INTRODUCTION

Visible light communication is a viable technology to accommodate the need for faster and better wireless communications in the coming years. The basic idea, is that instead of using traditional methods of communication over cables or radio frequencies, VLC systems send data by turning light on (logic 1) and off (logic 0). This report describes and evaluates the visible light communication system design the team created.

The first part of the process is preparing a file or string of bytes for transmission. In order to synchronize the transmitter and receiver, the system divides the data into units called "frames", each starting with a preamble to let the receiver know that a transmission has started. The transmitter takes a file, breaks into frames, and inserts preamble sequences before each frame. Then it sends the modified file to a microcontroller unit (MCU) over the serial port. The MCU controls the gate of a transistor based on the data it receives, switching an array of LEDs on when it sees a 1 and turning it off when it sees a 0. This light is picked up by an array of photodiodes on the receiver side

By VLC we tend to be referring to an illumination source (e.g. a light bulb) which in addition to illumination can send information using the same light signal. So in our terms:

VLC = Illumination + Communication

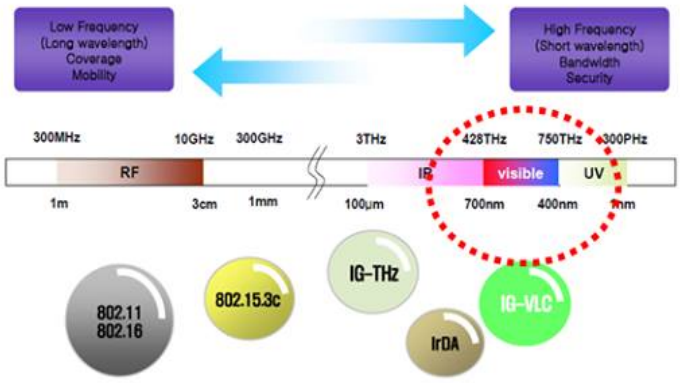
Imagine a flash light which you might use to send a morse code signal. When operated manually this is sending data using the light signal, but because it is flashing off and on it cannot be considered to be a useful illumination source, so it is not really VLC by our definition. Now imagine that the flash light is switched on and off extremely quickly via a computer, then we cannot see the data and the flash light appears to emitting a constant light, so now we have illumination and communication and this does fits our definition of VLC. Of course we would need a receiver capable of receiving the information but that is not too difficult to achieve.

In literal terms any form of information that can be sent using a light signal that is visible to humans could be considered to be VLC, but by our definition we should be able to see the light, but cannot “see” the data.

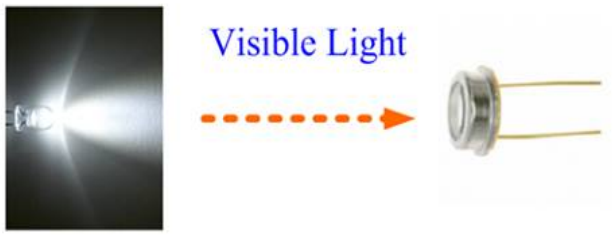
1.1. PURPOSE

VLC System has the following purposes which it aims to fulfill

1- **Security:** VLC is use light communication and it's visible so in this case it's easy to determine who can receive the message and it's impossible to tap the communication without breaking the link.
2- **Human Safety:** VLC doesn't effect at the human body. Thus, the transmission power can be kept high if needed.
3- **Bandwidth**: VLC has a bandwidth range from 430 THz to 750 THz and this range is larger than the bandwidth in the RF Communications from3 kHz to 300 GHz.
4- **High Data Rates.**
5- **Unlicensed Spectrum**: No company owns property rights for visible light and thus no royalty fees have to be paid nor does expensive patent-license have to be purchased in order to use visible light for communication purposes **[5]**.
6- **Ubiquitous Nature**: visible light is present in many places, sothere is the opportunity to combine light communication withlighting design to let Visible Light Communication (VLC) coexist with the lighting setup present inmany offices, homes, or institutions.



- IG-THz : 300 GHz to 10 THz (contribution 15-07-0623-01)
- 802.15.3c : 57 GHz to 64 GHz
- IrDA : 334THz(900nm) to 353THz (850nm)



Transmitter device Receiver device

- **VLC (Visible Light Communication): NEW wireless technique using "Visible Light"**
- **Wavelength: Between ~400nm (750THz) and ~700nm (428THz)**
- **General Characteristic**
  - Visibility : Aesthetically pleasing
  - Security : What You See Is What You Send.
  - Health : Harmless for human body
  - Unregulated : no regulation in optical frequency
  - Using in the restricted area : aircraft, spaceship, hospital
  - Eye safety

1.2. **Potential Application & Scope**

We hope that the achievement of this project in reality will make everyone in everywhere and at all times be able to send and receive text, image, audio and video with lowest cost, reasonable rate and more security without using the internet. Light is glowing ubiquitously, opening many ways to VLCS. There are different areas where VLCS can offer communication services without interfering and overcrowding the radio spectrum. Such area includes but not limited to traffic lights, television screens, and hospitals etc.

A. Traffic Lights

Upon red in traffic signals, pedestrian and drivers are idle and this kills their precious time during travelling. The VLCS provides an opportunity to connect the travelers to the stop/traffic light through their mobile phones or car head light for obtaining some kind of information. This system provides another means of accessing information free of cost.

B. Television Set

Unlike a stop light, hundreds of pixels that are continuously glowing in television set to project an image to the audiences. Among the many LEDs it is possible to dedicate a few to communication via VLCS. This system will ease the user in getting information of what is broadcasting on other channels without toggling the current channel and internet connection.

C. Hospitals

There are thousands reasons to adopt wireless technology in hospitals. For example, updating and monitoring patient records, viewing health condition and medical images of patients. Accuracy and in-time delivery of information in health-care systems is the main concern. Other important concerns are the data confidentiality of patients, and interference from wireless devices may cause medical devices to shutdown or produce inaccurate results. The VLCS do not permit mobility and cannot communicate through barriers, hence making communication secure. The VLCS is a better choice to be installed at hospitals, because it does not use Industrial Scientific and Medical band and will not be interfering with medical devices.

WHAT IS VISIBLE LIGHT COMMUNICATION?

Visible Light Communications is essentially communication by means of optical light. It falls under the category of free-space optical communications. Transmitting data via light is achieved by having the light source flicker on and off to represent a logic high and logic low signal respectively. A receiver (either photodiodes or a digital camera) will detect the light coming from the transmitter and will interpret the signal. When the receiver detects light, it is represented as a logic high and when it detects no light at all from the transmitter, it is represented as a logic low. By turning the light on and off, the transmitter can transmit 0s and 1s. This is the simplest method that visual light can be used for digital communication. Varying levels of light between on and off could allow for the transfer of more than one bit of information.

Any light technically could be used to transfer data but what matters the most is the brightness and the frequency of the light at which it modulates. The data rate of the transmission will depend on how fast the lights can turn on and off. LEDs are a popular choice for VLC communication as they can be switched on and off at a very high speed. Fluorescent lights used indoors can also be used as they flicker at a speed that is fast enough that the human eye cannot see. There is one issue with fluorescent lights however. While they could be used for communications, they can only do so at relatively low frequencies, due to the fact that fluorescent bulbs cannot be turned on and off at high speed. The resulting transmission rate would be approximately 10 kbps. This rate is not high enough to support the transmission of data such as video or audio, while LEDs, have a much faster switching speed, as they are capable of providing up to 500 Mbps and possibly even more.

Currently, one major VLC system is operating within a few countries. Named the RONJA (*Reasonable Optical Near Joint Access)* system, it can transmit at speeds of 10 Mbps while transmitting light to the receiver at a distance of nearly 1 mile apart. It makes use of a high brightness LED and shines it through loupe lens to increase the brightness. This allows for a longer transmitting distance. The RONJA system is a free technology project and was developed by Twibright Labs. The purpose of this system is to network neighbouring houses with cross-street Ethernet access, extend the internet connection to houses close by that are not covered by ISPs by the last mile, or just provide a link layer for fast neighbourhood mesh networks.

Currently, one major VLC system is operating within a few countries. Named the RONJA (*Reasonable Optical Near Joint Access)* system, it can transmit at speeds of 10 Mbps while transmitting light to the receiver at a distance of nearly 1 mile apart. It makes use of a high brightness LED and shines it through loupe lens to increase the brightness. This allows for a longer transmitting distance. The RONJA system is a free technology project and was developed by Twibright Labs. The purpose of this system is to network neighbouring houses with cross-street Ethernet access, extend the internet connection to houses close by that are not covered by ISPs by the last mile, or just provide a link layer for fast neighbourhood mesh networks.

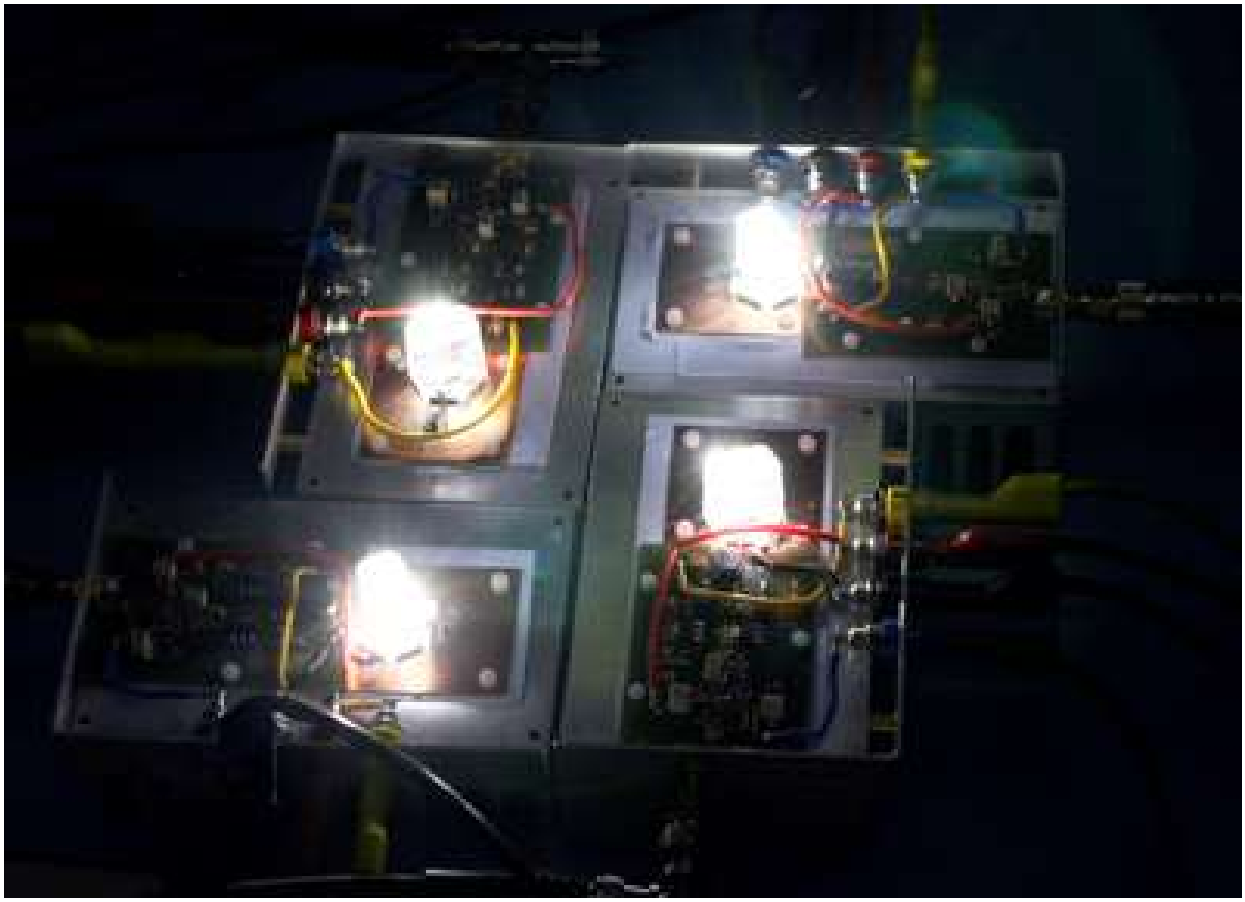
With VLC technology improving and becoming more prominent in the world, Li-Fi, a subset of VLC is becoming more of a reality. Li-Fi is built upon VLC and is a high-speed bidirectional wireless network similar to Wi-Fi. This is achieved by having a transmitter and receiver built in each device where it transmit light and receives incoming light from another device. However, most VLC systems today are unidirectional. As time progresses, more bidirectional VLC systems will emerge.

