Greenhouse

Dorin Botan & Eriks Markeviks

Intro

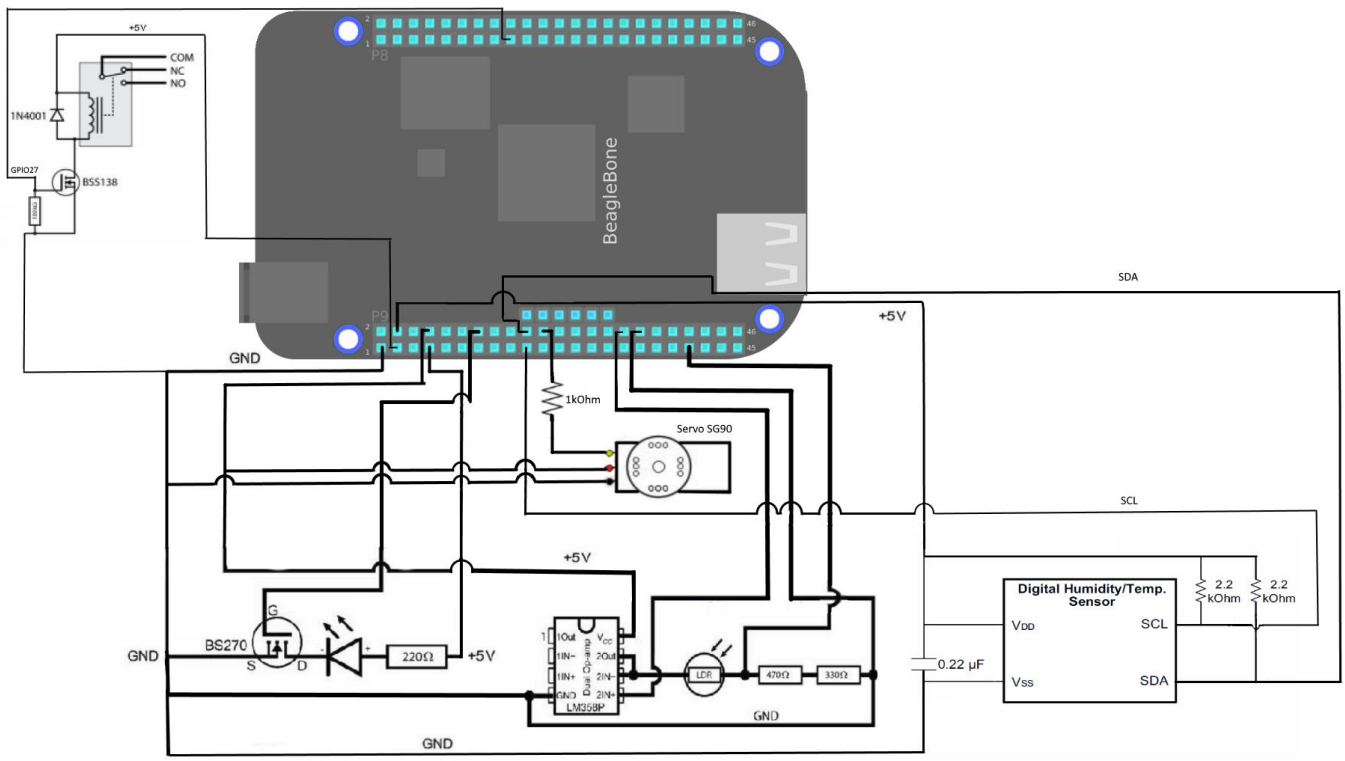
Following is the description of the smart greenhouse project developed as a final project for EOS1-A17 course. The project consists of three parts – physical circuit built around the BBB board and containing sensors and actuators for monitoring and controlling the greenhouse, a web server providing remote access to the circuit and a client application for easy tracking and controlling of the greenhouse.
The project is currently in proof-of-concept stage and was designed with an initial scope of demonstrating the capabilities of the system. The number of features had a higher priority than code quality.

