Table of Contents

Objectives

The objectives of this project were to:

- Develop a system that can capture and interpret hand gestures as input commands for a computer.
- Use an Arduino board to read sensor data from an ultrasonic and convert it into specific commands for the computer.
- Explore the potential applications of hand gesture control in various fields such as gaming, accessibility, and human-computer interaction.
- To provide a novel way of interaction with computer which is more intuitive and natural, compare to traditional input devices like keyboard and mouse.

By achieving these objectives, the project aimed to demonstrate the feasibility and potential of using hand gestures as an input method for computers and other devices, and to pave the way for future research and development in this area.

Introduction

An embedded system is a combination of computer hardware and software designed for a specific function. Embedded systems may also function within a larger system. The systems can be programmable or have a fixed functionality. Industrial machines, consumer electronics, agricultural and processing industry devices, automobiles, medical equipment, cameras, digital watches, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

Touchscreens are now fairly standard in Windows 8 notebooks, but HP's new Envy 17 Leap Motion SE has integrated yet another type of input: touchless. The Envy has Leap Motion's gesture recognition technology built directly into the palmrest, allowing users to move seamlessly between the touchpad, touchscreen, and 3D gesture control.

This integration is powered by a new Leap Motion microsensor that's 70 percent smaller than the standalone peripheral. The new module is specifically designed to be embedded into various devices, which could allow the company to grow larger than it would as just a USB accessory manufacturer.

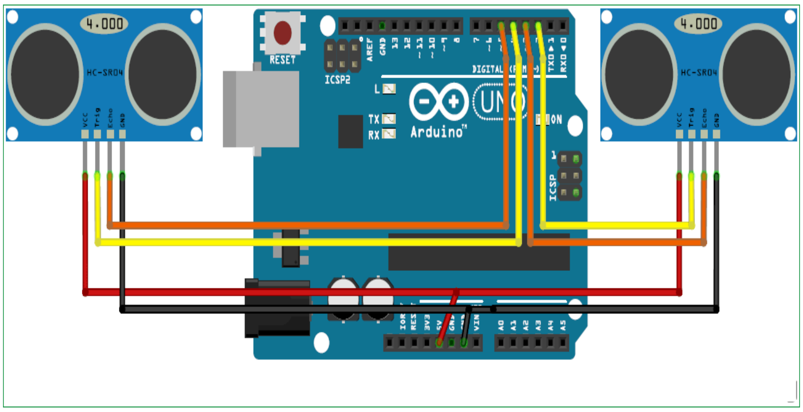
This technique is called Leap motion which enables us to control certain functions on our computer/Laptop by simply waving our hand in front of it. It is very cool and fun to do it, but these laptops are really priced very high. So in this project let us try building our own **Gesture control Laptop/Computer by combining the Power of Arduino and Python**.

We will use two **Ultrasonic sensors** to determine the position of our hand and **control a media player (VLC) based on the position**. I have used this for demonstration, but once you have understood the project, you can do anything by just changing few lines of code and control your favorite application in your favorite way.

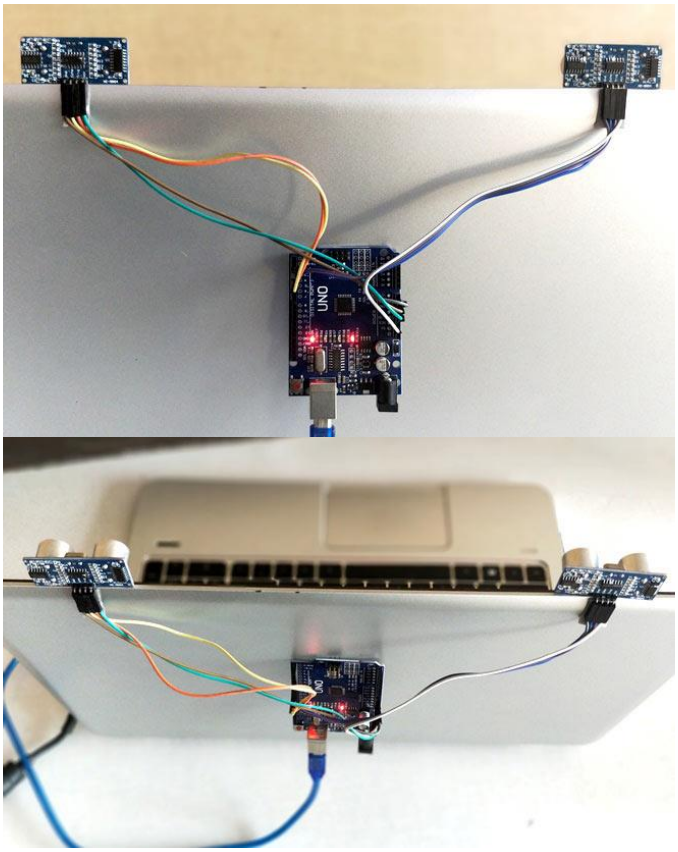
Concept behind the project:

The concept behind the project is very simple. We will place two Ultrasonic (US) sensors on top of our monitor and will read the distance between the monitor and our hand using Arduino, based on this value of distance we will perform certain actions. To perform actions on our computer we use Python **pyautogui** library. The commands from Arduino are sent to the computer through serial port (USB). This data will be then read by python which is running on the computer and based on the read data an action will be performed.

Circuit Diagram:

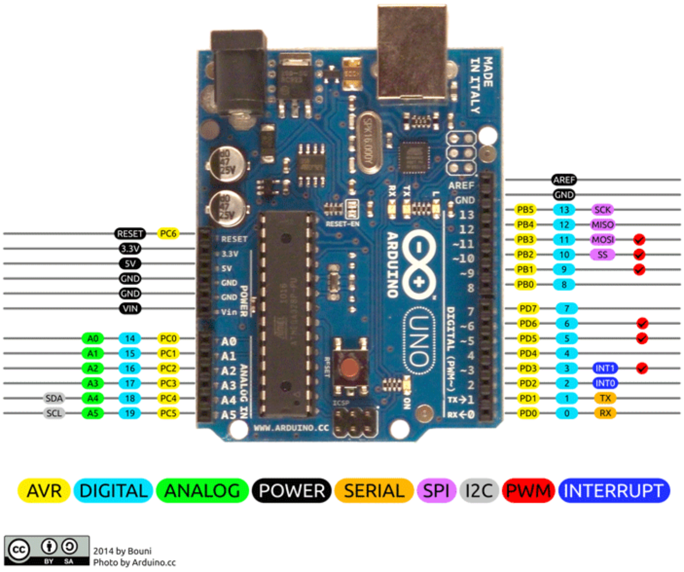


To **control the PC with Hand Gestures**, just connect the two Ultrasonic sensors with Arduino. We know US sensor work with 5V and hence they are powered by the on board Voltage regulator of Arduino. The Arduino can be connected to the PC/Laptop for powering the module and also for Serial communication. Once the connections are done place them on your monitor as shown below. I have used a double side tape to stick it on my monitor but you can use your own creativity. After securing it in a place we can proceed with the Programming.



Components

Arduino UNO



Arduino Uno is a popular microcontroller development board based on 8 bit ATmega328P microcontroller. Along with ATmega328P MCU IC, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller.

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

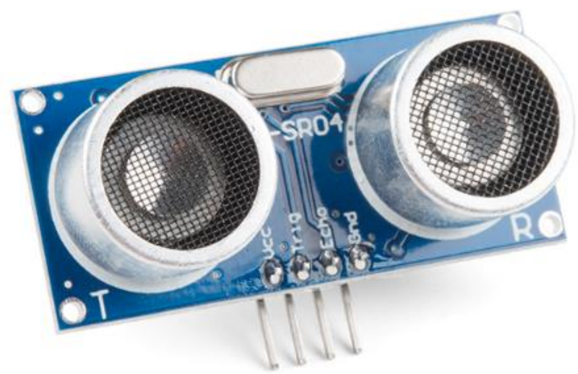
- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11**: These pins provide an 8-bit PWM output by using analogWrite() function.
- **SPI Pins 10 (SS),** 11 (MOSI), 12 (MISO) and 13 (SCK): These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e. 1024 different values. They measure from 0 to 5 volts but this limit can be increased by using AREF pin with analog Reference() function.
Analog pin 4 (SDA) and pin 5 (SCA) also used for TWI communication using Wire library.
Arduino Uno has a couple of other pins as explained below:

- **AREF:** Used to provide reference voltage for analog inputs with analogReference() function.
- **Reset Pin:** Making this pin LOW, resets the microcontroller.

