Bachelorarbeit

Raspberry-Controlled Sensors For Smart User Trial Environments

Verfasser

Silvio Schmidt

angestrebter akademischer Grad

Bachelor of Science (BSc)

Wien, 2016

Studienkennzahl lt. Studienblatt:	033 521
Fachrichtung:	Informatik - Medieninformatik
Betreuer:	Univ.-Prof. Dipl.-Math Dr. Peter Reichl, M.A. St. Dipl.-Ing. Patrick Zwickl, BSc Svenja Schröder, MSc

Eidesstattliche Erklärung

Hiermit erkläre ich eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Wien, am 30. 01. 2016

Silvio Schmidt

Contents

List of Figures

List of Tables

Abstract

Faculty of Information Technology

Bachelor of Science

Raspberry-Controlled Sensors For Smart User Trial Environments

by Silvio Schmidt

Humans can easily be distracted by environmental conditions like temperature or brightness. Therefore user testing can be influenced by different room conditions. A former project focused on designing and implementing a user testing platform will therefore be extended by a display of the current room conditions in order to make stable user testing easier.

The aim of this work is to build an inexpensive device capable of gathering data about the current room conditions, storing and displaying the gathered data cloudless on a different local device, just using a Raspberry Pi 2 Model B and a self-assembled breadboard with sensors. The breadboard will be controlled by a Python script running on the Raspberry device and will furthermore send the data to a Ruby on Rails server running in the same network. The Rails platform did already exist and was extended for this work with the abilities to receive, display and save the data locally.

Abbreviations

IoT: **I**nternet **o**f **T**hings
RPi: **R**asberry **Pi**
RoR: **R**uby **o**n **R**ails
°**C**: degree **C**elsius
lx: **L**u**x**

Acknowledgment

I would first like to thank my thesis advisors Univ.-Prof. Dipl.-Math Dr. Peter Reichl, M.A. St., Svenja Schröder, MSc and Dipl.-Ing. Patrick Zwickl, BSc of the research group Cooperative Systems for their support and advice. Their doors were always open whenever I ran into a problem. In addition to that I want to thank all of my friends and colleagues who checked this work for errors.

1 Introduction

This section should enlighten the reader what the motivation is behind this work and what the objectives are.

1.1 Motivation

Humans can easily be distracted by environmental conditions like temperature or brightness. Due to this user testing can be influenced by different room conditions. But an operator testing a subject can not just interrupt testing in order to consistently check on the actual state. Therefore a device constantly checking on these conditions to provide information about present conditions would be a great help for an operator, so he can make the right decision on what counter measures to be made without being in use of interrupting the test. This is called home automation, an objective of the research area of the Internet of Things. IoT aims at making life easier for everyone by connecting things of everyday use to the Internet, starting with tiny sensors right up to washing machines [26]. A former project focused on designing and implementing a user testing platform will therefore be extended by a display of the current room conditions in order to make stable user testing easier. In addition to that this work could be the foundation for further projects to not only monitor conditions but also take automated counter measures if a value is off limits.

1.2 Objectives

The aim of this work is to build an inexpensive device capable of gathering data about the current room conditions, storing and displaying the gathered data cloudless on a different local device, just using a Raspberry Pi 2 Model B and a self-assembled breadboard with sensors. The breadboard will be controlled by a Python script running on the RPi device and will furthermore send the data to a RoR server running in the same network. The Rails platform did already exist and was extended for this work with the abilities to receive, display and save the data locally. In addition to that, comfort values had to be evaluated in order to know when a tested subject feels uncomfortable and could be distracted from executing tests. It is also shown to the operator on the RoR platform if a value is off limits.

2 Related Work

There are some works that cover similar topics, like Andrew K. Dennis' *Raspberry Pi Home Automation with Arduino* [33] for example. But in contrary to that book, this work will not be a step by step tutorial on how to set up a RPi and will also not make use of a Arduino [8] like Dennis did in his work. The article *Raspberry Pi based Interactive Home Automation System through E-mail* [37] from the international conference on optimization, reliability, and

information technology 2014 describes a way to control LEDs assembled on a breadboard connected to a RPi via E-mail. They use the RPi as a receiving device in order to change the current state, whereas this work uses the RPi as an indicator of changing conditions, in order to change the displayed values. Furthermore there was no scientific work that matches this work completely.

