GESTURE RECOGNITION USING PROXIMITY SENSORS WITH MOJO

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment of the Requirements for the Degree
Master of Science in Computer Science

by
Nagendra Balachandra
Spring 2016
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the
Thesis of Nagendra Balachandra:

Gesture Recognition Using Proximity Sensors with Mojo

Carl Eckberg, Chair
Department of Computer Science

Alan Riggins
Department of Computer Science

Frederick J. Harris
Department of Electrical and Computer Engineering

April 13, 2016
Approval Date
ABSTRACT OF THE THESIS

Gesture Recognition Using Proximity Sensors with Mojo
by
Nagendra Balachandra
Master of Science in Computer Science
San Diego State University, 2016

Gesture-based interfaces provide an intuitive way for users to specify commands and interact with computers, it pertains to recognizing meaningful expressions of motion by a human, involving the hands, arms, and body. In this project we will be using proximity sensors to recognize the gesture.

This project concentrates on embedding this feature into GIS which helps disabled users to interact with the tool in effective way. Ultrasonic sensors are one of the most effective proximity computing devices which use speed of sound pulse to compute the distance between the transmitter and object. Arduino open source electronic help us to process the data from these sensors and help transmitting these data in to our GIS system where the data is used to evaluate gestures made by the user.

The data sent by the Arduino is later processed and events are triggered based on gesture evaluation. Gesture evaluation uses advanced algorithms which are more efficient than other gesture evaluation algorithms like image subtractions etc. GIS is a venerable tool which used widely by users for interacting with map objects. This project extends this tool by adding the feature called Gesture mode where the user can use gestures to control the map. This plays a major role for people who suffer with polio, who have a hard time using their hand for typing purpose. Proximity based gesture evaluation also opens up the door for all desktop games, which can easily access this tool for advance development.

Language and Technologies used are Arduino, APL (Arduino programming language), JAVA, ESRI map package, JSSC Java serial connector.
TABLE OF CONTENTS

ABSTRACT ............................................................................................................................. iv
LIST OF TABLES .................................................................................................................... viii
LIST OF FIGURES ................................................................................................................. ix
ACKNOWLEDGEMENTS ....................................................................................................... x

CHAPTER

1 INTRODUCTION .............................................................................................................. 1

2 TECHNOLOGIES USED IN APPLICATION .................................................................... 2

   2.1 Arduino .................................................................................................................. 2

   2.2 APL (Arduino Programming Language) ............................................................... 4

   2.3 Ultrasonic Sensors ............................................................................................... 4

       2.3.1 Distance Calculation ................................................................................. 4

       2.3.2 The HC-SR04 Sensor .............................................................................. 5

   2.4 Java ....................................................................................................................... 5

   2.5 ESRI Map Package ............................................................................................ 7

   2.6 Map Objects Java Edition ................................................................................... 8

   2.7 jSSC Java Serial Connector .............................................................................. 9

   2.8 Shapefile .......................................................................................................... 9

       2.8.1 Why Shape File? ..................................................................................... 9

       2.8.2 How Shape Files are Created? ................................................................. 9

       2.8.3 Shape File Technical Description ......................................................... 10

       2.8.4 Naming Conventions ............................................................................ 10

       2.8.5 Numeric Types ...................................................................................... 11

       2.8.6 Organization of the Main File ............................................................. 11

       2.8.7 Byte Order .......................................................................................... 11

       2.8.8 The Main File Header ....................................................................... 12
LIST OF TABLES

PAGE

Table 2.1. Description of the Main File Header .................................................................12
Table 2.2. Description of Main File Record Headers ........................................................14
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Arduino Architecture</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Ultrasonic sensors.</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Organization of the Main File.</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>Shape Type.</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>Architecture</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Ultrasonic sensor.</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Data Flow</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>Serial Monitor</td>
<td>22</td>
</tr>
<tr>
<td>4.5</td>
<td>Default map package.</td>
<td>23</td>
</tr>
<tr>
<td>4.6</td>
<td>Class Diagram of the Package Interfaces and Classes.</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Gesture mode</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>Field of operation</td>
<td>27</td>
</tr>
<tr>
<td>5.3</td>
<td>Gesture Mode</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Gesture developer console</td>
<td>28</td>
</tr>
<tr>
<td>5.5</td>
<td>Event loop</td>
<td>29</td>
</tr>
<tr>
<td>5.6</td>
<td>Pseudo – Code</td>
<td>30</td>
</tr>
<tr>
<td>6.1</td>
<td>Gesture Edit Mode</td>
<td>31</td>
</tr>
<tr>
<td>6.2</td>
<td>Gesture mode 1x Scaling</td>
<td>32</td>
</tr>
<tr>
<td>6.3</td>
<td>Scaling Is 0.5x</td>
<td>33</td>
</tr>
<tr>
<td>6.4</td>
<td>Serial-In</td>
<td>34</td>
</tr>
<tr>
<td>6.5</td>
<td>Scaling 3X</td>
<td>34</td>
</tr>
<tr>
<td>6.6</td>
<td>Scaling 0.3X</td>
<td>35</td>
</tr>
<tr>
<td>6.7</td>
<td>Connection Not Found</td>
<td>36</td>
</tr>
<tr>
<td>6.8</td>
<td>Event Triggered</td>
<td>37</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to express my special thanks and gratitude to my committee chair Professor Carl Eckberg for advising and giving suggestions throughout my thesis. Without his guidance and help this dissertation would not have been possible.

