ΕΞΩΦΥΛΟ

Acknowledgements

## Abstract

Contents

[Abstract 2](#_Toc394964125)

[1. Introduction 6](#_Toc394964126)

[1.1. Motivation 6](#_Toc394964127)

[1.2. Purposes and targets 6](#_Toc394964128)

[1.3. Project Structure 6](#_Toc394964129)

[2. Implementation Methodology 6](#_Toc394964130)

[3. Project Plan 6](#_Toc394964131)

[3.1. State Of The Art 7](#_Toc394964132)

[4. Basic Part 8](#_Toc394964133)

[4.1. Hardware Components 8](#_Toc394964134)

[4.1.1. Arduino 8](#_Toc394964135)

[4.1.2. Sensors 11](#_Toc394964136)

[4.1.3. Prototype Architecture 21](#_Toc394964137)

[4.2. Software Analysis 22](#_Toc394964138)

[4.2.1. Arduino Sketch [23] 22](#_Toc394964139)

[4.2.2. Server – Side PHP Scripts 34](#_Toc394964140)

[References 34](#_Toc394964141)

Nomenclature

* MC / MCU Microcontroller
* RH Relative Humidity
* °C Degrees Celsius
* EMC Electromagnetic Compatibility
* URL Uniform Resource Locator
* URI Uniform Resource Identifier

[Figure 1 - The Arduino Mega 2560 R3 Microcontroller Board 9](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967939)

[Figure 2 - The Arduino Ethernet Shield 10](#_Toc394967940)

[Figure 3 - RHT03 11](#_Toc394967941)

[Figure 4 - RHT03 Communication Process 13](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967942)

[Figure 5 - BMP085 Sensor 13](#_Toc394967943)

[Figure 6 - BMP085 Application Circuit 14](#_Toc394967944)

[Figure 7 - Calculation Process of Pressure and Temperature for BMP085 15](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967945)

[Figure 8 - BMP085 Temperature and Pressure Reading Process 15](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967946)

[Figure 9 - Bouncing problem at a digital input. MCU will read nine state changes instead of one 17](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967947)

[Figure 10 – Rain Reading And Switch Debouncing Implementation 17](#_Toc394967948)

[Figure 11 - Connection of the Rain Gauge to the Arduino Mega 18](#_Toc394967949)

[Figure 12 - Anemometer Reading And Switch Debouncing Implementation 18](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967950)

[Figure 13 - Connection of the Anemometer to the Arduino Mega 19](#_Toc394967951)

[Figure 14 - Wind Vane resistor arrangement 19](#_Toc394967952)

[Figure 15 - Wind Vane Powering Up 20](#_Toc394967953)

[Figure 16 - Connection of the Wind Vane to the Arduino Mega 21](#_Toc394967954)

[Figure 17 - Schematic wiring of all components 22](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967955)

[Figure 18 - Function to read 1 byte from the bmp085 24](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967956)

[Figure 19 - Function to read 2 bytes from the bmp085 24](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967957)

[Figure 20 - BMP085 Calibration 24](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967958)

[Figure 21 - BMP085 Read Uncompensated Pressure Function 25](#_Toc394967959)

[Figure 22 - BMP085 Read Uncompensated Temperature Function 25](#_Toc394967960)

[Figure 23 - BMP085 Final Temperature Calculations 26](#_Toc394967961)

[Figure 24 - BMP085 Final Pressure Calculations 27](#_Toc394967962)

[Figure 25 - Weather Meters Configuration 28](#_Toc394967963)

[Figure 26 - Rain Final Calculation 28](#_Toc394967964)

[Figure 27 - Wind Final Calculation 28](#_Toc394967965)

[Figure 28 - URL contruction 30](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967966)

[Figure 29 - HTTP Request Execution 31](#_Toc394967967)

[Figure 30 - HTTP Response 32](#_Toc394967968)

[Figure 31 - Remote Camera Control 32](#_Toc394967969)

[Figure 32 - Generated HTTP Request on button click 33](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967970)

[Figure 33 33](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967971)

[Figure 34 - Servo Control Example 33](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394967972)

[Figure 35 - Script to store data to database 34](#_Toc394967973)

[Table 1 - RHT03 Technical Specifications 12](#_Toc394960346)

[Table 2 - RHT03 Pin Sequence Number (1, 2, 3, 4 from left to right) 12](#_Toc394960347)

[Table 3 - Main Properties of the BMP085 13](#_Toc394960348)

[Table 4 - Measurement modes of the BMP085 and resolution effects 16](#_Toc394960349)

[Table 5 - BMP085 Pin Connections to Arduino Mega 16](#_Toc394960350)

[Table 6 - Analog Voltage output with corresponding direction 20](#_Toc394960351)

[Table 7 - BMP085 Calibration Coefficients 24](file:///D:\Dropbox\GraduationalProject\SenseBox\Report2%20(Autosaved).docx#_Toc394960352)

# Introduction

The purpose of this project is to measure the weather and virtualize the data. The data gathering process is implemented upon Arduino, an open source hardware prototyping platform, which allows an easy implementation of sensors and interactive elements. Data gathered by Arduino are stored on a database and on the next level these data are used to make graphs on different platforms. Also the project included a live weather camera that rotates to spot almost everywhere the user wants.

## Motivation

Building my own weather station was an idea that came into my mind about two years ago. I first thought of building it for educational purposes and as a hobby. Later on I realized that the project could also serve as my final year’s project so I declared it.  
  
My interested on weather data has been inherited from my favorite sport – Sailing. For sailors observing the weather is a major part of their daily routine so this was the icing on the cake to begin!

## Purposes and targets

Sdasdasd

## Project Structure

The project is split in four parts.

1. The Arduino part in which data are gathered.
2. The server part that hosts the server-side PHP code that is used to store and retrieve data from the database.
3. The Android Application part that visualizes the data.
4. The Web Site part that also visualizes the data.

# Implementation Methodology

Cvsdfsdf

# Project Plan

The project has been implemented using the latest technologies available for its needs. The next chapter state of the art provides a more in depth analysis about the background of the project.

## State Of The Art

* **Open – Source Hardware**

Open - Source has become a popular expression, but mostly with regard to software. The principles and definitions of Open-Source Hardware (OSHW) are closely related to those of Open-Source Software (OSS) from the Open Source Initiative [1, 2]. Interested people can study, modify, distribute, make and sell designs based on those of OSHW products [3]. Through open development, people get the possibility to learn and understand how OSHW works, so that they are able to control and modify their technology. Machines, devices, or other physical things produced under the OSHW license must comply with the definitions of the Open Source Hardware Association [1].

* **Arduino Platform**

Arduino is a prototyping platform containing a microcontroller board (MC) as core element, a programming language and an integrated development environment (IDE) [4]. OSHW MCs from Arduino are based on ATmega8, ATmega168, ATmega2560 and latest boards on [ATmega32u4](http://arduino.cc/en/Main/ArduinoBoardYun), [Sitara AM335x](http://arduino.cc/en/Main/ArduinoBoardTre). They can be used to develop interactive prototypes, which are using input from sensors to control output devices connected to the same board. The capabilities of the MCs allow several methods for in- and outputs of signals.

The simplified IDE with a wire-based programming language also allows beginners to realize complex projects in short time. Documentation of the IDE and the language reference can be found on the Arduino homepage [5]. All code samples are released into the public domain. Additionally, the language can be extended with C++ libraries to enable more functionality. All official software tools from Arduino are published under the OSS license and are platform-independent.

The well documented Arduino hard- and software makes it even possible to rebuild MCs by oneself. However, pre-assembled Arduino boards are relatively inexpensive compared to other commercial MC platforms available on the market

* **Raspberry PI**

The Raspberry PI is a credit – card – sized single board computer developed mainly for educational purposes. As core element it has a Broadcom BCM2835 system on a chip (SoC) [9] which includes an ARM11 family 700MHz processor, a VideoCore IV GPU and 512MB RAM.

The Raspberry PI foundation provides a lightweight version of Debian (Raspbian) and Arch Linux ARM distribution as official operating systems.

Tools are available for Python as the main programming language, C, Java and Perl.

For the needs of the project, the RPI is used as server, hosting the script files that are used to store and retrieve weather data in the database as well as hosting the web page. Later it is explained why the RPI is the perfect small server.