Features

- Remote monitoring and control of temperature, humidity and lighting
- Automatic temperature, humidity and lighting control
- Regular logging of temperature, humidity and lighting
- Representation of data in an easy to read form

Hardware

Electric circuit follows the instructions given during the EOS1-A17 course and additional explanations will be superfluous. Following is the complete circuit diagram.



Server

Web server runs over TCP on port 8091. The greenhouse is accessible via LAN through a RESTful interface. Data is being exchanged as simple string streams.
Following is the complete set of available methods.

	GET	POST	DELETE
temperature	Get temperature value in °C	Automaticaly keep temperature at the given value	Disable automatic temperature control (disables heater)
temperature/mode	Get temperature control mode [0 - manual, 1 - auto]	Automaticaly keep temperature at its current value	Disable automatic temperature control (disables heater)
humidity	Get humidity value in %	Automaticaly keep humidity at the given value	Disable automatic humidity control (closes the lid)
humidity/mode	Get humidity control mode [0 - manual, 1 - auto]	Automaticaly keep humidity at its current value	Disable automatic humidity control (closes the lid)
light	Get light level raw value	Automaticaly keep light at the given value	Disable automatic light level control (turns off the lamp)
light/mode	Get light control mode [0 - manual, 1 - auto]	Automaticaly keep light at its current value	Disable automatic light level control (turns off the lamp)
heater	Get light control mode [0 - manual, 1 - auto]	Manually set heater to the given value	Automaticaly keep temperature at its current value
lid	get lid state in degrees	Manually set lid to the given value	Automaticaly keep humidity at its current value
lamp	get get lamp value in %	Manually set lamp to the given value	Automaticaly keep light at its current value

The web server is entirely implemented in Qt/C++ with using Signal & Slots system and can handle multiple clients at a time.

Automatic temperature, humidity and lightning control is carried out by different threads and is also implemented in Qt/C++.
Following is the thread controlling the light.