Recently, a company named Axrtek launched a bidirectional RGB LED VLC system. Named the MOMO, it can transmit at a high bandwidth speed of 300 Mbit/s at a range of up to 25 feet.

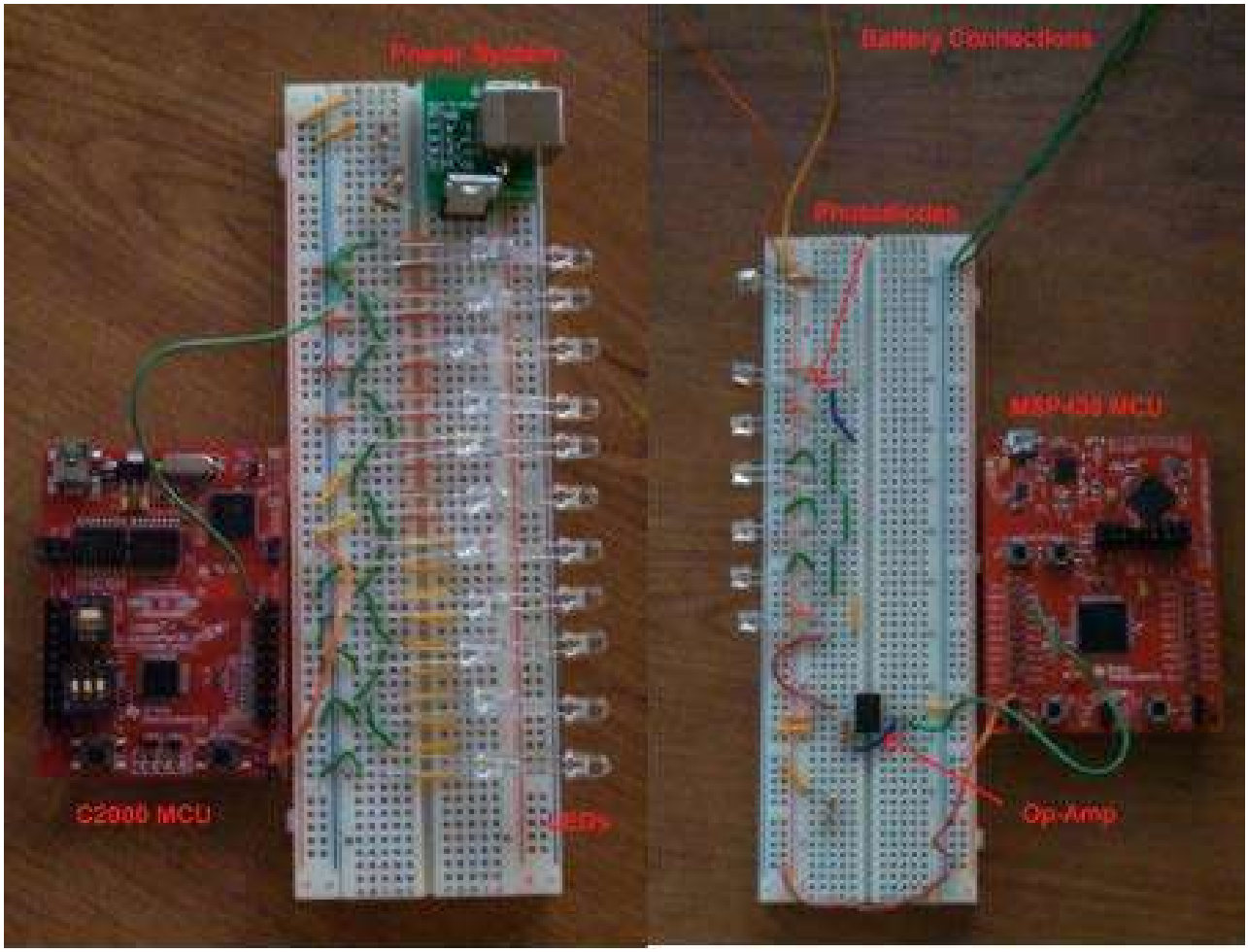
HISTORY

The first account of transmitting information via visible light communication was back in the 1880s where Alexander Graham Bell invented the photophone. It was an analog phone system, which could transmit voice via modulated light . The brightness of the light used in the photophone is altered by changing the position of the mirror, which changes accordingly to the sound waves from the user of the phone. On the receiver side, a selenium cell with a parabolic mirror was used. The resistance of the cell would change inversely proportional to the amount of sunlight that hits the cell. The selenium cell would replace the carbon microphone and alter the current flowing through a regular telephone.

More recently, in 2003, a team at the Nakagawa Laboratory located at Keio University in Japan used LEDs to transmit data via visible light . This is the first account of transmitting digital information via LEDs. In 2010, a collaboration between researchers from Siemens and from Heinrich Hertz Institute in Berlin were able to transmit at a speed of 500 Mbit/s over a distance of 5 meters and transfer at 100 Mbit/s over an even longer distance with 5 white LEDs manufactured by Ostar .



In July 2011, at TED Global, there was a demonstration of the D-light project, a VLC project led by Harald Haas, a professor at the University of Edinburgh .The demonstration showed a HD video being transmitted from a standard LED lamp. The data rate of the VLC system was approximately 10 Mbps which is roughly the data rate of a DVD playing back. As time passes by, VLC is expected to grow immensely and is expected to be a possible method of providing the Internet of Things.



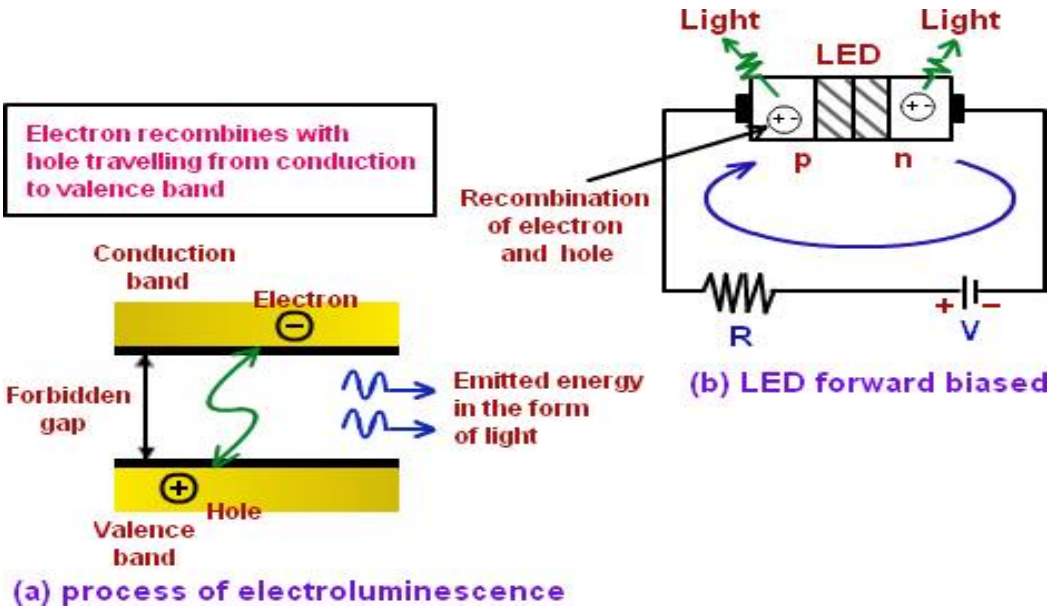
COMPONENTS USED

4.1. Light Emitting Diode (LED)-

The purpose of the LEDs in the VLC system is to provide light to be used to transmit data. LEDs accomplish this by turning on which represents a logic 1 and turning off which represents a logic 0. In order to be successful in attaining our goals, we needed to make sure that the LEDs we selected were bright and can switch at a high frequency. Without LEDs capable of those features, the transmitting distance and data rate would be small. This would cause the transmission of audio to be difficult, especially with ambient light present.

In order to choose the most optimal LEDs, the team considered the important factors shown in *Table 1* below. The team organized the importance of each feature in a table to aid in the selection of the best LED. For the VLC project, there are two goals: to transmit CD quality audio from the transmitter to the receiver and have it transmitted across one meter. With those two goals in mind, the brightness and frequency speed are both important to us when choosing a LED.

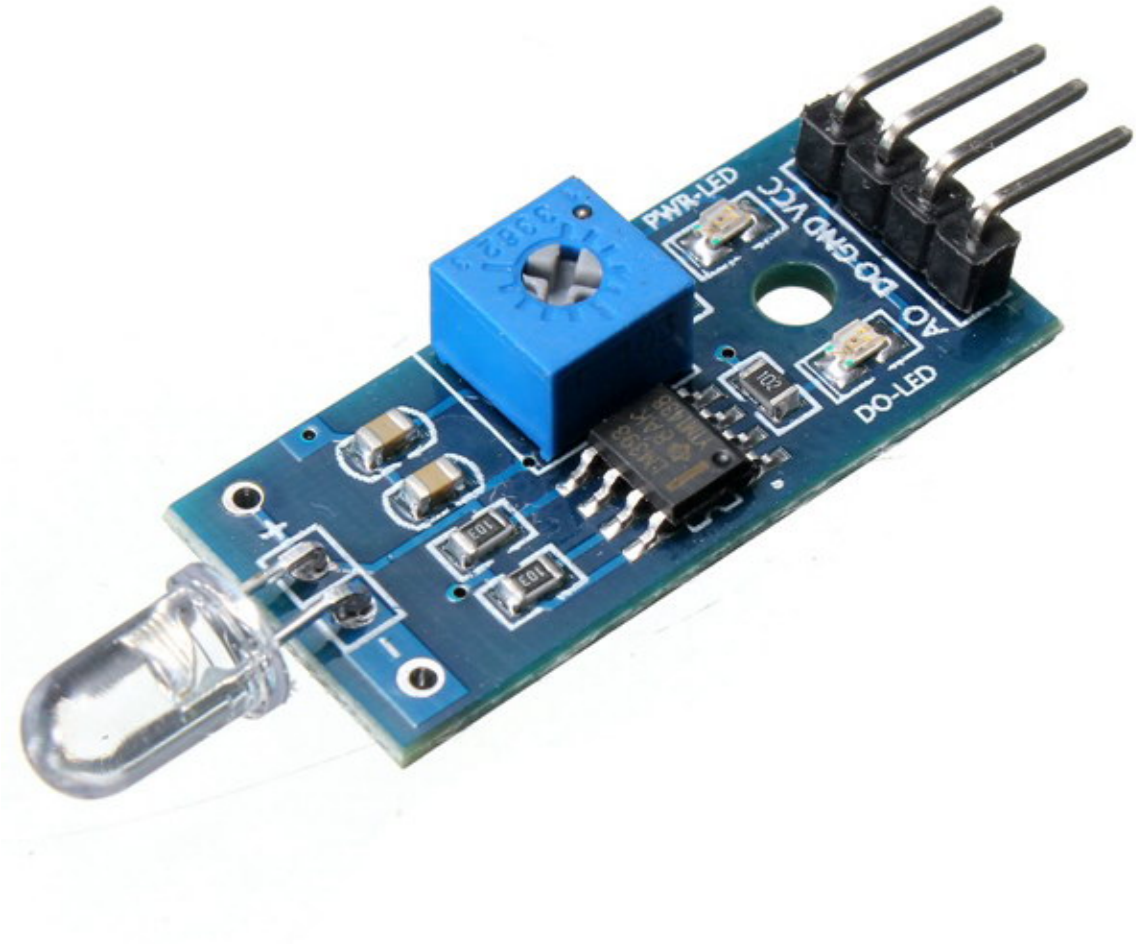
A high level of brightness is preferable so that data can be transmitted reliably under ambient light and across larger distances. The frequency speed is also crucial. In order to transmit data quickly enough, the LEDs have to turn on and off quickly. A typical song from a CD (16-bit resolution) has a frequency of 44.1 KHz. By multiplying the sampling frequency by 16 bits, we get 705,600 bits per second or 705.6 kbps. Having a high frequency LED is very important if transmitting uncompressed music via light. A transmission rate of at least 1 Mbps would be needed to accomplish that.