3 Explanation of decisions made on what conditions to be monitored and devices, components and programming languages to be used

3.1 Conditions

Adaruit [21] provides a wide range of sensors able to measure environmental conditions. On the one hand conditions possible to be monitored are therefore limited by the sensors being available. But not every condition is worth being monitored in a laboratory on the other hand. An anemometer [7] to measure wind speed for example, would just be odd. The ones being available and the ones making sense in respect of human distraction are temperature, brightness and noise level.

3.2 Raspberry Pi 2 Model B

The RPi is an inexpensive and tiny but powerful computer originally conceived for kids [19]. The Rpi 2 Model B ships with a quad-core ARM Cortex-A7 processor and one gigabyte of memory capacity [18]. Out of the box it runs the Raspbian operating system on a Micro SD Card, which is based on Debian and optimized for the RPi [20].

3.3 Sensors

3.3.1 Photo cell CdS photoresistor

Decision explanation The photo cell [11] in order to measure brightness was already shipped with the Raspberry Pi starter kit [17]. There were no additional decisions to be made.

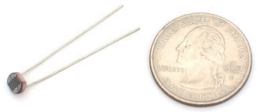

Figure 1: Photo cell [11]

Short description The photocell is a small and inexpensive sensor that reacts on light and is able to detect changes of brightness. It will be used like a resistor because its resistance changes in dependency on the room's current brightness. How the whole electric circuit works will be explained in detail in the section 5.1 *Assembling breadboard and sensors*.

3.3.2 Analogue temperature sensor TMP36

Decision explanation Adfruit also provides a wide range of sensors for just one condition [24]. The TMP36 [4] was the only analogue temperature-only sensor. The other ones had extras like being waterproof [25] or had other means like measuring the temperature of soil [22]. The analogue sensor was chosen because it is thereby possible to use it the same way the photo cell is being used.

Figure 2: TMP36 [6]

Technical overview The TMP36 is a small chip for measuring temperature. As seen in figure 3 it has three pins: the left one is for voltage input, the one in the center is the analogue output and the right one is for grounding. The way this sensor is being used makes the center pin obsolete, though, because it is used as a resistor. The resistance of the TMP36 changes dependent on the current temperature. How the whole electric circuit works will be explained in detail in the section 5.1 *Assembling breadboard and sensors*.

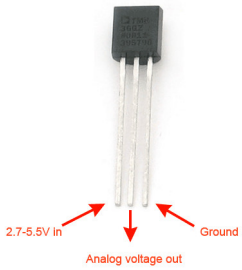

Figure 3: TMP36's backside view [5]

3.3.3 Electret microphone amplifier MAX4466 with adjustable gain

Decision explanation The MAX4466 seemed to be the only fitting microphone.

Technical overview The MAX4466 has to be attached to a header strip. As seen in figure 4 the output is on the left, the middle one is for the grounding and the right one is for voltage input. It ships with an adjustable gain and is able to spot sounds in the range of 20 to 20KHz.

Figure 4: MAX4466 overview [9]

Admission Due to a lack of knowledge on electric circuits and components, it was not possible to get the microphone working on the breadboard successfully. Nevertheless will the noise level be treated in the condition section below.

3.4 Programming languages and framework

The following section contains a short introduction to every programming language and framework being used and a short explanation why it has been chosen for this project.

3.4.1 Python

Python was first designed in 1980 by Guido van Rossum as a follow up on ABC language [16]. it is perfect for the script executed on the RPi, because it is a language that allows short scripts to execute tasks, programs written in different languages (like Java for example) would have needed more work on. [38]

3.4.2 Ruby on Rails and its components

Ruby is a very popular high-level programming language developed by Yukihoro Matsumoto in 1995. He took his favourite parts of Perl, Smalltalk, Eiffel, Ada and Lisp and assembled a new language "that mirrors life" as Matsumoto

stated. Furthermore everything is an object in Ruby, so rules applying to objects also apply to primitive types like numbers whereby it eases the use of Ruby. In addition to that Ruby is called a flexible language because it makes no restrictions on what of its elements being manipulated or not. [29]

Ruby on Rails is an open source framework written in Ruby which uses Model-View-Controller like most web-application frameworks do.

