Media Engineering and Technology Faculty
German University in Cairo

Estimation of Light Directivity

Bachelor Thesis

Author: Mohamed Bahgat Salah
Supervisors: Dr. Hisham Othman
Submission Date: 7 February, 2021

Media Engineering and Technology Faculty
German University in Cairo

Estimation of Light Directivity

Bachelor Thesis

Author: Mohamed Bahgat Salah
Supervisors: Dr. Hisham Othman
Submission Date: 7 February, 2021

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowledgement has been made in the text to all other material used

---

Mohamed Bahgat Salah
7 February, 2021

Acknowledgments

Without you my Lord, I am truly nothing. Thank you for everything. Thank you my mom, thank you my dad, thank you to all of those who believed in me and who prayed for me. Thank you my grandfather, I will never forget your everyday prayers. Special thanks to one of my dearest Dr. I have ever met, special thanks to you Dr. Hisham. Thank you for being a father before being my Dr. and my supervisor.

Abstract

Goals without paths and well-known directions are just dreams. So knowing the direction and the position of a specific object is truly important, as you will have a clear vision about that object and how to deal with it. In this paper, I will argue how to estimate the light directivity at a certain place and how to get the overall average intensity of the light in it. By knowing the directivity and the intensity of light you will be able to determine where the shadows are and which of those places will be darker or lighter than the others. In photography, this is very useful, as it allows photographers and movie-makers to adjust their cameras' positions, so they can capture the clearest image. Moreover, it allows us to determine the position of any light source even if it is moving and not fixed at a certain position. Last but not least, the most amazing and interesting part of the project is allowing us to apply it on any other waveform other than the light wave. For simplicity, the idea of the project is applicable to be applied on any shape of waves that can traverse and it will determine the position of its source and calculate the overall average intensity at this location. We can achieve this by replacing those light sensors with the suitable sensors of that field; like sound sensors, heat sensors, infrared sensors and in communication systems between transmitters and receivers. The idea of estimation of light directivity goes as the following: let the light sensors read the light intensity at your location and transmit it to a microcontroller which is called Arduino so it can analyze it. The microcontroller shares the output results as a text message on a screen connected to it through a breadboard and wires and also as a voice message played by a smartphone connected to it using Bluetooth.

Contents

Chapter 1

Introduction

1.1 Estimation of Light Intensity and Directivity

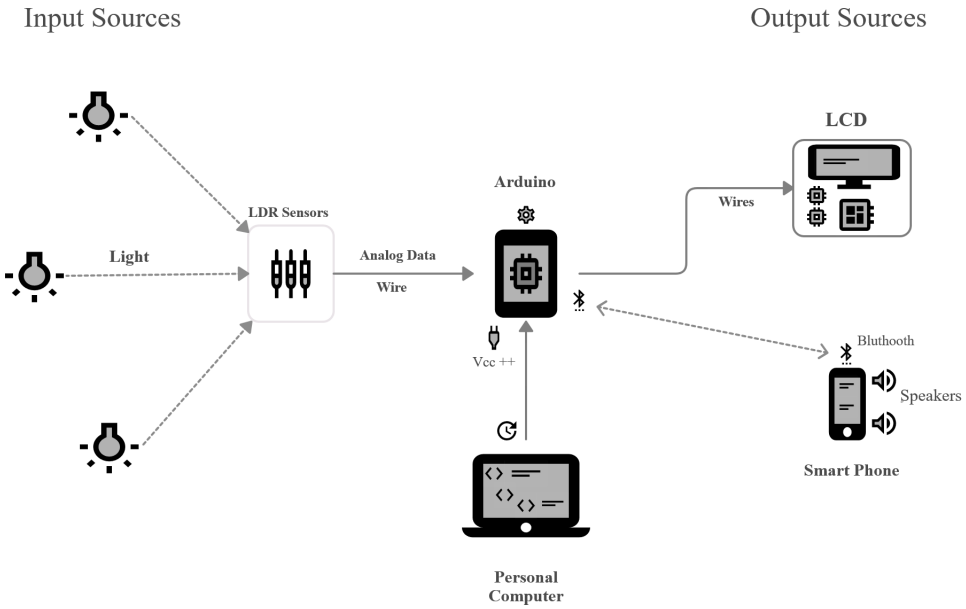


Figure 1.1: Estimation of Light Intensity and Directivity.

Estimation of light intensity and its directivity plays a crucial role in many fields nowadays. Due to the growing demands of science and technology of our daily life, the light became an essential element in many fields and attracted much attention to exploring its unrevealed benefits. Light intensity and directivity estimation are one of the most challenging problems recently in Computer Vision, Photography, Photo-chemical Reactions, Solar Systems. This paper presents a real hardware model that can detect and estimate the direction of the out-coming light from a fixed or a moving light source and calculate the overall average intensity of the light in that place.

1.2 Waves

Since light is an electromagnetic wave, We can not talk about it and forget to mention what are waves. When you float in the pool on a warm summer day, the up-and-down movement of the water tells you waves are moving past. Sometimes the waves are so strong they almost push you over. Other times, the waves just gently rock you. You know about water waves because you can see and feel their movement, but there are other types of waves, also. Different types of waves carry signals to televisions and radios. Sound and light waves move all around you and enable you to hear and see. Waves are even responsible for the damage caused by earthquakes.[1]

Waves is a disturbance that moves through a matter or space. Waves carry energy from one place to another. You can see that in Fig. 1.2

Figure 1.2: The movement of the fishing bob produces water waves that carry energy through the water.

Waves usually are produced by something moving back and forth, or vibrating. It is the energy of the vibrating object that waves carry outward. This energy can spread out from the vibrating object in different types of waves. Some waves, known as mechanical waves, can travel only through matter such as sound, water waves. Other waves called electromagnetic waves can travel either through matter or through empty space such as light and radio waves.[1]

1.3 Light

1.3.1 Nature

Figure 1.3: The Moon reflects light from the Sun. These light waves travel through space to reach your eyes.

On a clear night, you might see the Moon shining brightly, as in Fig. 1.3. Like other waves, light waves can travel through matter, but light waves are different from water waves and sound waves. Light from the Moon has travelled through space that contains almost no matter[2]. You can see light from the moon, distant stars, and galaxies because the light is an electromagnetic wave. Electromagnetic waves are waves that can travel through matter or through empty space. See more details in Fig. 1.4

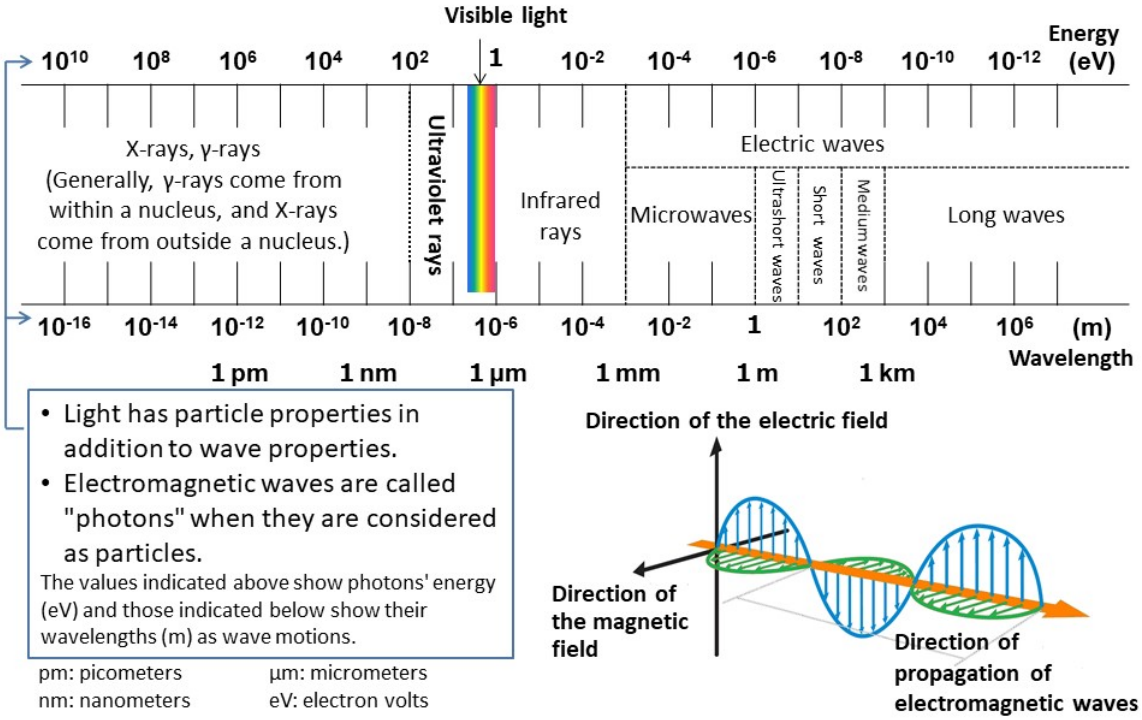


Figure 1.4: Electromagnetic waves.

There was an ancient belief, which is regularly reinvented by children, that you see something by sending out some kind of probe from your eyes. A more scientific view is that we see things because the light comes from them to our eyes. But only a few things generate their own light. Before the middle of the nineteenth century, practically all light came from a few kinds of a luminous object like the sun, the stars and fires[13]. So those were the only objects that could be seen by their own light. To see other things we need a luminous object as a source of light.[12]

Light travels from the luminous source to the object and then to our eyes. We normally assume that a thing is located in the direction where the light comes from. So it would seem that when it is not actually bouncing off something light must travel in more or less straight lines as in Fig. 1.5

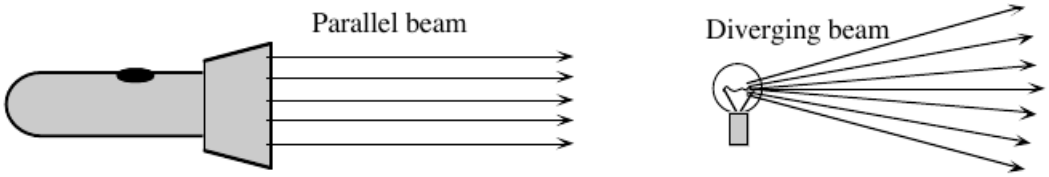


Figure 1.5: Beams of light represented as bundles of rays.

1.3.2 Speed

Have you ever seen a movie where a spaceship travels faster than the speed of light? In reality, nothing travels faster than the speed of light. In empty space, light travels at a speed of about 300,000 km/s. Light travels so fast that light emitted from the Sun travels 150 million km to Earth in only about eight and a half minutes. However, when light travels in the matter, it interacts with the atoms and molecules in the material and slows down. As a result, light travels fastest in empty space and travels slowest in solids. In the glass, for example, light travels about 197,000 km/s.[12]

1.3.3 Wavelength and Frequency

Wavelengths of light are usually expressed in units of nanometers (nm). One nanometer is equal to one billionth of a meter. For example, the green light has a wavelength of about 500 nm or 500 billionths of a meter. A light wave with this wavelength has a frequency of 600 trillion Hz[10]. see Fig. 1.6

Figure 1.6: Colors' Wavelengths ranges

1.3.4 Intensity

The intensity of waves is the measure of the amount of energy that the waves carry. For light waves, the intensity determines the brightness of the light. A dim light has lower intensity because the waves carry less energy. However, as you move away from a light source, the energy spreads out and the intensity decreases. see Fig. 1.7

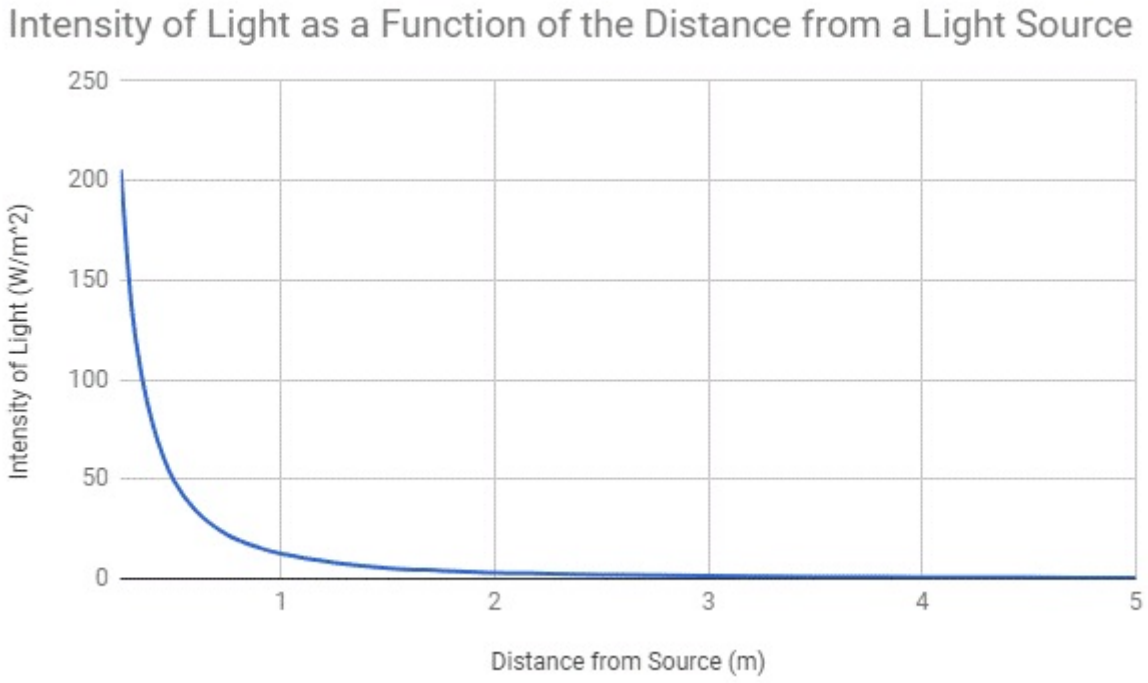


Figure 1.7: Intensity vs Distance

1.3.5 Duality property

The dual nature of light means that, in some experiments, light behaves as a wave. In other experiments, light behaves as a particle. In 1801, Thomas Young shined light between two adjacent slots. The light waves interfered with each other and formed an alternating pattern of light and dark bands; the light bands are the constructive interferences, and the dark bands are the destructive interferences. If the light consisted of small particles, the alternating light and dark bands would not have occurred[5]. Fig. 1.8 explains it.

