# Image result for lpu logo

# **Home Automation using PIC**

# [**IR Remote Controlled Home Automation using PIC Microcontroller**](https://circuitdigest.com/microcontroller-projects/ir-remote-controlled-home-automation-using-pic)

In this project, we are going to use a PIC microcontroller to remotely control few AC loads by just using an IR remote. At the end of this project you will be able to toggle (ON/OFF) any AC load using an ordinary Remote from the comfort of your Chair/Bed. To make this project more interesting we have also enabled a feature to control the speed of the fan with the help of Triac. All these can be done with simple clicks on your IR remote. You can use any of your TV/DVD/MP3 remote for this project. The different IR signals from the remote are received by the microcontroller which then controls the respective relays via a relay driver circuit. These relays are used to connect and disconnect the AC Loads (Lights/Fan).

### **Working Explanation:**

The working of this project is fairly simple to understand. When a button is pressed on the**IR Remote** it sends a sequence of code in form of encoded pulses using 38 Khz modulating frequency. These pulses are received by the **TSOP1738** sensor and then read by the Controller. The Controller then decodes the received train of the pulses into a hex value and compares it with the predefined hex values in our program.

If any match occurs then the controller performs a relative operation by triggering the respective Relay/Triac and the corresponding result is also indicated by on-board LEDs. Here in this project, we have used 4 bulbs (small bulbs) of different colours as lighting loads and another bulb (bigger bulb) is considered to be a fan for demonstration purpose.

We have selected key 1 to toggle the relay1, 2 to toggle the relay2, 3 to toggle the relay3, 4 to toggle the relay4, and Vol+ to increase fan speed and Vol- to decrease speed of the fan.

Note: Here we have used 100watt bulb instead of a fan.

There are many types of IR Remotes available for different devices, but most of them work around 38 KHz Frequency. Here in this project, we control home appliances using IR TV remote and for detecting the IR signals, we use a TSOP1738 IR Receiver. This TSOP1738 sensor can sense 38 Khz Frequency signal. The working of IR remote and the TSOP1738 is covered in detail in this article: IR Transmitter and Receiver.

Our PIC microcontroller operates at +5V and the Relays operate at +12V, Hence we use a transformer to step down the 220V AC and rectify it using a full bridge rectifier. This rectified DC voltage is then regulated to +12V and +5V by using the regulator ICs 7812 and 7805 respectively.

To trigger the relay we make use of transistors like BC547 which can act as an electronic switch to turn ON/OFF the relays based on the signal from the PIC microcontroller. Further to control the speed of the fan we are using a TRIAC. TRIAC is a power semiconductor which is capable of controlling the output voltage; this capability is used to control the speed of the fan.

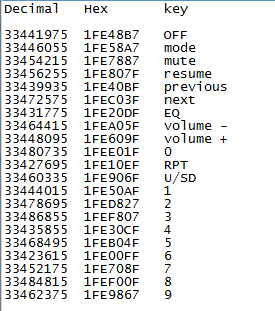
We have also used a Triac Driver to control the Triac using our PIC microcontroller. This driver is used to give a firing angle pulse to Triac, so that the output power can be controlled. Here we have used 6 level of speed control. When the level is 0 then the fan will be off. When level will be 1 then speed will be 1/5th of full speed. When level will be 2 then speed will be 2/5th of full speed and respectively for others. The current level of the speed can be monitored using the on-board 7-segment display.

### **Components:**

1. PIC18F458
2. Relay
3. IR remote
4. IR Receiver

**Decoding the IR Remote:**

As said earlier you can use any kind remote for your project. But we have to know what kind of signal is generated for from that particular remote. For every individual key on the remote there will be an equivalent HEX value for that key. Using this HEX value we can distinguish between each key on our microcontroller side. So before we decide to use a remote we should know the HEX value for the keys preset in that particular remote. In this project, we have used a NEC remote. The HEX values for the keys on a NEC remote is given below.



As you can notice the HEX value has 7 characters out of which only the last two differs, hence we can consider only the last two digits to distinguish between each keys.

**SIMULATION:**

**PROGRAM:**

#include <xc.h>

#include<string.h>

#include<stdlib.h>

#include<config.h>

#define ir PORTBbits.RB2

#define relay1 PORTCbits.RC2

#define relay2 PORTCbits.RC3

#define rly1LED PORTBbits.RB3

#define rly2LED PORTBbits.RB4

int flag=0;

int cmd=0;

unsigned int dat[100];

int i=0;

char result[10];

int j=0;

void delay(int time)

{

for(int i=0;i<time;i++)

for(int j=0;j<800;j++);

}

void timer() // 10 -> 1us

{

T0CONbits.T0PS0=0;

T0CONbits.T0PS1=0;

T0CONbits.T0PS2=0;

T0CONbits.PSA=0; //Timer Clock Source is from Prescaler

T0CONbits.T0CS=0; //Prescaler gets clock from FCPU (5MHz)

T0CONbits.T08BIT=0; //16 BIT MODE

T0CONbits.TMR0IE=1; //Enable TIMER0 Interrupt

INTCONbits.PEIE=1; //Enable Peripheral Interrupt

INTCONbits.GIE=1; //Enable INTs globally

T0CONbits.TMR0ON=1; //Now start the timer!

}

void main(void)

{

ADCON1=0b00001111;

TRISBbits.TRISB1=0;

TRISBbits.TRISB2=1;

TRISBbits.TRISB3=0;

TRISBbits.TRISB4=0;

TRISC=0x00;

relay1=0;

relay2=0;

rly1LED=0;

rly2LED=0;

i=0;

ir=0;

timer();

INTCON2bits.INTEDG0 = 0; // Interrupt on falling edge

INTCONbits.INT0IE = 1; // Enable the INT0 external interrupt (RB0)

INTCONbits.INT0IF = 0; // Clears INT0 External Interrupt Flag bit

INTCONbits.PEIE=1; //Enable Peripheral Interrupt

INTCONbits.GIE=1; //Enable INTs globally

while(1)

{

while(ir == 1);

INTCONbits.INT0IE = 0;

while(ir == 0);

TMR0=0;

while(ir == 1);

i++;

dat[i]=TMR0;

if(dat[1] > 5000 && dat[1]<12000)

{

}

else

{

i=0;

INTCONbits.INT0IE = 1;

}

if(i>=33)

{

INTCONbits.GIE=0;

delay(50);

cmd=0;

for(j=26;j<34;j++)

{

if(dat[j]>1000 && dat[j]<2000)

cmd<<=1;

else if(dat[j]>3500 && dat[j]<4500)

{

cmd|=0x01;

cmd<<=1;

}

}

cmd>>=1;

if(cmd == 0xAF)

{

relay1=~relay1;

rly1LED=~rly1LED;

}

else if(cmd == 0x27)

{

relay2=~relay2;

rly2LED=~rly2LED;

}

i=0;

INTCONbits.INT0IE = 1;

INTCONbits.GIE=1;

}

}

}

void interrupt isr()

{

if(TMR0IF) //Check if it is TMR0 Overflow ISR

{

TMR0IF=0;

}

}