SQLite SQLite is a built in relational database management system. It is serverless, which means there is no client to server connection, but the database itself is included in the program [23]. RoR ships with SQLite and is being used as the model of the MVC-concept.

RoR, Ruby and SQLite are used because the already existing platform that this work extends has been built this way.

4 Condition standards

As part of this work condition standards for interior rooms had to be researched. But due to the reason that everybody sense those conditions differently dependent on various factors (gender for example) the research's focus changed from condition standards to so called comfort values. A comfort value is an average value range at which most people feel comfortable. This work focuses on comfort values for office environments because the office is as close as it gets to the laboratory this project is designed for.

4.1 Brightness

Brightness is being measured in Lux (lx) and is calculated by Lumen (lm) divided by square meter as seen in figure 5.

$$1lx = 1\frac{lm}{m^2}$$

Figure 5: Lux formula

According to Wikipedia a typical room brightness (for example in an office) is 500 lx for instance [10]. Butler and Biner prove in their article *Preferred Lighting Levels - Variability Among Settings, Behaviors, and Individuals* that comfort values of individuals differ according to their gender and current actions being executed [32, P. 713]. Even though the CIE's publication *Lighting of Indoor Work Places* [31] suggests a value of 500 lx as recommended for an office environment, the article *A field survey of visual comfort and lighting energy consumption in open plan offices* shows that approximately 72 percent of the tested subjects are still comfortable between an illuminance of 787 lx and 288

lx [34]. Given those values the range of the comfort value for brightness in this project is set between 500 lx +/- 200 lx.

4.2 Temperature

This work has been written in a germanophone country, therefore the unit used for temperature is degree Celsius (°C). The freezing point on a Celsius scale is at 0 °C. The article *Thermal comfort in residential buildings: Comfort values and scales for building energy simulation* depicts that the sense of temperature is dependent of clothing, the current activity, the gap between indoor and outdoor temperature and gender. Oseland's field study shows, that his testing subjects felt comfortable between 21.8°C and 20.4 °C in an office during winter. This exact comfort value is inherited in this work [36].

4.3 Noise level

Noise is being measured in Dezibel (dB). A running sewing machine creates a noise of 60 dB for instance [27]. Unlike the first two conditions there are no real comfort values in order to measure if it is too noisy. But the article *The Effects Of Nonphysical Noise Characteristics, Ongoing Task And Noise Sensitivity On Annoyance And Distraction Due To Noise At Work* includes a field study at what noise level office workers get annoyed and take counter measures in order to reduce this noise level [30, 127]. So in this work the comfort value equals 0 dB to the point the first office workers got annoyed in Kjellberg's field study, which is 45 db.

5 Implementation

This section contains descriptions of all implemented elements and how those parts are connected.

5.1 Assembling breadboard and sensors

The used sensors have analogue output and the RPi is only able to read digital input. Therefore an other way to read the sensors' input had to be found. As described in a tutorial on the Adafruit homepage [2] and an other tutorial on how to read analogue input with a RPi [28], a simple electrolytic capacitor as indicated in figure 6 can be used to work around that problem. A electrolytic capacitor is a energy storage and is often used to protect electronic devices against too high voltage [35, P. 384]. In this electronic circuit it is used in order to be able to measure the time needed to entirely fill the capacitor. As a result of the time needed to entirely fill a capacitor in dependency on the resistance of all electronic components contained in the electric circuit and the sensors having a different resistance in dependency of the current room conditions (temperature or brightness), the time needed to entirely fill a capacitor is also dependent

on the current room conditions. Therefore the time needed to entirely fill the capacitor represents the current room condition.