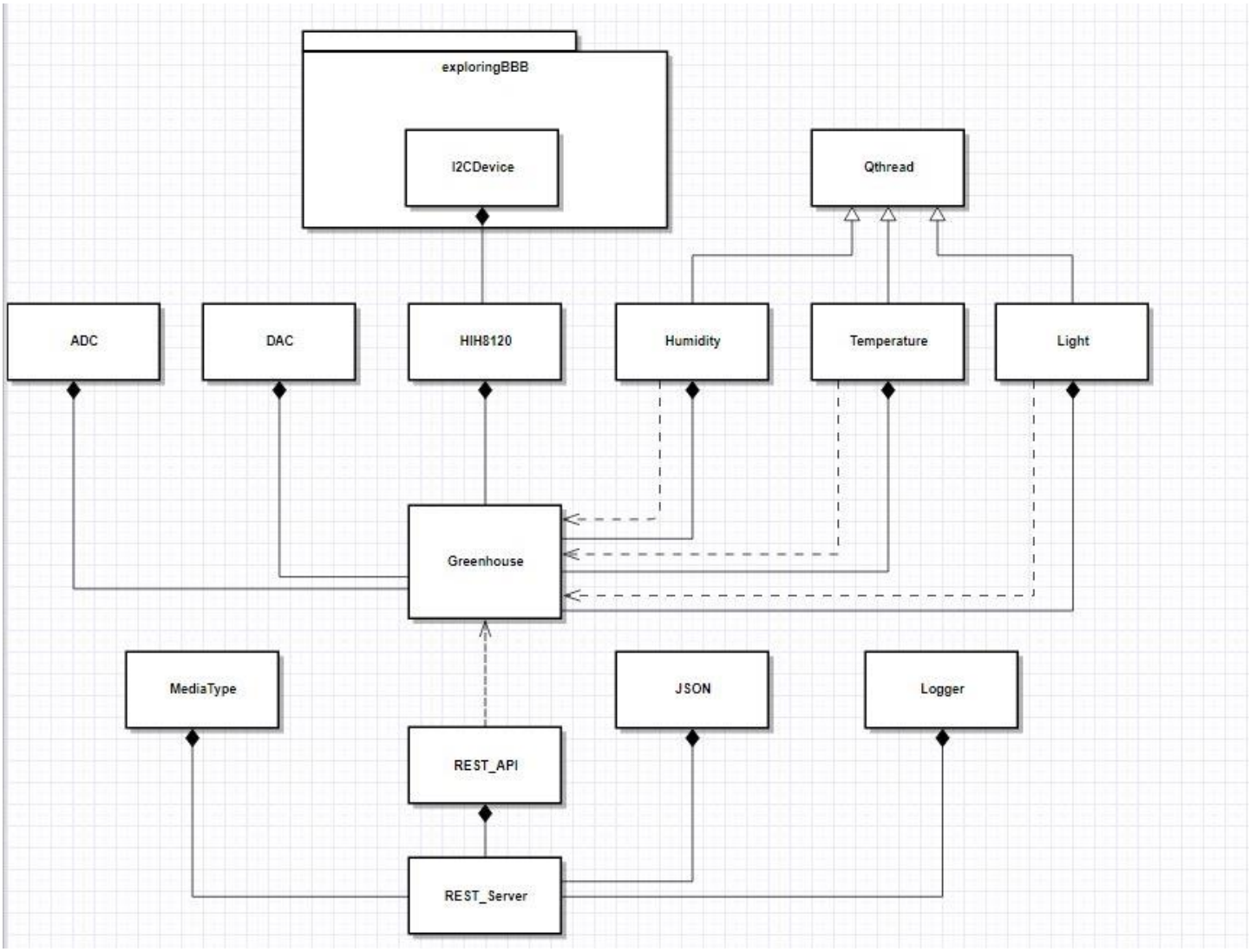
```
// .h file
class LightController : public QThread
{
    Q_OBJECT

public:
    int value;

protected:
    void run();
};

// .cpp file
void LightController::run()
{
    Greenhouse &greenhouse = Greenhouse::Instance();

    while( 1 )
    {
        if( greenhouse.getLight() < value && greenhouse.getLamp() < 100 )
        {
            greenhouse.setLamp( greenhouse.getLamp() + 1 );
        }
        else if( greenhouse.getLight() > value && greenhouse.getLamp() > 0 )
        {
            greenhouse.setLamp( greenhouse.getLamp() - 1 );
        }

        usleep(10000);
    }
}
```

When necessary, the thread can be started this way.

LightController `lightController;`
```
lightController.value = value;
lightController.start();
```

As can be seen, the control is extremely trivial and may require more detailed elaboration in case we receive a detailed use case.