4.2. PHOTODIODE –

A **photodiode** is a semiconductor device that converts light into current The current is generated when photons are absorbed in the photodiode. A small amount of current is also produced when no light is present. Photodiodes may contain optical filters, built-in lenses, and may have large or small surface areas. Photodiodes usually have a slower response time as their surface area increases. The common, traditional solar cell used to generate electric solar power is a large area photodiode.

Photodiodes are similar to regular semiconductor diodes except that they may be either exposed (to detect vacuum UV or X-rays) or packaged with a window or optical fibre connection to allow light to reach the sensitive part of the device. Many diodes designed for use specifically as a photodiode use a PIN junction rather than a p–n junction, to increase the speed of response. A photodiode is designed to operate in reverse bias.



When selecting the photodiodes, a number of features needs to be taken into account. These features are listed in *Table 2*.

Category	**Importance**	**Desirable**	**Undesirable**
Response Time	11	< 10 ns	> 100 ns
Wavelength Range	3	Difference of 700 nm or more	Difference of 300 nm or less
Price	2	Less than $0.50	More than $1
Field of Vision	4	> 30°	< 10°

Each of the features was rated for its importance. The most important feature was the response time. If the photodiode was incapable of detecting the flashing of the LED fast enough, then it would be incapable of meeting our design requirement. A response time of less than 10 ns is desirable as a shorter response time means that the photodiodes can react faster to each bit that is transmitted. With a shorter response time, the photodiodes can support faster transmission rates. With a transmission speed of 1 MHz, the receiver would need to be accurate and be able to account for each bit. A longer response time would lead to a higher chance of inaccuracies when receiving data. When the LEDs are transmitting at a frequency of 1 MHz, a response time of 10 ns or less would not have a huge effect on the signal being received.

The second most important feature was price. Pricing is important because the intended design would utilize a large number of photodiodes strung together in parallel. This design feature was chosen due to the very low current and small viewing angle of each photodiode. Connecting multiple photodiodes together would alleviate this problem so cheap photodiodes are strongly desirable. The third important feature is the range of wavelengths that the photodiode can detect. The larger the range, the easier the light of the selected LEDs would be able to pick up. Field of vision was also chosen as another important feature. A viewing angle that was too narrow might be incapable of identifying light from the LEDs. Knowing what the desired features were allowed a scale of what was suitable and undesirable to be produced. This scale was then used to categorize all of the photodiodes that were researched.

4.3. Universal Serial Bus (USB)

Used to communicate via the USB protocol with a host computer (for programming or sending/receiving serial data).

USB, short for **Universal Serial Bus**, is an industry standard initially developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices. It is currently developed by the USB Implementers Forum (USB IF).

USB was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has become commonplace on other devices, such as smartphones, PDAs and video game consoles. USB has effectively replaced a variety of earlier interfaces, such as serial ports and parallel ports, as well as separate power chargers for portable devices.

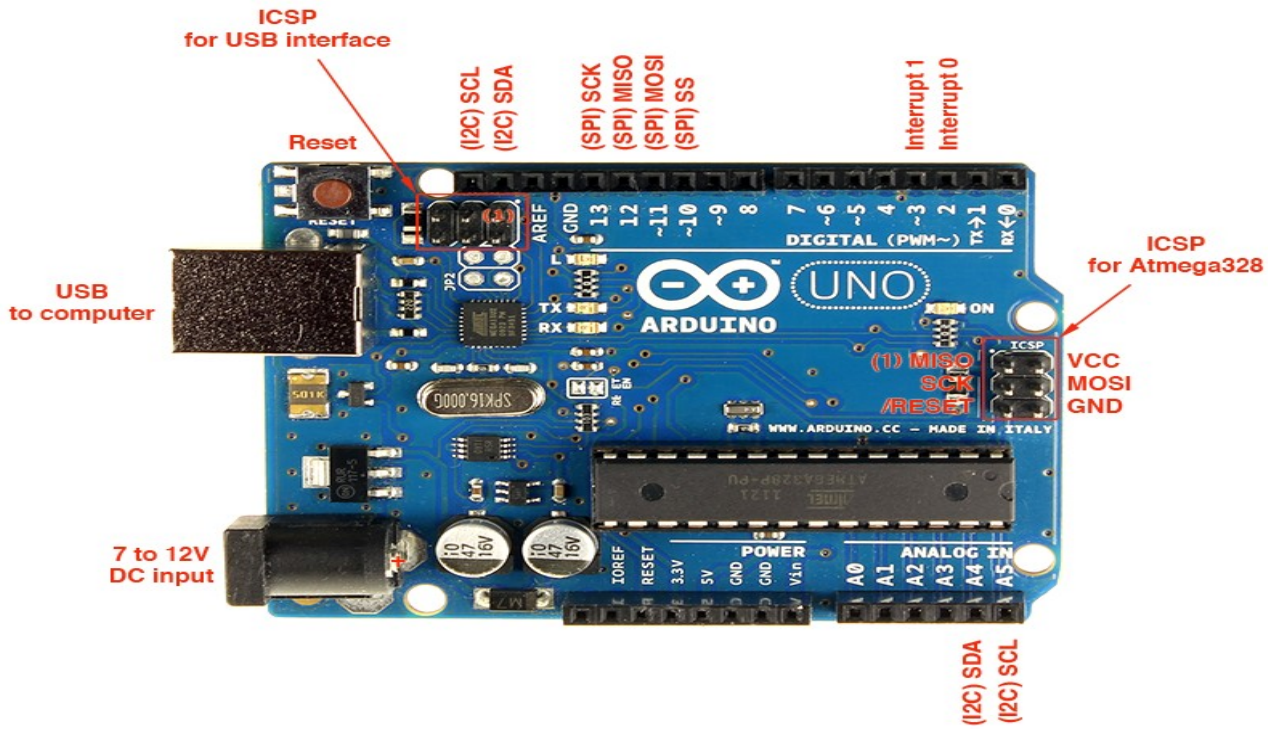
4.4. ARDUINO –

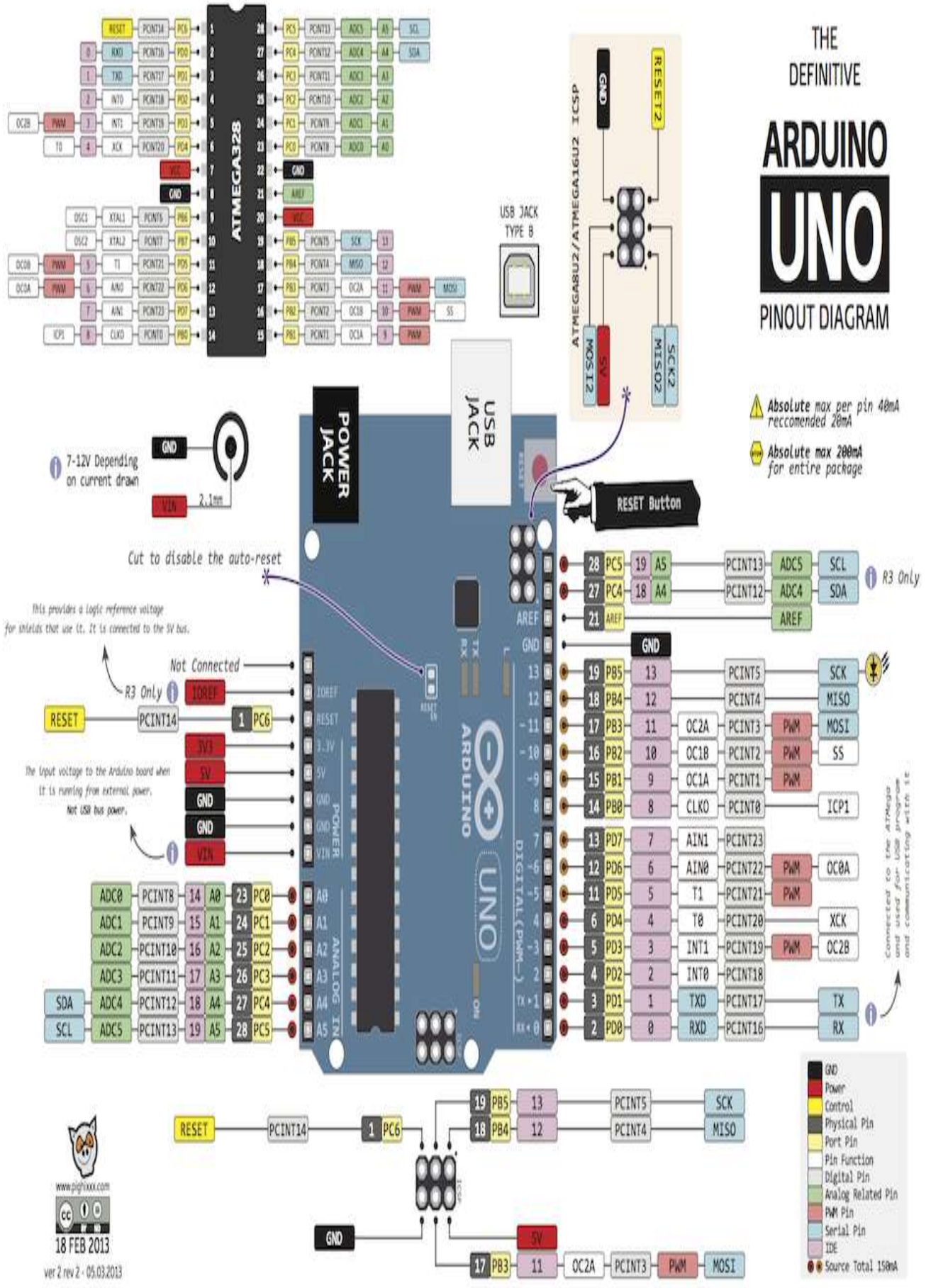
Arduino is an open source, computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2003 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy,[2] aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

The name *Arduino* comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

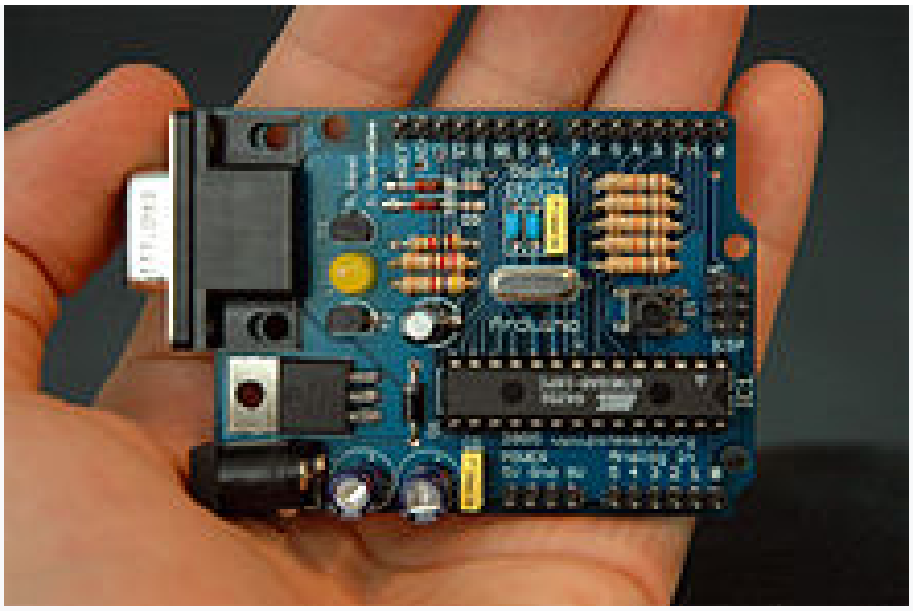



THE
DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
ATMEGA8U2/ATMEGA16U2 ICSP
USB JACK
TYPE B
POWER
JACK
USB
JACK
RESET Button
7-12V Depending
on current drawn
2.1mm
Cut to disable the auto-reset
This provides a logic reference voltage
for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when
it is running from external power.
Not USB bus power.
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
R3 Only
Connected to the ATMega
and used for USB program
and communicating with it
ARDUINO
UNO
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

HARDWARE-

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the GNU General Public License, version 2. Nevertheless, an official Bill of Materials of Arduino boards has never been released by Arduino staff.

