MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

TECHNICAL REPORT:

HOME AUTOMATION USING SPEECH RECOGNITION

PROJECT GUIDE:

Prof. D.K. Raghuwanshi

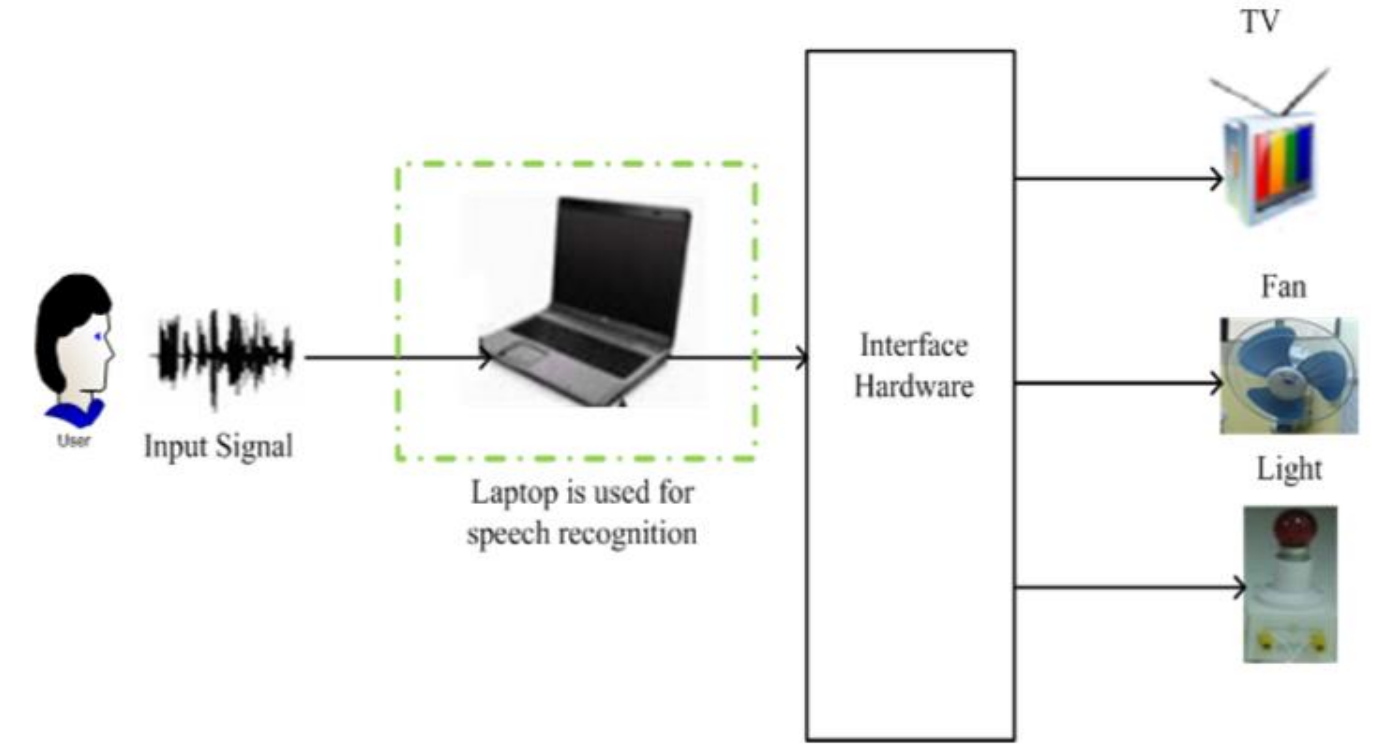
1. Akshay Patidar (141114010)
2. Gaurav Thakre (141114045)
3. Abhishek Mishra (141114144)
4. Shivam Choudhary (141114035)
5. Balveer Narvariya (141114068)

TABLE OF CONTENTS:

1. **ABSTRACT**
2. **COMPONENT'S LIST**
3. **SPEECH RECOGNITION SYSTEM**
    - A. **Speech Analysis**
    - B. **Pattern Recognition**
4. **ELECTRONIC CONTROL SYSTEM**
    - A. **Transmission Section**
    - B. **Receiver Section**
5. **CODE SECTION**
    - A. **MATLAB CODE**
    - B. **Process of Speech Recognition**
    - C. **Arduino Code**
6. **POWER SUPPLY DESIGN**
7. **PHOTOS OF WORKING MODEL**
8. **APPLICATIONS**

ABSTRACT

Speech Recognition is a technology allowing the computer to identify and understand words spoken by a person using a microphone or telephone. Using a set of pre-programmed commands and instructions, user can talk with computer. Computer system that understands input speech enables user to have conversations with the computer. User and the computer speaking as commands or in response to events, input, or other feedback would be included in these conversations. Speaking is easier and more sensitive than selecting buttons and menu items. Human speech has changed over many thousands of years to become an efficient method of sharing information and giving instructions. The goal of this paper is to implement smart home appliances controlled system that can be operated by spoken words. The spoken words chosen for recognition are "Fan On", "Fan Off", "Light On", "Light Off", "TV On" and "TV Off". The proposed system is composed of two main parts: speech recognition system and home appliances control system. The first and main part of system is personal computer based speech recognition system. Speech recognition system is composed of also two main parts. The first part is speech recognition based on Digital Signal Processing. Second part is interfacing with hardware. The overall block diagram is shown in Figure.



COMPONENT'S LIST

1) TRANSMITTER SECTION:-

a) MICROCONTROLLER- ARDUINO UNO
b) RF TRANSMITTER MODULE(433MHz)

2) RECEIVER SECTION:-

a) MICROCONTROLLER- ARDUINO UNO
b) RF RECEIVER MODULE(433MHz)

3) POWER SUPPLY SECTION:-

a) STEP DOWN TRANSFORMER
b) RELAY MODULE
c) CAPACITOR(470uF and 0.1nF)
d) DIODES(IN4007)
e) CONTROLLED APPLIANCES
f) VOLTAGE REGULATOR(7805)
g) PERFBOARD

SPEECH RECOGNITION SYSTEM

A speech recognition roughly consists of two portions. They are speech analysis and pattern recognition.

A. Speech Analysis

The purpose of the speech analysis block is to transform the speech waveform into a parsimonious representation which characterizes the time varying properties of the speech. The speech analysis typically includes two modules, namely data acquisition and feature extraction. The data acquisition module usually contains a microphone and a code from which digitized speech data are generated. The feature extraction is the computation of a sequence of feature vectors which provides a compact representation of the given speech signal. The feature extraction is done by the method of cross-correlation. The speech signal is separated into overlapped fixed-length frames.