* **Automated Weather Stations**

Automated weather stations (AWS) are meteorological stations, which perform climatic observations and data transmission automatically [6]. Main advantages compared to human weather observations are, that measurements, read out by a central data-acquisition unit, are more reliable and can be performed much more frequently. Sensors of an AWS are operated by a microprocessor. It allows exact sampling of sensor data and processing for averaging or filtering of the samples. The result will be a series of observations which are representative over a limited area.

* **Android Platform**

Android is a mobile operating system based on the Linux Kernel that is currently developed by Google. It powers hundreds of millions of mobile devices around the world. It is the largest installed base of any mobile operating system platform.

Android’s source code is released by Google under open source licenses (Apache License 2.0 and GNU General Public License 2.0 [7, 8]).

Its open nature has encouraged a large community of developers and enthusiasts to use the open – source code as a foundation for community-driven projects, which adds new features for advanced users.

* **Google Chart API**

The Google Chart API [10] is a tool that lets people easily create a chart from some data and embed it in a web page. Google creates a [PNG](http://en.wikipedia.org/wiki/Portable_Network_Graphics) image of a chart from data and formatting parameters in an [HTTP](http://en.wikipedia.org/wiki/HTTP) request. Many types of charts are supported, and by making the request into an image tag, people can simply include the chart in a web page.

# Basic Part

//TODO REQUIREMENT ANALYSIS

## Hardware Components

The Microcontroller controls the data storage process and sampling frequency. Sampling rates of sensors are depending on the time constant value that we set. Sensors and Microcontroller functionality are explained in depth in the upcoming section.

### Arduino

The Arduino and the Ethernet Shield can be imagined as the ‘brain’ of the weather station. It provides all the core functionality from sensor manipulation up to data logging. The Arduino Mega and the Arduino Ethernet Shield are introduced below.

#### Arduino Mega 2560 R3

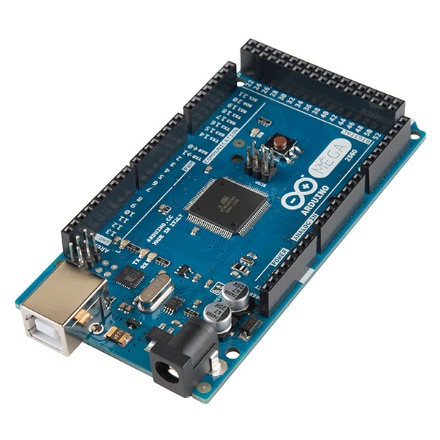


Figure 1 - The Arduino Mega 2560 R3 Microcontroller Board

Arduino Mega 2560 [11] is the board that is powered by ATmega2560. Its main advantages compared to other Arduino boards are the clock speed of 16MHz, the flash memory of 256Kbytes of the ATmega2560 chip and the 8Kbytes SRAM. The large flash and SRAM memories offer the Arduino vast of space for more interactive components to be added in future improvements.

The Arduino Mega provides 54 digital pins. They can be used as input or output and are operating at 5V. Each pin can provide or receive a maximum of 40mA and has an internal pull-up resistor of 20 – 50 kΩ. The pull-up resistor is disconnected by default but they can be accessed from software by setting the **pinMode()** as **INPUT\_PULLUP**. This effectively inverts the behavior of the INPUT mode, where HIGH means the sensor is off, and LOW means the sensor is on. In addition, some pins have specialized functions:

* **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).**

Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

* **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).**

These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

* **PWM: 2 to 13 and 44 to 46.**

Provide 8-bit PWM output with the [analogWrite()](http://arduino.cc/en/Reference/AnalogWrite) function.

* **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).**

These pins support SPI communication using the SPI library.

* **LED: 13.**

There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

* **TWI: 20 (SDA) and 21 (SCL).**

Support TWI communication using the [Wire library](http://arduino.cc/en/Reference/Wire).

(BMP085 sensor uses the Wire protocol to communicate with Arduino)

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 210 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

#### Arduino Ethernet Shield

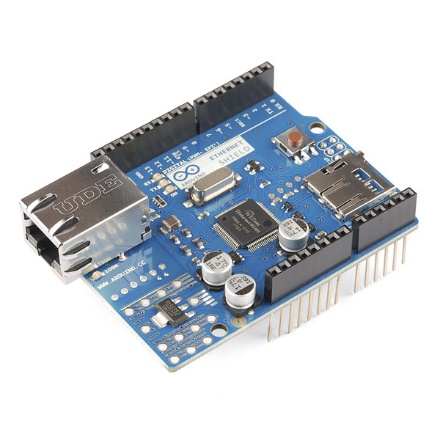


Figure 2 - The Arduino Ethernet Shield

The Ethernet shield [12] is powered by a Wiznet W5100 ethernet controller and also includes a micro SD card slot for storage purposes. It is made exclusively for Arduino boards and connects over the SPI port.

The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. The ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The Ethernet Shield has a standard RJ-45 connection, with an integrated line transformer and Power over Ethernet enabled.

It worth mentioning that pins SPI pins cannot be used as digital in or output any more when using the Ethernet Shield as they are being used by the SPI bus. Pins 4, 10 are used for the SD card and cannot be used as well. All the other pins are free to be used.

The Ethernet Shield is used in the project for data logging and as server for controlling the pan-tilt camera mount.

To integrate functionalities of the Ethernet Shield into the whole system, Ethernet [13] and SD [14] libraries are used.

### Sensors

In this section the functionality and the characteristics of each sensors used by the prototype will be analyzed. In addition, connection and communication examples are given in this section.

#### Relative Humidity and Air Temperature (DHT22 or RHT03)

The RHT03 sensor from is made of two parts, a capacitive humidity sensor and a thermistor. There is also a basic chip inside that does some analog to digital conversion and offers a digital signal with the temperature and humidity.

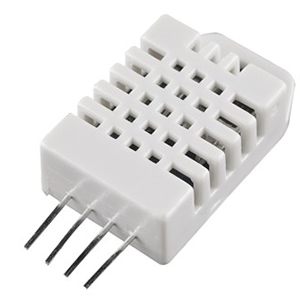


Figure 3 - RHT03

The RHT03 was chosen because of its high Resolution and Accuracy. Technical Specifications are shown in the table below [15].

|  |  |
| --- | --- |
| **Model** | **DHT 22 also known as RHT03** |
| Power Supply | 3.3 – 6V DC |
| Output Signal | Digital Signal via MaxDetect 1-wire bus |
| Sensing Element | Polymer Humidity Capacitor |
| Operating Range | Humidity: 0 – 100% RH  Temperature: - 40 ~ 80 °C |
| Accuracy | Humidity: +- 2% RH (Max +-5% RH)  Temperature: +- 0.5 °C |
| Resolution or Sensitivity | Humidity: 0.1% RH  Temperature: 0.1 °C |
| Repeatability | Humidity: +- 1% RH  Temperature: +- 0.2 °C |

Table 1 - RHT03 Technical Specifications

|  |  |
| --- | --- |
| Pin | Function |
| 1 | Power Supply |
| 2 | Data – Signal |
| 3 | NULL |
| 4 | Ground |

Table 2 - RHT03 Pin Sequence Number (1, 2, 3, 4 from left to right)

##### Communication and signal

Wire Bus is used for communication between the MCU and RHTY03.

**Illustration of MaxDetect’s wire bus:**

Data is comprised of integral and decimal part, the following is the formula for data.

* 8bit integral Relative Humidity data
* 8bit decimal Relative Humidity data
* 8bit integral Temperature data
* 8bit decimal Temperature data
* 8but checksum

When MCU send start signal, RHT03 change from standby-status to running-status. When MCU finishs sending the start signal, RHT03 will send response signal of 40-bit data that reflect the relative humidity and temperature to MCU. Without start signal from MCU, RHT03 will not give response signal to MCU. One start signal for one response data from RHT03 that reflect the relative humidity and temperature. RHT03 will change to standby status when data collecting finished if it don't receive start signal from MCU again.

Below one can find the overall communication process. The interval of whole process must beyond two seconds.

