Becky Scott

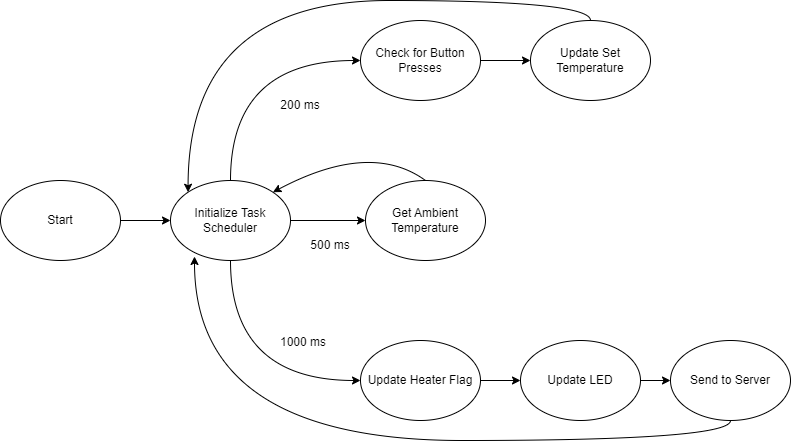
CS 350

Project 1 - Report

6/18/2023

**Task Scheduler Technical Documentation**

| **Algorithm** | The task scheduler used in this prototype uses a timer with different periods to call functions at the appropriate interval as described in Vahid et al. The base timer rate is 100ms, and will trigger a flag to check for the specified periods. Within the task scheduler, time rates have been set for each task. When the task timer is hit, the task timer resets to 0 and the appropriate function is called. |
| --- | --- |
| **Input** | The input is an initialized timer which has been set to 0. |
| **Output** | The outputs will be function calls at the appropriate defined times. |
| **Expected Result** | The task scheduler should call a function to check the button pressed flag every 200ms, a function to obtain the temperature from the sensor every 500ms, and a function to update the heater flag and LED, and send all data to the server every 1000 ms. |

****

**Explain how the thermostat supports the peripherals used in the project.**

The prototype built on the Texas Instruments SimpleLink Wi-Fi CC3200SF supports the peripherals used in the thermostat app using 2 main interfaces: GPIO and I2C. The prototype additionally uses the UART interface to send data off of the embedded system onto a separate cloud device. The TI CC3200 series includes General-Purpose Input and Output (GPIO) interfaces, which are connected to the onboard LEDs and buttons (Butler, 2022). In the prototype, I used GPIO to interface to the side buttons included on the embedded system, which are configured to control the setpoint temperature for the thermostat. Additionally, I used GPIO to interface with the onboard LED, which will display the status of the thermostat heating state.

The TI CC3200 also includes support for the I2C interface, which is an Inter-Integrated Circuit interface used to communicate efficiently between systems or between the system and peripherals. In the case of the prototype, I used the I2C interface to communicate with the onboard TMP006 sensor, which measures the ambient temperature on the device.

Other embedded systems, including ones manufactured by Microchip and Freescale, also include support for GPIO, I2C, and UART to support the peripherals on those embedded systems. Similar to the TI board, both Microchip and Freescale offer a board that includes a temperature sensor, 2 buttons, and LEDs, such as the PIC32MZ (Microchip) and the iMX RT1060 Crossover MCU (Freescale) (Microchip, n.d.; NXP, n.d.).

**Explain how the thermostat connects to the cloud via Wi-Fi. Discuss all three architectures in your work.**

The prototype, which is built on the TI CC3200SF board, connects to the cloud via Wi-Fi, and sends thermostat information via a UART interface. UART stands for Universal Asynchronous Receiver-Transmitter, and is a commonly used interface for asynchronous data transfers (Pena & Legaspi, 2020). In the prototype, 4 pieces of information are sent over this interface: The current temperature taken from the sensor, the setpoint configured by the user through the onboard buttons, the heater status calculated by the system, and the time value maintained by the system. Both Microchip and Freescale have systems that could support UART transfers to the cloud via Wi-Fi, but the developer must be sure the embedded system chosen has Wi-Fi capabilities such as the TI CC3200SF.

**Discuss the architecture’s Flash and RAM that supports the code. Include all three architectures in your discussion.**

The prototype, built on the TI CC3200SF board, uses 256KB of SRAM to load the code instructions, and 64KB of persistent ROM for device initialization, bootloader, and driver libraries (Texas Instruments, n.d.). For a project like the thermostat program, this is plenty of space to run the instruction set and any additional memory requirements that go along with such a program, especially since data is being sent to the cloud and can be stored there. If there is a requirement to retain some of the thermostat data on the device itself, there may be concerns with the storage capabilities of this embedded system. The comparable systems from Microchip and Freescale contain 512 KB and 1 MB of SRAM, respectively (Microchip, n.d.; NXP, n.d.). If the application requires a larger space for instruction in memory, the Microchip or Freescale systems may be a better choice.

References

Butler, S. (2022, April 8). What is GPIO, and what can you use it for?. How. https://www.howtogeek.com/787928/what-is-gpio/

Microchip, (n.d.). PIC32MZ EF Family of Microcontrollers Diagram. <https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/pic32-32-bit-mcus/pic32mz-ef>

NXP, (n.d.). i.MX RT1060 Crossover MCU Block Diagram. <https://www.nxp.com/assets/images/en/block-diagrams/iMX-RT1060-Block-Diagram.jpg>

Pena, E., & Legaspi, M. G. (2020, December). UART: A hardware communication protocol understanding universal asynchronous receiver/transmitter. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices. https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

Texas Instruments, (n.d.). TI CC3200 MCU Block Diagram. <https://www.ti.com/ds_dgm/images/fbd_swas032F.gif>

Vahid, F., Givargis, T., & Miller, B. (2013). Chapter 4. In CS 350: Emerging Systems Architectures and Technologies. essay, Zyante Inc.