UNIVERSITY OF MUMBAI

A PROJECT REPORT ON

BIDIRECTIONAL VISITORS COUNTER

SUBMITTED BY-

Advait Ambeskar	13103A0013
Sayli Dalvi	13013A0014
Hetal Gosavi	13103A0015

UNDER THE GUIDANCE OF–

PROF. RAJASHREE SOMAN
PROF. MUGDHA JOSHI

DEPARTMENT OF ELECTRONICS ENGINEERING
VIDYALANKAR INSTITUTE OF TECHNOLOGY
WADALA (EAST) MUMBAI – 400307

YEAR 2015-2016

CERTIFICATE

THIS IS TO CERTIFY THAT

Advait Ambeskar 13103A0013

Sayli Dalvi 13103A0014

Hetal Gosavi 13103A0015

have successfully completed the project titled

"BIDIRECTIONAL VISITOR COUNTER"

THIS PROJECT WAS UNDERTAKEN AS A PART OF THE CURRICULUM IN PARTIAL FULFILLMENT OF T.E. DEGREE IN ELECTRONICS ENGINEERING

(UNIVERSITY OF MUMBAI)

FOR THE ACADEMIC YEAR 2015-2016

PROJECT GUIDE 1 PROJECT GUIDE 2

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

This is to place on record my appreciation and deep feeling of gratitude for people without whom this project would never have seen the light of the day.

We wish to express my deep gratitude for **Mr. Milind Tadvalkar** for his extensive guidance, encouragement and all the facilities provided for projects.

We also wish to express my sincere thanks to **Dr Anjali Deshpande**, Head of Department, Electronics, VIT for extending her help for the project.

We also find immense sense of pleasure in thanking **Prof. Mugdha Joshi**, Department of Electronics for her help to complete this project successfully.

Finally we wish to express my gratitude to **Prof. Rajashree Soman**, Department of Electronics and all members of faculty and our friends who contributed their valuable advice and help to complete this project successfully.

Advait Ambeskar 13103A0013

Sayli Dalvi 13103A0014

Hetal Gosavi 13103A0015

ABSTRACT

The following project describes the construction, working and application of **BIDIRECTIONAL VISITOR COUNTER**. This project is of importance in situations where a certain counting capacity is to be maintained. Having said so, it is essential that we can have a numeric count of incidences to reach a certain statistical or logical output. Through this project, we intend to supplement various usages of a counter by increasing its efficiency to count a certain incidence in a given room. The aim of this project is to create a basic counter capable of bidirectional counting, which can be utilized as a small component of bigger projects where efficient counting is of utmost importance or to utilize the output of the circuit to generate a dataset which can be intelligently utilized to attain statistically accurate output.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Today automated decision making is of utmost importance, especially when we consider the need of quick and efficient responses to be provided for statistical responses given to a certain query. Considering the implications of numeric count of a given occurrence and the conditional responses that we may expect as the output, an efficient counter holds utmost importance in today's world.

Our design idea for the project was to replicate a distinct entry-and-exit room, where a person would enter a room, from one side, and exit the room, from the other side. Many a times we need to monitor the person/people visiting the same place like seminar hall, conference room or shopping mall or temple. This project can be used to count and display the number of visitors entering inside any conference room or seminar hall. This is a bidirectional counter which means it works in a two way. That means counter will incremented if person enters the room and will be decremented if person leaves the room. 7-segmeted LED displays this value which is placed outside the room.

This system is useful for counting the number of people in an auditorium or halls for seminar to avoid congestion. Moreover it can also be used to check the number of people who have come to an event or a museum to watch a certain exhibition. Arduino is a reliable circuit that takes over the task of counting the number of persons/visitors in the room very accurately. We will be showing both In count i.e. number of people entering the room and Out count i.e. number of people exiting the room on 7-segmented display. A LDR is used to monitor the person entering and exiting the room.

