Anh Nguyen

Collecting and Visualizing Environmental Factors

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Electronics

Bachelor's Thesis

April 20th 2020

Author Title Number of Pages Date	Anh Nguyen Collecting and Visualizing Environmental Factors 42 pages + 6 appendices April 20th 2020
Degree	Bachelor of Engineering
Degree Programme	Electronics
Professional Major	
Instructors	Janne Mäntykoski, Senior Lecturer

Environmental factors contribute a great impact on many industries nowadays. Especially in agriculture and animal husbandry, the measures are extremely important as they help farmers control and improve the quality of their products. The main goal of this project was targeted in collecting and presenting data from surrounding environmental factors. From the provided data, users are able to monitor humidity, temperature, soil moisture and luminosity.

Measurements of humidity and temperature from sensor DHT11, as well as a light dependent resistor module, and a soil moisture signal amplifier were focused on in this project. Several communication standards were used, including serial connection, wireless communication and HTTP post request method. The microcontrollers used in this project were Raspberry pi 3 and Arduino Uno R3. Program for Arduino was written in C programming language, and Raspberry pi was programmed by using Python scripting language. Metropolia student virtual server was used as a host to where collected data are sent and stored. A single web page application was created for user to monitor data which is drawn by using Google chart. The idea of how to collect analogue data on Raspberry pi by using an Arduino was also presented on this project.

A small installation of internet of things concept was successfully created by this project. The sensors unit was capable to sense and demonstrate environmental factors measurements. The microcontrollers unit performed as expected and data were transferred effectively. The communication line worked as well as cloud storage operated unsurprisingly. With further development, this setup will be able to improve its quality, in addition to its performance.

Keywords	arduino, raspberry pi, analogue reading, visualization, Google chart, serial connection, DHT11, soil moisture, LDR

Contents

List of Abbreviations

ACED	Apparatus for collecting environmental data (name of this project device)
CSV	Comma Separated Values
GND	Common ground
HTTP	Hypertext transferring protocol.
I^2C	Inter-Integrated Circuit
IC	Integrated circuit.
IoT	Internet of things.
JSON	JavaScript object notation
LDR	Light dependent resistor
LED	Light emitting diode
MCU	Microcontroller
NTC	Negative temperature coefficient.
RAM	Radom access memory.
SPI	Serial peripheral interface
URL	Uniform resource locator
USB	Universal serial bus
Vcc	Collector supply voltage

1 Introduction

Agriculture is dramatically changing nowadays aside with the evolution of Industry 4.0. The old-fashioned way of farming and husbandry is gradually alternating by machinery and advanced technologies. Environmental factors play a significant role in this agricultural science. The quality of living things such as plants and animals are greatly influenced by humidity, temperature, soil moisture and luminosity. Because of the need for monitoring and controlling environmental factors, this project was designed.

The main goal of this project was concentrated on collecting and illustrating data. A small set up of IoT concept was also introduced. The ACED device worked as a physical interacting gadget which harvests environmental factors indicators. It comprises three essential divisions: sensors unit, reader units and collector units. Data collected from sensors will be transferred wirelessly to a server. From there, a graph will be created by visualization tool.

Arduino and Raspberry Pi were chosen to be used on this project. As both have serials connection, the communication link between two microcontrollers has been established easier. Raspberry Pi was not designed to have analogue signal reading feature, which causes a difficulty for sensors which have analogue output. This problem was solved by using Arduino as an intermediate. With the ATmega328P-AU chip and analogue reading pins of the Arduino, collecting analogue data from real world becomes simpler.

Theoretical knowledge was presented on the next chapter, where information related to microcontroller, sensors, serial connection, HTTP requests and Google chart can be found. The third chapter demonstrated components which are used on making this project. Chapter four discussed about how the ACED device has been created, transferring data to server and Google chart usage. The papers in chapter five showed the performance evaluation, measurements and advantages of this study. In final chapter the conclusion was illustrated as well as some future development ideas.

2 Theoretical Background

2.1 Microprocessor and Microcontroller

Microprocessor is a vital element on any electronics device. Since the revolution of transistor in 1948, microprocessor is one of the most valuable innovation in the electronics field [1]. Microprocessor appears everywhere in every digital application, from kids toy to space station. Microprocessor is a processor on a computer which functions as a central processing unit. It is usually integrated on a single or up to eight IC(s) [2]. Figure 1 below shows the basic operation of a microcontroller with a microprocessor.

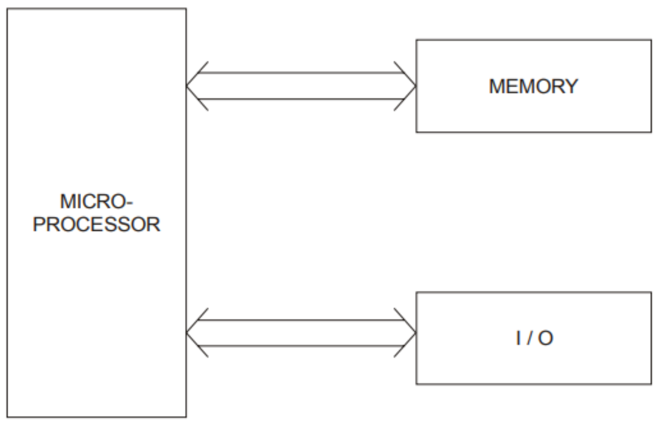


Figure 1. Block diagram of a microcontroller. (Reprinted from [1])

Microcontroller (also referred as microcomputer) is a low-cost, low-power computer which usually include CPU, RAM, programmable flash memory, input/ output interface, timer and analogue to digital conversion and/ or digital to analogue conversion. The device is often programmed by an instructed code to control one or multiple task. Since microcontroller is commonly embedded into the device it controls, it is also called embedded controller. [3]

2.2 Sensors

Sensors have become an essential component in each electronics design since they were created. They appear in homes, offices, vehicles etc. in order to make life easier. The office light can turn on itself when there are people present, the room temperature is automatically adjusted when it meets conditions, a warning shows up when car’s engine is too hot etc. These actions are done thanks to sensors. Some applications of sensors in a vehicle are demonstrated in figure 2.

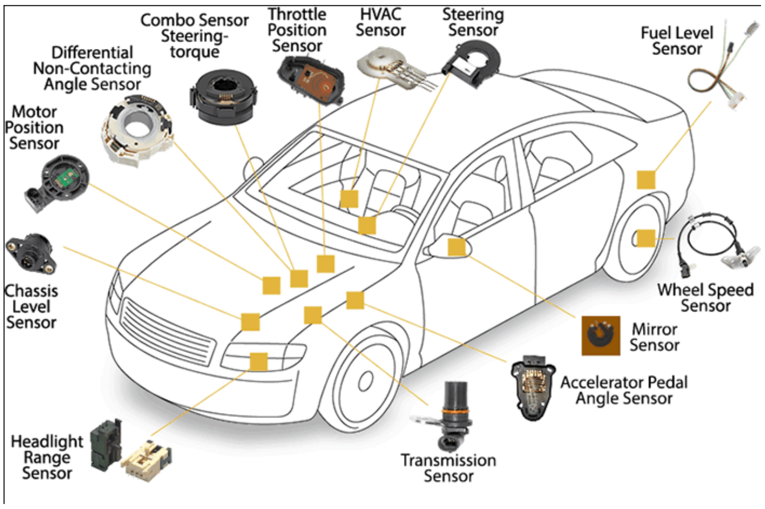


Figure 2. Sensors in a personal auto. (Reprinted from [4])

Transducer is a device where energies are converted from one form to another. Sensor is a type of transducer, where physical environment is taken as an input and produces output as a digital or optical signal. The output is created as a result of the device’s sensitivity towards light, heat, motion, smoke, etc. Sensors are classified by two main type, active sensor and passive sensor. That require an external power source to operate are called active sensor, while passive sensor on the other hand, are that do not necessitate any peripheral power and directly generate output signal response. Sensors can spawn digital or analogue output [5]. There are different types of sensing devices, for instance temperature sensor, humidity sensor, soil moisture sensor and light sensor.

2.2.1 Temperature Sensor

Temperature sensor is one of the most popular one in smart system. As its name, this device seizes environment heat as input, and results the changes in resistance. The value of the output is corresponded to surrounding temperature. Different types of temperature will create different output. Currently there are thermocouple, thermistor, resistor temperature detector, infrared, semiconductor and thermometer. Each one has its advantages and disadvantages, therefore choosing right one for usage is very important [6].

Thermistor (short for thermally sensitive resistor) is a commonly used sensor, which is cheap and easy to use. Device's resistance changes when the surrounding temperature changes. It is vulnerable to damage since mostly thermistors are made from manganese and oxide nickel [6]. Negative temperature coefficient thermistors are the most wide-spread used type of temperature sensor as it can be operated everywhere. NTC thermistors have a negative electrical versus environment temperature connection, which means the higher temperature is, the lower the resistance becomes. A small changes in temperature can cause a weighty displacement in electrical resistance [7].

The relativity between resistance and temperature of a typical 10kΩ NTC thermistor is illustrated in figure 3. Note that its resistance decreases when temperature increases due to negative coefficient.

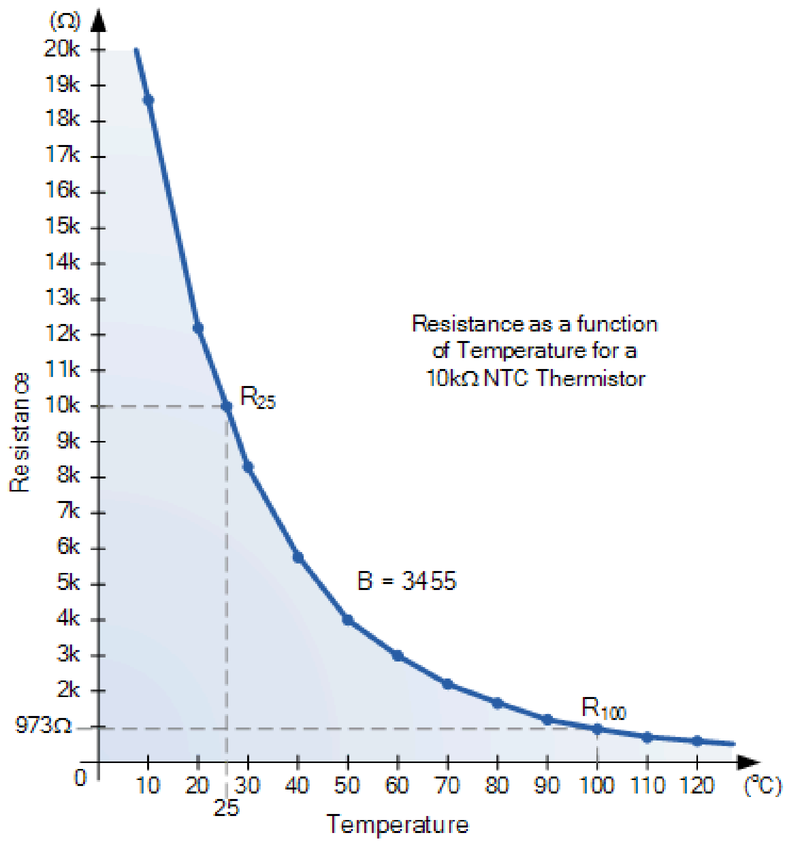


Figure 3. The relation between resistance and temperature. (Reprinted from [7])

The electrical resistance in a NTC output can be calculated by the following equation:

$$B = \frac{T2 * T1}{T2 - T1} * \ln\left(\frac{R1}{R2}\right) \qquad (1)$$

Where:

- T1 is the first temperature point in Kelvin (K).
- T2 is the second temperature point in Kelvin (K).
- R1 is the thermistor resistance at temperature T1 in Ohms (Ω).
- R2 is the thermistor resistance at temperature T2 in Ohms (Ω). [7]

2.2.2 Humidity Sensor

Moisture produced by water condensation in the air creates humidity. Humidity is the amount of water vapors appear in the air. In agriculture and many industries, humidity becomes an important element. Humidity sensor is a necessary factor in any devices, since a small value exceeds threshold can lead to damage or malfunctioning [8]. It senses the quantity of water vapor in surrounding environment.

Generally there are three types of humidity sensor: capacitive, resistive and thermal. The capacitive one is commonly used in electronics application. A capacitive sensor senses humidity by using capacitor with a slim band of metal oxide between two conductors. The metal's electrical capacity varies in respect to relative humidity of atmosphere. When there are changes in humidity, the electrical permittivity of the dielectric material is assessed to result a humidity value [8]. A relative humidity sensor typically comprises a humidity sensing factor and a thermistor. Humidity output also depends on temperature, mass, pressure, resistivity.

Figure 4 demonstrates a capacitive humidity sensor (also known as hygrometer).

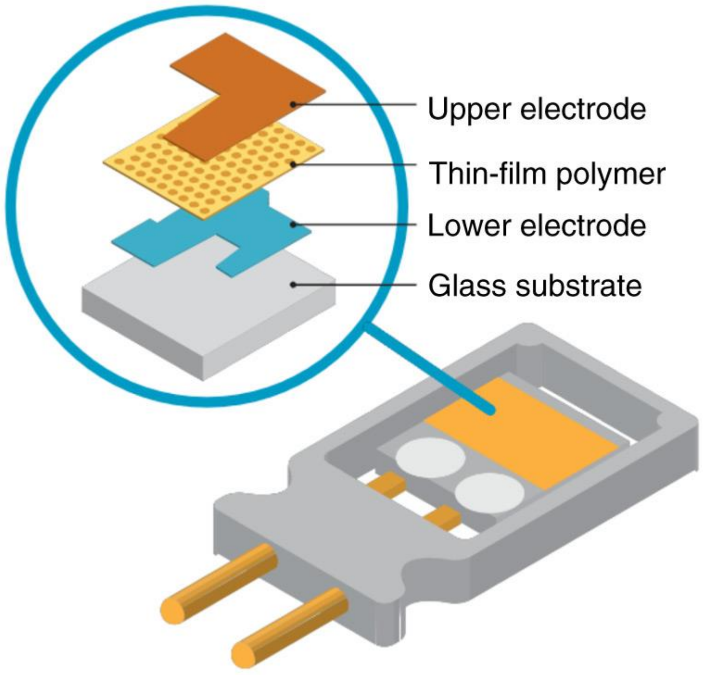


Figure 4. Structure of a capacitive humidity sensor. (Reprinted from [26])

Capacitive humidity sensors are used for several applications such as weather station, food processing, automobile and refrigerator. Some humidity sensors appear widely in market are DHT11, DHT22 and SHT71.

