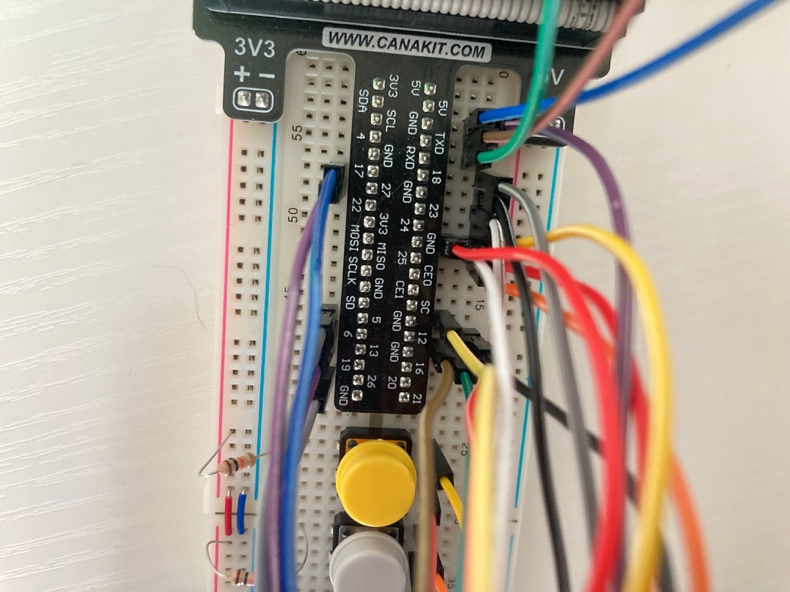
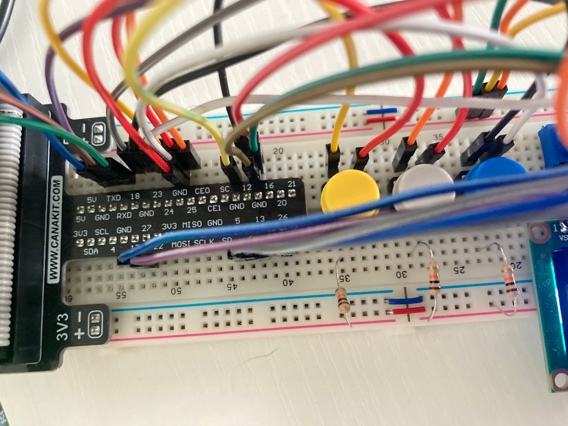
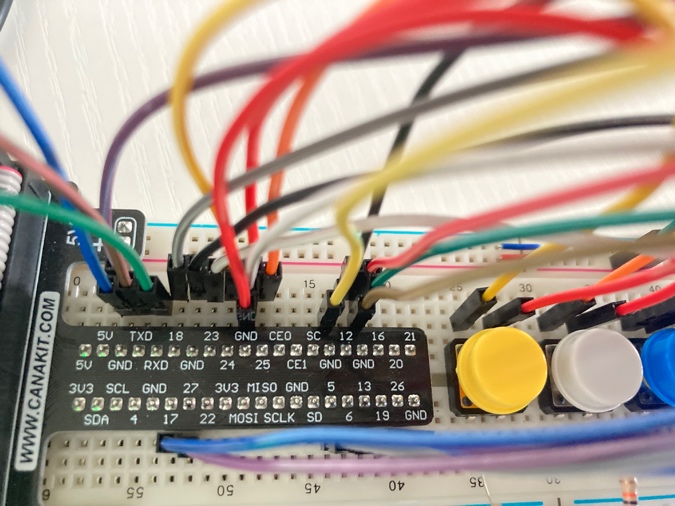