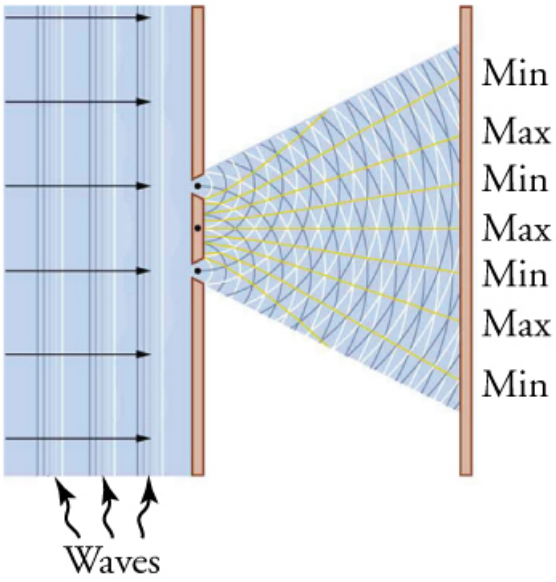


Figure 1.8: Light behaves as a wave.

In 1905, Albert Einstein's photoelectric effect experiment showed that a beam of light could eject electrons from a metal. He proposed that light consists of photons with an energy that depended on the frequency (V) of the light and that a photon with a frequency over a certain level (Vo) would have sufficient energy to eject an electron. The light was behaving as a stream of particles[5]. Fig. 1.9 explains it .

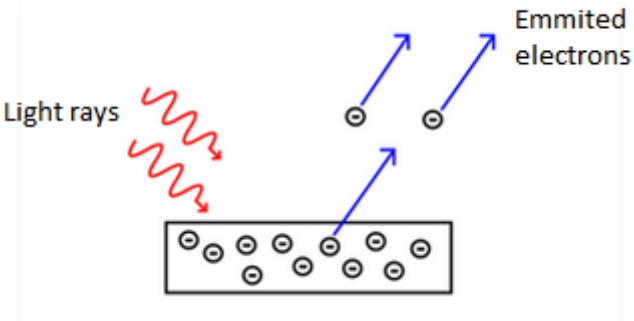


Figure 1.9: Light behaves as a particle.

1.3.6 Sources of Light

It is possible to distinguish between light sources, which possess i) a continuous polychromatic spectrum (type A), ii) a discrete polychromatic spectrum (type B), or iii) a narrow-band monochromatic spectrum (type C)[11].

1.3.6.1 a) Continuous Polychromatic Spectrum

Typical examples for light sources of type A are incandescence lamps (light bulbs), halogen lamps, xenon arc lamps, standard and compact fluorescence lamps (CFL), and the sun. These light sources emit a continuous spectrum over a wide spectral range. To ensure reaction control, the emitted light has to be filtered. Such filtering can be crucial for reactions, where different reactants and/or products have a similar absorption spectrum. The geometry of these light sources can vary from a bulb to tubular shape Fig. 1.10 explains it.

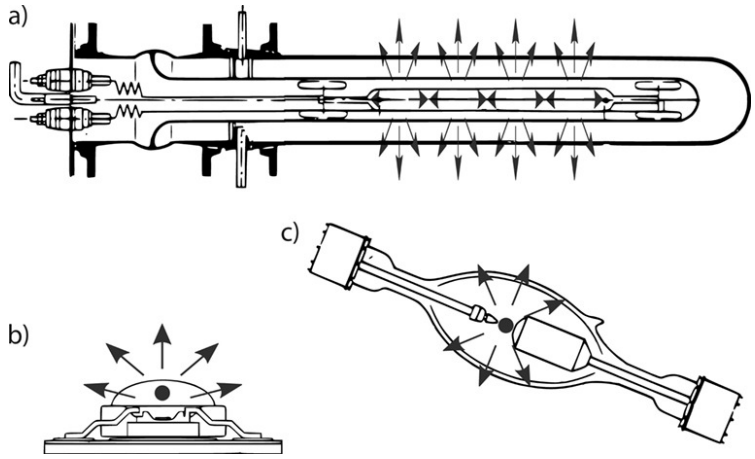


Figure 1.10: MP-Hg lamp (a), SMD LED (b), and a Xe arc lamp (c). The arrows indicate the direction of emission.

1.3.6.2 b) Discrete Polychromatic Spectrum

Mercury-vapor lamps are light sources with discrete, polychromatic emission (type B). Depending on the pressure used, mercury-vapor lamps can also show a continuous, polychromatic emission. The same applies for certain phosphor coatings for fluorescence lamps. A discrete emission tends to be beneficial since filtering of undesired wavelengths can be realized easier. The optical properties of band-pass filters can be less demanding. The geometrical properties of light sources of type B are usually similar to that of type A. Light emission in most cases is circular. Despite the differences in the shape of the lamp tubes, this also applies for CFLs, as long as the radiation field outside the entire light source is considered.

1.3.6.3 c) Narrow-Band Monochromatic Spectrum

Besides LEDs and organic light-emitting diodes (OLEDs), excimer lamps, and laser emit narrow-band monochromatic light (type C). The adjective narrow-band is important from a technical perspective as it reflects the fact that especially commercially available LEDs

and OLEDs can possess emission bands of up to several tens of nanometer. This is still narrower than the emission of type A light sources, but by far broader than the emission of, e.g., a laser.

1.4 Importance

knowing the directivity and the intensity of light we will be able to determine where the shadows are and which of those places will be darker or lighter than the others. In photography, this is very useful, as it allows photographers and movie-makers to adjust their cameras' positions, so they can capture the clearest image. Moreover, it allows us to determine the position of any light source even if it is moving and not fixed at a certain position.

Light has an extremely huge role in every photochemical reaction happens. The photochemical reaction is none other than a chemical reaction that starts with light being absorbed as a form of energy [7]. Chemical reaction rates increase or decrease according to factors including temperature, pressure and light. So by controlling the amount of intensity and the incoming light rays you can increase or decrease the rate of those reactions[8].

Chapter 2

Background

We can divide the project into two parts; Hardware and Software. The hardware part that is responsible for any physical connections between devices and choosing the appropriate components from hardware devices that will help us to get what we want from light intensity and light directivity such as sensors. While the software part is responsible for installing the suitable apps on the PC and the smartphone and writing the code that would be burned on the microprocessor chip of the Arduino board. We will discuss them in more details in the following sections.

Figure 2.1: The project contains two parts Software and Hardware.

2.1 Hardware Part

2.1.1 Arduino Microcontroller

2.1.1.1 What is Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software[16]. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

2.1.1.2 why Arduino

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community[6].

2.1.2 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560[6]. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started see Fig. 2.2 and Fig. 2.3.

Figure 2.2: ArduinoMega from the top.

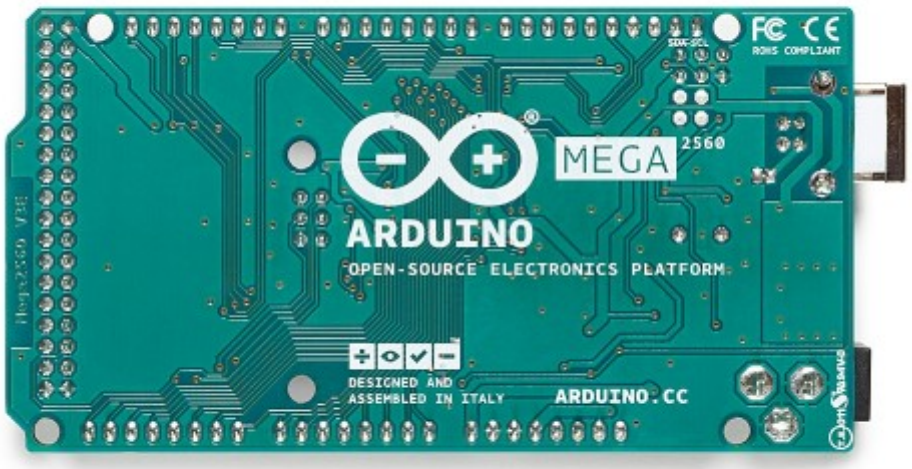


Figure 2.3: ArduinoMega from the bottom.

2.1.2.1 Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.[6]

2.1.2.1.1 Vin

The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

2.1.2.1.2 5V

The regulated power supply used to power the microcontroller and other components on the board. This can come either from Vin via an on-board regulator, or be supplied by USB or another regulated 5V supply. They are 4 pins in Arduino Mega 2560.

2.1.2.1.3 3.3V

volt supply generated by the on-board regulator. Maximum current draw is 50 mA.It is only 1 in Arduino Mega 2560.

2.1.2.1.4 GND

Ground pins.They are 5 pins in Arduino Mega 2560.

2.1.2.2 Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the boot-loader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

2.1.2.3 Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 k Ohms. Summary in Fig. 2.4

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Figure 2.4: ArduinoMega 2560 microcontroller.

ArduinoMega 2560 Pins sheet in this Fig. 2.5

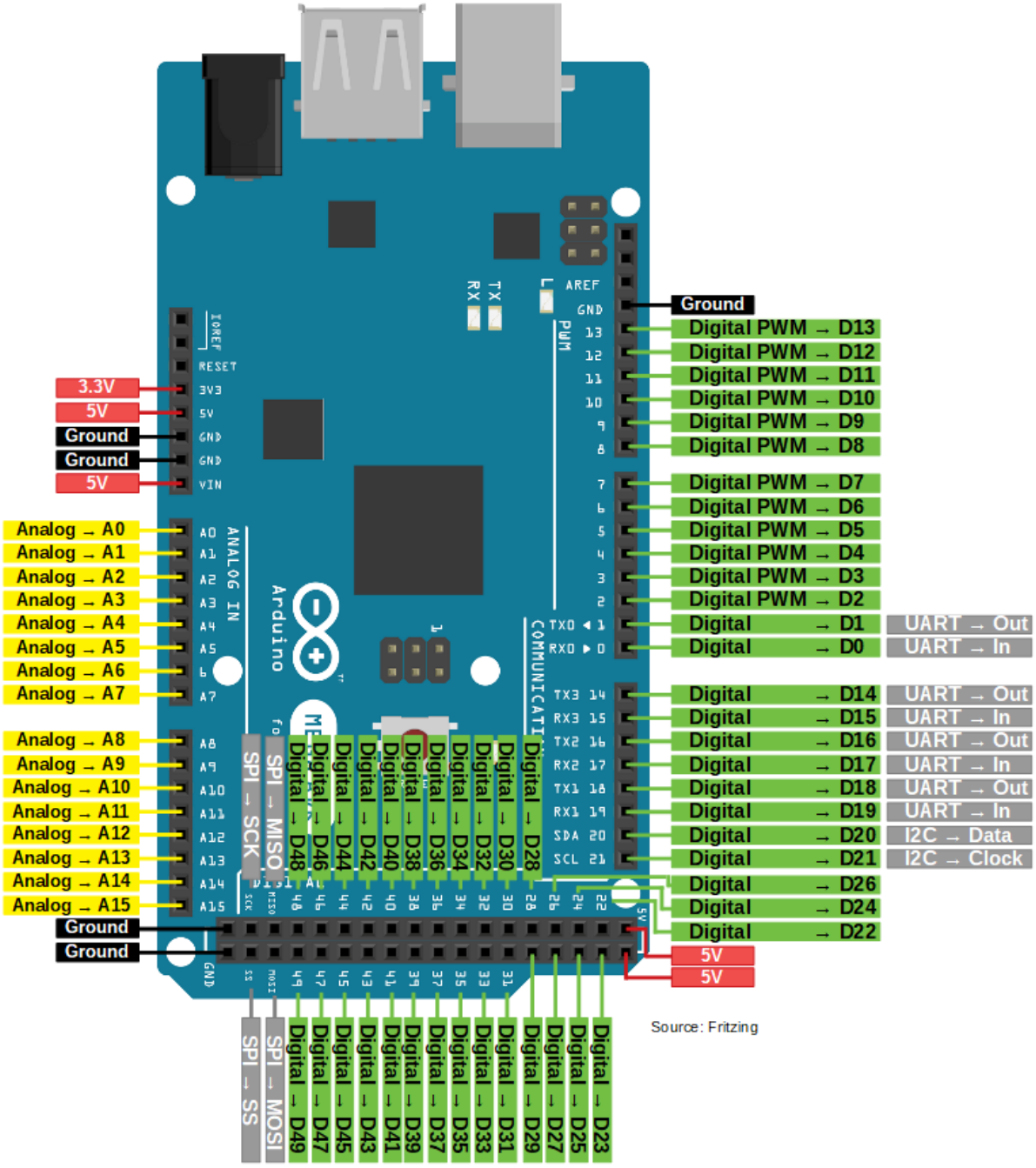


Figure 2.5: ArduinoMega 2560 Input/Output pins.