2.2.3 Soil Moisture Sensor

In order to keep track of the amount of water, soil moisture sensor appears to be a necessary tool. This device measures water volumetric in soil. Typically a soil moisture sensor includes a two-branch probe whose shape looks like a fork. These parts are in fact electrodes which act as variable resistors. In figure 5 the shape of the two-branch probe is shown.

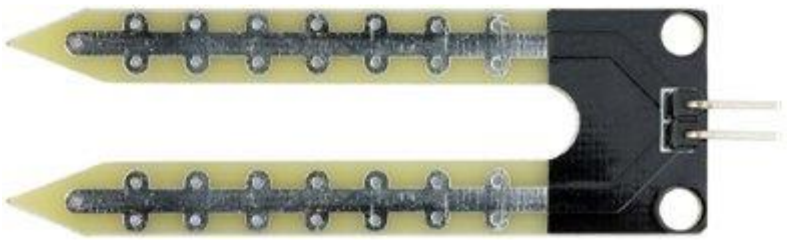


Figure 5. A two-branch probe of a soil moisture sensor. (Reprinted from [9])

The resistance value changes when the amount of water in soil change. Having more water in soil makes the conductivity increase which leads to resistance decrease. Vice versa, the resistance value escalates when there is less water in soil. The soil moisture sensor then results output voltage according to the resistance value. Higher voltage means more water and lower output means less water.

Soil moisture sensors can be used in different purposes. In agriculture determining moisture in soil is crucial since it helps the irrigation system works more efficient. For climate researches, agricultural sciences or environmental studies, it play an important role in providing information to the scientists.

2.2.4 Light Sensor

Controlling light is an essential characteristic in smart system in nowadays technologies. As its name suggests, this device is used for sensing the light. Any device that transforms light energy into electrical energy is identified as a light sensor. In other words a light sensor is a component that converts light photons into electrons. Commonly there are three types of light sensors: Photo resistor, photodiode and phototransistor.

2.2.4.1 Photodiode

Photodiode (also known as photo detector or photo sensor) is a diode which changes light into an electric currents. Photodiodes are primarily created from germanium and silicon and also comprise of optical filter and lens. When a photodiode is illustrated by a beam of light, it loses some electrons which creates electron holes. As a result, it allows electrical current to flow through. Having more light means greater current running [10]. A photodiode is also responsive to infrared light.

Photodiodes are used for many purposes. It is a favorable light sensor for applications which require fast light response changes, since it generates currents that straightly proportional to the light intensity. A photodiode can be used in medical instruments for analysis or measuring purposes. It also operates in a remote control, compact disc player or a smoke detector.

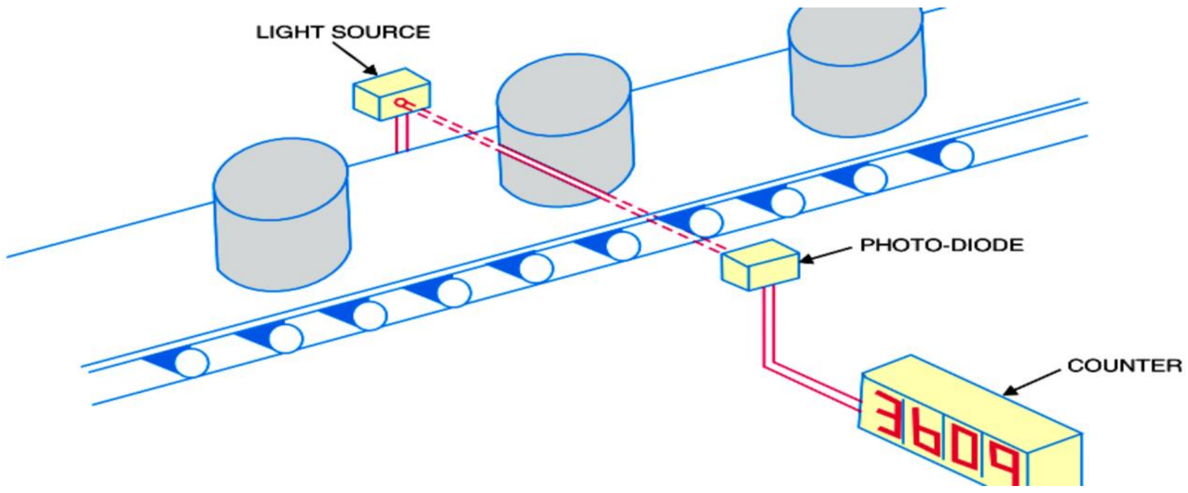


Figure 6. Product counter using photodiode. (Reprinted from [27])

One of the most common application of a photodiode as a products counter is demonstrated in figure 6. Each time a product blocks the light source, a photodiode generates a digital signal to counter, which then increase its number.

2.2.4.2 Photo Resistor (LDR)

Another type of light sensor is photo resistor, also known as light dependent resistor, photocell or photoconductor, is an active component whose resistance value nonlinearly increases or decreases respect to intensity of light fall upon it. LDR has lower sensitivity than a photodiode and photo transistor. When the light is absent, its resistance value rises significantly, sometimes up to 1M Ω. When it is exposed to light, the conductivity increases, result in the resistance drops considerably to few ohms. Its sensitivity diverges to the light applied [11].

There are two type of LDR, intrinsic and extrinsic. In intrinsic photo resistor, silicon or germanium is used as materials. When photons spread on its surface, they stimulate electrons to the conduction band from the valence band, therefore more free electrons resulting in current flow and less resistance. Extrinsic LDRs are usually made by material doped with impurities (dopants), above the existing valence band where they are added new energy band, inhabited by electrons. They need less energy to make the transition thanks to smaller energy gap. This results in the device sensitive to different wavelengths. Nevertheless, both type of LDR will increase their conductivity when illustrated. If the wavelength of a light source is outside its working range, the resistance will not be affected at all [11].

In the past silicon and germanium are researched to make photo resistor. In modern day cadmium sulfide is a high resistance semiconductor material which is used to produce LDR. Highly purified cadmium sulfide powder and still binding material are required to create photo resistor. In some countries cadmium LDR are banned due to environmental concern [11]. Figure 7 explains a structure and characteristic of a photo resistor.

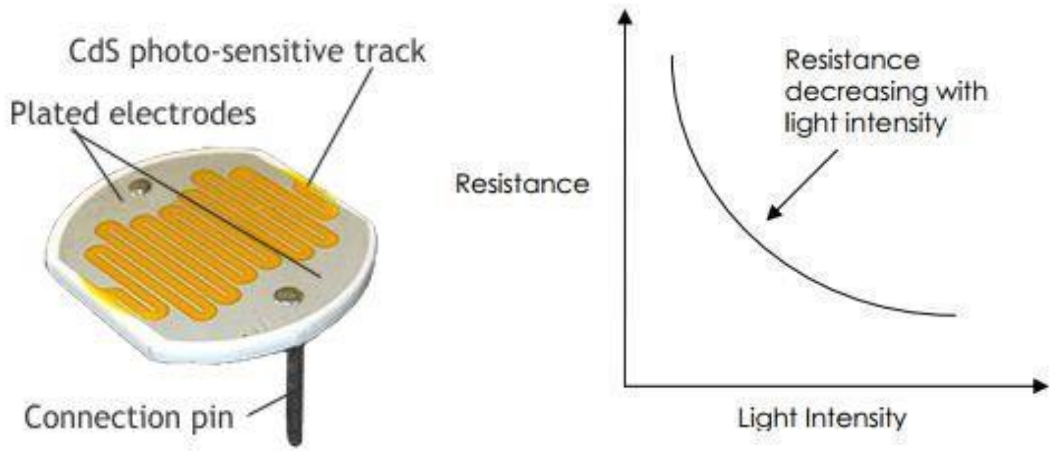

CdS photo-sensitive track
Plated electrodes
Connection pin
Resistance
Resistance decreasing with light intensity
Light Intensity

Figure 7. Structure and characteristic of a photo resistor. (Reprinted from [11])

Time latency between changes in illumination and changes in resistance is also a curious property of a LDR. Ten millisecond is usually a duration for a fully drop in resistance when light is applied after completely darkness. And after light is entirely removed, it can take up to one second for photo resistor to recover its starting value. Due to this reason, light dependent resistors are not used in rapid movement devices [11].

2.3 Communication Protocols

Communication links help embedded electronics in a wonderful ways to make it fully functional. A common protocol is required for circuits or system to exchange information. There are many protocols defined for communicating between devices, in general they can be divided into two main categories: parallel communication and serial communication. Using each requires different resources as they are implemented in various ways. Each one has its advantages and disadvantages.

2.3.1 Parallel Communication Protocol

Parallel communication interface is extremely simple. It transfers all the bits in data simultaneously. This method requires as many buses and ports as number of bits to be transferred. In term of resources, this is more expensive as it requires more wires and pins. However, it is extremely easy to implement. In figure 8 the implementation of parallel communication protocol is demonstrated.

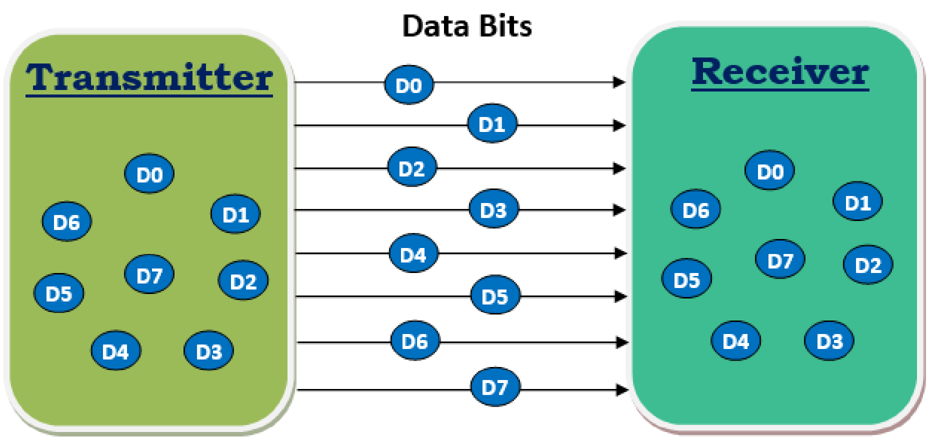


Figure 8. An implementation of parallel communication. (Reprinted from [28])

As every bit is transferred at the same time, parallel communication is fast and easy to read. This method is implemented in cases in which large data need to be transferred in real time quickly. Note that parallel communication is only used in short-distance data transferring. Since the wires are extremely close to each other and the signal strength diminishes as it goes further down the line (due to resistance of the wires), interfering is likely to happen when signal is weak enough. It usually causes crosstalk and attenuation on the transferred signal. [32]

2.3.2 Serial Communication Protocol

On the other hand, serial communication protocol has more complexity but resources are not as much of parallel communication. It transfers only one bit at a time, which results in increasing time for exchanging data. Serial communication variants can be divided into two main types: synchronous and asynchronous. Synchronous communication requires a common clock which determines the speed of transferring data. In contrast, asynchronous communication means that data can be exchanged without an external clock signal. The speed is agreed by baud rate beforehand.

There are rules or mechanisms that are required to make data transferring happen smoothly and error-free. Normally a serial communication interface needs data bits, synchronization bits, parity bits and baud rate. Baud rate states the speed of data sent over time. It usually has unit as bits-per-second (bps). Some baud rate are most commonly used are 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200 [12].

A structure of a data packet transferred by serial communication can be seen in figure 9. Start bit and stop bits play role as a synchronization bits.

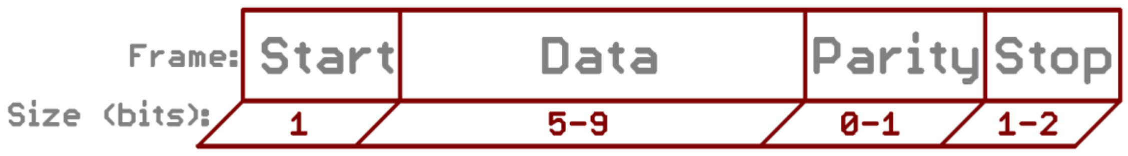


Figure 9. A serial communication packet frame. (Reprinted from [12])

There are various ways to use serial communication, but the simplest one is having two lines: transmit (TX) and receive (RX). The transmission can be simply described in figure 10

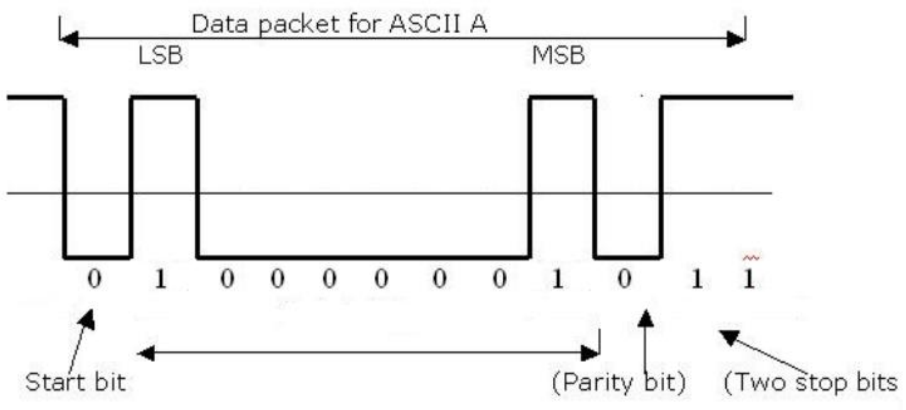


Figure 10. Transmission of a letter ‘A’ by using serial communication. (Reprinted from [12])

- Transmit idles high when there is no communication
- Start bit goes low for 1 bit
- Data bits are transmitted, least-significant bit goes first
- Parity bits (optional) using for error checking are sent
- Stop bits are sent

Serial communication can be implemented in circuits that do not require fast data transferring. Universal serial bus (USB), Ethernet, SPI and I^{2}C are using serial communication to exchange data. [12]

2.4 HTTP Requests

Internet is a massive dispersed information system. In order to handle the data conveniently between host and client, many protocols have been created, such as hypertext transfer protocol (HTTP), files transfer protocol (FTP), simple mail transfer protocol (SMTP) and etc.

HTTP is the most popular protocol used in the internet among several ones. It is the foundation of data interaction in the World Wide Web. HTTP is an asymmetric request-response client-server protocol. It is also stateless, meaning the current request is not aware of the earlier requests. [13] The relationship of HTTP requests and responses is illustrated in figure 11 below.