CHAPTER 2

BLOCK DIAGRAM AND CIRCUIT DIAGRAM

Block Diagram of Bidirectional Visitor Counter

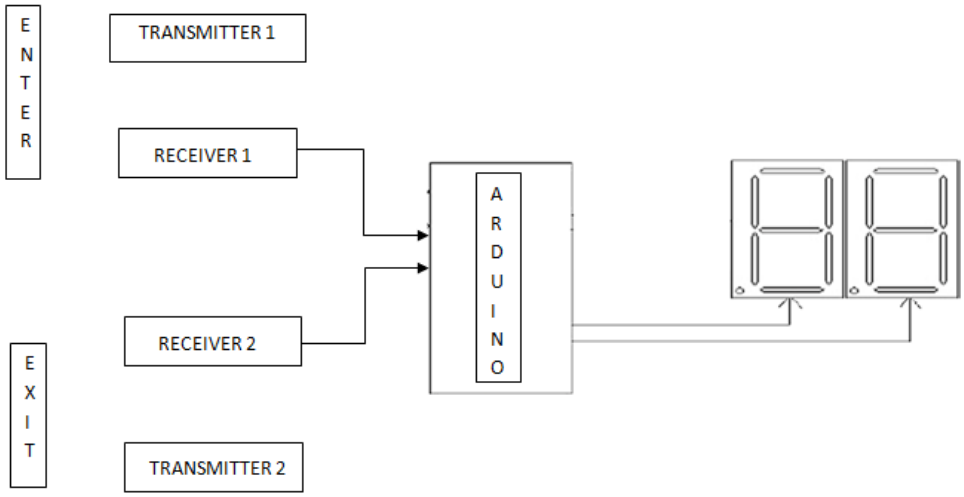


Fig 2.1 Block Diagram of Bidirectional Visitor Counter

Circuit Diagram of Bidirectional Visitor Counter

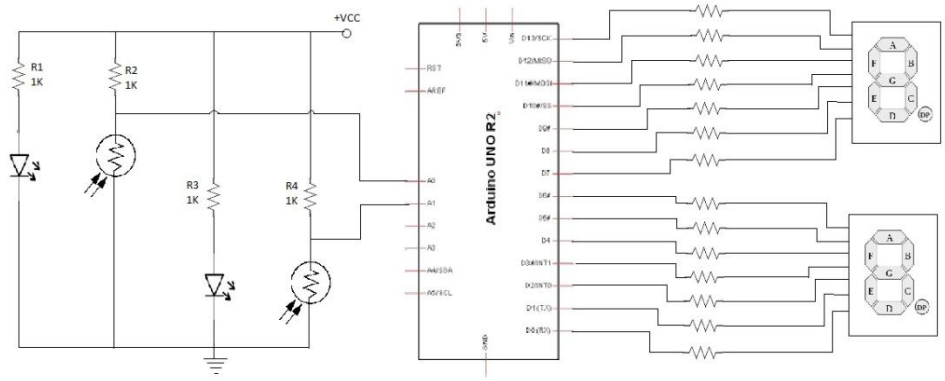


Fig 2.2 Circuit Diagram of Bidirectional Visitor Counter

CHAPTER 3

LIST OF COMPONENTS

Sr. No	Components	Quantity	Cost
1.	Arduino Uno R3	1	700
2.	7-segment display	2	30
3.	LDR Sensors	2	10
4.	LEDs	4	8
5.	Resistors	18	9
6.	Male headers	2	15
7.	Connecting wires	16	48
8.	Copper clad Epoxy sheet	1	30

Chapter 4

Description of Hardware

Arduino Uno R3

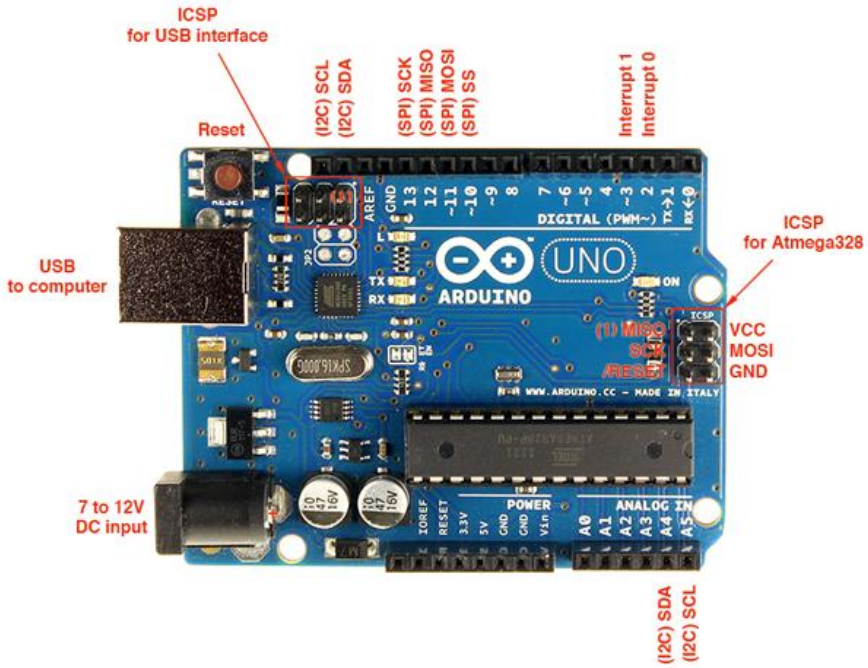


Fig 4.1 Arduino Uno R3

Technical Specifications:

Microcontroller…….....ATmega328P
Operating Voltage……...5V
Input Voltage (recommended)……7-12V
Input Voltage (limit)……...…6-20V
Digital I/O Pins……...14 (of which 6 provide PWM output)
PWM Digital I/O Pins……6
Analog Input Pins……...6
DC Current per I/O Pin……..20 mA
DC Current for 3.3V Pin……50 mA
Flash Memory……....32 KB (ATmega328P)
SRAM…….2 KB (ATmega328P)
EEPROM……....1 KB (ATmega328P)
Clock Speed...……16 MHz
Length…….68.6 mm
Width……..53.4 mm

Weight……25 g

Programming:

The Arduino Uno can be programmed with the Arduino Software (IDE). Select "Arduino Uno" from the Tools > Board menu (according to the microcontroller on your board).

The ATmega328 on the Arduino Uno comes preprogramed with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

Power:

The Arduino Uno board 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 from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

Vin

It is the Input Voltage to the Arduino board when it is 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.

5V

This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

3.3V

A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND:

Ground pins.

IOREF:

This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Memory:
The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output
See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical.ATmega328P. Each of the 14 digital pins on the Uno 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 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

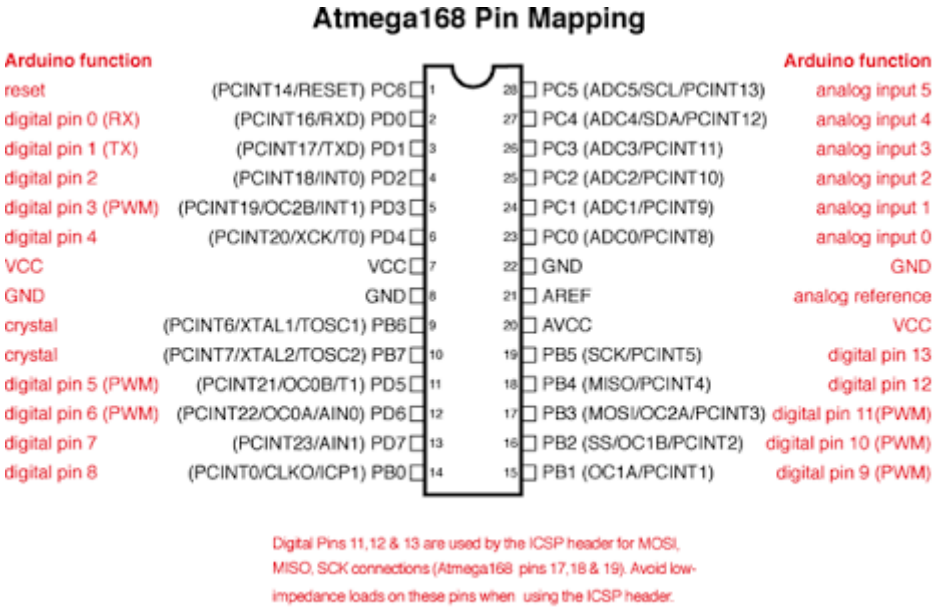


Fig 4.2 Atmega168 Pin Mapping

In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
The Uno has 6 analog inputs, labelled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.

There are a couple of other pins on the board:

AREF: Reference voltage for the analog inputs. Used with analogReference().

Reset: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication
Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows serial communication on any of the Uno's digital pins.
The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Automatic (Software) Reset:

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data

when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

LIGHT DEPENDENT RESISTORS

A Light Dependent Resistor (LDR) or a photo resistor is a device whose resistivity is a function of the incident electromagnetic radiation. Hence, they are light sensitive devices. They are also called as photo conductors, photo conductive cells or simply photocells. They are made up of semiconductor materials having high resistance. There are many different symbols used to indicate a LDR , one of the most commonly used symbol is shown in the figure below. The arrow indicates light falling on it.

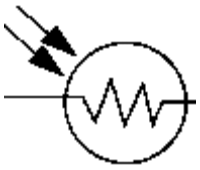


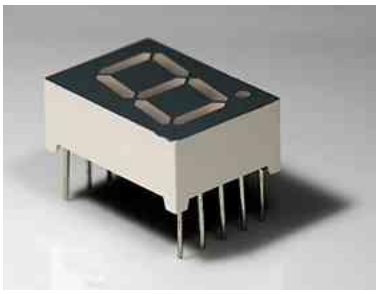
Fig 4.3 LDR

A light dependent resistor works on the principle of photo conductivity. Photo conductivity is an optical phenomenon in which the materials conductivity (Hence resistivity) reduces when light is absorbed by the material.