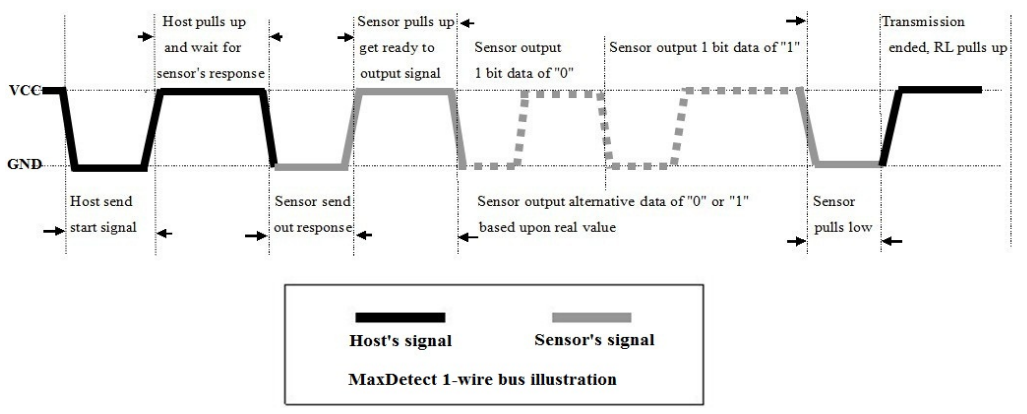


Figure - RHT03 Communication Process

#### Air Pressure and Temperature (BMP085)

For air pressure readings and temperature the digital sensor BMP085 from Bosch Sensortec is used [16]. It was chosen for the prototype because of its good accuracy through individual precise calibration and its long term stability feature. The main characteristics are summarized in the table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Condition** | **Min** | **Typical** | **Max** | **Unit** |
| Resolution | Typical | 0.04 | 0.01 | 0.01 | hPA |
| Accuracy | Typical |  | +- 1.0 |  | hPA |
| Response Time | Mode | 4.5 |  | 25.5 | Ms |
| Operation Range |  | 700 |  | 1100 | hPA |
| Supply Voltage |  | 1.8 | 2.5 | 3.6 | Volts |

Table 3 - Main Properties of the BMP085

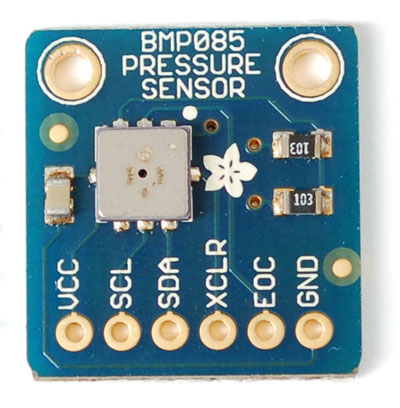


Figure 6 - BMP085

Figure 5 - BMP085 Sensor

It is based on piezo - resistive

Technology for EMC robustness, high accuracy and linearity as well as long term stability. It is designed to be connected directly to a MCU via the I2C bus. The pressure and temperature data has to be compensated by the calibration data of the E2PROM of the BMP085.

The sensor consists of a piezo – resistive sensor, an analog to digital converter and a control unit with E2PROM and a serial I2C interface. It delivers the uncompensated value of pressure and temperature. The E2PROM has stored 176 bit of individual calibration data. This is used to compensate offset, temperature dependence and other parameters of the sensor.

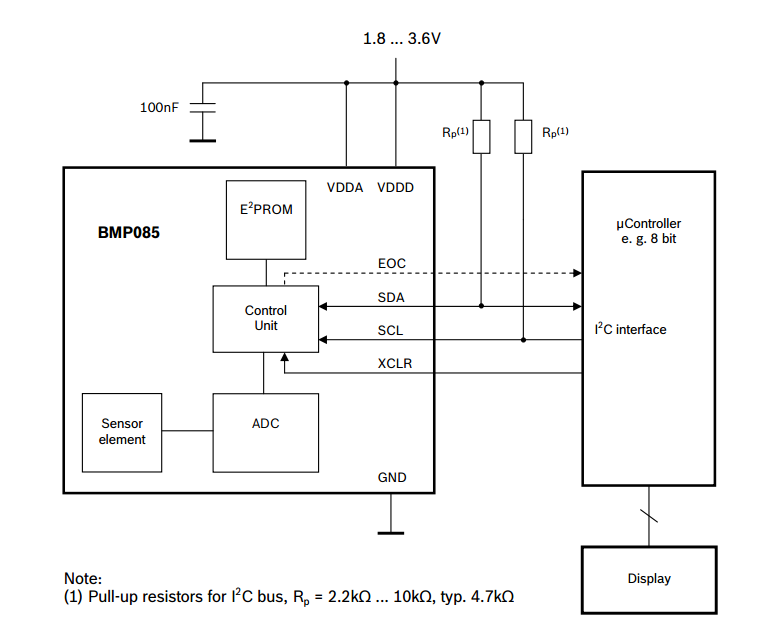


Figure 6 - BMP085 Application Circuit

The I²C interface and a read-only-memory register (E²PROM) are part of the control unit of the BMP085. In the E²PROM eleven 16bit calibration coefficients are stored and used to compensate offset of temperature and pressure readings. As temperature is a factor for air pressure calculation, it has to be known to calculate the true pressure. Reading of the temperature can be done with the piezo-resistive sensor as well.

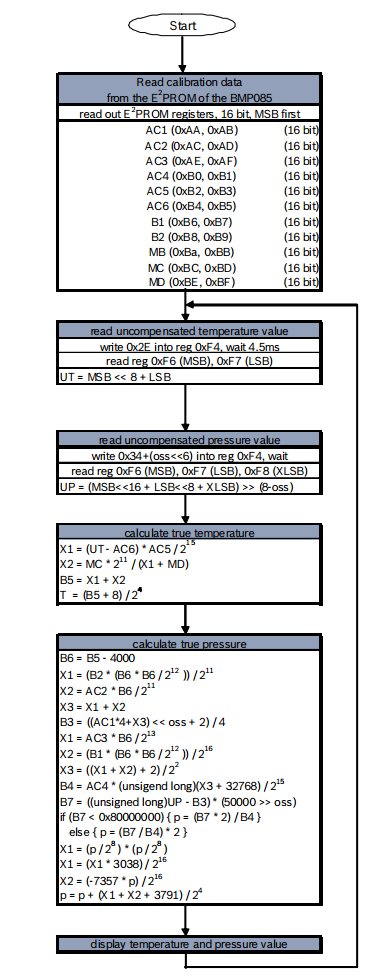


Figure 7 - Calculation Process of Pressure and Temperature for BMP085

A measurement sequence with the complete algorithm is shown in Figure 9. It has to be implemented to the MCU according to this order.

After the MC has sent a start sequence, the calibration data is requested from the E²PROM registers. The 16 constants need to be read out only once and are stored on the MCU. Then the MCU has to wait for uncompensated temperature (UT) and pressure (UP) readings from the sensors. The stored calibration data is now used to calculate temperature in °C and pressure in Pa. After applying the algorithms, the sensor waits for the next measurement command.

As additional factor for true pressure calculation, an oversampling mode can be chosen to set the internal sampling of the sensor for one measurement. Changing the mode to a higher resolution will increase the accuracy of a measurement but also has an impact on energy consumption, reaction time and the long term stability [17].

Table 4 gives an overview about the four oversampling setting modes. As the weather station is designed for long and continuous measurements, standard mode will be sufficient.

Figure 8 - BMP085 Temperature and Pressure Reading Process

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mode** | **Oversamping Setting** | **Internal Samples** | **Conversion Time (ms)** | **Noise (hPA)** |
| Ultra Low Power | 0 | 1 | 4.5 | 0.0.6 |
| Standard | 1 | 2 | 7.5 | 0.05 |
| High Resolution | 2 | 4 | 13.5 | 0.0.4 |
| Ultra High Resolution | 3 | 8 | 25.5 | 0.03 |

Table 4 - Measurement modes of the BMP085 and resolution effects

| **BMP085** | **Arduino Mega Pin** |
| --- | --- |
| SDA | 20 |
| SCL | 21 |
| XCLR | Not Connected |
| EOC | Not Connected |
| GND | GND |
| VCC | 3.3V |

As mentioned before, the BMP085 sensor is using the I2C protocol to communicate with the MCU and has to be connected to SDA and SCL pins of the Arduino Mega as shown below.

Table 5 - BMP085 Pin Connections to Arduino Mega

The altitude of the instrument has to be taken into account as well, because air pressure decreases with increasing height. For getting comparable values it has to be calculated to sea level pressure [6].

