Suman Basnet

Interaction with Images Using Hand Gestures

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics Engineering

Thesis

31 May 2016

Author(s) Title Number of Pages Date	Suman Basnet Interaction with images using hand gestures 24 pages + 2 appendices 31 May 2016
Degree	Bachelor of Engineering
Degree Programme	Electronics
Specialisation option	
Instructor(s)	Janne Mäntykoski, Senior Lecturer, Helsinki Metropolia AMK

The main objective of this Final Year Project (FYP) is to achieve prototype of an embedded system where any user can control the flow of the images in the Graphical User Interface (GUI).

This report starts with working mechanism of the sensors where gesture recognition pattern of the sensors is discussed. Then, the hardware and software requirements are enlisted with their features' description in the system specifications heading. Eventually, stepwise elaboration of two different phases of the system is discussed in the methodology heading.

The main constituents of this FYP are Arduino Nano, Ultrasonic sensors and Python Programming Language. The whole system activates when ultrasonic sensors connected to the Arduino Nano follow the hand gestures of the user. After that the Arduino Nano forward the information about the hand movements to the serial port of the computer and Python Programming language executes the commands indicated by the hand movement, changing the images in the Graphical User Interface (GUI) accordingly.

Finally; challenges, limitations and possible upgrade in the system design are discussed in the conclusion part of this report.

Keywords	Ultrasonic sensors, Arduino Nano, Interaction with images using hand gesture

Contents

Abbreviations

GUI	Graphical User Interface
CLI	Command Line Interface
PPL	Python Programming Language
GTL	Gimp Toolkit
HCI	Human Computer Interaction
FYP	Final Year Project
SONAR	Sound Navigation and Ranging
LED	Light Emitting Diode

1 Introduction

Indulging more onto the complexity of an embedded system design, we find CLI and GUI interacting between user and system functions. For instance, if you click Mozilla icon in a personal computer, Mozilla web-browser opens where you can plug and play with websites of different domain names. Till some years back, if anybody would want to establish a communication in any embedded systems, CLI and GUI would always come in tandem. Due to the continuous innovation and upgrade in technology, CLI's function has been limited to some extent. Recent development has given options like hand gesture for end-user to interact with the GUI.

From the beginning of time, hand related gestures has always been fundamental part of communication and it used in expressing variety of emotions These emotions are as diverse as taunting, disapproval, joy and affection, to commands and invocations. In other words, other than verbal communication, hand gestures are the most natural way for humans to communicate with the environment and fellow human. From the technological perspective, application of gestures can be seen in surfing images in Samsung phones, remotely controlling television and many more. Hence, we can state that the use of hand gestures provides an attractive alternative to cornerstone for HCI.

This objective of this FYP is to create a prototype where any user can control the flow or animation of images in the GUI with the help of hand gesture. It deals with the use of components like ultrasonic sensors, Arduino gadgets and other less sophisticated components like resistors, LEDs.

1.1 Thesis Structure

This thesis report starts with the introduction of the thesis topic with a brief information about its objective.

The second part is titled working mechanism where the theoretical background of the thesis is discussed.

System specifications is the third part where the components used in the development of prototype are enlisted along with their functionalities.

In the fourth part, the methodology used in the creation of final product is described in two different phases.

The FYP is reviewed in the fifth part. It summarises the project with key parts covered.

The sixth part discusses the conclusion about the project, challenges incurred and the possible upgrade of the prototype.

2 THEORETICAL BACKGROUND

In our circuitry, three ultrasonic sensors are connected with Arduino Nano. Simultaneously, Arduino Nano is connected to the serial port of computer. As soon as ultrasonic sensors achieve the operating voltage from the Arduino Nano, the TRIG pin of sensors are activated and its transmitter start to generate the ultrasonic waves of 40 KHz at regular intervals [1]. When these wave strikes upon any obstacle (hand gesture in our case) in its proximity of 20 cm, it sends back the input signal back to Arduino Nano using SONAR technology. SONAR refers to the feature of sensor which uses acoustic signal or sound waves to navigate, communicate or detect object lying in its proximity.

The receiver in the ultrasonic module receives the input signal at the pin connected to ECHO of sensor. These received waves are buffered and stored in the microcontroller. The Arduino Nano analyses the signal for a pattern. If the pattern matches a gesture in the database (encoded into Arduino), it follows the command associated with the gesture. Subsequently, Arduino Nano sent the matched commands back to the serial port of the computer in the form of logic HIGH or LOW [1].

Eventually, the GUI window (filled with images) created by the Python Programming Language reacts with respect to the gesture command sent by the Arduino Nano.

2.1 Gesture Recognition

Over the past few years, gesture recognition has become a commonplace technology and it has enabled humans and machines to interface more easily in the home [2]. Just like verbal communication, hand gestures has always been a communication tool in any culture. Moreover, it has become an innovative tool in HCI. Figure 1 illustrates the creation of gesture (by who performs gesture).It also shows how a specified gesture is co-related to the observer and the related information along with it. The representation of this mechanism is depicted in Figure 1. According to the model, gesturer's mental status define the creation of gesture which is eventually expressed through the motion of hands. In this FYP, user and observer perceives gesture as an input for stream of visual images in GUI. The production and perception model of gestures is summarised in Figure 1 [3].

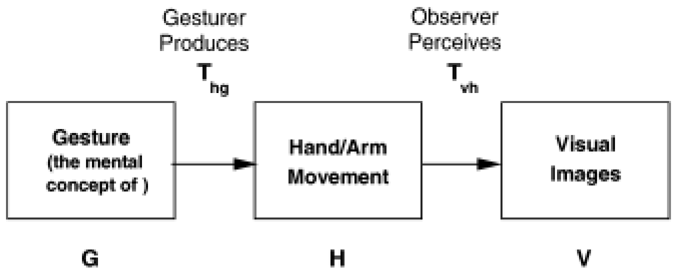


Figure 1 Copied from Huang, Pavlovic, Sharma [3]. Production and perception of gestures. Hand gestures originate as a mental concept G, are expressed (T_{hg}) through arm and hand motion H, and are perceived (T_{vh}) as visual images V.

2.1.1 Gesture recognition by Arduino with the help of Ultrasonic sensors

The three ultrasonic sensors are aligned horizontally in the breadboard to make project comprehensive from visual as well as programming side. The alignment can be seen in Figure 2.

