**Light to Letter Converter**

**in**

Electronics and Communication Engineering

Submitted by

**Piyush Rohit(23ECE1038)**

**Ameer(23ECE1039)**

**Sahil Kesarkar(22ECE1027)**

To

**Dr. Anirban Chatterjee**

Associate Professor at NIT Goa



Department of Electronics and Communication Engineering

National Institute of Technology Goa

INTRODUCTION:-

The Rapidly of evolving communication technologies, alternative and assistive methods of information transfer have gained significant importance. One such innovation is the light-to-letter converter, a system designed to interpret light signals and convert them into corresponding alphabetic characters.

This technology is particularly useful in optical communication, assistive devices for the differently-abled, and hands-free or silent communication systems. By using a light source—such as an LED, laser, or flashlight—to transmit coded signals, and a light sensor to receive them, the system can decode the pattern and display or output the corresponding letter.

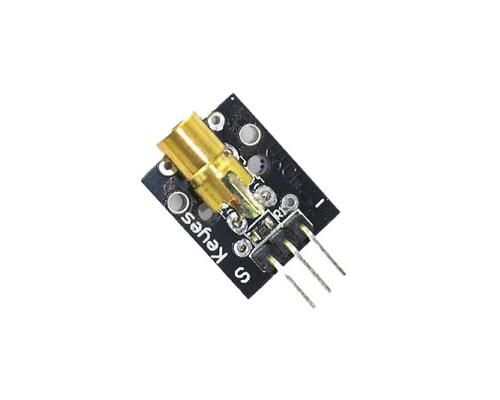
The concept is based on the principle of detecting variations in light intensity or blinking patterns, which are then processed by a microcontroller or software algorithm. These patterns may follow predefined formats such as Morse code, binary, or custom signal mapping.

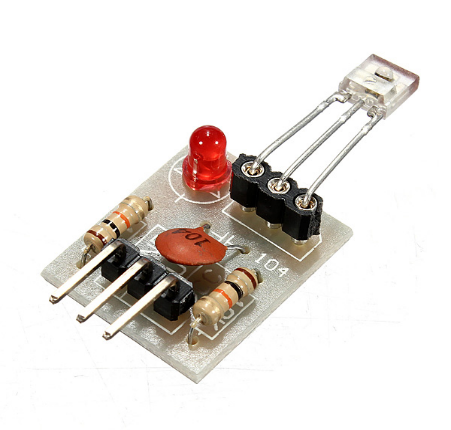
This project not only explores the fascinating interaction between light and digital decoding but also serves as a foundation for more advanced systems like optical keyboards, vision-based communication tools, and smart IoT devices.

**HARDWARE DESCRIPTION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.no** | **Components** | **Value** | **Quantity** |
| **1** | **Laser Transmitter Module** |  | **1** |
| **2** | **Laser Receiver Module** |  | **1** |
| **3** | **Arduino Microcontroller** |  | **2** |
| **4** | **Connecting Wires** |  |  |
| **5** | **Oled display** |  | **2** |
| **6** | **5 key keyboard module** |  | **1** |

**Component discription**

1. **Laser Transmitter Module:** The laser diode within the module is energized by an electrical current. This process leads to the emission of photons. The structure of the laser diode, including a resonant cavity with mirrors, causes these photons to stimulate the emission of more photons at the same wavelength and phase, resulting in a coherent and focused laser beam **2)Laser receiver Module:** A typical laser receiver module utilizes a photodetector, such as a photodiode or a phototransistor, which is sensitive to light at the wavelength of the incoming laser. When the laser beam strikes the photodetector, it generates a change in electrical current or voltage. This change is then processed by internal circuitry within the module to produce a usable output signal, often a digital signal (HIGH or LOW)



