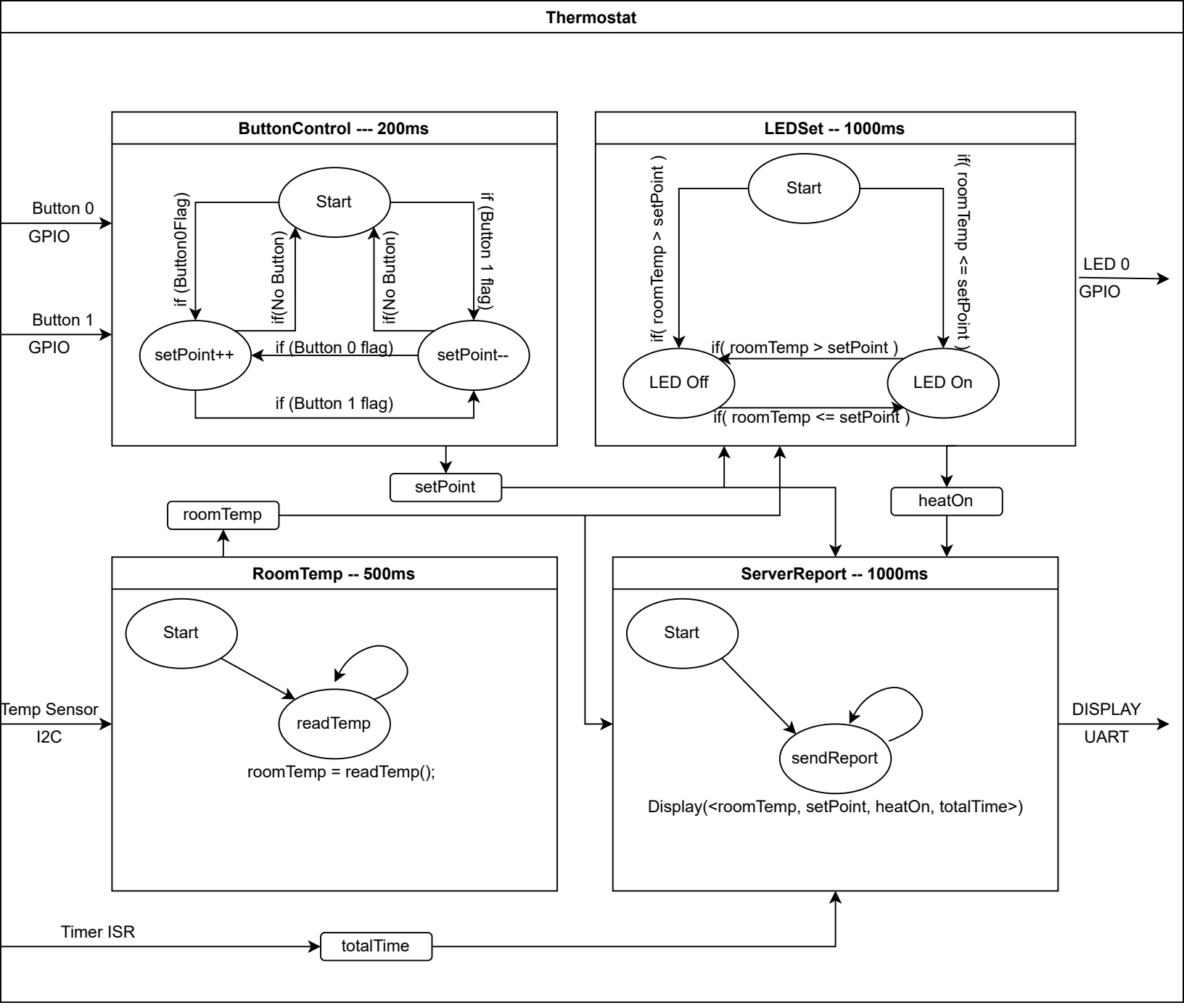
Task Scheduler

The task scheduler is used to create a modular and reusable way of managing multiple concurrent state machines with different periods. The task scheduler also enables us to move the state machines to their own structs and functions improving readability.

In order to utilize the task scheduler a few steps must be taken. First a struct definition for a task must be created with the attributes: state, period, elapsedTime, and TickFunction. Next an array of task struct instances must be created. This will allow the task scheduler to iterate through the tasks. Finally a tick function must be defined for each task. This is where the state machine for each task will be executed.

The algorithm used for the task scheduler is a fairly simple loop that is called each timer the timer interrupt is activated. Each time the task scheduler is called it loops through all of the current tasks. The task scheduler checks the elapsed time since each task last "ticked" and if that time is equal to or greater than the period of that task, the corresponding state machine is called and the elapsed time reset. After each pass the task scheduler adds the period of the timer interrupt to the elapsed time of each task.


Thermostat
ButtonControl --- 200ms
Start
if (Button0Flag)
if (Button 1 flag)
if(No Button)
if(No Button)
setPoint++
setPoint--
if (Button 0 flag)
if (Button 1 flag)
Button 0
GPIO
Button 1
GPIO
LEDSet -- 1000ms
Start
if(roomTemp > setPoint)
if(roomTemp <= setPoint)
if(roomTemp > setPoint)
LED Off
LED On
if(roomTemp <= setPoint)
LED 0
GPIO
setPoint
roomTemp
heatOn
RoomTemp -- 500ms
Start
readTemp
roomTemp = readTemp();
Temp Sensor
I2C
ServerReport -- 1000ms
Start
sendReport
Display(<roomTemp, setPoint, heatOn, totalTime>)
DISPLAY
UART
Timer ISR
totalTime