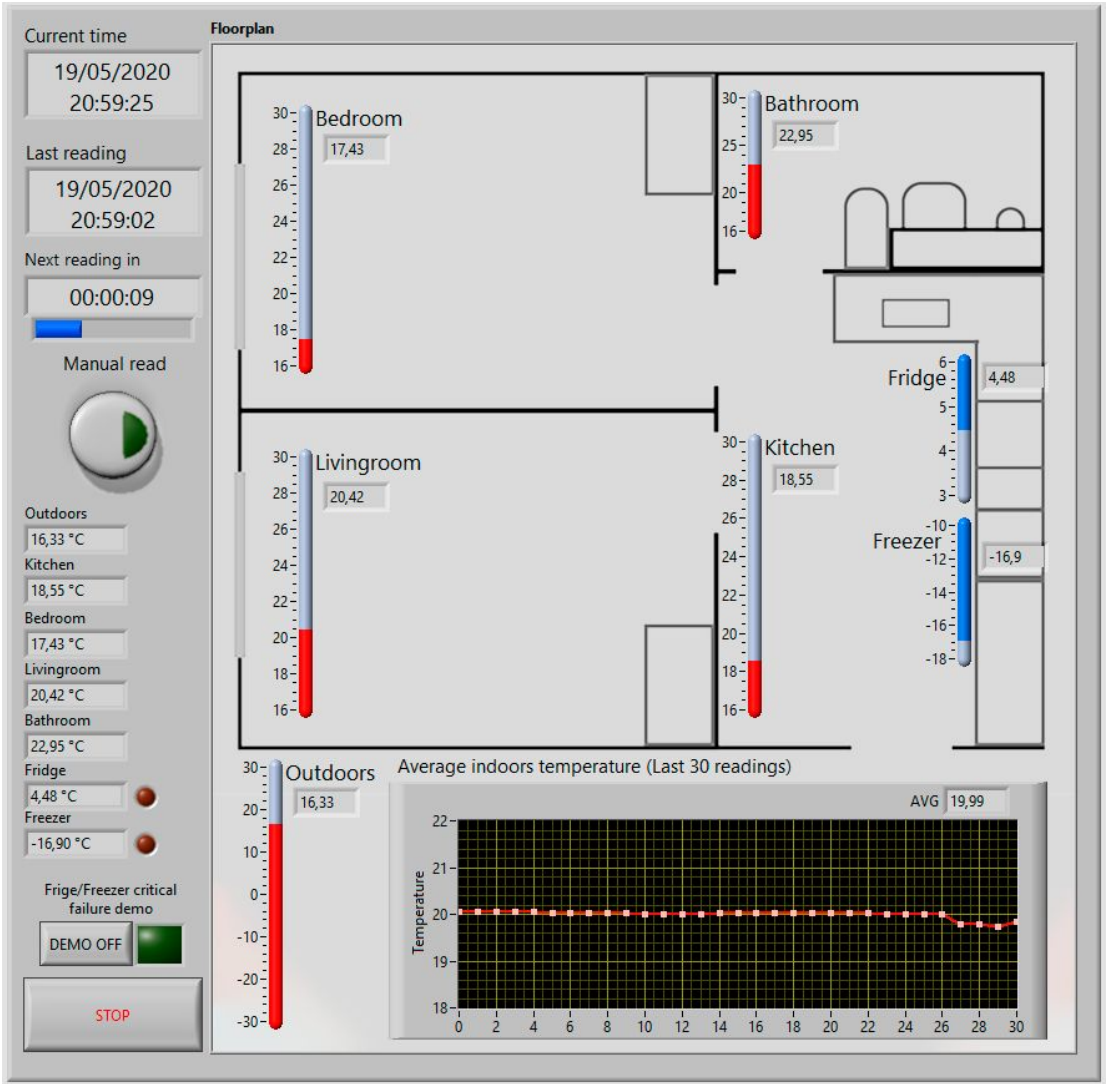
Home temperature logger



Metropolia University of applied sciences
Smart systems, Virtual instrumentation
project documentation
Mikko Larke, 19.5.2020

Metropolia University of applied sciences
Smart systems, Virtual instrumentation
project documentation
Mikko Larke, 19.5.2020

Key features

- Temperature logging for multiple rooms, outdoors and fridge/freezer
- User adjustable log file name and automatic polling rate
- Temperature visualisations over floorplan
- List of all temperatures for fast comparing
- Average temperature of indoors shown in trend
- Warning indicators when fridge/freezer temperatures are critical
- Visual representation of current time, timestamp of last reading and time left until next reading
- Every reading logged in formatted excel file
- Interactive UI
- Communication with MCU
- Can run two different programs from MCU
    - By default set to temperature readings and demo of fridge/freezer failure