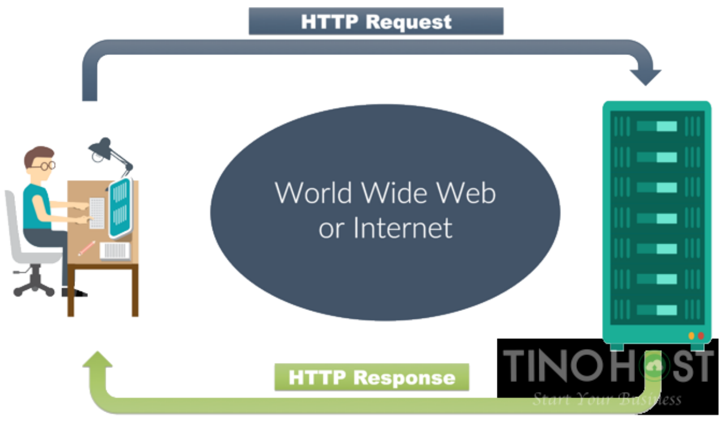


Figure 11. Hypertext transfer protocol. (Reprinted from [29])

HTTP request is an action where a URL performs action on an identified resource. It comprises a set of request methods to satisfy user's needs. Each of them is technically different semantic, however some common features may be shared. It works as a transportation between a client and a host. Client sends a request to the server where it is analyzed, and server answers it by a response. There are various types of HTTP request, which are GET, POST, HEAD, CONNET, TRACE, PATCH, OPTIONS and DELETE. The most commonly used are GET and POST method.

GET method is usually used for request data from a particular source. GET method request syntax is demonstrated in listing 1:

```
GET request-URI HTTP-version
(Optional request headers)
(Blank line)
(Optional request body)
```

Listing 1. A GET request syntax

A GET request can be sent by using URL as query string (name/ value). For example:

```
users.metropolia.fi/~theng/get_request.php?nimi=AnhNguyen&class=TXD16S1
```

In the example demonstrated above, there are two parameters. First one whose name is **nimi** has value "Anh Nguyen" and the variable whose name is **class** has value "TXD16S1".

GET request is cacheable, remains in browser history, can be bookmarked, have length restriction. It should never be used for sending sensitive data (such as password or pin code) [14].

POST request in other hand is designed to send loads of data to server (i.e. sending HTML form or uploading a picture). A POST request syntax in demonstrated in listing 2.

```
POST request-URI HTTP-version
Content-Type: mime-type
Content-Length: number-of-bytes
(Other optional request headers)

(URL-encoded query string) [13]
```

Listing 2. A POST request syntax

Unlike GET method, POST request cannot be triggered by using URL. It has a need of an HTML form where method = “post” or by specific program.

POST request is not cacheable nor bookmarked and does not remain in browser history. It has no data length limitation [14].

2.5 Google Chart

Google chart is a powerful, simple, free-to-use web service which is developed by Google. The application creates interactive graphical charts from data that are supplied by users. There is a variety of charts, for example line chart, bar chart, column chart, geographic chart, area chart, etc. Web developers are able to customize their own graph easily. The most common way to create a chart is by using JavaScript on a web page, listing data, selecting options and finally creating a div tag with a specific id. Charts are shown in HTML web page as JavaScript classes [15].

Figure 12 below demonstrates a line chart which is drawn by using Google Chart.

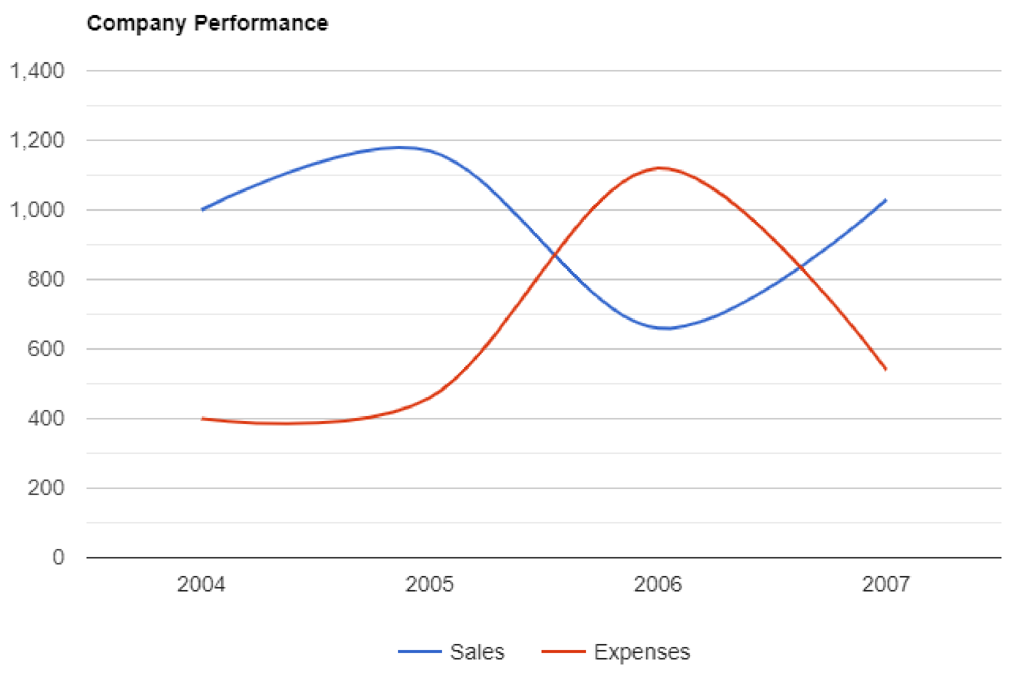


Figure 12. A line chart drawn by Google Chart. (Reprinted from [15])

Code generates this chart can be found in Appendix 1.

2.6 Internet of Things

Internet of things (sorted as IoT) is an innovative concept which is described as a system where objects, such as sensors, computers, smartphone and wearable devices, are connected through the internet by the same Internet Protocol (IP) [16]. It has an advanced ability that does not require human interaction to transfer data. IoT definition has evolved time by time because of the combination of many technologies, commodity sensors, machine learning and embedded system.

The idea that describe a system that connects everything by using internet has appeared in 1970s. In 1999 a pioneer whose name is Kevin Ashton has invented the term Internet of things while he was working in supply chain optimization in Procter & Gamble. However, not until ten years later, in summer 2010, the concept gains it popularity. From then it has continuously thrived in nowadays technologies and become the most famous concept of decade. It appears in various industries such as defense, healthcare, retailing, finance and etc. [16]

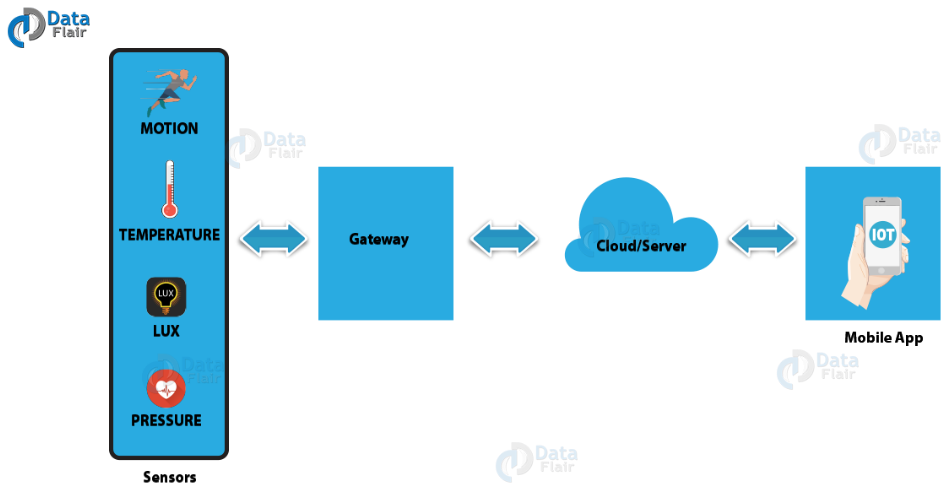


Figure 13. Architecture of a simple IoT setup. (Reprinted from [17])

Figure 13 shows an IoT system architecture applied in a health tracking system combined with a mobile application. There are basically four elemental components in an IoT structure, which are sensors, connectivity, data processing and user interface.

In general, an IoT system works as follow:

- First, sensors or devices collects data from surrounding environment in real-time. Data collected has all the complexity, from temperature to GPS.
- Next, the data harvested will be transferred to the storage. In order to do so, a medium is required. Information can be sent to the cloud by a variety of communication protocols, such as wireless communication, satellite, Bluetooth, and etc.
- Then, data will be stored in cloud storage. A specific backend software processes the information. This range from logging number to analyzing object by computer vision.
- Finally, the information is available for end-user by user interface visualization. Users are also able to interact with these data.

3 Components

Figure 14 below illustrates components which are included in this project.

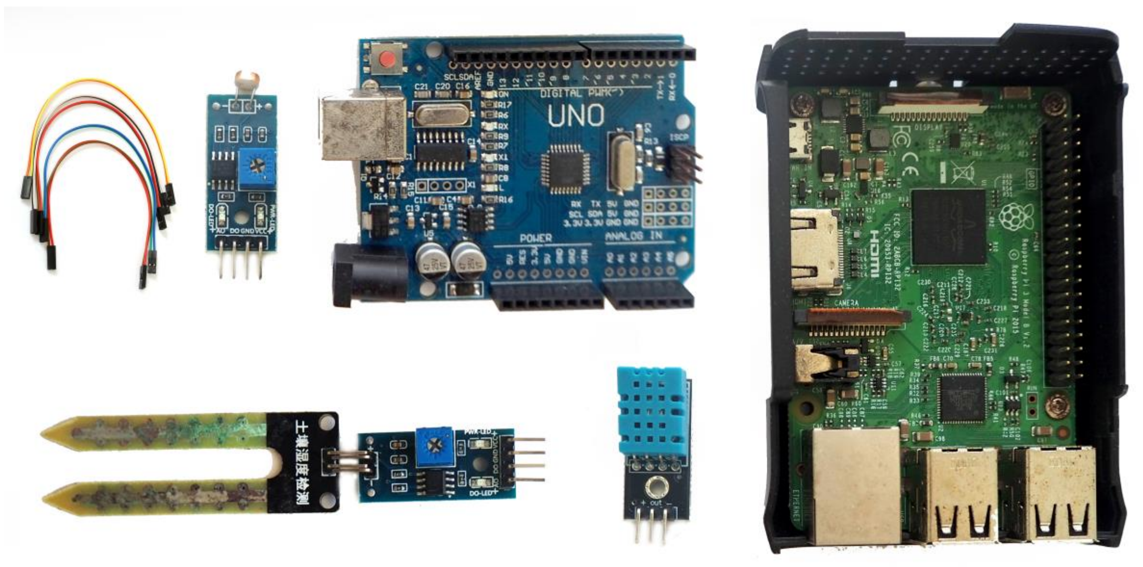


Figure 14. Hardware components used for ACED device. *In clockwise order from top left: jumper wires, light sensor LDR, Arduino Uno R3, Raspberry Pi 3B+, DHT11 sensor, soil moisture sensor*

The main goal of this project was to create a small setup based on IoT concept. The data will be sent through wireless communication to a cloud server. Choosing appropriate sensors and microcontroller boards is crucial in order to make the setup run functionally.

3.1 Sensors Unit

Environmental factors such as temperature, humidity, soil moisture and luminosity are selected for monitoring. There are three sensor modules included in this project. DHT11 works as a temperature and humidity sensor (two-in-one), where soil moisture is measured by module FC-28, and light intensity is logged by using a photo resistor module. Note that sensors that result measured output as analogue signal will be read in range from 0 to 1023 on the Arduino boards. In order to have a well presented data visualization, the analogue values are scaled into range from 0 to 102. This is done simply by having analogue value divided by 10. Therefore when data are displayed on charts, curves of temperature and humidity are clearly shown altogether with scaled value of light intensity and soil moisture.

Most of the sensors and microcontrollers included in this project is low-priced and simple to implement. There are many components can be used in order to increase the measurement accuracy.

3.1.1 Temperature and Humidity Sensor DHT11

DHT11 temperature and humidity is one of the most frequently implemented sensor in electronics world. DHT11 is created by two parts, a thermistor for reading temperature and a capacitive sensor for sensing humidity. It also has a basic analogue to digital converter which creates simplification for any microcontroller to read its values. This sensor is plain and slow but inexpensive and easy to use. It can be applied in simple logging temperature and humidity projects. Figure 15 demonstrates a DHT11 temperature and humidity. Note that DHT11 is only the blue part which has 4 pins, while the whole module will have 3 pins and a 5kΩ pull-up resistor embedded on it.

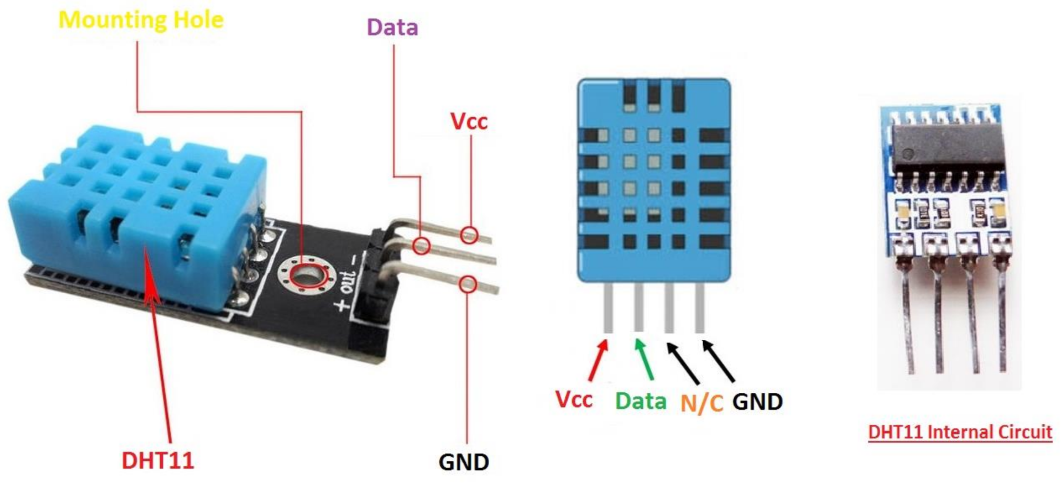


Figure 15. DHT11 temperature and humidity sensor. (Reprinted from [30])

Typically a DHT11 sensor operates at 3V to 5V power supply, a maximum 2.5mA current during conversion. Temperature functions stably in range from 0°C to 50°C with ±2°C error. It gives best result for humidity in range of 20% to 90% with 5% accuracy. DHT11 activates with sampling rate at 1Hz, meaning each sample to be read at every 1 second [18]. The pins connection for DHT11 is shown in table 1 below.

