Project Title

Project Documentation

Grad Laurentiu-Calin
Group 30435

Computer Science
December, 2024

Contents

1 Introduction: Purpose and Motivation

- Why did you choose this project?
- What scope do you want to fulfill?
- Is it a well-known idea that you wanted to implement for yourself? Or did you want to try something new because no existing solution satisfied you?

2 Bibliographic Research

- Find some existing solutions for your idea (or close to your idea) and compare them.
- Resume to a very short description of the solutions and compare them from different perspectives: power consumption, difficulty to implement, cost of resources and implementation, adaptation to user's needs, etc.

There are multiple implementation and design project ideas related to greenhouses, temperature or humidity monitoring in different media that are available on-line, some interesting ones that I have encountered are: [1], [2], [3]. However, I chose to implement this in my own way, without taking inspiration from any of these. Consequently, my research focused on searching, finding, purchasing and working with sensors, development boards, electrical circuits and diagrams.

A wide variety of temperature or temperature & humidity sensors is available on the market. I set on three of them:

- HR202 $\rightarrow$ humidity sensor.
- DHT11 $\rightarrow$ temperature sensor.
- DHT22 $\rightarrow$ temperature & humidity sensor.

The core differences between the above mentioned sensors are summarized in the table below:

Feature	HR202	DHT22	DHT11
Type	Resistive Humidity Sensor	DIgital Humidity and Temperature Sensor	Digital Humidity and Temperature Sensor
Humidity Range	20% to 95% RH	0% to 100% RH	5% to 95% RH
Temperature Range	0°C to 60°C	-40°C to 80°C	-20°C to 60°C
Humidity Accuracy	±15% RH	±2% to ±5% RH	±5% RH
Temperature Accuracy	*Undefined*	±0.5°C	±2°C
Output	Digital	Digital	Digital
Power Supply	3.3V - 5V DC	3V - 6V DC	3V - 5.5V DC
Communication	Analog	One-wire	One-wire
Response time	Not Specified	2 seconds	2 seconds
Common applications	Weather monitoring, Dew detection	Weather stations, Greenhouse monitoring	Home weather stations, Environmental monitoring

Table 1: Comparison of **HR202**, **DHT22**, and **DHT11** sensors

3 Proposed Solution and Implementation

- Overall description of the solution chosen.
- Theoretical description of your algorithm(s).
- Implementation:
  - Hardware: draw a circuit of your hardware parts and their connections.
  - Software: describe the relevant functions and how they work together.

I have chosen to implement my project in a bottom-up manner, meaning that I started with small parts that had nothing to do with each other at first glance. Once all these modules had been designed, implemented and tested, I moved to creating the inter-component logic. The main advantage my approach offered was to let me think in object-oriented style. Thus, each component (printed circuit boards, simple LED-s, temperature or humidity sensors) was wrapped inside an abstract object (class). Being a relatively complex project thanks to the amount of components that have different behaviors, the development process was accelerated because I had access to the high-level abstraction mechanisms provided by C++. By following the *SOLID* principles and some *good practice* industry rules, the final form of the source code appears to me as maintainable, clear to use and improve, and most importantly free of bugs, useless code and verbosity.

3.1 The DHT22 Sensor

DHT22 is a temperature and humidity sensor. It is a really popular measuring device because it has a low production cost and is easy to use. Thanks to this popularity a lot of open-source libraries that implement high-level operations already exist: *SimpleDHT, Adafruit Unified Library, etc.*