Working with the BMP085 should be done carefully, because it could be damaged by shocks or when getting in contact with water. It should be handled as Electrostatic Sensitive Device (ESD) [18] too. Therefore, it should be installed in a dry environment with constant temperature.

#### Precipitation

Rainfall is measured by a Tipping Bucket Rain Gauge included in the Weather Meters Kit [19] available from Sparkfun. Each time the self – emptying tipping bucket is emptied, a reed switch closes once. This corresponds to 0.2794 mm of rain fall [20]. The state change is caught by the MCU using Arduino’s Interrupt input.

A problem that frequently appears at trying to capture the state change of a switch is called bouncing problem. State change means that the input at a digital pin on the Arduino changes from high to low (equals from 5V to 0V) or vice versa. The problem is that each switching contains interferences in the signal before it reaches the level of 5V or 0V. This can lead to multiple counts from only one state switch. A visualization of the problem is shown in Figure 5.

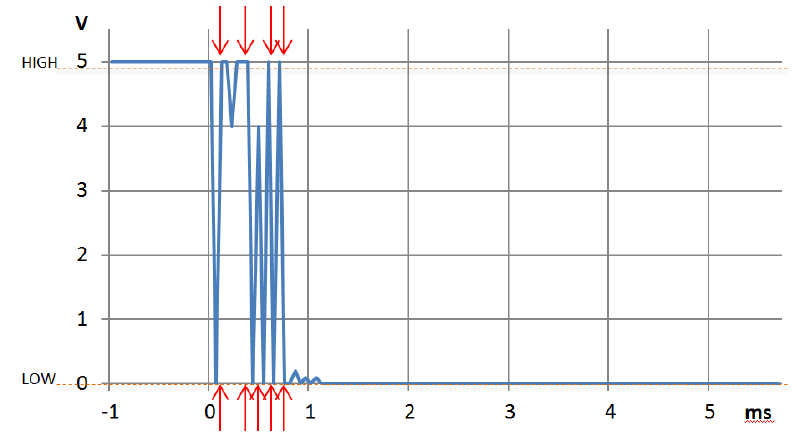


Figure 9 - Bouncing problem at a digital input. MCU will read nine state changes instead of one

To overcome this problem there are two solutions. The first is to integrate a capacitor to the circuit and the second is to define a software method with a debounce interval of some microseconds (μs), which is called very time a state change is recognized.

For the project the second method is used. The implementation is show below.

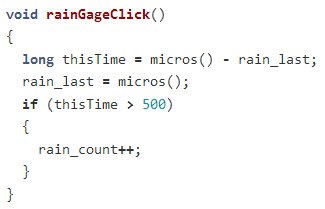


Figure 10 – Rain Reading And Switch Debouncing Implementation

The code above will ignore any interrupt that occurs within 500μs (0.0005s) of the previous one. This technique will limit the max rain this configuration can measure to a very high value that do not corresponds to an earthly value.

##### Wiring the Rain Gauge

The two wires coming from the rain gauge, through the RJ11 cable, have to be connected to the Arduino Mega as shown in the figure below.



Figure 11 - Connection of the Rain Gauge to the Arduino Mega

The red line must be connected to Ground while the green line is pulled up with a 10kΩ resistor to 5V. The output of the sensor has to be connected to digital pin 19 on the Arduino Mega.

A disadvantage of this method is that only rain fall can be measured. To include snow fall in the precipitation measurement a self heated device would be needed. Temperatures below 0° Celsius could also lock up the anemometer. This should be taken into account when analyzing the data of the rain gauge.

#### Wind Speed

The wind speed is measured with a cup – type anemometer included in Sparkfun’s Weather Meters Kit [19]. In principle, it works exactly like the rain gauge. At every turn, it closes a contact as a magnet moves past a switch, which is connected to a digital counter. 0.67 meter per second or 1.29651 knots causes the switch to close once per second [20].

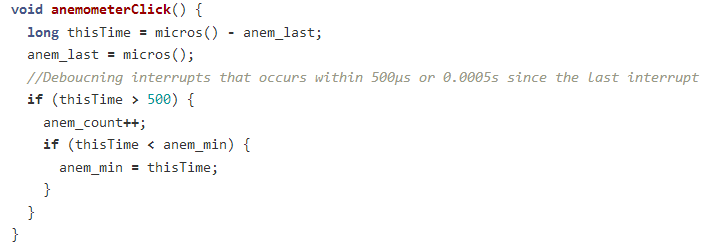
Switch debouncing is implemented for the anemometer exactly the same way as the rain gauge. The max wind speed this configuration can measure is limited to about 2,700 knots that does not affect the system since that kind of values are not earthly.

Figure 12 - Anemometer Reading And Switch Debouncing Implementation

##### Wiring the Anemometer

The anemometer is connected with a two wired cable to the wind vane. The two center wires (yellow and red) of the cable coming from the wind vane are used by the anemometer. Again, the red cable needs a pull-up resistor before connecting it to 5V. The yellow cable is connected to GND.



Figure 13 - Connection of the Anemometer to the Arduino Mega

#### Wind Direction

The wind vane of the Weather Sensor Assembly [19] contains eight switches, each of them connected to a resistor with different resistance values. A magnet switch can close two switches at once which allows to indicate 16 different positions. The voltage of the output cable can be measured at an analog input on the Arduino Mega. Directions for corresponding voltage output can be found

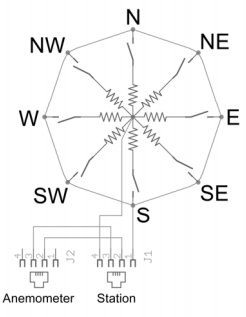
In table 6.

Figure 14 - Wind Vane resistor arrangement

|  |  |  |
| --- | --- | --- |
| **Direction (Compass)** | **Direction (Degrees)** | **Voltage Output** |
| N | 0 / 360 | 3.84 |
| N - NE | 22.5 | 1.98 |
| NE | 45 | 2.25 |
| E - NE | 67.5 | 0.41 |
| E | 90 | 0.45 |
| E – SE | 112.5 | 0.32 |
| SE | 135 | 0.90 |
| S – SE | 157.5 | 0.62 |
| S | 180 | 1.40 |
| S – SW | 202.5 | 1.19 |
| SW | 225 | 3.08 |
| W – SW | 247.5 | 2.93 |
| W | 270 | 4.62 |
| W – NW | 292.5 | 4.04 |
| NE | 315 | 4.78 (4.33) |
| N – NW | 337.5 | 3.43 |

Table 6 - Analog Voltage output with corresponding direction

Testing of the wind vane showed that voltage values at 315o is not 4.78V as mentioned at the datasheet. A 10kΩ resistor in series with a 64.9kΩ resistor, the voltage drop across the vane will be 4.33V.

For better power management, the wind vane is powered by the digital pin VANE\_PWR right before it is used. A 100ms delay is inserted after the power pin is switched to HIGH just to let things settle down.

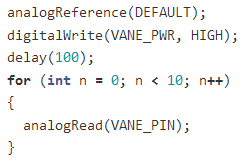


Figure 15 - Wind Vane Powering Up

##### Wiring the Wind Vane

The green and black outer wires of the cable from the wind vane have to be connected to the Arduino Mega as shown in Figure 8.

The wind vane and the anemometer shares the same RJ11 cable.



Figure 16 - Connection of the Wind Vane to the Arduino Mega

As mentioned before, the incoming voltage can be measured with an analog pin on the Arduino Mega. The values, which are read out at the analog input, have to be converted from 10 bit (0-1023) to a scale between 0 and 5V (0 to 5000). The resulting voltage can then be used to grab the corresponding value from a list according to Table 6.

### Prototype Architecture

//TODO: Physical setup description (mounts, containers etc.)

A schematic overview, created on Fritzing [21], of the wiring is show in figure 17.