I would like to thank Professor Alan Riggins for being part of the thesis committee. Also, I would like to thank Professor Freaderic j Harris for being part of the thesis committee.
CHAPTER 1

INTRODUCTION

Gesture-based interfaces provide an intuitive way for users to specify commands and interact with computers, it pertains to recognizing meaningful expressions of motion by a human, involving the hands, arms, and body. In this project we will be using proximity sensors to recognize gestures.

This project is built for helping people suffering from polio and other physically impairments and helps them to interact with the system effectively and efficiently. Project uses proximity based gesture evaluation which way different compared to a vision based system. Document will go through the advantages of using proximity based gesture evaluation and shows detail comparison with the vision based gesture evaluation. Further more it explains the architectural implementation of this project.

This project builds an add-on to the ESRI map package (legacy mojo software) which provide the map package which comes with map controlling objects used in the development of map based application. This project extends their functionality by building a gesture mode tool using proximity based gesture evaluation to control the map on the screen. This add-on is not package specific; this architectural implementation can be extended to various applications and can be used in different ways.

The user has to get into the gesture mode in order to use the complete functionality the gestures. User can also set the focus point which specifies the system range of operation (ROP). Gesture mode reads the value specified in the ROP and sets the field of interest where the gesture takes place. Detailed explanation of these features are found in the up coming chapters.
CHAPTER 2

TECHNOLOGIES USED IN APPLICATION

The emphasis of this chapter is on the technologies used in the implementation of this project.

The following technologies have been used:

- Arduino
- APL (Arduino programming language)
- Ultrasonic Sensors
- JAVA
- ESRI map package
- JSSC Java serial connector

All the technologies/ software’s mentioned above are open source and one can use them for free to build an application.

2.1 ARDUINO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects. Arduino senses the environment by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs [1].

Arduino has a simple and accessible user experience. It runs on Mac, Windows, and Linux. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced
users. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.

Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works [2].

Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino Board Layout:
Starting clockwise from the top center:

1. Analog Reference pin (orange)
2. Digital Ground (light green)
3. Digital Pins 2-13 (green)
4. Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (digital Read and digital Write) if you are also using serial communication (e.g. Serial. begin). Reset Button - S1 (dark blue)
5. In-circuit Serial Programmer (blue-green)
6. Analog In Pins 0-5 (light blue)
7. Power and Ground Pins (power: orange, grounds: light orange)
8. External Power Supply In (9-12VDC) - X1 (pink)
9. Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
10. USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

### 2.2 APL (ARDUINO PROGRAMMING LANGUAGE)

Arduino language is merely a set of C/C++ functions that can be called from your code. Your sketch undergoes minor changes (e.g. automatic generation of function prototypes) and then is passed directly to a C/C++ compiler (avr-g++). Arduino programs can be divided in three main parts: structure, values (variables and constants), and functions [2].

### 2.3 ULTRASONIC SENSORS

#### 2.3.1 Distance Calculation

The speed of sound varies according to the temperature of the environment. Let’s assume the temperature of the environment is 20°C: the speed v of sound is 343,3 m/s. The sensor returns the measured time in microseconds (µs); however, may be useful to calculate the distance in centimeters. So, we calculate the speed of sound in cm/µs [4]:

\[
1 \text{ m/s} = 10^2 \text{ cm} / 10^6 \text{ µs}
\]

\[
= 10^{-4} \text{ cm/µs} \Rightarrow V = 343.5 \text{ m/s} * 10^{-4}
\]

\[
= 3.4*10^{-2} \text{ cm/µs}
\]
Now, we calculate the distance traveled by the sound wave: \( s = v \times t = 3.4 \times 10^{-4} \times t \).

Pay attention: the wave travel the same distance twice, going and return. In conclusion, the final formula for calculating the distance is:

\[ S = 1.7 \times 10^{-2} \times t \]

2.3.2 The HC-SR04 Sensor

The HC-SR04 has 4 pins: Vcc (+5V), Trigger, Echo and GND. You send a high impulse on the Trigger pin for 10 microseconds. In this way, the sensor send the sound waves and waits for its return. Then, the sensor will respond on the Echo pin, with an high impulse of duration directly proportional to the duration of the sound waves’ travel. You need to wait at least 60 milliseconds from an impulse to another one, in order to avoid interferences with the next measurement [4].

![Figure 2.2. Ultrasonic sensors. Source: [4]](image)

2.4 JAVA

The Java Programming Language is a general-purpose, concurrent, strongly typed, class-based object-oriented language. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture [5]. The advantages of Java are as follows:

- Easy to learn -
  Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn compared to other programming languages.
Object-oriented -
This allows you to create modular programs and reusable code.

Platform-independent -
One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

Distributed -
Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it. Writing network programs in Java is like sending and receiving data to and from a file.