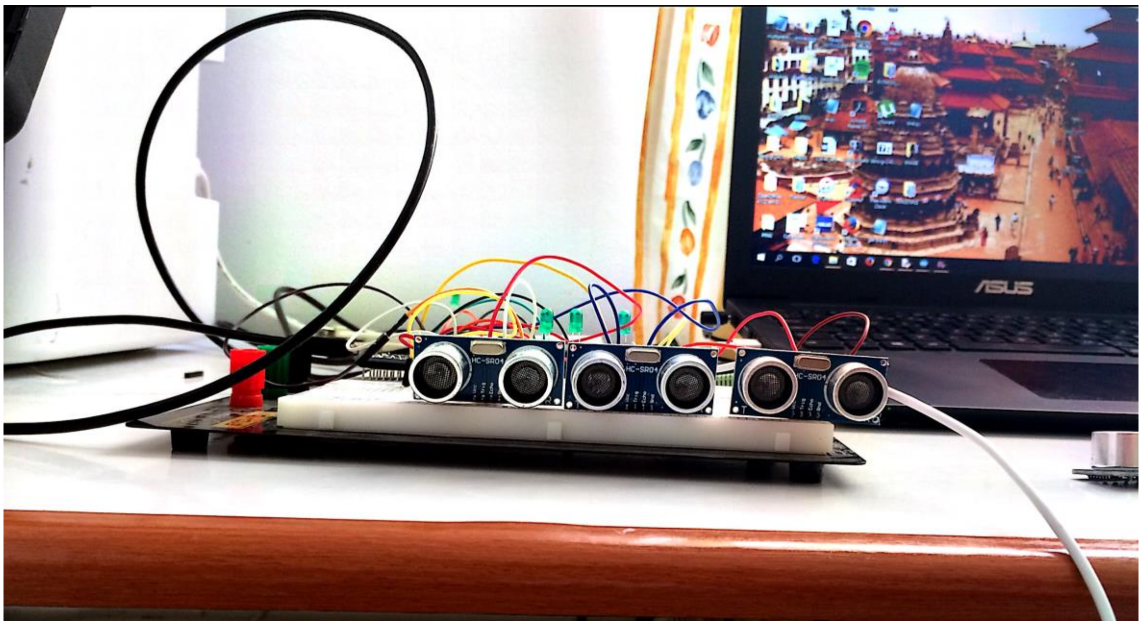


Figure 2. Three Ultrasonic sensors aligned in the breadboard

Depending on the movement of hand either from left to right or right to left, the sensors are triggered and returns the true or false values in the loop with respect to each sensor at its range of 20cm. Two gesture recognition functions are defined for the forward and backward movement of the images from the directory. If the user moves hand starting from left sensor toward right sensor (in the backward direction), the Arduino sends a ‘B’ slide command to the serial port. Similarly, if gesture is read by the sensors from right to left (in the forward direction), the Arduino sends an ‘F’ slide command to the serial port. The code for this algorithm is given in Listing 1.

Listing 1 Function for determining the direction of hand movement [4]

```
//if user moves hand left to right
  if (slide >= SLIDENONE) { // only if we are not already in
opposite move
    if ( (left) && (!right) )
      slide = SLIDE_FROM_RIGHT_SENSOR;
    if (center && (slide == SLIDE_FROM_RIGHT_SENSOR))
      slide = RIGHT_TO_CENTER;
    if (right && (slide == RIGHT_TO_CENTER))
      slideNow('T');
  }
  if (slide <= SLIDENONE) {
    if (right && (!left))
```

```
      slide = SLIDE_FROM_LEFT_SENSOR;
    if (center && slide == SLIDE_FROM_LEFT_SENSOR)
      slide = LEFT_TO_CENTER;
    if (left && slide == LEFT_TO_CENTER) {
      slideNow('G');
    }
```

2.1.2 Gesture recognition by PPL

Initially, PPL reads the command from the Arduino Nano using serial library. After that, it creates a window using GUI libraries gtk and gobject. This window is the interface which pop up the images supposed to be animated. In the following step, PPL stretches the images in the window to the full screen mode to make the GUI more user friendly. Ultimately, with respect to the command received in serial port regarding the hand gesture, PPL starts animating these images. As shown in Listing 1, when Arduino sends ‘F’ command, the images saved in the directory starts moving from first to last (in the forward direction). Similarly, if Arduino sends ‘B’ command, the images starts to move from last to first (in the backward direction). The synopsis for these PPL tasks is shown in Listing 2.

Listing 2 Creation of window to load the images to be animated [4]

```
//reading serial port and creating GUI
import serial, gtk, gobject, sys
ser = serial.Serial('COM3', 9600)
def pollSerial():
sys.stdout.write(ser.read(1))
sys.stdout.flush()
return True
if (ser):
print("Serial port " + ser.portstr + " opened.")
gobject.timeout_add(100, pollSerial)
gtk.main()

//Controlling images stored in GUI using hand gesture
def main():
```

```
global bg, image, ser
bg=newPix(gtk.gdk.screen_width(), gtk.gdk.screen_height())
loadImages()
image=gtk.image_new_from_pixbuf(pixbufs[pos])
ser = serial.Serial('COM3', 9600, timeout=0)
gobject.timeout_add(100, pollSerial)
window = gtk.Window()
window.connect("destroy", gtk.main_quit)
window.connect("key-press-event", keyEvent)
window.fullscreen()
window.add(image)
window.show_all()
gtk.main()
```

The representation of the whole system can reviewed in the flow chart as stated in the Figure 3. It discusses the stepwise task processed in the project.

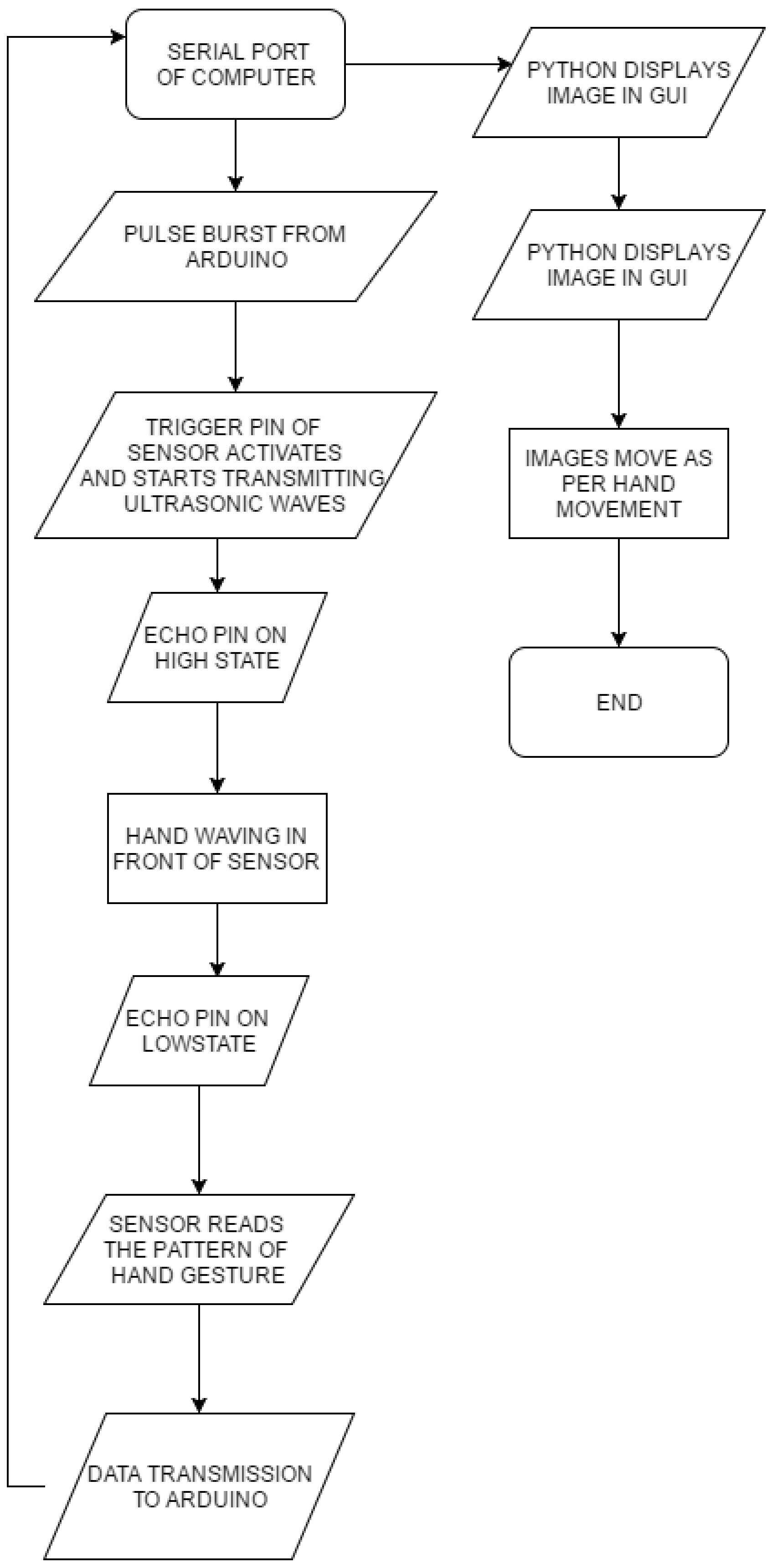

SERIAL PORT OF COMPUTER
PYTHON DISPLAYS IMAGE IN GUI
PULSE BURST FROM ARDUINO
PYTHON DISPLAYS IMAGE IN GUI
TRIGGER PIN OF SENSOR ACTIVATES AND STARTS TRANSMITTING ULTRASONIC WAVES
IMAGES MOVE AS PER HAND MOVEMENT
ECHO PIN ON HIGH STATE
END
HAND WAVING IN FRONT OF SENSOR
ECHO PIN ON LOWSTATE
SENSOR READS THE PATTERN OF HAND GESTURE
DATA TRANSMISSION TO ARDUINO

Figure 3. Flow chart of the system design

.

