GESTURE BASED TV REMOTE CONTROL

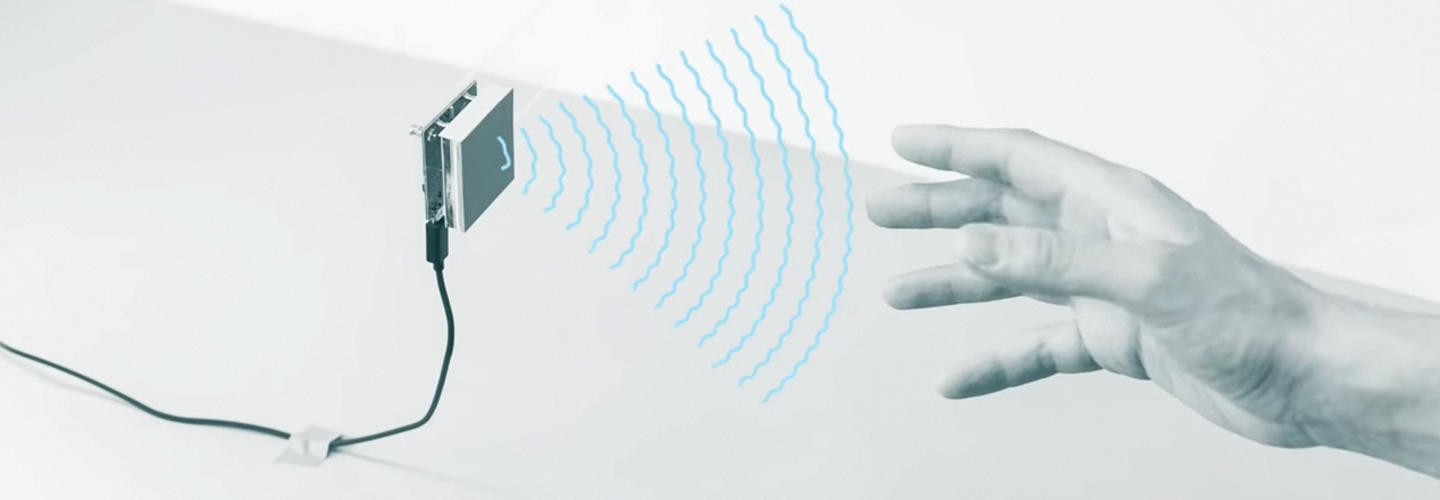
USING ULTRASONIC SENSOR AND IR TRANSMITTER

ABSTRACT BY

R. SHAM HIRUTHIK ECE III-YEAR

I­NTRODUCTION:

It is a project on GESTURE BASED TV REMOTE CONTROL using **Ultrasonic sensor** and **IR transmitter** interfacing with Arduino UNO. For television we are using HEX code but here we are using RGB LED where the all commands will be shown a serial monitor and with different colours on RGB. We can interface with our TV Where swiping over within the specified range in code shows different colours on RGB and different operations if you are using the TV.



HARDWARE REQUIRED:

* [Arduino uno with USB cable – 1](http://rees52.com/arduino-boards/4-aa001)
* [Ultrasonic sensor HC SR04 – 1](https://rees52.com/arduino-sensors/37-ultrasonic-sensor-module-hc-sr-04-sr014?search_query=ultrasonic+sensor&results=351)
* [Breadboard 400 point – 1](http://rees52.com/circuit-board/171-400-point-solderless-breadboard-ck010?search_query=BREADBOARD&results=220)
* [RGB Led – 1](https://rees52.com/led/1164-10x-5-mm-co-mmon-anode-4-pin-tri-color-diffused-light-lamp-led-diode-rgb-rs463?search_query=rgb&results=31)
* [IR led – 1](https://rees52.com/arduino-sensors/1228-ir-transmitter-led-for-lsa08-line-following-sensor-bar-rs056?search_query=ir+led&results=348)
* [Single stand wire 2m - 1](http://rees52.com/connectors/809-20-meter-multistand-wires-for-diy-electronics-projects-rc045)
* [Resistor 220 ohm – 3](https://rees52.com/home/1366-220k-ohm-carbon-film-resistors-25-watt-tolerance-5-5pcs-rs613?search_query=resistor+220+ohm&results=305)
* [Jumper wires male to female– 40 Pieces](http://rees52.com/connectors/153-jumper-wire-male-to-female-rk010)
* [Jumper wires male to male– 40 Pieces](http://rees52.com/connectors/110-jumper-wire-male-to-male-connector-single-pair-ec003)

SOFTWARE REQUIRED:

* Arduino IDE 1.8.5

SPECIFICATIONS:

* ULTRASONIC SENSOR:
  + - Operating Voltage
    - Static current
    - Induction Angle: 15°
    - Induction Angle: 15°
    - Induction Angle: 15°
* RGB LED:
* 4 pins: GND, R, G and B
* Operating Voltage: RGB LED (common cathode)

RED: 2.5V

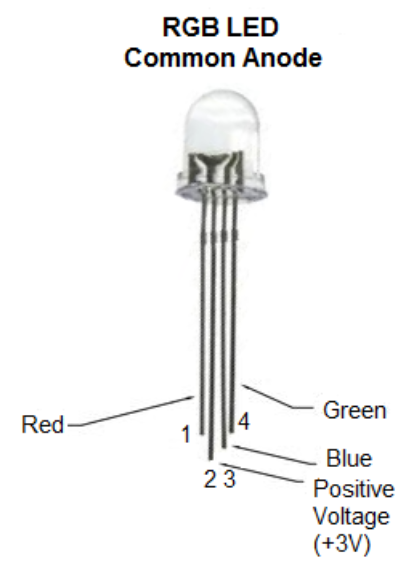
BLUE: 3.5V

GREEN: 3.5V

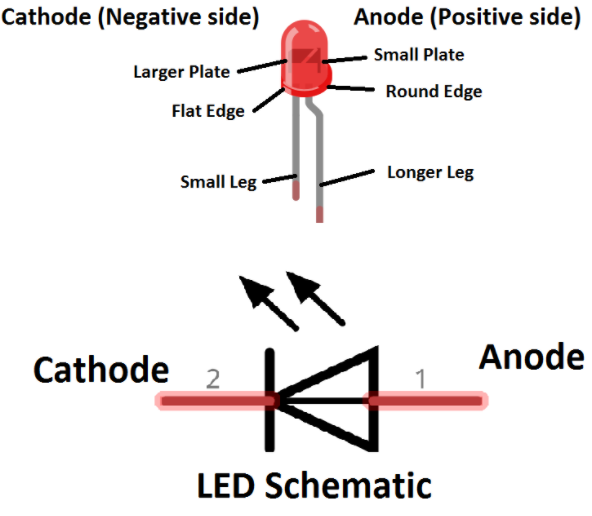
* IR LED:
* VOLTAGE = 1.8V
* Power = 170 Mw
* Viewing Angle = 60
* Current = 100mA

PIN DESCRIPTION:

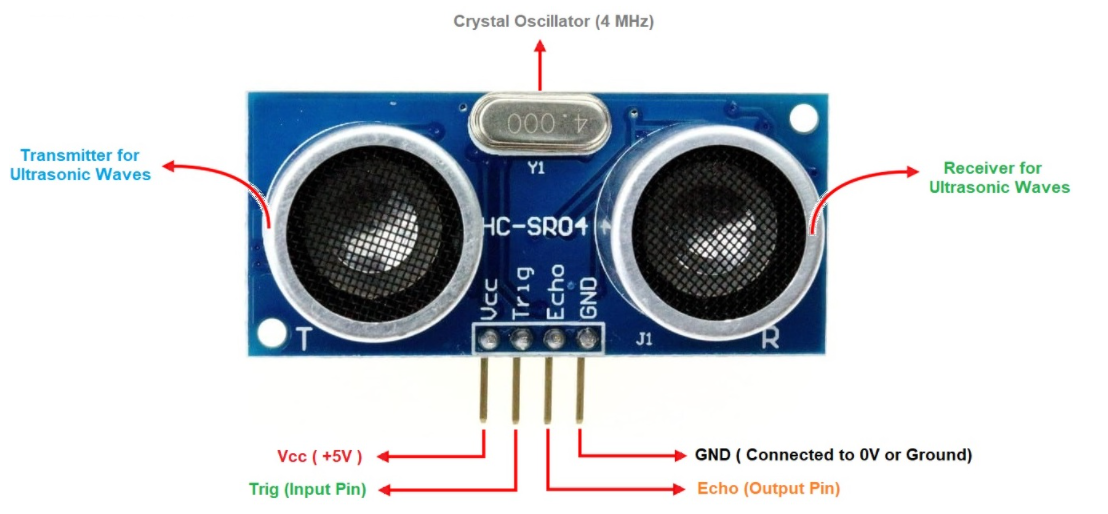
* RGB LED:



* LED:



* ULTRASONIC SENSOR:



