the prototype and have access to various alternatives. However, TI appears to be better positioned with low energy units that also have a good deal of security. As we are making use of WiFi for sending and receiving data to the thermostat, security for the unit should be a high consideration. Freescale is another choice that could be put into use. There appears to be a lot of hardware compatibility between TI and Freescale, although there are some differences in the pin configuration. When it comes to security, Freescale also has a solid history. Based upon the review of the various architectures, it may be the most simple to stick with TI and still achieve the desired results. The performance required for the system is not very demanding, and there would be no additional work required to get the coding and peripherals to work well, as may be required with the other systems. TI offers systems that are built for this type of application, with the temperature sensors and embedded Wi-Fi network processor. For the device to connect to the cloud, the embedded Wi-Fi network processor communicates to the wireless router in the home using IEEE 802.11b/g/n protocol (local area network standard). All architectures have chips/boards that use this protocol, as it is the standard. This allows for the device to access the internet, where it can communicate with the cloud server(s). The cloud network is where the user of the device would also interact with in order to monitor and set the temperature without using the physical buttons. All three architectures have solutions that support the needed flash and RAM for the program.