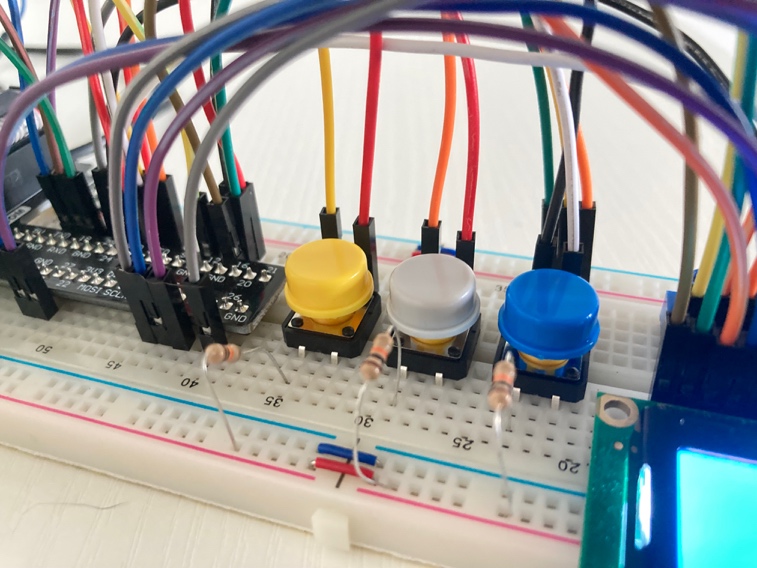
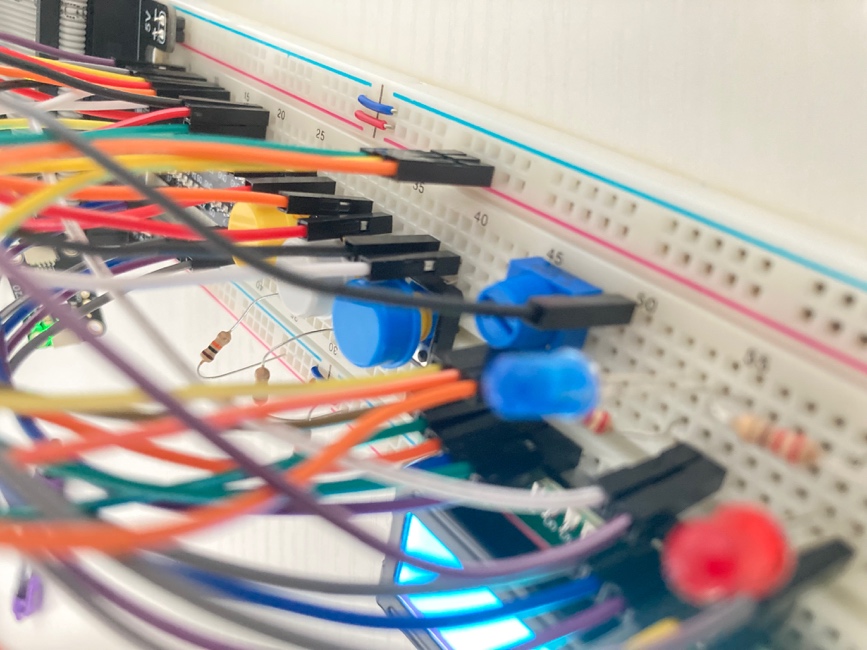
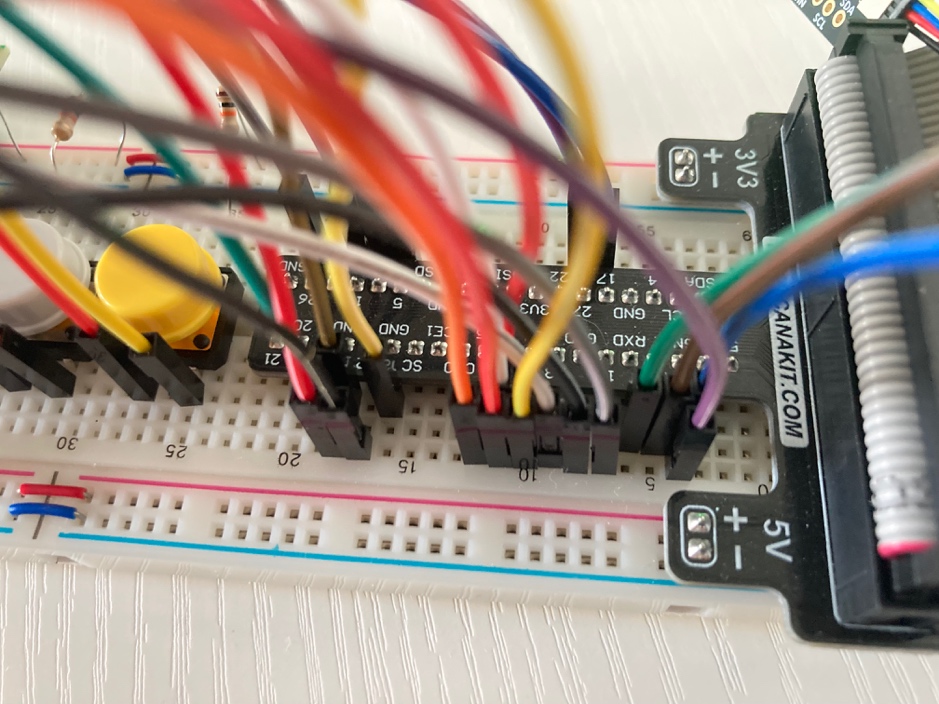
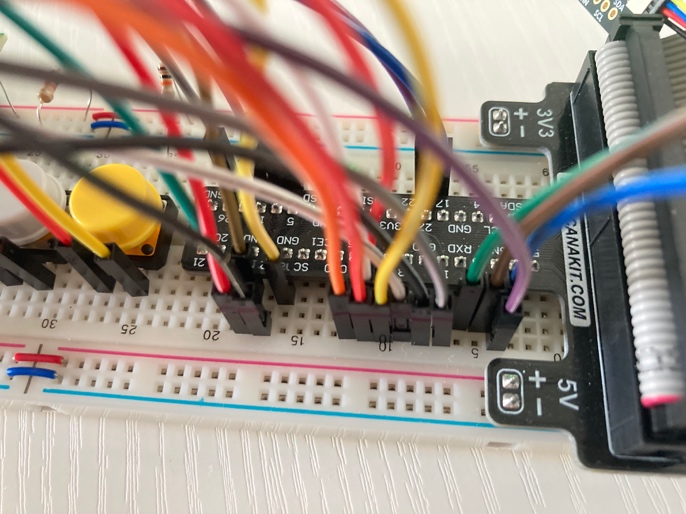