State functions

State	functions
Init	● Initialize variables ● Show startup config elements ● Hide UI indicators
Set_startup_config	● Set name for log file, polling rate and serial port ● Build file path for log file
Check_startup_config	● Check file path ○ Append to existing file or create new file ● Check valid serial port and time formats ○ Return to config and show warning if needed
Read_serial	● Open and configure serial connection ● Run one of configured UART modes ● Show UI indicators ● Hide startup config elements
Check_serial	● Check that received all readings from serial connection ● If valid save readings to log file ● Also appends trend array with new indoors average ● If reading wasn't valid poll readings again
Update	● Update UI indicators with new readings. ● Shown UI elements: ○ List of temperatures ○ Thermometers over floorplan ○ Indoors trend ○ Fridge/Freezer failure warnings ○ Demo mode indicator
Wait	● Wait for next temperature reading ● Show time until next reading ● Get reading manually with button ● Toggle UART mode

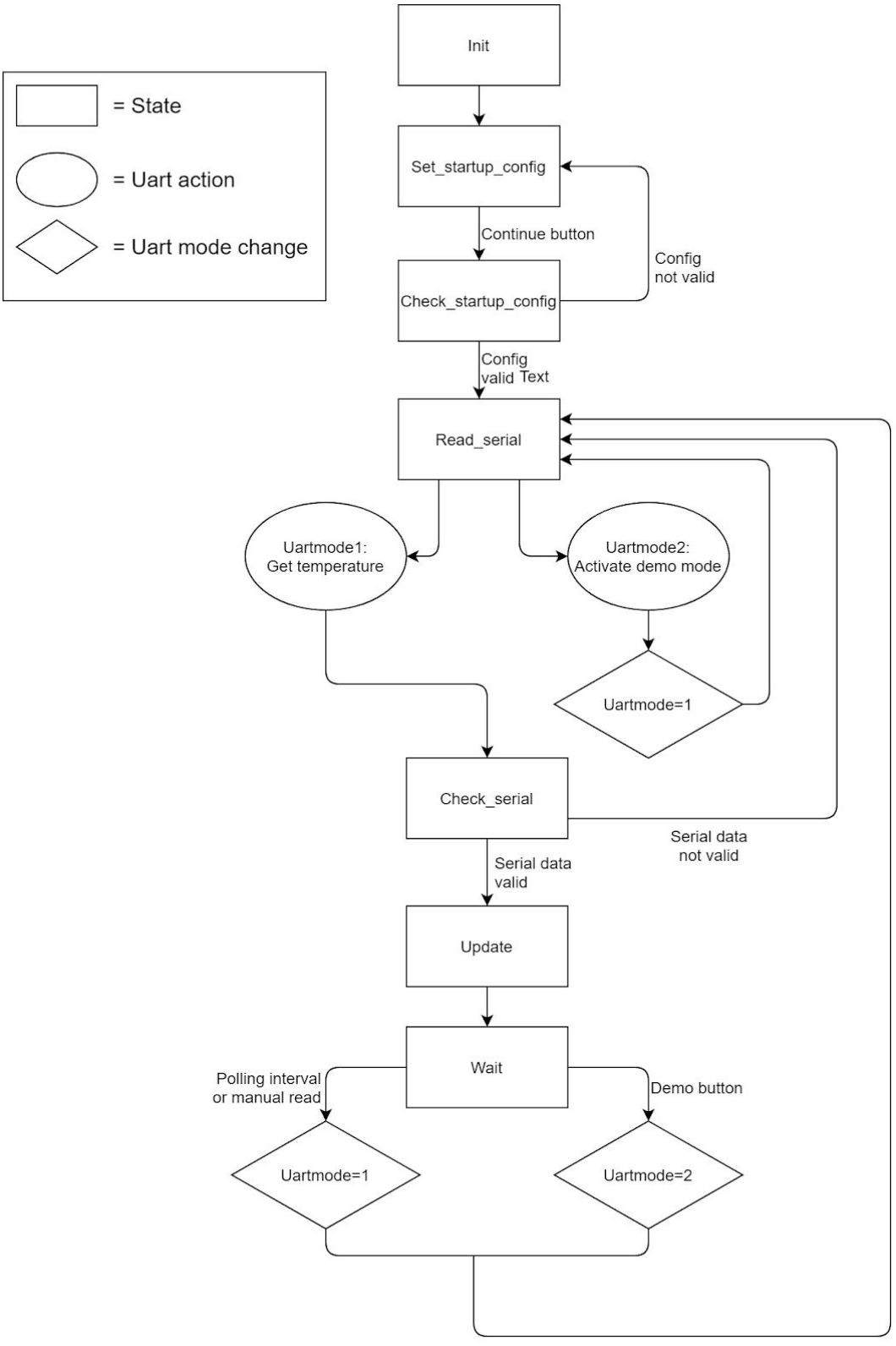
Uart mode functions

Uart mode	functions
Get temperatures	● Send command to MCU for sending the temperature string ● Read temperature string from MCU ● Add timestamp to temperature string ● Build 2 arrays from temperature string ○ String array for log file ○ Float array for UI ● Close serial connection ● Change state tag for next state
Activate demo mode	● Send command to MCU for changing to fridge/freezer fail demo ● Turn demo mode indicator led on/off ● Close serial connection ● Return to Read_serial state with get temperatures UART mode

