



SYSTEM REQUIREMENT SPECIFICATION

CS350-Thermostat IoT

Final Project

Gonzalo Patino
Gonzalo.patino@snhu.edu

Contents

1. Purpose	2
2. Scope	3
3. System Overview	4
4. Functional Requirements	5
4.1 Hardware Requirements	5
UART Output.....	6
4.2 Firmware Requirements	7
5. Non-Functional Requirements	9
5.1 Hardware	9
5.2 Firmware	9
5.3 Cloud Dashboard (Future Phase).....	9
6. Future Expansion	10
7. System Specifications	11
7.1 Hardware Specifications.....	11
7.2Firmware Specifications	12
8. Design Overview.....	13
8.1 System	13
8.1.1 System Context Diagram	13
8.2 Hardware	14
8.3 Software	18
Flattened FSM (simple to code)	19
Events and inputs	19
Transition rules	20

1. Purpose

The purpose of this project is to design, implement, and test a functional prototype of a smart thermostat.

The system will monitor temperature and humidity, control heating and cooling modes, display operational data, and transmit information over UART.

Additionally, the next phase will incorporate a cloud-based dashboard for real-time monitoring, historical trend analysis, and remote-control functionality.

2. Scope

The scope of this project includes the development of a physical thermostat prototype using a Raspberry Pi and connected peripherals, the implementation of heating and cooling control logic, and user input through physical push buttons. It also covers the use of LEDs as visual indicators, displaying data on an LCD screen, and providing UART serial output of the system state.

Additionally, the scope includes creating the design and specifications for a future cloud-based dashboard.

3. System Overview

The thermostat will operate in **OFF**, **HEATING**, or **COOLING** mode.

User input will be provided via three physical push buttons or via remote control from a cloud-based dashboard (future phase).

Environmental data will be obtained using an AHT20 temperature and humidity sensor and displayed on a 16x2 LCD.

Two LEDs will visually indicate heating or cooling status, while system state will also be transmitted over UART.

4. Functional Requirements

4.1 Hardware Requirements

1. Temperature and Humidity Sensing

- The system shall use an AHT20 sensor via I2C to read temperature and humidity.

2. Physical User Input

- Button 1 shall toggle between OFF, HEATING, and COOLING modes.
- Button 2 shall increase the temperature set point by 1°F.
- Button 3 shall decrease the temperature set point by 1°F.

3. LED Indicators

- Red LED shall indicate HEATING mode:
 - Fade when temperature < set point.
 - Solid when temperature \geq set point.
- Blue LED shall indicate COOLING mode:
 - Fade when temperature > set point.
 - Solid when temperature \leq set point.

4. LCD Display

- First line shall display date and time.
- Second line shall alternate between:
 - Current temperature and humidity.

UART Output

- The system shall transmit data every 30 seconds in the following comma-separated format:

"state,current_temperature,current_humidity,set_point".

4.2 Firmware Requirements

5. Default Set Point

- The system shall initialize with a default set point of 72°F.

6. State Machine

- The firmware shall implement OFF, HEATING, and COOLING modes with defined transitions based on button input.

7. Control Logic

- In HEATING mode:
 - If temperature < set point → fade red LED.
 - If temperature \geq set point → solid red LED.
- In COOLING mode:
 - If temperature > set point → fade blue LED.
 - If temperature \leq set point → solid blue LED.

8. LCD User Interface

- The firmware shall update the LCD display without noticeable flicker.
- The display shall alternate between data views at a fixed interval.

9. UART Communication

- The firmware shall handle UART communication with error checking.

10. Cloud Dashboard Integration (Future Phase)

- The system shall support a secure API for cloud connectivity.
- Remote commands shall replicate the functionality of the three physical push buttons.
- The cloud dashboard shall:

- Display real-time temperature and humidity readings.
- Plot historical trends for temperature and humidity over time.
- Show the current thermostat mode, set point, and environmental readings.
- Allow remote mode switching and set point adjustments.