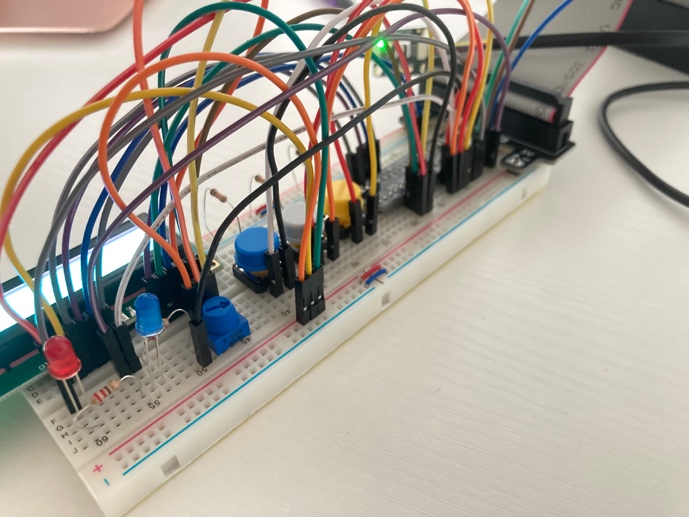
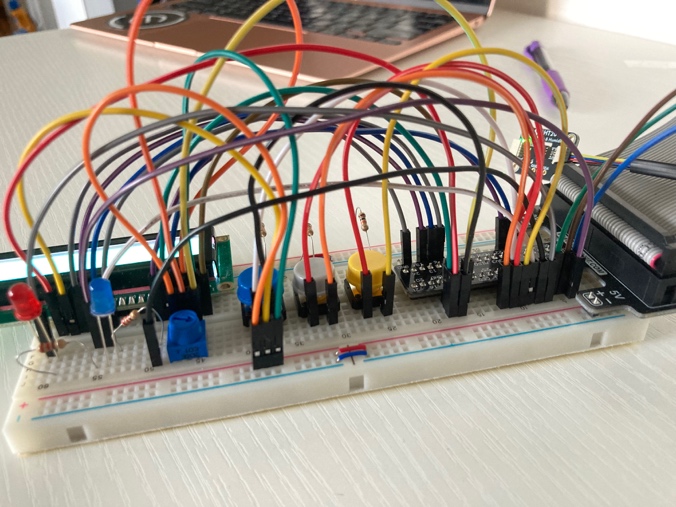
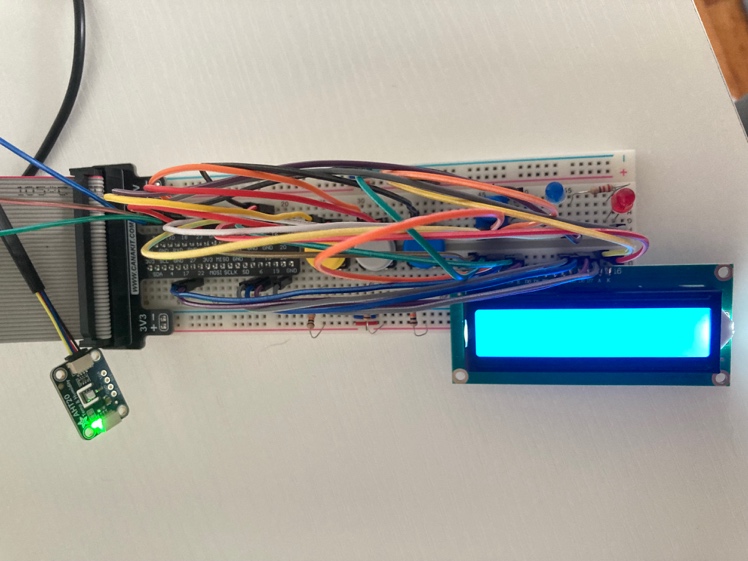
Shayna Mitchell