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button.

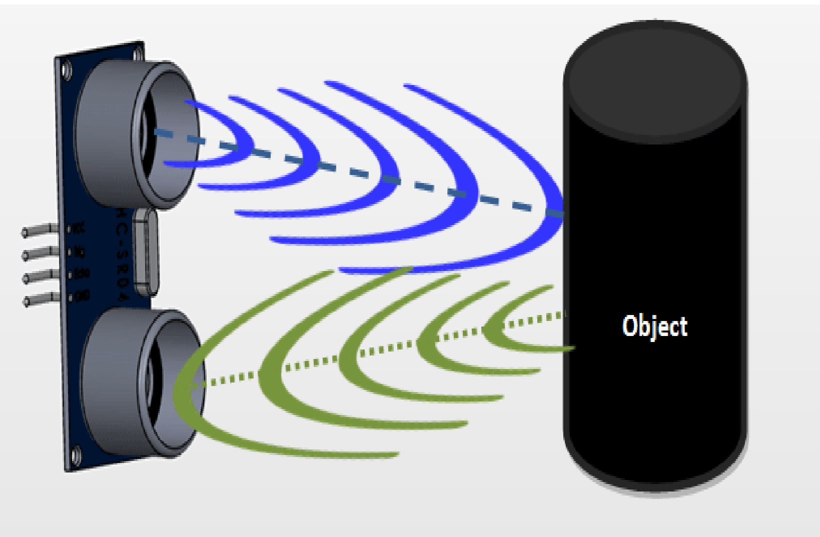
Ultrasonic Sensor



As shown above the **HC-SR04 Ultrasonic (US) sensor** is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below



Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

Programming your Arduino:

The Arduino should be programmed to read the distance of hand from the US sensor. The **complete program** is given at the end of this page; just below I have given the explanation for the program. If you are new to Ultrasonic sensor, just go through Arduino & Ultrasonic Sensor Based Distance Measurement.

By reading the value of distance we can arrive at certain actions to be controlled with gestures, for example in this program I have programmed **5 actions** as a demo.

Action 1: When both the hands are placed up before the sensor at a particular far distance then the video in VLC player should Play/Pause.

Action 2: When right hand is placed up before the sensor at a particular far distance then the video should Fast Forward one step.

Action 3: When left hand is placed up before the sensor at a particular far distance then the video should Rewind one step.

Action 4: When right hand is placed up before the sensor at a particular near distance and then if moved towards the sensor the video should fast forward and if moved away the video should Rewind.

Action 5: When left hand is placed up before the sensor at a particular near distance and then if moved towards the sensor the volume of video should increase and if moved away the volume should Decrease.

Let us see how the program is written to perform the above actions. So, like all programs we **start with defining the I/O pins** as shown below. The two US sensors are connected to Digital pins 2,3,4 and 5 and are powered by +5V pin. The trigger pins are output pin and Echo pins are input pins.

The Serial communication between Arduino and python takes places at a baud rate of 9600.

```
const int trigger1 = 2; //Trigger pin of 1st Sesnor

const int echo1 = 3; //Echo pin of 1st Sesnor

const int trigger2 = 4; //Trigger pin of 2nd Sesnor

const int echo2 = 5;//Echo pin of 2nd Sesnor

void setup() {

Serial.begin(9600);


pinMode(trigger1, OUTPUT);

pinMode(echo1, INPUT);

pinMode(trigger2, OUTPUT);

pinMode(echo2, INPUT);

}
```

We need to **calculate the distance between the Sensor and the hand** each time before concluding on any action. So we have to do it many times, which means this code should be used as a function. We have written a function named *calculate_distance()* which will return us the distance between the sensor and the hand.

```
/*###Function to calculate distance###*/

void calculate_distance(int trigger, int echo)

{

digitalWrite(trigger, LOW);
```

```
delayMicroseconds(2);
digitalWrite(trigger, HIGH);
delayMicroseconds(10);
digitalWrite(trigger, LOW);

time_taken = pulseIn(echo, HIGH);
dist= time_taken*0.034/2;
if (dist>50)
dist = 50;
}
```