As we can see in ArduinoMega 2560 microcontroller it contains 16 analog-to-digital (A/D) converter. The main function of the analog pins is to read data from analog sensors and return an integer value from 0 to 1023 according to us equivalent to the analog data that has been read. The analog pins also have all the functionality of general

purpose input/output (GPIO) pins. i.e. It can work as those digital pins. The analog pins also have pull-up resistors, which work identically to pull-up resistors on the digital pins. Those 16 Analog pins have been used as input analog pins that were connected to LDR sensors in order to estimate the intensity and directivity of light.

2.1.3 1Sheeld+

1Sheeld+ is an Arduino shield. It is connected to a mobile app that allow the usage of smartphones' capabilities such as LCD Screen, Keyboard, Switches, LEDs, Speakers. We can use 1Sheeld to take inputs or show outputs using the sensors and peripherals already available on our smartphone[3]. See Fig. 2.6

It consists of two parts. The first part is a shield that is physically connected to the Arduino board and acts as a wireless middle-man, piping data between Arduino and any smartphone via Bluetooth. The second part is a software app on smartphones that manages the communication between the shield and the smartphone.

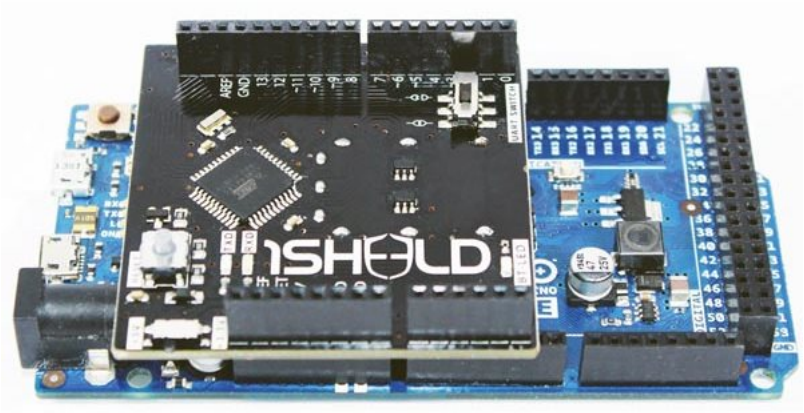


Figure 2.6: 1sheeld+ connected to an ArduinoMega 2560.

2.1.4 Light Dependent Resistor Module (LDR module)

LDR sensor module is used to detect the intensity of light[15]. It is associated with both analog output pin and digital output pin labelled as AO and DO respectively on the board. When there is light, the resistance of LDR will become low according to the intensity of light. The greater the intensity of light, the lower the resistance of LDR. The sensor has a potentiometer knob that can be adjusted to change the sensitivity of LDR towards light. See Fig. 2.7 and Fig.2.8

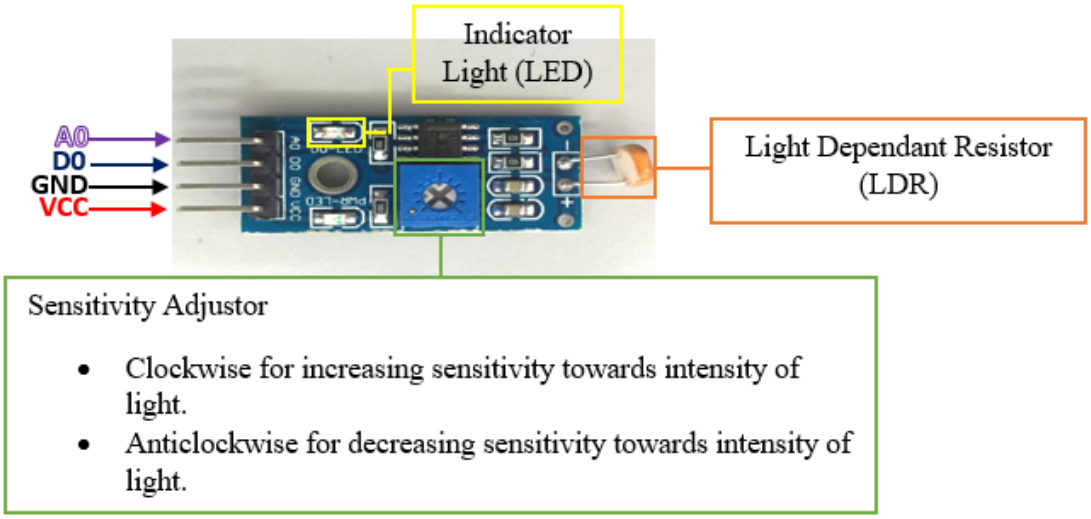


Figure 2.7: LDR Module

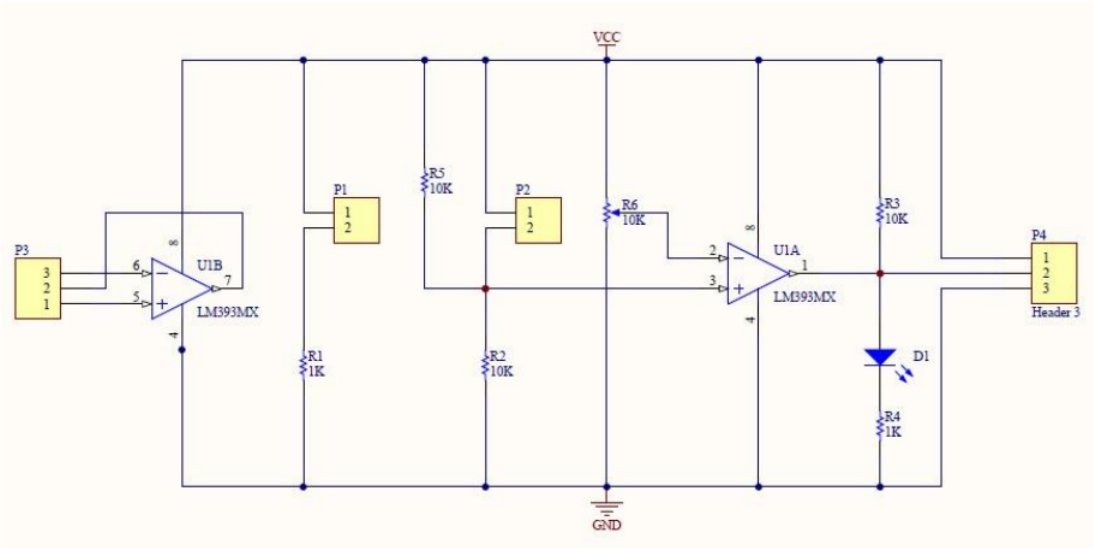


Figure 2.8: LDR Module Schema

2.1.4.1 Connections

1. Connect your Arduino microcontroller to the computer. 2. Connect the VCC pin of your module to the to the 5V pin of your Arduino. 3. Connect the GND pin of your module to the GND pin of your Arduino. 4. Connect the Output pin of your module to the A0 pin of your Arduino. See Fig. 2.9

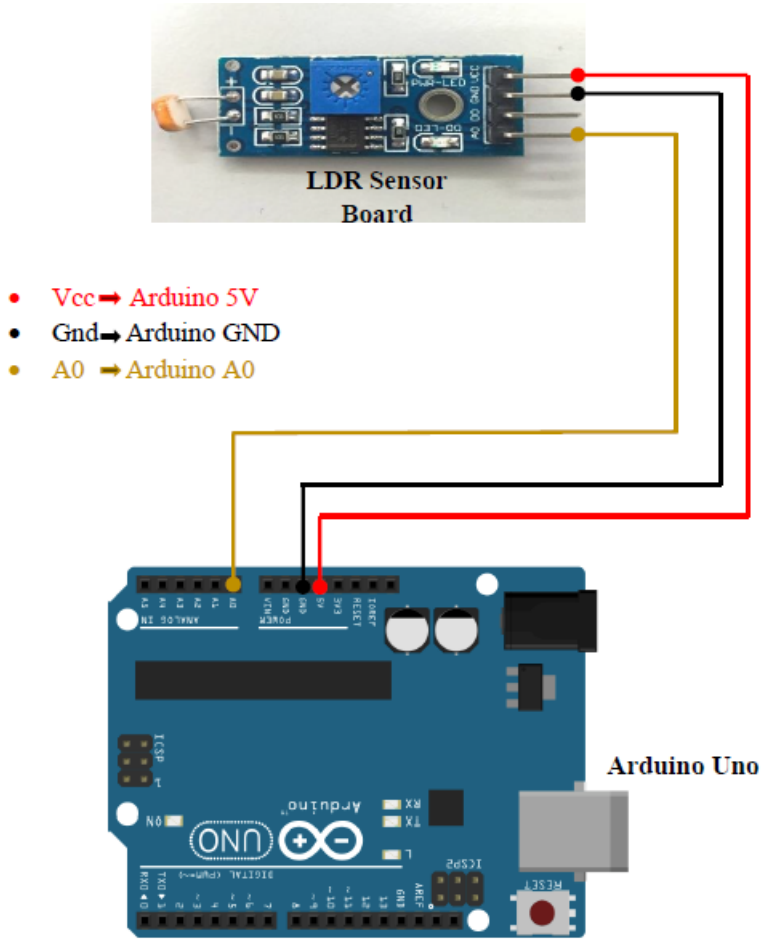


Figure 2.9: LDR Module connection

2.1.5 LCD

LMB162AFC Liquid Crystal Display (LCD) 2×16 character with blue back light as in Fig. 2.10, its role is to display the output results that we want to see from our project.

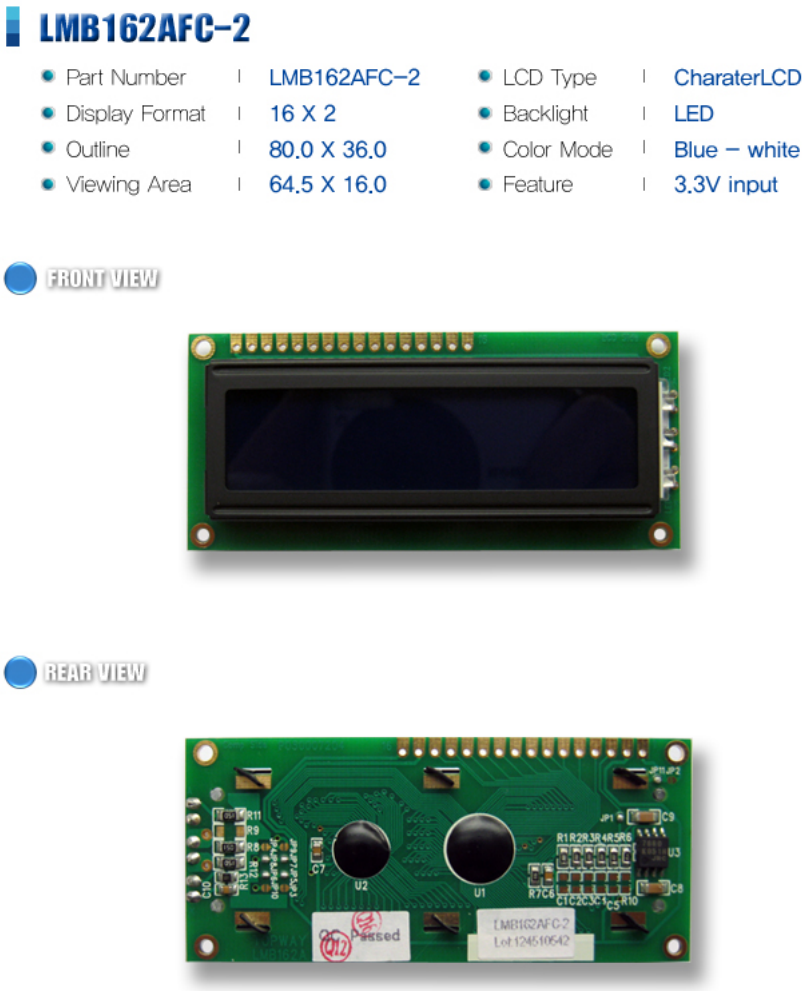


Figure 2.10: LMB162AFC LCD 2×16

2.1.5.1 Connections

It has 11 terminals, 3 GND, 2 Vin and 6 pins connected to the Arduino. Fig. 2.11 and Fig. 2.12 explains it.