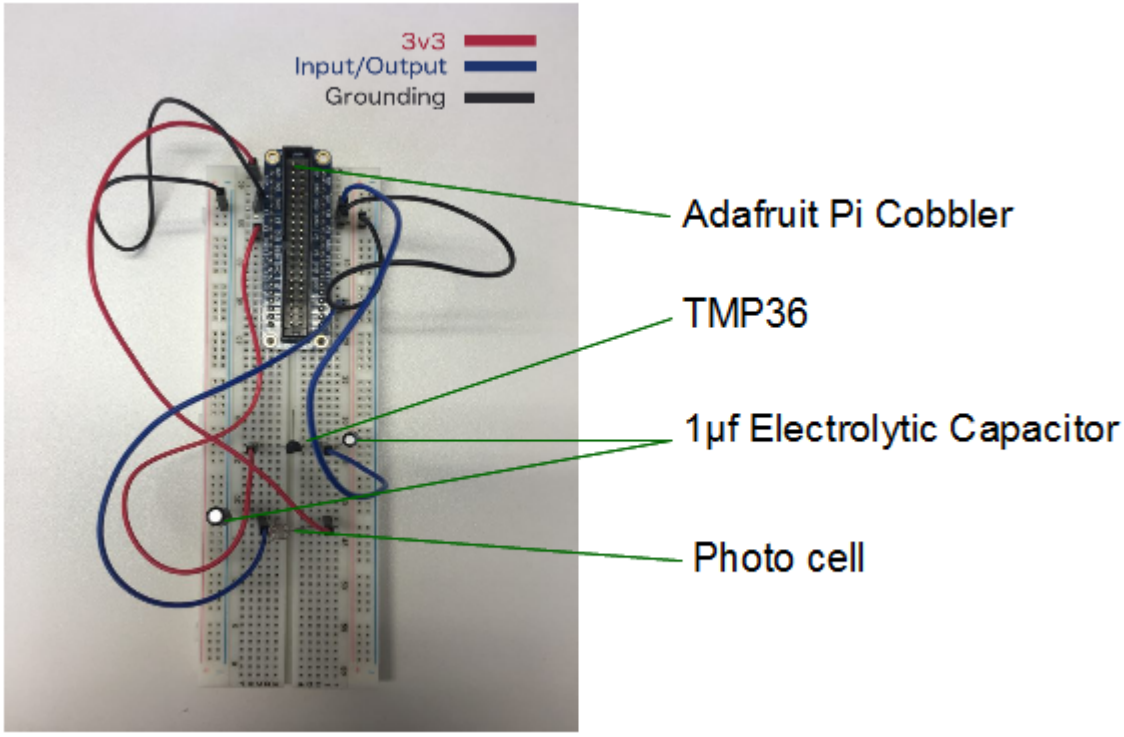

Figure 6: Assembled breadboard overview

The red wires seen in figure 6 are used for voltage input (3.3 volt being used), the black wires indicate grounding and the blue wires are for output (in order to empty the electronic circuit beforehand) on the one hand and input (to get the information when the capacitor is entirely filled) on the other hand. In addition to that a Adafruit Pi Cobbler [3] is used, which imitates the pins on the RPi so the breadboard is easier to assemble and no soldering is needed. Cobbler and RPi are connected through a Downgrade GPIO Ribbon Cable [1].

5.2 Python

The Python script running on the RPi consists of four elements: receiving data from a sensor, processing received data, sending data to the rails server and an infinite loop that loops those three parts.

5.2.1 Receiving data from sensor

Receving data from sensor is achieved just like it is suggested in the Adafruit resistor sensor tutorial [2]. At the beginning the variable *reading* is set to zero. *Reading* will be incremented later in order to measure the time. The connected pin to this circuit will be set as output and set to LOW in order to make sure that the capacitor is empty. After that the same pin is set as input and a while loop is started to increment *reading* for as long as the pin reads LOW. When the capacitor is entirely filled the input pin reads $HIGH$ and ends the while loop. *Reading* is being returned as the time needed to entirely fill the capacitor. To perform actions with the pins the *RPi.GPIO* Python package is being used [13].

5.2.2 Processing received data

In order to translate the received values into actual degree Celsius (°C) and Lux (lx) some samples were taken:

lx	Sensor value	Description
0	3400	darkening with hands
1	3000	darkening with hands
5	2200	darkening with hands
20	750	darkening with hands
80	300	darkening with hands
100	220	living room; daytime; winter
165	165	brightening with flashlight
350	120	bath room; daytime; winter
500	70	brightening with flashlight
1500	50	brightening with flashlight
1900	45	brightening with flashlight
3000	20	brightening with flashlight
13000	1	brightening with flashlight

Table 1: Evaluated Lux samples

The lx and °C-values have bin measured with a lux- and thermometer like shown in figure 7.

The values shown in table 1 and table 2 are the foundation of the interpolation been done in the Python script in order to evaluate the values between the taken samples.