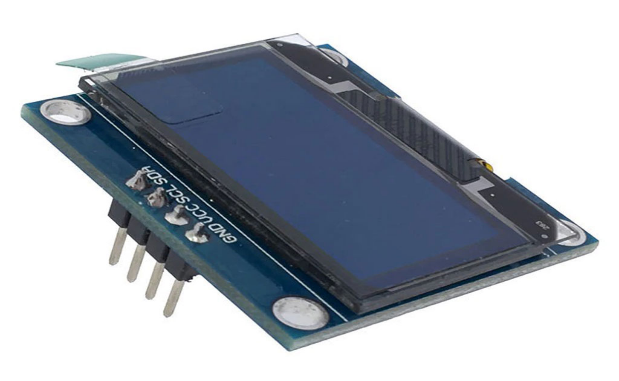
**3)Arduino Microcontroller:** This serves as the central processing unit for the decoder. It will read the signal from the laser receiver, process it to identify Morse code elements (dots and dashes), and eventually translate them into characters.

****

4) **Connecting Wires:** Used to establish electrical connections between the modules and the Arduino.

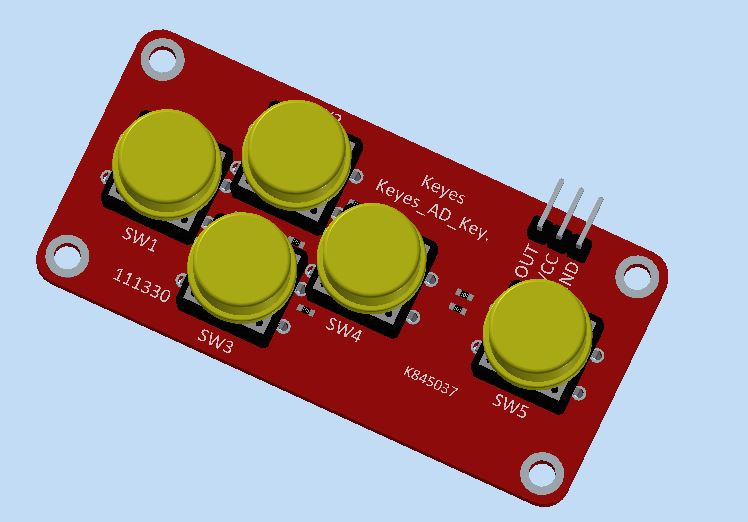
**5)Oled display:-**

An OLED display (Organic Light Emitting Diode display) is a type of flat-panel display technology made up of organic compounds that emit light when an electric current is applied. Unlike traditional LCDs (Liquid Crystal Displays), OLEDs do not require a backlight, as each pixel generates its own light. This allows for deeper blacks, higher contrast ratios, and thinner, more flexible screens.

****

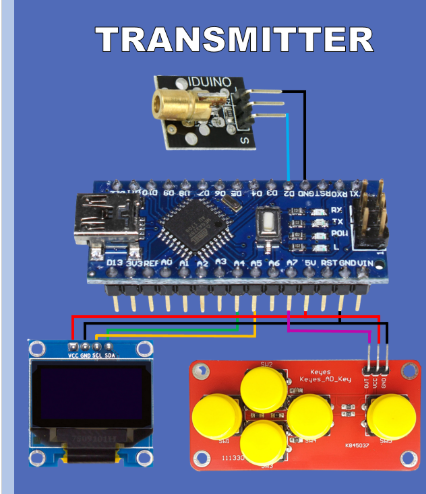
**6)5 key keyboard module:-**

A 5-key keyboard module is a small electronic component containing five individual push buttons arranged for directional and selection input, commonly used in microcontroller projects for user interaction.