B. Pattern Recognition

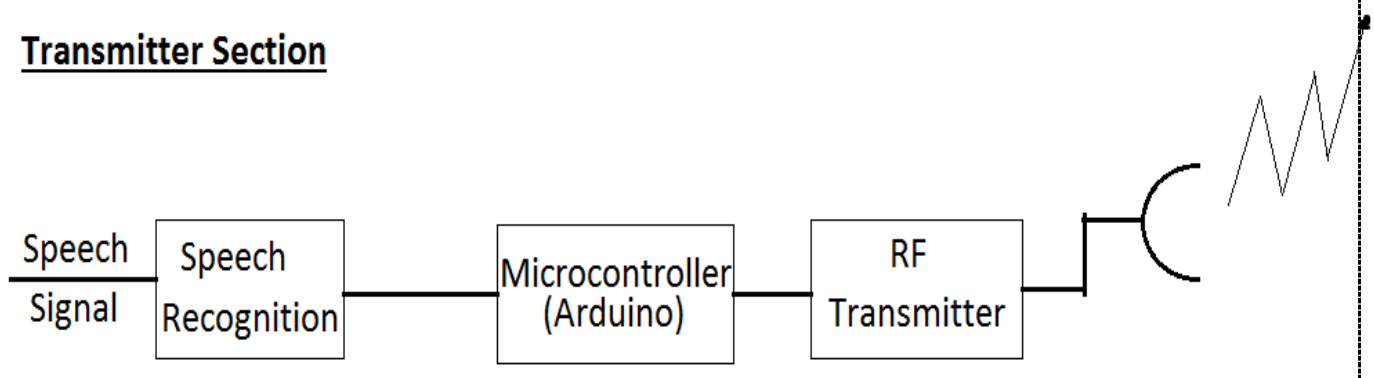
The speech signal is first analyzed and a feature representation is obtained for comparison with stored reference templates in the pattern matching block. A decision scheme determines the word of the unknown speech based on the matching scores with respect to the stored reference patterns.

ELECTRONIC CONTROL SYSTEM

The block diagram of home appliances control system for speech recognition is illustrated in Fig. In this diagram, speech instruction is firstly taken as input to control home appliances and then a microphone is used to record the person speech. Secondly, the speech instruction is caught and transferred the analog signal to digital signal and the recorded speech is sent to the speech based verification/identification system. Thirdly, the digital information of speech instruction is processed and compared by using the MATLAB programming. Fourthly, the digital information of speech instruction is outputted through USB port. Finally, Arduino receives data from speech recognition block and gives instructions to control home appliances. The control system consists of two sections: transmission section and receiving sections.

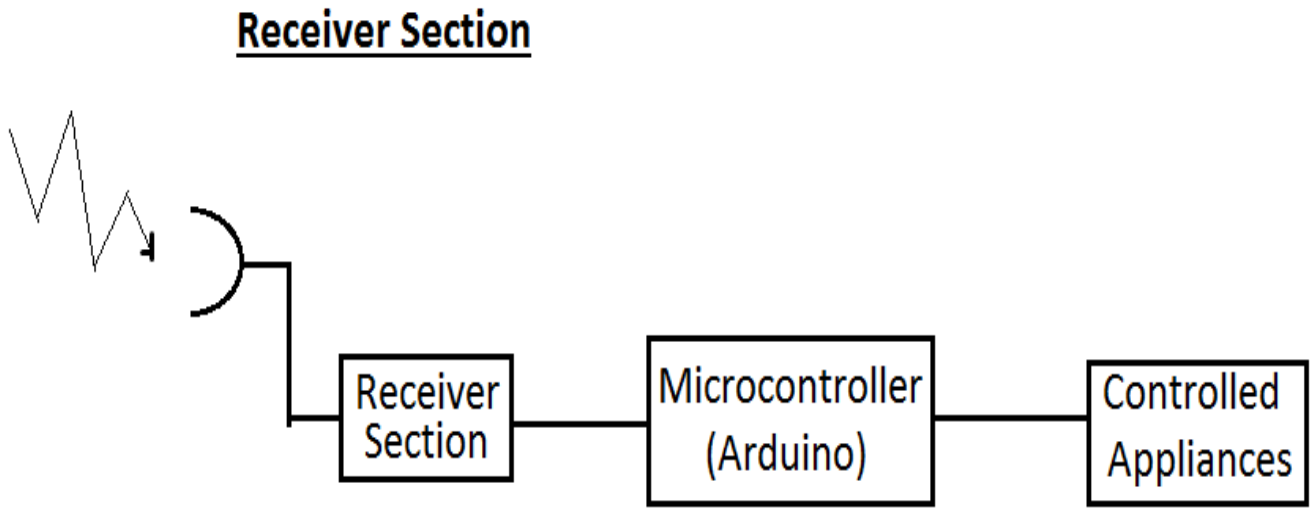
A. Transmission Section

In the transmission section, there is 434MHz RF transmitter module (Radio Frequency transmitter module). The USB 2.0 is used to carry the signal from PC to Microcontroller unit. The signal is retransmitted with baud rate 9600 for RF transmission by 434MHz RF transmitter module. This module has four pins: supply pin, data pin, GND pin, and ANT pin. Specifications of wireless transmitter module are (1) transmit power: 1W, (2) operating frequency: 315MHZ~433.92MHZ, (3) operating temperature:-40℃~80℃, (4) operating voltage: 3V~5V and (5) modulation type: ASK.



B. Receiver Section

The receiver section consists of 434MHz RF receiver module, Arduino UNO, relays, relay drivers and motor driver. In this section, Receiver section firstly accept radio signal and then microcontroller read radio signal with baud rate 9600. Microcontroller drives relay which controls other appliances.



CODE

1. SPEECH RECOGNITION THROUGH MATLAB:-

```
function speechrecognition231()
%Speech Recognition Using Correlation Method
%Write Following Command On Command Window
%speechrecognition('test.wav')
KP=audiorecorder(44100,16,2);
disp('enter the command');
recordblocking(KP,2.5790);
kq=getaudiodata(KP);
save kq.mat ;
audiowrite('voice1.wav',kq,44100);
voice=wavread('voice1.wav');
x=voice;
x=x';
x=x(1,:);
x=x';
y1=wavread('start.wav');
y1=y1';
y1=y1(1,:);
y1=y1';
z1=xcorr(x,y1);
m1=max(z1);
l1=length(z1);
t1=-((l1-1)/2):1:((l1-1)/2);
t1=t1';
%subplot(3,2,1);
%plot(t1,z1);
y2=wavread('lightoff.wav');
```

```
y2=y2';
y2=y2(1,:);
y2=y2';
z2=xcorr(x,y2);
m2=max(z2);
l2=length(z2);
t2=-((l2-1)/2):1:((l2-1)/2);
t2=t2';
%subplot(3,2,2);
%figure
%plot(t2,z2);
m6=40;
a=[m1 m2 m6];
m=max(a);
h=wavread('allow.wav');
if m<=m1
   soundsc(wavread('start.wav'),50000)
        soundsc(h,50000)
           i=1;
    answer=1; % this is where we'll store the user's answer
    arduino=serial('COM4','BaudRate',115200); % create serial communication
object on port COM4
     fopen(arduino); % initiate arduino communication

    while (i<275)
    fprintf(arduino,'%s',char(answer)); % send answer variable content to
arduino
    answer=1;i=i+1;
    end
    fclose(arduino); % end communication with arduino



elseif m<=m2
    soundsc(wavread('lightoff.wav'),50000)
        soundsc(h,50000)
       i=1;
    answer=2; % this is where we'll store the user's answer
    arduino=serial('COM4','BaudRate',115200); % create serial communication
object on port COM4
     fopen(arduino); % initiate arduino communication

    while (i<500)
    fprintf(arduino,'%s',char(answer)); % send answer variable content to
arduino
    answer=2;
    i=i+1;
    end
    fclose(arduino); % end communication with arduino


else
   soundsc(wavread('accessdenied.wav'),50000)

end
```

Process of Speech Recognition

Speech recognition is implemented in the MATLAB environment. In the first stage of this block we have recorded some commands to control the appliances like Light On, Light Off etc. these commands are our reference commands by which we will crosscheck the input command given by user to control the appliances.