**VCC -** Connects to 5v of positive voltage for power.

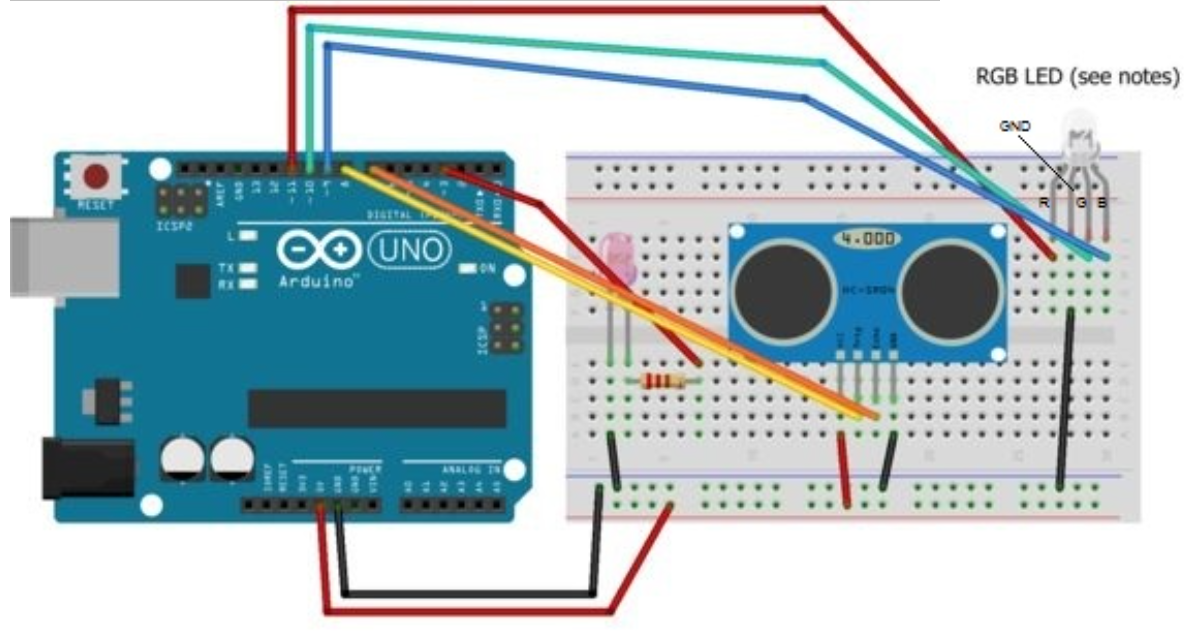
**TRIG-** A pulse is sent here for the sensor to go into the ranging mode for object

Detection.

**ECHO-** The Echo sends a signal back if an object has been detected or not. If a signal returned, an object has been detected, If not no object has been detected.

**GND-** Completes the electrical pathway of the power.

CIRCUIT DESCRIPTION:



* Connect the IR LED to pin 3 via an appropriate resistor, and then connect the cathode to GND.
* Connect the ultrasonic sensor to 5v and GND where the pins specify. Connect the trigger pin to pin 8 and the echo pin to 7. You can configure these pins in the sketch.
* Connect the RGB LED to GND and pins 11 (red), 10 (green), 9 (blue). These pins are also configurable.

CODE:

*/\* Swipe Remote Control*

*This sketch uses an ultrasonic rangefinder to determine the user's gesture and outputs an IR signal to a sony TV based on the command given.*

*- High swipe (> 10in) = Channel Up*

*- Low swipe = Channel Down*

*- High hold (> 10in) = Volume Up*

*- Low hold = Volume Down*

*- Cover sensor (< 3in) = Turn On / Off*

*\*/*

#include <IRremote.h>          // Library for IR Remote

// Defines for control functions

#define CONTROL\_CH 1     // Channel change

#define CONTROL\_VOL 2   // Volume

#define CONTROL\_POW 3 // Power

#define CONTROL\_UP 1

#define CONTROL\_DOWN -1

#define DIST\_MAX 20        // Maximum distance in inches, anything above is ignored.

#define DIST\_DOWN 10    // Threshold for up/down commands. If higher, command is "up". If

                                              lower, "down".

