# CS 350 Thermostat Project Report

The TI SimpleLink Wi-Fi CC3220S Launchpad has been configured and coded to act as a thermostat. It has been configured with a task scheduler that will utilize a timer to track time, will read input from button presses, read the temperature of the room every second, and will change the state of the built in state machine that controls when the heat is turned on as well as the indicator that notifies the person that the heat is on. A display will also be continuously displayed and updated every second that passes the timer. Secondary timers will be utilized to continually monitor button presses that will raise a flag to either increase the set temperature or decrease the set temperature depending on what button is pushed. Flags will be reset when the setTemp has been re-adjusted based on the input. The state machine will re-evaluate its status every half second. This state machine will test when the setTemp is higher than the temperature being read and if it is, this will trigger the heat to be turned on. In the case that the temperature in the room raises pass the setTemp, the heater will turn off.

The thermostat we created takes advantage of the peripherals included with this hardware such as the I2C peripheral, the GPIO peripheral, and the UART peripheral. The UART peripheral allows for the constant communication between devices. In this case, our launchpad is equipped with wireless technology that can communicate the status such as the current set temperature, current thermometer temperature, the heat status, and the time. UART is also very important these days in our current research of the IoT world where most of our devices will be able to communicate with each other. These being sent wirelessly will allow for other items to log the information or be sent to other devices that will need to utilize the information such as the actual heating unit. The GPIO allows for input into the board that allows for different triggers. In this case, we utilize the buttons to allow for a decrease or increase in temperature. The code itself will search for any button presses and update based on how often it’s evaluated. Performing the increases or decreases in set temperature this way allows for more control. If the buttons themselves increased and decreased temperature, it would be harder to do since when the code is in the loop, one press may equal multiple inputs as we have learned in previous modules. The last of the peripherals we used is the I2C peripheral which allows for the multitasking we have been able to take advantage of with the task scheduler. It also allows for all the hardware we have equipped on the launchpad to be able to interact with each other at quick speeds.

Based on our needs of peripherals and communication requirements the following architectures have been assessed: TI, Microchip, and Freescale. Microchip has a focus on memory and RAM for larger complex products that require a large amount of these resources. Our code is fairly simple so I wouldn’t say that the Microchip is a priority for this project. Freescale has more of a focus on communication powers which is important for us however we also have a priority for including the peripherals we used in our prototype. For that reason, the TI will be most fit for our needs in this product moving forward. With regards to memory, the device used for the prototype comes with 1MB flash memory and 256kb of RAM. During our testing process, our code was shown to use about 25% of those resources. This should be noted when looking for devices that can be used.

References

DN Staff. (2017, May 22). *Freescale, microchip, TI roll out microcontrollers*. DesignNews. Retrieved June 21, 2022, from https://www.designnews.com/freescale-microchip-ti-roll-out-microcontrollers-0

Pena, E., & Legaspi, M. G. (2020, December). *UART: A hardware communication protocol understanding universal asynchronous receiver/transmitter*. AnalogDialogue. Retrieved June 21, 2022, from https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

SFUPTOWNMAKER. (n.d.). *I2C*. Sparkfun. Retrieved June 21, 2022, from https://learn.sparkfun.com/tutorials/i2c/all

Westfw. (2017, November 12). *How to choose a microcontroller*. Instructables. Retrieved June 21, 2022, from https://www.instructables.com/How-to-choose-a-MicroController/

Wiki by STM32 MPU. (2021, August 12). *GPIO internal peripheral*. GPIO internal peripheral - stm32mpu. Retrieved June 21, 2022, from https://wiki.st.com/stm32mpu/wiki/GPIO\_internal\_peripheral