Algorithm used for matching the given input with reference data set is **Cross Correlation**. Cross correlation is a method to measure the degree of similarity between two different signals to find a feature representing the signal. Cross Correlation of two discrete time signals is given by

$$r_{xy} = r(m) = \sum_{n=-\infty}^{\infty} x(n)y(n+m), m = 0, \pm 1, \pm 2, \pm 3, \ldots.$$

Walkthrough of code

In the first stage MATLAB will prompt user to give the command, using the sound card of pc we are recording this signal and MATLAB will convert this signal into a useful data set. Recording of this signal is done by using inbuilt functions like AUDIORECORDER, which is making an object which is capable to hold the audio data.

Now the correlation between, the command given by user and pre recorded signals is done and based on the similarity it will give a value which will be used to decide the which command was spoken by the user.

By this we have identified the command given by user and for different commands we have some control strings which will be sent to our transmitter section where our microcontroller ARDUINO will send this control string using RF modules to our receiver section.

Detailed Description of Commands Used

1. **Audiorecorder(FS, 'N', number of channels)-**
   This command is used to create a audiorecorder Object which is capable of holding audio type of data. It requires following three parameters Sampling rate 'FS' to sample the analog signal, 'N' stands for no of bits MATLAB will use to store each sample of audio data, no of channel are for mono or stereo recording.
2. **Recordblocking(a,x)-**
   This command is used for taking the analog input. Parameters given to this function are audiorecorder object and time in seconds for which it will record the analog input.

3. **Getaudiodata(a**)-

   This command is used to take the sampled values of signal which are stored in our audiorecorder object. This requires only a audiorecorder object.

4. **audiowrite('FILENAME', Y, FS)**

   This command is to covert the audio data into a wav file which is default format used by MATLAB to read audio files.

   Parameters-filename (it will create a wavfile of name specified in filename), Y is matrix where our audio data is residing and FS is the sampling rate used for conversion.

5. **wavread('FILENAME.WAV')**

   This command is used to read a wavfile into MATLAB workspace.

6. **xcorr(x,y)**

   This command is used to perform correlation between two signals. This is used to perform the correlation of two signals which we are using for matching the given input command with our stored references.

Serial commands used to transfer data from MATLAB to ARDUINO

1. **ARDUINO=serial ('COM4','BAUDRATE', N)**

   Serial command is used to create a serial object named ARDUINO. It requires following parameters 'COM' port where our ARDUINO is connected to our pc, 'BAUDRATE' for synchronizing communication between ARDUINO and MATLAB.

2. **fopen(ARDUINO)**

   This is used to start the communication between ARDUINO and MATLAB.

3. **fprintf()**

   This command is used to write the value (control string generated by MATLAB) to the ARDUINO.

4. **fclose(ARDUINO)**

   This is used to terminate the connection between MATLAB and ARDUINO.

2. ARDUINO CODE:-

TRANSMITTER SECTION

On the transmitter side we are using an arduino for two purposes:

1. To accept the serial data sent through MATLAB using the serial port.
2. To send data to the data pin of the RF transmitter

The Baudrate for serial communication with Laptop used here is 115200 and the incoming serial data is stored in the variable matlabdata. The serial.available() function gives the number of bytes available for reading from the serial port so if there is data in receive buffer then data available is read and stored in matlab data variable .

The RF send function is used to send the byte by adding framing bits and data bits corresponding to the input byte.

Here we send character '2' if matlabdata is '1' and '3 ' if matlabdata is '2'.

❖ RFSEND function

The function first transmits start bits to synchronize with the receiver and then performs data transmission.

The bitread function reads bits of the input number and transmits 1's and 0's depending on what is read.

Here we have used Manchester coding that is '1' is represented by HIGH for 500 microseconds and LOW for 500 microseconds.

'0' is represented by LOW for 500 microseconds and HIGH for 500 microseconds.

❖ available()

Description:

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).

Syntax**:** Serial.available()

❖ read()

Description: Reads incoming serial data.

Returns*:* The first byte of incoming serial data available (or -1 if no data is available) – *int*

❖ byte

Description: A byte stores an 8-bit unsigned number, from 0 to 255.

❖ attachInterrupt()

Description: It attaches an interrupt to the pins 2, 3 and an interrupt service routine is performed when an event ('Rising', 'Falling') occurs in these pins.

Digital Pins with Interrupts

Board	Digital Pins Usable For Interrupts
Uno, Nano, Mini, other 328-based	2, 3

Generally, an ISR should be as short and fast as possible. We cannot use delay function inside an ISR since delay() requires interrupts to work. However, delayMicroseconds() does not use any counter, so it will work as normal. Typically global variables are used to pass data between an ISR and the main program. To make sure variables shared between an ISR and the main program are updated correctly, declare them as volatile.

Syntax *:* attachInterrupt(digitalPinToInterrupt(pin), ISR,mode);

Parameters:

interrupt: the number of interrupt(int).

pin: the pin number

ISR: the ISR to call when interrupt occurs; this function must take no parameters and return nothing. This function sometimes refers to as an interrupt service routine.

mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

- **LOW** to trigger the interrupt whenever the pin is low,
- **CHANGE** to trigger the interrupt whenever the pin changes the values
- **RISING** to trigger when the pin goes from low to high.
- **FALLING** for when the pin goes from high to low.

❖ detachInterrupt()

Description: Turns off the given interrupt.

Syntax: detachInterrupt(interrupt)

Parameters interrupt: the number of the interrupt to disable.

❖ bitWrite()

Description: Writes a bit of a numeric variable.
Syntax: bitWrite(x, n, b)

Parameters

1. x: the numeric variable to which to write
2. n: which bit of the number to write, starting at 0 for the least-significant (rightmost) bit
3. b: the value to write to the bit (0 or 1)

❖ bitRead()

Description: Reads a bit of a number.
Syntax: bitRead(x, n)

Parameters:

x: the number from which to read

n: which bit to read, starting at 0 for the least-significant (rightmost) bit

Returns: the value of the bit (0 or 1).