Table 1. Pin configuration of a DHT11 sensor

Pin No.	**Pin name**	**Description**
1	Vcc	Power supply for sensor, in range from 3V to 5V
2	Data	Temperature and humidity serial data output
3	N/C	Not connected
4	GND	Connects to the common ground

In this project DHT11 sensor is connected to 5V power supply. DHT11 sensor module is already calibrated. In figure 16 connection diagram of the module can be seen. Note that when connection is less than 20 meters, a 5kΩ pull-up resistor is connected. For connection greater than 20 meters, uses appropriate pull-up resistor instead [18].

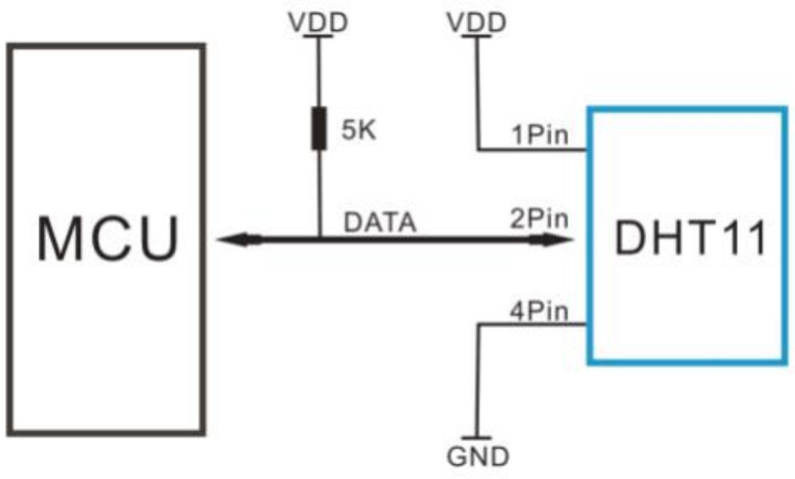


Figure 16. Connection diagram of DHT11 temperature and humidity sensor. (Reprinted from [18])

Temperature and humidity information can be recorded easily by using a microcontroller which has digital input pin. DHT11 uses serial communication technique to transfer data, where single-bus data is used. It requires only one data bus to connect between devices. The length of a transmission is 40 bits and the sensor is able to send both decimal and integral part. It has format as 8-bit integral relative humidity data + 8-bit decimal relative humidity data + 8-bit integral temperature data + 8-bit decimal temperature data + 8-bit check sum [18]. All the conversions from raw electrical value into temperature and humidity are done inside DHT11. It gives temperature and humidity information straight forward. Data transmission process is illustrated on figure 17.

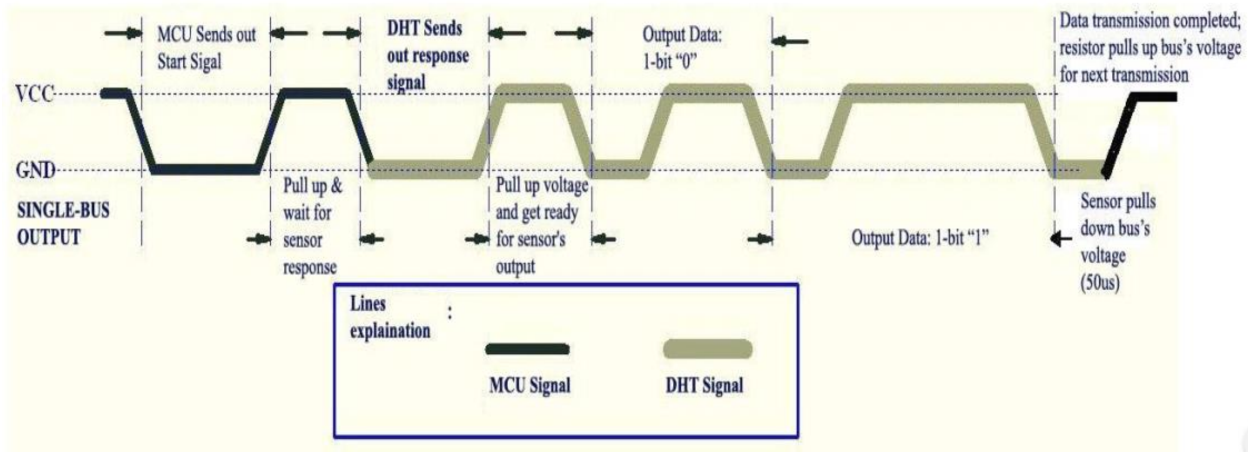

MCU Sends out Start Sigal
DHT Sends out response signal
Output Data: 1-bit "0"
Data transmission completed; resistor pulls up bus's voltage for next transmission
VCC
GND
SINGLE-BUS OUTPUT
Pull up & wait for sensor response
Pull up voltage and get ready for sensor's output
Output Data: 1-bit "1"
Sensor pulls down bus's voltage (50us)
Lines explaination :
MCU Signal
DHT Signal

Figure 17. Data timing diagram. (Reprinted from [18])

For more accuracy and faster sampling rate, use DHT22 instead. It has smaller error rate and faster reading frequency.

3.1.2 Soil Moisture Sensor FC-28

FC-28 is a soil moisture sensor module that is used for measuring soil moisture and similar material. It usually has two parts, a two long probes operates as a variable resistor and a data processing unit. This sensor is very straight forward to use. The probes will be pushed into soil to measure the amount of water. Depends on the water volumetric, an electrical current will be formed with a specific resistance value which determines the soil moisture. If there is more water in the soil, more current conducted, hence lower resistance value and vice versa.

Soil moisture sensor module FC-28 has some essential features, such as a digital output with a threshold adjustable by using a potentiometer, both analogue and digital input and LED indicators. It functions with power supply in range from 3.3V to 5V (3.3V is used in this project). There are four pins in the data processing part, Vcc (power supply), GND (ground), A0 (analogue pin) and D0 (digital pin). Figure 18 shows a FC-28 soil moisture sensor module.

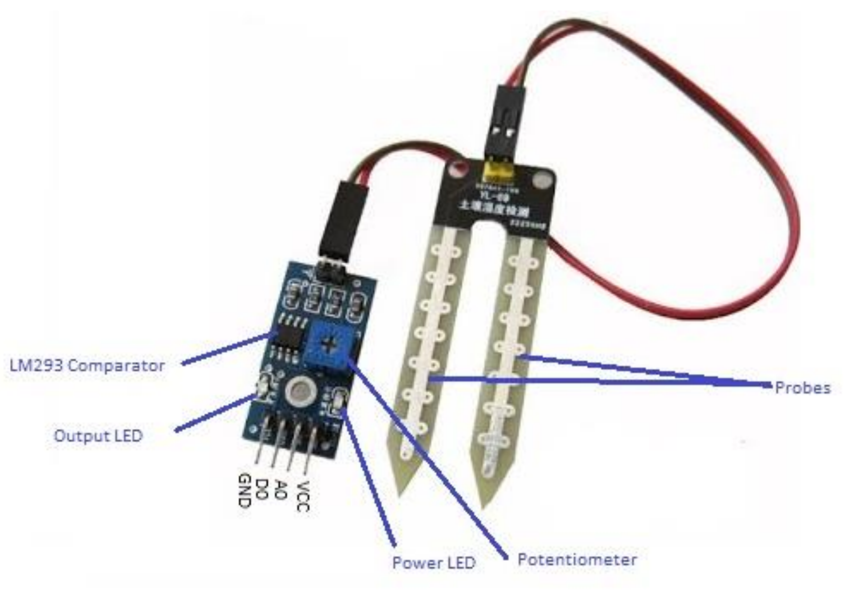


Figure 18. A soil moisture sensor module FC-28. (Reprinted from [19]).

FC-28 module also can work in digital output mode or analogue output mode. If the sensor is used in digital mode, the LM393 comparator will compare the analogue output and the threshold value adjusted by potentiometer in order to result digital output. If the sensor output value is greater than threshold value, a HIGH digital signal (usually 5V) will be given into D0 pin and the LED will light up [19].

In analogue interface, the percentage value of soil moisture is given. Its value ranges from 0 to 1023 [19]. The mapping value of the sensor is shown in table 2 below.

Table 2. Mapping value for FC-28 soil moisture sensor module

Value range	Soil status
More than 1000	The sensor is not in soil of disconnected

From 600 to 1000	The soil is dry
From 370 to 600	The soil is humid
Less than 370	The sensor is in water

The analogue pin is used in this project. In order to create visualization tool appearance be more convenient, the value will be logged in reversed way and scaled into range 0-102 (as explained earlier in section 3.1). This means that more water in soil creates greater value and vice versa. When it is completely in water, its value will be 102. On the other hand, if the soil is very dry the value will be very small.

3.1.3 Photo Resistor Sensor LDR

Light intensity is measured by LDR photo resistor module in this project. It is extremely easy to use with any microcontroller. In this module there are both digital interface and analogue interface as in FC-28 soil moisture sensor. When there more light intensity, the LDR conducts more current, hence the resistance value decrease. In contrast, if there is no light at all, the resistance value increases due to lack of conductivity. For more information of how a photo resistor works, see section 2.2.4.2 Photo resistor (LDR) above. A photo resistor sensor module is demonstrated in figure 19. A typical LDR sensor module should have four pins, Vcc (power supply), GND (ground), D0 (digital pin) and A0 (analogue pin).

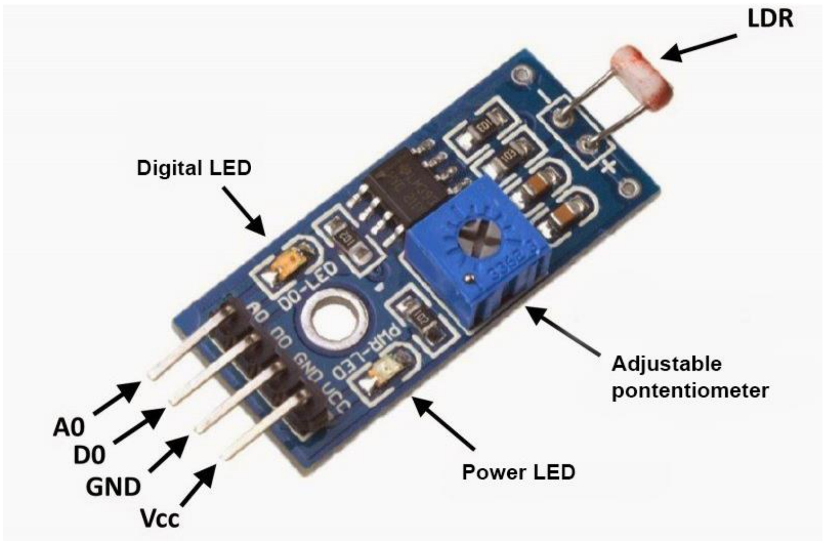


Figure 19. A photo resistor sensor module using LDR

In digital mode, value of LDR output and a pre-adjusted potentiometer are compared by a comparator LM393, which then results in a digital signal. If the LDR output is greater, digital pin D0 will have value HIGH (usually 5V) and the digital LED lights up. Vice versa, the digital D0 outputs a LOW signal and the LED indicator turns off. In this project a 5V Vcc is connected to power the light sensors.

In analogue interface, value of LDR output goes straight forward to analogue pin A0. The value is mapped in range from 0 to 1023 if logged by a microcontroller. The analogue pin is taken into account in for measure luminosity in this project. For more eye-catching visualization, value from analogue pin A0 is read reversely and scaled into range 0-102 (as explained in section 3.1). When there is a lot of light intensity, data will be logged as 102. Vice versa, value will be 0 when there is completely no light.

3.2 Microcontrollers

Microcomputer boards were chosen to perform in this project are Arduino Uno 3 and Raspberry Pi 3. Arduino Uno 3 has a very important ability which is analogue reading. It helps the device reads analog value from sensors, which a Raspberry Pi microcontroller is unable to do. Raspberry Pi 3 on the other hand has Wi-Fi connectivity, which supports a lot for communicating to the cloud.

3.2.1 Arduino Board

Arduino is an open-source teaching platform based on easy-to-use hardware and software which intends to improve electronics teaching in universities. It has been developed by Arduino.cc since 2005. The products are under GNU Lesser General Public License which gives anyone the right to manufacture an Arduino board. It is an inexpensive microcontroller which has features such as cross-platform, open source hardware and software, simple and clear programming environment. [20]

There are many available variants of Arduino. Arduino Uno is the most popular one in the world. There are also Arduino Nano, Arduino Mega, Arduino Leonardo, Arduino micro and etc. In this project Arduino Uno is selected since it has a wonderful feature that is analogue reading by using serial communication technique. It operates as a data reader which helps collecting data from sensors.

Arduino Uno is a microcontroller based on high-performance ATmega328p chip. Fourteen pins are dedicated for digital input/output (D0 to D13), in which 6 pins (D3, D5, D6, D9, D10 and D11) can operate as pulse width modulation pins with 8-bit resolution. There are also 6 analogue input/ output pins. Arduino Uno has USB connection for uploading code or/ and powering the board. It can also operates by an external battery (from 5V to 12V) through a power jack. In figure 20 the pin-out map of Arduino Uno R3 is illustrated.