5. Non-Functional Requirements

5.1 Hardware

- All components shall operate at 3.3V logic levels.
- Breadboard wiring shall ensure secure, stable connections.
- Buttons shall provide tactile feedback and respond within 50 ms.
- LEDs shall be visible under standard indoor lighting.

5.2 Firmware

- Temperature readings shall refresh at least once per second.
- LCD updates shall not block main control loops.
- UART communication shall ensure less than 1% transmission error rate.
- Button inputs shall be debounced in software.

5.3 Cloud Dashboard (Future Phase)

- Dashboard shall have latency of less than 2 seconds for remote commands.
- Dashboard shall store at least 12 months of historical data.
- Dashboard shall be accessible via modern web browsers and mobile devices.
- Communication between thermostat and cloud shall use encrypted protocols (TLS 1.2 or higher).

6. Future Expansion

- Integrate Wi-Fi-enabled microcontroller for production version.
- Add adaptive temperature control mode.
- Support additional sensors such as occupancy detection for energy optimization.

7. System Specifications

7.1 Hardware Specifications

Component	Specification	Justification
Processing Unit	Raspberry Pi 4 Model B (4GB RAM) for prototype; Wi-Fi-enabled MCU (ESP32) for production	Raspberry Pi allows rapid prototyping with Linux and Python; ESP32 offers low cost, integrated Wi-Fi, and low power consumption for production
Temperature & Humidity Sensor	AHT20 via I2C, accuracy $\pm 0.3^{\circ}\text{C}$, $\pm 2\%$ RH	Compact, accurate, I2C compatible
User Input	3x momentary push buttons with $10\text{K}\Omega$ pull-up resistors	Simple GPIO control, low cost
Indicators	2x LEDs (red & blue), PWM controlled	Visual feedback for system state
Display	16x2 I2C LCD	Compact, low power, easy to read
Connectivity	UART (debug), Wi-Fi (MQTT/HTTPS)	UART for local debugging, Wi-Fi for cloud integration
Power Supply	5V DC, 2.5A	Supports Pi, LCD, and peripherals