Inside our main *loop* we **check for the value of distance and perform the actions** mentioned above. Before that we use two variables *distL* and *distR* which gets updated with current distance value.

```
calculate_distance(trigger1,echo1);
distL =dist; //get distance of left sensor

calculate_distance(trigger2,echo2);
distR =dist; //get distance of right sensor
```

Since we know the distance between both the sensors, we can now compare it with predefined values and arrive at certain actions. For example if both the hands are placed at a distance of 40 mc then we play/pause the video. Here the word **"Play/Pause"** will be sent out through serial port

```
if ((distL >40 && distR>40) && (distL <50 && distR<50)) //Detect both hands
{Serial.println("Play/Pause"); delay (500);}
```

If the Right hand alone is placed before the module then we fast forward the video by one step and if it is left hand we rewind by one step. Based on the action, here the word **"Rewind" or "Forward"** will be sent out through serial port

```
if ((distL >40 && distL<50) && (distR ==50)) //Detect Left Hand
{Serial.println("Rewind"); delay (500);}

if ((distR >40 && distR<50) && (distL ==50)) //Detect Right Hand
{Serial.println("Forward"); delay (500);}
```

Foe detailed control of volume and track we use a different methodology so as to prevent false triggers. To **control the volume** we have to place the left hand approx. At a distance of 15 cm

, then you can either move it towards the sensor to decrease the volume of move it away from the sensor to increase the volume. The code for the same is shown below. Based on the action, here the word “Vup” or “Vdown” will be sent out through serial port

```
//Lock Left - Control Mode
if (distL>=13 && distL<=17)
{
  delay(100); //Hand Hold Time
  calculate_distance(trigger1,echo1);
  distL =dist;
  if (distL>=13 && distL<=17)
  {
    Serial.println("Left Locked");
    while(distL<=40)
    {
      calculate_distance(trigger1,echo1);
      distL =dist;
      if (distL<10) //Hand pushed in
      {Serial.println ("Vup"); delay (300);}
      if (distL>20) //Hand pulled out
      {Serial.println ("Vdown"); delay (300);}
    }
  }
}
```

We can use the same method for the right side sensor also, to **control the track of the video**. That is if we move the right hand towards the sensor it will fast forward the movie and if you move it away from the sensor it will rewind the movie. Based on the action, here the word “Rewind” or “Forward” will be sent out through serial port

You can now read over the **complete code for this gesture controlled PC given at the end of the page** and try understating it as an whole and then copy it to your Arduino IDE.

Programming your Python:

The python program for this project is very simple. We just have to establish a serial communication with Arduino through the correct baud rate and then perform some basic keyboard actions. The first step with python would be to install the ***pyautogui*** module. Make sure you follow this step because the **program will not work without pyautogui module**.

Installing pyautogui module for windows:

Follow the below steps to install *pyautogui* for windows. If you are using other platforms the steps will also be more or less similar. Make sure your computer/Laptop is connected to internet and proceed with steps below

Step 1: Open Windows Command prompt and change the directory to the folder where you have installed python. By default the command should be