Sub VI's functions

Sub VI	Functions
hsm_to_s	● Takes in three integers (hour, minute and second) ● Outputs hours, minutes and seconds converted to seconds ● Range of each input is set to be valid (hour 0-23, seconds and minutes 0-59)
FileExistence	● Takes in file path ● Checks if file exists in given path ● If file doesn't exist vi creates that file ● Vi also creates appropriate file header automatically

Flow chart


= State
= Uart action
= Uart mode change
Init
Set_startup_config
Continue button
Config
not valid
Check_startup_config
Config
valid Text
Read_serial
Uartmode1:
Get temperature
Uartmode2:
Activate demo mode
Uartmode=1
Check_serial
Serial data
not valid
Serial data
valid
Update
Wait
Polling interval
or manual read
Demo button
Uartmode=1
Uartmode=2

C++ demo code

```
1
    //Create base values of each temperature sensor according following formula:
    //<Classname> <description>(<initial temperature>,<maximum change step>,
    //<minimum target temperature>,<maximum target temperature>)
    ThermalData outdoors(15,0.9,-25,30);
    ThermalData kitchen(20,0.15,17,23);
    ThermalData bedroom(22,0.15,15,20);
    ThermalData livingroom(21,0.15,18,22);
    ThermalData bathroom(23.5,0.15,18,26);
    ThermalData fridge(4,0.01,3,6);
    ThermalData freezer(-17,0.01,-18,-15);

    //Randomizer seed taken from time
    srand (time(NULL));

    int tempChangeCount=0;  //When this reaches certain value temperature is changed
    int uartcode;           //Code from labview to run correct code
    char buffer[256];       //buffer for character string containing temperatures
    bool demo=false;        //keep track if fridge/freezer fail demo is on/off

    while(1){
        tempChangeCount++;
        //Change temperature of each "sensor"
2       if(tempChangeCount==500){
            outdoors.changeTemp();
            kitchen.changeTemp();
            bedroom.changeTemp();
            livingroom.changeTemp();
            bathroom.changeTemp();
            fridge.changeTemp();
            freezer.changeTemp();

            tempChangeCount=0;
        }
        //Parse temperatures to one string
3       snprintf(buffer,256,"%.2f;%.2f;%.2f;%.2f;%.2f;%.2f;%.2f;%.2f\n\r",
                outdoors.getTemp(),kitchen.getTemp(),bedroom.getTemp(),
                livingroom.getTemp(),bathroom.getTemp(),fridge.getTemp(),freezer.getTemp());

        //Read UART code from VI
4       uartcode = Board_UARTGetChar();
        switch(uartcode){
        //get temperatures
        case '1':
            //Send parsed string via UART
            Board_UARTPutSTR(buffer);
            ITM_write("String sent to VI\n");
            break;
            //turn fridge/freezer fail demo on/off
        case '2':
            if(!demo){
                fridge.setTemp(15);
                freezer.setTemp(5.27);
                demo=true;
                ITM_write("Demo on\n");
            }
            else {
                fridge.setTemp(4);
                freezer.setTemp(-17);
                demo=false;
                ITM_write("Demo off\n");
            }

            break;
            //Show temperature string in console for testing
        default:    //
            if(tempChangeCount==0){
                ITM_write(buffer);
            }
        }
        //Sleep for 10ms
        Sleep(10);
    }
```

```
class ThermalData {
public:
    ThermalData(float temp_in, float step_in, float minTemp_in, float maxTemp_in);
    virtual ~ThermalData(){};
    float changeTemp();
    float getTemp(){return temp;}
    void setTemp(float newTemp);

private:
    float temp;
    float step;
    float maxTemp;
    float minTemp;
};
```

```
ThermalData::ThermalData(float temp_in, float step_in, float minTemp_in, float maxTemp_in){
    temp=temp_in;
    step=step_in;
    maxTemp=maxTemp_in;
    minTemp=minTemp_in;

}

5   float ThermalData::changeTemp(){
        float min,max;

        if(temp <= minTemp){
            min=0;
            max=step*10;
        }
        else if(temp >= maxTemp){
            min=-(step*10);
            max=0;
        }
        else{
            min=-step;
            max=step;
        }

        float random = ((float) rand()) / (float) RAND_MAX;
        float range = max - min;
        temp+= (random*range) + min;

        return temp;
    }
void ThermalData::setTemp(float newTemp){
    temp=newTemp;
}
```

1. Initialize each demo sensor with starting temperature, step size of change and range of target temperatures
2. Change temperature of each sensor every 5 seconds (500*10ms)
3. parse temperature string from last sensor values
4. Code has 2 modes set by UART code send by VI, which is select by switch case.
   - 4.1. Send temperature string to VI by UART
   - 4.2. Turn fridge/freezer failure demo on/off
5. Temperature change is simulated by getting random float between negative and positive step size. If temperature exceeds set temperature range temperatures only approach values within the range

Features to be added

- Compatibility with more than 2 UART modes
- Ability to change file and polling rate without restarting program
- Ability to set polling rate greater than 24 hours
- Option for users to append indoors trend from log file
- Option for users to change how many readings trend shows at time