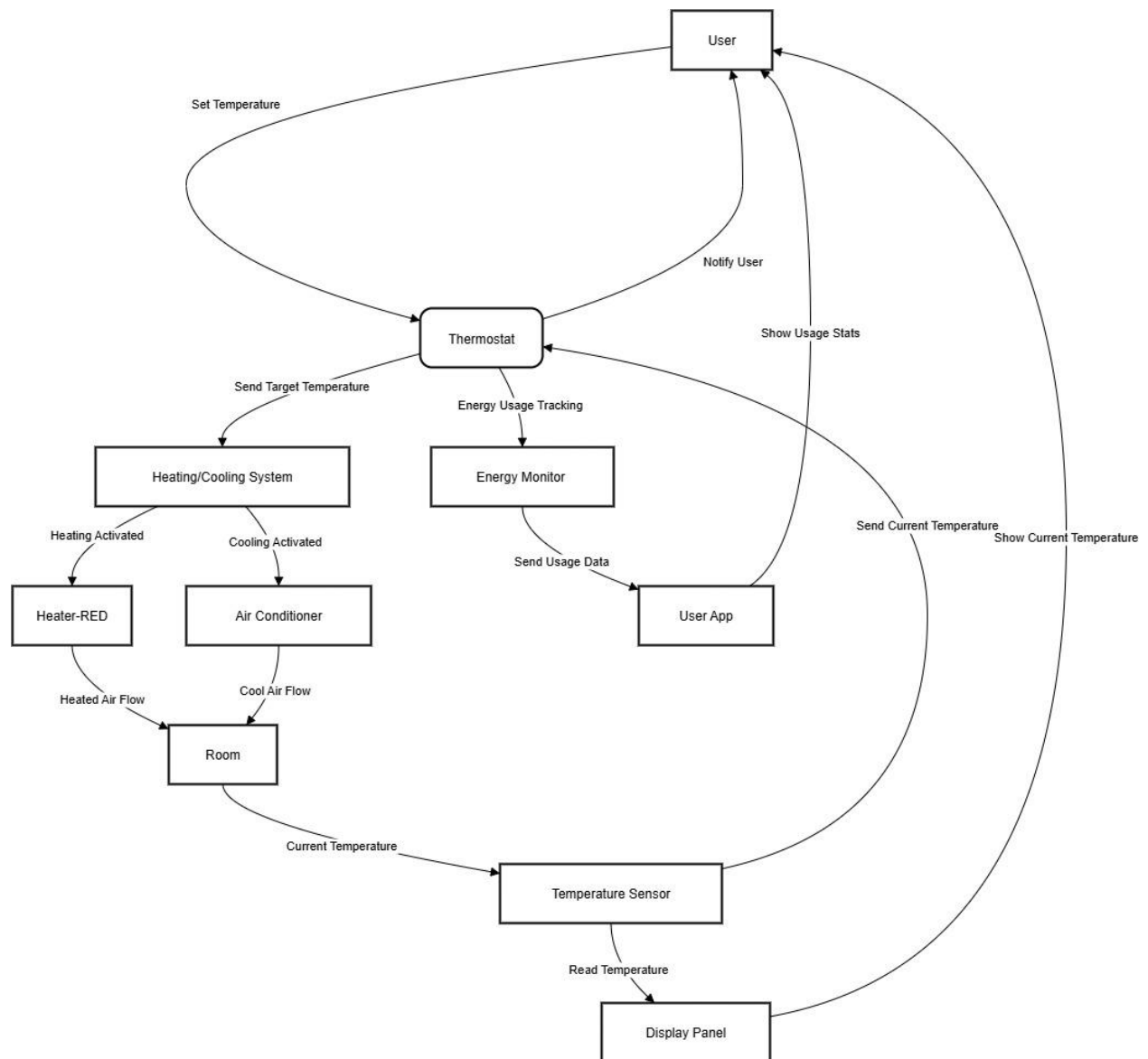
7.2 Firmware Specifications

Function	Specification
Default Set Point	72°F
State Machine	OFF → HEATING → COOLING with button input or cloud command
Temperature Control	HEATING: Fade red LED when temp < set point, solid when ≥ set point; COOLING: Fade blue LED when temp > set point, solid when ≤ set point
LCD Display	Line 1: Date/time; Line 2: Alternates between current temp/humidity and state/set point
Sensor Polling Rate	1 Hz
UART Output	Every 30 seconds: "state,temp_C,humidity_pct,set_point_F"
Cloud Communication	MQTT to AWS IoT Core, JSON payload format
Security	TLS 1.2 mutual authentication for MQTT