June 2025

Thermostat Report

SNHU CS350 Embedded Systems



The request for SysTec to develop and build a smart thermostat now has a working prototype. The addition of embedded sytems to the Systec software development team has allowed for new products and services to be fulfilled in the global smart thermostat market. The prototype will be able to send data to SysTec sever software. This will be completed via wifi once entirely complete. The project consists of the following components: Raspberry Pi, an LCD screen, three physical buttons (circuit with breadboard, LED lights, and an AHT2 Sensor. It utilizes the I2C peripheral, UART peripheral, and GPIOZero Library. The peripherals used in this prototype are theAHT2 Sensor, the I2C interface, the LCD screen, Buttons with GPIO capabilities, and the UART serial port. The sensor measures the temperature and updates the UART every 30 seconds with current data. The lcd screen is live as the program runs constantly updating as it receives new data. The GPIO capabilities allow the buttons to control specific pins in the breadboard and ultimately exemplify all user prompted results. Wifi will be the ultimate integration of the final system, but for now the UART serial communication will suffice. It sends data as received to the console in real time. All of these current capabilities are needed in a good and reliable embedded system.

Because the temperature sensor is utilizing the I2C peripheral, the prototype is able to read the current room temperature and in turn, display data on the lcd screen and light up led lights according to the acquired data. In addition to this, the prototype has the capability to change states according to a button press. Three buttons are installed on the prototype and each control a specific outcome. The yellow button changes the state of the system: heating, cooling, and off. The Gray button increases the user set temperature by 1 degree Fahrenheit. The Blue button decreases the user set temperature by 1 degree Fahrenheit. The set temperature is automatically 72 degrees upon program start and the user is then able to use the buttons to change the set temperature as desired. When the state is changed to heating, the lcd screen displays “changing state to heat” and the red LED light turns on. If the current temperature in the room is less than or equal to the set temperature at any given time, this light will pulse instead of remaining solid. When the state is changed to cooling, the red LED light turns off and the blue LED light turns on while the lcd screen displays “changing state to cool”. If the current temperature in the room is greater than the set temperature at any given time, the light will pulse instead of remaining solid. When the state is changed to off, the blue LED light turns off and the red LED light remains off with it. The lcd screen then displays “changing state to off”.

The next version of this build will change from using the UART serial port. It will use REST API’s to send data specifically over wifi. This data will be sent to Systec software servers as needed in real time. This capability can be achieve through libraries available in Python. This will be kept secure by only accepting HTTPS transmissions. When transferring data, the next system will be utilizing the cloud and updating data in real time. When deciding on the next version of this system, there are architecture options. We could use an NXP, a Microchip, or a Raspberry Pi again. The chosen architect needs to work well with the cloud, have plenty of hardware specs for great performance, and has to support the already mentioned peripherals. When looking at the Microchip, we could run into some memory issues down the line, and is specific to the C coding language. Another option is the Freescale route. When looking at this option, we could run into compatibility and difficult setup issues. In addition to these hiccups, we would also need to purchase, set up hardware, and install modules to experience wifi capabilities. The Freescale architecture is a good microcontroller for low energy systems, but for this one, we need something that supports more memory. For all of these reasons, the Raspberry Pi is the architecture we will keep for future temperature sensor projects at SysTec.

Sources:

G. C. Fox, S. Kamburugamuve and R. D. Hartman, "Architecture and measured characteristics of a cloud based internet of things," 2012 International Conference on Collaboration Technologies and Systems (CTS), Denver, CO, USA, 2012, pp. 6-12, doi: 10.1109/CTS.2012.6261020.

Raspberry Pi Foundation. (n.d.). *Raspberry Pi documentation*. https://www.raspberrypi.com/documentation/

<https://www.nxp.com/docs/en/reference-manual/FSPRM.pdf>

Microchip Technology. (n.d.). *Microcontrollers*. https://www.microchip.com/en-us/products/microcontrollers

<https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/uart.html>

NXP Semiconductors. (n.d.). *Kinetis K Series Reference Manual*. https://www.nxp.com/docs/en/reference-manual/FSPRM.pdf

<https://www.tomshardware.com/news/raspberry-pi-4-how-much-ram>

<https://cloud.google.com/apis/docs/http>

<https://www.adafruit.com/product/4566?srsltid=AfmBOorwyfpiimPJot4sKGl_28pvrFyyYiBKFIyOnq16wAHije-D5NcO>