Secure -
Java considers security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.

Robust -
Robust means reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detect many problems that would first show up during execution time in other languages.

Multithreaded -
Multithreaded is the capability for a program to perform several tasks simultaneously within a program. In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading [6].

The latest version is Java 8. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms. The version used for this project is Java 8 which has lot of new features like:

- It enables the Internet of Things. - Both Java SE 8 and Java ME 8 the two platforms that make up Java 8, along with Oracle’s related embedded products provide a scalable, flexible, secure development platform for the Internet of Things.

- Less code means more productivity. - Java 8 facilitates improved developer productivity through reduced boilerplate code. This is due in large part to lambda expressions, which are the most significant and anticipated new feature in Java SE 8.

- Modernize your apps. - Java 8 takes a giant step forward in modernizing the Java language and Java libraries. One example is a new date and time application programming interface that reduces the complexity for developers when handling date and time, especially when dealing with internationalization and localization for different markets.

- Create eye-popping graphics. - JavaFX 8 is a graphics toolkit included in JDK 8 that lets developers build client applications using standard Java development tools. It includes an embedded graphics stack, new UI controls and 3D graphics features, and
HTML5 support. JavaFX 8 also brings a fresh, modern look to applications with its new Modena theme.

Java 8 is integrated with JavaScript. Java SE 8 includes Nashorn, a JavaScript engine that runs on the Java Virtual Machine (JVM) and allows Java applications to contain components written in JavaScript.

### 2.5 ESRI Map Package

Map packages can streamline workflows in several ways. They can standardize the cartography and data representation used by an organization through supplying the schemas and symbolization. Data collection tasks can be more quickly and correctly accomplished by supplying staff or contractors with map packages. Map packages can be used to provide clients with deliverables as a fully functional digital version of the map that they can interact with and better understand. Finally, map packages are an efficient method for digitally archiving maps locally [7].

Maps are phenomenally efficient in presenting a great deal of information in a manner that is easily comprehended. Because GIS maps are interactive, they can be used to tell stories and let us interact with the results of analysis, understand the relationships in systems like water utilities, or understand the operational status of emergency response situations.

Map packages can be created interactively through the Arc-Map interface or by using the geo processing tools in the Package toolset. When using either method, ArcGIS prepares and validates the data that will be included in the map package along with the map document. Different data types are handled in different ways during this packaging process:

- If a layer references data in a personal geodatabase, the reference will be included, but the layer will be converted to a file geodatabase.
- Map packages let you share complete maps that include the symbology and data of the source map. By default, data in an Arc SDE geodatabase will be referenced rather than included. Consequently, anyone using the map package that uses data from an Arc SDE geodatabase will need to have access to that geodatabase. Alternatively, you can include the data sourced from an Arc SDE geodatabase in the map package by checking the Include Arc SDE geodatabase data check box when using the Package tool.
The data in layers that contain a join or participate in a relationship class will be consolidated into the map package. Likewise, when datasets reference other datasets (e.g., geometric networks, topology, locators), those participating datasets will also be packaged [7].

Raster data capability is included in map packages. Uncompressed rasters will be clipped based on the extent parameter specified in the Data Frame properties. Compressed rasters will not be clipped even if an extent has been specified.

### 2.6 Map Objects Java Edition

Mojo is a java toolkit for developers to work on server or client side of GIS or mapping applications. ESRI (Environmental System Research Institute) provides an API (Application Programming Interface) for ESRI map objects in java. We have many versions of map objects available for development like MOJ10, MOJ20 and MOJ23. So it is easy for a developer to build functionality using this MOJO API’s and also these API’s are in java, which makes it much easier to work [8, 9].

It is very convenient and easy for map objects to be combined with other vendors like database objects, multimedia and graphics for building applications for specific requirements of end users [9].

The following features can be achieved from MOJO API’s:

- Flexibility to add more layers for interactive map and also to remove layers. We can add more layers for displaying roads, streams and boundaries.
- It has pre built Zoom Pan Tool Bar and Selection Tool Bar, which gives power to access functionality in a detailed manner on maps.
- As API for drawing lines, polygons and points easily. Along with this, it has many True Type Fonts for rendering points which makes it possible to select symbols from these fonts and use them in a point theme in a place of the usual ‘base symbol’.
- To draw descriptive text [9].
- Easy to create shape files. Also, provide feature to save map as Arc XML file, which is actually ESRI Arc XML.
- Provides free set of mapping components with which it is easy to create our own customized GIS tools with more access to base objects.
- Allows us to add more toolbars and menu items using standard java classes, borrowing MOJO methods where needed [10].
2.7 JSSC JAVA SERIAL CONNECTOR

jSSC (Java Simple Serial Connector) - library for work with serial port from Java. The library was created as a simple and reliable replacement of existing facilities. Briefly we discuss about the opportunities it provides. With jSSC you can get the port names, read and write data, control lines RTS and DTR, receive Event etc. jSSC is designed to operate 24/7 multi-threaded systems and is currently successfully used in automation, data collection and recording. jSSC support Win32(Win98-Win8), Win64, Linux(x86, x86-64, ARM), Solaris(x86, x86-64), Mac OS X 10.5 and higher(x86, x86-64, PPC, PPC64) [11].