8. Design Overview

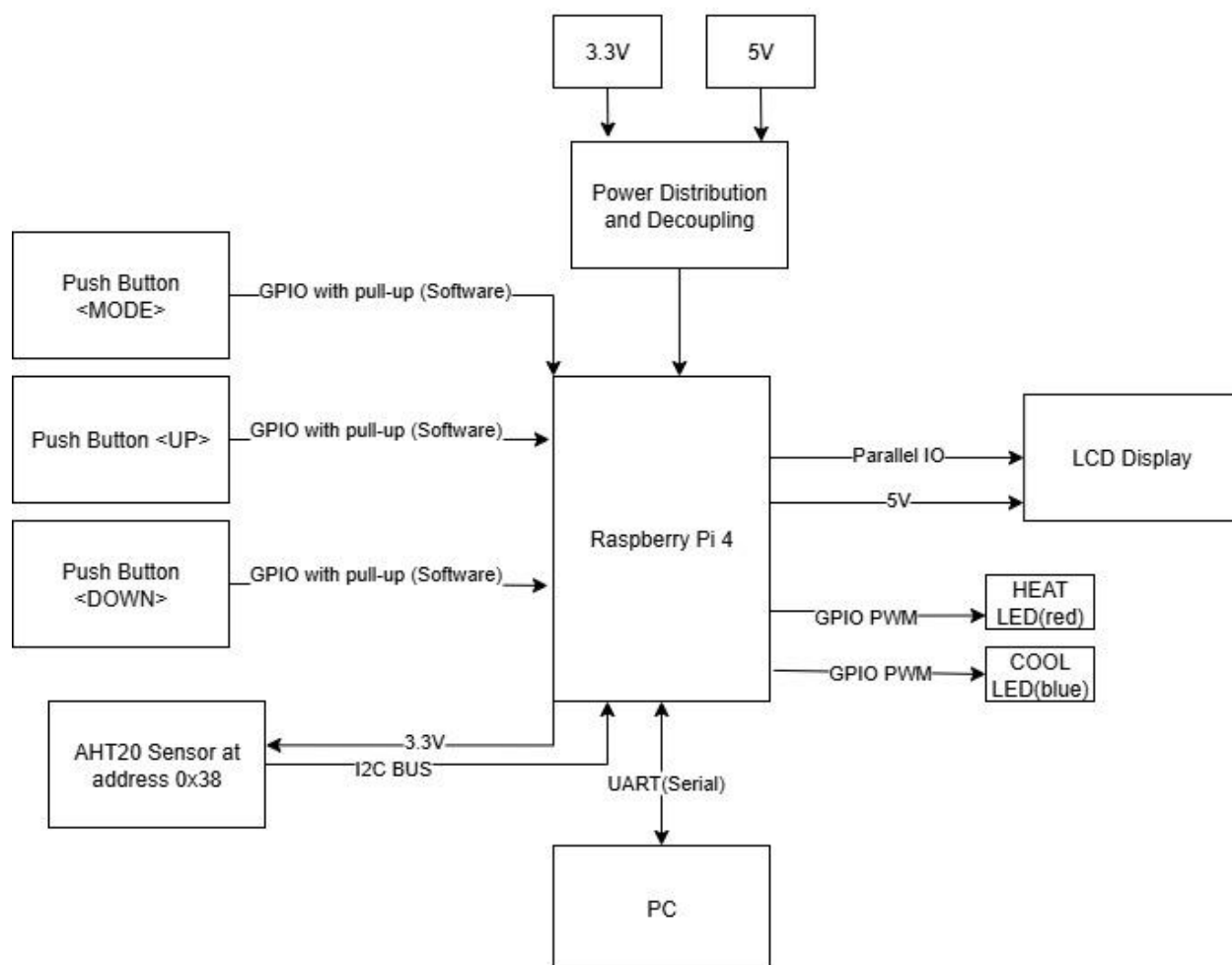
8.1 System

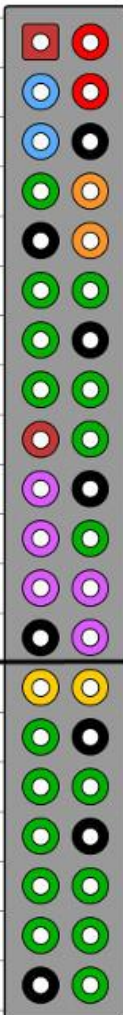
