In the project we created a thermostat embedded application. This application uses GPIO, UART, I2C and 32-bit Timer features or functionality for this project. Other specifications used in the comparison of other embedded boards is the need for wireless for communicating with the cloud, based on the ARM architecture, SRAM on-chip flash memory, and serial interface via USB. The first board compared against the TI CC3220S is the Microchip EV75S95A Development Board with features the SAMD21G18 microcontroller (Microchip), the second board is the KW40Z (NXP Semiconductors, 2021) from NXP. Below is a table that shows the comparison between all three boards.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **C3220S** | **EV75S95A** | **FRDM-KW40Z** |
| **Price** | $39.99 | $39.59 | $224.25 |
| **Memory** | 256KB RAM/ 1MB | 256KB/32KB | 152KB RAM/640KB |
| **Wireless** | Single-band 2.4Ghz | Single-band 2.4Ghz | Single-band 2.4Ghz |
| **USB Interface** | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| **UART** | 2 | 6 | 2 |
| **GPIO** | 27 | 38 | 22 |
| **I2C** | 1 | 6 | 2 |
| **Timer** | 4 | 2 | 4 |
| **Processor** | ARM Cortex-M4 | Arm® Cortex®-M0+ | Arm® Cortex®-M4 |
| **Temperature Sensor** | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |

As can be seen from the table above the 3 microcontrollers are similar and would be suffice for the thermostat application. All boards have at least one UART for communicating with the server, one 12C for reading the temperature sensor, one timer that features a real-time clock for running the scheduler, one LED for showing the status on the unit. However, one major difference amongst the three is the KW40Z board does not include any external buttons that I could find but it was the only comparable option for a Freescale architecture board (NXP Semiconductors, 2021). Another major difference between the three boards is the KW40Z. and the **EV75S95A** support USART which is Universal Synchronous/Asynchronous Receiver/Transmitter (NXP Semiconductors, 2021) (Microchip). It seems this feature is an upgrade to UART. The flash memory and SRAM that is supported by each board varies quite a bit, all devices have standard flash memory and SRAM but the main difference is the amount each board has. The TI CC3220 has 1MB of SRAM and 256KB of flash memory which is plenty for our current application (Texas Instruments, 2017), the Microchip has 32KB of SRAM and 256KB of flash memory (Microchip), this is sufficient for memory but the SRAM may cause the unit to run a little slow (Microchip), and finally the KW40Z has 20KB SRAM and 160KB of flash memory (NXP Semiconductors, 2021), this board may be pushing it on both SRAM and flash memory for our application. All boards use an ARM Cortex processor and is equipped with a temperature sensor via I2C.

They all support Wi-Fi 802.11n so that meets the requirements of being internet accessible. They all feature WPA/WPA2 web security so there are no issues there, for connecting the thermostat to the internet. Even though I could not find any documentation for connecting the wireless on the other two devices I would imagine it would be the same for all three boards since there will be a user interface setup on the device and setup of the wireless could be done through the interface allowing the device access to the internet.

The technical functionality of the application is as follows:

* On unit startup it will send to the server the welcome message
* The inits for the timer, GPIO, I2C, and UART is initialized.
* The timer starts counting and once the period expires the application calls the timer callback.
* The timer callback then loops through each tasks checking the elapsed time against the period of the task being evaluated.
* If the elapsed time is greater than or equal to the period it then changes the triggered property for the task to true, sets runTasks to true and elapsed time to zero.
* If elapsed time is less than the task period it will add the value of TICK\_TIMER which is 100ms to the elapsed time and then continues the loop.
* In parallel to that on the main thread if runTasks equals true it will then start a loop that checks the triggered value for true, if it is true then it runs the function for the specific task. It then sets the triggered value back to false before continuing the loop.
* Once all the tasks are looped through and all applicable tasks are executed it then sets runTasks back to false
* It then waits again for the timer interrupt to take place.
* While all of this is happening, the application is also waiting for a button interrupt to take place. When a button is pressed it calls the button interrupt function that changes the value of the button state based on which button was pressed.
* If button 0 is pressed it changes the button state to INCREASE and if button 0 is pressed is changes the button state to DECREASE.
* Now for the individual tasks that are executed when triggered:
  + taskCheckButton – This task is called every 200ms and when ran it checks the button state for an increase or decrease. If the state is one of those two it will either decrease or increase the value of setpoint. After it changes the value it then sets the button state to NEUTRAL.
  + taskCheckTemp – This task executes every 500ms and calls a function getUnitState which reads the temperature from the I2C temperature sensor and then compares it to the setPoint and the maxTemp and changes the unit state based on the result.
    - If the temperature is less than the setPoint the function sets the unit state to HEATING.
    - If the temperature is greater than the setPoint and less than maxTemp it sets the unit state to OFF
    - Here is functionality I added to the application. If the temperature is equal to or greater than the maxTemp then is sets the unit state to COOLING.

After the getUnitState function is complete the task then continues to execute, based on the current value of the unit state it will perform various actions. If the unit state is HEATING it turns the LED on, COOLING it will blink the LED and OFF the LED will be turned off.

* taskUpdateSvr – This task executes every second and the only thing it does is send a message to the server via a serial connection on a COM port.
* This is the entire functionality of the application; it runs continuously until the timer is stopped. The timer runs for 100ms and the the timer callback is ran checking for any tasks that should be triggered and if so they will be triggered executing the functions set in the tasks structure.

In conclusion, the application is not overly complex and technically all three boards meet the specifications for the thermostat application, they all have a RTC (real-time clock) timer, temperature sensor, wi-fi integration, LED, and at least one UART, there is one board that I would recommend over all the others. The board I would recommend would be the TI CC3220 board, this decision does not come because it is the board used to develop the application but the stats of the board and the pricing. The KW40Z is expensive for the specifications that it offers and while the Microchip is priced closer to the CC3220 the lack of SRAM or flash memory leaves no choice.

# Bibliography

Microchip. (n.d.). *ATSAMD21G18.* Retrieved from microchip.com: https://www.microchip.com/en-us/product/ATSAMD21G18#purchase-from-store

NXP Semiconductors. (2021, May). *K32W061/K32W041.* Retrieved from nxp.com: https://www.nxp.com/docs/en/data-sheet/K32W061.pdf

Texas Instruments. (2017, Oct). *Designing Thermostats With CC3220 SimpleLink™ Single-Chip Wi-Fi® MCU System-on-Chip.* Retrieved from ti.com: https://www.ti.com/lit/an/swaa168/swaa168.pdf?ts=1676873902705&ref\_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTIDM-1020