2.8 SHAPEFILE

2.8.1 Why Shape File?

Shape files contain many attributes for various geometries. This information is partly stored in database format. A geometry can be identified using vector coordinates in shape files. All the tables in this chapters have been taken from http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf [12].

It is not easy to process data directly from the shape file into any data structure; this has both advantages and disadvantages for developers. Advantage is it’s faster to draw then raster data, easily editable, takes less disk space, easy to read and write over other data sources. Disadvantage is that we need to write a program to process this dataset from a shape file, which makes a programmers life bit difficult. An overwhelming percentage of map layers are represented in shape file format [12].

Features of shape file are:

- They support geometries like polygon, line and point.
- Area features attribute is represented as closed loop, double digitized polygons.
- Attributes or values are stored in .dbf file format.
- Shape file follows one to one relationship for attribute and record in file.

2.8.2 How Shape Files are Created?

There are many ways to create shape files. But below are the usual method used to create them [12].

- Using software like BusinessMAP, Spatial Database Engine, ArcView GIS and QGIS ARC/INFO. We can export data source to shapefiles.
- Using ARCViw GIS creation tool feature, where we can easily create shapefiles by digitizing shapes.
- Using languages like AML (ARC Macro Language) or SML (Simple Marco Language) we can write programs to create shapefiles.
- Using CSV files and map objects.
- Using SelectionToolBar and mapobjects.

### 2.8.3 Shape File Technical Description

Shape file contains mainly three important files, which are:

1. Main file
2. Index file and

**DBase File:** Extension used for this file is ‘.dbf’. In this file there is a one to one relationship between attributes and record of geometry, and the attribute order maintained in this file should be same as main file [13].

**Index File:** Extension of this file is ‘.shx’. It contain offsets corresponding to records in main file from starting point.

**Main File:** Extension of this file is ‘.shp’. This file will have direct access, variable record length file in which each record describes a shape with a list of its vertices or endpoints for a particular geometry.

### 2.8.4 Naming Conventions

ESRI shape files follow 8.3 naming convention. All three files mentioned before should contain same prefix. This prefix can start with alphabet or numeric character i.e, (a-z,A-Z, 0-9), followed by 0 to 7 characters (a-z,A-Z,,-,_,0-9) including ‘_’ (underscore) and ‘-’ (dash). Suffix should follow the extension like ‘.shp’ for ‘main file’, ‘.shx’ for ‘index file’ and ‘.dbf’ for ‘dBase file’, and mainly all suffix should be in lower case letters [14].

Examples

- Main file: usa.shp
- Index file: usa.shx
- dBASE table: usa.dbf
2.8.5 Numeric Types

A shapefile contains double precision and integer numbers and text(String). More details regarding this can be obtained from link (http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf) [12].

- Integer: Signed 32-bit integer (4 bytes)
- Double: Signed 64-bit IEEE double-precision floating point number (8 bytes)

2.8.6 Organization of the Main File

Main shape file contains two parts:
1. Fixed length Header.
2. Variable length records.

As shown in Figure 2.3. Each variable length records contains fixed length record header followed by variable length record.

<table>
<thead>
<tr>
<th>File Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Header</td>
</tr>
<tr>
<td>Record Header</td>
</tr>
<tr>
<td>Record Header</td>
</tr>
<tr>
<td>Record Header</td>
</tr>
</tbody>
</table>

Figure 2.3. Organization of the Main File. Source: [12]

2.8.7 Byte Order

All the contents in a shapefile can be divided into two categories [12]:
1. Data related
   - Main file record contents
   - Main file header’s data description fields (Shape Type, Bounding Box, etc.)
2. File management related
   - File and record lengths
   - Record offsets, and so on
The integers and double-precision integers that make up the data description fields in the file header (identified below) and record contents in the main file are in little endian (PC or Intel®) byte order. The integers and double-precision floating point numbers that make up the rest of the file and file management are in big endian (Sun® or Motorola®) byte order [12].

2.8.8 The Main File Header

Length of main file header is of 100 bytes. We can explain main file using Table 2.1, which shows they byte order, type, values and byte position [12].

Table 2.1. Description of the Main File Header

<table>
<thead>
<tr>
<th>Position</th>
<th>Field</th>
<th>Value</th>
<th>Type</th>
<th>Byte Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>File Code</td>
<td>9994</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 4</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 8</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 12</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 16</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 20</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 24</td>
<td>File Length</td>
<td>File Length</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 28</td>
<td>Version</td>
<td>1000</td>
<td>Integer</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 32</td>
<td>Shape Type</td>
<td>Shape Type</td>
<td>Integer</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 36</td>
<td>Bounding Box</td>
<td>Xmin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 44</td>
<td>Bounding Box</td>
<td>Ymin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 52</td>
<td>Bounding Box</td>
<td>Xmax</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 60</td>
<td>Bounding Box</td>
<td>Ymax</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 68*</td>
<td>Bounding Box</td>
<td>Zmin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 76*</td>
<td>Bounding Box</td>
<td>Zmax</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 84*</td>
<td>Bounding Box</td>
<td>Mmin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 92*</td>
<td>Bounding Box</td>
<td>Mmax</td>
<td>Double</td>
<td>Little</td>
</tr>
</tbody>
</table>

* Unused, with value 0.0, if not Measured or Z type

Source: [12]
Below are the types of shape we can have in shape files.