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested that the name *Arduino* be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the project name by using various names ending in *–duino*.



An early Arduino board with an RS-232 serial interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

SOFTWARE DEVELOPEMENT-

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a *sketch*. Sketches are saved on the development computer as text files with the file extension *.ino*. Arduino Software (IDE) pre-1.0 saved sketches with the extension *.pde*.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

6.1. SOFTWARE-

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consist of only two functions:

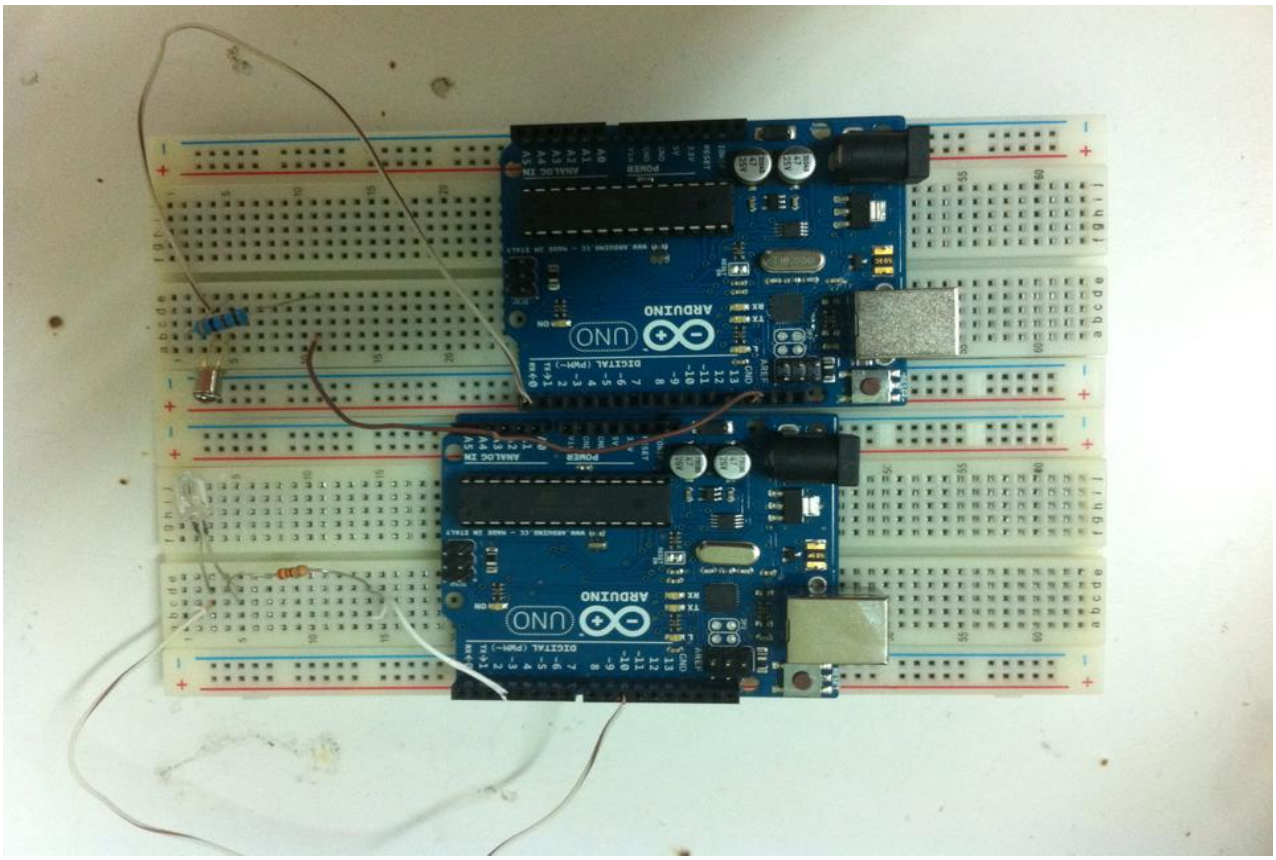
- *setup*: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
- *loop*: After *setup* has been called, function *loop* is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

Most Arduino boards contain a light-emitting diode (LED) and a load resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program for a beginning Arduino programmer blinks an LED repeatedly.

```
#define LED_PIN 13              // Pin number attached to LED.

void setup() {
        pinMode(LED_PIN, OUTPUT);       // Configure pin 13 to be a digital output.
}

void loop() {
        digitalWrite(LED_PIN, HIGH);    // Turn on the LED.
        delay(1000);                    // Wait 1 second (1000 milliseconds).
        digitalWrite(LED_PIN, LOW);     // Turn off the LED.
        delay(1000);                    // Wait 1 second.
}
```

This program uses the functions *pinMode*, *digitalWrite*, and *delay*, which are provided by the internal libraries included in the IDE environment. The program is usually loaded in the Arduino by the manufacturer.

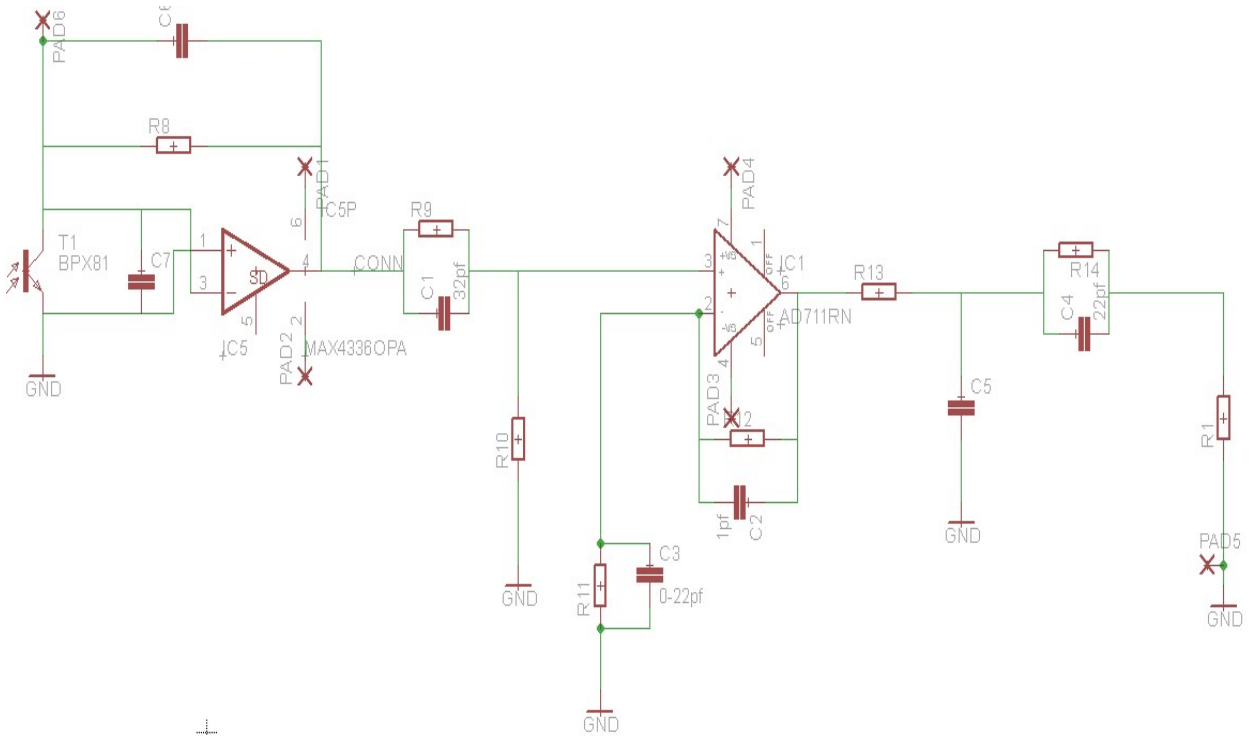
DESIGN OF VLC PROTOTYPE



The VLC circuit

VLC is typically implemented using white LED light bulbs at the transmitter. These devices are normally used for illumination only by applying a constant current. However, by fast and subtle variations of the current, the optical output can be made to vary at extremely high speeds. This very property of optical current is used in VLC setup. The operational procedure is very simple, if the LED is on, you transmit a digital 1, if it's off you transmit a 0. The LEDs can be switched on and off very quickly, which gives nice opportunities for transmitting data. Hence all that is required is some LEDs and an Arduino that code data into those LEDs. All one has to do is to vary the rate at which the LED's flicker depending upon the data we want to encode. Further enhancements can be made in this method, like using an array of LEDs for parallel data transmission to transmit larger data like, videos, audios and pictures.

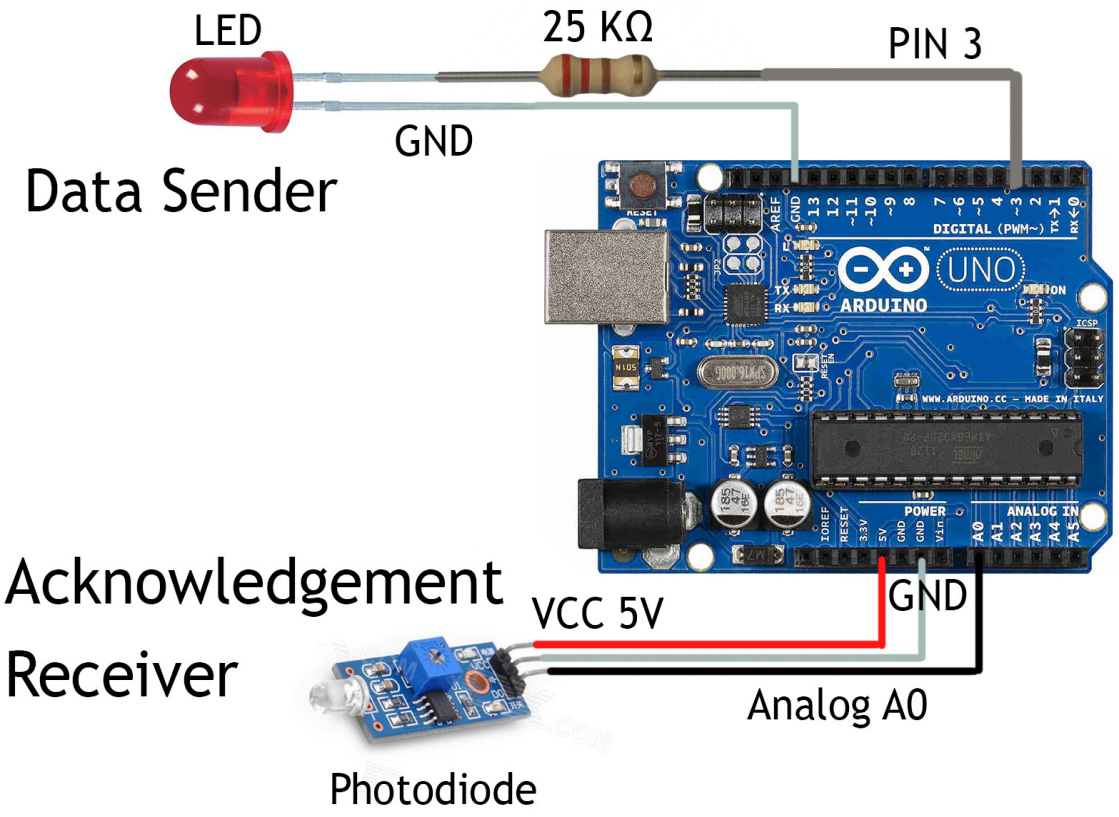
After building the previous circuit that shown in Fig.4.1 and sending a message from 5cm distance between the transmitter and receiver circuits we worked to improve our VLC system in order to increase the distance and to send more types of data in addition to the message such as images and voice.