3 System Specifications

In order to make the whole system functional, different components are used. They are categorised into primary and secondary subgroups depending on their essence in the FYP.

3.1 Primary components

The primary components of this FYP are Ultrasonic sensors, Arduino Nano and PPL. These components, their features and functionalities are discussed in the following sub-headings.

3.1.1 Ultrasonic sensors

An ultrasonic sensor is a transceiver which propagates electrical energy as sound energy and upon sensing the echo, determines the attribute of targets. It works on the same principles as radar or sonar systems; meaning that it transmits high frequency signal and determines the proximity of an object with corresponding reflection.

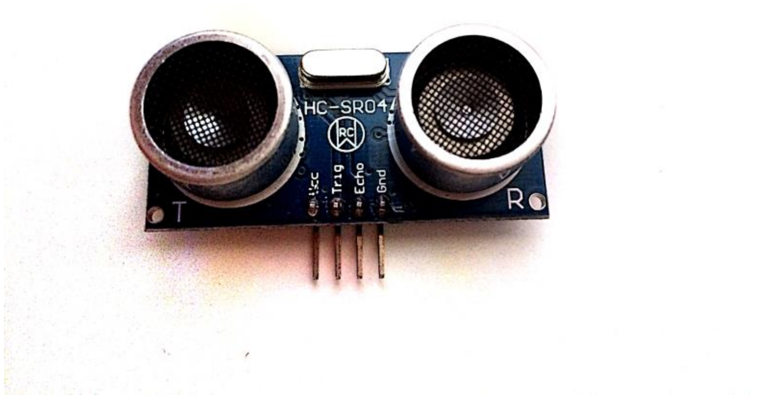


Figure 4. HC-SR04 module ultrasonic sensor

As mentioned above, the notable trait of an ultrasonic senor is determination and distance measurement of an object lying within its range. Similarly, light has no effect on it which makes it function in absolute darkness. Contrarily, its functionality is limited depending on the surface of reflection. It may not detect objects on a steep angle as very little or no sound wave is reflected back to the sensor.

In this project, HC-SR04 modelled ultrasonic sensor is used. This ultrasonic module comes with 4 pins namely Vcc, trigger, echo and ground as seen in Figure 4. Its electrical parameters are enlisted in the Table 1.

Table 1 gathered from [5; 6]. Electric parameters of HC-SR04 module

Operating voltage	5 V (DC)
Operating current	15 mA
Working frequency	40 kHz
Maximum range	4m
Minimum range	2cm
Measuring angle	15 degree
Trigger input signal	10µs TTL pulse
Echo output signal	Input TTL lever signal and range in proportion
Dimension	45X20X15mm

For the operation of this module, a short pulse of 10 µs is supplied to the trigger the ultrasonic waves. As soon as the trigger pin receives the pulse bursts, it generates 8 cycles of 40 KHz ultrasonic waves regularly. This phenomenon will raise the echo and bring the echo pin to high state. If the ultrasonic sensor senses any object in its proximity (2cm to 400cm), it returns back the burst and echo pin goes back to low state. Thus, the period of echo line (between the high and low state) is used for distance calculation. In our case, the sensors will be triggered only if user waves hand 20 cm or less from sensors. It is elaborated in Figure 5 [5; 6.]

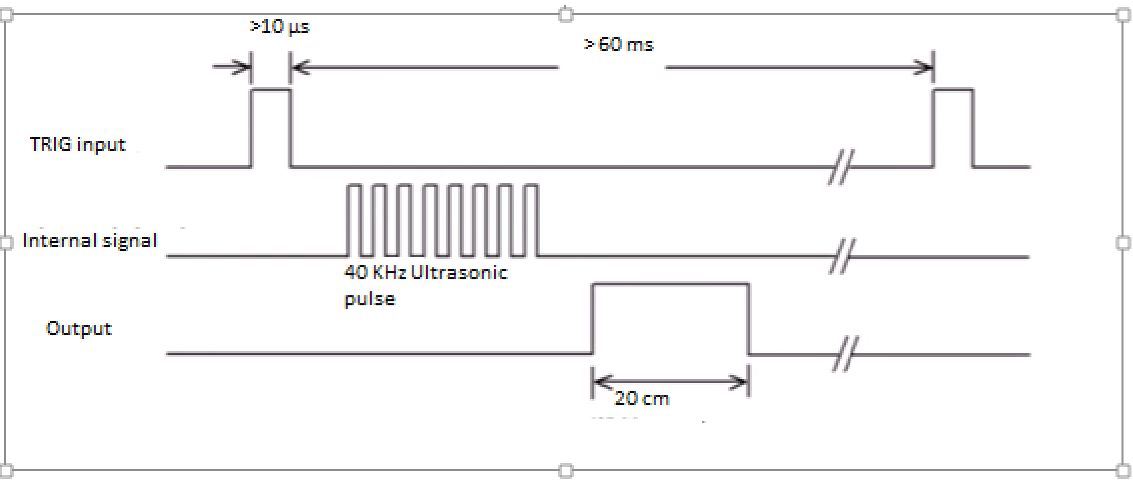


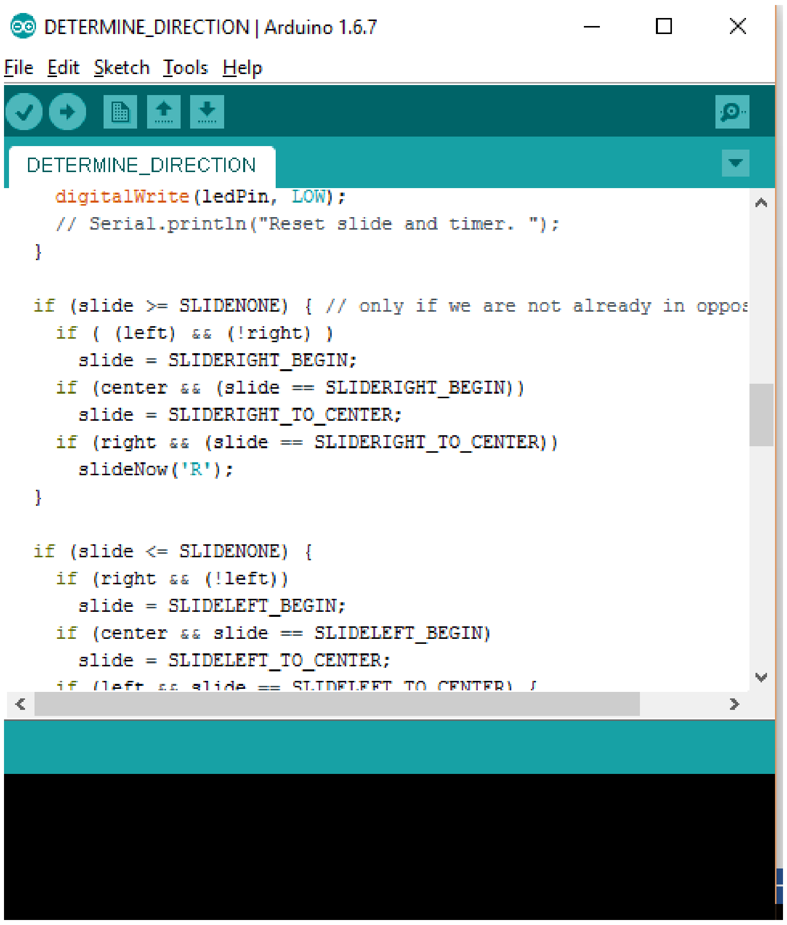
Figure 5. Timing diagram of HC-SR04 module

3.1.2 Arduino

Arduino is an open source physical computing platform used in the creation of interactive objects that either stand alone or collaborate with software in the computer. It consists of two different sections; hardware and software [7].