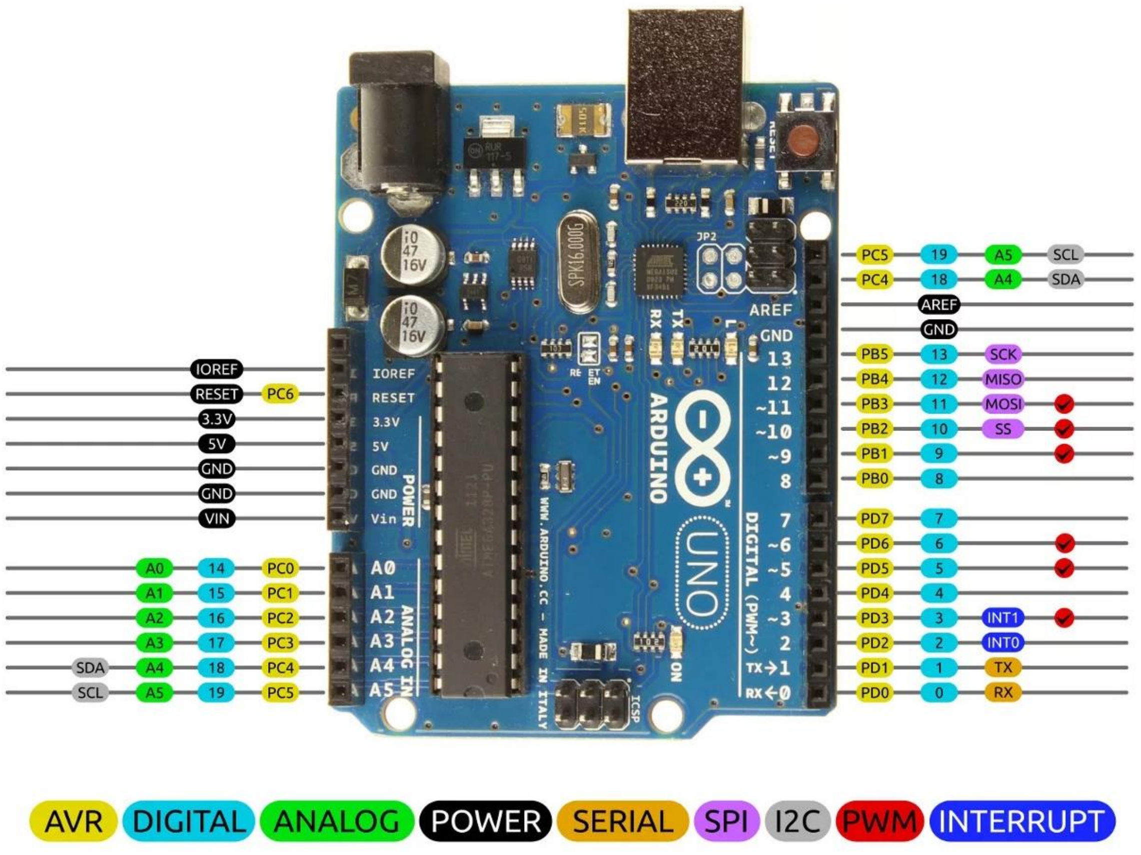


Figure 20. Arduino Uno pin-out. (Reprinted from [34])

Arduino Uno also has several features such as interrupts, counters, timers, built-in LED indicator for pin D13 and 16 MHz quartz crystal clock. A reset button is attached into the board for resetting the running program to initial stage. Serial UART communication can be set up by connecting pin RX (D0) and TX (D1).

Technical specification of an Arduino Uno can be found in table 3 below.

Table 3. Technical specification of Arduino Uno (Reprinted from [21])

Microcontroller	ATmega328p 8-bit AVR family microcontroller
Working voltage	5V
Voltage input (recommended)	7V to 12V (limited to 20V)
Digital pins	14 pins (D0 to D13), out of which 6 pins provide PWM
Analogue pin	6 pins (A0 to A5)
DC current on I/O pins	40mA
DC current on 3.3V pins	50mA
Flash memory	32kB of with 0.5kB used for boot loader
Clock speed	16MHz
Dimension (length x width)	68.6 mm x 53.4 mm

Arduino IDE (integrated development environment) is a free software which is made for coding and uploading it to any Arduino board. It is compatible with both Windows OS, Linux OS and Mac OS. The main programming language is used for coding is C/C++.

3.2.2 Raspberry Pi 3 Board

The general idea of developing a Raspberry Pi was for teaching basics of computer science in developing countries. As a result, a small-size single-board computer was created in the United Kingdom by the Raspberry Pi Foundation. It becomes more and more popular over-time.

Raspberry Pi is a low-cost computer that operates in Linux operating system. It also comprises of a set of GPIO (general purpose input/ output) that interacts with real world through sensors and controls electronics components such as LED. There have been many models of Raspberry Pi computers manufactured, from Raspberry Pi 1 released in 2012 to the newest Raspberry Pi 4B model in 2019. Raspberry Pi 3 model B+ has been chosen to perform in this project. It works as a data collector which harvests data from data reader (Arduino Uno).

Raspberry Pi 3 model B+ was released in 2016. The specifications are shown as below:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1 GB of RAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN connection
- Low energy Bluetooth 4.2
- Gigabit Ethernet over USB 2.0 (maximum 300Mbps)
- Extended 40 GPIO pins (see figure 21)
- Four USB 2.0 ports
- A full-size HDMI connection
- CSI camera port for a Raspberry Pi camera
- DSI display port for connecting Raspberry Pi touchscreen
- Micro SD port for loading operating system
- 5V and 2.5V DC power input [23]

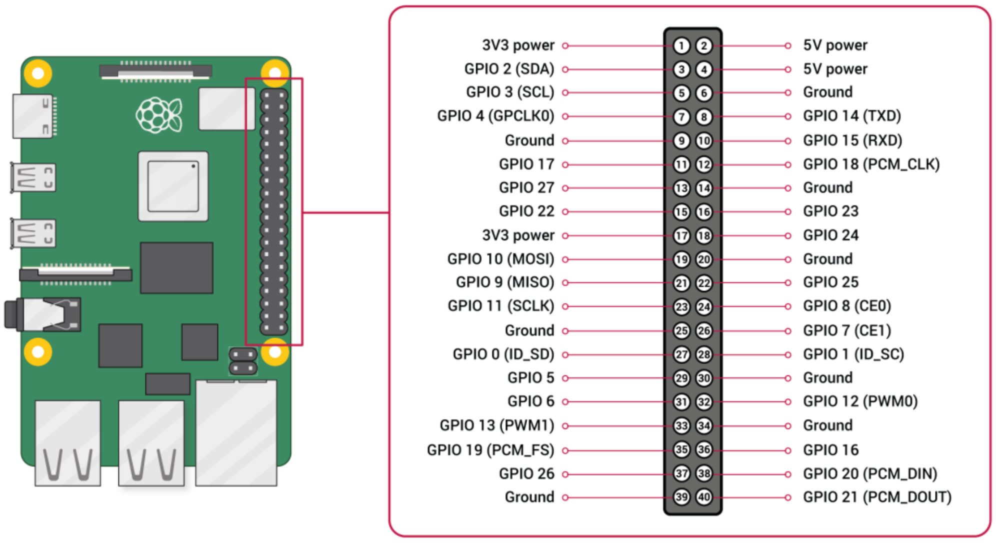


Figure 21. Pin out of Raspberry Pi 3 model B. (Reprinted from [31])

Raspberry Pi hardware operates on an open source and optimized operating system which is named Raspbian which based on Debian of Linux OS. It contains over 35000 packages and many pre-compiled software [22]. The GPIO pins and serial communication can be controlled by Raspbian. Python is the main programming language used in Raspberry Pi and in this project.

4 Design

This project aim was to create a small setup based on internet of things concept. Hence, four fundamental components, which are sensors, connectivity, data processing and user interface, were designed carefully.

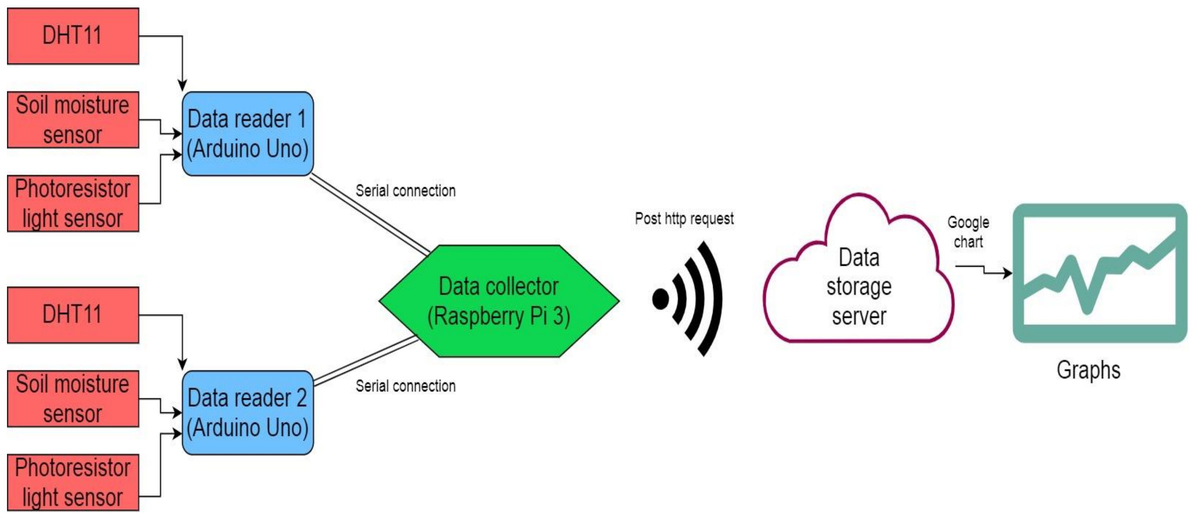


Figure 22. Working diagram of the project. (Drawn by using draw.io)

The basic idea of the setup is shown in figure 22. This project has Arduino Uno works as a data reader which collects information from sensors (DHT11, soil moisture sensor and light sensor). There are two data readers perform in two different environment. There is also one data collector made from Raspberry Pi 3 model 3+ which harvests data from data readers and sends them to data storage in the internet. Wireless LAN is used to connect the data collector with the internet. A HTTP POST request which contains information collected from data collector will be sent every 10 minutes to the cloud. A simple PHP program in the data storage will process the data and log it in files. Finally a web page application will draw graph based on stored data for a user interface experience.

Details of hardware connection, data transferring and creating graphs are explained in following sections. Note that for simplification, only one data reader will be introduced. A Raspberry Pi 3 can have up to four data readers connecting to it. Each one has a unique device ID.

4.1 Wiring

There are two departments that using wire to link to each other. Sensors are connected to data reader by using jumper wires. On the other hand, thanks to USB ports on both Arduino Uno (type B) and Raspberry Pi 3 B+ (type A), the interconnection between these two becomes much simpler. A USB cable connecting Raspberry Pi and Arduino forms a virtual serial communication between them.

Wiring map between data reader (Arduino Uno) and sensors are demonstrated in table 4 below.

Table 4. Wiring map of Arduino and sensors

Sensors	**DHT11**			**Light sensor LDR**			**Soil moisture sensor**		
	(+)	OUT	(-)	Vcc	GND	A0	Vcc	GND	A0
Arduino	5V	D5	GND	D11	GND	A0	3.3V	GND	A1

Figure 23 below shows the interconnection of the sensors to Arduino Uno.

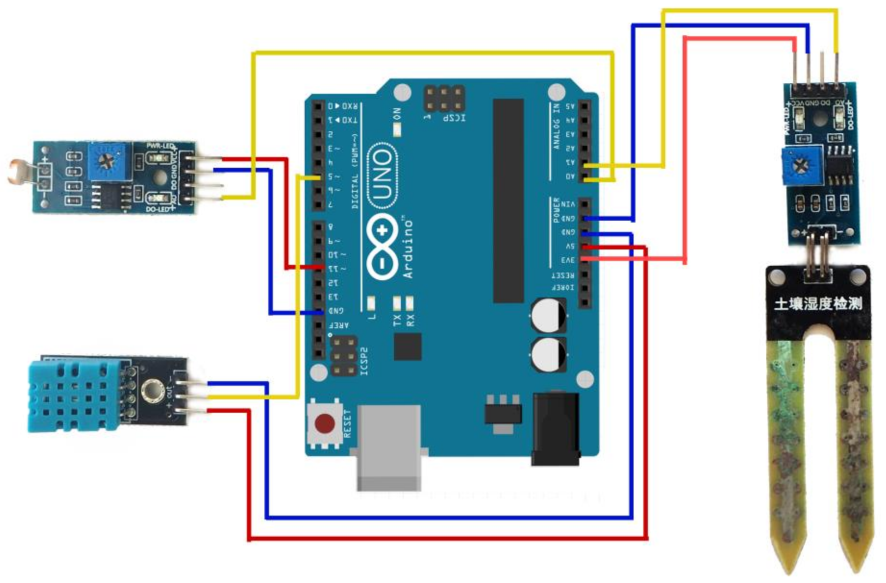


Figure 23. Wiring diagram of data reader (Arduino Uno) to sensors.

DHT11 temperature and humidity sensor is connected to 5V power supply and to common ground of Arduino Uno. Since DHT11 sensor transfers data by using serial communication, signal output pin from DHT11 is connected to a digital pin (D5) of Arduino. A special library whose name is “DHT” can be used to read and convert output data of DHT11 into human understandable format, Celsius degree (°C) and humidity percentage (%). This library can be included simply by implementing as demonstrated in listing 3

```
#include “DHT.h”
```

Listing 3. Including library used for DHT11. (C++ language)

For using this library for reading and converting, an object must be created by using following line which is demonstrated in listing 4.

```
dht <name_of_DHT_object>; //For example: dht DHT_sensor;
```

Listing 4. Creating object for DHT11. (C++ language)

For reading and converting temperature and humidity, following code lines is used respectively. The code lines are shown listing 5

```
DHT_sensor.read11(<DHT11_signal_pin>);    //Reading data for DHT11

DHT_sensor.temperature;    //Return temperature value in degree Celsius with
two floating number

DHT_sensor.humidity;       //Return humidity value in percentage with two
floating number
//See full code for Arduino in Appendix 2
```

Listing 5. Reading and converting temperature and humidity. (C++ language)

In this project, light sensor LDR output data is read and stored as analogue raw format. Pin GND in the sensor connects to pin GND of Arduino Uno as common ground. Pin A0 of the sensor is wired to analogue pin A0 of Arduino. Due to lack of power supply pin in Arduino Uno (there are only one 5V pin and one 3.3V pin, which have been dedicated for DHT11 and soil moisture sensor), a digital input/ output pin was manipulated to work as a power supply pin. Pin D11 has been chosen for this purpose. It will be set to HIGH signal whenever sensor needs power to operate. HIGH signal output from digital pin in Arduino Uno usually has 5V output voltage and maximum 40mA output current [24] which is suitable for this light sensor that requires 5V power supply and 15mA current [33].