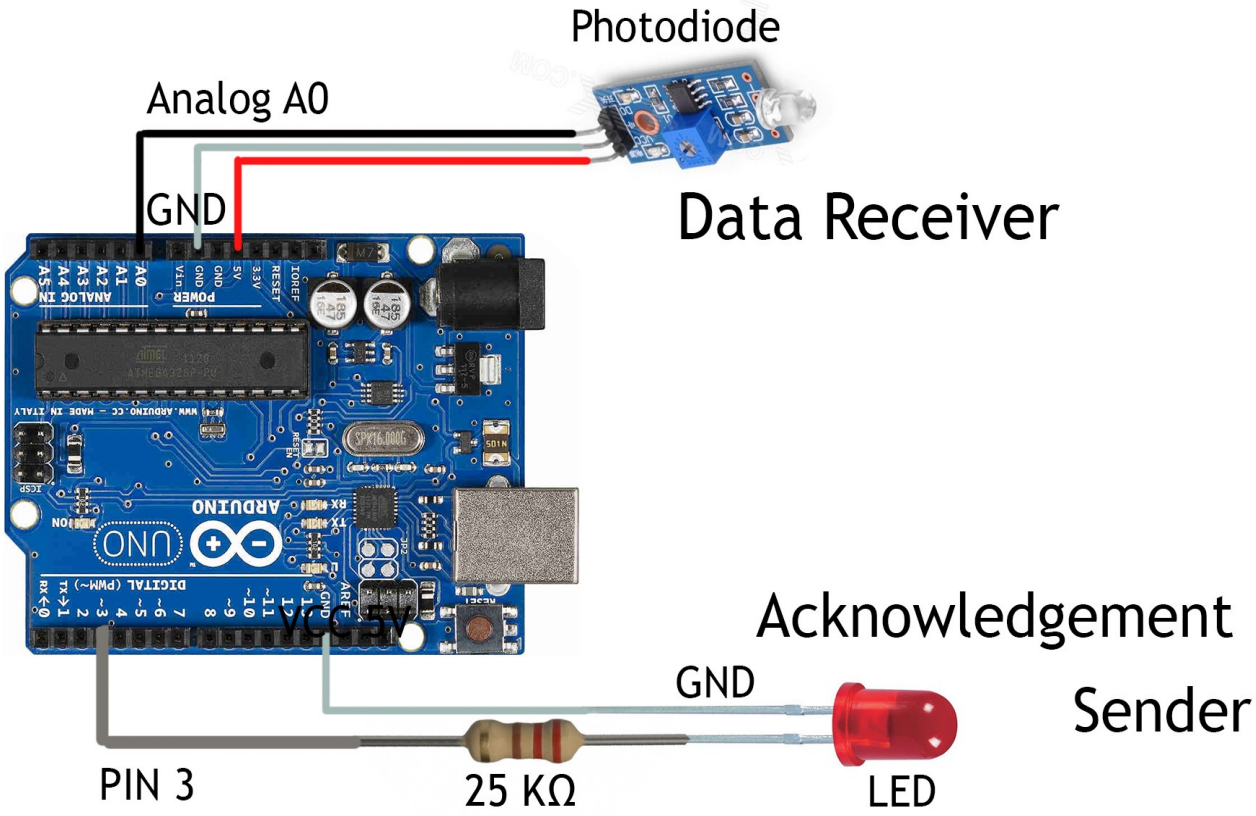
Experimental circuit

Circuit Diagram

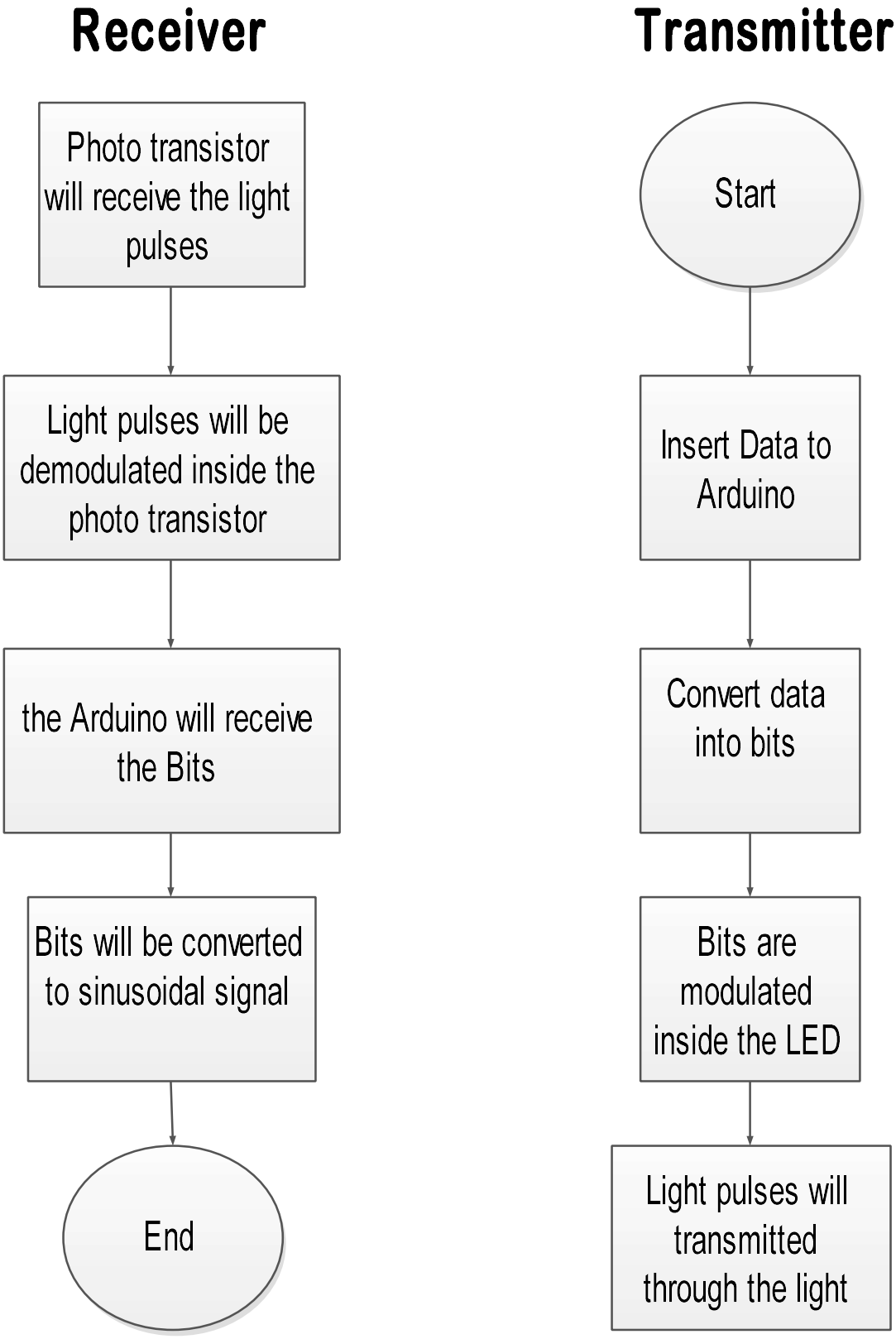
8.1. Sender Side



8.2. Receiver Side



FLOW CHART-

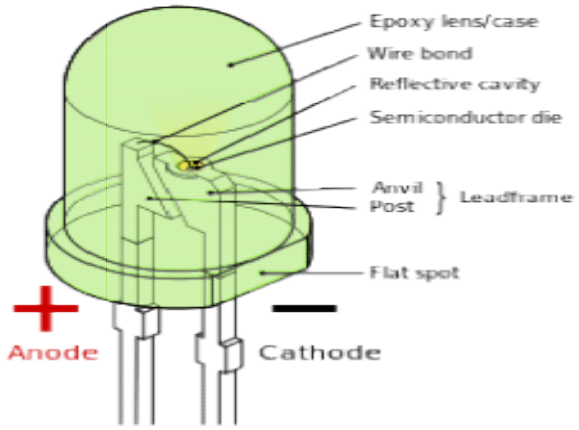

Receiver
Photo transistor will receive the light pulses
Light pulses will be demodulated inside the photo transistor
the Arduino will receive the Bits
Bits will be converted to sinusoidal signal
End
Transmitter
Start
Insert Data to Arduino
Convert data into bits
Bits are modulated inside the LED
Light pulses will transmitted through the light

Arduino

We used it to convert the input data into bits in order to transmit it into the LED.

LED:

It modulates the Bits received from the Arduino by converts the electrical current into light pulses



Why we used LEDs ?

With LEDs, it is possible to control light brightness at a frequency much higher than conventional light bulbs: LEDs can be switched on and off at very high rates. As result, LED-based lighting can be used for wireless communication services by modulating the intensity of the emitted light. Further, LEDs can also be used as receivers just like photodiodes. We call this concept Visible Light Communication (VLC) with LED-to-LED networking **.**

Phototransistor

In order for data transmission to have any significance there must be a way to receive the signal at the other end of the design. This is the purpose of the photodiodes as they react to the light emitted from the LEDs and allow for current to flow to the rest of the receiver circuit. When there is no light emitted from the LEDs the photodiodes do not allow current to flow through to the Arduino on the receiver.

ARDUINO CODE –

10.1. SENDER SIDE –

```
String s;
static int TRESHOLD = 300;
int q=0;
static unsigned int standardDelay = 40000; //1350 is limit
void setup() {
  Serial.begin(9600); // 9600 bits per second
  pinMode(3,OUTPUT); //digital PWM 3 on output
}
void loop() {
  //read every 100ms
  //high A0 value is one, low A0 value is zero
    if(q==0 && analogRead(A0)<TRESHOLD)
    {
      if(Serial.available() > 0) // Don't read unless
        s = Serial.readString();
    }
    else{

          for(int i=0 ;i< s.length(); i++)
          {
            WriteChar(s[i]);
            if(analogRead(A0)>TRESHOLD)
            i--;
          }
          q=1;
    }
  //}
}
void WriteChar(char str){
  switch (str - '0'){
    case 49:
      //code for a is 011 0001
      LightFlash(false, true, true, false, false, false, true);
      break;
    case 50:
      //code for b is 011 0010
      LightFlash(false, true, true, false, false, true, false);
      break;
    case 51:
      //code for c is 011 0011
      LightFlash(false, true, true, false, false, true, true);
```

```
  break;
case 52:
  //code for d is 011 0100
  LightFlash(false, true, true, false, true, false, false);
  break;
case 53:
  //code for e is 011 0101
  LightFlash(false, true, true, false, true, false, true);
  break;
case 54:
  //code for f is 011 0110
  LightFlash(false, true, true, false, true, true, false);
  break;
case 55:
  //code for g is 011 0111
  LightFlash(false, true, true, false, true, true, true);
  break;
case 56:
  //code for h is 011 1000
  LightFlash(false, true, true, true, false, false, false);
  break;
case 57:
  //code for i is 011 1001
  LightFlash(false, true, true, true, false, false, true);
  break;
case 58:
  //code for j is 011 1010
  LightFlash(false, true, true, true, false, true, false);
  break;
case 59:
  //code for k is 011 1011
  LightFlash(false, true, true, true, false, true, true);
  break;
case 60:
  //code for l is 011 1100
  LightFlash(false, true, true, true, true, false, false);
  break;
case 61:
  //code for m is 011 1101
  LightFlash(false, true, true, true, true, false, true);
  break;
case 62:
  //code for n is 011 1110
  LightFlash(false, true, true, true, true, true, false);
  break;
case 63:
  //code for o is 011 1111
  LightFlash(false, true, true, true, true, true, true);
```

```
  break;
case 64:
  //code for p is 100 0000
  LightFlash(true, false, false, false, false, false, false);
  break;
case 65:
  //code for q is 100 0001
  LightFlash(true, false, false, false, false, false, true);
  break;
case 66:
  //code for r is 100 0010
  LightFlash(true, false, false, false, false, true, false);
  break;
case 67:
  //code for s is 100 0011
  LightFlash(true, false, false, false, false, true, true);
  break;
case 68:
  //code for t is 100 0100
  LightFlash(true, false, false, false, true, false, false);
  break;
case 69:
  //code for u is 100 0101
  LightFlash(true, false, false, false, true, false, true);
  break;
case 70:
  //code for v is 100 0110
  LightFlash(true, false, false, false, true, true, false);
  break;
case 71:
  //code for w is 100 0111
  LightFlash(true, false, false, false, true, true, true);
  break;
case 72:
  //code for x is 100 1000
  LightFlash(true, false, false, true, false, false, false);
  break;
case 73:
  //code for y is 100 1001
  LightFlash(true, false, false, true, false, false, true);
  break;
case 74:
  //code for z is 100 1010
  LightFlash(true, false, false, true, false, true, false);
  break;
default:
  Serial.println(str - '0');
  Serial.println("CAME IN DEFAULT IN WRITECHAR");
```

```
    break;
  }
}

void LightFlash(boolean a, boolean b, boolean c, boolean d, boolean e, boolean f, boolean g)
{
  digitalWrite(3,LOW);
  if (a == true){
    digitalWrite(3,HIGH);
  }
  delayMicroseconds(standardDelay);
  //boolean sensorValue1 = (analogRead(A0)>TRESHOLD);
 // delayMicroseconds(standardDelay);

  digitalWrite(3,LOW);
  if (b == true){
    digitalWrite(3,HIGH);
  }
  //delayMicroseconds(standardDelay);
  //boolean sensorValue2 = (analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);

  digitalWrite(3,LOW);
  if (c == true){
    digitalWrite(3,HIGH);
  }
  delayMicroseconds(standardDelay);
  //boolean sensorValue3 = (analogRead(A0)>TRESHOLD);
  //delayMicroseconds(standardDelay);

  digitalWrite(3,LOW);
  if (d == true){
    digitalWrite(3,HIGH);
  }
  //delayMicroseconds(standardDelay);
  //boolean sensorValue4 = (analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);

  digitalWrite(3,LOW);
  if (e == true){
    digitalWrite(3,HIGH);
  }
  delayMicroseconds(standardDelay);
 // boolean sensorValue5 = (analogRead(A0)>TRESHOLD);
  //delayMicroseconds(standardDelay);

  digitalWrite(3,LOW);
  if (f == true){
```

```
    digitalWrite(3,HIGH);
   }
   delayMicroseconds(standardDelay);
  // boolean sensorValue6 = (analogRead(A0)>TRESHOLD);
   //delayMicroseconds(standardDelay);

   digitalWrite(3,LOW);
   if (g == true){
    digitalWrite(3,HIGH);
   }
   delayMicroseconds(standardDelay);
//  boolean sensorValue7 = (analogRead(A0)>TRESHOLD);
   //delayMicroseconds(standardDelay);
   delayMicroseconds(3*standardDelay); //1 second passed
}
```