**Circuit diagram:-**

**1)Transmitter circuit:-**

****

**Code:-**  **// Mario's Ideas**

**// Text input using OLED display an 5 Key Keyboard and encoding it to Morse Code**

**#include <SPI.h>**

**#include <Wire.h>**

**#include <Adafruit\_GFX.h>**

**#include <Adafruit\_SSD1306.h>**

**// Oled display size**

**#define SCREEN\_WIDTH 128 // OLED display width, in pixels**

**#define SCREEN\_HEIGHT 64 // OLED display height, in pixels**

**// Array holding all Morse code letter dot dash combinations**

**char MorseCode[26][6] = {**

**{'.','-','x','x','x','A'},**

**{'-','.','.','.','x','B'},**

**{'-','.','-','.','x','C'},**

**{'-','.','.','x','x','D'},**

**{'.','x','x','x','x','E'},**

**{'.','.','-','.','x','F'},**

**{'-','-','.','x','x','G'},**

**{'.','.','.','.','x','H'},**

**{'.','.','x','x','x','I'},**

**{'.','-','-','-','x','J'},**

**{'-','.','-','x','x','K'},**

**{'.','-','.','.','x','L'},**

**{'-','-','x','x','x','M'},**

**{'-','.','x','x','x','N'},**

**{'-','-','-','x','x','O'},**

**{'.','-','-','.','x','P'},**

**{'-','-','.','-','x','Q'},**

**{'.','-','.','x','x','R'},**

**{'.','.','.','x','x','S'},**

**{'-','x','x','x','x','T'},**

**{'.','.','-','x','x','U'},**

**{'.','-','-','x','x','W'},**

**{'.','.','.','-','x','V'},**

**{'.','-','-','.','x','X'},**

**{'-','.','-','-','x','Y'},**

**{'-','-','.','.','x','Z'},**

**//Potentiometer PIN A1**

**int Keyboard=A2;**

**// Variables capturing current and newly calculated position on the letter board (9x3 - 27 postions)**

**int New\_X=0;**

**int Old\_X=0;**

**int New\_Y=0;**

**int Old\_Y=0;**

**// Variable capturing output from Keyboard pin (Values 0 1023)**

**int Key\_read=0;**

**int Prev\_Key\_read=1023;**

**boolean Key\_pressed=false;**

**// String variable holding the text to transmit**

**String To\_Transmit="";**

**//Unit length**

**int unit=300;**

**// Length of the text to transmit**

**int To\_Transmit\_Length=0;**

**// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)**

**#define OLED\_RESET 4 // Reset pin # (or -1 if sharing Arduino reset pin)**

**Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);**

**// Used for displaying Leter board**

**char Letters[3][9]={"ABCDEFGHI",**

**"JKLMNOPQR",**

**"STUVWXYZ\_"};**

**// Play/Display Morse code representation of the letter**

**void Play\_Letter (char Letter){**

**// searching in MorseCode array for the corresponding letter**

**if (Letter=='\_') delay(5\*unit); else {**

**for (int j=0; j<26; j++){**

**if (MorseCode[j][5]==Letter)**

**// if the right letter is detected run Play\_Dot\_Dash for . or -**

**for (int k=0; k<4;k++){**

**if (MorseCode[j][k]!