Transmitter Section:-

```
int dataPin=2;

int ledPin=7;

int matlabData;

void setup()

{
```

```
  pinMode(dataPin,OUTPUT);
  pinMode(ledPin,OUTPUT);
  Serial.begin(115200);
}
void loop()
{
   if(Serial.available()>0) // if there is data to read
    {
     matlabData=Serial.read(); // read data
     if(matlabData==1)
  {
         rf_send('2');
        } // turn light on
    else if(matlabData==2)
  {
         rf_send('3');
    }
}
}
void rf_send(byte input){
  int i;
  for(i=0; i<20; i++){
  digitalWrite(2, HIGH);
  delayMicroseconds(500);
  digitalWrite(2, LOW);
  delayMicroseconds(500);
}
  digitalWrite(2, HIGH);
```

```
  delayMicroseconds(3000);
  digitalWrite(2, LOW);
  delayMicroseconds(500);
  for(i=0; i<8; i++){
  if(bitRead(input,i)==1)
  digitalWrite(2, HIGH);
  else
  digitalWrite(2, LOW);
  delayMicroseconds(500);
  if(bitRead(input,i)==1)
  digitalWrite(2, LOW);
  else
  digitalWrite(2, HIGH);
  delayMicroseconds(500);
}
}
```

Receiver Side

On the receiver side the arduino uses attachinterrupt() to keep on checking the pin 2 for the event: “RISING” edge of the input signal and then runs the data_incoming() function if event happens. Inside the ISR, code inside the first loop runs to synchronize the receiver with the transmitter and then enters the while loop. Here the bit received is stored in data_in variable. This variable now has the character which was transmitted and conditions are applied on it to control the light.

Receiver Section:-

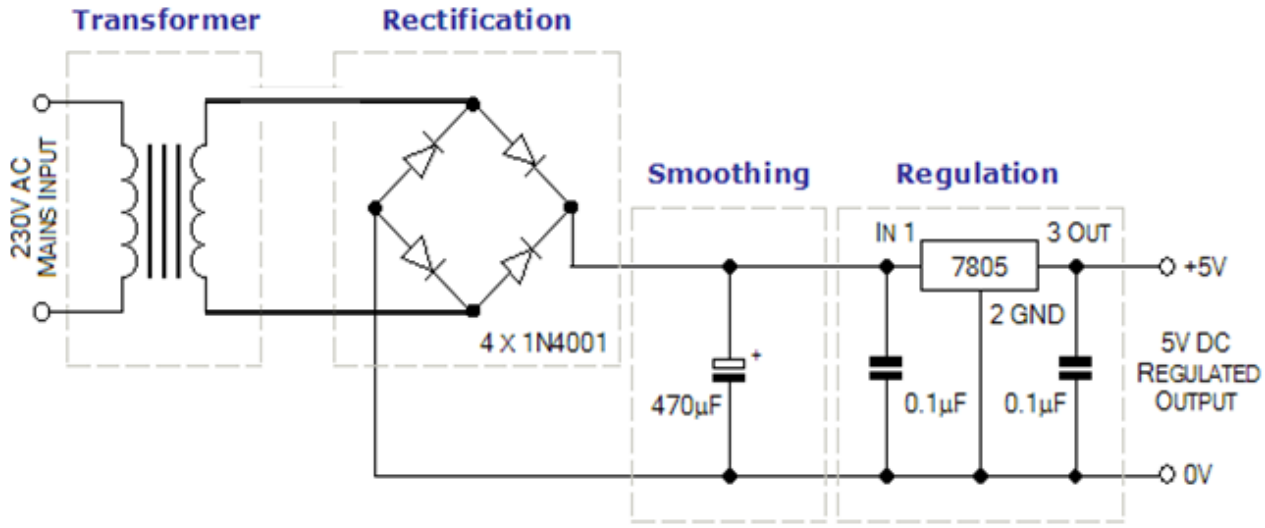
```
int i, good, k;
byte data_in;
char mishra;
void setup(){
```

```
  attachInterrupt(1,data_incoming,RISING);
  pinMode(3, INPUT);
  pinMode(7, OUTPUT);
  Serial.begin(115200);
}//setup
void loop(){
  if(mishra=='1')
digitalWrite(7,HIGH);
else
digitalWrite(7,LOW);
}//loop
void data_incoming(){
    for(i=0; i<100; i++){
      delayMicroseconds(20);
      good=1;
      if(digitalRead(3)==LOW){
      good=0;
      i=100;
      }
    }//for
    if(good==1){
    detachInterrupt(1);
    while(1){
      if(digitalRead(3)==LOW){
        delayMicroseconds(750);
        for(i=0; i<8; i++){
          if(digitalRead(3)==HIGH)
          bitWrite(data_in, i, 1);
```

```
          else
          bitWrite(data_in, i, 0);
          delayMicroseconds(1000);
        }//for
        Serial.print(char(data_in));
        if(char(data_in)=='2')
        mishra='1';
        else if(char(data_in)=='3')
        mishra='0';
       break;//secondtwhile
      }//low kickoff
    }//second while
    }//good check
  attachInterrupt(1,data_incoming,RISING);
}//routine
```

POWER SUPPLY DESIGN

In order to drive our receiver section, we have designed a power supply. The circuit diagram of the power supply is as shown below:



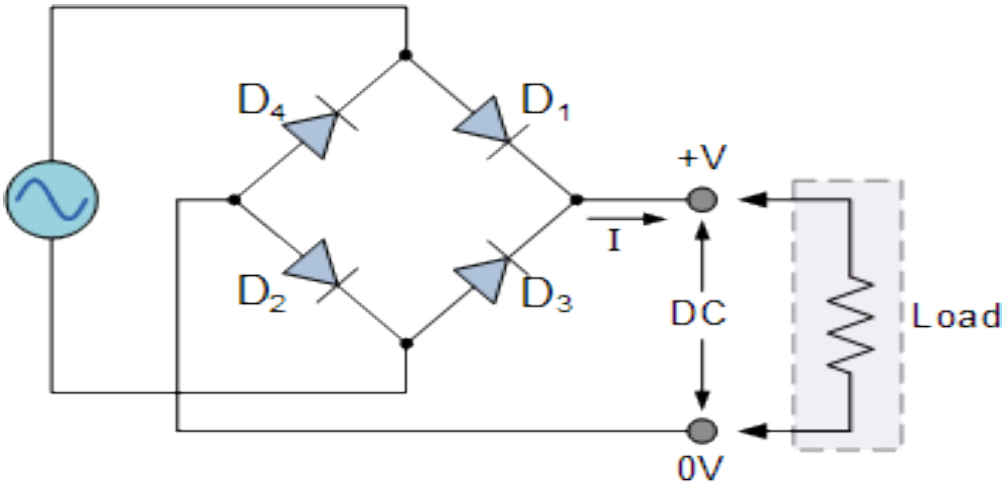
Transformer Action

Here we have used a Step-Down Transformer which is converting 230V AC supply into 12V AC. Now this 12V AC will be supplied to our bridge rectifier circuit.