To decide on which interpolation to take a graphical output had been generated as seen in figure 8.

Figure 8 clearly shows that a cubic interpolation is not an option in both cases because it generates spikes way off limits. In contrast to that linear inter-

°C	Sensor value	Description
0	20000	outside; winter
8.4	18500	bed room; winter
10	17000	second bed room; winter
23.2	15500	living room; winter
34.8	12750	heated tiled stove; winter
42	11250	radiator; winter

Table 2: Evaluated °C samples

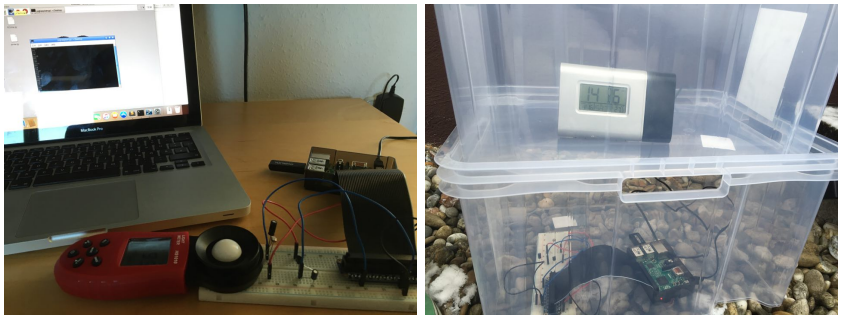

Figure 7: Data Evaluation: lx on the lefthand side, °C on the righthand side

polation is a good approach for approximate values between the taken samples.

In order to not break the script, values below or beyond the taken samples are taken out beforehand because the interpolation would fail otherwise and stop the script, even though those values are very unlikely. Therefore a *way off limits* message is sent to the server, so the operator knows what is going on.

The interpolation is executed by the *scipy.interpolate* package [14].

5.2.3 Sending data to the rails server

After receiving and processing the data the script executes a *POST* command on the URL ***IP**:3000/get_values* with the temperature and brightness data combined as a string "*lux:**bri**,cel:**tmp***", where **bri** and **tmp** are variables. The **IP** of the rails server must be known and hard coded into the script beforehand. At the end the servers answer (if available) is being fetched. For this task the *urllib* library is being used [15].

5.2.4 Infinite Loop

The infinite loop arranges the three elements described above. At first the data of both sensors is being received and saved as variables. This data is then

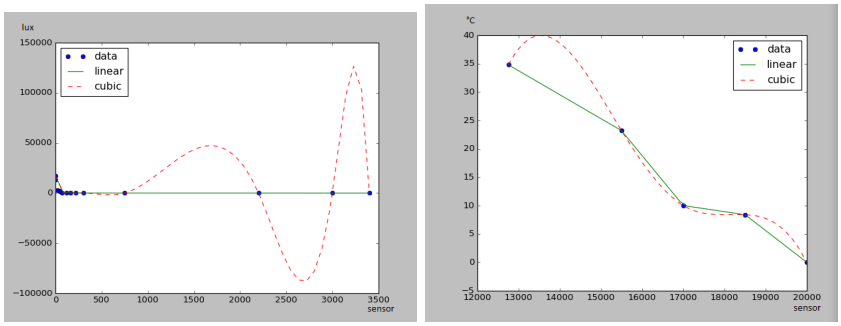

Figure 8: Interpolation Graphs: lux on the righthand side, °C on the lefthand side

processed with linear interpolation and the variables are overwritten with the interpolation results. At last the sending task is being executed with gathered data. If there was no error the loop starts all over again.

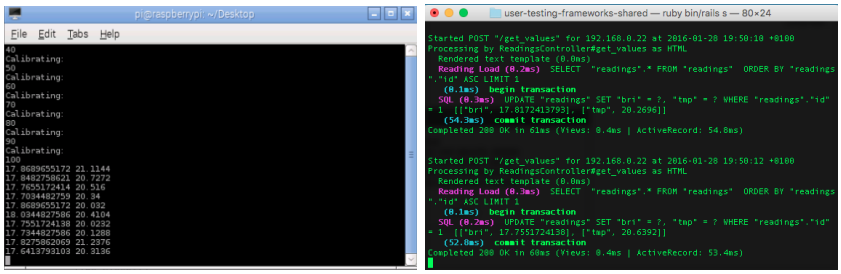

Figure 9: Response of the server displayed on the RPi on the lefthand side, the RoR server receiving data on the lefthand side

5.2.5 Calibrating

Additionally the Python script executes on start up the step *receiving data from sensor* for ten times without using the collected data afterwards as seen in figure 9 because the first four to eight measurements were most of the time way off limits. This way distorted data gets eradicated.