='x') Play\_Dot\_Dash(MorseCode[j][k]);**

**}**

**}**

**delay(2\*unit);**

**}**

**}**

**// Playing/Displaying . or -**

**void Play\_Dot\_Dash(char sign){**

**if (sign=='.'){**

**analogWrite(A3, 255);**

**delay(unit);**

**analogWrite(A3,0);**

**delay(unit);**

**}**

**if (sign=='-'){**

**analogWrite(A3, 255);**

**delay(3\*unit);**

**analogWrite(A3, 0);**

**delay(unit);**

**}**

**}**

**void setup() {**

**Serial.begin(9600);**

**// SSD1306\_SWITCHCAPVCC = generate display voltage from 3.3V internally**

**if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64**

**Serial.println(F("SSD1306 allocation failed"));**

**for(;;); // Don't proceed, loop forever**

**}**

**// Show initial display buffer contents on the screen --**

**// the library initializes this with an Adafruit splash screen.**

**display.display();**

**delay(2000); // Pause for 2 seconds**

**// Clear the buffer**

**display.clearDisplay();**

**display.display();**

**// Display filled in rect in the top section of the display when To\_Transfer would be output**

**display.fillRect(0, 0, 128, 15, SSD1306\_INVERSE);**

**display.drawRect(110, 2, 16, 12, SSD1306\_BLACK);**

**display.setTextSize(1);**

**display.setTextColor(SSD1306\_BLACK);**

**display.setCursor(113,4);**

**display.println("OK");**

**display.display();**

**// Display Letter Board 3 rows 9 character in each row**

**display.setTextSize(2);**

**display.setTextColor(SSD1306\_WHITE);**

**for (int j=0; j<3;j++){**

**for (int i=0; i<9;i++){**

**display.setCursor(i\*12+2\*i+1,j\*16+17);**

**display.println(Letters[j][i]);**

**delay(10);**

**display.display();**

**}**

**}**

**// Highlight character A by displaying Inverse rect at first position**

**display.fillRect(0, 16, 12, 16, SSD1306\_INVERSE);**

**display.display();**

**}**

**void Highlight\_letter(int X, int X\_Old, int Y, int Y\_Old){**

**// When position changes**

**// Draw the inverse rect in the Old\_pos to deactivate the highlight in the old spot**

**// Draw the inverse rect to Highlite the new spot**

**// Displaying Inverse rect in a new position to highlight**

**display.fillRect(X\*12+2\*X, Y\*16 +16, 12, 16, SSD1306\_INVERSE);**

**// Displaying Inverse rect in the old positon to unhighlight**

**display.fillRect(X\_Old\*12+2\*X\_Old, Y\_Old\*16 +16, 12, 16, SSD1306\_INVERSE);**

**display.display();**

**}**

**void loop() {**

**Key\_read =analogRead(Keyboard);**

**if (Prev\_Key\_read>1000 and Key\_read<1000){**

**Key\_pressed=true;**

**if (Key\_read<10 and Old\_X>0) New\_X=Old\_X-1;**

**if (Key\_read>500 and Key\_read<515 and Old\_X<9) New\_X=Old\_X+1;**

**if (Key\_read>140 and Key\_read<150 and Old\_Y>-1) New\_Y=Old\_Y-1;**

**if (Key\_read>325 and Key\_read<335 and Old\_Y<2 ) New\_Y=Old\_Y+1;**

**if (Key\_read>735 and Key\_read<745) {**

**if (New\_Y!