3.1.2.1 Arduino Software

In Arduino, software programs are called sketches. They are created on the computer using Integrated Development Environment (IDE) as shown in Figure 6. The sketches written on the IDE instructs the Arduino board to perform the functionalities [7; 8.]


DETERMINE_DIRECTION | Arduino 1.6.7
File Edit Sketch Tools Help
DETERMINE_DIRECTION

```
    digitalWrite(ledPin, LOW);
    // Serial.println("Reset slide and timer. ");
  }

  if (slide >= SLIDENONE) { // only if we are not already in oppos
    if ( (left) && (!right) )
      slide = SLIDERIGHT_BEGIN;
    if (center && (slide == SLIDERIGHT_BEGIN))
      slide = SLIDERIGHT_TO_CENTER;
    if (right && (slide == SLIDERIGHT_TO_CENTER))
      slideNow('R');
  }

  if (slide <= SLIDENONE) {
    if (right && (!left))
      slide = SLIDELEFT_BEGIN;
    if (center && slide == SLIDELEFT_BEGIN)
      slide = SLIDELEFT_TO_CENTER;
    if (left && slide == SLIDELEFT_TO_CENTER) {
```

Figure 6. IDE environment of Arduino

3.1.2.2 Arduino Hardware

The code written on the IDE is executed in the Arduino board. The communication between board and IDE is established using an USB cable. Figure 7 shows the top view of Arduino Nano.

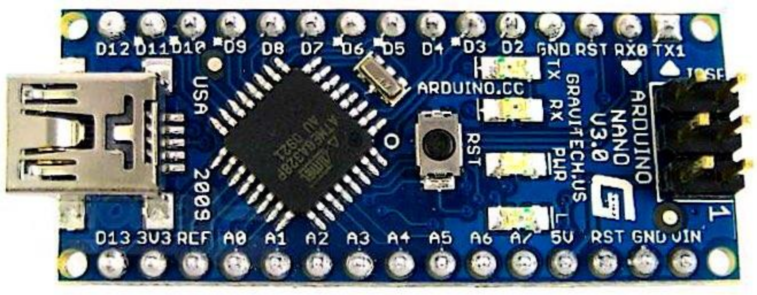

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
ARDUINO.CC
ARDUINO NANO V3.0
GRAVITECH.US
USA 2009

Figure 7. Top view of Arduino Nano V3.0

Arduino Nano V3.0 is used in this project as hardware component and Arduino 1.6.7 is used as IDE. Arduino Nano 3.0 is an open source platform based on ATmega168 micro-controller. It is powered by PC using retractable USB cable. The pin layout of Arduino Nano is outlined Figure 9.

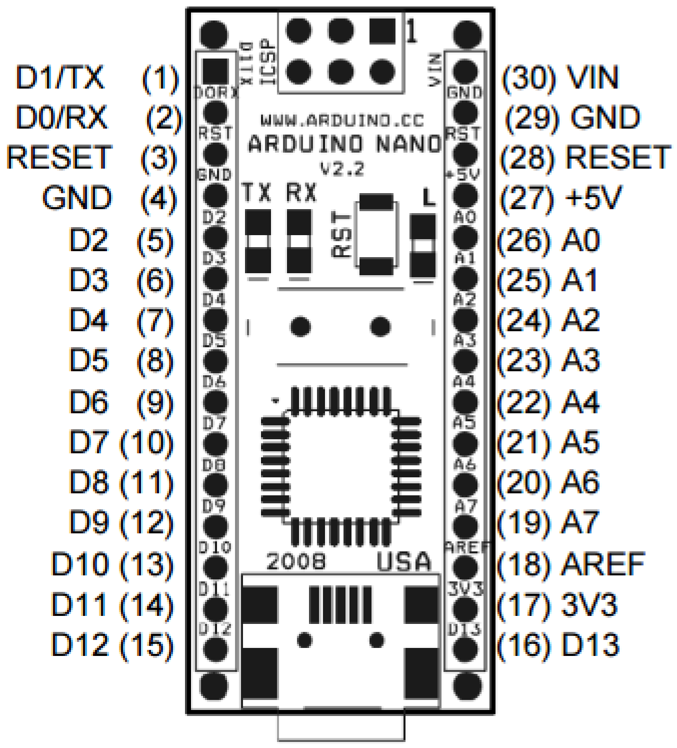


Figure 8. Pin layout of Arduino Nano as extracted from its official manual [Reprinted from 9]

The details about the pin configurations of Arduino Nano are provided in the table 2. Along with that the specific functions of the pins are also explained in the same table.

Table 2. Pin configuration of Arduino Nano [Copied from 9]

Pin number	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output(Port 0-13)
3 and 28	Reset	Input	Reset(Active low)
4 and 29	GND	PWR	Supply ground

17	3V3	Output	+3.3V(from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog-input (channel 0 to 7)
27	+5V	Output or input	+5V output (from regulator) or +5V input (from power supply)
30	VIN	PWR	Supply voltage

3.1.3 Python Programming Language

Python Programming language helps us in the graphical window where it animates the images in full-screen mode. PyGTK (python wrappers), GTK+ library, PyCairo (Python bindings), PyGObject (Python extension module), PySerial Library and PyWin32 Extensions, are installed along with main Python software. Furthermore, the functionalities of these libraries are discussed in Methodology heading under Phase II subheading.

3.2 Secondary Components

Besides the main components like Arduino, sensors and PPL, several other secondary components are used in the build-up of this FYP.

The circuitry of the prototype is developed in a breadboard modelled EIC-104. All of the components used in this FYP (except PPL) are mounted in this breadboard.

Three green LEDs (COM-09592) are used to determine the position of user. Ultrasonic sensors are programmed to be functional within 20 cm. These LEDs will blink if the user moves hand 20cm or closer from ultrasonic sensors.

Since, LEDs are highly sensitive and require a small amount of current for their operation, three resistors (260 Ohm, 240 Ohm and 300 Ohm are used) to drop the 5V voltage on the Arduino’s digital pins to a level safe to connect the LEDs.

In order to maintain the circuitry between all of these primary and secondary components, basic jumper wires are used

A laptop (produced by ASUS), having Windows 10 installed, is used to carry out the software part of the project. Along with writing code, the GUI is also created in the laptop where the images are stretched to full screen with the help of python.

4 Methodology

It is quite evident by now that the whole project comprises of two phases. First phase determines hand gesture with the help of Ultrasonic sensor and Arduino Nano. Whereas, Python Programming Language moves the images with respect to the data received in the first phase . The details concerning the illustration of Phase 1 and Phase 2 are given in the subheadings below:

4.1 Phase I

As explained by the Figure 9, the trigger pin and echo pin of ultrasonic modules are connected to the digital input/output pins of Arduino (D2, D3 and D4). Similarly, Vcc (5V) pins are connected to pin 27 of Arduino, namely +5V. Along with these, the remaining ground pins are commonly grounded with ground pin of Arduino.