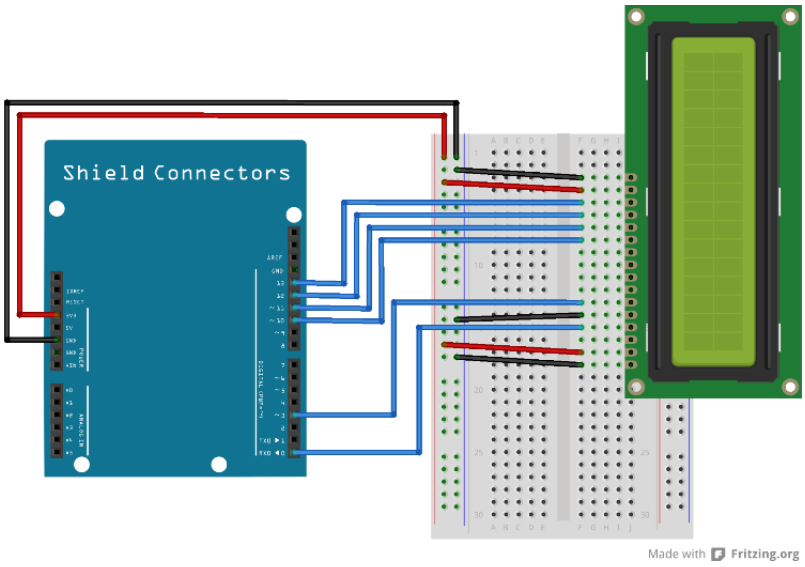


Figure 2.11: LCD 2×16 connected with an Arduino

Pin No.	Pin Name	I/O	Descriptions
1	VSS	Power	Power supply, Ground (0V)
2	VDD	Power	Positive power supply
3	V0	Power	LCD contrast reference supply
4	RS	Input	Register Select RS=HIGH: transferring display data RS=LOW: transferring instruction data
5	R/W	Input	Read / Write Control bus: R/W=HIGH: Read mode selected R/W=LOW: Write mode selected
6	E	Input	Data Enable
7	DB0	I/O	Bi-directional tri-state Data bus In 8 bit mode, DB0 ~ DB7 are in use In 4 bit mode, DB4 ~ DB7 are in use, DB0~DB3 leave open
:	:		
14	DB7		
15	BLA	Power	Backlight positive supply
16	BLK	Power	Backlight negative supply

Figure 2.12: LMB162AFC LCD Terminals Functions

2.1.6 Breadboard

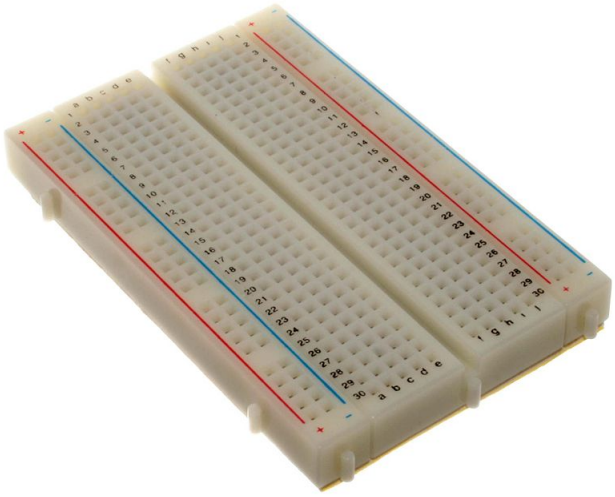


Figure 2.13: Breadboard Overview

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate[4]. See Fig 2.13.

The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically. Fig. explains that 2.14.

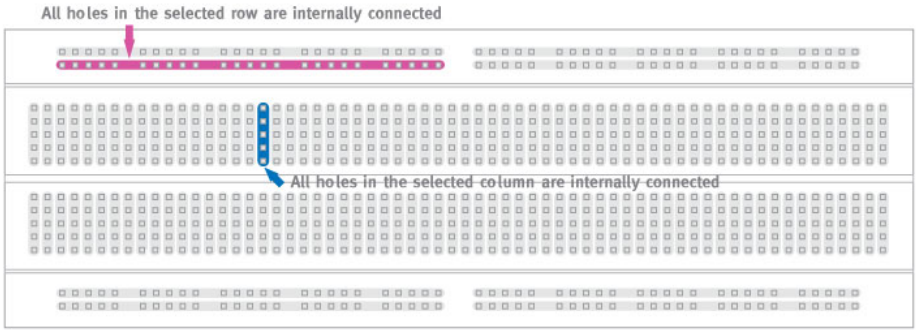


Figure 2.14: Breadboard working scheme

2.1.7 Wires

Figure 2.15: The 3 types of wires that have been used in the project.

Electrical wires were used to do the enter connections between the hardware components in the project. I have used three type of them as in Fig. 2.15 1) Male-Male Jumper wire, 2) Male-Female connecting wire and 3) Female-Female extender wire.

2.1.8 All together

Fig. 2.16 shows all the previous hardware components together.

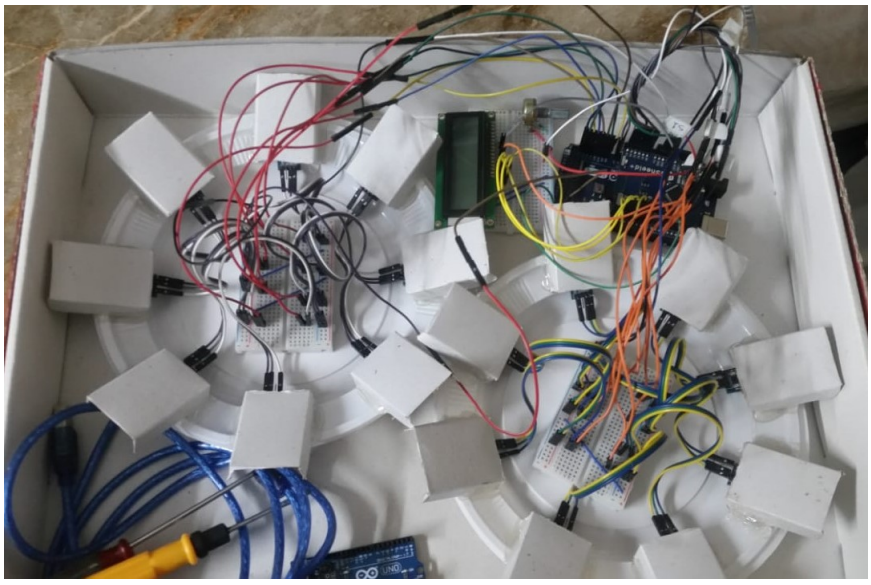


Figure 2.16: Picture for the project's Hardware components.

2.2 Software Part

2.2.1 Arduino IDE Software

2.2.1.1 What is Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards and communicate with them, but also, with the help of third-party cores, other vendor development boards. Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus[6].

2.2.1.2 How it looks

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. see Fig. 2.17

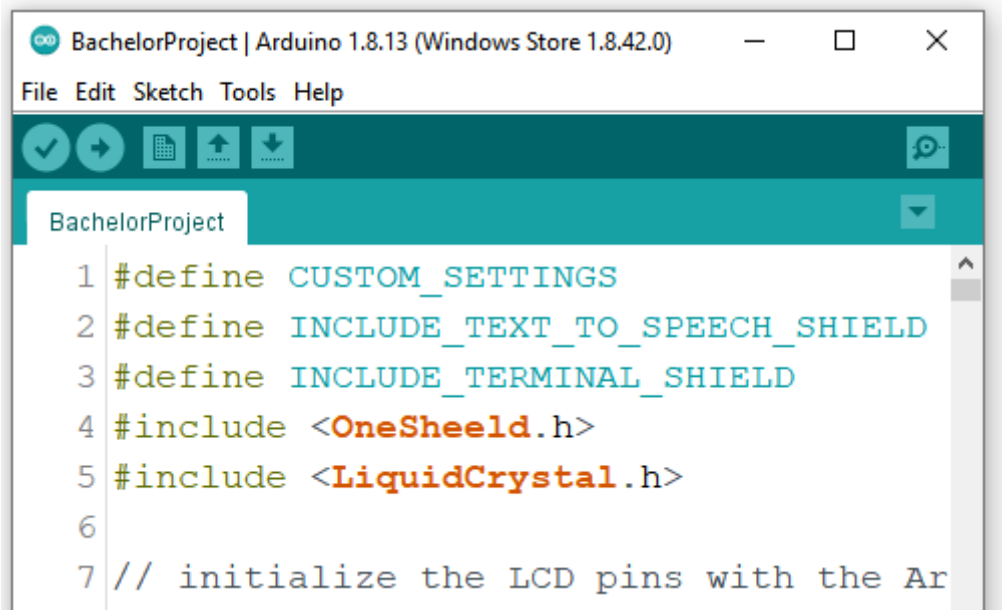


Figure 2.17: Arduino IDE toolbar

2.2.1.3 How to install it

2.2.1.3.1 Download the app

We can get different versions of Arduino IDE from the https://www.arduino.cc/en/software the Arduino Official website. But you must select your software, which is compatible with your own operating system (Windows, IOS, or Linux).

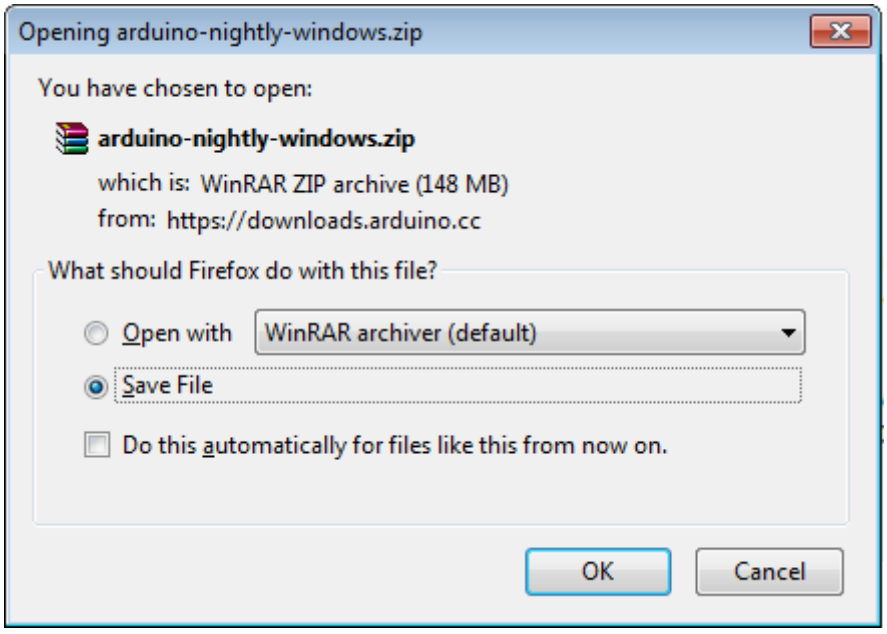


Figure 2.18: Downloading the Arduino IDE

2.2.1.4 How to Launch Arduino IDE

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE[16].