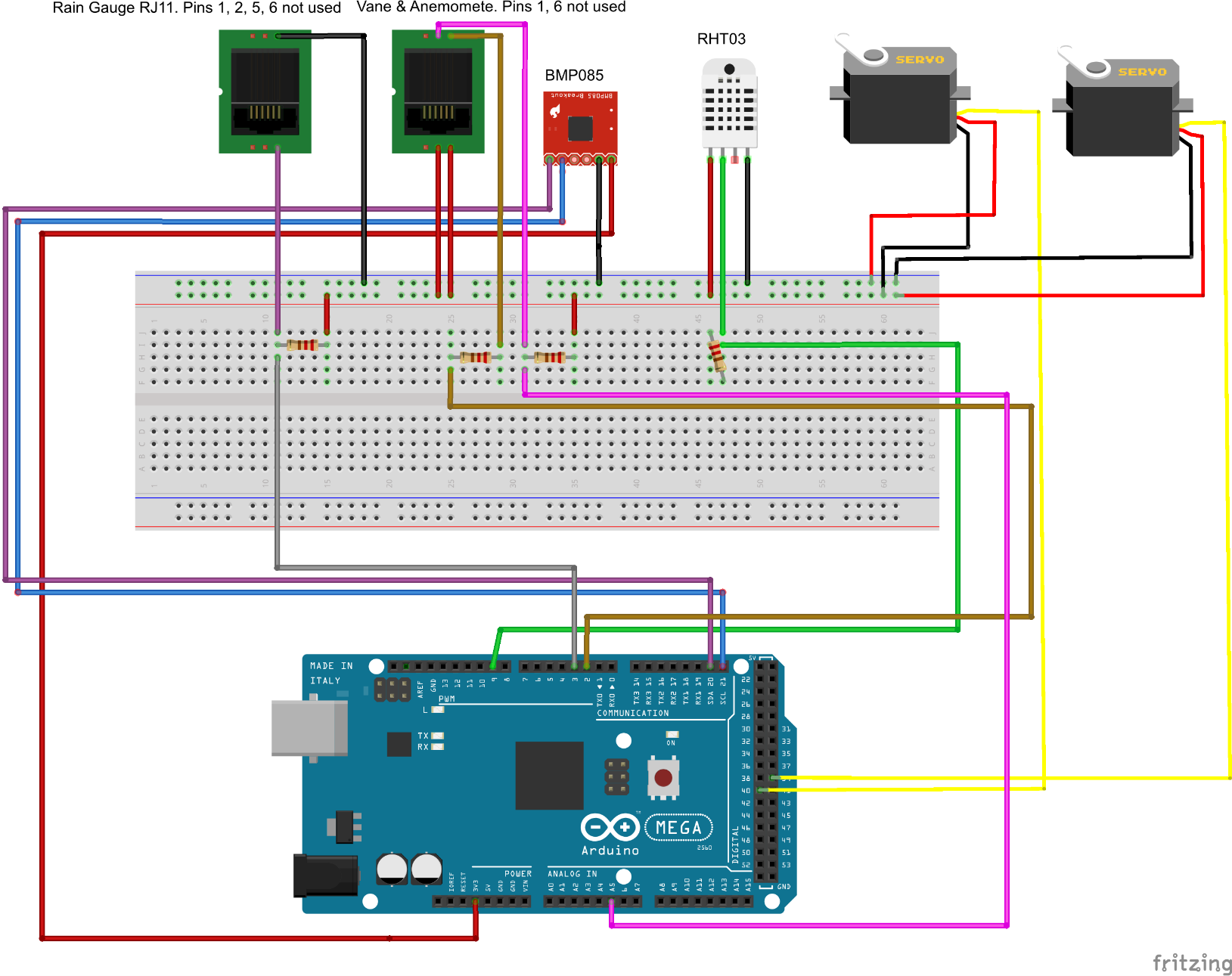


Figure 17 - Schematic wiring of all components

## In Depth Project Analysis

Following the Hardware components review, the Software analysis will dive deep into all the code that was used for the project to work.

This section will be split into four parts:

* Arduino Sketch
* Server – Side PHP scripts
* Android Application
* Web Site

### [Arduino Sketch](https://github.com/mKontakis/Sensebox/blob/master/WeatherMeters/ver11/ver11.ino) [23]

//TODO: include watchdog timer

The Arduino Sketch is the most basic part of the project. It is the beginning. If this does not work perfectly nothing else can be done.

The Sketch consists of three major parts.

* The sensors values readings
* The HTTP Requests
* The Pan – Tilt camera mount control

#### Sensors Values Readings

Each sensor has its own style for reading its value.

##### RHT03

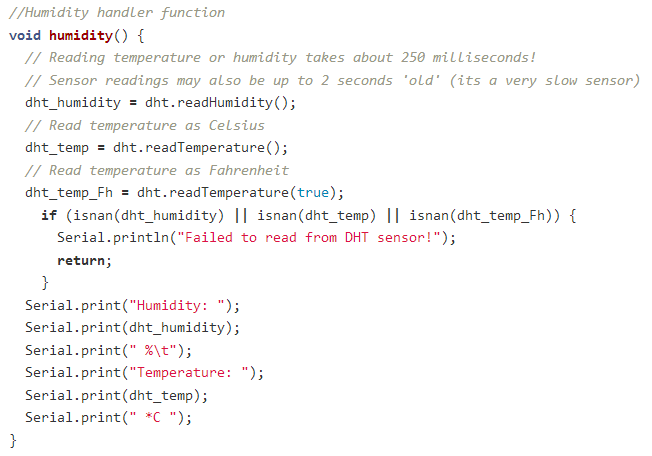
To integrate RHT03’s functionality to the project the library written by Adafruit Industries [22] is used.

First of all, the library must be imported and a DHT object is created.

#include <DHT.h>

DHT **dht**(DHTPIN, DHTTYPE);

To get RHT03’s readings the library does the heavy job.



The code above is very simple, since DHT library undertakes to deal with all the heavy job. At the beginning, temperature and humidity values are stored at the global variables ***dht\_humidity, dht\_temp***. Then an if else statement checks if the values read are numbers by calling **isnan** function (Is Not A Number). This check is being done to ensure that everything works well in the sensor.

##### BMP035

Collecting BMP035’s values was a challenging task, since the algorithm described in Figure 9 had to be implemented in the MCU.

The first thing during the development of the sketch, was to make sure that I can actually communicate with the sensor. For many digital sensors, once the communication is established it is halfway done.

First of all, the functions used to read 8 and 16 bit values from the BMP085 were implemented.

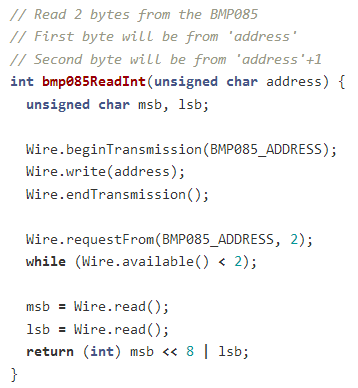
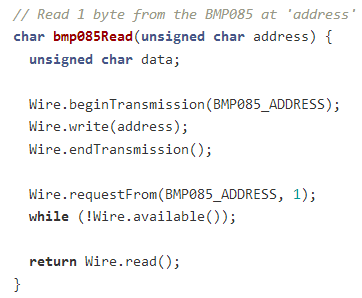


Figure 18 - Function to read 1 byte from the bmp085

Figure 19 - Function to read 2 bytes from the bmp085

According to sensor’s datasheet, the 176 bit E2PROM memory is partitioned in eleven words of 16 bit each. These contain eleven calibration coefficients. Before the first calculation of temperature and pressure, the master reads out the E2PROM data (The function at figure 20 is responsible for this task. It is being called only once during the setup process). The data communication can be checked by checking that none of these words has the value of 0 or 0xFFFF. The parameter association with the Registers Addresses can be seen in the Table 7.

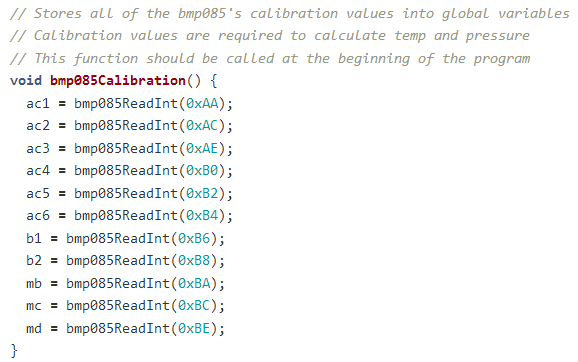
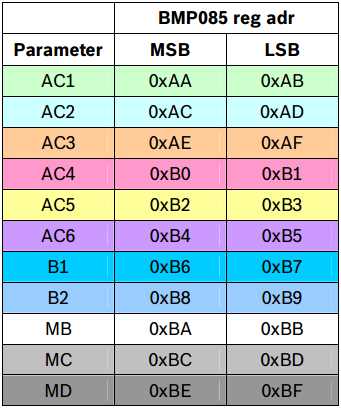


Figure - BMP085 Calibration

Table - BMP085 Calibration Coefficients

Once we have read the calibration values, we just need two more variables in order to calculator temperature and pressure. ‘UT’ and ‘UP’ are the uncompensated temperature and pressure values. Every time the temperature or pressure is measured, first ‘UT’ and ‘UP’ values must be read. They are the starting point for calculating the actual temperature and pressure values. The uncompensated temperature is an unsigned 16-bit integer number while uncompensated pressure is an unsigned 32-bit long number. The two functions can be seen at figures 21, 22.