Full Wave Bridge Rectifier

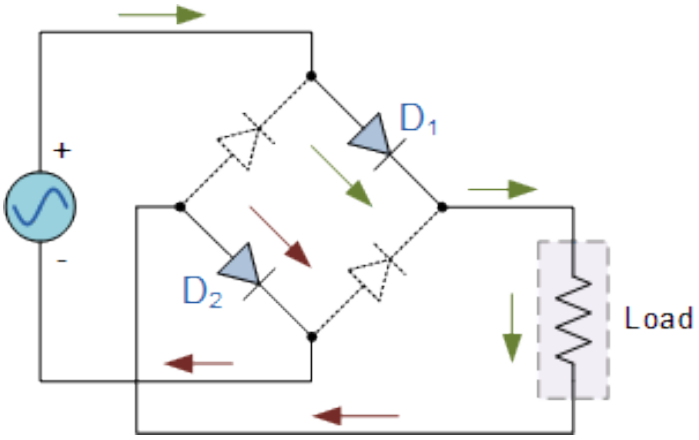
This type of single phase rectifier uses four individual rectifying diodes connected in a closed loop "bridge" configuration to produce the desired output. The main advantage of this bridge circuit is that it does not require a special centre tapped transformer, thereby reducing its size and cost. The single secondary winding is connected to one side of the diode bridge network and the load to the other side as shown below.

The Diode Bridge Rectifier:



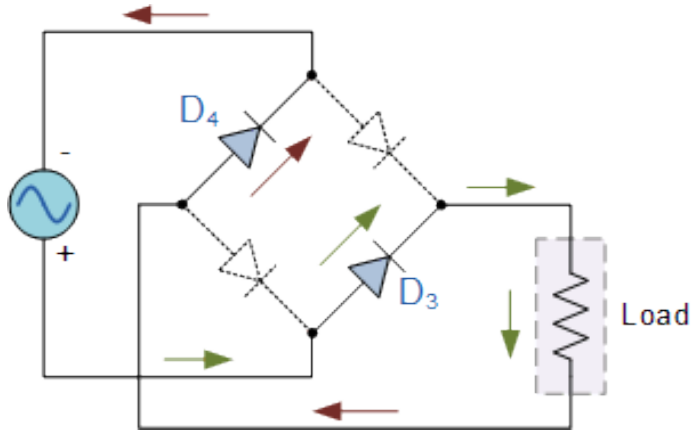
The four diodes labelled D1 to D4 are arranged in "series pairs" with only two diodes conducting current during each half cycle. During the positive half cycle of the supply, diodes D1 and D2 conduct in series while diodes D3 and D4 are reverse biased and the current flows through the load as shown below.

The Positive Half-cycle:



During the negative half cycle of the supply, diodes D3 and D4 conduct in series, but diodes D1 and D2 switch "OFF" as they are now reverse biased. The current flowing through the load is the same direction as before.

The Negative Half-cycle:



As the current flowing through the load is unidirectional, so the voltage developed across the load is also unidirectional the same as for the previous two diode full-wave rectifier, therefore the average DC voltage across the load is 0.637Vmax.

Typical Bridge Rectifier

However in reality, during each half cycle the current flows through two diodes instead of just one so the amplitude of the output voltage is two voltage drops (2 x 0.7 = 1.4V) less than the input VMAX amplitude. The ripple frequency is now twice the supply frequency (e.g. 100Hz for a 50Hz supply or 120Hz for a 60Hz supply.)

Although we can use four individual power diodes to make a full wave bridge rectifier, pre-made bridge rectifier components are available "off-the-shelf" in a range of different voltage and current sizes that can be soldered directly into a PCB circuit board or be connected by spade connectors.

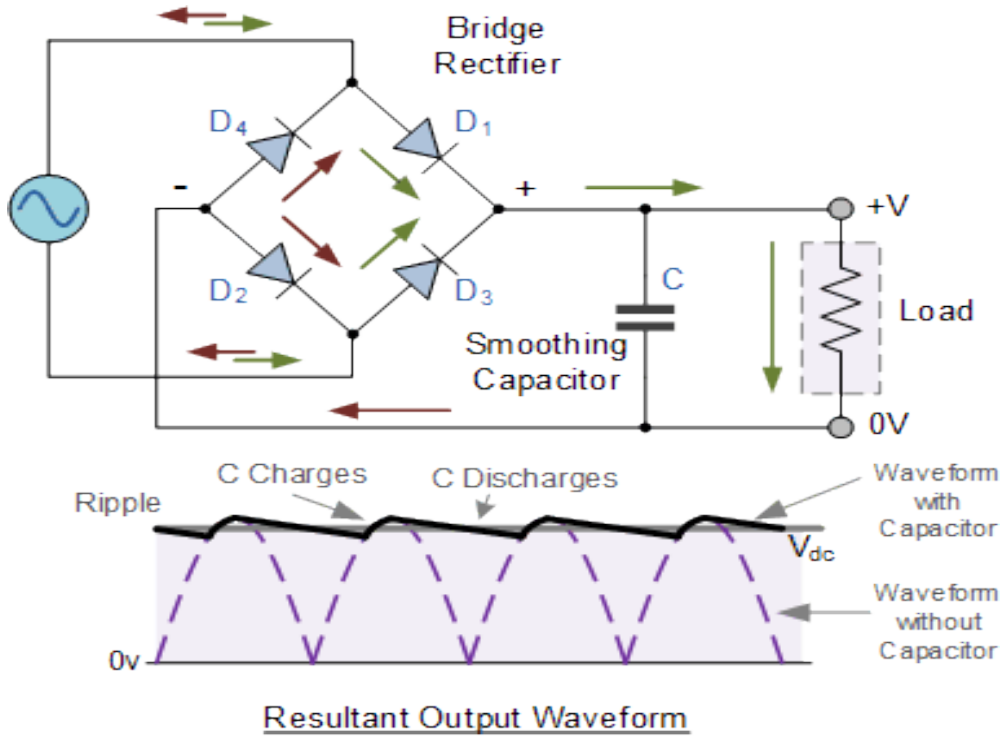
The image to the right shows a typical single phase bridge rectifier with one corner cut off. This cut-off corner indicates that the terminal nearest to the corner is the positive or +ve output terminal or lead with the opposite (diagonal) lead being the negative or -ve output lead. The other two connecting leads are for the input alternating voltage from a transformer secondary winding.

The Smoothing Capacitor

The single phase half-wave rectifier produces an output wave every half cycle and that it was not practical to use this type of circuit to produce a steady DC supply. The full-wave bridge rectifier however, gives us a greater mean DC value (0.637 Vmax) with less superimposed ripple while the output waveform is twice that of the frequency of the input supply frequency. We can therefore increase its average DC output level even higher by

connecting a suitable smoothing capacitor across the output of the bridge circuit as shown below.

Full-wave Rectifier with Smoothing Capacitor



Resultant Output Waveform

The smoothing capacitor converts the full-wave rippled output of the rectifier into a smooth DC output voltage. Generally, for DC power supply circuits the smoothing capacitor is an Aluminium Electrolytic type that has a capacitance value of 100uF or more with repeated DC voltage pulses from the rectifier charging up the capacitor to peak voltage.

However, there are two important parameters to consider when choosing a suitable smoothing capacitor and these are its working voltage, which must be higher than the no-load output value of the rectifier and its capacitance value, which determines the amount of ripple that will appear superimposed on top of the DC voltage.

Too low a capacitance value and the capacitor has little effect on the output waveform. But if the smoothing capacitor is sufficiently large enough (parallel capacitors can be used) and the load current is not too large, the output voltage will be almost as smooth as pure DC. As a general rule of thumb, we are looking to have a ripple voltage of less than 100mV peak to peak.