Similarly, soil moisture sensor is connected to Arduino as shown in table 4 above. Since comparator in this sensor operates at 3.3V power supply, sensor has to be wired to 3.3V power supply in Arduino Uno. Analogue data is also logged in this project, therefore analogue pin out A0 is linked to pin A1 in Arduino. Because soil moisture sensor is processed using analogue to digital converter, soil moisture percentage can be calculated as follow [25]

$$AnalogOutput = \frac{ADCValue}{1023} \tag{2}$$

$$Moisture\ (in\ \%) = 100 - (AnalogOutput * 100) \tag{3}$$

This project used two data readers with two set of sensors units. The other set also connects as explained earlier.

4.2 Transferring Data

4.2.1 Communicating between Arduino and Raspberry Pi

Serial communication UART manages the transferring data between Arduino and Raspberry Pi. Asynchronous serial communication is operated in baud rate 9600. These two devices join to each other by using a typical USB type-B connection. Arduino is also powered by Raspberry Pi through this USB connection.

The working principal between Raspberry Pi and Arduino Uno as follow:

- Raspberry Pi will send to Arduino Uno a “collecting request” (character ‘1’, ASCII code 49)
- Arduino Uno reads and converts data into a string
- Arduino returns response answer back to Raspberry Pi

4.2.1.1 On Data Reader Arduino

When serial interface is available, Arduino Uno will check if a “collecting request” is sent from Raspberry Pi by following code lines which are demonstrated in listing 6

```
if (Serial.available())
     char request = Serial.read();   //check request on serial interface
     if (request == ‘1’)             //if collecting request (‘1’) is sent
          //{<process request here>}
```

Listing 6. Checking for a collecting request. (C++ language)

After receiving a request, Arduino Uno will start processing. Data will be read and converted into understandable format. A string data, which contains humidity value, temperature value, soil moisture value, luminosity value and device ID, will be formed by following code line which is shown in listing 7.

```
sprint(str, “{\"humid\":\"%d\",\"tempe\":\"%d\",\"moist\":\"%d\",\"light\":\"
%d\",\"ID\":\"%d\"}”, <humidity_value>, <temperature_value>,
<soil_moisture_value>, <light_intensity_value>, <device_ID>);
```

Listing 7. Forming a response string to answer for collecting request. (C++ language)

For example, a string line of data to be sent is similar to following line. The data line is illustrated in listing 8:

```
{“humid”:”70”, “temp”:”29”, “moist”:”65”, “light”:”76”, “ID”:”1”}
```

Listing 8. Example of a string data line.

(Humidity = 70%, temperature = 29°C, soil moisture = 65%, luminosity = 76 and device ID number = 1)

In order to establish a serial communication and send data to serial interface, runs the following code line respectively. The code lines are demonstrated in listing 9

```
Serial.begin(9600);        //Start the serial communication at baud rate 9600
Serial.println(str);       //Print the string data str to serial interface
```

Listing 9. Establishing a serial communication and sending data in Arduino. (C++ language)

To see full code of data reader (Arduino Uno), check Appendix 2

4.2.1.2 On Data Collector Raspberry Pi

Firstly, a serial communication establishment is needed. The code lines are shown in listing 10.

```
import serial
<serial_interface_name> = serial.Serial('/dev/tty<USB_port_ID>', 9600)
     //Establish serial connection
```

Listing 10. Starting the serial interface with baud rate 9600. (Python script)

Then, Raspberry Pi will sent a collecting request to Arduino using Serial interface. After that it collects data which has been sent from Arduino. The code lines which are illustrated in listing 11 shows how to do it.

```
<serial_interface_name>.write('1')   //send '1' to serial interface
<data_string> = <serial_interface_name>.readline()
```

Listing 11. Sending request and receiving data on Raspberry Pi. (Python script)

4.2.2 Transferring Data from Raspberry Pi to Server

Data transferring to storage server is done by a simple HTTP request with POST method. Data that has been sent from Arduino has format as a dictionary in Python programming language. It then will be converted into JSON format. This can be done by the following code lines which are demonstrated in listing 12

```
import json     #import library that helps converting a dictionary to JSON
<json_data> = json.loads(<dictionary_data_string>)   #converting to JSON
```

Listing 12. Converting data to JSON format. (Python script)

Data after being converted to JSON will be transferred to storage server by POST request. Listing 13 shows the code lines for sending data to the server

```
import requests      #import library requests to send data as http requests
requests.post(<URL>, <json_data>)    #send data to URL
```

Listing 13. Sending data to server from Raspberry Pi. (Python script)

The URL used in this project are

```
URL = 'http://users.metropolia.fi/~theng/thesis/get.php'
```

To see full code that runs on Raspberry Pi, check Appendix 3.

On storage server side, a receiving data application which is written in PHP will log data into files. Each device ID will have its own directory, which has format as *data<device_ID>* (i.e. data1, data2, etc.). Data will be saved daily in separated files. Each file represents one day and has name format as *<yyyymmdd>.txt* (i.e. 20200327.txt). Data in each file are saved in separate line for each request received. Data is logged in CSV format as following syntax:

```
'<time_stamp>',<humidity_value>,<temperature_value>,<soil_moisture_value>,<lum
inosity_value>
```

For example, data of device 1 which has humidity = 60%, temperature = 29, soil moisture = 50% and light intensity = 90, is logged in March 14th 2020 at 16:15 will be saved in directory */data1/20200314.txt* in a line *'16:15',60,29,50,90.*

A demonstration of how data is stored is shown in listing 14. This is data from device ID 1 on March 25th 2020, from 08:04 to 09:25.

```
'08:04',68,30,46,96
'08:14',69,30,46,96
'08:24',68,30,47,96
'08:34',68,30,52,96
'08:44',67,31,52,96
'08:55',68,30,46,96
'09:05',67,30,46,96
'09:15',68,30,46,97
'09:25',67,31,49,97
```

Listing 14. Stored data demonstration. *Values separated by a comma, from left to right: timestamp, humidity, temperature, soil moisture and light intensity.*

Appendix 4 contains code lines for logging data received.

4.3 Drawing Charts

Charts are drawn by using Google Charts. A chart is considered as a JavaScript class in a HTML page. In order to create a chart, the application must include script from Google Charts source. The following code line in listing 15 demonstrates how to do it.

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js">
```

Listing 15. Including library for drawing chart by using Google Charts service. (HTML script)

Then a division with a unique ID must be created. In listing 16 code lines are shown.

```
<div id="chart" style="width:100%; height:500px">
<!—chart will have width as 100% of the screen, and 500 pixels hight -->
```

Listing 16. Creating a division for chart in webpage. (HTML script)

Then a JavaScript runs as below to create chart. A PHP code inside JavaScript returns data which has been logged in files. Listing 17 shows the code lines

```
google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Time', 'Humidity', 'Temperature', 'Moisture', 'Luminosity'],
     //Insert PHP code in here (see code below)

     // Data has format of ['<time_stap>', <humidity_value>,
<temperature_value>, <soil_moisture_value>, <luminosity_value>],
     // For example,
     // '09:35',69,31,89,96
     // '09:45',70,30,89,96
```

Listing 17. Importing data to webpage script. (JavaScript script)

In order to draw a chart, application needs device ID and date. This can be obtained by sending a GET request to page application. PHP code lines generating data are shown in listing 18:

```
<?php
     $id = $_GET["ID"];    //gets ID from GET request
     $date = date("Ymd", strtotime($_GET["date"])); //gets date and converts
it to yyyymmdd format
     $f = fopen("./data".$id."/".$date.".txt", "r");
     while(!feof($f)) {
          $line = fgets($f);
          if ($line != "")
               echo "[".$line."],";
     }
?>
```

Listing 18. Generating data from storage. (PHP script)

4.4 Web Page Application

The web application create a user interface for visualization. This is the most important part in designing a visualization tool. There are two pages,

- index.php: where query information will be sent to draw a graph (see figure 24)
- graph.php: this page draws a graph by using Google Charts as explain in section 4.3 (see figure 25)

Figure 24 and figure 25 below show user interface of the visualization tool of this project.

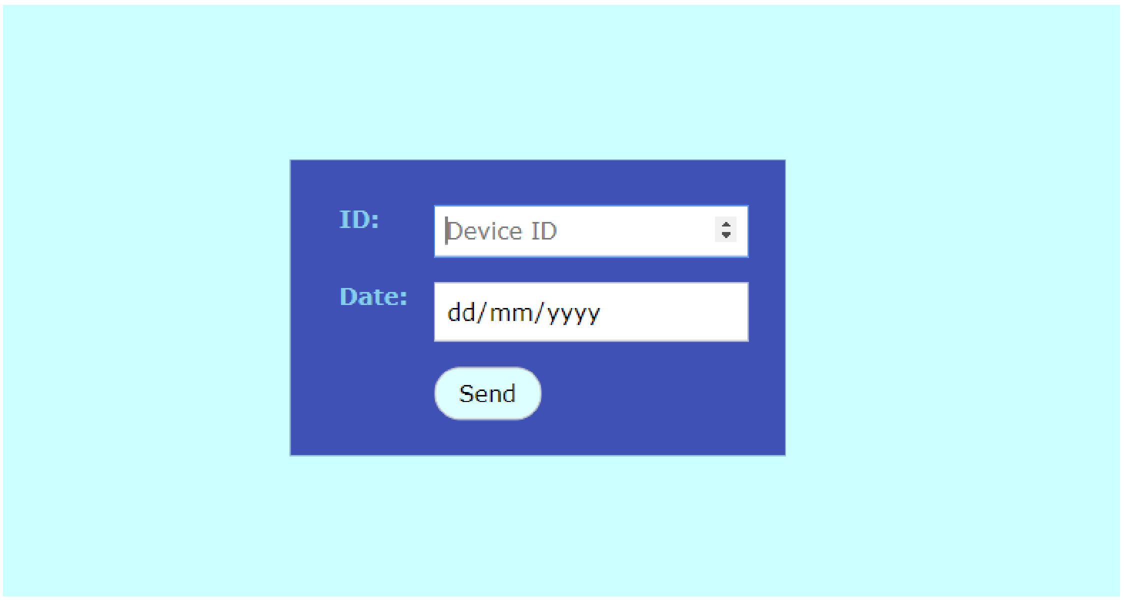


Figure 24. User interface of web page where query can be sent to drawn chart.

When user gives invalid information, such as non-existing ID number, wrong date format or date that does not have logged data, the web page application will show an error message saying:

"Your input is invalid or date was not logged. Please try again"

Otherwise, it will lead user to another web page where a graph is drawn with given information.

Chart of device ID1 that shows information in March 24th 2020 is demonstrated as in figure 25. Typing ‘1’ in ID field, and ‘24/03/2020’ (without quotes) in Date field in the previous interface gives the result.

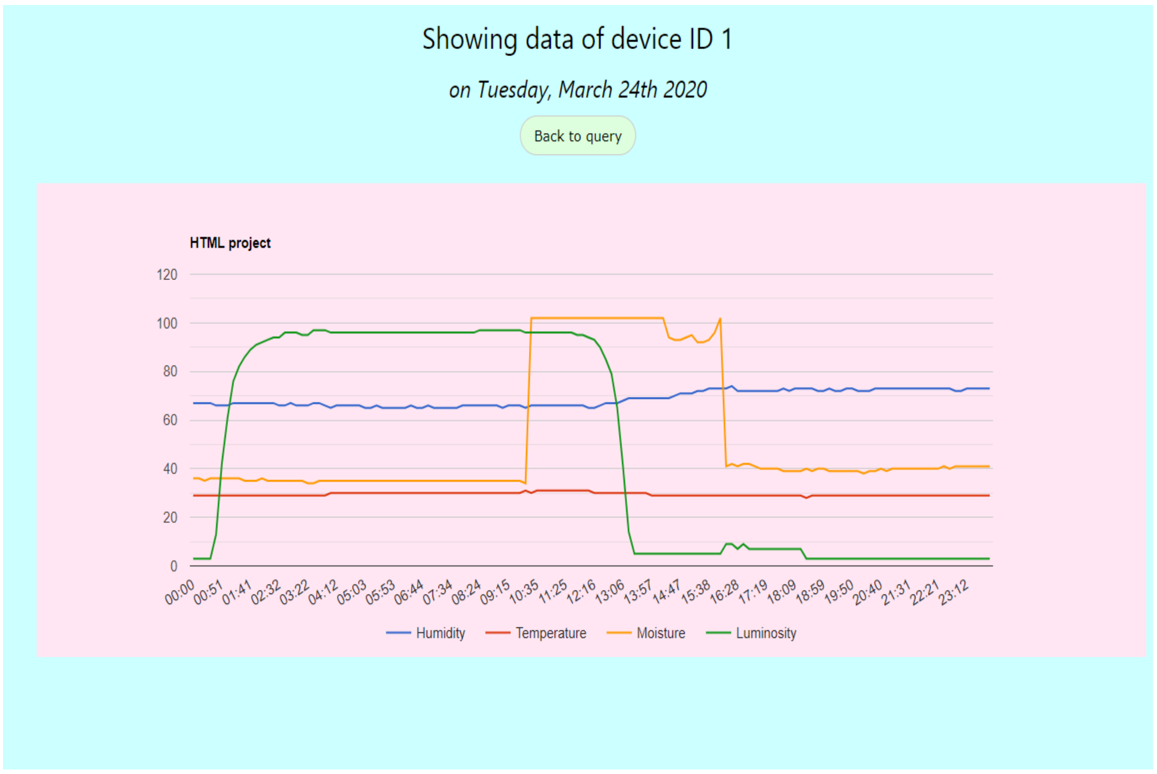


Figure 25. A chart drawn for device ID1 on Tuesday, March 24th 2020.

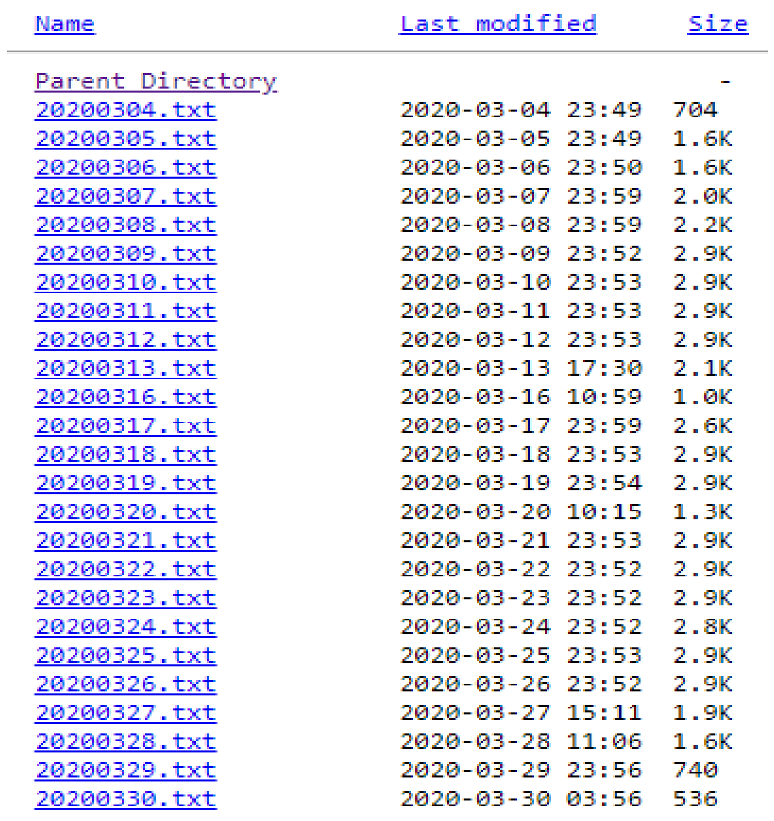
To see full code of web page application, check:

- Appendix 5 which contains code lines for query web page, and
- Appendix 6 which contains code line for graph drawing web page.

5 Performance and Development

5.1 Performance

By the time of this was written (on March 30th 2020), data has been logged for 24 days almost continuously. Figure 26 below demonstrated how much data is stored.