=-1){**

**To\_Transmit=To\_Transmit + Letters[New\_Y][New\_X];**

**To\_Transmit\_Length++;**

**display.setTextSize(1);**

**display.setCursor(3,1);**

**display.setTextColor(BLACK );**

**display.fillRect(0, 0, 100, 15, SSD1306\_WHITE);**

**display.println(To\_Transmit);**

**display.display();**

**}**

**else{**

**for (int i=0; i<To\_Transmit.length();i++ ){**

**Play\_Letter(To\_Transmit.charAt(i));**

**}**

**}**

**}**

**if (New\_Y==-1 and Old\_Y==0){**

**display.fillRect(110, 2, 16, 12, SSD1306\_INVERSE);**

**display.fillRect(Old\_X\*12+2\*Old\_X, Old\_Y\*16 +16, 12, 16, SSD1306\_INVERSE);**

**}**

**if (New\_Y==0 and Old\_Y==-1){**

**display.fillRect(110, 2, 16, 12, SSD1306\_INVERSE);**

**display.fillRect(New\_X\*12+2\*New\_X, New\_Y\*16 +16, 12, 16, SSD1306\_INVERSE);**

**Prev\_Key\_read=Key\_read;**

**Old\_X=New\_X;**

**Old\_Y=New\_Y;;**

**}**

**if ((Old\_X!=New\_X or Old\_Y!=New\_Y) and Old\_Y!=-1 ){**

**if (New\_Y!=-1 )Highlight\_letter (New\_X,Old\_X,New\_Y,Old\_Y);**

**Old\_X=New\_X;**

**Old\_Y=New\_Y;**

**}**

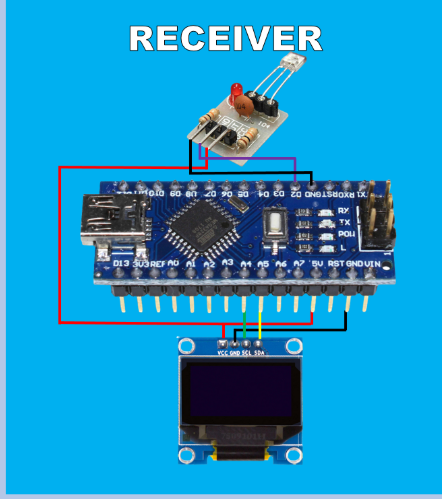
**}**

**display.display();**

**Prev\_Key\_read=Key\_read;**

**}**

**2) receiver circuit:-**

****

**Code:-**

**#include <Wire.h>**

**#include <Adafruit\_GFX.h>**

**#include <Adafruit\_SSD1306.h>**

**#include <avr/pgmspace.h>**

**const char alfabeth[] PROGMEM = {"ABCDEFGHIJKLMNOPQRSTUVWXYZ"};**

**const char morse\_A[] PROGMEM = ".-";**

**const char morse\_B[] PROGMEM = "-...";**

**const char morse\_C[] PROGMEM = "-.-.";**

**const char morse\_D[] PROGMEM = "-..";**

**const char morse\_E[] PROGMEM = ".";**

**const char morse\_F[] PROGMEM = "..-.";**

**const char morse\_G[] PROGMEM = "--.";**

**const char morse\_H[] PROGMEM = "....";**

**const char morse\_I[] PROGMEM = "..";**

**const char morse\_J[] PROGMEM = ".---";**

**const char morse\_K[] PROGMEM = "-.-";**

**const char morse\_L[] PROGMEM = ".-..";**

**const char morse\_M[] PROGMEM = "--";**

**const char morse\_N[] PROGMEM = "-.";**

**const char morse\_O[] PROGMEM = "---";**

**const char morse\_P[] PROGMEM = ".--.";**

**const char morse\_Q[] PROGMEM = "--.-";**

**const char morse\_R[] PROGMEM = ".-.";**

**const char morse\_S[] PROGMEM = "...";**

**const char morse\_T[] PROGMEM = "-";**

**const char morse\_U[] PROGMEM = "..-";**

**const char morse\_V[] PROGMEM = "...-";**

**const char morse\_W[] PROGMEM = ".--";**

**const char morse\_X[] PROGMEM = "-..-";**

**const char morse\_Y[] PROGMEM = "-.--";**

**const char morse\_Z[] PROGMEM = "--..";**

**const char \*const morse\_code[] PROGMEM = {morse\_A, morse\_B, morse\_C, morse\_D, morse\_E, morse\_F,**

**morse\_G, morse\_H, morse\_I, morse\_J, morse\_K, morse\_L,**

**morse\_M, morse\_N, morse\_O, morse\_P, morse\_Q, morse\_R,**

**morse\_S, morse\_T, morse\_U, morse\_V, morse\_W, morse\_X,**

**morse\_Y, morse\_Z};**

**char buffer[5];**

**#define OLED\_RESET -1// Reset pin # (or -1 if sharing Arduino reset pin)**

**Adafruit\_SSD1306 display(128, 64, &Wire, OLED\_RESET);**

**String rcvd="";**

**String Message="";**

**char Matched\_Letter;**

**int tol=10;**

**int unit=300;**

**#define LaserPin 2**

**int letter\_index=-1;**

**unsigned long Lastrun=0;**

**unsigned long Start;**

**unsigned long End;**

**unsigned long Signal\_Len=0;**

**unsigned long Pause\_Len=0;**

**boolean TransferInProgress=false;**