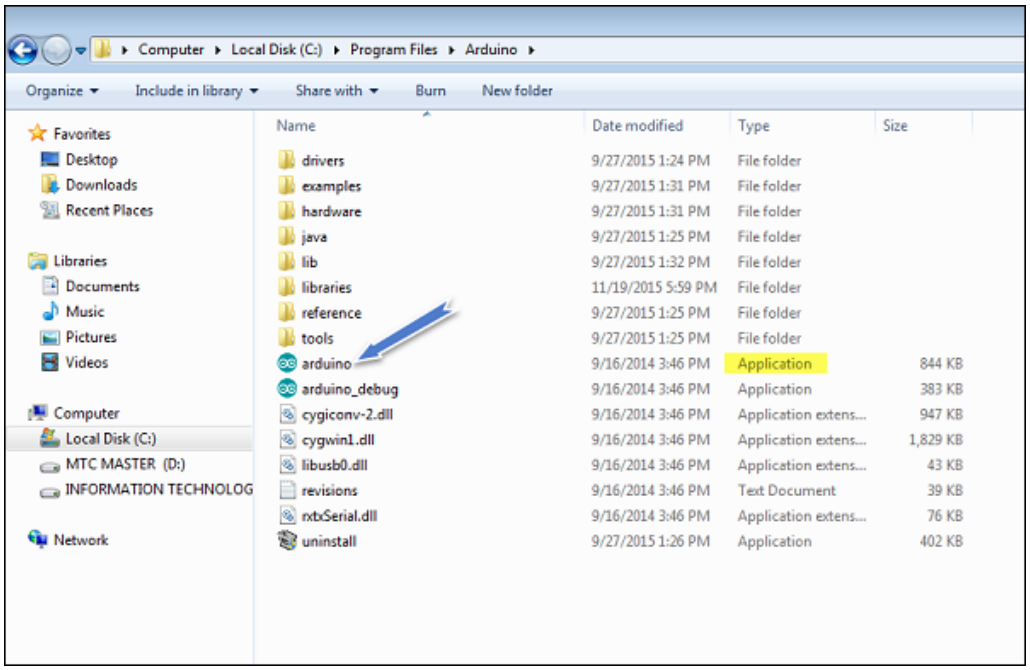


Figure 2.19: Launch the Arduino IDE

2.2.1.5 How to Open your first project

Once the software starts, you have two options: 1)Create a new project, 2) Open an existing project example. To create a new project, select File → New.

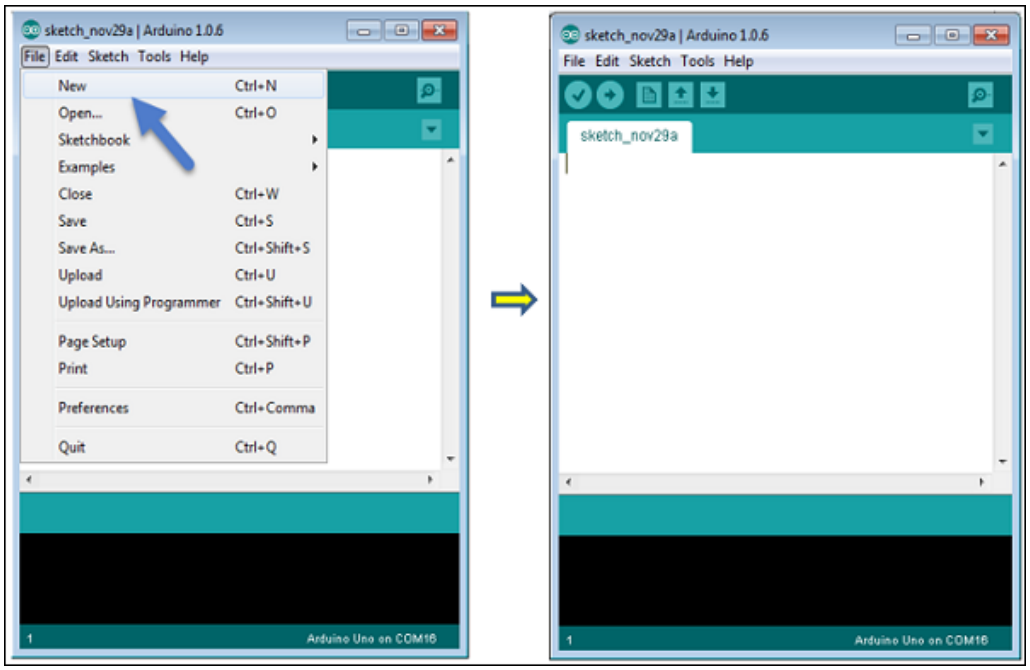


Figure 2.20: Open a new Arduino project

2.2.1.6 How to Select your Arduino board

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer. Go to Tools → Board and select your board.

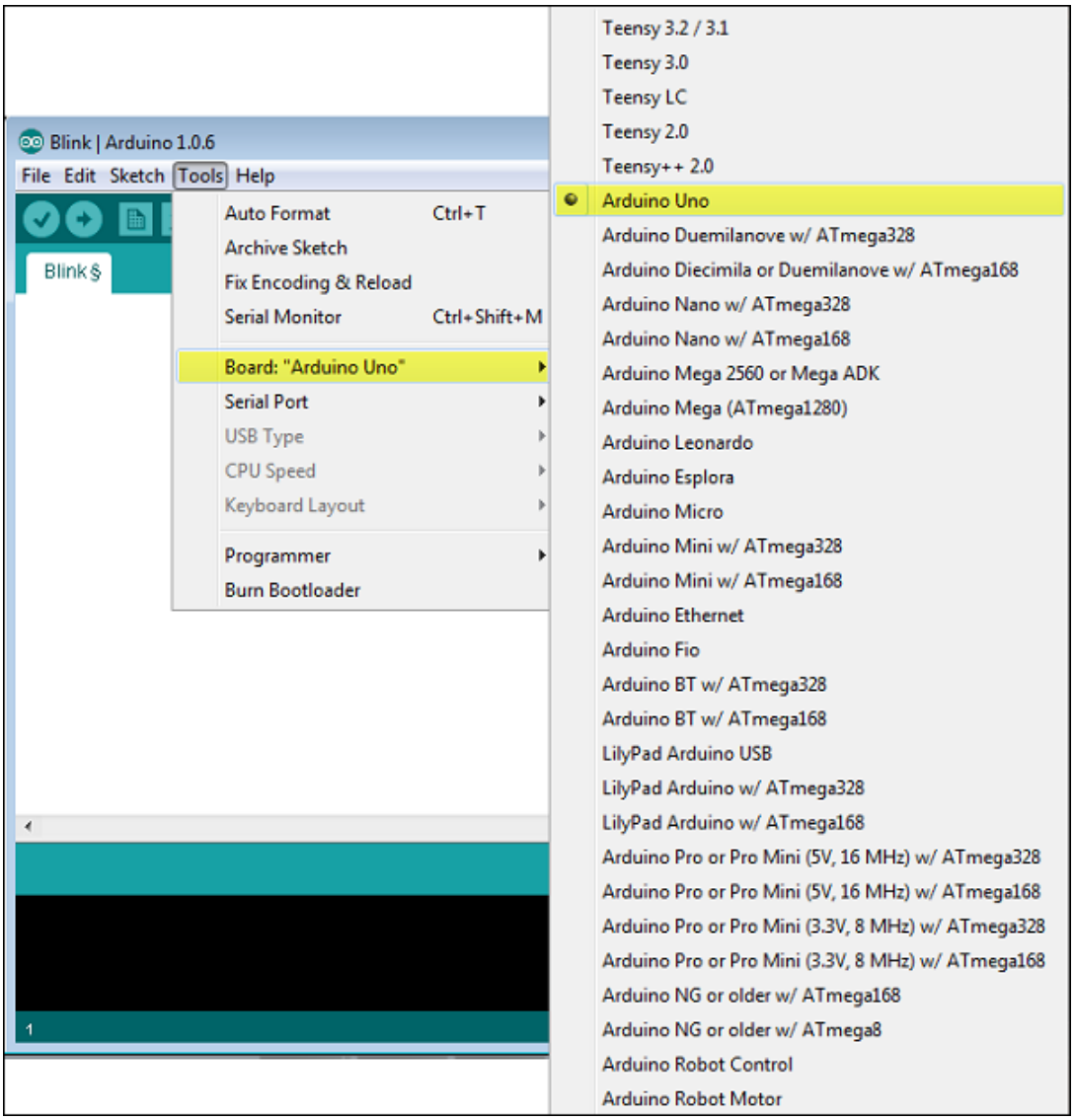


Figure 2.21: Select your Arduino board

2.2.1.7 How to Select your serial port

Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port. see Fig. 2.22

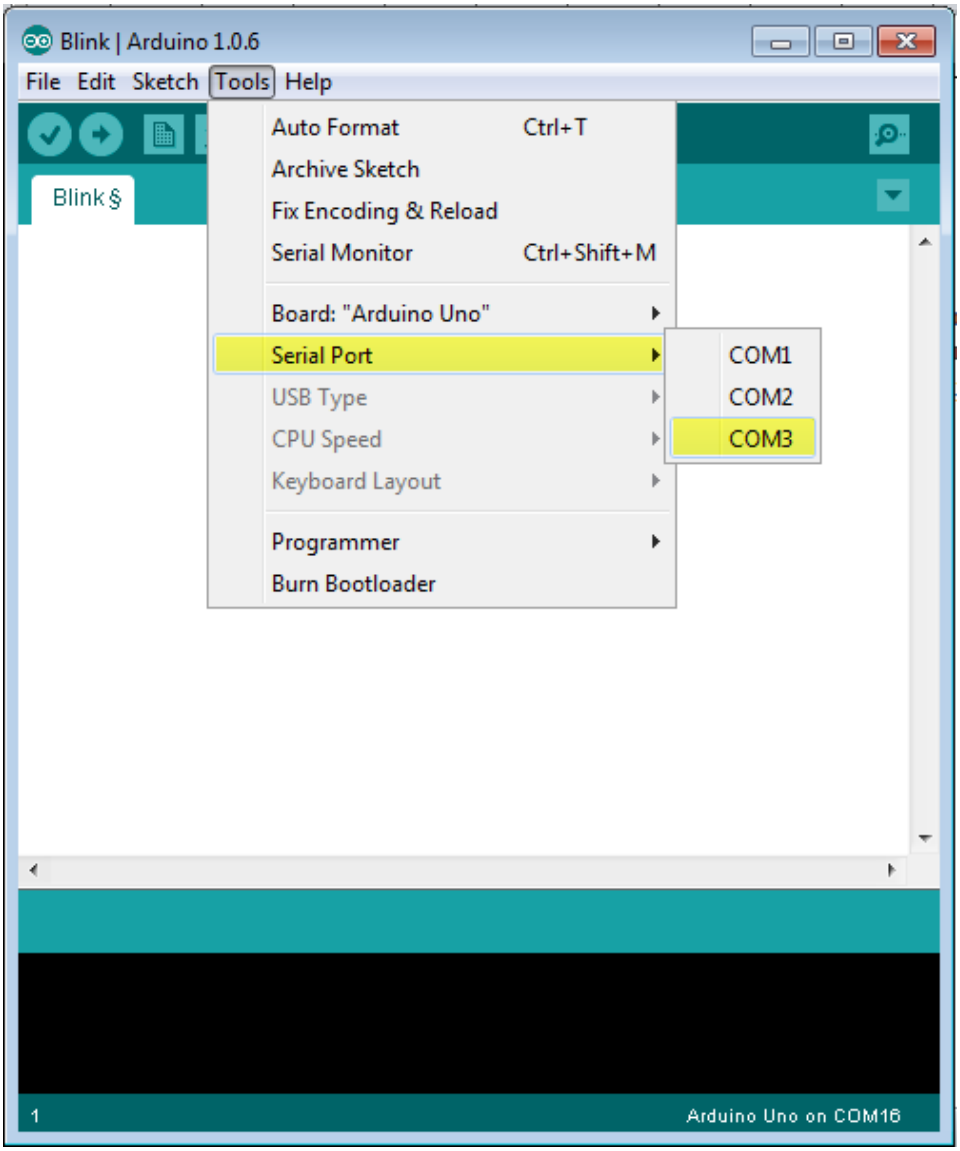
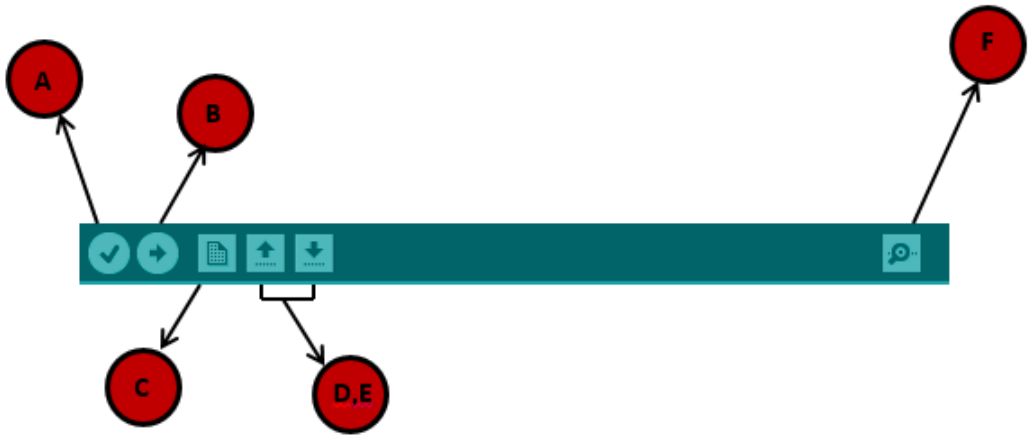


Figure 2.22: Select your Arduino serial board

2.2.1.8 How to Upload the program to your board

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A – Used to check if there is any compilation error.