```
Name                     Last modified      Size
------------------------------------------------
Parent Directory                             -
20200304.txt             2020-03-04 23:49   704
20200305.txt             2020-03-05 23:49   1.6K
20200306.txt             2020-03-06 23:50   1.6K
20200307.txt             2020-03-07 23:59   2.0K
20200308.txt             2020-03-08 23:59   2.2K
20200309.txt             2020-03-09 23:52   2.9K
20200310.txt             2020-03-10 23:53   2.9K
20200311.txt             2020-03-11 23:53   2.9K
20200312.txt             2020-03-12 23:53   2.9K
20200313.txt             2020-03-13 17:30   2.1K
20200316.txt             2020-03-16 10:59   1.0K
20200317.txt             2020-03-17 23:59   2.6K
20200318.txt             2020-03-18 23:53   2.9K
20200319.txt             2020-03-19 23:54   2.9K
20200320.txt             2020-03-20 10:15   1.3K
20200321.txt             2020-03-21 23:53   2.9K
20200322.txt             2020-03-22 23:52   2.9K
20200323.txt             2020-03-23 23:52   2.9K
20200324.txt             2020-03-24 23:52   2.8K
20200325.txt             2020-03-25 23:53   2.9K
20200326.txt             2020-03-26 23:52   2.9K
20200327.txt             2020-03-27 15:11   1.9K
20200328.txt             2020-03-28 11:06   1.6K
20200329.txt             2020-03-29 23:56   740
20200330.txt             2020-03-30 03:56   536
```

Figure 26. A directory where data are stored for device ID1.

As can be seen in figure 26, a typical file occupies 2.9 kilobytes of data storage. By calculating, a device will take around 1.05 megabytes yearly.

A real-life setup of this project can be seen on figure 27. There are two set of data readers and sensors connecting to a data collector. A DHT11 temperature and humidity sensor is on the top left corner, while LDR light sensor is the right side, and soil moisture probe is in the bottom middle of the picture. A Raspberry Pi stands between DHT11 and light sensor.

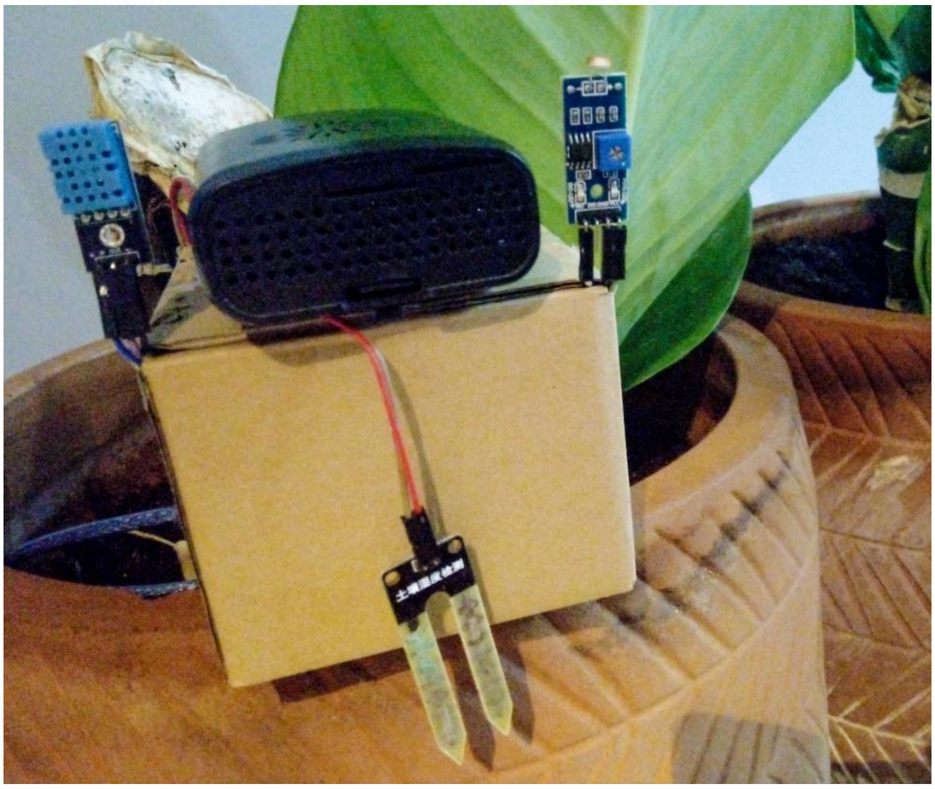


Figure 27. A picture of ACED device in real-life.

Generally speaking, the hardware and software function as expected. A successful result of this project was obtained. The information of environmental factors from surrounding area are logged effectively. Data transferring between devices works comfortably. Sending and receiving data by using wireless LAN also functions as expected. The web page application generates chart as required.

5.2 Measurement

In figure 28 and 29 measurements on Wednesday March 25th 2020 from device ID 1 and device ID 2 are shown respectively. Device ID 1 was placed outdoor and device ID 2 was set indoor.

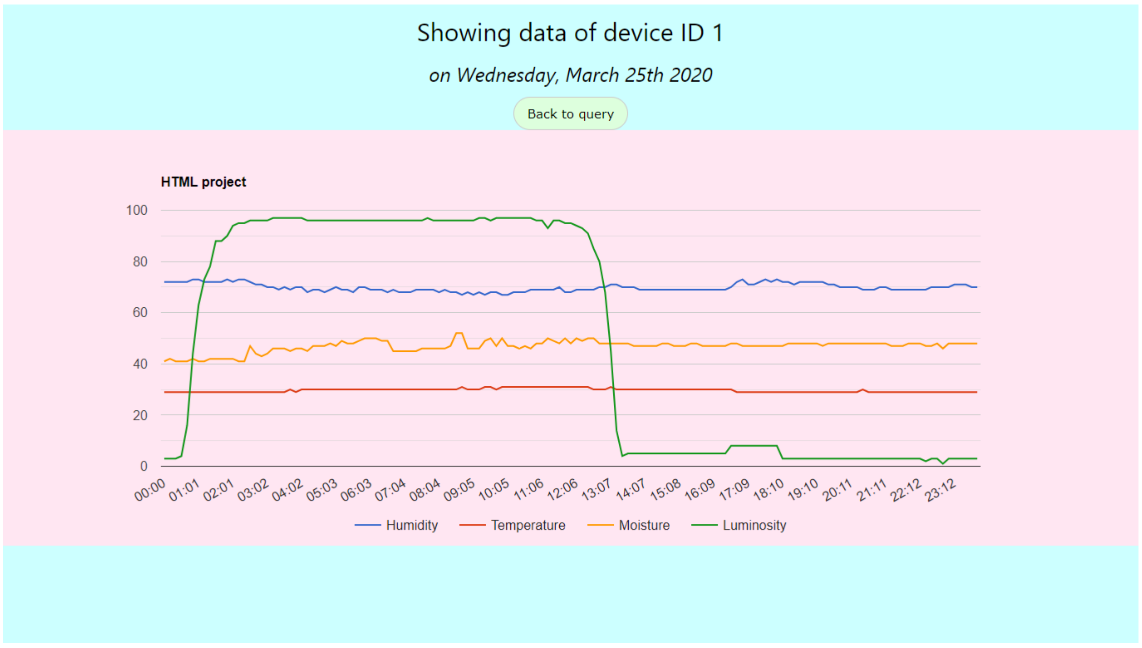


Figure 28. Data from device ID 1 on March 25th 2020

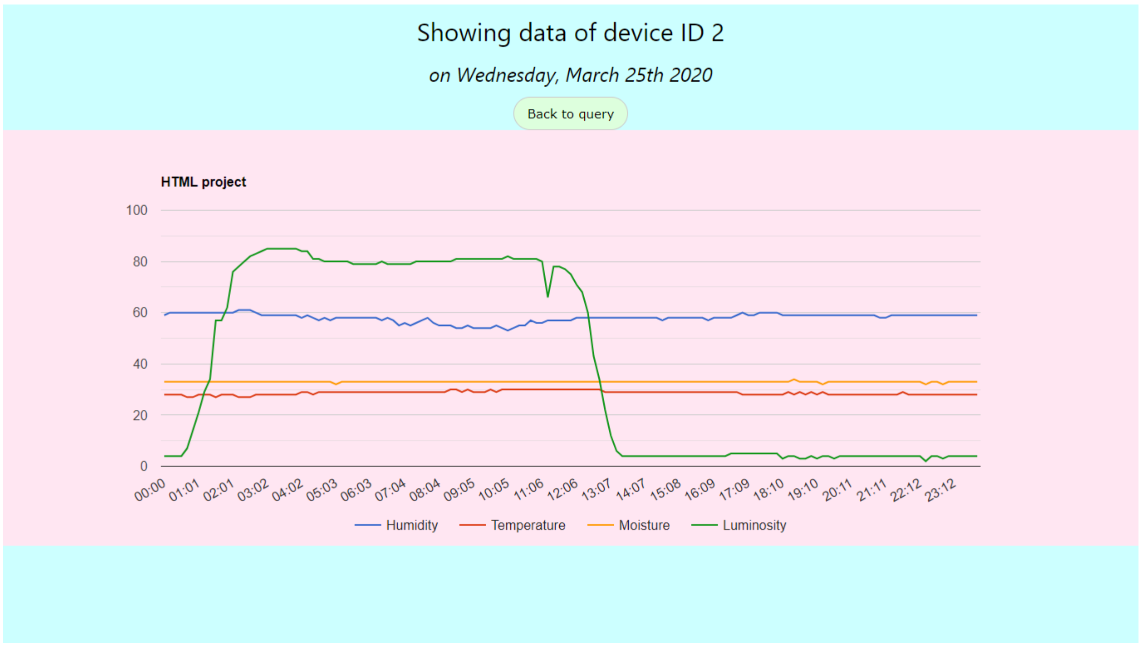


Figure 29. Data from device ID 2 on March 25th 2020

As can be seen in the figures above, graphs are drawn on a web page application inside pink area. Humidity curve is in blue color, while temperature curve has red color, soil moisture has orange curve and light intensity curve is in green color. For more examples, visit the web page application of this project at http://users.metropolia.fi/~theng/thesis.

The measurements can be used for analyzing and monitoring environmental factors. This project is also capable to combine with other device for a smart system such as automatic watering system or smart ventilation application. In term of agriculture, these measurements help famers to monitor surrounding environment. By merging with other tools, famers will have a smart controlling system for their garden. They are also able to monitor and compare daily environmental factors data in order to optimize the usage of resources or improve the products quality.

5.3 Advantages and Disadvantages

This project was done with an affordable price. In table 5 below, the amount of money spent for hardware were illustrated. Note that this is only inexpensive and low-end components bought from the internet. Price was calculated with one set of sensors and a data reader. A data collector has capacity to connect up to four data readers with sensors included.

Table 5. Cost for hardware implemented in this project

Units	**Components**	**Cost (in EUR)**	**Total (in EUR)**
Data reader (1)	DHT11 sensor	1	5
	Soil moisture sensor	0.5	
	Light intensity sensor	0.4	
	Arduino Uno R3	3.1	
Data collector (1)	Raspberry Pi 3B+	41	41
One data collector and one set of data reader with sensors			**46**

As demonstrated in the table above, this project which uses a Raspberry Pi 3B+ as data collector and two set of data reader (including two Arduino Uno and two set of sensors) costs only EUR 51. A set of data reader (an Arduino Uno and three sensors) costs only EUR 5. For a full capacity device (one data collector connects to four data readers with four sensors unit) it costs only EUR 61. There are also less expensive devices in market which are able to reduce the cost.

Small amount of data stored in server is another strength of this project, since a set of data reader occupies only 1.05 MB a year. Thanks to the linear storing method, data processing is also easier.

Despite having data (which is returned from sensors) quite useful, the accuracy of the measurement should have more precision by increasing the quality of sensors. Furthermore, power supply for the microcontroller boards sometimes causes discontinuous data transferring. There have been days when problems occurred, causing data lost, malfunction hardware etc. Data dropping were mostly caused by poor internet connectivity or power grid shutdown.

6 Conclusion

The principal goal of this project was to harvest information from environmental factors and visualize its data into graph. The project was divided into two parts, electronics and visualization. In order to complete the electronics aspect, a functional ACED device has been developed by sensors and microcontroller boards. This was the most challenging part of the project as it took time and research. By applying theoretical knowledge, the sensors were able to recognize the change of environmental factors. The microcontroller boards had been operating without internal error. The final version of ACED device functioned and resulted measurements as expected. On the other hand, visualization tool presented data as required. User interface of the web application has been improved to meet user friendly criteria. The device had been running for testing the accuracy of the measurements. By April 8th 2020 the ACED device had been operating for more than a month. All the data acquired are accurate.

In general two parts of this project successfully operated as intended. The captured data by ACED device had low error and high precision. Transferring data into server by wireless communication worked appropriately. Values were kept in storage by order as required. Data were presented satisfactorily by visualization tool on web page application. In conclusion everything had been operating as expected.

Although the project operates smoothly, there are some modification needed in order to enhance the results and user experience, such as user interface or the accuracy of the sensors. Some ideas for further improvement can be seen below:

- Improving quality of sensors which results in more accurate and reliable measurement.
- Replacing Arduino Uno and Raspberry Pi with more economic devices that have similar functionality which reduces the cost.
- Upgrading user interface by more responsive web page application.
- Running devices by using external power source which has more portable ability.
- Transferring data through the internet by using cellular network.
- Creating more secured data transferring over the internet.
- Restarting system automatic when there are problems occur.

References

1. Judith Cardell. Overview of Microprocessor. [Online] [Cited on March 11th 2020]. http://www.science.smith.edu/~jcardell/Courses/EGR328/Readings/uProc%20Ovw.pdf

2. Osborne, Adam. An Introduction to Microcomputers. Volume 1: Basic Concepts (2nd edition). Berkeley, California: Osborne-McGraw Hill. ISBN 978-0-931988-34-9.

3. Muhammad Shaaban. Microcontroller Basic. Publish on 2-9-2000. [Online] [Cited on March 11th 2020]. http://meseec.ce.rit.edu/eecc250-winter99/250-2-9-2000.pdf

4. RayPCB. The schematic diagram of automotive sensor signal conditioning. [Online] [Cited on March 12th 2020]. https://www.raypcb.com/the-schematic-diagram-of-automo-tive-sensor-signal-conditioning/

5. Ravi Teja. What is a sensor? Different type of sensors, application. [Online] [Cited on March 12th 2020] https://www.electronicshub.org/different-types-sensors/

6. Dave. Know about Various Types of Temperature Sensors. [Online] [Cited on March 12th 2020] https://www.watelectrical.com/6-different-types-of-temperature-sensors-with-their-specifications/

7. Electronics-Tutorials. Thermistor and NTC Thermistor. [Online] [Cited on March 12th 2020] https://www.electronics-tutorials.ws/io/thermistors.html

8. Elprocus. Humidity Sensor – Working principles and its application. [Online] [Cited on March 13th 2020] https://www.elprocus.com/a-memoir-on-humidity-sensor/

9. Lastminuteenginneers. How soil moisture sensor works and interface it with Ar-duino. [Online] [Cited on March 16th 2020]. https://lastminuteengineers.com/soil-mois-ture-sensor-arduino-tutorial/

10. Seedstudio. What is a light sensor? [Online] [Cited on March 17th 2020]. https://www.seeedstudio.com/blog/2020/01/08/what-is-a-light-sensor-types-uses-ar-duino-guide/

11. Resistorguide. Photo resistor. Light dependent resistor. [Online] [Cited on March 18th 2020]. http://www.resistorguide.com/photoresistor/

12. Jimblom. Serial communication. [Online] [Cited on March 19th 2020] https://learn.sparkfun.com/tutorials/serial-communication/

13. Chua Hock Chuan. Introduction to HTTP basics. [Online] [Cited on March 23rd 2020] https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

14. W3school. HTTP methods GET and POST. [Online] [Cited on March 23rd 2020] https://www.w3schools.com/tags/ref_httpmethods.asp

15. Google Charts. Charts | Google developers. [Online] [Cited on March 23rd 2020] https://developers-dot-devsite-v2-prod.appspot.com/chart

16. Knud Lasse Lueth. Internet of Things – Definition, history and disambiguation. [Online] [Cited March 24th 2020] https://iot-analytics.com/internet-of-things-definition/

17. DataFlair. How IoT works – 4 main components of IoT System. [Online] [Cited on March 24th 2020] https://data-flair.training/blogs/how-iot-works/

18. Mouser. DHT11 Humidity & Temperature Sensor Datasheet. [Online] [Cited on March 25th 2020] https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

19. Mohammad Abu Sayed. Soil Moisture Measurement with Arduino. [Online] [Cited on March 26th 2020] https://www.instructables.com/id/Soil-Moisture-Measurement-With-Arduino/

20. Arduino. Arduino Introduction. [Online] [Cited on March 26th 2020] https://www.arduino.cc/en/guide/introduction

21. Arduino.cc. Arduino Uno R3. [Online] [Cited on March 26th 2020] https://store.arduino.cc/usa/arduino-uno-rev3

22. Raspbian Foundation. About Raspbian. [Online] [Cited on March 26th 2020] https://www.raspbian.org/RaspbianAbout

23. Raspberry Pi Foundation. Raspberry Pi 3 Model B+. [Online] [Cited March 27th 2020] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

24. CactusIO. Arduino Uno R3. [Online] [Cited on April 18th 2020] http://cactus.io/platform/arduino/arduino-uno

25. Electronicswings. Soil Moisture Sensor. [Online] [Cited on March 27th 2020] https://www.electronicwings.com/arduino/soil-moisture-sensor-interfacing-with-arduino-uno

26. Researchgate. Structure of a capacitive humidity sensor. [Online] [Cited on April 14th 2020]. https://www.researchgate.net/figure/Structure-of-capacitive-humidity-sensor-Source-Reproduced-with-permission-from24_fig4_312326500

27. EngineeringTutorial. Application of Photo diodes. [Online] [Cited on April 14th 2020] https://engineeringtutorial.com/applications-photo-diodes/

28. ElectricalFundaBlog. Parallel Communication. [Online] [Cited on April 14th 2020] https://electricalfundablog.com/parallel-communication-characteristics/

29. TinoBlog. HTTP requests and HTTP responses. [Online] [Cited on April 14th 2020] https://blog.tinohost.com/request-va-response-trong-lap-trinh-web/

30. TheEngineeringProject. Introduction to DHT11. [Online] [Cited on April 14th 2020] https://www.theengineeringprojects.com/2019/03/introduction-to-dht11.html

31. Raspberry Pi Foundation. GPIO Raspberry Pi Documentation. [Online] [Cited on April 14th 2020] https://www.raspberrypi.org/documentation/usage/gpio/

32. Phil Croucher. Parallel Communication Explained. [Online] [Cited on April 18th 2020] http://hardwarehell.com/articles/parallel.htm

33. REES52. LM393 Optical Photosensitive. [Online] [Cited on April 18th 2020] https://rees52.com/arduino-modules/748-lm393-optical-photosensitive-ldr-light-sensor-module-for-arduino-shield-dc-3-5v-ml026

34. Yahya Tawil. Understanding Arduino UNO Hardware Design. [Online] [Cited on April 20th 2020] https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/

Appendix 1: Code for Generating Line Chart

This code is cloned from: https://developers-dot-devsite-v2-prod.appspot.com/chart/interactive/docs/gallery/linechart.html