**struct MorseCodeSignal{**

**char Letter;**

**String Signal;**

**};**

**void setup() {**

**Serial.begin(9600);**

**attachInterrupt(digitalPinToInterrupt(2), Signal\_change,CHANGE);**

**// SSD1306\_SWITCHCAPVCC = generate display voltage from 3.3V internally**

**if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64**

**Serial.println(F("SSD1306 allocation failed"));**

**for(;;); // Don't proceed, loop forever**

**}**

**display.display();**

**delay(2000); // Pause for 2 seconds**

**// Clear the buffer**

**display.clearDisplay();**

**display.fillRect(0,0,128,16,SSD1306\_WHITE);**

**display.display();**

**}**

**void DrawSequence(int index, String instr){**

**int offset\_X;**

**int offset\_Y;**

**offset\_X=9+int(index/6)\*43;**

**offset\_Y=17+(index-int(index/6)\*6)\*8;**

**display.fillRect(offset\_X-2,offset\_Y,2,6,SSD1306\_WHITE);**

**for (int i=0; i<instr.length();i++ ){**

**if(instr.charAt(i)=='.'){**

**display.fillRect(offset\_X,offset\_Y,4,6,SSD1306\_WHITE);**

**display.fillRect(offset\_X,offset\_Y+2,2,2,SSD1306\_BLACK);**

**offset\_X+=4;**

**}**

**if(instr.charAt(i)=='-'){**

**display.fillRect(offset\_X,offset\_Y,8,6,SSD1306\_WHITE);**

**display.fillRect(offset\_X,offset\_Y+2,6,2,SSD1306\_BLACK);**

**offset\_X+=8;**

**}**

**}**

**display.display();**

**}**

**void DrawLetter(int index, char Letter){**

**int offset\_X;**

**int offset\_Y;**

**offset\_X=int(index/6)\*43;**

**offset\_Y=16+(index-int(index/6)\*6)\*8;**

**display.setCursor(offset\_X,offset\_Y);**

**display.setTextSize(1);**

**display.setTextColor(SSD1306\_WHITE);**

**display.println(Letter);**

**display.display();**

**}**

**void DrawMessage(String TextIn){**

**if (TextIn.length()>0){**

**display.fillRect(0,0,128,16,SSD1306\_WHITE);**

**display.setCursor(3,1);**

**display.setTextSize(2);**

**display.setTextColor(SSD1306\_BLACK);**

**display.println(TextIn);**

**display.display();**

**void Signal\_change(){**

**if (millis()-Lastrun>10){**

**if(digitalRead(LaserPin)==1) {**

**Start=millis();**

**Pause\_Len=Start-End;**

**if (TransferInProgress==false) {TransferInProgress=true;rcvd="";}**

**}**

**else {**

**End=millis();**

**Signal\_Len=End-Start;**

**}**

**Lastrun=millis();**

**}**

**}**

**char Match\_Letter(String input){**

**char Matched\_letter;**

**String Sequence;**

**for (int i = 0; i < 26; i++) {**

**strcpy\_P(buffer, (char \*)pgm\_read\_word(&(morse\_code[i]))); // Necessary casts and dereferencing, just copy.**

**Sequence=buffer;**

**if (Sequence==input) Matched\_letter=pgm\_read\_byte\_near(alfabeth+i);**

**}**

**return Matched\_letter;**

**}**

**void loop() {**

**if (TransferInProgress){**

**if (Signal\_Len>0){**

**if(Signal\_Len>=(unit- unit/tol) and Signal\_Len<=(unit+unit/tol)) {rcvd+='.';DrawSequence(letter\_index,rcvd);}**

**if(Signal\_Len>=(3\*unit- unit/tol) and Signal\_Len<=(3\*unit+unit/tol)) {rcvd+='-';DrawSequence(letter\_index,rcvd);};**