10.2. RECEIVER SIDE –

```
static int TRESHOLD = 500;
static unsigned int standardDelay = 13500; //1350 is limit
int q=0;;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // 9600 bits per second
  int sensorValue = 0;
  pinMode(3,OUTPUT);
}
void loop() {
  if(q==0)
  {
    digitalWrite(3,HIGH);
    delay(10000);
    q=1;
    digitalWrite(3,LOW);
  }
  else{
  // put your main code here, to run repeatedly:
  delayMicroseconds(standardDelay/2);
  boolean a=(analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);
  //delayMicroseconds(standardDelay/2);

  boolean b=(analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);
  //delayMicroseconds(standardDelay/2);

  boolean c=(analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);
 // delayMicroseconds(standardDelay/2);

  boolean d=(analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);
  //delayMicroseconds(standardDelay/2);

  boolean e=(analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);
  //delayMicroseconds(standardDelay/2);

  boolean f=(analogRead(A0)>TRESHOLD);
  delayMicroseconds(standardDelay);
  //delayMicroseconds(standardDelay/2);

  boolean g=(analogRead(A0)>TRESHOLD);
```

```
  delayMicroseconds(standardDelay);
  //delayMicroseconds(standardDelay/2);

  delayMicroseconds(3*standardDelay); //1 second passed
  long result = 0;
  result = 1000000*a + 100000*b + 10000*c  + 1000*d + 100*e + 10*f + 1*g;
  //Serial.println(result);

  long int
a1[]={110001,110010,110011,110100,110101,110110,110111,111000,111001,111010,1110
11,111100,111101,111110,111111,1000000,1000001,1000010,1000011,1000100,1000101,1
000110,1000111,1001000,1001001,1001010};
  for(int i=0;i<26;i++)
  {
    digitalWrite(3,LOW);
    if(a1[i]==result)
    {
      PrintChar(result);
      break;
    }
    else
    {
      digitalWrite(3,HIGH);
    }
  }}
}
void PrintChar(long binary){
  switch (binary){
    case 110001:
      //code for a is 011 0001
      Serial.print("a");
      break;
    case 110010:
      //code for b is 011 0010
      Serial.print("b");
      break;
    case 110011:
      //code for c is 011 0011
      Serial.print("c");
      break;
    case 110100:
      //code for d is 011 0100
      Serial.print("d");
      break;
    case 110101:
      //code for e is 011 0101
      Serial.print("e");
      break;
```

```
case 110110:
  //code for f is 011 0110
  Serial.print("f");
  break;
case 110111:
  //code for g is 011 0111
  Serial.print("g");
  break;
case 111000:
  //code for h is 011 1000
  Serial.print("h");
  break;
case 111001:
  //code for i is 011 1001
  Serial.print("i");
  break;
case 111010:
  //code for j is 011 1010
  Serial.print("j");
  break;
case 111011:
  //code for k is 011 1011
  Serial.print("k");
  break;
case 111100:
  //code for l is 011 1100
  Serial.print("l");
  break;
case 111101:
  //code for m is 011 1101
  Serial.print("m");
  break;
case 111110:
  //code for n is 011 1110
  Serial.print("n");
  break;
case 111111:
  //code for o is 011 1111
  Serial.print("o");
  break;
case 1000000:
  //code for p is 100 0000
  Serial.print("p");
  break;
case 1000001:
  //code for q is 100 0001
  Serial.print("q");
  break;
```

```
    case 1000010:
      //code for r is 100 0010
      Serial.print("r");
      break;
    case 1000011:
      //code for s is 100 0011
      Serial.print("s");
      break;
    case 1000100:
      //code for t is 100 0100
      Serial.print("t");
      break;
    case 1000101:
      //code for u is 100 0101
      Serial.print("u");
      break;
    case 1000110:
      //code for v is 100 0110
      Serial.print("v");
      break;
    case 1000111:
      //code for w is 100 0111
      Serial.print("w");
      break;
    case 1001000:
      //code for x is 100 1000
      Serial.print("x");
      break;
    case 1001001:
      //code for y is 100 1001
      Serial.print("y");
      break;
    case 1001010:
      //code for z is 100 1010
      Serial.print("z");
      break;
    default:
      Serial.println("");
      break;
  }
}
```

C# CODE –

11.1. SENDER SIDE -

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
namespace WindowsFormsApplication3
{
   public partial class Form1 : Form
   {
      public string strFileName = "";
      public static System.IO.Ports.SerialPort serialPort1;
        public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {

      }
      private void btnConnect_Click_1(object sender, EventArgs e)
      {

         System.ComponentModel.IContainer components = new
System.ComponentModel.Container();
         serialPort1 = new System.IO.Ports.SerialPort(components); // Creating the new
object.
         serialPort1.PortName = "COM" + numCom.Value.ToString(); // Setting what port
number.
         serialPort1.BaudRate = 9600; // Setting baudrate.
         serialPort1.DtrEnable = true; // Enable the Data Terminal Ready
         serialPort1.Open(); // Open the port for use.
         btnConnect.Text = "Connected.";
         btnConnect.Enabled = false;
         numCom.Enabled = false;
      }
```

```
        private void btnSelect_Click(object sender, EventArgs e)
        {
            lblSucceed.Visible = false;
            OpenFileDialog fd = new OpenFileDialog();
            // string strFileName = null;

            fd.Title = "Open File Dialog";
            fd.InitialDirectory = "C:\\";
            fd.Filter = "All files (*.*)|*.*|All files (*.*)|*.*";
            fd.FilterIndex = 2;
            fd.RestoreDirectory = true;

            if (fd.ShowDialog() == DialogResult.OK)
            {
                strFileName = fd.FileName;
                lblFileName.Text = lblFileName.Text + strFileName;
            }
        }
        private void btnSend_Click(object sender, EventArgs e)
        {
            try
            {