```
<html>
  <head>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Year', 'Sales', 'Expenses'],
          ['2004',  1000,      400],
          ['2005',  1170,      460],
          ['2006',  660,       1120],
          ['2007',  1030,      540]
        ]);

        var options = {
          title: 'Company Performance',
          curveType: 'function',
          legend: { position: 'bottom' }
        };

        var chart = new google.visualization.LineChart(document.getEl-
ementById('curve_chart'));

        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="curve_chart" style="width: 900px; height: 500px"></div>
  </body>
</html
```

Appendix 2: Code for Data Reader in Arduino Uno

File name: dataRead.ino

```
#include <dht.h>       //library uses for DHT11
#include <string.h>    //library uses for string operations
#define DHT11_pin 5    //DHT11 pin
#define LDR_pin A0     //LDR pin
#define SM_pin A1      //Soil moisture pin
#define PW_pin_LDR 11       //Power supply 5V pin for LDR
#define DHT11_delay 1000     //DHT11 delay time
#define Device_ID 2    //Device ID ***Important *** CHECK WHEN UPLOAD-
ING CODE!
//red = ID1, yellow = ID2

dht DHT;

int getHumidity(void);
int getTemperature(void);
int getMoisture(void);
int getLuminosity(void);
int initial(void);

void setup() {
  Serial.begin(9600);        //Starts Serial connection at 9600 baud
  delay(1000);               //Lets Arduino have time to startup
  pinMode(LDR_pin, INPUT);
  pinMode(SM_pin, INPUT);
  pinMode(DHT11_pin, INPUT);
  pinMode(PW_pin_LDR, OUTPUT);    //Needs 1 pin for power supply 5V
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);                //Turns off LED13 to save en-
ergy
  initial();                                    //initialize sensors
}

void loop() {
  char str[100];
  if (Serial.available())
  {
    char sign = Serial.read(); //Check collecting request from Rasp-
berry Pi
    if (sign==49)
    {
     sprintf(str, "{\"hu-
mid\":\"%d\",\"tempe\":\"%d\",\"moist\":\"%d\",\"light\":\"%d\",\"ID\"
:\"%d\"}",
         getHumidity(), getTemperature(), getMoisture(), getLuminos-
ity(), Device_ID);     //Print data format to str
     Serial.println(str);      //Send str data to the collector
  }
}

int getHumidity()
{
  delay(DHT11_delay);  //Gives DHT11 time to read data
  int chk = DHT.read11(DHT11_pin);
  return (int)DHT.humidity;
}

int getTemperature()
{
  delay(DHT11_delay);  //Gives DHT11 time to read data
  int chk = DHT.read11(DHT11_pin);
  return (int)DHT.temperature;
}

int getMoisture()
{
  int val = (1023 - (int)analogRead(SM_pin))/10;    //scale the number 
to 102
  return val;
}

int getLuminosity()
{
  digitalWrite(PW_pin_LDR, HIGH);
  delay(100);
  int val = (1023 - (int)analogRead(LDR_pin))/10;    //scale the num-
ber to 102
  digitalWrite(PW_pin_LDR, LOW);
  return val;
}

int initial()
{
  getHumidity();
  getTemperature();
  getMoisture();
  getLuminosity();
  return 0;
}
```

Appendix 3: Code for Data Collector in Raspberry Pi

File name: processData.py

```
import requests
import json
import time
import serial

sleep_time = 60*10     #log data every 10 minutes

#Start serial communication
ser1 = serial.Serial('/dev/ttyACM0', 9600, timeout = 5)
ser2 = serial.Serial('/dev/ttyUSB0', 9600, timeout = 5)
flag = 0

url = 'http://users.metropolia.fi/~theng/thesis/get.php'

print("Starting up in 10s...")

while True:
        ser1.write('1') #sends collecting request
        ser2.write('1')
        frA1 = ser1.readline() #collects data returned from reader
        frA2 = ser2.readline()
        if flag != 0:   #if this is not the first loop
                data1 = json.loads(frA1)        #converts into JSON for-
mat
                data2 = json.loads(frA2)
                requests.post(url, data1)       #sends data to server
with specified URL
                requests.post(url, data2)
                #print("Data is sent")
                time.sleep(sleep_time)          #wait for 10 minutes un-
til next loop
        else:                   #if this is the first loop then skip
                time.sleep(5)
        flag = 1        #skip first loop because it causes trouble
```

Appendix 4: Code for Data Logging on Server Side

File name: get.php

```
<?php
	//print_r($_POST);
	//humid tempe moist light
	//Opens file where data will be logged
	//Has directory as "./data<ID_number>/<date_yyyymmdd>.txt
	$file = "./data".$_POST["ID"]."/" . date("Ymd").".txt";
	//Each line has format as "<time_stamp>, <humidity_value>,
<temperature_value>, <soil_moisture_value>, <luminosity_value>"
	$toWrite = date("'H:i'").','.$_POST["hu-
mid"].','.$_POST["tempe"].','.$_POST["moist"].','.$_POST["light"]."\r\
n";
	$f = fopen($file, "a");
	fwrite($f, $toWrite);
	fclose($f);
 ?>
```

Appendix 5: Code for Sending Query Web Page Application

File name: index.php

```
<html>
        <head>
                <title>Environmental factors monitoring project</title>
                <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
                <meta name="theme-color" content="#ccffff" />
        </head>
        <body style="text-align:center; background-color:#ccffff">
                <div class="w3-container w3-panel w3-indigo w3-display-
middle w3-mobile">
                        <br>
                        <form class="w3-table w3-mobile" ac-
tion="graph.php" method="get">
                        <table>
                                <tr><th class="w3-text-light-blue">ID:
</th><td><input class="w3-input " autofocus="true" min = "1" name="ID"
placeholder="Device ID" type="number"></td></tr>
                                <tr><th class="w3-text-light-blue">Date:
</th><td><input class="w3-input" name="date"   type="date"></td></tr>
                                <tr><th></th><td><input class="w3-button
w3-border w3-round-xxlarge w3-pale-blue" type="submit"
value="Send"></td></tr>
                        </table>
                </div>
                <p>Last date logged: <?php $a = scandir("./data1", 1);
echo date("F jS Y",strtotime(substr($a[0],0,8))); ?> for ID 1</p>
                <p>Last date logged: <?php $a = scandir("./data2", 1);
echo date("F jS Y",strtotime(substr($a[0],0,8))); ?> for ID 2</p>
                <h2 class="w3-text-red"><?php if (isset($_GET["info"]))
echo "Your input is invalid or date was not logged. Please try again";
?></h2>
        </body>
</html>
```

Appendix 6: Code for Drawing Graph Web Page Application

File name: graph.php

```
<html>
  <head>
        <?php
                if (!isset($_GET["ID"]) || !isset($_GET["date"]) ||
empty($_GET["ID"]) || empty($_GET["date"]) || !file_ex-
ists("./data".$_GET["ID"]."/".date("Ymd", strto-
time($_GET["date"])).".txt"))
                        {
                                header("Location: ./in-
dex.php?info=false");
                                die();
                        }
        ?>
        <meta name="theme-color" content="#ccffff" />
        <title>Showing data</title>
        <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Time', 'Humidity', 'Temperature', 'Moisture', 'Luminosi-
ty'],
          //['2004',  1000,      400],
                <?php
                        $id = $_GET["ID"];
                        $date = date("Ymd", strtotime($_GET["date"]));
                        $f = fopen("./data".$id."/".$date.".txt", "r");
                        while(!feof($f)) {
                                $line = fgets($f);
                                if ($line != "")
                                        echo "[".$line."],";
                        }
                ?>
        ]);
        var options = {
                backgroundColor: '#ffe6f2',
          title: 'HTML project',
          legend: { position: 'bottom' }
        };
        var chart = new google.visualization.LineChart(document.getEl-
ementById('curve_chart'));
        chart.draw(data, options);
      }
    </script>
  </head>

  <body style="text-align:center; background-color:#ccffff;">
        <h2>Showing data of device ID <?php echo $_GET["ID"]?></h2>
        <h3><i>on <?php echo date("l, F jS Y", strto-
time($_GET["date"]))?></i></h3>
        <a class="w3-button w3-border w3-round-xxlarge w3-pale-green"
href="./index.php">Back to query</a>
    <div class="" id="curve_chart" style="width: 100%; height:
500px"></div>
  </body>
</html>
```