5.3 Ruby on Rails

In this section the RoR parts of this project will be explained, which extend the former work.

Controller The time a *POST* command is being executed by the Python script on the RPi the controller of */get_value* fetches the sent data. The received

string "*lux:**bri**,cel:**tmp***" is then being split, first at the comma into two new strings and subsequent those two new strings at the colons into two new strings containing anything but the needed values. Those values are then saved into the reading table of the database.

View The already existing dashboard made by a former colleague has been extended with a condition display by rendering the view of */get_value* as seen in figure 10. When the dashboard is rendered it fetches the conditions from the readings table and compares them to the hard coded comfortable levels of the respective condition. Finally the values are written into the text fields of the */get_value*'s view and text and that text's colour are adjusted depending on if the value matches the comfort level or not.

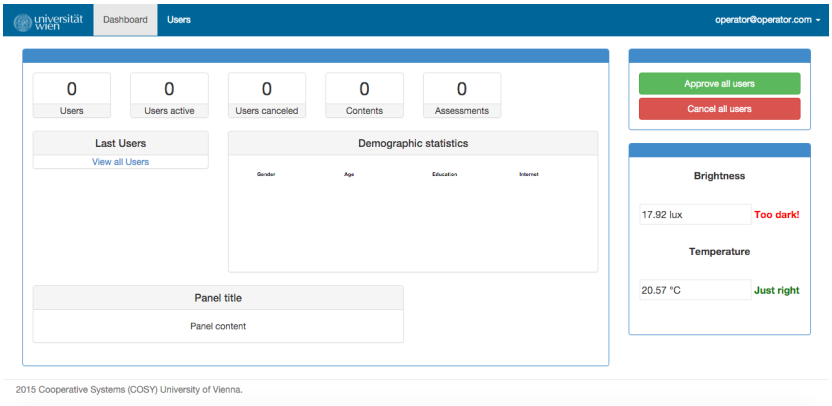

Figure 10: Dashboard View, Condition Display on the righthand side

Model The platform's database has been extended with a table labeled *readings*. It consists of three columns, the id as primary key and auto incremented integer, as well as temperature (labeled tmp) and brightness (labeled bri) as floats.

6 Future Work

This section shows possible extensions to the existing project.

Counter measures Home automation and the Internet of Things are a big deal in modern information technology. Of course it does not stop at being able to monitor things, the great goal would be that the RPi would automatically take counter measures if a condition is off limits. For instance if it is way to dark in the room and the sensor detects a lx-level that is below the desired value

additional lamps could be switched on using a PowerSwitch Tail [12] connected to the breadboard.

Expanding the breadboard In addition to the existing sensors there is also the possibility of adding different new sensors. For example if for whatever reason the operator decides that the wind speed distracts the testing subjects an anemometer could be mounted at anytime. Furthermore multiple identical sensors could be added in order to get a more stable reading on the current conditions because a single sensor on its own proves to be quite volatile.

Extending the graphical representation If at some point the operator decides he is in need of precise data of conditions over a certain time so he can track certain user behaviour at a certain condition, the database could be extended and the graphical depiction could be improved and visualized by graphs.

7 Conclusion

In this bachelor thesis an inexpensive fully functional room condition monitor with local data storage has been developed using a RPi, analogue sensors, RoR and a simple Python script. Additionally comfort values for conditions being able to distract test subjects and endanger stable testing have been researched successfully. After a short description of every used device, component and programming language a detailed explanation of how the components work and intertwine was given.

References

[1] Adafruit downgrade gpio ribbon cable - adafruit industries. https://www.adafruit.com/product/1986.

[2] Adafruit photocell tutorial. https://learn.adafruit.com/basic-resistor-sensor-reading-on-raspberry-pi/basic-photocell-reading.

[3] Adafruit pi cobbler plus - adafruit industries. https://www.adafruit.com/products/2029.