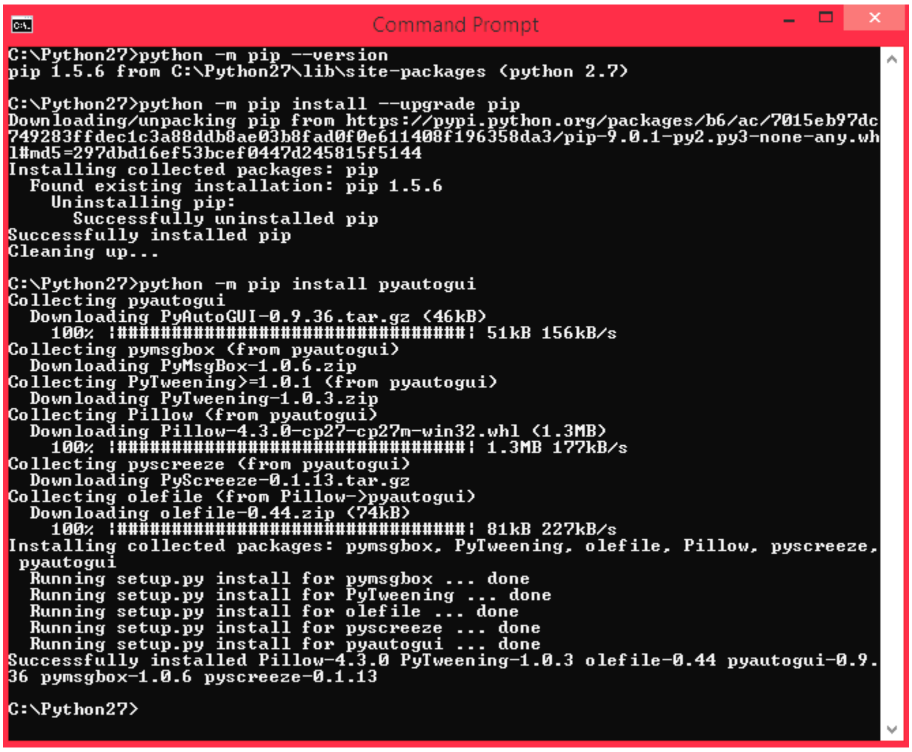
```
cd C:\Python27
```

Step 2: Inside your python directory use the command *python –m pip install –upgrade pip* to upgrade your pip. Pip is a tool in python which helps us to install python modules easily. Once this module is upgraded (as shown in picture below) proceed to next step.

```
python -m pip install -upgrade pip
```

Step 3: Use the command "*python –m pip install pyautogui*" to install the pyautogui module. Once the process is successful you should see a screen something similar to this below.

```
python -m pip install -upgrade pip
```



Now that the *pyautogui* module and *pyserial* module (installed in previous tutorial) is successful installed with the python, we can proceed with the python program. The **complete python code is given at the end of the tutorial** but the explanation for the same is as follows.

Let us import all the three required modules for this project. They are pyautogui, serial python and time.

```
import serial #Serial imported for Serial communication

import time #Required to use delay functions

import pyautogui
```

Next we **establish connection with the Arduino through COM port.** In my computer the Arduino is connected to COM 18. Use device manager to find to which COM port your Arduino is connected to and correct the following line accordingly.

```
ArduinoSerial = serial.Serial('com18',9600) #Create Serial port object call
ed arduinoSerialData

time.sleep(2) #wait for 2 seconds for the communication to get established
```

Inside the infinite *while* loop, we repeatedly listen to the COM port and **compare the key words with any pre-defied works** and make key board presses accordingly.

```
while 1:

    incoming = str (ArduinoSerial.readline()) #read the serial data and pri
nt it as line

    print incoming

    if 'Play/Pause' in incoming:

        pyautogui.typewrite(['space'], 0.2)

    if 'Rewind' in incoming:

        pyautogui.hotkey('ctrl', 'left')

    if 'Forward' in incoming:

        pyautogui.hotkey('ctrl', 'right')

    if 'Vup' in incoming:

        pyautogui.hotkey('ctrl', 'down')

    if 'Vdown' in incoming:

        pyautogui.hotkey('ctrl', 'up')
```

As you can see, to press a key we simply have to use the command *"pyautogui.typewrite(['space'], 0.2)"* which will press the key space for 0.2sec. If you need hot keys like ctrl+S then you can use the hot key command *"pyautogui.hotkey('ctrl', 's')"*.

I have used these combinations because they work on VLC media player you can tweak them in any way you like to create your own applications to **control anything in computer with gestures**.

Gesture Controlled Computer in Action:

Make the connections as defined above and upload the Arduino code on your Arduino board. Then use the python script below and launch the program on your laptop/computer.

Now you can play any movie on your computer using the VLC media player and **use your hand to control the movie** as shown in the **video given** below.

Hope you understood the project and enjoyed playing with it. This is just a demo and you can use your creativity to build a lot more cool gesture controlled stuff around this. Let me know if this was useful and what you will create using this in the comment section and I will be happy to know it.