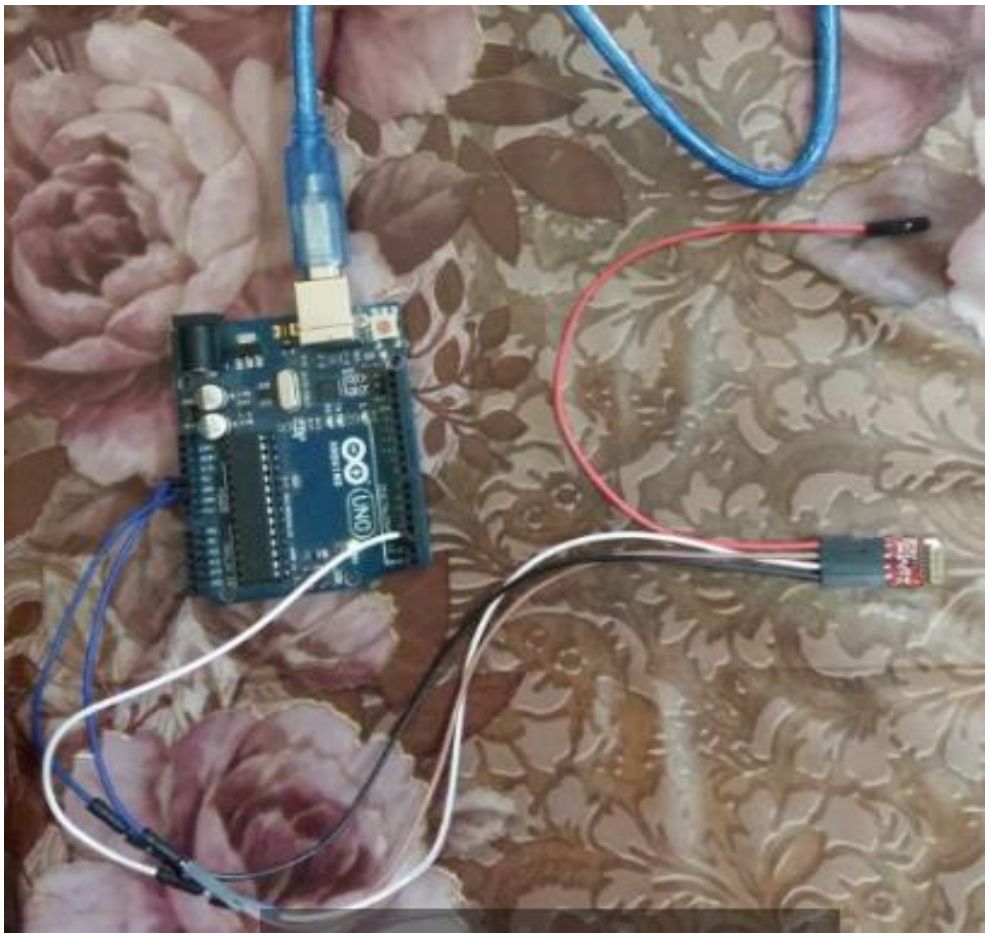
Voltage Regulator (IC 7805)

A voltage regulator is a device that maintains a relatively constant output voltage even though its input voltage may be highly variable. There are a variety of specific types of voltage regulators based on the particular method they use to control the voltage in a circuit. In general, a voltage regulator functions by comparing its output voltage to a fixed reference and minimizing this difference with a negative feedback loop.

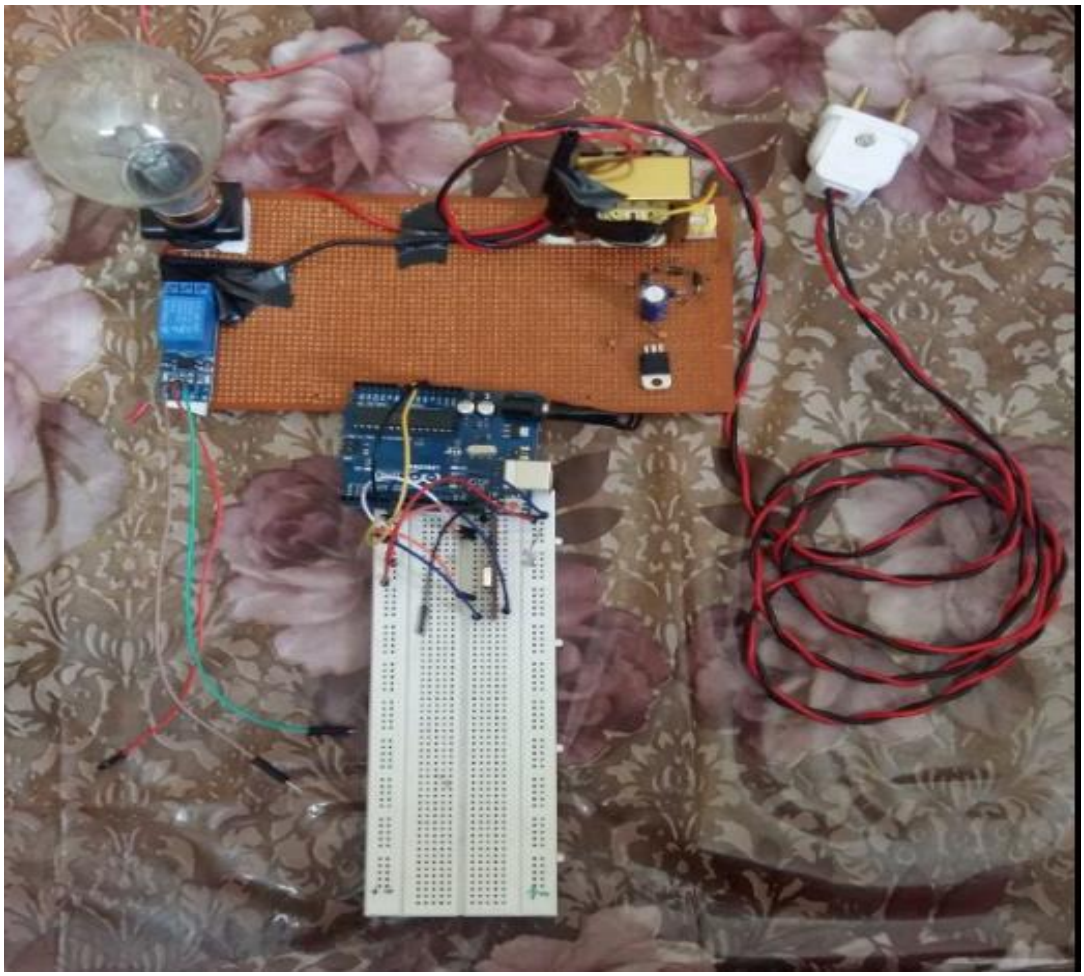
IC 7805 is a 5V Voltage Regulator that restricts the voltage output to 5V and draws 5V regulated power supply. It comes with provision to add heatsink. The maximum value for input to the voltage regulator is 35V. It can provide a constant steady voltage flow of 5V for higher voltage input till the threshold limit of 35V. If the voltage is near to 7.5V then it does not produce any heat and hence no need for heatsink. If the voltage input is more, then excess electricity is liberated as heat from 7805.