When light falls i.e. when the photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band. These photons in the incident light should have energy greater than the band gap of the semiconductor material to make the electrons jump from the valence band to the conduction band.

Hence when light having enough energy is incident on the device more & more electrons are excited to the conduction band which results in large number of charge carriers. The result of this process is more and more current starts flowing and hence it is said that the resistance of the device has decreased .This is the most common working principle of LDR.

SEVEN SEGMENT DISPLAY



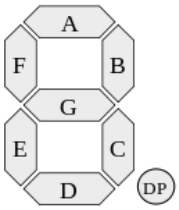


Fig 4.4 Seven Segment Display

Seven-segment display (SSD), or seven-segment indicator , is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The seven segments are arranged as a rectangle of two vertical segments on each side with one horizontal segment on the top, middle, and bottom. Additionally, the seventh segment bisects the rectangle horizontally. There are also fourteen-segment displays and sixteen-segment displays (for full alphanumeric); however, these have mostly been replaced by dot matrix displays.

The segments of a 7-segment display are referred to by the letters A to G, where the optional decimal point (an "eighth segment", referred to as DP) is used for the display of non-integer numbers.

LIGHT EMITTING DIODES

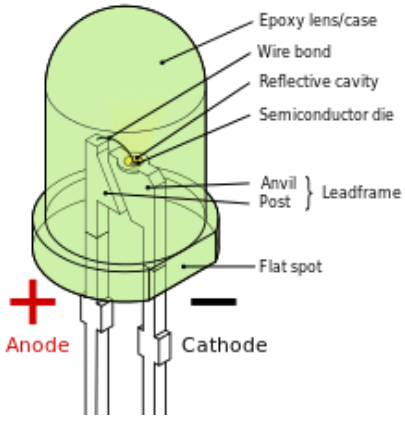


Fig 4.5 LED

A light-emitting diode (LED) is a two- lead semiconductor light source. It is a p–n junction diode, which emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is

called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor. An LED is often small in area (less than 1 mm 2) and integrated optical components may be used to shape its radiation pattern.

CHAPTER 5

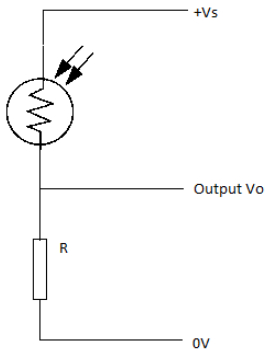
WORKING OF HARDWARE

Programming the Arduino:

- Open the Arduino IDE.
- Write the code.
- Save the sketch. On the File menu, click Save As.
- Upload the sketch to your Arduino and watch the output.
- Connect the Arduino to computer with USB cable.
- In the IDE, on the toolbar, click the Upload button.
- On the toolbar, click the serial monitor button.
- Detecting changes.

Interfacing LDR with Arduino:

- Output of the LDR is analog.
- The Output of the LDR is connected to the analog pins of Arduino.
- The Arduino is given a decided threshold, to work upon, such that we can utilize it to get decisive output whether the count should increase or decrease.
- The count is then incremented or decremented, based on the LDR Values.
- The Count is then sent to the 7-segment display.



Shown above is the breadboard connection. The LDR and 1K ohms resistor are powered by supply voltage. A wire is connected from this circuit to analog input 0 i.e. A_0 on the Arduino. Resistors lower the voltage passing through them, and so to read the changes in light from the circuit, we use Arduino's analog value to an integer in the range 0 through 1023.

When the photo resistor is exposed to light, its resistance decreases and so the voltage reading will be higher. When the light is blocked, the resistance of the photo resistor increases and so the voltage reading will be lower.

The photo resistor is a two-terminal passive component and has no polarity- it does not matter which way round you place it in the circuit.