B – Used to upload a program to the Arduino board.

C – Shortcut used to create a new sketch.

D – Used to directly open one of the example sketch.

E – Used to save your sketch.

F – Serial monitor used to receive serial data from the board and send the serial data to the board.

Figure 2.23: Upload the program to your board

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

2.2.2 1Sheeld+ Mobile App

It is a simple two screens app. The first screen shows you a list of all of the available shields, you can select the shields you want and click next to go to the second screen. In this screen you'll have all of your selected shields on the right and you can toggle between them. See Fig. 2.24

Figure 2.24: 1sheeld App Launched on a smartphone

Chapter 3

Design and Implementation

3.1 Hardware

3.1.1 Main Connections

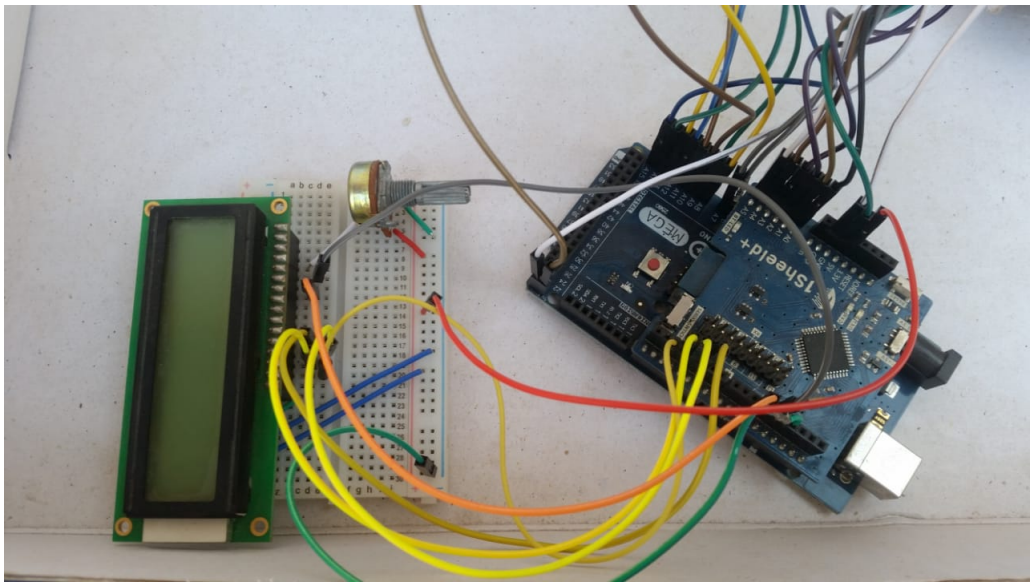


Figure 3.1: Arduino with the 1sheeld and the LCD

As in Fig. 3.1 The Arduino Mega mounts the 1sheeld+ on its top. We can see also the LCD is also mounted on a breadboard. To see how these connection have been done return back to Fig 2.11.

3.1.2 Directions Estimation

3.1.2.1 Cardinal directions

Cardinal directions are the four main points of a compass: north, south, east, and west which are also known by the first letters: N,S,E, and W. These four directions are also known as cardinal points[14]. See Fig. 3.2

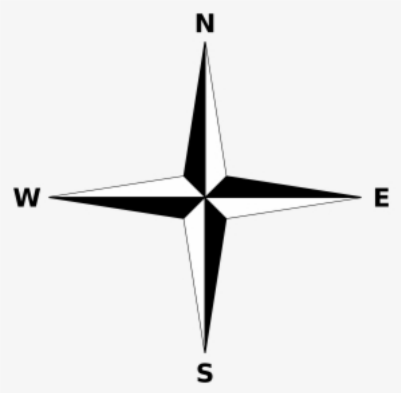


Figure 3.2: Cardinal Directions

Ordinal directions refer to the direction found at the point equally between each cardinal direction.

3.1.2.2 Ordinal directions

Ordinal directions are: northeast (NE), southeast (SE), southwest (SW), and northwest (NW). Ordinal directions are also known as intercardinal directions[14]. See Fig. 3.3

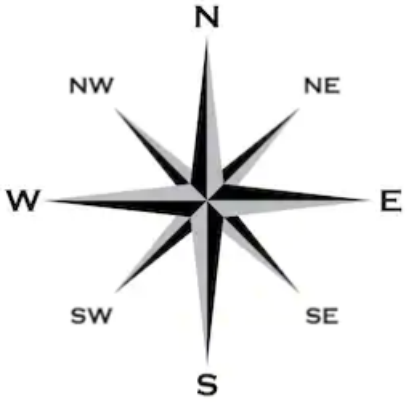


Figure 3.3: Ordinal Directions

In order to estimate both of them correctly, I used 8 LDR sensors and placed them 45 degree away from each other as in Fig. 3.4

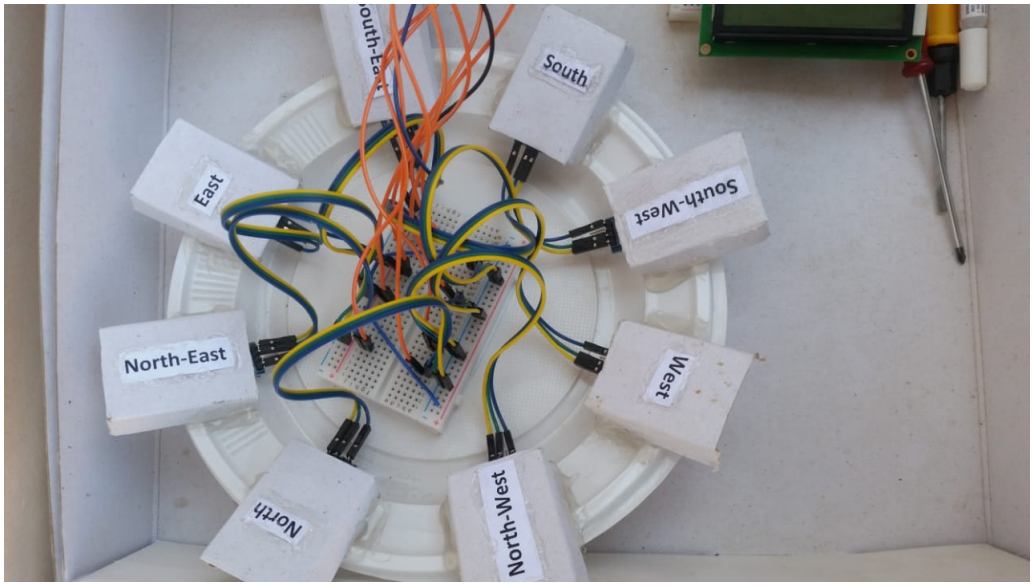


Figure 3.4: 8 LDR sensors fixed on the 8 Directions

3.1.3 Azimuth Estimation

The azimuth angle is the compass direction from which the sunlight is coming. At solar noon, the sun is always directly south in the northern hemisphere and directly north in the southern hemisphere[9]. The azimuth angle varies throughout the day as shown in the animation below. At the equinoxes, the sun rises directly east and sets directly west regardless of the latitude, thus making the azimuth angles 90° at sunrise and 270° at sunset. In general however, the azimuth angle varies with the latitude and time of year and the full equations to calculate the sun's position throughout the day See Fig. 3.5

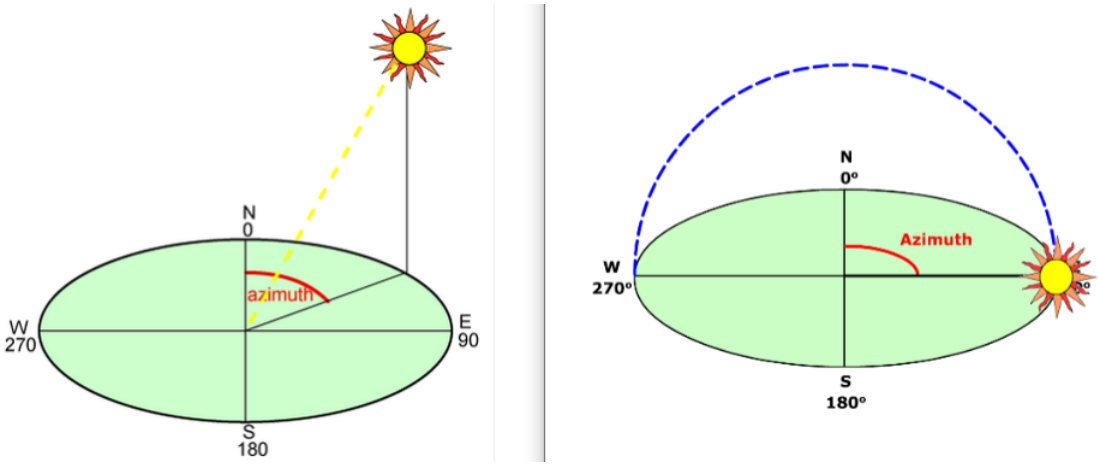


Figure 3.5: Azimuth Angle

In order to estimate it correctly, I used 8 LDR sensors and placed them 45 degree away from each other as in Fig. 3.6 and hanged them perpendicular to the plane of cardinal directions.

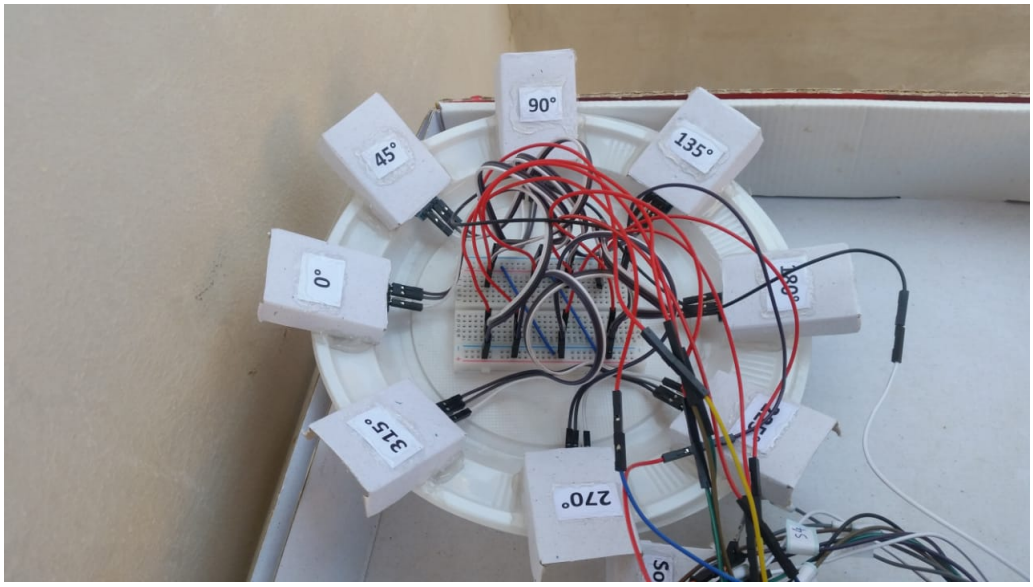


Figure 3.6: 8 LDR sensors fixed on the 8 Directions

3.1.4 LDR Covering and Isolating

In order to get precise reading from the light dependent resistor (LDR), I surrounded it with a U-shape cardboard. The reason of this is I found the LDR sensor was so sensitive in reading all the light that came to it from all the directions and not only those perpendicular rays. For me that was a big problem, because I would not be able to determine the true direction of the incoming light. After deep thinking and long trials, I got the idea that if I can prevent all other incoming light rays except that one that is directly falling on its surface perpendicularly then I can make sure the LDR that will read the highest intensity from the 8 LDR sensors is the one that is truly directed to the light source. See Fig. 3.7