8.1.1 System Context Diagram



8.2 Hardware

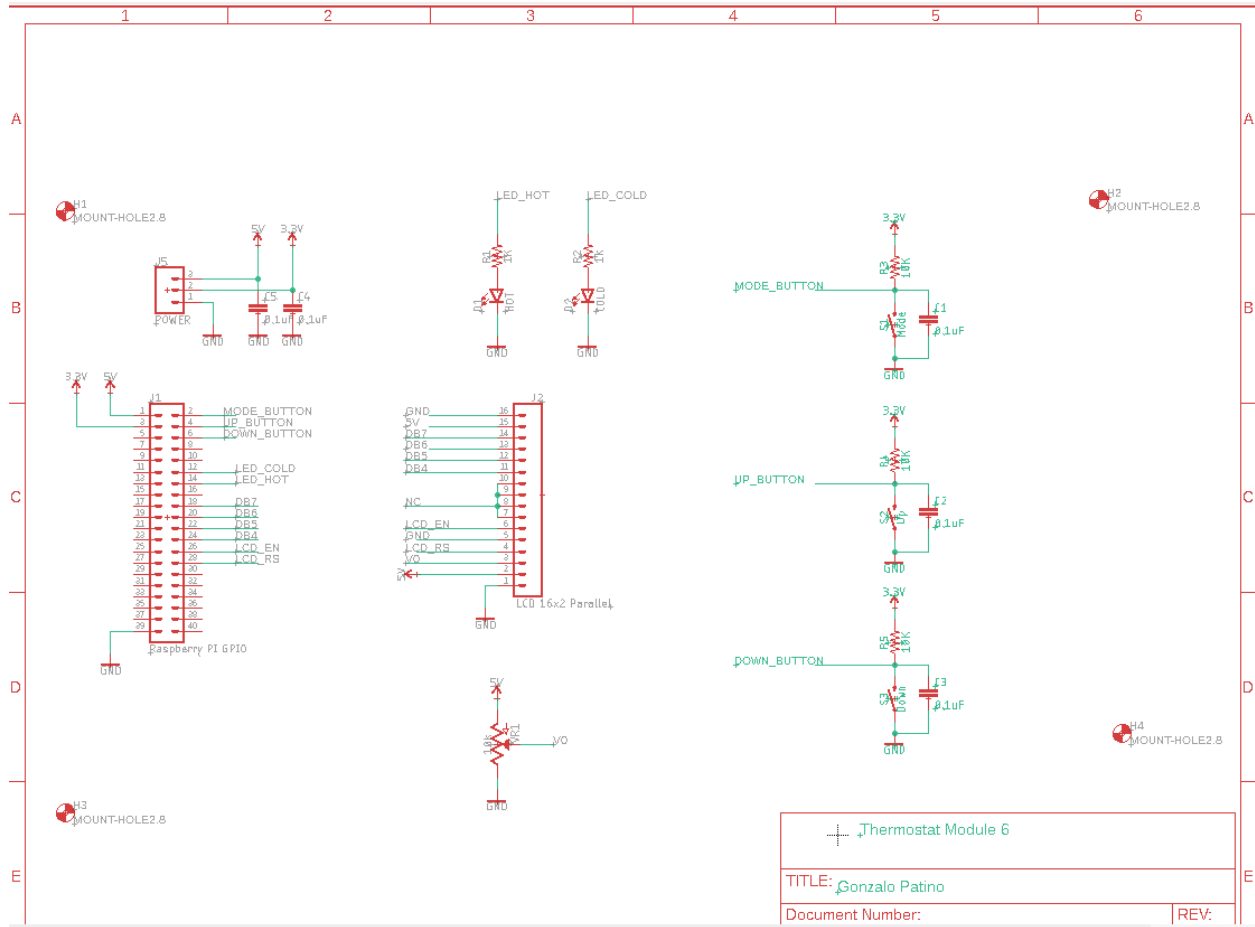
Hardware Block Diagram