To check everything is functioning correctly, we can create a basic sketch that reads the voltage change.

```
//code for interfacing of LDR with Arduino

int sensA;

int sensB;

int thresh;

int ctr=0;

int lights=9;

int sens_cutoff=A2;

int relay=2;

void setup()

{

	Serial.begin(9600);

}

void loop()

{

	sensA = analogRead(A0);

	sensB= analogRead(A1);

	thresh = 200;

	if(sensA<thresh && sensB>thresh)

	{

		ctr=ctr+1;

		delay(900);

	}
```

```
        else
        {
                ctr=ctr;
        }

        if(sensA>thresh && sensB<thresh)
        {
        ctr=ctr-1;
        delay(900);
        }
        else
        {
        ctr=ctr;
        }

        Serial.print("sensA is :  ");
        Serial.print(sensA   );
        Serial.print("      sensB is :  ");
        Serial.print(sensB    );
        Serial.print("     counter :  ");
        Serial.print(ctr);
        Serial.print("  thresh val : ");
        Serial.println(thresh);
}
```

The exact values output from the sketch above will vary depending on several factors:

1. The power supply of the Arduino. Particularly when powered over a USB cable, it is common for the Arduino's 5V power supply to be a little less than that ideal;
2. The minimum and maximum resistance values of the photo resistor used;
3. The accuracy of the 10K ohms resistor;
4. The construction of the breadboard and wires used – both of these have small levels of resistance that can affect the ADCs;
5. And the amount of ambient light in the room.
6. It is more important to detect changes in the light level than to be concerned with the actual numbers.