<table>
<thead>
<tr>
<th>Value</th>
<th>Shape Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Null Shape</td>
</tr>
<tr>
<td>1</td>
<td>Point</td>
</tr>
<tr>
<td>3</td>
<td>PolyLine</td>
</tr>
<tr>
<td>5</td>
<td>Polygon</td>
</tr>
<tr>
<td>8</td>
<td>MultiPoint</td>
</tr>
<tr>
<td>11</td>
<td>PointZ</td>
</tr>
<tr>
<td>13</td>
<td>PolyLineZ</td>
</tr>
<tr>
<td>15</td>
<td>PolygonZ</td>
</tr>
<tr>
<td>18</td>
<td>MultiPointZ</td>
</tr>
<tr>
<td>21</td>
<td>PointM</td>
</tr>
<tr>
<td>23</td>
<td>PolyLineM</td>
</tr>
<tr>
<td>25</td>
<td>PolygonM</td>
</tr>
<tr>
<td>28</td>
<td>MultiPointM</td>
</tr>
<tr>
<td>31</td>
<td>MultiPatch</td>
</tr>
</tbody>
</table>

Figure 2.4. Shape Type. Source: [12]

As you can notice from Figure 2.4 shape types of 2,4,6…33 are not used. This is because, all the spots are reserved for future use like mixed shape types and allowing shape file to contain more than one type of shape type and so on. But for now, we are forced to use any one of above shape type in shape file [12].

2.8.9 Record Headers

The size of record headers is of 8 bytes. Table 2.2 shows byte order, type and value and byte position. In the table, position is with respect of the start of the record. This record header will also contain record number and content length for each record [12].
Table 2.2. Description of Main File Record Headers

<table>
<thead>
<tr>
<th>Position</th>
<th>Field</th>
<th>Value</th>
<th>Type</th>
<th>Byte Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>Record Number</td>
<td>Record Number</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 4</td>
<td>Content Length</td>
<td>Content Length</td>
<td>Integer</td>
<td>Big</td>
</tr>
</tbody>
</table>

Source: [12]

2.8.10 Main File Record Contents

This shape file record mainly consists of vector coordinates of a geometry and type of shape for provided data. We can calculate the total length of main file record using number of parts and vertices in given shape. So for every shape, we need shape followed by it’s mapping to record contents on disk [12].

2.8.11 Organization of the dBASE File

There are few points, which have to be followed in this dBase file [3].
1. Year value should be some year since 1900.
2. Record order in .dbf file should be same as that of main file .shp.
3. Tables can be joined on featured attributes.
4. Should have only one record per file.
5. Must have same name has main file.
6. Suffix should be .dbf

2.9 Eclipse

Eclipse is a free open-source Integrated Development Environment (IDE) which can be used for development in Java, PHP, C/C++, HTML5 and several other programming languages. Besides supporting multiple programming languages, NetBeans also supports various different platforms including Microsoft Windows, Linux, Mac OS X and Solaris depending on JVM compatibility [11].

With the IDE itself written in Java, it provides a range of software modules for simplifying the process of building mobile, desktop and web applications. Owing to this effective modular build approach, components developed on Eclipse can be easily modified and even extended later by third party programmers.
CHAPTER 3

REQUIREMENT

A good understanding of the requirements of an application before starting the development phase is essential. The detail and the extent of correctness of the requirements have a large role to play in the overall success of the project. This chapter brings to light the various requirements gathered for this application.

The requirements gathered for this tool are classified as follows:

- Data requirements.
- Platform requirements.
- Functional requirements.
- User Interface requirements.

3.1 DATA REQUIREMENTS

Data requirements for this comes from the user gestures. These gestures are read by the sensors and they are transmitted via serial connector to the ESRI package where this data is processed.

3.2 PLATFORM REQUIREMENTS

Platform requirements refer to the OS, development platform, the programing language and the other tools used in developing this tool. I chose to perform the analysis using java. Advantage of using java is it is platform independent and can run on almost all operating system with the support of java virtual machine.

3.3 FUNCTIONAL REQUIREMENTS

A functional requirement describes what a software system should do. This tool was designed based on the following functional requirements:
• The software must display the world map on startup.
• The software must display the points on the map. These points represent the 4 countries hosting the predicted data. Information about the predicted factors is displayed when clicking on the point using the hotlink tool.
• All the graphical features and fonts should be neat and clean.
• The software shall provide tools to zoom in and zoom out and zoom to full extent.
• There must be instructions provided to use the software.
• Hyperlink should allow the user to go to an external webpage to get more information.