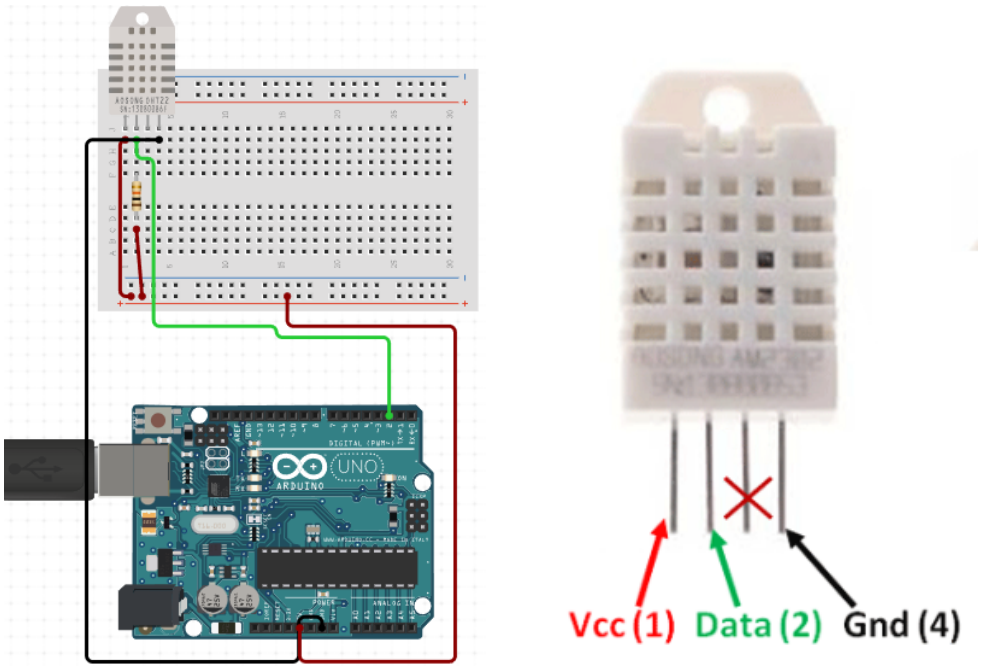

Figure 1: The DHT22 sensor Arduino diagram and a close-up of the component

```cpp
#include <stdint.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

class DHT22Sensor {
    public:
        DHT22Sensor() : dht(12, DHT22) {};
        void setupSensor();
        void sensorLoop();
    private:
        // the pin on the Arduino board that is
        // connected to the data pin of the DHT22
        const int dataPin = 12;
        // sensor wrapper provided by
        // Adafruit_sensor.h
        DHT_Unified dht;
        // how many miliseconds we have to wait
        //to read data from the sensor again
        uint32_t delayMS;
};
```

Listing 1: C++ class definition of the DHT22 sensor

3.2 RGB led

A simple common cathode RGB led.

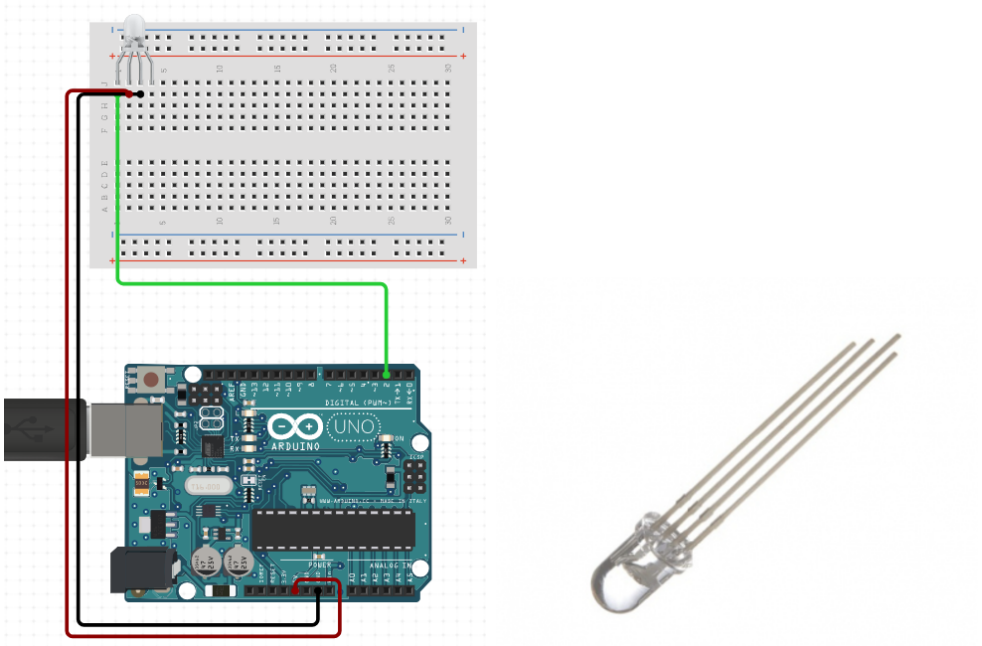

Figure 2: The RGB led Arduino diagram and a close-up of the component

```cpp
#include "Types.hpp"
#include "Arduino.h"

class LED
{
    public:
        explicit LED(Color newColor) : color(newColor) {};
        void setupLed() const;
        int getRedPin() const { return redPin; };
        int getGreenPin() const { return greenPin; };
        int getBluePin() const { return bluePin; };
        Color getColor() const { return color; };
        void setColor(Color newColor) { color = newColor; };
        void displayColor() const;
    private:
        const int redPin{11};   // PWM pin
        const int greenPin{10}; // PWM pin
        const int bluePin{9};   // PWM pin
        Color color;
};
```

Listing 2: C++ class definition of the RGB LED module

3.3 8-LED module

This is a fairly basic printed circuit board that houses 8 LED-s in common anode configuration.

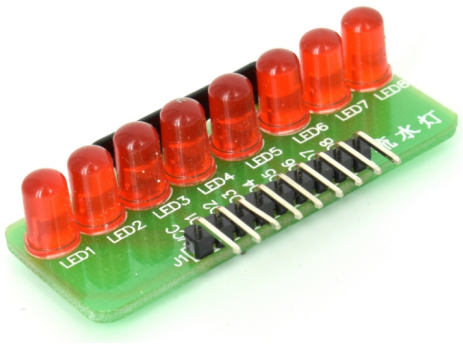

Figure 3: The led module printed circuit board

```
#include "Arduino.h"

class LedModule
{
    public:
        LedModule() {};
        void setupLedModule() const;
        int getLedPin(int index) const;
        void turnOnLed(int index) const;
        void turnOffLed(int index) const;
    private:
        // the 8 pins of the Arduino board that are connected
        // to the 8 individual cathodes of the led module
        const int leds[8]{1, 2, 3, 4, 5, 6, 7, 8};
        const int numberOfLeds{8};
};
```

Listing 3: C++ class definition of the LED module

3.4 Buzzer

Another basic component that emits sounds with respect to a channel that supports PWM.