                // Sends the text as a byte.
                //  serialPort1.Write(new byte[] { Convert.ToByte(txtDatasend.Text) }, 0, 1);
                string text = System.IO.File.ReadAllText(strFileName);
                serialPort1.Write(text);
            }
            catch (Exception exc)
            {
                Console.WriteLine("Exception caught in process: {0}", exc.ToString());
            }
        }
    }
}
```

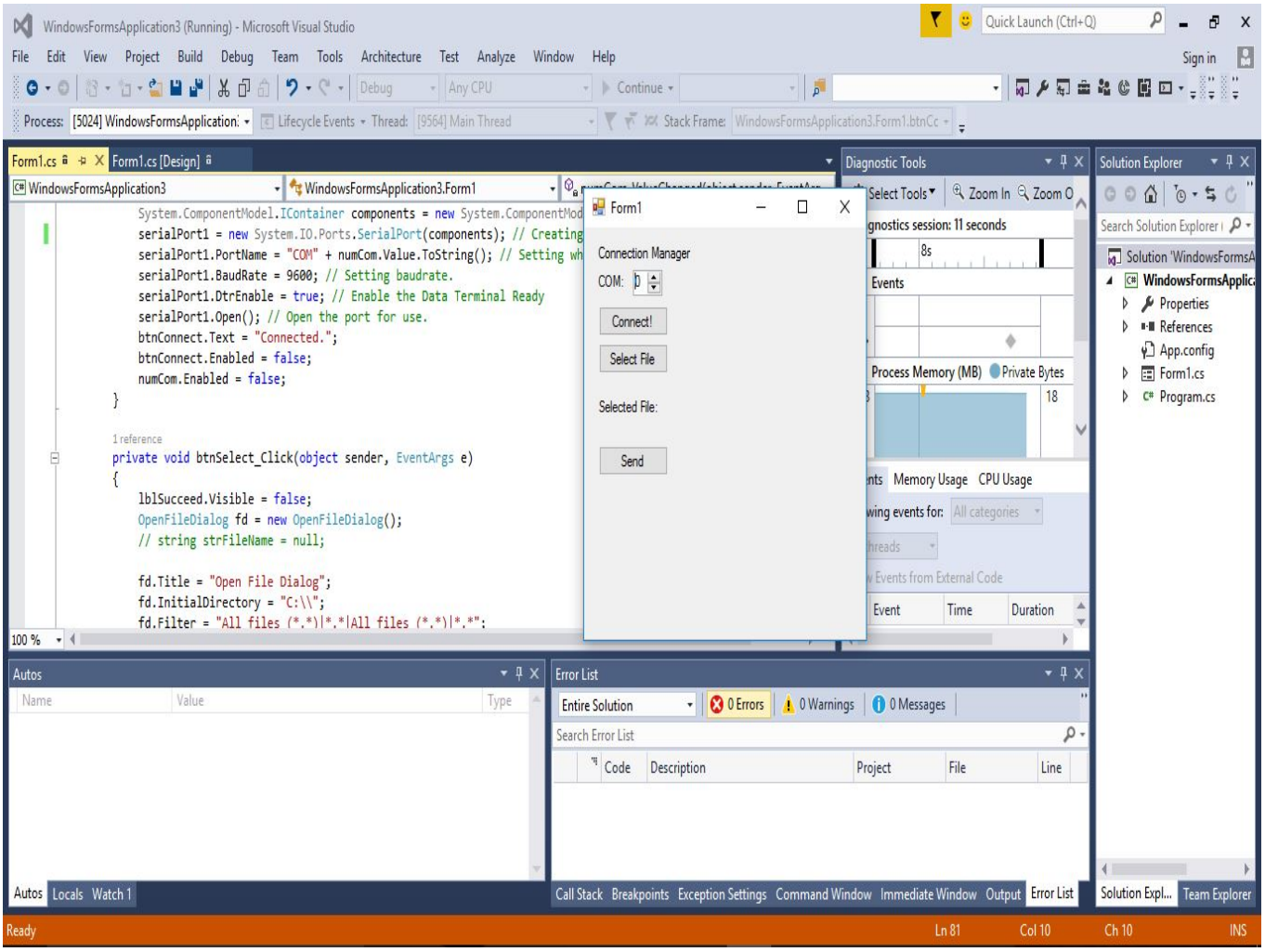
11.2. RECEIVER SIDE –

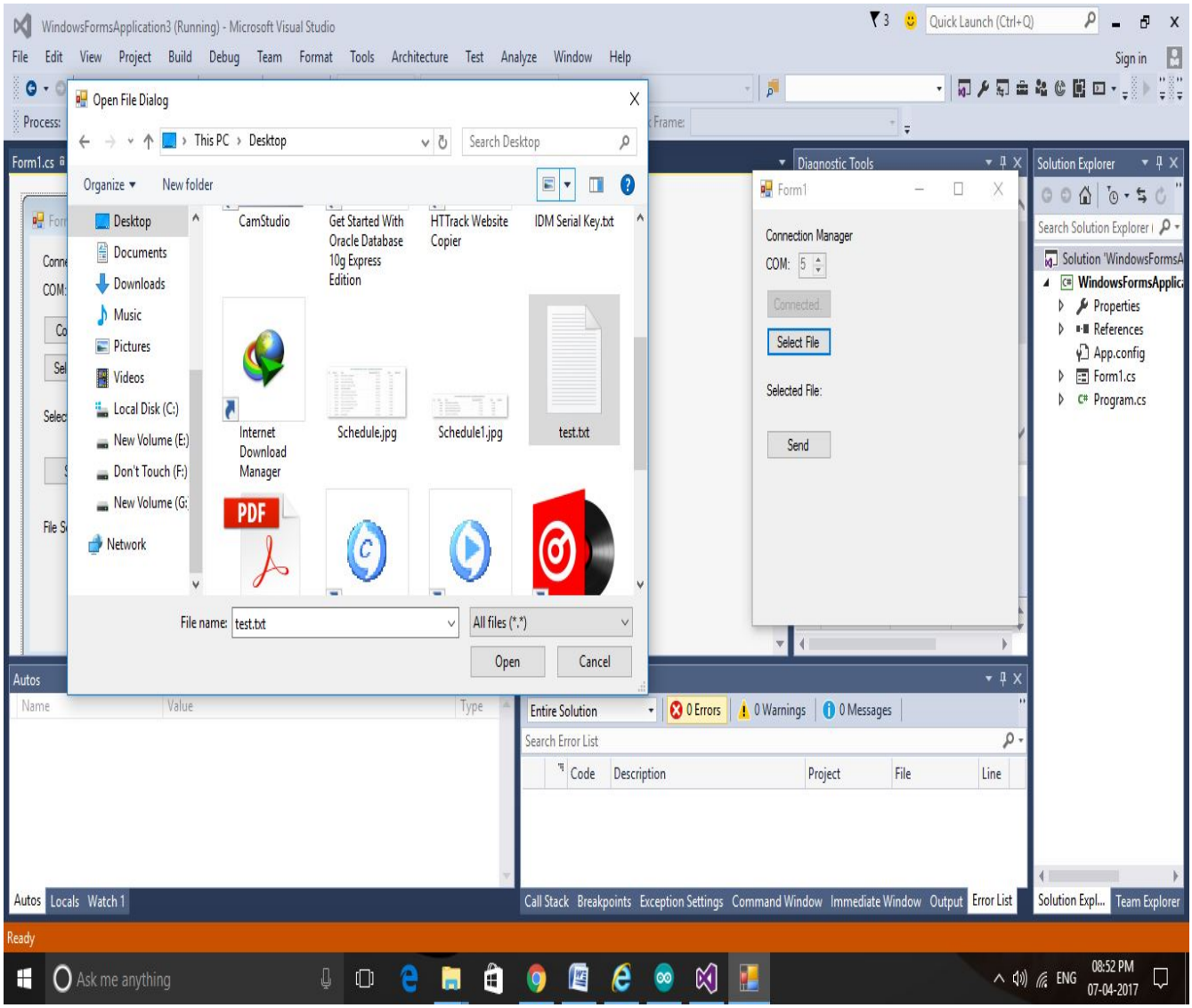
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
namespace WindowsFormsApplication5

{
   public partial class Form1 : Form

{
      private SerialPort serialPort1;
      string str = null;
      public Form1()

        {
           InitializeComponent();
        }

private void btnConnect_Click(object sender, EventArgs e)

        {
                 System.ComponentModel.IContainer components = new
                System.ComponentModel.Container();
                 serialPort1 = new System.IO.Ports.SerialPort(components); // Creating the new
                object.
                serialPort1.PortName = "COM" + numCom.Value.ToString(); // Setting what
                port number.
                serialPort1.BaudRate = 9600; // Setting baudrate.
                serialPort1.DtrEnable = true; // Enable the Data Terminal Ready
                serialPort1.Open(); // Open the port for use.
                btnConnect.Text = "Connected.";
                btnConnect.Enabled = false;
                 numCom.Enabled = false;
      }
```

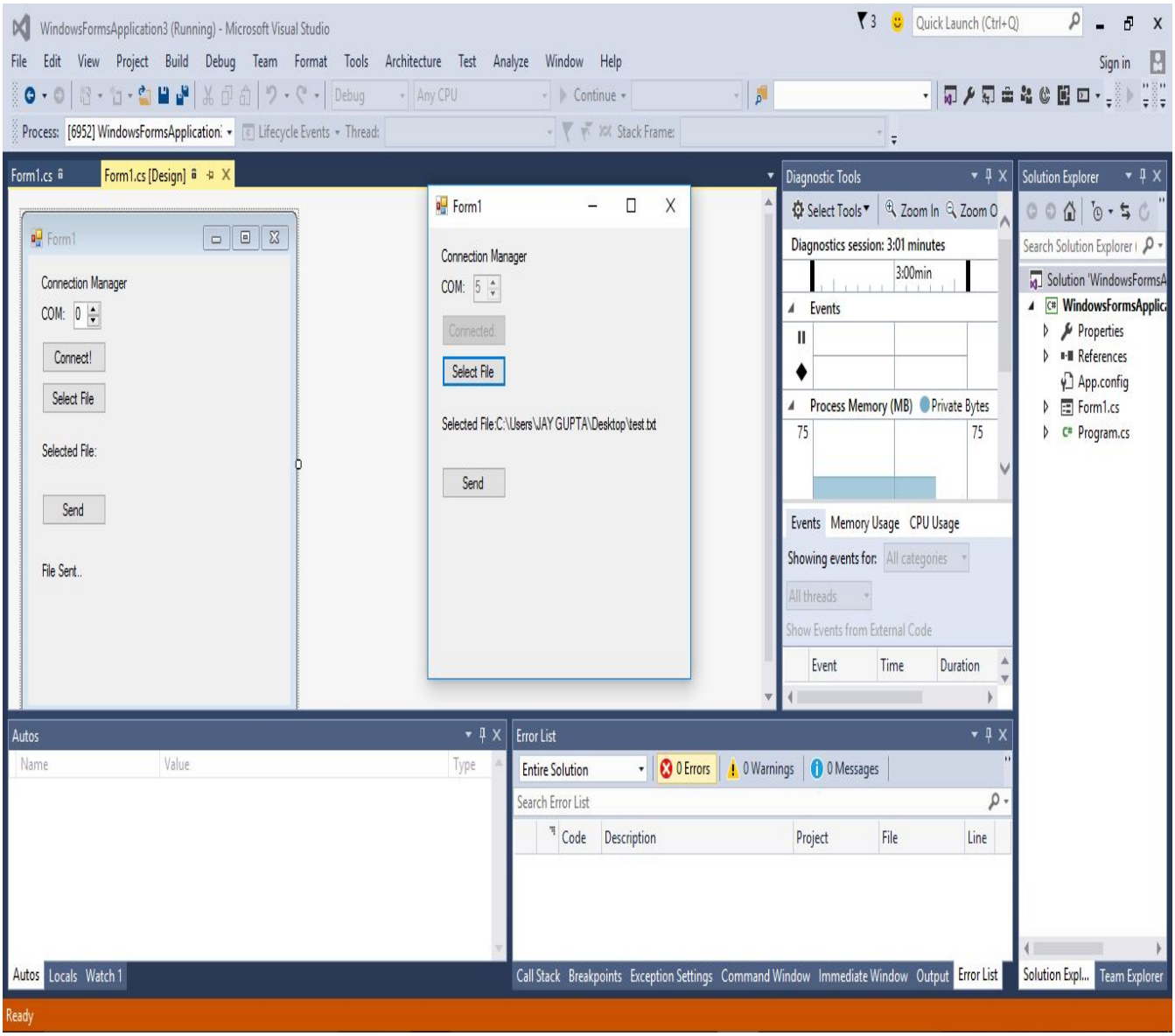
```
private void button1Show_Click(object sender, EventArgs e)
        {
            while (true)
            {
                str += serialPort1.ReadLine();
                textBox1.Text = str;
            }
        }

        private void button2Close_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
        }
    }
}
```