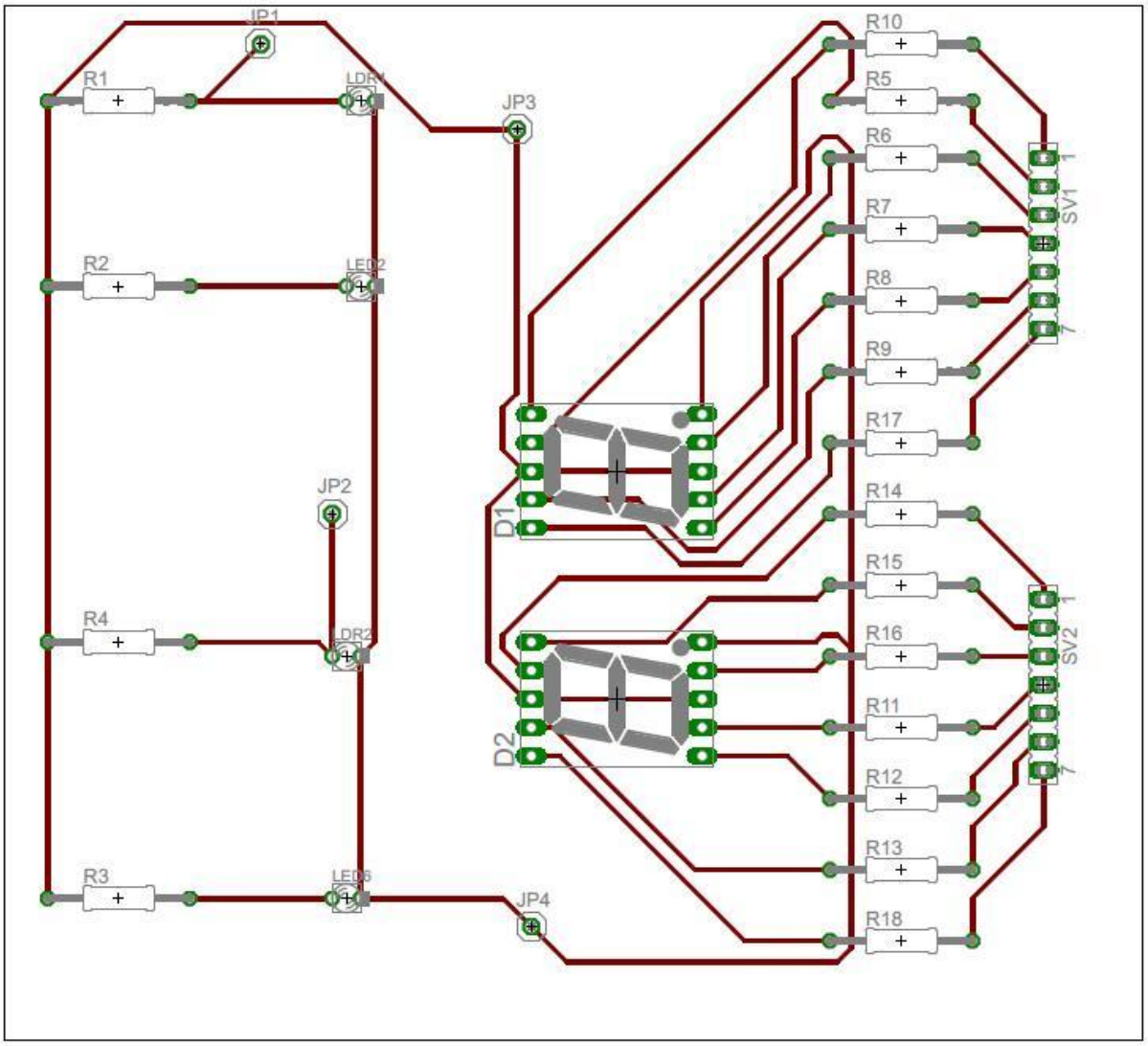


Fig 5.1 SCHEMATIC OF THE IMPLEMENTED HARDWARE

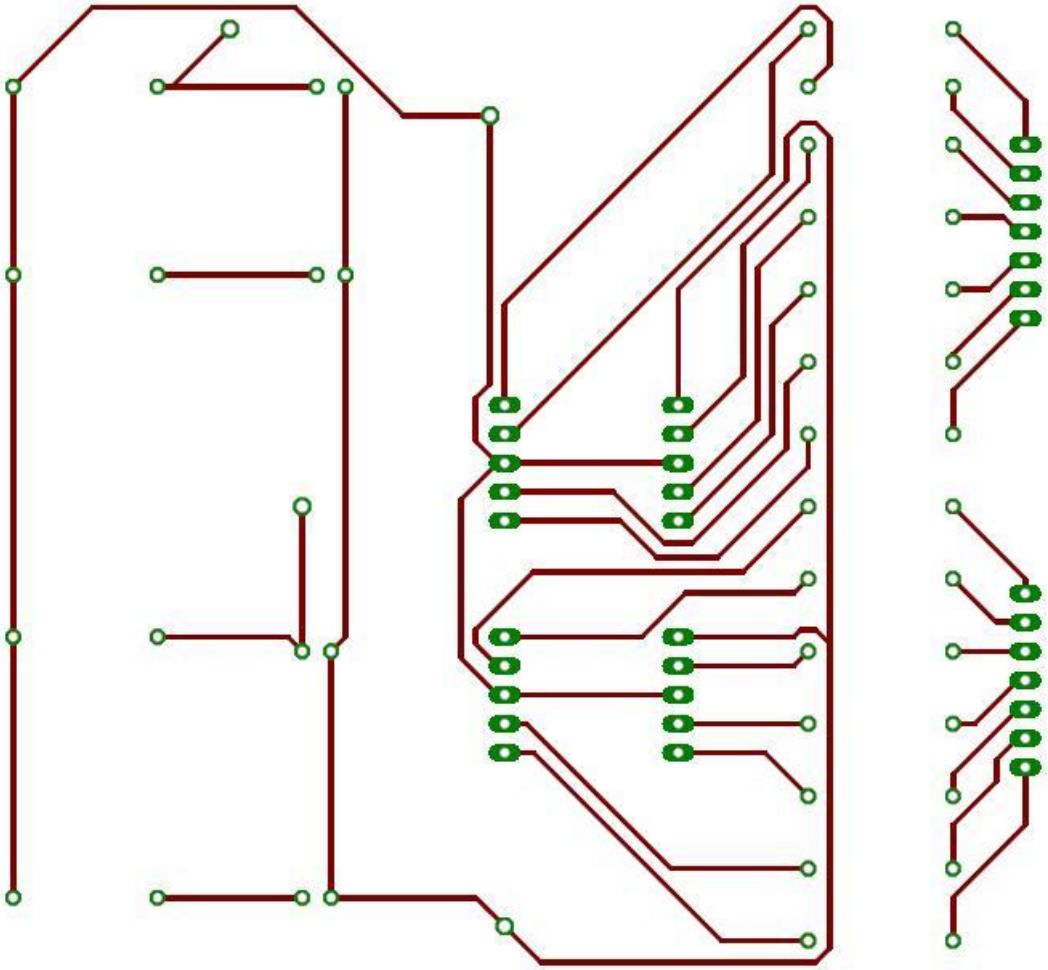


Fig 5.2 PCB LAYOUT OF BIDIRECTIONAL VISITOR COUNTER

CHAPTER 6

ADVANTAGES AND DISADVANTAGES

Advantages of Bidirectional Visitor Counter:

1. No need of human intervention.
2. Can work 24x7 without any problem.
3. Low cost of building.
4. Easy Implementation.
5. A Single Arduino can be used to control a large number of rooms.
6. The count does not depend on the precursor factualities.
7. The project can be modified to enable utilization of multiple exit and entry points.

Disadvantages of Bidirectional Visitor Counter:

1. IR sensor cannot detect if multiple people are entering at one time.
2. The Circuit cannot account for same door or path being used for both entry and exit.
3. Output of LDR is not entirely reliable, as it depends on ambience of the surrounding. However, we can utilize the Laser Diodes for increasing reliability for a certain higher economic cost.