Code

```
Arduino Code:

const int trigger1 = 2; //Trigger pin of 1st Sesnor

const int echo1 = 3; //Echo pin of 1st Sesnor

const int trigger2 = 4; //Trigger pin of 2nd Sesnor

const int echo2 = 5;//Echo pin of 2nd Sesnor

long time_taken;

int dist,distL,distR;


void setup() {

Serial.begin(9600);


pinMode(trigger1, OUTPUT);

pinMode(echo1, INPUT);

pinMode(trigger2, OUTPUT);

pinMode(echo2, INPUT);
```

```
}
/*###Function to calculate distance###*/
void calculate_distance(int trigger, int echo)
{
digitalWrite(trigger, LOW);
delayMicroseconds(2);
digitalWrite(trigger, HIGH);
delayMicroseconds(10);
digitalWrite(trigger, LOW);
time_taken = pulseIn(echo, HIGH);
dist= time_taken*0.034/2;
if (dist>50)
dist = 50;
}
void loop() { //infinite loopy
calculate_distance(trigger1,echo1);
distL =dist; //get distance of left sensor
calculate_distance(trigger2,echo2);
distR =dist; //get distance of right sensor
//Uncomment for debudding
/*Serial.print("L=");
Serial.println(distL);
Serial.print("R=");
Serial.println(distR);
*/
//Pause Modes -Hold
if ((distL >40 && distR>40) && (distL <50 && distR<50)) //Detect both hands
{Serial.println("Play/Pause"); delay (500);}
calculate_distance(trigger1,echo1);
distL =dist;
```

```
calculate_distance(trigger2,echo2);
distR =dist;


//Control Modes
//Lock Left - Control Mode
if (distL>=13 && distL<=17)
{
  delay(100); //Hand Hold Time
  calculate_distance(trigger1,echo1);
  distL =dist;
  if (distL>=13 && distL<=17)
  {
    Serial.println("Left Locked");
    while(distL<=40)
    {
      calculate_distance(trigger1,echo1);
      distL =dist;
      if (distL<10) //Hand pushed in
      {Serial.println ("Vup"); delay (300);}
      if (distL>20) //Hand pulled out
      {Serial.println ("Vdown"); delay (300);}
    }
  }
}
//Lock Right - Control Mode
if (distR>=13 && distR<=17)
{
  delay(100); //Hand Hold Time
  calculate_distance(trigger2,echo2);
  distR =dist;
```

```
  if (distR>=13 && distR<=17)
  {
    Serial.println("Right Locked");
    while(distR<=40)
    {
      calculate_distance(trigger2,echo2);
      distR =dist;
      if (distR<10) //Right hand pushed in
      {Serial.println ("Rewind"); delay (300);}
      if (distR>20) //Right hand pulled out
      {Serial.println ("Forward"); delay (300);}
  }
}
}

delay(200);
}
```

Python Code:

```
import serial #Serial imported for Serial communication
import time #Required to use delay functions
import pyautogui
ArduinoSerial = serial.Serial('com18',9600) #Create Serial port object call
ed arduinoSerialData
time.sleep(2) #wait for 2 seconds for the communication to get established

while 1:
    incoming = str (ArduinoSerial.readline()) #read the serial data and pri
nt it as line
    print incoming
```

```
    if 'Play/Pause' in incoming:

        pyautogui.typewrite(['space'], 0.2)

    if 'Rewind' in incoming:

        pyautogui.hotkey('ctrl', 'left')

    if 'Forward' in incoming:

        pyautogui.hotkey('ctrl', 'right')

    if 'Vup' in incoming:

        pyautogui.hotkey('ctrl', 'down')


    if 'Vdown' in incoming:

        pyautogui.hotkey('ctrl', 'up')

    incoming = "";
```

Conclusion

In this project, we have successfully demonstrated how to control a computer using hand gestures with the help of an Arduino board. By using a combination of an ultrasonic sensor and a microcontroller, we were able to capture hand gestures and convert them into specific commands for the computer.

The project can be further improved by incorporating more gestures and expanding the range of actions that can be performed by the computer. Additionally, it can be enhanced by adding more advanced features such as machine learning algorithms to improve gesture recognition and reduce false positives.

Overall, the project provides a unique and innovative way to interact with a computer, and has the potential to be used in various applications such as gaming, accessibility, and human-computer interaction. It opens up a new range of possibilities for controlling devices without the need for traditional input devices like keyboard and mouse.

In conclusion, the project of Control your Computer with Hand Gestures using Arduino is a valuable and exciting project that showcases the potential of using hand gestures as an input method for computers and other devices. It demonstrates the power of combining hardware and software to create new and innovative ways of interacting with technology.