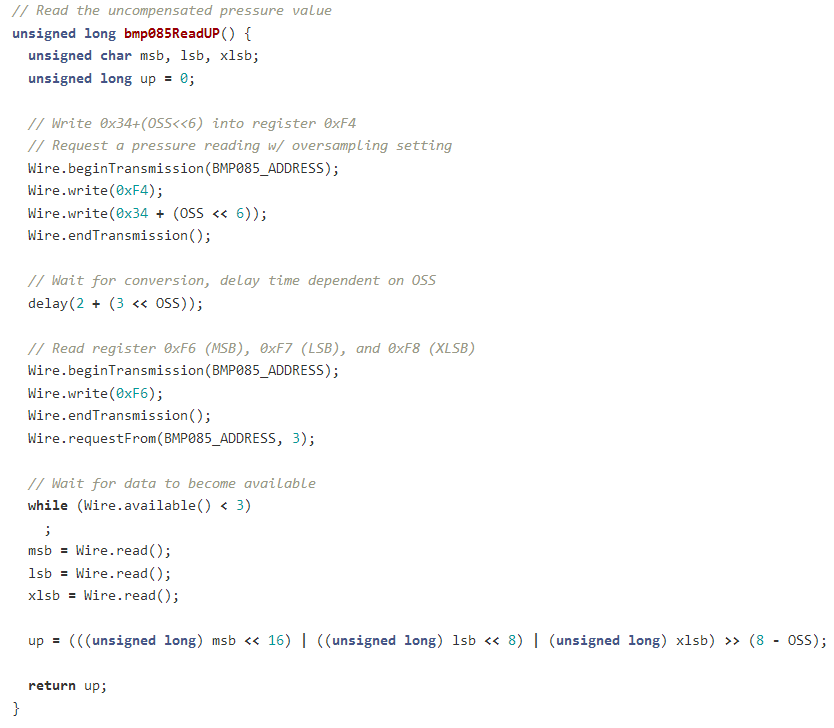
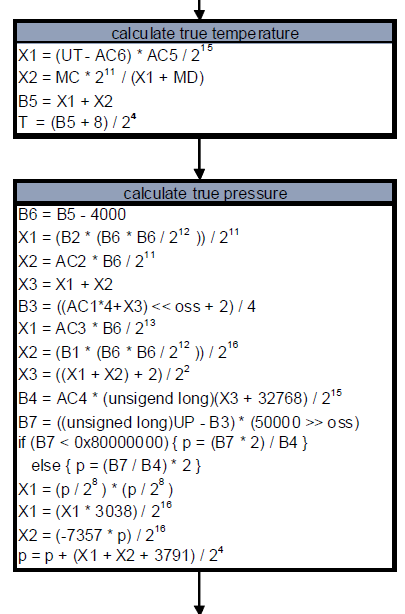


Figure 21 - BMP085 Read Uncompensated Pressure Function



Figure 22 - BMP085 Read Uncompensated Temperature Function

For last part of the calculation process, two functions are needed to take care of the following:



The temperature calculation is rather a simple task, but the pressure ones are more complicated.

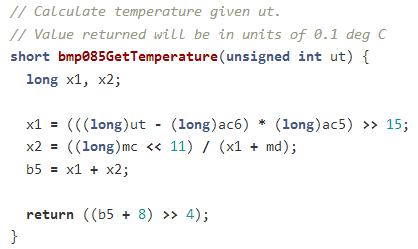


Figure 23 - BMP085 Final Temperature Calculations

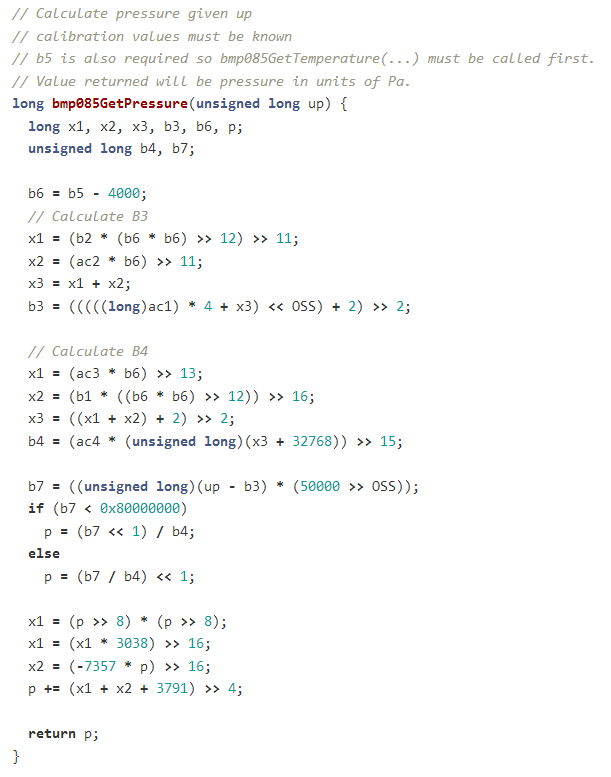


Figure 24 - BMP085 Final Pressure Calculations

##### Anemometer and Rain Gauge

Gathering the data from the anemometer and the rain gauge was an almost similar procedure, this is why they are analyzed together. The concept behind is the Interrupts mentioned at 4.1.1.1, 4.1.2.3, 4.1.2.4.

Firstly, pins and interrupts must be configured. Interrupts are set to FALLING mode, for when the pin goes from HIGH to LOW (Figure 25).

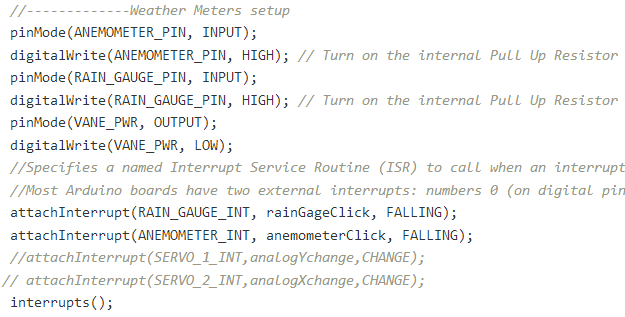


Figure 25 - Weather Meters Configuration

The next step is to implement rainGageClick (Figure 10) and anemometerClick (Figure 12) functions that are called whether an interrupt occurs. These functions are very simple. They just count how many times an interrupt occurred.

Finally, the counters values are used to calculate the final wind and rain depending on each sensor’s factor. (i.e. For Rain each interrupt occurred corresponds to 0.2794mm of rain, instead for wind it is 2.4 miles per hour).