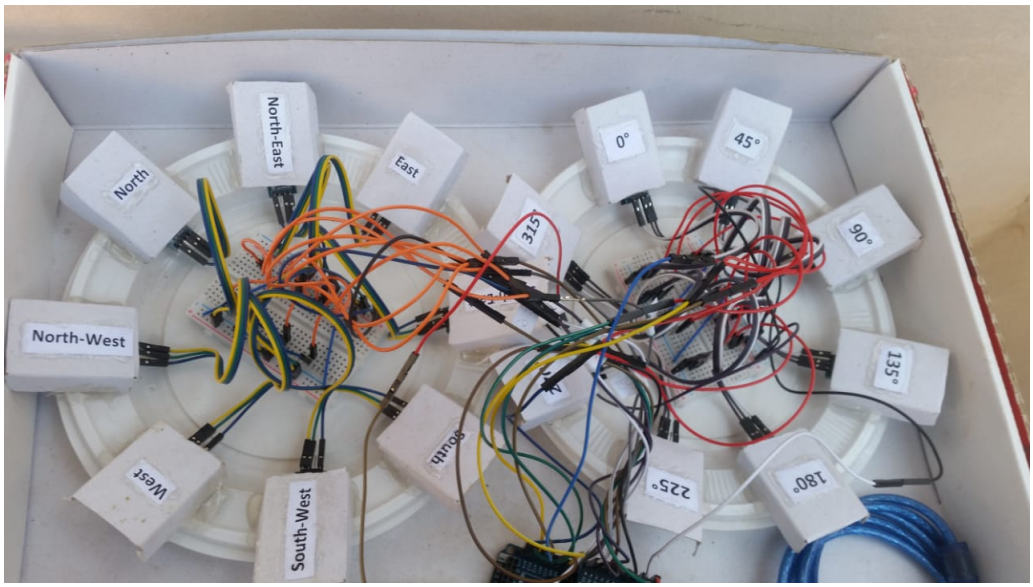


Figure 3.7: The LDR sensors are surrounded by U-shape cardboard for the precise reading

3.2 Software

3.2.1 1sheeld+ App

In order to make our smartphone say the output as a voice message, we should select the shields that are responsible for that. We will choose the Text-to-Voice shield and the Terminal shield[3]. See Fig. 3.8

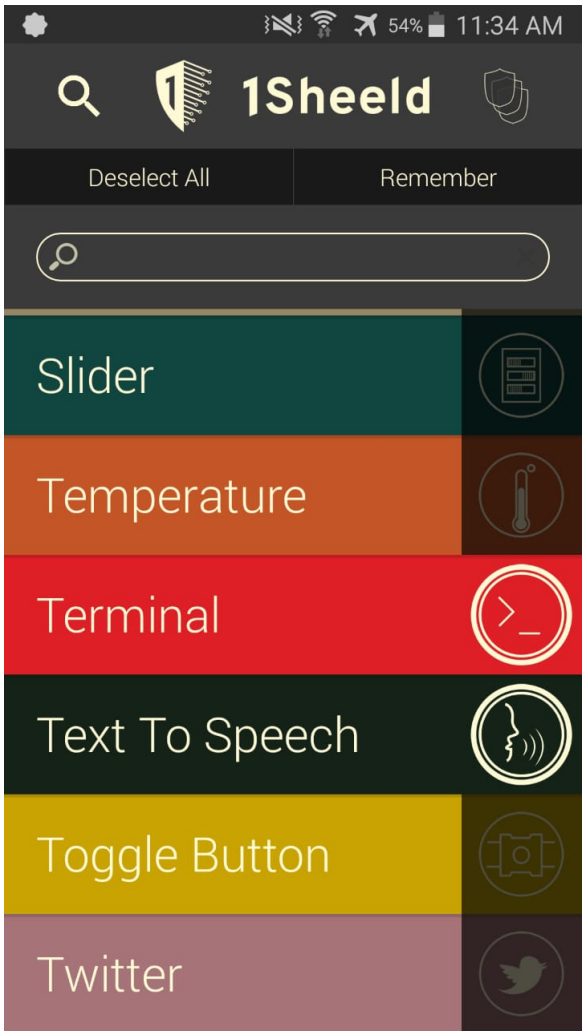


Figure 3.8: 1sheeldApp chosen shields

3.2.2 Project Code

```
#define CUSTOM_SETTINGS
#define INCLUDE_TEXT_TO_SPEECH_SHIELD
#define INCLUDE_TERMINAL_SHIELD
#include <OneSheeld.h>
#include <LiquidCrystal.h>

// initialize the LCD pins with the Arduino pins that is connected to.
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
// String screenData; // the data "msg" that will be written on the LCD
String data; // data that will be taken from the sensors
```

```
String outputMsg;
String directions;
String azimuthAngle;
bool outputSwitching = true;
// Method to write the msg on the LCD in a correct and arranged way
void writeOnLCD(String screenData) {
  if (screenData.length() > 16) {
    lcd.clear();
    lcd.setCursor(0, 0);
    // (note: line 1 is the second row, since counting begins with 0):
    for (int i = 0; i <= 16; i++) {
      lcd.print(screenData[i]);
    }
    // set the cursor to column 0, line 1 after writting the char number 16
    lcd.setCursor(0, 1);
    for (int j = 16 ; j <= screenData.length() - 1 ; j++) {
      lcd.print(screenData[j]);
    }
  }
  else {
    lcd.clear();
    lcd.print(screenData);
  }
}

// Method to sort an Array
void sort(int a[], int size) {
  for (int i = 0; i < (size - 1); i++) {
    for (int o = 0; o < (size - (i + 1)); o++) {
      if (a[o] > a[o + 1]) {
        int t = a[o];
        a[o] = a[o + 1];
        a[o + 1] = t;
      }
    }
  }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  lcd.begin(16, 2); // set up the LCD's number of columns and rows:
  OneSheeld.begin(); // Turn on OneSheeld

}

void loop() {
```

```
int a1 , a2, a3, a4, a5, a6, a7, a8 ;
a1 = analogRead(A8);
a2 = analogRead(A9);
a3 = analogRead(A10);
a4 = analogRead(A11);
a5 = analogRead(A12);
a6 = analogRead(A13);
a7 = analogRead(A14);
a8 = analogRead(A15);
int arrAz[8] = {a1 , a2, a3, a4, a5, a6, a7, a8};
sort(arrAz, 8);

azimuthAngle = "new";

if (arrAz[0] == a1) {
  azimuthAngle = "Azimuth angle = 0 Degree";
}
else if (arrAz[0] == a2) {
  azimuthAngle = "Azimuth angle = 45 Degree";
}
else if (arrAz[0] == a3) {
  azimuthAngle = "Azimuth angle = 90 Degree";
}
else if (arrAz[0] == a4) {
  azimuthAngle = "Azimuth angle = 135 Degree";
}
else if (arrAz[0] == a5) {
  azimuthAngle = "Azimuth angle = 180 Degree";
}
else if (arrAz[0] == a6) {
  azimuthAngle = "Azimuth angle = 225 Degree";
}
else if (arrAz[0] == a7) {
  azimuthAngle = "Azimuth angle = 270 Degree";
}
else if (arrAz[0] == a8) {
  azimuthAngle = "Azimuth angle = 315 Degree";
}

int s1 , s2, s3, s4, s5, s6, s7, s8 ;
s1 = analogRead(A0);
s2 = analogRead(A1);
s3 = analogRead(A2);
s4 = analogRead(A3);
s5 = analogRead(A4);
s6 = analogRead(A5);
s7 = analogRead(A6);
```

```
s8 = analogRead(A7);

int arr[8] = {s1 , s2, s3, s4, s5, s6, s7, s8};
sort(arr, 8);

directions = "new";
if (arr[0] == s1) { // Max is sensor 1
  directions = "North";
}
else if (arr[0] == s2) { // Max is sensor 2
  if (s2 + 75 < s1 && s2 + 75 < s3) {
    directions = "45 Degree North-East";
  }
  else if (s8 - 10 <= s2 && s2 <= s8 + 10) {
    directions = "North";
  }
  else if (s4 - 10 <= s2 && s2 <= s4 + 10) {
    directions = "East";
  }
  else if (s1 <= s3) {
    directions = "North-East";
  }
  else {
    directions = "East";
  }
}
else if (arr[0] == s3) { // Max is sensor 3
  directions = "East";
}
else if (arr[0] == s4) { // Max is sensor 4
  if (s4 + 75 < s3 && s4 + 75 < s5) {
    directions = "45 Degree South-West";
  }
  else if (s2 - 10 <= s4 && s4 <= s2 + 10) {
    directions = "East";
  }
  else if (s6 - 10 <= s4 && s4 <= s6 + 10) {
    directions = "South";
  }
  else if (s3 <= s5) {
    directions = "East";
  }
  else {
    directions = "South-East";
  }
}
else if (arr[0] == s5) { // Max is sensor 5
```

```
    directions = "South";
  }
  else if (arr[0] == s6) { // Max is sensor 6
    if (s6 + 75 < s5 && s6 + 75 < s7) {
      directions = "45 Degree South-West";
    }
    else if (s4 - 10 <= s6 && s6 <= s4 + 10) {
      directions = "South";
    }
    else if (s8 - 10 <= s6 && s6 <= s8 + 10) {
      directions = "West";
    }
    else if (s5 <= s7) {
      directions = "South-West";
    }
    else {
      directions = "West";
    }
  }
  else if (arr[0] == s7) { // Max is sensor 7
    directions = "West";
  }
  else if (arr[0] == s8) { // Max is sensor 8
    if (s8 + 75 < s7 && s8 + 75 < s1) {
      directions = "45 Degree North-West";
    }
    else if (s6 - 10 <= s8 && s8 <= s6 + 10) {
      directions = "West";
    }
    else if (s2 - 10 <= s8 && s8 <= s2 + 10) {
      directions = "North";
    }
    else if (s7 <= s1) {
      directions = "West";
    }
    else {
      directions = "North-West";
    }
  }
  else directions = "Error";

  for (int i = 0; i < 2; i++) {
    if (outputSwitching) {
      String outPut = "smallest value: " + String(arr[0]);
      TextToSpeech.say(directions);
      Serial.println("\n");
      writeOnLCD(directions);
```

```
      Serial.println(outPut);
      outputSwitching = !outputSwitching;

      Serial.println("S1 value: " + String(s1));
      Serial.println("S2 Value: " + String(s2));
      Serial.println("S3 Value: " + String(s3));
      Serial.println("S4 Value: " + String(s4));
      Serial.println("S5 Value: " + String(s5));
      Serial.println("S6 Value: " + String(s6));
      Serial.println("S7 Value: " + String(s7));
      Serial.println("S8 Value: " + String(s8));
      delay(5000);
    }
    else {
      String outPut = "smallest value: " + String(arrAz[0]);
      TextToSpeech.say(azimuthAngle);
      Serial.println("\n");
      writeOnLCD(azimuthAngle);
      Serial.println(outPut);
      Serial.println(azimuthAngle);
      outputSwitching = !outputSwitching;
      Serial.println("A1 value: " + String(a1));
      Serial.println("A2 Value: " + String(a2));
      Serial.println("A3 Value: " + String(a3));
      Serial.println("A4 Value: " + String(a4));
      Serial.println("A5 Value: " + String(a5));
      Serial.println("A6 Value: " + String(a6));
      Serial.println("A7 Value: " + String(a7));
      Serial.println("A8 Value: " + String(a8));
      delay(5000);
    }
  }

  int avg = (1024-s1+1024-s2+1024-s3+1024-s4+1024-s5+1024
      -s6+1024-s7+1024-s8+1024-a1+1024-a2+1024-a3+1024-a4+1024
      -a5+1024-a6+1024-a7+1024-a8)/16;
  Serial.println("The Avg = "+String(avg));
  writeOnLCD("The Avg = "+String(avg));
  TextToSpeech.say("The Average light in this place is "+String(avg));
  Serial.println("\n");
  delay(5000);
}
```