3.4 USER INTERFACE REQUIREMENTS

The User Interface (UI) Requirements refer to the look and feel of the tool. The following describe the UI requirements:

• The tool should use easy to understand terminology and expressions.
• The tool should have high quality images. The web pages should be interactive and interesting and must appeal to the end users.
• The tool must be easy to use and navigating from one tab to another must be smooth and
CHAPTER 4

APPROACH AND IMPLEMENTATION

This section explains the various modules in the system and how they interact with each other. The different modules associated in this project are as given below:

- Arduino & Ultrasonic distance computer
- Serial Communicator (COM Port communicating)
- ESRI MOJO package integration
- Gesture Evaluation (Focus point adjustor)

4.1 ARCHITECTURE

For any project to be efficient architecture plays a key role. The architecture of this project plays a key role too Figure 4.1 shows different modules of the project and how they interact with each other.

![Figure 4.1. Architecture.](image)

4.2 DISTANCE MEASUREMENT USING ULTRASONIC SENSOR AND ARDUINO

Ultrasonic sensor is a distance measurement sensor which uses ultrasonic sound waves to measure distance. Ultrasonic sensor use high frequency sound waves of 40 KHz . Ultrasonic sensor consists of two basic modules, transmitter and receiver [4]. Transmitter
acts as speaker and receiver acts as a microphone. Speaker emits ultrasonic waves and
Microphone detect ultrasonic waves which are produced by speaker. Basic functionality of
ultrasonic sensor is shown in Figure 4.2:

![Ultrasonic sensor diagram](https://via.placeholder.com/150)

**Figure 4.2. Ultrasonic sensor. Source: [15]**

As shown in Figure 4.2 part is acting as speaker which emit ultrasonic waves which,
after collision with any object, return back. These returning waves are detected by
Microphone. The time taken by the microphone to receive ultrasonic waves from transmitter
after collision with any object is used to measure distance. I will discuss later how this time is
used to measure distance from any object. Ultrasonic sensor works on the same principle as
radio based radar system.

### 4.2.1 Applications of Ultrasonic Sensor

There are many application of ultrasonic sensor from domestic use to industrial use
[4]. Some of them are given below:

- Obstacle avoidance robot
- Robotics
- object detection
- distance measurement
- liquid level monitoring system
• Height measurement
• vehicle collision protection

4.3 DESIGN CONSIDERATION

Our system is designed based on four design considerations:

1. Automatically detect gesture boundaries: A common challenge of gesture recognition is the uncertainty of when does a gesture begins or ends. We do not require a user to press a key to indicate the presence of a gesture since it would be inconvenient to do so.

2. Recognition must be real-time: Gesture interface must be very responsive, so no time-consuming post processing is allowed.

3. False alarms need to be minimized: Executing a wrong command is generally worse than missing a command.

4. No user-dependent model training process for new users: Although supervised learning can optimize the performance for a specific user, collecting training data can be time consuming and not desirable for users.

![Figure 4.3. Data Flow.](image)

4.3.1 Proximity Sensor Data Acquisition

We now describe each system component shown in Figure 4.3. The proximity sensor consists of two IR LEDs and a IR receiver, which are placed underneath a plastic/glass screen surface, surrounded by optical barriers. The LEDs emit IR strobos in turn as two
separate channels using time-division multiplexing. When a hand or any object is near, the receiver detects the reflection of the IR light, whose intensity increases as the object distance decreases. The light intensities of the two IR channels are sampled by the firmware at 100Hz [16].

4.3.2 Framing
Since the start and end of a gesture is not specified by the user, our program uses a moving window to scan the input IR intensity data and decide if any gesture signature is observed. The data is divided into 50% overlapping frames, each of which is 140 ms. After framing, three types of feature are extracted from each frame.

4.3.3 Infrared Feature Extraction
The feature measures the pair-wise time delay between the sensor data of two channels, which shows how a hand approaches the IR LEDs at different instants. This corresponds to different moving directions of the hand.

4.4 Recognition Performance
We use the widely used precision/recall metric to evaluate the recognition performance:

precision = \( \frac{TP}{TP + FP} \) (4.1)

recall = \( \frac{TP}{TP + FN} \) (4.2)

where TP, FP, FN refer to true positive, false positive, and false negative. The system achieved 98% precision in average, and is robust from user to user. The high precision implies low false alarm rate, which is ideal for gesture recognition because executing a wrong command is usually worse than missing a command. The recall rate is lower than precision because the system can miss gestures when the hand is too far from the sensor, or when a gesture is performed much slower than usual [17].

4.5 User and System Factors
We further design two experiments on user and system factors to evaluate the robustness and limitation of the system. User-to-Device Distance first, we evaluate the
influence of user-to-device distance on the system performance. The distance is measured from the user’s hand to the proximity sensors. The system can achieve over 80% accuracy when the user’s hand is within 3 inches. The effective range can be increased by increasing the power of IR LEDs, with a tradeoff of a higher power consumption. One can balance the tradeoff according to the system needs on user experience and battery life. Speed of Gesture Next, we evaluate the system performance when the user perform gestures at different speeds. In this experiment, the user listens to a specific tempo given by an electronic metronome; the first beat “tic” indicates the start of a gesture, and the second beat “toc” indicates the end of a gesture. According to our observation, most users naturally make gestures at the speed of 2 to 4 gestures per second. In other words, it usually take 0.5 to 0.25 seconds for general users to complete a gesture. The system achieves over 90% accuracy at general gesture speeds, and also maintains a robust performance of over 80% at very slow (1 gesture per second) or very fast (5 gestures per second) gesture speeds.