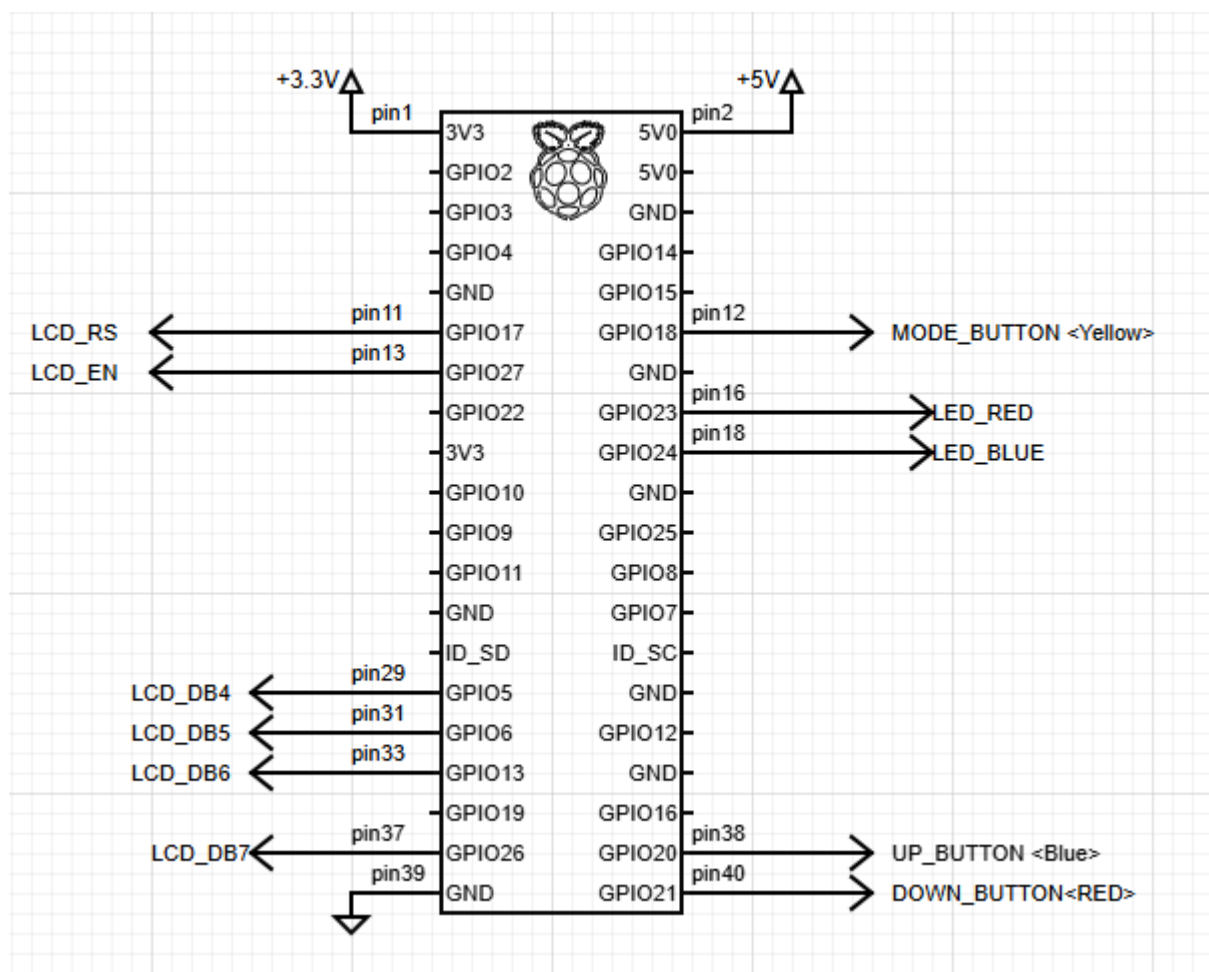


Raspberry Pi2 GPIO Header				
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 1
26/01/2014