#define DIST\_POW 3        // Threshold for power command, lower than = power on/off

// IR PIN

const int irPin = 3;             // this is defined in the library, this var is just a reminder. CHANGING

                                            THIS WILL NOT CHANGE PIN IN LIBRARY

// 2 Pin Ping Sensor

const int pingPin = 8;       // Digital Pin 8 for Trigger Pin of Ultrasonic Sensor

const int echoPin = 7;      // Digital Pin 7 for Echo Pin of Ultrasonic Sensor

// Confirmation LED Pins

const int led = 13;      // internal LED for up/down debugging

const int ledR = 11;   // Pin for Red

const int ledG = 10;  // Pin for Green

const int ledB = 9;    // Pin for Blue

// LED on timer

unsigned long timer;

// IR transmitter object

IRsend irsend;

// Power confirmation flag (needs two swipes to send signal)

boolean powerConfirmed = false;

*/\* The setup() function is called when a sketch starts. It is used to initialize variables, pin modes, start using libraries, etc. This function will only run once, after each power up or reset of the Arduino board. \*/*

void setup()

{

  // initialize serial communication and set pins

  Serial.begin(9600);                    // Baud Rate

  pinMode(led, OUTPUT);           // LED acts as Output Pin

  pinMode(ledR, OUTPUT);        // LED (Red) acts as Output Pin

  pinMode(ledG, OUTPUT);        // LED (Green) acts as Output Pin

  pinMode(ledB, OUTPUT);        // LED (Blue) acts as Output Pin

pinMode(pingPin, OUTPUT);   // Ping Pin acts as Output Pin

pinMode(echoPin, INPUT);      // Echo Pin acts as Input Pin

timer = millis();                         // Set timer to milliseconds

}

*/\* This Particular Function is used for Repeated Execution of the Circuit until Specified. \*/*

void loop()

{

  // Serial.println(millis());

  long duration, inches;

  int value;                            // Initialize variable

  // Check for a reading

  duration = doPing();

  // Timer to confirm actions (currently only power)

  if (timer && timer < (millis() - 5000) && (millis() > 5000))

  {

    Serial.println("timer reset");  // Print the Message

    timer = false;                        // Timer value

  }

   digitalWrite(led, LOW);       // LED is inactive

   setColor(0, 0, 0);                // OFF condition

  // convert the time into a distance

  inches = microsecondsToInches(duration);

// If less than max inches away, act

  if (inches < DIST\_MAX)

  {

 // Debug output

    Serial.print(inches);

    Serial.println("in");

 // If very close, it is a "power" signal

    if (inches < DIST\_POW)

    {

      Serial.println(timer);

      // on or off

      if (timer)

      {

        doIR(CONTROL\_POW, 0);

        timer = false;

        delay(2000); // don't want to be sending this more than once. 2 second delay

      }

      else

      {

        Serial.println("power flag set");   // Print the Message

        timer = millis();

        setColor(255,50,50);                  // Set variations of RGB

        delay(500);                                 // Wait for 500 ms

      }

    }

    else // is volume or channel

    {

      // Distance determines control direction

      value = handleDist(inches);

// wait half a second

      delay(300);

      // check again, has hand disappeared?

      if (microsecondsToInches(doPing()) > DIST\_MAX)

{

        doIR(CONTROL\_CH, value); // swipe

 }

      else

{

        // volume

        int d = 500; // first delay is longer for single volume change

        // repeat until hand is removed

        while (inches < DIST\_MAX)

        {

          value = handleDist(inches); // is up or down?

          doIR(CONTROL\_VOL, value); // fire off IR signal

          delay(d); // wait

          inches = microsecondsToInches(doPing()); // check for hand again

          d = 100;     // delays are shorter for quick multiple volume adjustment

        }

        delay(500);  // this stops accidental channel change after volume adjustment

      }

]=    }

  }

  delay(50);  // Short enough to detect all swipes.

}

*/\* If distance is within threshold, mark as 'up' and turn on corresponding LED.\*/*

int handleDist(int inches)

{

  if (inches > DIST\_DOWN)

  {

  digitalWrite(led, HIGH);     // LED is High

    return CONTROL\_UP;

  }

  else

  {

  digitalWrite(led, LOW);     // LED is Low

   return CONTROL\_DOWN;

  }

}

*/\**

*\* Fire off correct IR code*

*\*/*

void doIR(int control, int val)

{

  switch(control)

  {

  case CONTROL\_POW:

    // power

    Serial.println("power on / off 0xa90");      // Print the Message

    for (int i = 0; i < 3; i++)

    {

      setColor(255, 0, 0);       // Set variations of RGB

      irsend.sendSony(0xa90, 12); // Sony TV power code

      delay(40);

    }

break;

    case CONTROL\_CH:

    setColor(0, 255, 0);        // Set variations of RGB

    // output 'channel up / down' depending on val

   if (val == CONTROL\_UP)          // When both the values are equal

    {

      digitalWrite(led, HIGH);           // LED is High

      for (int i = 0; i < 3; i++)