Chapter 7

APPLICATIONS

As this project ‘Automated Car Parking Slot Indicator’ is itself a practical application based project. This project can be implemented in following places:

1. Offices.
2. Public Places
3. Restrooms at Railway Stations.
4. Roadways to understand Traffic Density.

Chapter 8

Conclusion and Future Scope

Conclusion:

This prototype system is automated and can be applied in the real counter situation. Using this project, we can develop more complex logic to enable reduction in disadvantages or bugs in the code.

By the end of this project:

1. Coding and compiling of a C program in Arduino IDE is studied.
2. Hardware implementation by connecting Schematic and making Board layout EAGLE is done successfully.
3. The hardware kit is tested successfully by embedding the C program – file in the Arduino UNO
4. The operation of microcontroller is analyzed in simulation and practically.

Future Scope:

1. Lights can be turned ON/OFF according to the number of people in the room
2. We can check the ambient light intensity and then decide if the light needs to be turned ON or OFF.
3. Metal detector can be added for security reasons. Security Camera's Activated based on Room Count can be implemented.
4. Security Systems can be implemented to avoid any system failure or theft conditions.

APPENDIX A

```
//Arduino Code designed for the project
int sensA,sensB,thresh,ctr,n;
int units,tens;
int gx=0,fx=1,ax=2,bx=3,ex=4,dx=5,cx=6;
int g=7,f=8,a=9,b=10,e=11,d=12,c=13;
int onex()
{       digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,0);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,0);     digitalWrite(ex,0);
        digitalWrite(fx,0);     digitalWrite(gx,0);
}
int twox()
{       digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,1);     digitalWrite(cx,0);
        digitalWrite(dx,1);     digitalWrite(ex,1);
        digitalWrite(fx,0);     digitalWrite(gx,1);
}
```

```
int threex()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,1);     digitalWrite(ex,0);
        digitalWrite(fx,0);     digitalWrite(gx,1);
}
int fourx()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,0);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,0);     digitalWrite(ex,0);
        digitalWrite(fx,1);     digitalWrite(gx,1);
}
int fivex()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
```

```
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,0);     digitalWrite(cx,1);
        digitalWrite(dx,1);     digitalWrite(ex,0);
        digitalWrite(fx,1);     digitalWrite(gx,1);
}
int sixx()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,0);     digitalWrite(cx,1);
        digitalWrite(dx,1);     digitalWrite(ex,1);
        digitalWrite(fx,1);     digitalWrite(gx,1);
}
int sevenx()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,0);     digitalWrite(ex,0);
        digitalWrite(fx,0);     digitalWrite(gx,0);
}
```

```
int eightx()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,1);     digitalWrite(ex,1);
        digitalWrite(fx,1);     digitalWrite(gx,1);
}
int ninex()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
        digitalWrite(e,0);      digitalWrite(f,0);
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,1);     digitalWrite(ex,0);
        digitalWrite(fx,1);     digitalWrite(gx,1);
}
int zerox()
{
        digitalWrite(a,0);      digitalWrite(b,0);
        digitalWrite(c,0);      digitalWrite(d,0);
         digitalWrite(e,0);      digitalWrite(f,0);
```

```
        digitalWrite(g,0);      digitalWrite(ax,1);
        digitalWrite(bx,1);     digitalWrite(cx,1);
        digitalWrite(dx,1);     digitalWrite(ex,1);
        digitalWrite(fx,1);     digitalWrite(gx,0);
}
int one()
{
        digitalWrite(ax,0);     digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
        digitalWrite(gx,0);     digitalWrite(a,0);
        digitalWrite(b,1);      digitalWrite(c,1);
        digitalWrite(d,0);      digitalWrite(e,0);
        digitalWrite(f,0);      digitalWrite(g,0);
}
int two()
{
         digitalWrite(ax,0);    digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