[4] Analog temperature sensor - tmp36: Adafruit industries. https://www.adafruit.com/products/165.

[5] Analog temperature sensor - tmp36: Adafruit industries, overview picture. https://learn.adafruit.com/system/assets/assets/000/000/471/original/temperature_tmp36pinout.gif?144

[6] Analog temperature sensor - tmp36: Adafruit industries, picture. https://learn.adafruit.com/system/assets/assets/000/000/470/small360/temperature_TMP36_LRG.jpg?1

[7] Anemometer wind speed sensor: Adafruit industries. https://www.adafruit.com/products/1733.

[8] Arduino website. https://www.arduino.cc/.

[9] Electret microphone amplifier - max4466: Adafruit industries, overview picture. https://learn.adafruit.com/system/assets/assets/000/003/631/medium640/sensors_2013_01_12_IMG_1024.jpg?1396799199.

[10] Lux, wikipedia. https://www.wikipedia.org/wiki/Lux_(Einheit).

[11] Photo cell: Adafruit industries. https://www.adafruit.com/products/161.

[12] Powerswitch tail. http://www.powerswitchtail.com/Pages/default.aspx.

[13] Python rpi.gpio package. https://pypi.python.org/pypi/RPi.GPIO.

[14] Python scipy.interpolate package. http://docs.scipy.org/doc/scipy/reference/interpolate.html.

[15] Python urllib library. https://docs.python.org/2/library/urllib.html.

[16] Python wikipedia. https://en.wikipedia.org/wiki/Python_(programming_language).

[17] Raspberry pi 2 model b starter pack: Adafruit industries. https://www.adafruit.com/products/2380.

[18] Raspberry pi 2 model b technical details. https://www.adafruit.com/pdfs/raspberrypi2modelb.pdf.

[19] Raspberry pi about. https://www.raspberrypi.org/about/.

[20] Raspbian about. https://www.raspbian.org/.

[21] Sensors: Adafruit industries. https://www.adafruit.com/category/35.

[22] Soil temperature/moisture sensor - sht10: Adafruit industries. https://www.adafruit.com/products/1298.

[23] Sqlite about. https://www.sqlite.org/about.html.

[24] Temperature: Adafruit industries. https://www.adafruit.com/categories/56.

[25] Waterproof digital temperature sensor - ds18b20: Adafruit industries. https://www.adafruit.com/products/381.

[26] Wlan washing machine. http://www.waschmaschinentest.net/wlan-waschmaschinen/.

[27] Vom ticken der uhr bis zum presslufthammer, 2004. http://www.welt.de/print-welt/article334313/Vom-Ticken-der-Uhr-bis-zum-Presslufthammer.html.

[28] Read analog input using a raspberry pi, 2012. http://www.raspberrypi-spy.co.uk/2012/08/reading-analogue-sensors-with-one-gpio-pin/.

[29] Ruby about, 2016. https://www.ruby-lang.org/en/about/.

[30] Anders Kjellberg, Ulf Landstrm, M. T. L. S. E. A. *Journal of Environmental Psychology 16*. http://www.sciencedirect.com/science/article/pii/S0272494496900109.

[31] CIE. Lighting of indoor work places, 2001. Publication D-008/E-2001.

[32] Darrell L. Butler, P. M. B. *EnvironmentBehavior 19*. http://eab.sagepub.com/content/19/6/695.short.

[33] Dennis, A. K. *Raspberry Pi Home Automation with Arduino*. PACKT, 2013.

[34] Geun Young Yun, Hyo Joo Kong, H. K. J. T. K. *Energy and Buildings 46*. http://www.sciencedirect.com/science/article/pii/S0378778811004981.

[35] Michael Kofler, Charly Kühnast, C. S. *Raspberry Pi Das umfassende Handbuch*, vol. 2. Rheinwerk Computing.

[36] Oseland, N. *Energy and Buildings 21*. http://www.sciencedirect.com/science/article/pii/0378778894900159.

[37] Sarthak Jain, Anant Vaibhav, L. G. Raspberry pi based interactive home automation system through e-mail. *Optimization, Reliabilty, and Information Technology (ICROIT)* (2014), 277–280.

[38] Summerfield, M. *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. PRENTICE HALL.