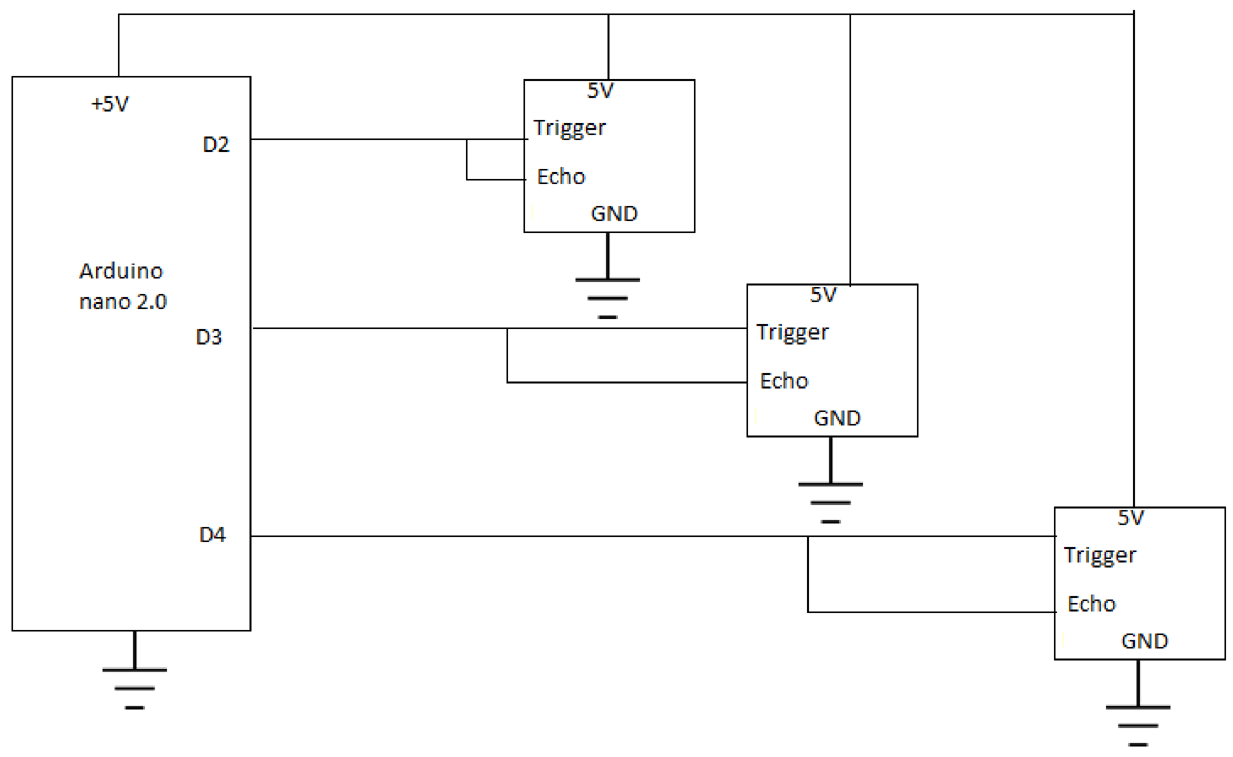


Figure 9. Connection diagram between sensor and Arduino in breadboard

In the following step, the circuitry was tested using inbuilt sensor testing program of Arduino environment called Ping. After the successive testing of all three sensors using same program, the circuit was tested using LEDs. First of all, each ultrasonic sensor is connected with a LED. The cathode of LEDs are grounded using resistors and anodes were connected to Arduino data pins. After that, the program is written in such a way that if a user moves hand closer than 20cm to any sensor, the LED connected to respective will blink. Figure 10 shows the inactivity of sensors if there is no obstacle in its proximity.

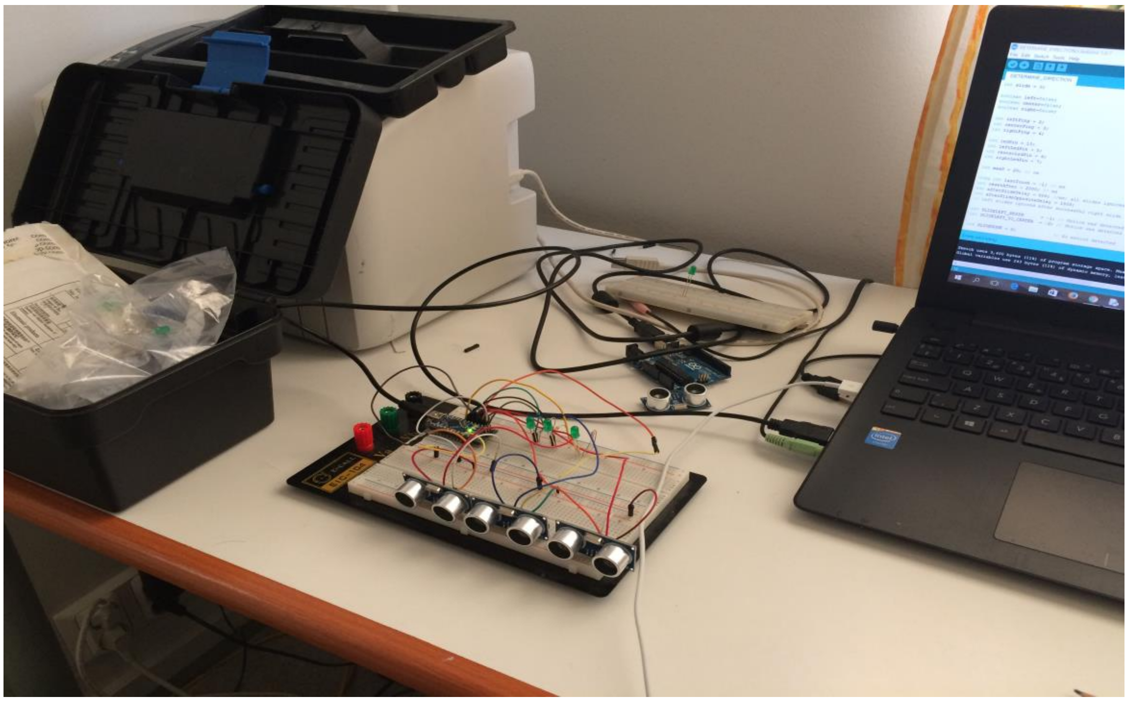


Figure 10. LEDs do not lit on if there is no obstruction

Whereas, Figure 11 clarifies that if there is any obstacle in its proximity as understated above (20 cm), the LEDs starts to blink.