4.6 SERIAL COMMUNICATION

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output [2].

Arduino environment's has a built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

The Arduino Mega has three additional serial ports: Serial1 on pins 19 (RX) and 18 (TX), Serial2 on pins 17 (RX) and 16 (TX), Serial3 on pins 15 (RX) and 14 (TX). To use these pins to communicate with your personal computer, you will need an additional USB-to-serial adaptor, as they are not connected to the Mega's USB-to-serial adaptor. To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your Mega to your device's ground. (Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.)
The Arduino Due has three additional 3.3V TTL serial ports: Serial1 on pins 19 (RX) and 18 (TX); Serial2 on pins 17 (RX) and 16 (TX), Serial3 on pins 15 (RX) and 14 (TX). Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip, which is connected to the USB debug port. Additionally, there is a native USB-serial port on the SAM3X chip, Serial USB.

![Figure 4.4. Serial Monitor.](image)

The Arduino IDE has a feature that can be a great help in debugging sketches or controlling Arduino from your computer's keyboard.

The Serial Monitor is a separate pop-up window that acts as a separate terminal that communicates by receiving and sending Serial Data. See the Figure 4.4 [4].

Serial Data is sent over a single wire (but usually travels over USB in our case) and consists of a series of 1's and 0's sent over the wire. Data can be sent in both directions (In our case on two wires).
4.7 ESRI Package

Esri's GIS (geographic information systems) mapping software helps you understand and visualize data to make decisions based on the best information [8]. The default map package obtained from the ESRI package looks as shown in Figure 4.5.

![Default map package](image)

**Figure 4.5. Default map package.**

Map Objects—Java is a suite of Java-based developer components for creating client- or server-side mapping and GIS applications. Map Objects—Java contains a collection of Java Archive (JAR) files that can be used to develop custom Java-based solutions. Version 2.0 of Map Objects—Java will also begin to extend into the server side of the client-server model, featuring the notion of a new kind of server-side layout component and map component. The core libraries provide a rich resource of GIS and mapping components built completely on the Java 2 platform. They can be used to integrate GIS and mapping into existing Java systems, for instance, or to leverage your Arc IMS services through thick clients. These Map Objects—Java 2.0 server-side objects have the ability to live within servlet containers, as well as the ability to be wrapped into Enterprise Java Beans (EJBs) that operate with the J2EE container.
Figure 4.6. Class Diagram of the Package Interfaces and Classes. Source: [18]
Provides components for developing server-side applications that could be run within any J2EE server or as a standalone application.

This package contains implementations of Map interface and its supporting classes that can be used to develop server-side mapping applications and a Layout class that could be used to develop server-side Layout applications. Contrary to the client-side Map implementation, these server-side components are single threaded so that they could be used to develop Enterprise JavaBeans to run in any J2EE-compliant application servers.
CHAPTER 5

GESTURE RECOGNITION

Gesture recognition is an upcoming technology which can modified and technology and our work in daily life. Gesture means signs made by human beings which originate from face, hands or any part of the body. These gestures can be captured using scanning or video methods and by processing these gestures in to human signals.

GUI related interfaces are used in which text is taken as input from mouse and keyboard. In this new system gestures are used as inputs which do not require any mechanical elements to communicate between man and machine. If we move our hand on computer screen and based on our movement, cursor will accordingly move which will make work easier. But now with increase in technological knowledge, the concept slowly enhanced into voice, speech recognition and position recognition models.

These have enriched the domain and have roped in some very sophisticated means of human computer interaction. Finger tracking is one such advanced gesture innovation. It is the use of hands and their various positions to kick-start a computer application. It aims at minimizing the use of keyboard and mouse. Non-touch based interaction or giving the input to computers with eyes is one major breakthrough in the domain. It can certainly be adjudged as the ray of hope for disabled people or people busy with multitasking.

5.1 GESTURE WITH ESRI PACKAGE

This project concentrates on embedding this feature into GIS which helps disabled users to interact with the tool in effective way. Ultrasonic sensors are one of the most effective proximity computing devices which uses speed of sound pulse to compute the distance between the transmitter and object. Arduino open source electronic help us to process the data from these sensors and help transmit these data in to our GIS system where the data is used to evaluate gestures made by the user.
5.1 Gesture Edit Mode

Gesture edit mode is an additional feature that this project has included in the package. It helps the user to edit the field of operation to various regions. This feature helps to enhance the gesture feature.

5.1.1 What is Field of Operation?

Field of operation is the region in space that allows the user to use the gestures. Basically it’s a region where the user must reach out to make input to the system. This gives flexibility to make appropriate input to the system. Figure 5.2 shows how it works.

5.1.2 Gesture Mode

This is the mode where it triggers the communication between the sensors and the Esri package. Once the communication is established, the user has control over the field of operation.
operation explained in above paragraph. On successful communication user can start using gesture to control the scaling of the map. Figure 5.3 shows the button to get into the gesture mode.