**// if(Signal\_Len<(unit- unit\*tolerance/100) or (Signal\_Len>(unit+unit\*tolerance/100)and Signal\_Len<(3\*unit- unit\*tolerance/100)) or Signal\_Len>(3\*unit+unit\*tolerance/100) )**

**Signal\_Len=0;**

**}**

**if (Pause\_Len>0){**

**if(Pause\_Len>(unit- unit/tol) and Signal\_Len<(unit+unit/tol)) {;}**

**if(Pause\_Len>(3\*unit- unit/tol) and Signal\_Len<(3\*unit+unit/tol)) {**

**Matched\_Letter=Match\_Letter(rcvd);**

**Message+=Matched\_Letter;**

**DrawMessage(Message);**

**Serial.print(Matched\_Letter);**

**DrawLetter(letter\_index,Matched\_Letter);**

**display.display();**

**Serial.println(rcvd);**

**rcvd="";**

**letter\_index++;**

**Matched\_Letter="";**

**}**

**Pause\_Len=0;**

**}**

**if((millis()-End)>8\*unit and TransferInProgress){**

**Matched\_Letter=Match\_Letter(rcvd);**

**Message+=Matched\_Letter;**

**DrawMessage(Message);**

**display.display();**

**Serial.print(Matched\_Letter);**

**DrawLetter(letter\_index,Matched\_Letter);**

**display.display();**

**Serial.println(rcvd);**

**rcvd="";**

**//letter\_index=0;**

**TransferInProgress=false;**

**Message="";**

**Pause\_Len=0;**

**WORKING PRINCIPLE:**

A **light-to-letter converter** typically refers to a device or system that detects light (usually a specific pattern or code) and converts it into a corresponding letter or character. This concept is often used in optical communication, assistive technologies (like devices for the visually or physically impaired), or optical character recognition (OCR) systems.

**Working Principle of a Light-to-Letter Converter**

**1. Light Detection**

* **Sensor Used:** A **photodiode**, **phototransistor**, or **LDR (light-dependent resistor)** is used to detect light.
* **Input Light Type:** The light may be constant, blinking, or modulated in a specific pattern (e.g., Morse code or barcode-style).

**2. Pattern Recognition or Decoding**

* The digitized signal is sent to a **microcontroller**, **DSP**, or **computer** that:
  + Detects the timing and/or intensity pattern of the light
  + Matches it against a predefined **codebook** (like Morse code, binary encoding, or ASCII)
  + Determines which **letter or character** the pattern corresponds to

**3. Letter Output**

* The system converts the detected pattern into a **letter or character** and:
  + Displays it on a screen
  + Sends it to another device
  + Speaks it via a speech synthesis module (for accessibility applications)

**Example Application: Morse Code Light Reader**

If someone blinks a flashlight in Morse code:

* The sensor detects short and long light pulses (dots and dashes).
* The microcontroller decodes the Morse code.
* The corresponding letter (e.g., ... → S) is displayed or spoken.

**Technologies Involved**

* **Light sensor** (LDR, photodiode)
* **Microcontroller** (Arduino, Raspberry Pi, etc.)
* **Signal processing** (via code or hardware)
* **Software algorithm** for decoding patterns
* **Display/Speech output**

**RESULT:-**

The light-to-letter converter is an innovative system that bridges optical signals and textual communication. By detecting and decoding light patterns—such as blinks, flashes, or encoded pulses—it effectively translates non-verbal, non-auditory input into readable letters or characters. This technology holds significant potential in assistive communication, optical data transmission, and educational tools.

Its working relies on the integration of light sensors, signal processing, and intelligent decoding algorithms,

often controlled by a microcontroller or computer. Depending on its application, the system can enhance

accessibility for individuals with disabilities, enable silent or remote communication, or even support automated recognition systems like OCR.

Overall, the light-to-letter converter showcases how light—a simple, universal medium—can be harnessed for meaningful and inclusive communication across various domains.

**Contribution:-**

**Ameer:-** connection of the circuit

**Piyush:-**coding

**Sahil:-**report making