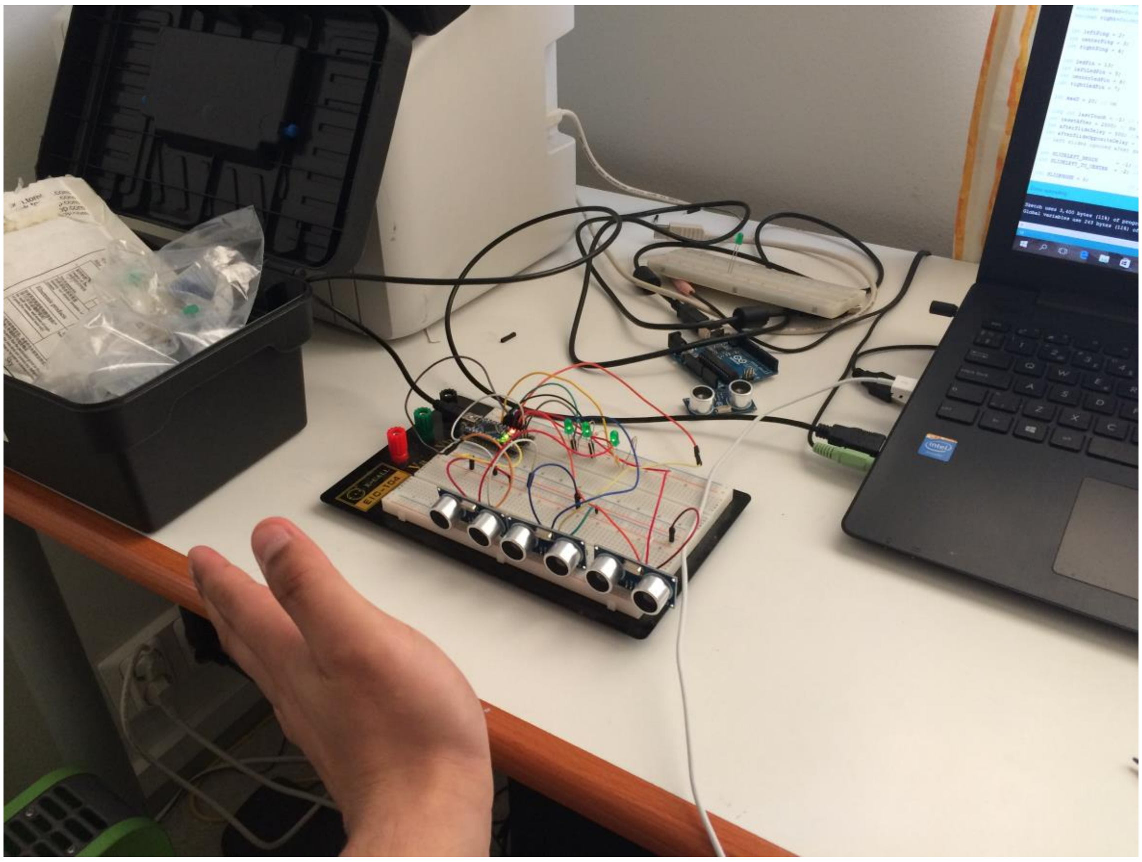


Figure 11. LEDs blinking when sensor found obstruction less than 20 cm

4.2 Phase II

After the completion of first phase, Python 2.0 is installed in the computer. Python is an open source, object oriented, high level programming language with dynamic semantics. The primary reasons behind Python being programmer's goto tool are outline below:

a) Quality

The readability, coherence, and software quality are the traits which sets it apart from other tools in the scripting world. Python code is designed to be reusable and maintainable—much more so than traditional scripting languages [10, 56.]

b) Productivity

If compared to other traditional programming languages like C, C++, and Java, developer can easily identify that Python boosts up the programming process by appearing in smaller size. Generally, Python code is three to five times smaller than equivalent C++ or Java code. That means there is less to lines to code, debug and maintain. Python programs also run immediately, without the lengthy compile and link steps required by some other tools [10, 56.]

c) Portability

Python works easily in cross- platform like Linux, Mac and Windows without any complacency. Likewise, it has functionalities over creation of GUI, database access and web-based programming [10, 56.]

d) Wide range of libraries and integration

It has large collection of standard library which is helpful in application level programming tasks. Similarly, its function can be easily called from other languages like C, C++, Java and .NET [10, 56.]

e) Pleasure

Due to these aforementioned qualities in terms of ease of use and built-in toolset, it makes the act of programming more than fun. The strong uphold in Google, YouTube, Bit torrent file sharing, NASA's Jet propulsion lab, etc. are the answers if questioned upon its credibility [10, 56].

As discussed in the introductory part of this report, GUI has become essence of any software package. The simplicity of Python programming is quite handy for the creation of GUIs in desktop. With the help of proper extensions and libraries, GUI support is used in different toolkits within Python such as GTK with PyGTK, Qt with PyQt, .NET with IRonPython and tkinter. In this FYP, Python uses Gimp Toolkit+ (GTK+) for the creation of GUI [11.]

4.2.1 GTK+ toolkit

It is a cross-platform (applicable in Windows, Linux and Mac) toolkit used in creation of GUIs. It is licensed under GNU Lesser General Public License (LGPL) which primarily means it is free to use. It requires several binding and wrappers to communicate with IDE sketches generated by the Arduino Nano. Bindings are the bridge between two different programming languages. Similarly, wrappers are the package of the functions which is called by the main function of the program to make the computing easier and less sophisticated.

4.2.2 GTK

It acts as a convenient wrapper for the GTK+ library. It helps to create function in the program where desired attributes are added. In our case, images required to move with the help of hand gesture are added in the GUI window created by the python [11,360].

4.2.3 Cairo Graphics library

Cairo is 2D graphics library which provides vector graphics-based and device independent Application Programming Interface (API). Cairo is implemented as a library written in the C programming language but there are several bindings to get it linked with different programming languages. In our case, PyCairo acts as a binding which maintains a co-relation between PPL and Arduino IDE (written in C language).

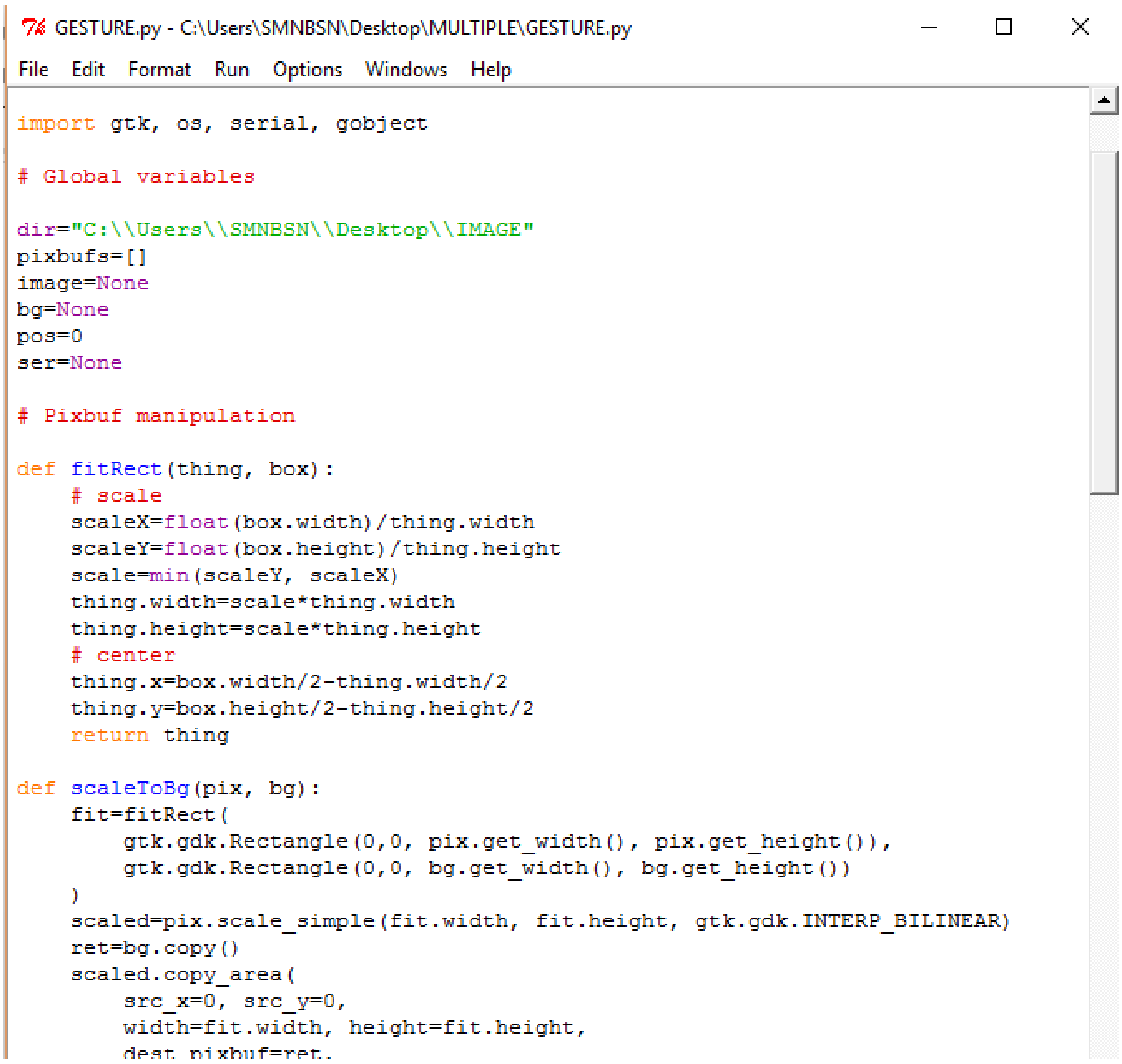
4.2.4 PySerial

This library obliges us with the access of the serial port. It provides backend for Python to run in the windows and accessibility to the port settings. In our case, this helps Python in reading the command for forward and backward hand gesture from the serial port to execute the GUI accordingly.