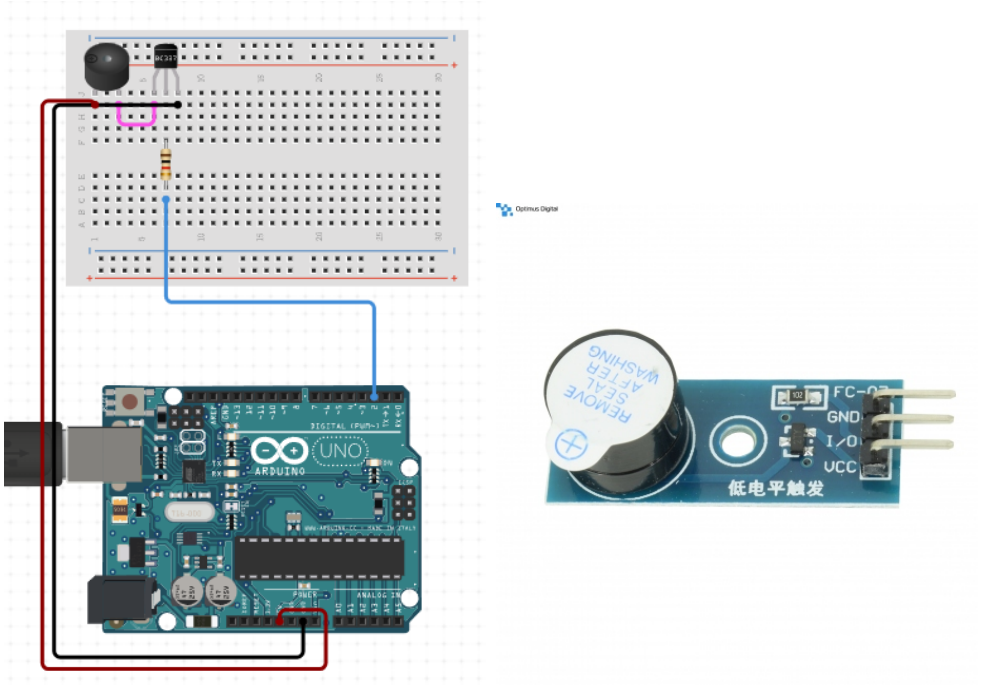

Figure 4: The diagram connection of a simple buzzer and a close-up of the component[1]

```cpp
#include "Arduino.h"
#include "Pitches.hpp"
#include "Types.hpp"

class Buzzer
{
    public:
        Buzzer() {};
        void setupBuzzer() const;
        void turnOn(int noteFrequency, Notes duration);
        void turnOff();
        void buzzerLoop();
    private:
        // PWM Arduino Pin that is connected to the data pin
        // of the buzzer
        const int dataPin{3};
        const float pauseStretch{1.3f};
};
```

[1] The PCB I use has an integrated transistor and resistance meaning that interfacing with the board is much simpler: 2 power connections (GND and VCC) and a data wire.

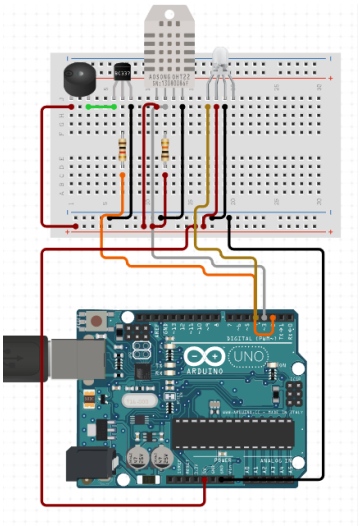

Figure 5: Arduino diagram for different phases of the project

4 Testing and Validation

- What problems did you encounter while implementing your solution?
- Make tables to compare your project's behavior when you changed some aspects in the environment.
- Describe the phases your project went through.

5 Conclusion

- Was your purpose fulfilled?
- How can your solution help others?
- How can it practically be improved?

References

[1] aleixj. Automated greenhouse with arduino. `https://projecthub.arduino.cc/aleixj/automated-greenhouse-with-arduino-974338`, December 2020. Accessed: 2024-11-20.

[2] ejshea. Displaying temperature and humidity on an lcd. `https://projecthub.arduino.cc/ejshea/displaying-temperature-and-humidity-on-an-lcd-91bc36`, July 2019. Accessed: 2024-11-20.

[3] Sgarro and christian26. Greenhouse. `https://projecthub.arduino.cc/Sgarro/greenhouse-f6fc53`, December 2019. Accessed: 2024-11-20.