      {

irsend.sendSony(0x90, 12);

      delay(40);

      }

Serial.println("channel up 0xD00A");      // Print the Message

    }

    Else            // down

    {

      for (int i = 0; i < 3; i++)

      {

        irsend.sendSony(0x890, 12);

        delay(40);

      }

      Serial.println("channel down 0x3002");    // Print the Message

    }

    break;

    case CONTROL\_VOL:

    setColor(0, 0, 255);

    // output 'volume up / down' depending on val

    if (val == CONTROL\_UP)     // When both the values are equal

    {

      digitalWrite(led, HIGH);            // LED is High

 for (int i = 0; i < 3; i++)

      {

        irsend.sendSony(0x490, 12);

        delay(40);

      }

Serial.println("volume up 0x490");         // Print the Message

    }

    else     //down

    {

      for (int i = 0; i < 3; i++)

      {

        irsend.sendSony(0xC90, 12);

 delay(40);

 }

      Serial.println("volume down 0xC90");    // Print the Message

    }

    break;

  }

}

*/\*AnalogWrite uses pulse width modulation (PWM), turning a digital pin on and off very quickly*

*with different ratio between on and off, to create a fading effect. \*/*

void setColor(int red, int green, int blue)

{

  analogWrite(ledR, red);        // Red is active

  analogWrite(ledG, green);   // Green is active

  analogWrite(ledB, blue);      // Blue is active

}

long doPing()

{

  digitalWrite(pingPin, LOW);

  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);

delayMicroseconds(5);

  digitalWrite(pingPin, LOW);

  return pulseIn(echoPin, HIGH);

}

long microsecondsToInches(long microseconds)

{

  // According to Parallax's datasheet for the PING))), there are

  // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per

  // second).  This gives the distance travelled by the ping, outbound

  // and return, so we divide by 2 to get the distance of the obstacle.

  return microseconds / 74 / 2;

}

long microsecondsToCentimeters(long microseconds)

{

  // The speed of sound is 340 m/s or 29 microseconds per centimeter.

  // The ping travels out and back, so to find the distance of the

  // object we take half of the distance travelled.

  return microseconds / 29 / 2;

}

WORKING:

* POWER: Swiping over the sensor **within 3 inches will** cause the LED to flash purple. This is the confirmation that a 'turn on / off' signal is ready to be sent. **To prevent it accidentally turning the TV off I have made the sketch wait for a second swipe within 5 seconds of the first to confirm. At this point, the signal is sent to the TV to turn on or off**.
* CHANNEL: Swiping **within 10 inches of the sensor (but not within 3)** will cause the channel to **change down**. Swiping **between 10 and 20 inches** will cause the channel to **change up.**
* VOLUME: Holding your hand **within 10 inches of the sensor (but not within 3)** will cause the volume to **change down**. Holding **between 10 and 20 inches** will cause the volume to change up. The volume will continue to change (up or down) until your hand is removed.

APPLICATION:

Gesture recognition is used more or less exactly as you would imagine in smart TVs. It allows users to change the channel, adjust volume and brightness, control the picture quality and even view multiple screens and access social networking sites.

PROS:

* Makes interacting with TVs **easier**.
* **Reduces** the need for remote controls.
* Provide a **richer experience** to users.
* Provide **easy integrations** with other technologies such as face recognition and voice recognition.

CONS:

* The software maybe **inaccurate at sometimes**.
* Objects in the background may **disrupt the input**.
* Viability of usage In day to day application is **low**.
* When the HC-SR04 sensor doesn’t have obstacles in front or when the obstacles are soft, it **returns 0** or erroneous results.
* The HC-SR04 sensor is **not the most accurate ultrasonic sensor** on the market.
* One of the major issues plaguing the Global Gesture Recognition for Smart TV Market is the **lack of standardization in the actual gestures** used to control TVs.
* As you can see from the litany of CONS, the technology is **hardly perfect yet**.

DEVELOPMENT & RESOLVE:

* Using High accuracy Ultrasonic sensor.
* We can increase the accuracy of the sensor by giving an Visual inputs like **camera recognition of Gestures.**
* Currently, the Global Gesture Recognition Market for Smart TV has several leading vendors, resulting from the commercialization of gesture-enabled consumer electronics. Many pure-play companies are being acquired by large corporations for their technological **expertise in gesture recognition.** This will lead to major development GESTURE BASED TECHNOLOGY.

CONCLUSION:

So by here we’ve made a **GESTURE BASED TV REMOTE CONTROL** Using Ultrasonic Sensor and IR sensor.