4.2.5 GObject

It provides the object- orienting platform and cross language interoperability like Py-Cairo. Since, we are working two different languages (Arduino IDE and Python), Py-GObject is the binding.

Figure 12 shows the GUI shell of python where the codes are written for animating the images from the designated directory. Similarly, it also shows libraries like gtk, os, serial and gobject running in it.


GESTURE.py - C:\Users\SMNBSN\Desktop\MULTIPLE\GESTURE.py
File Edit Format Run Options Windows Help

```
import gtk, os, serial, gobject

# Global variables

dir="C:\\Users\\SMNBSN\\Desktop\\IMAGE"
pixbufs=[]
image=None
bg=None
pos=0
ser=None

# Pixbuf manipulation

def fitRect(thing, box):
    # scale
    scaleX=float(box.width)/thing.width
    scaleY=float(box.height)/thing.height
    scale=min(scaleY, scaleX)
    thing.width=scale*thing.width
    thing.height=scale*thing.height
    # center
    thing.x=box.width/2-thing.width/2
    thing.y=box.height/2-thing.height/2
    return thing

def scaleToBg(pix, bg):
    fit=fitRect(
        gtk.gdk.Rectangle(0,0, pix.get_width(), pix.get_height()),
        gtk.gdk.Rectangle(0,0, bg.get_width(), bg.get_height())
    )
    scaled=pix.scale_simple(fit.width, fit.height, gtk.gdk.INTERP_BILINEAR)
    ret=bg.copy()
    scaled.copy_area(
        src_x=0, src_y=0,
        width=fit.width, height=fit.height,
        dest_pixbuf=ret,
```

Figure 12. GUI shell of Python

5 REVIEW

The control of GUI with hand gestures is really appealing. But the excitement comes with the challenge of limiting unintended hand motions. This step includes writing and testing the system made so far with a program whose supposed function is making the last sensor determine the direction of user's hand. This particular program identifies the direction of hand using the variable declared in the program. With the help of sensor, arduino informs the serial port whether user has waved hand left to right (B) or from right to left (F). Visually, we can identify the activity of particular sensor with the help of LEDs, meaning that LED will blink as user moves hand in front of sensor connected with it. This is also explained by the figure 7 and figure 8.

On the other hand, with respect to the Arduino Nano's command, PPL returns the output . First of all, code is written to read the serial port of the computer. The aim of this code is to read the information coming from Arduino through serial port of the computer. The second code picked images saved inside specified directory and stretched it to fullscreen. Following the subsequent success of reading serial and fitting images to full screen, the prominent part of coding is done where following the movement of hand (sent by the Arduino to the serial port), Python moves the image in the display accordingly.

Hence, the final prototype of the project was achieved where the user can move his/her hand to change the images in GUI.

The pictorial representation of how system works when user moves hand in the forward direction is given in Figure 13.

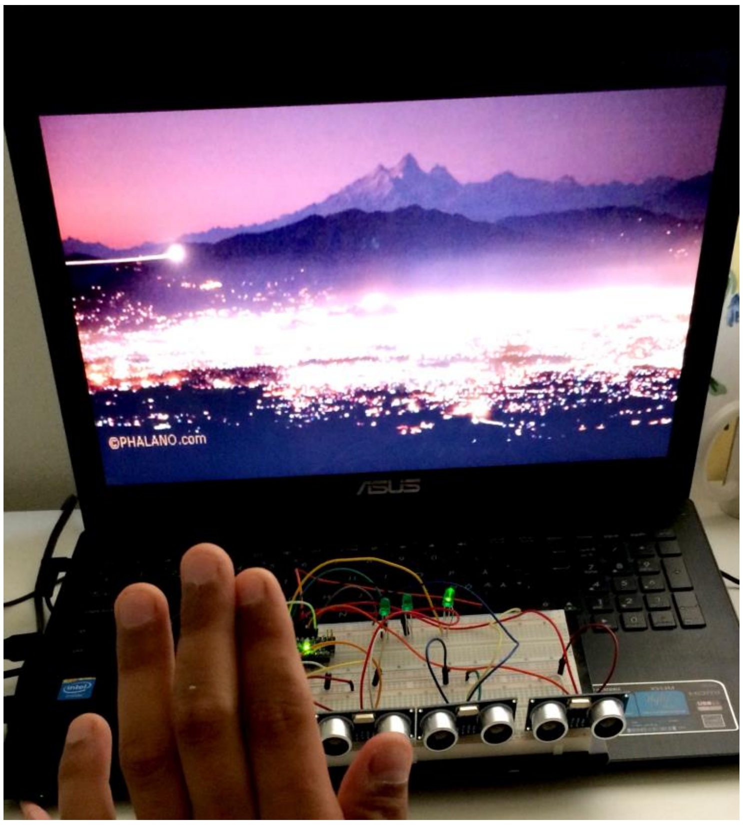


Figure 13. System when user moved hand forward

Likewise, Figure 14 shows the change of image in GUI when the user moved hand in the backward direction.

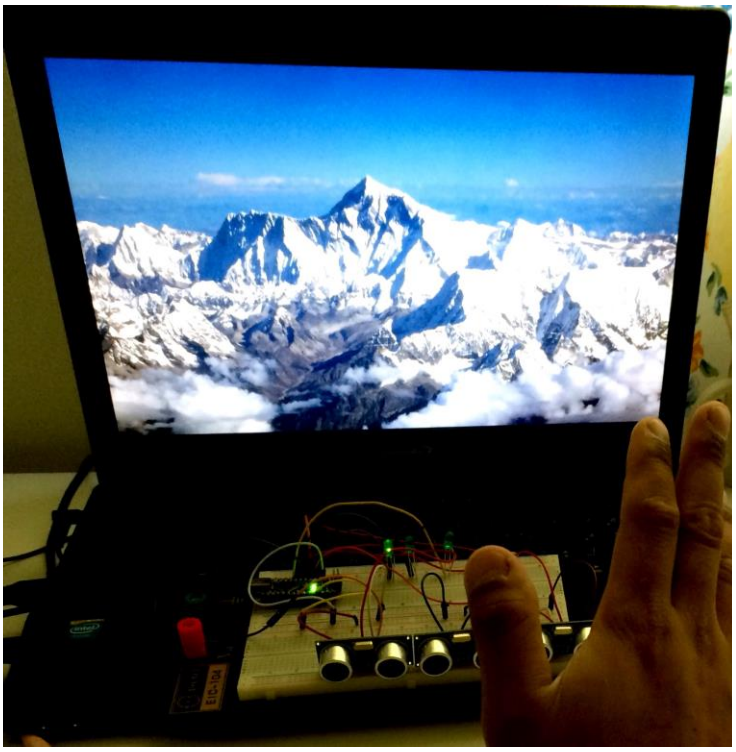


Figure 14. system when user moved hand backward

6 CONCLUSION

Though gesture recognition is still in its infancy state, the innovative market scenario which has seen Virtual Reality being a global phenomenon pushes us to admit that its application areas could keep on growing. Conclusively, the whole project proved to be a good learning curve. Some of the challenges that incurred during the FYP are as follows:

a) Hardware malfunction: Arduino Nano ordered from internet turned out to be faulty.

b) Short-circuiting: While building the circuitry, there were short-circuiting on several occasions because of bad pin configuration. Two ultrasonic sensors were lost in the process.

c) Coding errors: There were repetitive compiling errors in codes to determine the position of user and animating images.

d) Project management: At times, when some part of the prototype was not working, the whole project got stalled for considerable amount of time.

But, the theory learned and small projects accomplished in the courses like embedded systems, advanced programming and microcontrollers came with great help. Likewise, extensive resources of school like library and e-portal also came up with immense support.

As Scandinavian mentality says about room for perfection, there are also areas of improvement in this project. The whole project is carried out in the test board and personal computer in order to build the prototype. If used proper resources, a proper user-GUI orientation system can be built in sophisticated way. An example could be configured in the PC of first-floor lobby inside Metropolia Albertinkatu Campus.