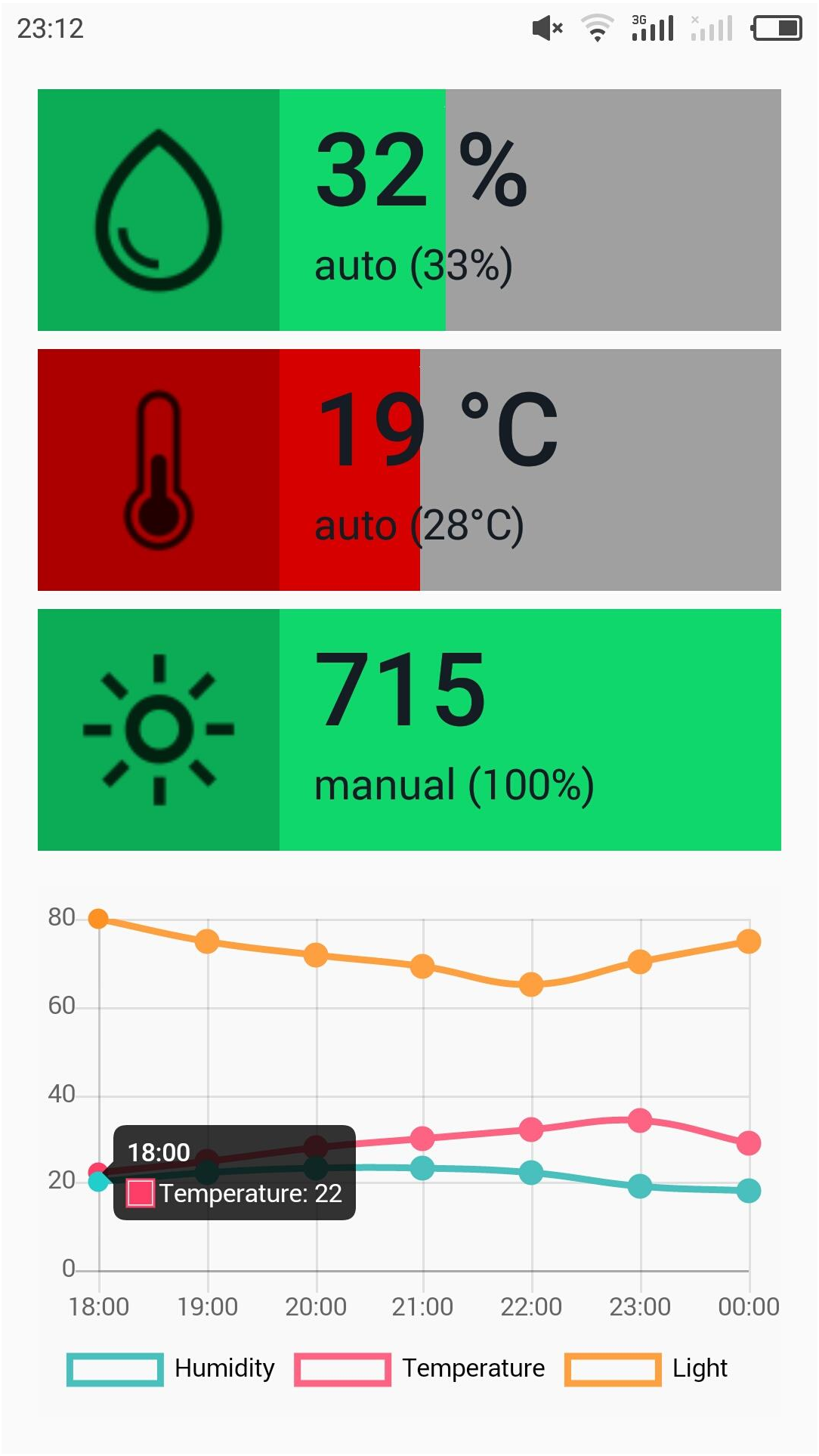
Following is the UML diagram of all the classes used. It is important to note that **Greenhouse** is a Singleton, as it won't make sense to have several of them.



Client application

Accessing the REST web server is possible via a hybrid android application, written in native android Java and partially JavaScript / HTML.
Following is a screenshot of the only screen available in the application.

Top part contains 3 sliders/buttons that show actual readings of humidity (%), temperature (degrees Celsius) and light (raw values 0 – 4095) from greenhouse sensors. Pressing the icon on the left switches between manual and automatic control modes, allowing the user to set desired values of temperature, humidity and light or manually control the heater (ON/OFF), lid (degrees) and lamp (%) manually. In case the sensor values deviate too much from the desired ones in automatic control modes, the slider is painted red.
Bottom part contains a graph with readings of temperature, humidity and light from last 6 hours. Pressing one of the labels toggles its parameter visibility, so that the user can hide the data he is currently not interested in.


23:12
32 %
auto (33%)
19 °C
auto (28°C)
715
manual (100%)
80
60
40
20
0
18:00
Temperature: 22
18:00
19:00
20:00
21:00
22:00
23:00
00:00
Humidity
Temperature
Light