Chapter 4

Analysis And Results

4.1 Experiment 1

Natural light has been used in this experiment as in Fig. 4.1

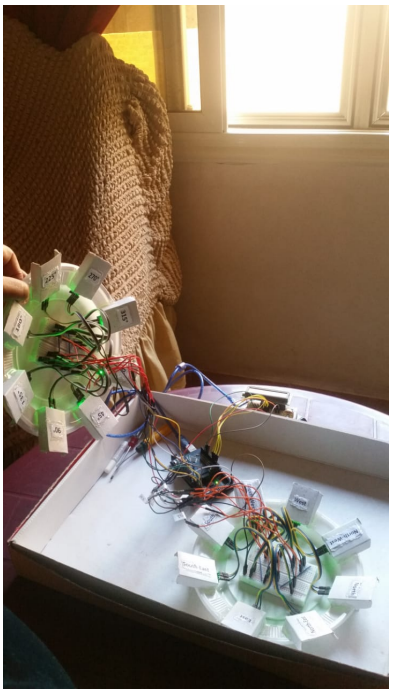


Figure 4.1: Experiment 1 using Natural Light

The output results in Fig. 4.2, Fig. 4.3, and Fig. 4.4

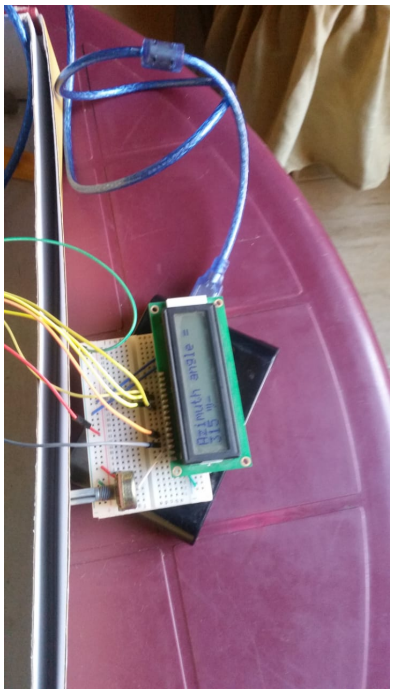


Figure 4.2: Experiment 1 Azimuth Angle Value

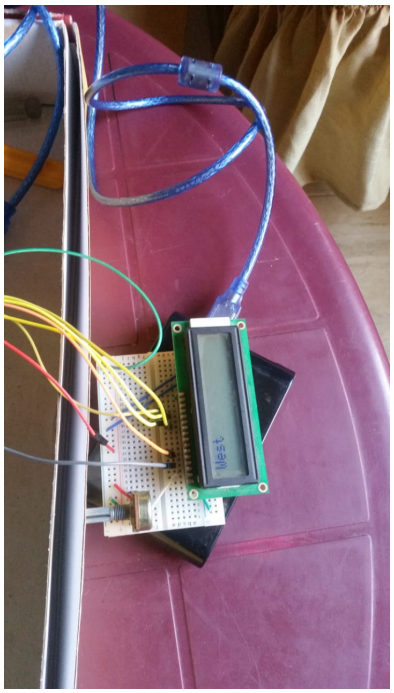


Figure 4.3: Experiment 1 Direction Estimation

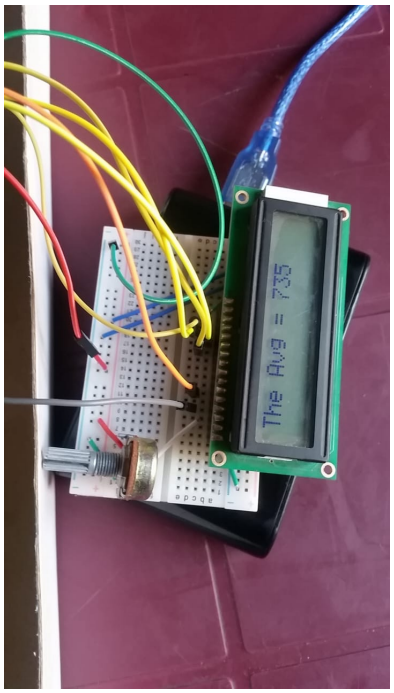


Figure 4.4: Experiment 1 Average Intensity

4.2 Experiment 2

Artificial light has been used in this experiment as in Fig. 4.5

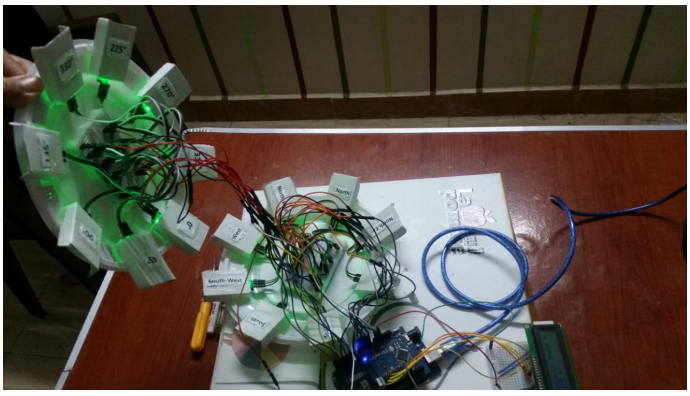


Figure 4.5: Experiment 2 using Artificial Light

The output results in Fig. 4.6, Fig. 4.7, and Fig. 4.8

Figure 4.6: Experiment 2 Azimuth Angle Value

Figure 4.7: Experiment 2 Direction Estimation

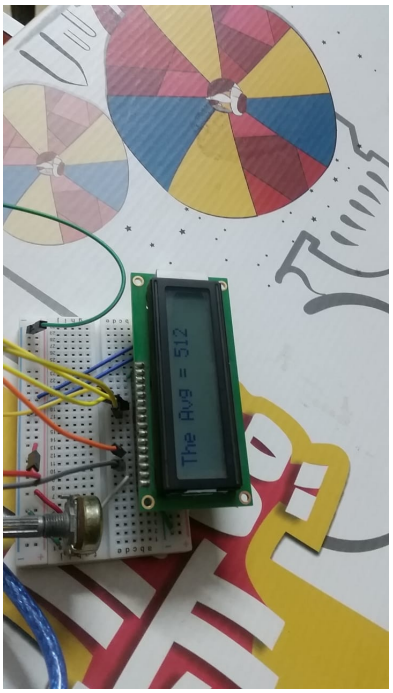


Figure 4.8: Experiment 2 Average Intensity

4.3 Analysis

As we have seen from the last two previous experiments, all the results were as we expected. Moreover, we have achieved all the results we wanted from Light Intensity and Light Directivity to achieve.

On the Exp. 1 the light were coming from the "West" as we saw in Fig. 4.1 and the output result of the direction estimation was also "West" as in Fig. 4.3. The same thing happens with the Azimuth angle, Fig. 4.1 showed that light is coming with an angle equals to 315 degree and output result in Fig. 4.2 was the same.

On the Exp. 2 the light were coming from the "45 degree North-West" as we saw in Fig. 4.5 and the output result of the direction estimation was also "45 degree North-West" as in Fig. 4.7. The same thing happens with the Azimuth angle, Fig. 4.5 showed that light is coming with an angle equals to 225 degree and output result in Fig. 4.6 was the same.

Chapter 5

Conclusion

5.1 Summary

Light intensity and directivity estimation are one of the most challenging problems recently in Computer Vision, Photography, Photo-chemical Reactions, Solar Systems. By knowing the directivity and the intensity of light you will be able to determine where the shadows are and which of those places will be darker or lighter than the others. In photography, this is very useful, as it allows photographers and movie-makers to adjust their cameras' positions, so they can capture the clearest image. Moreover, it allows us to determine the position of any light source even if it is moving and not fixed at a certain position.

The project consists of two parts; Hardware and Software. The hardware part that is responsible for any physical connections between devices and choosing the appropriate components from hardware devices that will help us to get what we want from light intensity and light directivity such as sensors. While the software part is responsible for installing the suitable apps on the PC and the smartphone and writing the code that would be burned on the microprocessor chip of the Arduino board.

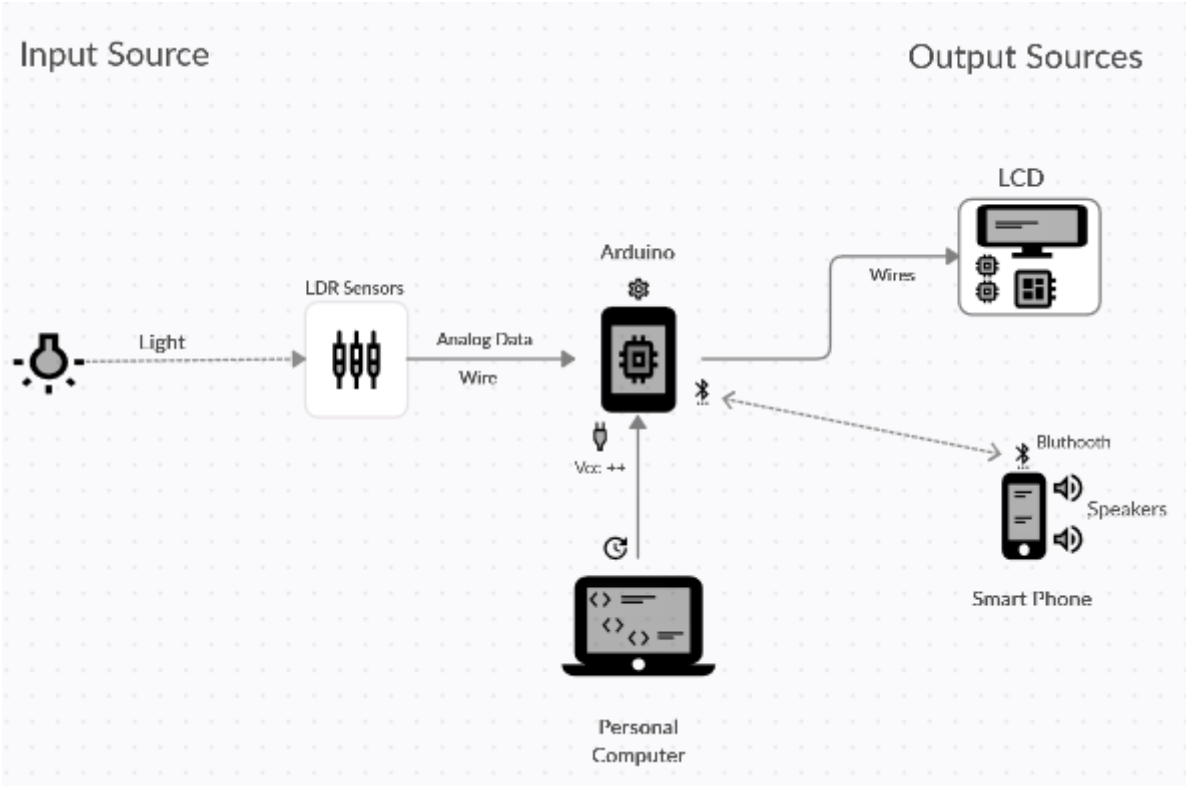


Figure 5.1: Project Flow Diagram

5.2 Limitations

This module is working perfectly with 1 single light source. It gets for you the exact direction of your incoming light which means it gets also the location of your light source pulse the overall average intensity at your location. The limitation of this module is when you have multiple light sources. This module was designed only for one single light source. But what happens when you will have multiple light sources and what will be the output results? The overall average intensity will be calculated estimated perfectly without any errors. But the direction of the light source that will be considered by the module is the strongest of them. i.e. The direction that will have the largest light intensity that will be read by the LDR sensors. That case was not required to be achieved in this project but for the scientific secretariat, it is a limitation on this module to help those who will continue working on my module in the future.

5.3 Future Works

We can enlarge the idea of this module so it can deal with multiple light sources and not just single. I have though about it deeply and reached that hardware components will

stay as it is, we will only try to change the code somehow with more conditions in order to narrow down the choices for our module.

As I mentioned in the previous parts, how much this topic is important and extremely helpful in different fields, it can be taken as it is without any single change in it and be used. It will get the exact light intensity in the location and will estimate for you the direction of the incoming light.

I have mentioned also, that this project idea and code can be taken and used in different fields other than the light field. The only thing that is needed is to replace those LDR sensors with the suitable sensors of your field such as sound sensors, heat sensors, infrared sensors and in communication systems between transmitters and receivers and wait for the amazing results.

Appendix

Appendix A

Lists

List of Figures

Bibliography

[1] Benjamin de Mayo. *Introduction to waves*. Morgan and Claypool, March 2015.

[2] Daniel Fleisch. *A Student's Guide to Waves*. Cambridge Universitys, April 1, 2005.

[3] https://1sheeld.com/.

[4] https://en.wikipedia.org/wiki/Breadboard.

[5] https://socratic.org/questions/52dfea.

[6] https://www.arduino.cc/en/guide/environment.

[7] https://www.bbc.co.uk/bitesize/guides/z96sdmn/revision/4.

[8] https://www.britannica.com/science/photochemical reaction.

[9] https://www.pveducation.org/pvcdrom/properties-of-sunlight/azimuth angle.

[10] https://www.rohm.com/electronics basics.

[11] Maximilian Sender and Dirk Ziegenbalg. Light Sources for Photochemical Processes – Estimation of Technological Potentials. *Foreign Language World*, 89, 2017.

[12] Gary Waldman. *Introduction to Light: The Physics of Light, Vision, and Color*. Dover Publications, June 14, 2002.

[13] Martin Weber and Roberto Cipolla. A Practical Method for Estimation of Point Light-Sources. *Foreign Language World*, 89, 2017.

[14] www.geographyrealm.com/cardinal-directions-ordinal directions/.

[15] www.instructables.com/LDR Sensor-Module-Users-Manual-V10/.

[16] www.tutorialspoint.com/arduino/.