        digitalWrite(gx,0);     digitalWrite(a,1);
        digitalWrite(b,1);      digitalWrite(c,0);
        digitalWrite(d,1);      digitalWrite(e,1);
        digitalWrite(f,0);      digitalWrite(g,1);
}
```

```
int three()
{
        digitalWrite(ax,0);     digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
        digitalWrite(gx,0);     digitalWrite(a,1);
        digitalWrite(b,1);      digitalWrite(c,1);
        digitalWrite(d,1);      digitalWrite(e,0);
        digitalWrite(f,0);      digitalWrite(g,1);
}
int four()
{
        digitalWrite(ax,0);     digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
        digitalWrite(gx,0);     digitalWrite(a,0);
        digitalWrite(b,1);      digitalWrite(c,1);
        digitalWrite(d,0);      digitalWrite(e,0);
        digitalWrite(f,1);      digitalWrite(g,1);
}
int five()
{
         digitalWrite(ax,0);    digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
```

```
        digitalWrite(gx,0);     digitalWrite(a,1);
        digitalWrite(b,0);      digitalWrite(c,1);
        digitalWrite(d,1);      digitalWrite(e,0);
        digitalWrite(f,1);      digitalWrite(g,1);
}
int six()
{
         digitalWrite(ax,0);   digitalWrite(bx,0);
        digitalWrite(cx,0);    digitalWrite(dx,0);
        digitalWrite(ex,0);    digitalWrite(fx,0);
        digitalWrite(gx,0);    digitalWrite(a,1);
        digitalWrite(b,0);     digitalWrite(c,1);
        digitalWrite(d,1);     digitalWrite(e,1);
        digitalWrite(f,1);     digitalWrite(g,1);
}
int seven()
{
         digitalWrite(ax,0);    digitalWrite(bx,0);
        digitalWrite(cx,0);    digitalWrite(dx,0);
        digitalWrite(ex,0);    digitalWrite(fx,0);
        digitalWrite(gx,0);    digitalWrite(a,1);
        digitalWrite(b,1);     digitalWrite(c,1);
        digitalWrite(d,0);     digitalWrite(e,0);
        digitalWrite(f,0);     digitalWrite(g,0);
}
```

```
int eight()
{
        digitalWrite(ax,0);     digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
        digitalWrite(gx,0);     digitalWrite(a,1);
        digitalWrite(b,1);      digitalWrite(c,1);
        digitalWrite(d,1);      digitalWrite(e,1);
        digitalWrite(f,1);      digitalWrite(g,1);
}
int nine()
{
        digitalWrite(ax,0);     digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
        digitalWrite(gx,0);     digitalWrite(a,1);
        digitalWrite(b,1);      digitalWrite(c,1);
        digitalWrite(d,1);      digitalWrite(e,0);
        digitalWrite(f,1);      digitalWrite(g,1);
}
int zero()
{
        digitalWrite(ax,0);     digitalWrite(bx,0);
        digitalWrite(cx,0);     digitalWrite(dx,0);
        digitalWrite(ex,0);     digitalWrite(fx,0);
```

```
        digitalWrite(gx,0);     digitalWrite(a,1);
        digitalWrite(b,1);      digitalWrite(c,1);
        digitalWrite(d,1);      digitalWrite(e,1);
        digitalWrite(f,1);      digitalWrite(g,0);
}
void setup()
{
 pinMode(0,OUTPUT); pinMode(1,OUTPUT);
 pinMode(2,OUTPUT); pinMode(3,OUTPUT);
 pinMode(4,OUTPUT); pinMode(5,OUTPUT);
 pinMode(6,OUTPUT); pinMode(7,OUTPUT);
 pinMode(8,OUTPUT); pinMode(9,OUTPUT);
 pinMode(10,OUTPUT); pinMode(11,OUTPUT);
 pinMode(12,OUTPUT); pinMode(13,OUTPUT);
 Serial.begin(9600);
}
void loop()
{
        int n=0;
        sensA=analogRead(A0);
        sensB=analogRead(A1);
        thresh=500;
        if(sensA>thresh&&sensB<thresh)
        {ctr++;delay(n);
  }
```

```
  if(sensA<thresh&&sensB>thresh)
  {    ctr--;        delay(n);
  }
  if(sensA>thresh||sensB>thresh)
  {    sensA=0;    sensB=0;    delay(n+500);
    n=millis();
  }
       units=ctr%10;                          // GET UNITS AND TENS DIGITS OF COUNTER
       tens=ctr/10;
if(units==0)
  zero();
if(units==1)
one();
if(units==2)
  two();
if(units==3)
  three();

if(units==4)
  four();
if(units==5)
  five();
if(units==6)
  six();
```

```
if(units==7)
  seven();
if(units==8)
  eight();
if(units==9)
  nine();
if(tens==0)
  zerox();
if(tens==1)
  onex();
if(tens==2)
  twox();
if(tens==3)
  threex();
if(tens==4)
  fourx();
if(tens==5)
  fivex();
if(tens==6)
  sixx();
if(tens==7)
  sevenx();
if(tens==8)
  eightx();
if(tens==9)
```

ninex();}}