Figure 5.3. Gesture Mode.

Once the user enters the gesture mode we can display the value read by the sensor on developer instance Figure 5.4 shows how the developer console looks like during the gesture mode.

Figure 5.4. Gesture developer console.

5.2 EVENT OPERATION

The system is programmed in such a way that it takes the input from the user and triggers an event during the gesture mode. The work flow and snap shot of the system is represented in Figure 5.5. Figure 5.5 shows how is it achieved.
Figure 5.5 explains how the work flow is designed and how the listener function is implemented the Pseudo-code shows how is it done.

The serial event function reads the data from the COM port and gives it to the Gesture evaluation function. Additional responsibilities of this function are to evaluate the field of operation the value which is given by the variable “n”, and it also specifies the scaling of the map.

Conditional Statements in the code checks if any event has taken place, if yes its starts the process.

This means this function executes only if an event is triggered.
/**
 * Handle an event on the serial port. Read the data and print it.
 */
public synchronized void serialEvent(SerialPortEvent oEvent) {
    try {
        inputLine = input.readLine();
        System.out.println(inputLine);
        int n = Integer.parseInt(inputLine);
        System.out.println("n is"+n);
        // m.zoom(n);

        if( n>1 && n<10)
            z=0.2;
        else if( n>10 && n<20)
            z=0.4;
        else if( n>20 && n<30)
            z=0.6;
        else if(n>30 && n<40)
            z=0.8;
        else if(n>40 && n<50)
            z=1;
        else if( n>50 && n<60)
            z=4;
        else if( n>60 && n<70)
            z=7;

    public void actionPerformed(ActionEvent ae)
    {
        Object source = ae.getSource();

        if (source == hotjb)
        {
            hotlink.setCursor(boltCursor);
            map.setSelectedTool(hotlink);
        }
        else if(source == pointer)
        {
            map.setSelectedTool(arrow);
        }

        else if(source == gesture)
        {
            System.out.println("i am the gesture listener");
            try {
                SerialTest.test(map,env);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            map.setSelectedTool(hotlink);
        }
    }

Figure 5.6. Pseudo – Code.
CHAPTER 6

OUTPUT

The following chapter shows how different options are rendered and how they acutely look in the live environment.

![Figure 6.1. Gesture Edit Mode.](image)

Names of the icons are

- Gesture edit mode
- Gesture mode
- Hot link mode
- Default selection mod
- Zoom +
- Zoom –
- Pan tool bar
Figure 6.2. Gesture mode 1x Scaling.

Figure 6.2. shows initial setup of the project.

- Run the Mysql service in the backend
- Load the project
- Check the COM port connectivity
- Run project

On successful connection and execution, the project renders the above display.
Figure 6.3. Scaling Is 0.5x.

Figure 6.3 shows scaling with 0.5x

The system is in the gesture mode and the user now can make gesture to control the scaling of the map.

6.1 INITIAL SETUP

- Verify the connectivity
- Make different gestures to check for appropriate scaling
6.2 LOGGER FUNCTION

Logger function is used for debugging purpose, and Figure 6.4 shows how the serial in can be logged and event triggers can be logged.

N is the distance between hand and sensor
Check point is the name of the event.
Figure 6.5 shows scaling with 3x (Maximum Scaling).

The system is in the gesture mode and the user now can make gestures to control the scaling of the map.

### 6.3 Initial Setup

- Verify the connectivity
- Make different gestures to check for appropriate scaling

Figure 6.6. Scaling 0.3X.

Figure 6.6 shows scaling with 0.3x (Minimum scaling).

The system is in the gesture mode and the user now can make gestures to control the scaling of the map.

### 6.4 Initial Setup

- Verify the connectivity
- Make different gestures to check for appropriate scaling
Figure 6.7. Connection Not Found.

Figure 6.7 shows how the connection for the port is authenticated.

Steps involved:
- Get the available COM port
- Ping the COM port for data
- Display exception is COM not found
Figure 6.8. Event Triggered.

Figure 6.8. explains how the event thread works in gesture mode

- Events can be triggered during gesture mode
- Proof of concept to show how two different action can be handled.
CHAPTER 7

CONCLUSION

Gesture recognition using proximity sensors is a project that focuses on using proximity sensors to achieve gesture recognition and a new way of interaction with external devices. This project emphasizes on the effective usage of proximity sensors and explains the architecture and different modules of the project, using the legacy tool ESRI map package. ESRI package is one of the widely used geographical software package widely used in geographical applications such as map-based application and geographical systems.
CHAPTER 8

FUTURE ENHANCEMENT

This tool provides a great framework, which can be expanded to add more features and functionalities. Since gestures are new way of interacting to devices and advancement in this field gives lot of flexibility in utilizing different kind of sensors and effectively bringing data into the system.

Following are some of the future enhancements for this application.

- Deploying the application on the web as browser hosted applet, so that it can be used without installing it on each machine.
- Application can be made for mobile devices and tablets.
- More customized tools can be added as per the requirement.
- Embedded to Operating system level. Which lets us use application in better way.
BIBLIOGRAPHY

WORKS CITED


**WORKS CONSULTED**