References

1. Nidhi Gupta, Ramandeep Singh , Sidharth Bhatia. Hand gesture recognition using ultrasonic sensor and atmega128 microcontroller; June 2014 [online] http://esatjournals.net/ijret/2014v03/i06/IJRET20140306107.pdf

2. Dong-Ik Go, Gaurav Agarwal. Gesture Recognition: Enabling natural interactions with electronics; April 2012 [online]
   URL: http://www.ti.com/lit/wp/spry199/spry199.pdf

3. Huang, Pavlovic, Sharma. Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review; July 1997 [online] http://www.cs.rutgers.edu/~vladimir/pub/pavlovic97pami.pdf

4. Kimmo Karvinen, Tero Karvinen. Make: Arduino Bots and Gadgets First Edition. United States of America: O'Reilly Media, Inc.; 2011

5. Lentin Joseph. Learning Robotics Using Python. Birmingham UK: Packt Publishing Ltd; May 2015

6. Fernandez, Mahtani, Crespo, Martinez. Learning ROS for Robotics Second Edition. Birmingham UK: Packt Publishing Ltd; May 2015

7. Massimo Banzi, Michael Shiloh. Make: Getting started with Arduino Third Edition. USA: Maker Media Inc.; December 2014

8. Michael Margolis. Arduino Cookbook First Edition. USA: O'Reilly Media, Inc.; 2011

9. Creative Commons Attribution Share-Alike 2.5 License. Arduino Nano (V 2.3) User Manual [online]
   URL: https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf

10. Mark Lutz. Learning Python Fifth Edition. USA: O’Reilly Media, Inc.; 2013

11. Mark Lutz. Programming Python Fourth Edition. USA: O’Reilly Media, Inc.; 2013

Determination of hand movement by Arduino

```
/*
 Based on code by David A.Melis,Tom Igoe
 Kimmo Karvinen, Tero Karvinen, Joe Savedra
 Edited as required by Suman Basnet, 2016
 */

int slide = 0;

boolean left=false;
boolean center=false;
boolean right=false;

int leftsensor = 2;
int centersensor = 3;
int rightsensor = 4;
int ledPin = 13;
int leftLED = 5;
int centerLED = 6;
int rightLED = 7;
int maxD = 20; // cm

long int lastTouch = -1;
int resetAfter = 2000;
int afterSlideDelay = 500;
int afterSlideOppositeDelay = 1500;

int SLIDE_FROM_LEFT_SENSOR = -1; // Motion was detected from right
int LEFT_TO_CENTER  = -2; // Motion was detected from right to center

int SLIDENONE = 0;             // No motion detected

int SLIDE_FROM_RIGHT_SENSOR     = 1;  // Motion was detected from left
int RIGHT_TO_CENTER = 2;  // Motion was detected from left to center
```

```
void setup() {
  Serial.begin(9600); // bit/s
  pinMode(leftLED, OUTPUT);
  pinMode(centerLED, OUTPUT);
  pinMode(rightLED, OUTPUT);
}
void loop() {
  left = ping(leftsensor, leftLED);
  center = ping(centersensor, centerLED);
  right = ping(rightsensor, rightLED);
  if (left || center || right) {
    lastTouch=millis();
  }
  if (millis()-lastTouch>resetAfter ) {
    slide=0;
    digitalWrite(ledPin, LOW);
    // Serial.println("Reset slide and timer. ");
  }
  if (slide >= SLIDENONE) { // only if we are not already in opposite move
    if ( (left) && (!right) )
      slide = SLIDE_FROM_RIGHT_SENSOR;
    if (center && (slide == SLIDE_FROM_RIGHT_SENSOR))
      slide = RIGHT_TO_CENTER;
    if (right && (slide == RIGHT_TO_CENTER))
      slideNow('T');
  }
  if (slide <= SLIDENONE) {
    if (right && (!left))
      slide = SLIDE_FROM_LEFT_SENSOR;
    if (center && slide == SLIDE_FROM_LEFT_SENSOR)
      slide = LEFT_TO_CENTER;
    if (left && slide == LEFT_TO_CENTER) {
      slideNow('G');
    }
  }
  delay(50);
}
boolean ping(int pingPin, int ledPin) {
  int d = getDistance(pingPin); //cm  bl
  boolean pinActivated = false;
  if (d < maxD) {
    digitalWrite(ledPin, HIGH);
    pinActivated = true;
  }
  else {
    digitalWrite(ledPin, LOW);
    pinActivated = false;
  }
  return pinActivated;
}
int getDistance(int pingPin) {
  long duration, inches, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);
   cm = microsecondsToCentimeters(duration);
  return(cm);
}
void slideNow(char direction) {
  if ('T' == direction)
     Serial.println("F");//Forward gesture forwarded to Serial PORT
  if ('G' == direction)
     Serial.println("B");//Backward gesture forwarded to Serial PORT
  digitalWrite(ledPin, HIGH);
  delay(afterSlideDelay);
  slide = SLIDENONE;
}
long microsecondsToCentimeters(long microseconds) {

  return microseconds / 29 / 2;
}
```

Changing images in GUI with Python

```
#!/usr/bin/env python
# moveimages.py - Choose full screen image by waving hand.
# (c) Kimmo Karvinen & Tero Karvinen
#  edited as system required by Suman Basnet
 import gtk, os, serial, gobject
# Global variables
dir="C:\\Users\\SMNBSN\\Desktop\\IMAGE"
pixbufs=[]
image=None
bg=None
pos=0
ser=None
# Pixbuf manipulation
                    def fitRect(thing, box):
                      # scale
                       scaleX=float(box.width)/thing.width
                    scaleY=float(box.height)/thing.height
                    scale=min(scaleY, scaleX)
                     thing.width=scale*thing.width
                    thing.height=scale*thing.height
                     # center
                     thing.x=box.width/2-thing.width/2
                    thing.y=box.height/2-thing.height/2
```

```
                 return thing

                def scaleToBg(pix, bg):
                 fit=fitRect(
                 gtk.gdk.Rectangle(0,0, pix.get_width(), pix.get_height()),
                gtk.gdk.Rectangle(0,0, bg.get_width(), bg.get_height())
    )
    scaled=pix.scale_simple(fit.width, fit.height, gtk.gdk.INTERP_BILINEAR)
    ret=bg.copy()
    scaled.copy_area(
        src_x=0, src_y=0,
        width=fit.width, height=fit.height,
        dest_pixbuf=ret,
        dest_x=fit.x, dest_y=fit.y
    )
    return ret
        def newPix(width, height, color=0x000000ff):
   pix=gtk.gdk.Pixbuf(gtk.gdk.COLORSPACE_RGB, True, 8, width , height)
    pix.fill(color)
    return pix
# File reading
def loadImages():
    global pixbufs
    for file in os.listdir(dir):
        filePath=os.path.join(dir, file)
        pix=gtk.gdk.pixbuf_new_from_file(filePath)
        pix=scaleToBg(pix, bg)
        pixbufs.append(pix)
        print("Loaded image "+filePath)
# Controls
def go(relativePos):
    global pos
    pos+=relativePos
    last=len(pixbufs)-1
      if pos<0:
        pos=last
    elif pos>last:
        pos=0
    image.set_from_pixbuf(pixbufs[pos])
                    def pollSerial():
                     cmd=ser.read(size=1)
                     print("Serial port read: \"%s\"" % cmd)
                    if cmd=="F"
                     go(1)
                      elif cmd=="B":
                    go(-1)
                     return True
 # Main
def main():
    global bg, image, ser
    bg=newPix(gtk.gdk.screen_width(), gtk.gdk.screen_height())
    loadImages()
    image=gtk.image_new_from_pixbuf(pixbufs[pos])

    ser = serial.Serial('COM3', 9600, timeout=0)
    gobject.timeout_add(100, pollSerial)
    window = gtk.Window()
    window.connect("destroy", gtk.main_quit)
    window.fullscreen()
    window.add(image)
    window.show_all()
    gtk.main()
                    if __name__ == "__main__":
                                        main()
```