SCREENSHOTS-

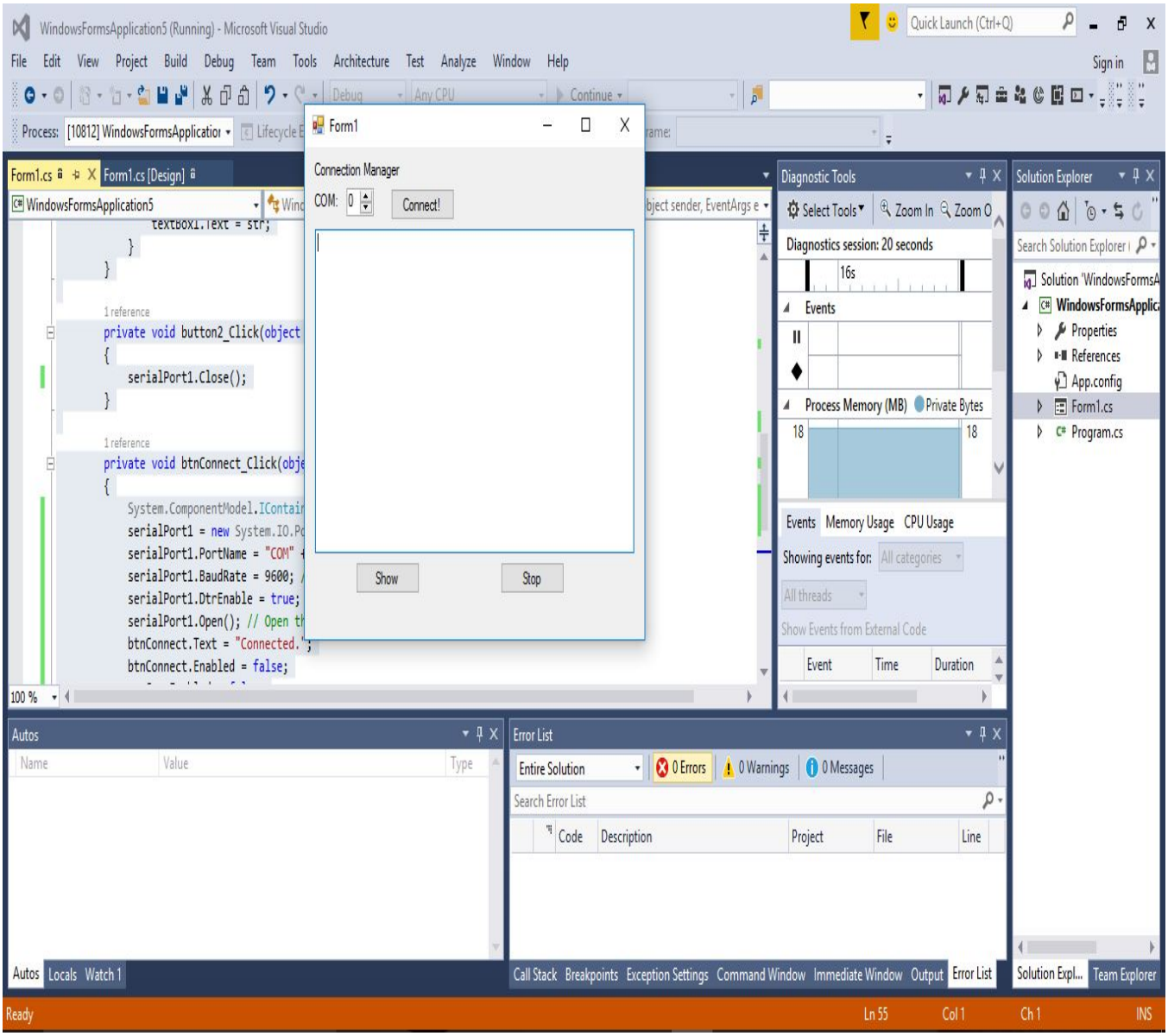
12.1. SENDER SIDE FORM App-

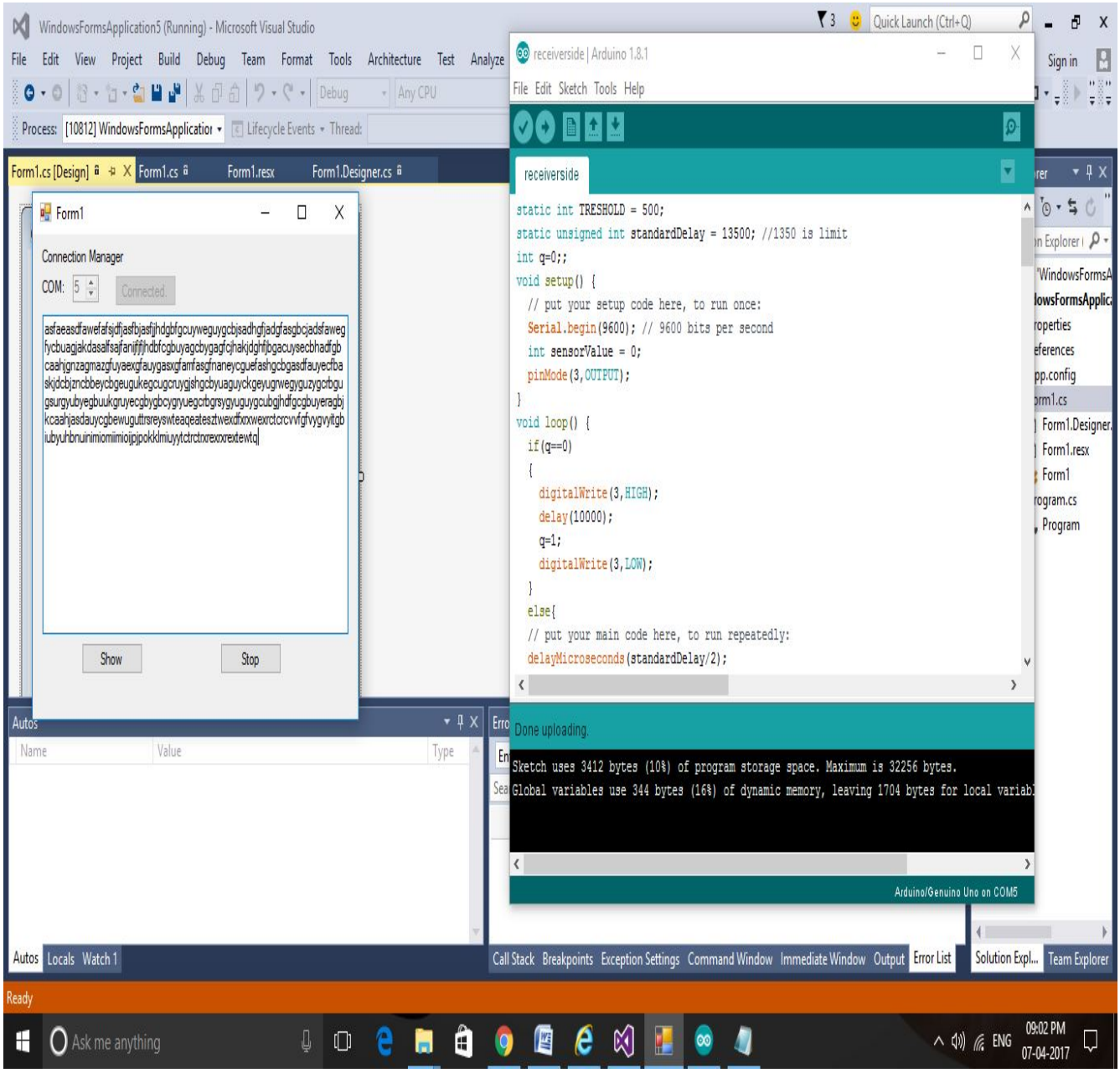



WindowsFormsApplication3 (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Architecture Test Analyze Window Help
Sign in
Quick Launch (Ctrl+Q)
Open File Dialog
This PC › Desktop
Search Desktop
Organize
New folder
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (E:)
Don't Touch (F:)
New Volume (G:)
Network
CamStudio
Get Started With Oracle Database 10g Express Edition
HTTrack Website Copier
IDM Serial Key.txt
Internet Download Manager
Schedule.jpg
Schedule1.jpg
test.txt
PDF
File name: test.txt
All files (*.*)
Open
Cancel
Form1
Connection Manager
COM: 5
Connected
Select File
Selected File:
Send
Diagnostic Tools
Solution Explorer
Search Solution Explorer
Solution 'WindowsFormsA
WindowsFormsApplica
Properties
References
App.config
Form1.cs
Program.cs
Autos
Name
Value
Type
Entire Solution
0 Errors
0 Warnings
0 Messages
Search Error List
Code
Description
Project
File
Line
Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List
Solution Expl... Team Explorer
Ready
Ask me anything
ENG
08:52 PM
07-04-2017


WindowsFormsApplication3 (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Architecture Test Analyze Window Help
Form1.cs
Form1.cs [Design]
Form1
Connection Manager
COM: 0
Connect!
Select File
Selected File:
Send
File Sent..
Form1
Connection Manager
COM: 5
Connected.
Select File
Selected File:C:\Users\JAY GUPTA\Desktop\test.txt
Send
Diagnostic Tools
Diagnostics session: 3:01 minutes
Events
Process Memory (MB)
Private Bytes
Events Memory Usage CPU Usage
Solution Explorer
Properties
References
App.config
Form1.cs
Program.cs
Autos
Name Value Type
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Code Description Project File Line
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List
Ready

12.2. RECEIVER SIDE FORM APPLICATION-




WindowsFormsApplication5 (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Architecture Test Analyze
Debug
Any CPU
Process: [10812] WindowsFormsApplicatior
Lifecycle Events
Thread:
Form1.cs [Design]
Form1.cs
Form1.resx
Form1.Designer.cs
Form1
Connection Manager
COM: 5
Connected.
asfaeasdfawefafsjdfjasfbjasfjjhdgbfgcuyweguygcbjsadhgfjadgfasgbcjadsfaweg
fycbuagjakdasalfsajfanjfjfjhdbfcgbuyagcbygagfcjhakjdghfjbgacuysecbhadfgb
caahjgnzagmazgfuyaexgfauygasxgfamfasgfnaneycguefashgcbgasdfauyecfba
skjdcbjzncbbeycbgeugukegcugcruygjshgcbyuaguyckgeyugrwegyguzygcrbgu
gsurgyubyegbuukgruyecgbygbcygryuegcrbgrsygyuguygcubgjhdfgcqbuyeragbj
kcaahjasdauycgbewuguttrsreyswteaqeateszwexdfxrxwexrctcrcvvfgfvygvyitgb
iubyuhbnuinimiomiimioijpjpokklmiuyytctrctnxrexnxrextewtq
Show
Stop
receiverside | Arduino 1.8.1
File Edit Sketch Tools Help
receiverside
static int TRESHOLD = 500;
static unsigned int standardDelay = 13500; //1350 is limit
int q=0;;
void setup() {
// put your setup code here, to run once:
Serial.begin(9600); // 9600 bits per second
int sensorValue = 0;
pinMode(3,OUTPUT);
}
void loop() {
if(q==0)
{
digitalWrite(3,HIGH);
delay(10000);
q=1;
digitalWrite(3,LOW);
}
else{
// put your main code here, to run repeatedly:
delayMicroseconds(standardDelay/2);
Done uploading.
Sketch uses 3412 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 344 bytes (16%) of dynamic memory, leaving 1704 bytes for local variabl
Arduino/Genuino Uno on COM5
Autos
Name
Value
Type
Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List
Solution Expl... Team Explorer
Ready
Ask me anything
ENG
09:02 PM
07-04-2017

CONCLUSION

At the end of this project we were able to send and receive text message through led-to-led communication. This proved that at the future we will be able to send and receive any kind of data using every light bulb everywhere like the Wi-Fi hotspots.

In the future we hope to send and receive all kind of data such as video and audio. In addition, we hope to have mobile-to-mobile communication instead of computer-to-computer communication.

This technology has a bright scope in future. This technology demonstrated a solution to the problem of integrating Visible Light Communication technology with present infrastructure, without having to make major changes to that infrastructure. Visible Light Communication is a rapidly growing segment of the field of communication. There are many advantages to using VLC. There are also many challenges. VLC will be able to solve many of the problems people have been facing for many years, mainly environmental and power usage issues.

VLC is still in its beginning stages, but improvements are being made rapidly, and soon this technology will be able to be used in our daily lives. In spite of the research problems it is our belief that the VLC system will become one of the most promising technologies for the future generation in optical wireless communication.

LIMITATION

There are numerous issues and limitations in the design and implementation of the VLCS ranging from analog to digital transmission as stated below.

A. Ambient Lighting Source

The presence of a stronger light other than our system light generate low signal-to-noise ratio (SNR), which causes distortion in data. This results in no or inaccurate information transfer.

B. Line-of-sight Propagation

The sender receiver pair must be in the line-of-sight with each other; otherwise there will be no communication, and even if the sender receiver angle is altered very slightly, the communication range and accuracy will be reduces considerably.

C. Digital Issues

It is difficult to find the precise sampling rate of the ADC. The output from the ADC is not constant. The CCS and IDE have issues in compiling program, where errors are not properly shown, and many times restarting the IDE is required.

D. Analog Issues

1) LED Brightness: Original LEDs are very dim and can perform very short range communication only. A power MOSFET is required for transmitting strong signal to the
LED during transmission.

2) MOSFET Limitations: MOSFET devices produce more heat and that could make the system fail, therefore, MOSFET is not a good choice to design with.

Scope of Future Work

VLC is an emerging technology and hence it has vast potential. A lot of research can be conducted in this field. Already, a lot of scientists are involved in extensive research in this field. This technology, pioneered by Harald Haas, can become one of the major technologies in the near future. If this technology can be used efficiently, we might soon have something of the kind of WI-FI hotspots wherever a light bulb is available. It will be cleaner and greener and the future of mankind will be safe. As the amount of available bandwidth is limited, the airwaves are becoming increasingly clogged, making it more and more difficult to get a reliable, high-speed signal. The VLC technology can solve this crisis. Moreover, it will allow inter access in places such as operation theaters and aircrafts where internet access is usually not allowed. The future of VLC is GI-FI. GI-FI or gigabit wireless refers to wireless communication at a data rate of more than one billion bits (gigabit) per second. In 2008 researchers at the University of Melbourne demonstrated a transceiver integrated on a single integrated circuit (chip) that operated at 60 GHz on the CMOS process. It will allow wireless transfer of audio and video data at up to 5 gigabits per second, ten times the current maximum wireless transfer rate, at one-tenth the cost. Researchers chose the 57–64 GHz unlicensed frequency band since the millimeter-wave range of the spectrum allowed high component on-chip integration as well as the integration of very small high gain arrays. The available 7 GHz of spectrum results in very high data rates, up to 5 gigabits per second to users within an indoor environment, usually within a range of 10 meters. Some press reports called this "Gi-Fi". It was developed by Melbourne University-based laboratories of NICTA (National ICT Australia Limited), Australia‘s Information and Communications Technology Research Centre of Excellence.

Logo of Gi-Fi

It’s estimated that the VLC market will be worth more than $6 billion by 2018, according to analysis by MarketsandMarkets. Haas is well aware of the need for many more players in the space for it to be viable, and he’s hoping Scotland will be central to this drive

REFERENCES

1. http://www.purelifi.com/

2. https://en.wikipedia.org/wiki/Li-Fi

3. https://www.youtube.com/watch?v=iHWIZsIBj3Q

4. https://www.youtube.com/watch?v=UulEFh8yhCg

5. https://embed.ted.com/d6e48260-3af2-4508-8ff1-a213e1709641

6. http://www.ijcta.com/documents/volumes/vol5issue1/ijcta2014050121.pdf

7. http://www.ripublication.com/irph/ijict_spl/ijictv4n16spl_11.pdf