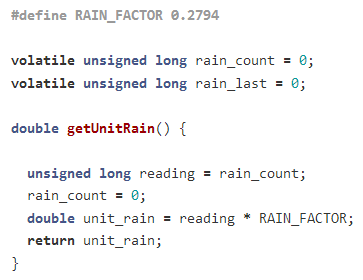


Figure - Rain Final Calculation

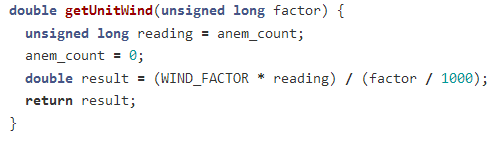
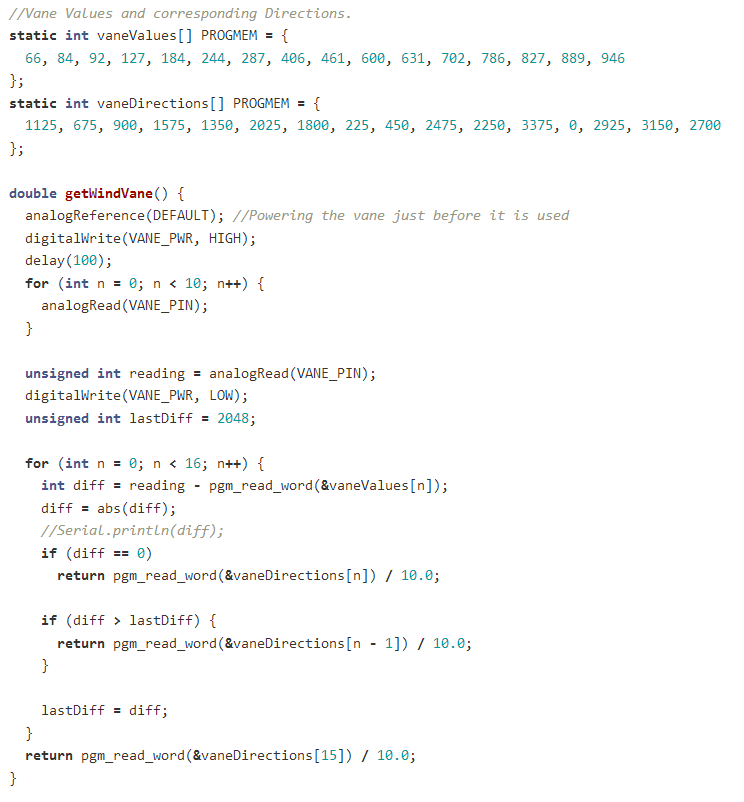


Figure 27 - Wind Final Calculation

##### Wind Direction

Wind Direction’s function compares the ADC value to the ideal values in vaneValues array and returns the corresponding wind direction in degrees.



To save SRAM memory, PROGMEM [25] is used to store data in flash instead of SRAM. For the purpose of the project it is not really useful since memory is enough, but it is a programming practice worth mentioning, especially when referring to the domain of MCUs that memory is valuable.

The pgm\_read\_word function [24] pulls the values out of FLASH and into SRAM when needed but only works with integers, so the actual wind directions are stored as 10x their value.

#### HTTP Request

The HTTP Request sends the sensors data to the MYSQL database. The request method used is GET [26]. GET can be used to send small amounts of data to the server through URL Encoding.

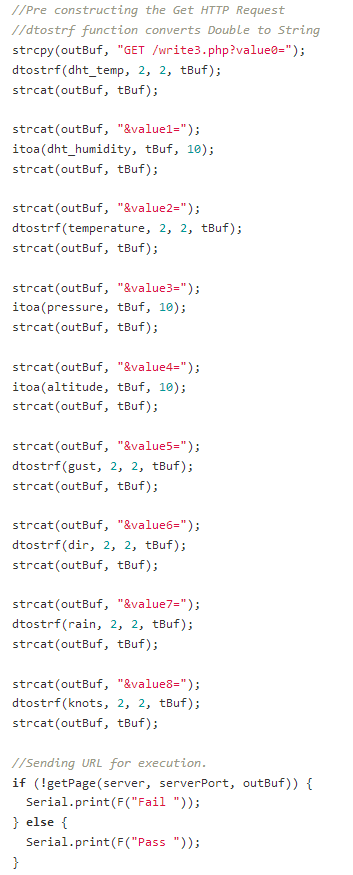
The construction of the URL (figure 28) is being done before the HTTP request process, so Arduino can send the data in one packet using the write function of Ethernet library.

Figure - URL contruction

When the URL assembly process is done, it is passed to getPage( ) (figure 29) function.

This function connects to the server and executes the HTTP Request with the data included within the URI.

The getPage function also has a connection status control. If the connection fails or the server stalls, the Arduino code shows a “Timeout” message after ten seconds and continue to attempt the connection every a time threshold we set.

Dtostrf function explained

Dtostrf (double val, signed char width, unsigned char prec, char\* s)

converts the double values passes into an ASCII representation that will be stored under s. The function returns the pointer to the converter string s.



Figure - HTTP Request Execution

#### Pan Tilt Camera Mount Control

The Remote Camera Control part of the Arduino sketch, controls the Pan Tilt [27] Bracket remotely. Arduino and Ethernet shield serves up a web page that allows the control of the servos that rotate the bracket. When a client connects to the Arduino server, it sends back to the client’s browser the HTML page (figure 30, 31).

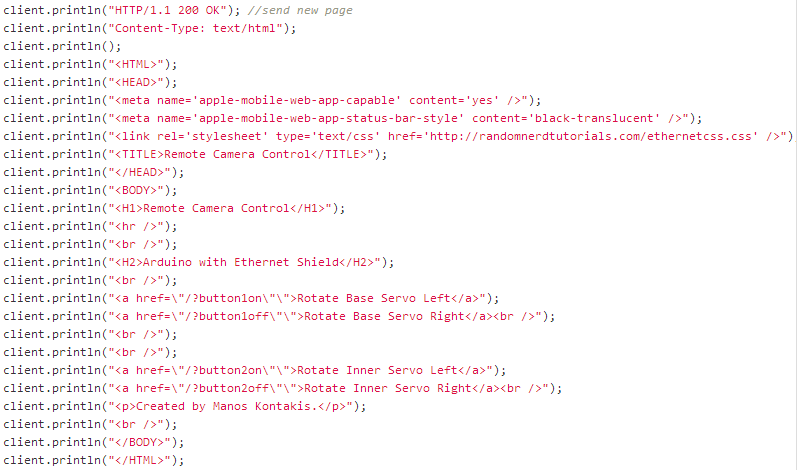


Figure - HTTP Response

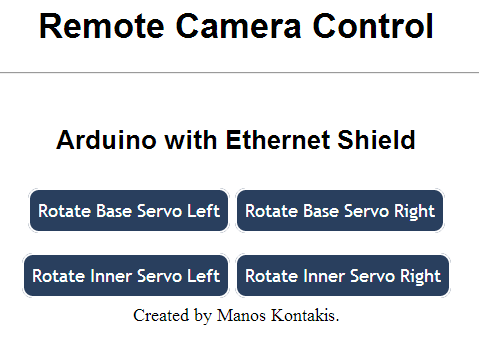


Figure 31 - Remote Camera Control

When the user clicks on a button, the browser generates a HTTP Request with the button name attached to the URL. Arduino reads the HTTP Request and checks which button triggered the request.

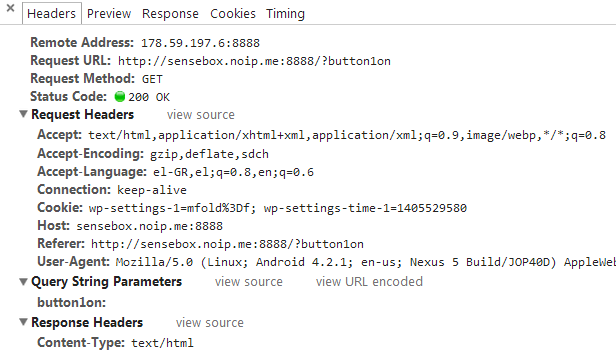
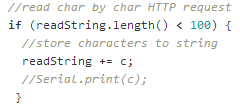


Figure 32 - Generated HTTP Request on button click

indexOf locates a character or String within another String. Returns the index of the desired String within the String or -1 if not found.

The rest of the code rotates the servo.

Figure

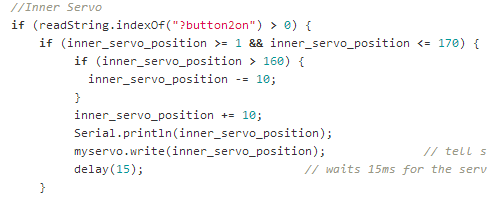


Figure - Servo Control Example

### Android Application

The Android Application processes the weather data and presents them in different formats:

* Charts
* Current Weather Conditions Report

Figure 35 - Android App Icon

* Min / Max Report
* Data presentation in Table

The Charts are generated using the GraphView [28] library. GraphView is a library for Android to programmatically create flexible and elegant diagrams. The reason why I picked this library is because it is easy to understand, to integrate and to customize it. The library supports two different types of diagrams. Line and Bar Charts.

As soon as the app is launched the Splash Screen appears. During the Splash Screen, the weather data are fetched, in JSON format, and parsed to generate the 24 Hours Charts (First Screen).