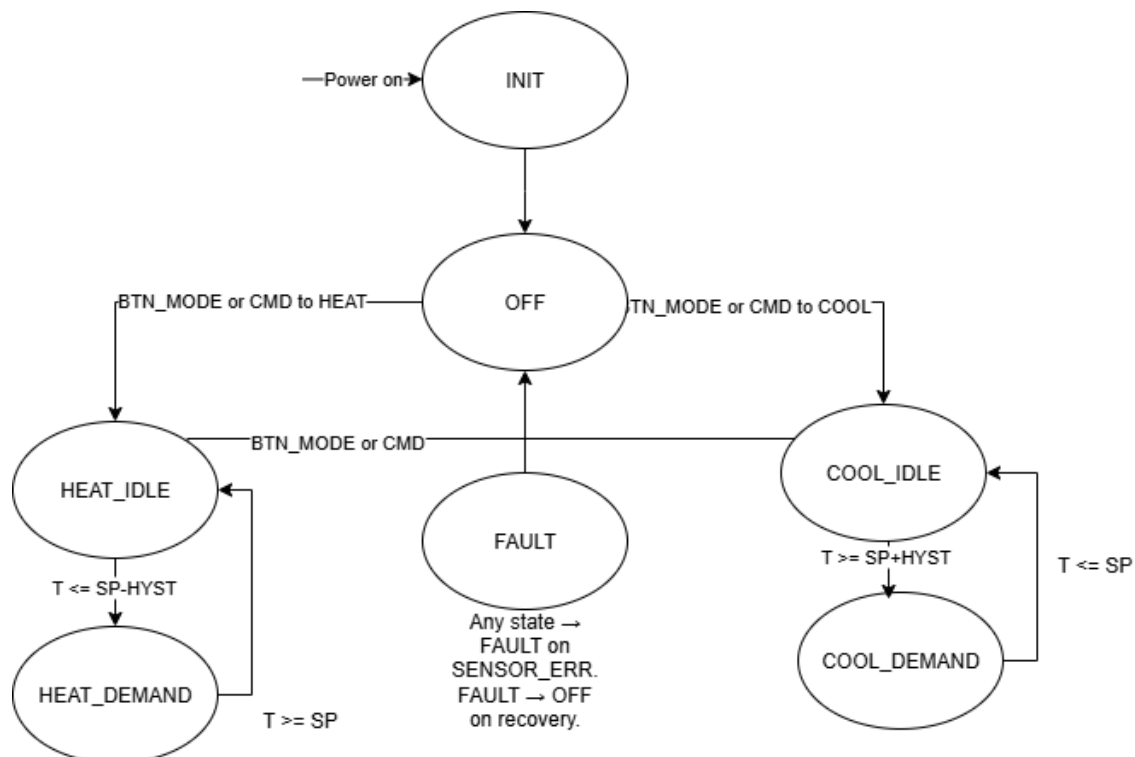
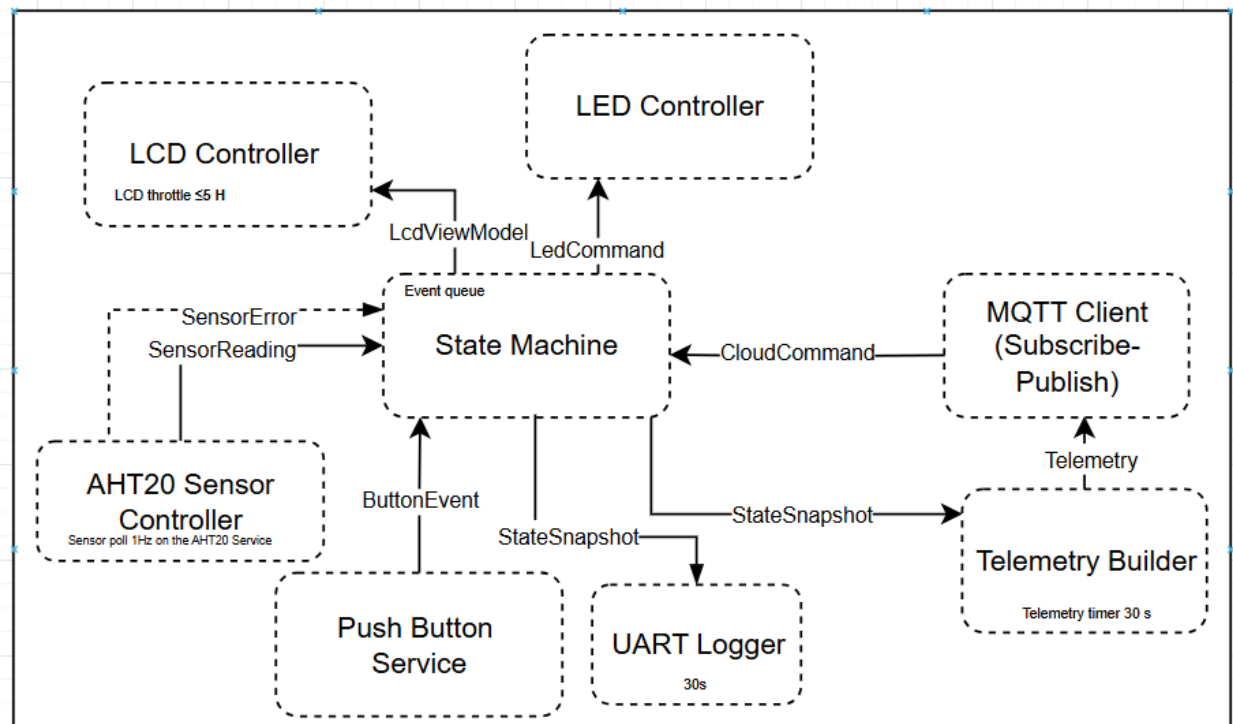
<http://www.element14.com>