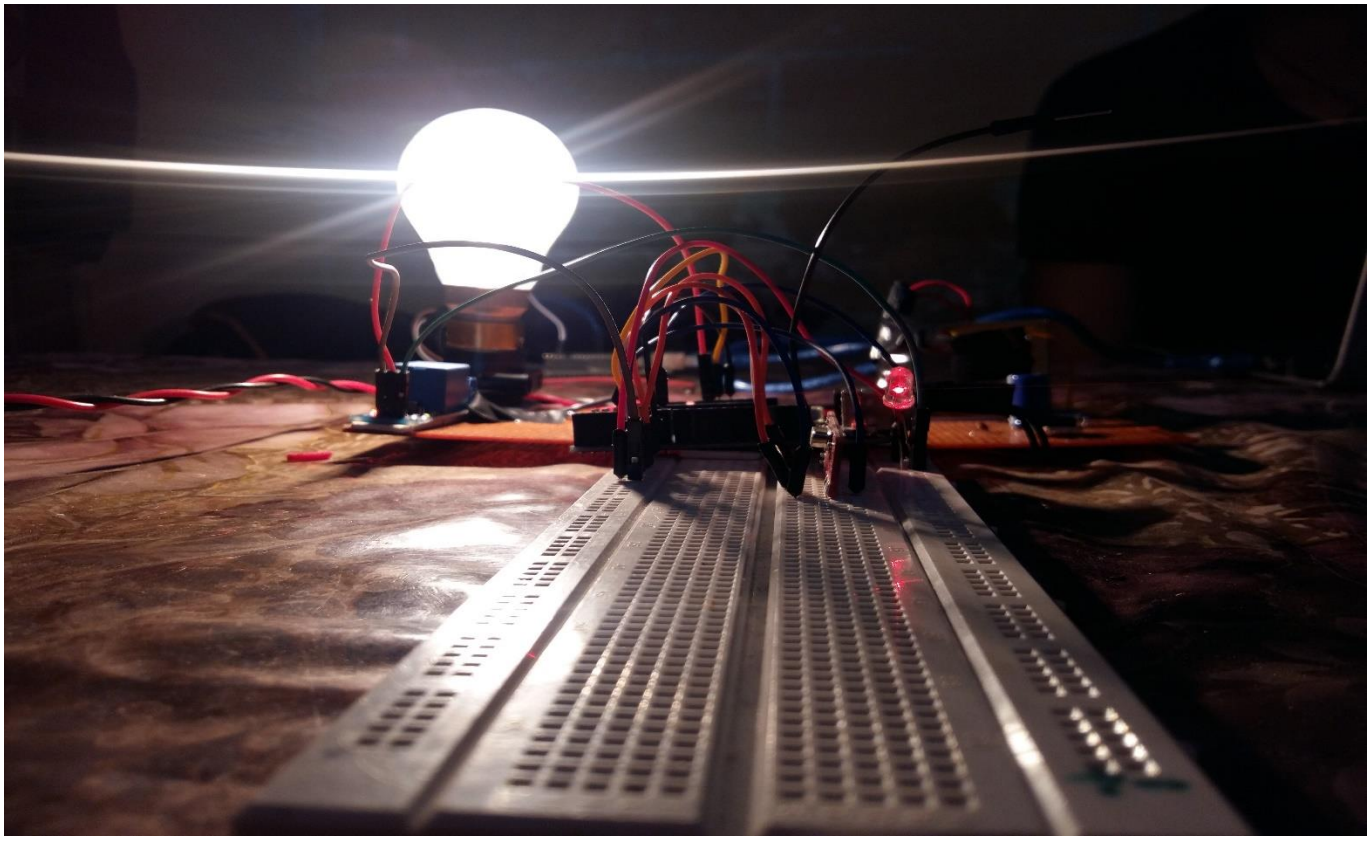
PHOTOS OF WORKING MODEL

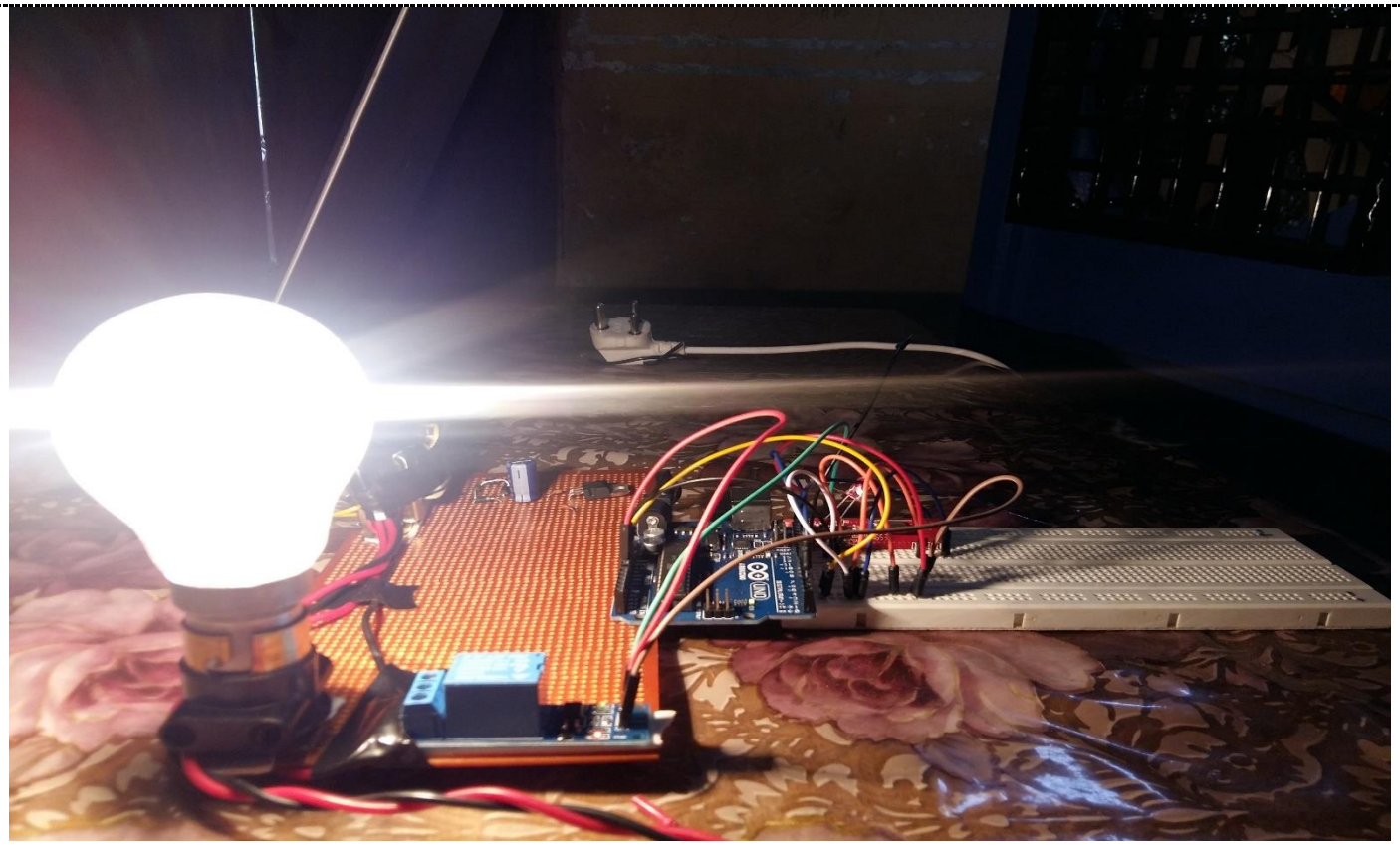
Transmitter Section:



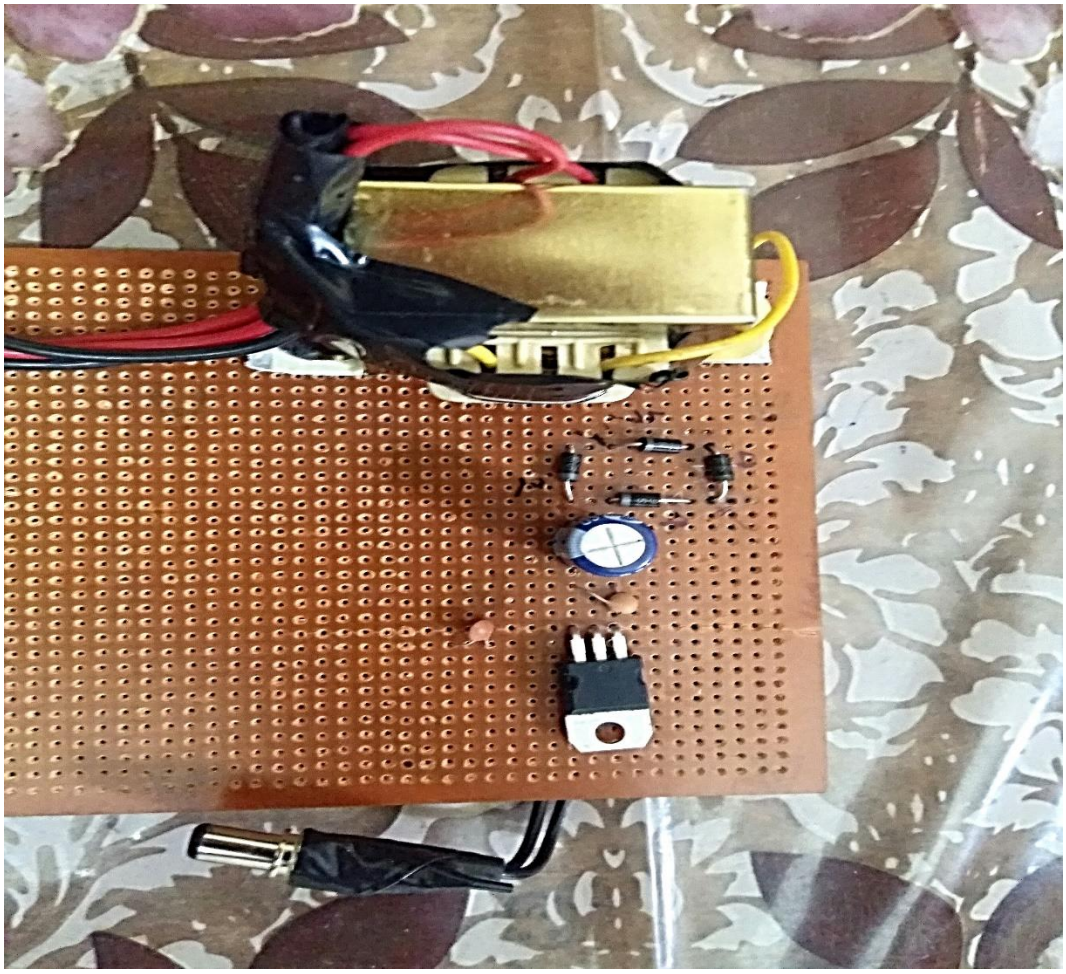
Receiving Section:







Power Supply:



APPLICATIONS

1. In car-systems, simple voice commands may be used to initiate phone calls, select radio stations or play music from a compatible smartphone, MP3 player or music-loaded flash drive. Voice recognition capabilities vary between car make and model. Recent car models offer natural-language speech recognition in place of a fixed set of commands, allowing the driver to use full sentences and common phrases. With such systems there is no need for the user to memorize a set of fixed command words.
2. Voice biometrics is often paired with facial recognition to bolster a multi-factor system. In combination with face biometrics, voice recognition can act as an additional authentication factor. Because of this specific combination of modalities is software based, it is often seen built in to banking apps.
3. In health care sector, speech recognition can be implemented in front-end or back-end of the medical documentation process. Voice recognition in hospitals is used for allowing authenticated persons inside the Intensive Care Unit. It will prevent any unauthorized person for any theft.
4. Speech recognition is also used in military high-performance fighter aircraft. Speech recognizers have been operated successfully in fighter aircraft, with applications including: setting radio frequencies, commanding an autopilot system, setting steer-point coordinates and weapons release parameters, and controlling flight display.
5. Training air traffic controllers(ATC) represents an excellent application for speech recognition systems. Speech recognition and synthesis techniques offer the potential to eliminate the need for a person to act as pseudo-pilot, thus reducing training and support personnel.
6. Leading software vendors like Google, Microsoft, Apple uses its own voice operated Voice Assistant like Google Now, Siri, Cortana.
   The improvement of mobile processors speed made feasible the speech-enabled Symbian and Windows Mobile smartphones.
7. Voice recognition can be used in education and daily life for learning a second language. It can teach proper pronunciation, in addition to helping a person develop fluency with their speaking skills.
8. Students who are blind or have very low vision can benefit from using the technology to convey words and then hear the computer recite them, as well as use a computer by commanding with their voice, instead of having to look at the screen and keyboard.
9. People with disabilities can benefit from speech recognition programs. Speech recognition software is used to automatically generate a closed-captioning of conversations such as discussions in conference rooms, classroom lectures. Speech recognition is also very useful for people who have difficulty using their hands.
10. Voice recognition can be used to reduce typing efforts. It will automatically convert speech to text and produce the required output.

CONCLUSIONS

For general conclusions, our designed system is working well but in the presence of noise there are chances of errors. If the reference signal is the same word as the target signal, so using this reference signal to model the target will have less errors.

So in order to improve the designed systems to make it work better, the further tasks are to enhance the systems' noise immunity and to find the common characteristics of the speech for the different people. Otherwise, to design some analog and digital filters for processing the input signals can reduce the effects of the input noise and to establish the large data base of the speech signals for different words.

REFERENCES

1. Joseph Picone, "Signal Modeling Techniques in Speech", Systems and Information Sciences Laboratory, Tsukuba Research and Development Center, Tsukuba, Japan.
2. John G.Proakis, Dimitris G.Manolakis, Digital Signal Processing, Principles, Algorithms, and Applications, 4th edition, Pearson Education inc., Upper Saddle River.
3. www.mathworks.com
4. www.arduino.cc