Figure - Splash Screen

### Server – Side PHP Scripts

PHP is a server – side scripting language designed mainly for web development but also as a general – purpose programming language. A PHP interpreter is usually responsible to process PHP, which is usually implemented as a web server’s native module. After the PHP code is interpreted and executed, the web server sends resulting output to its client. This is exactly how PHP is used in the project.

##### Storing Data in **Database**

In *Figure 28 - URL contruction* the URL is constructed and data are URL encoded. The script in Figure 37, extracts the data from the URL, establishes a connection to the MYSQL database and stores data. For debugging, it returns to the client the data stored.

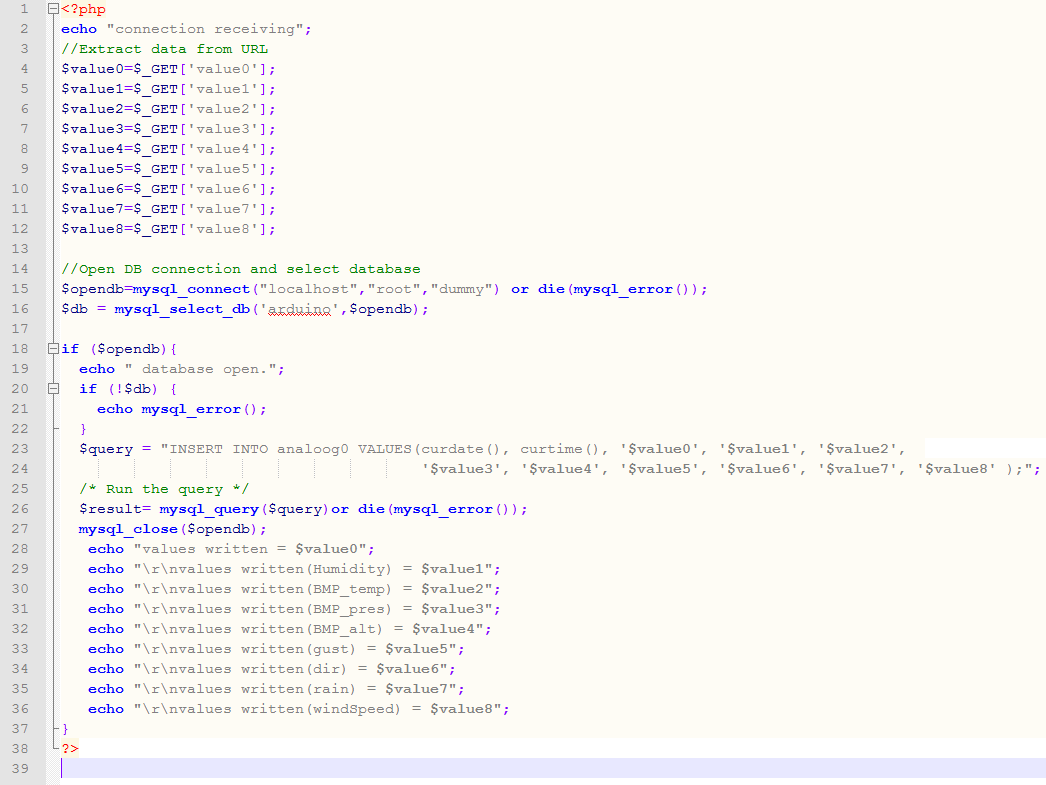


Figure 37 - Script to store data to database

//TODO: Fix google chart’s fetching data PHP script (generalize – optimize) and make report

##### Android Application Fetching Script

This PHP script constructs the corresponding SQL Query depending on three variables attached on the URL, the Sensor, the Flag and the Separator.

The Sensor variable declares the sensors whom data will be fetched. The Flag informs the script which selection was clicked from the Navigation Drawer menu while the Separator variable separates Min and Max values.

*http://sensebox.noip.me/dynamicQueryExecutor.php?sensor=BMP\_temp&flag=6&separator=ASC*

Figure 38 - URL example

Something that worth mentioning here are the cases of the Weekly, Monthly and Three Months Charts. In these three cases high resolution of the charts are of no importance. Also generating seven charts with so many samples (about 22000 for Three months, 7333 for one month per chart) makes the application too heavy for a mobile device. To fix this problem one row per two rows are selected from the table for the weekly graph, one per 8 for the monthly and one per 10 for the 3 months chart. The SQL Query for the Monthly chart is shown below.

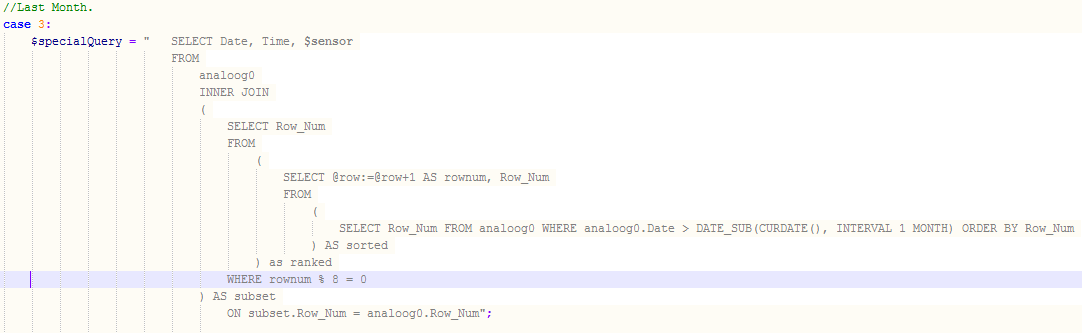


Figure 39 - SQL Query to select every n-th row

### Web Site

# References

1. Open Source Initiative: The Open Source Definition.

<http://opensource.org/docs/osd>

1. Open Source Hardware Association: OSHW Definition.

<http://www.oshwa.org/definition/>

1. Definition of Free Cultural Works: *Open Source Hardware.* [http://freedomdefined.org/OSHW](http://freedomdefined.org/OSHW%20)
2. Arduino: Homepage

<http://arduino.cc/en>

1. Arduino: *Reference.*

[http://arduino.cc/en/Reference/HomePage](http://arduino.cc/en/Reference/HomePage%20)

1. World Meteorological Organization, 1992: *International Meteorological Vocabulary*. Second edition, WMO-No. 182,Geneva.
2. Apache License, Version 2.0

<http://www.apache.org/licenses/LICENSE-2.0.html>

1. GNU General Public License, Version 2.0

<http://www.gnu.org/licenses/gpl-2.0.html>

1. System on a chip

<http://en.wikipedia.org/wiki/System_on_a_chip>

1. Google Charts API

<https://developers.google.com/chart/>

1. Arduino Mega 2560 R3 Web page.

<http://arduino.cc/en/Main/ArduinoBoardMega2560>

1. Arduino Ethernet Shield Web page.

<http://arduino.cc/en/Main/ArduinoEthernetShield>

1. Arduino Ethernet Library Web page.

<http://arduino.cc/en/reference/ethernet>

1. Arduino SD Library Web page.

<http://arduino.cc/en/Reference/SD>

1. RHT03 Official Datasheet.

<http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Weather/RHT03.pdf>

1. Bosch Sensortec: Pressure Sensors.

<http://goo.gl/SEpmgY>

1. BMP085 Datasheet.

<http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Pressure/BST-BMP085-DS000-06.pdf>

1. Electrostatic Sensitive Device

<http://en.wikipedia.org/wiki/Electrostatic-sensitive_device>

1. Sparkfun Electronics: Weather Meters Kit

<https://www.sparkfun.com/products/8942>

1. Sparkfun Electronics: Weahter Meters Datasheet

<https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly..pdf>

1. Fritzing

<http://fritzing.org/home/>

1. Adafruit Industries

<https://www.adafruit.com/>

1. Arduino Sketch Repository Link

<https://github.com/mKontakis/Sensebox/blob/master/WeatherMeters/ver11/ver11.ino>

1. <AVR/pgmspace.h>: Program Space Utilities:

<http://www.nongnu.org/avr-libc/user-manual/group__avr__pgmspace.html>

1. PROGMEM

<http://arduino.cc/en/Reference/PROGMEM>

1. HTTP Request: GET

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3>

1. Pan – Tilt Bracket

<https://www.sparkfun.com/products/10335>

1. Android GraphView

<http://android-graphview.org/>