8.3 Software

Software Container Diagram



Flattened FSM (simple to code)

Use one FSM with these states:

- **INIT**: hardware init, sensor warm-up, load default set point (72 F), clear displays
- **OFF**: system idle, outputs off
- **HEAT_IDLE**: heating mode, no demand
- **HEAT_DEMAND**: heating mode, demand active
- **COOL_IDLE**: cooling mode, no demand
- **COOL_DEMAND**: cooling mode, demand active
- **FAULT**: sensor or system error detected

Recommend a small hysteresis to prevent chatter: $HYST = 1.0\text{ F}$ (tuneable).

Events and inputs

- **BTN_MODE** short press: cycle OFF → HEAT → COOL → OFF
- **BTN_UP**: $set_point += 1\text{ F}$ (cap at max, for example 86 F)
- **BTN_DOWN**: $set_point -= 1\text{ F}$ (cap at min, for example 50 F)
- **CLOUD_CMD**: same actions as buttons (mode switch or set point change)
- **TEMP_UPDATE**: new sensor sample at 1 Hz
- **TIMER_30S**: push UART frame
- **TIMER_UI**: alternate LCD line 2 content every N seconds
- **SENSOR_ERR or RECOVERED**: sensor health change

Transition rules

Let T = current temperature, SP = set point, $HYST = 1.0$ F.

- **INIT → OFF**
 - After peripherals init, first valid sensor read, and LCD ready.
- **OFF → HEAT_IDLE**
 - BTN_MODE or $CLOUD_CMD$ selects HEAT.
- **OFF → COOL_IDLE**
 - BTN_MODE or $CLOUD_CMD$ selects COOL.
- **HEAT_IDLE → HEAT_DEMAND**
 - On $TEMP_UPDATE$, if $T \leq SP - HYST$.
- **HEAT_DEMAND → HEAT_IDLE**
 - On $TEMP_UPDATE$, if $T \geq SP$.
- **COOL_IDLE → COOL_DEMAND**
 - On $TEMP_UPDATE$, if $T \geq SP + HYST$.
- **COOL_DEMAND → COOL_IDLE**
 - On $TEMP_UPDATE$, if $T \leq SP$.
- ***Any HEAT_ → Any COOL_** and Any COOL_ → OFF/HEAT_*****
 - BTN_MODE cycles mode, or $CLOUD_CMD$ sets mode directly.

- When switching mode, go to the corresponding *_IDLE state first, then allow demand logic to move into *_DEMAND on the next TEMP_UPDATE.
- **Any state → FAULT**
 - On SENSOR_ERR or critical I2C failure.
- **FAULT → OFF**
 - On RECOVERED (valid sensor restored) or after manual reset.

Entry actions per state

- **INIT**
 - Configure GPIO, start PWM for LEDs (off initially), init I2C devices (AHT20, LCD), load SP = 72 F, show splash on LCD.
- **OFF**
 - LEDs off, LCD line 2 shows OFF, SP=XX F.
- **HEAT_IDLE**
 - Red LED solid on, blue LED off.
- **HEAT_DEMAND**
 - Red LED fading (PWM), blue LED off.
- **COOL_IDLE**
 - Blue LED solid on, red LED off.
- **COOL_DEMAND**

- Blue LED fading (PWM), red LED off.

- **FAULT**
 - Blink both LEDs slowly, LCD line 2 shows FAULT, UART includes an error flag.