II. REQUIREMENTS

For the system to interpret the gestures, it had to be passed through the following processes:

A. Detecting a Touch

Since the touch screen was a resistive touch, which depended on the mechanical force exerted on it; the system was to be designed to sense the contact between the top and bottom layer of the touch screen within a time interval.

B. Detect when the Finger was in Motion

Since motion is a function of distance travelled and time [6], that is it has to do with the change of the position of an object from a reference point to another within a time interval (Δt). The system was to be designed to check if there was a change in the position of the user’s finger along both the X and Y axis (Cartesian coordinates), within a time interval (Δt). This way it could detect if there was movement and the possible distance travelled.

C. Detecting the Direction of the Finger Movement

After a touch and the motion of the touch had been detected, the next step was to determine the direction the finger was moved. As earlier stated, the angular displacement
of a point of contact from a previous reference point of contact was used within a predefined time interval was used. The angle of displacement used was the angle in reference to the positive Y axis, following a clockwise movement round the centre point as shown in Fig. 2. This was because in analyzing the angle, a 360° degrees round the centre point was needed; and since the Y axis was used as the forward/backward movement for the system, this made it to have the 0° degrees mark if the sensed movement was along the positive Y axis only. So since the Y axis was 0°, a 90° to the right (a straight line along the X axis alone) was to be a right movement and 90° to the left (270°) would mean a left movement. As shown below in Fig. 1, based on the point readings taken along the X and Y axis, the system could calculate the distance and angle from point P(x,y) to Q(x,y) to R(x,y) (θ₁ and θ₂) respectively.

\[ PQ = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2} \]

And the angle of displacement from point P to Q will be (since the movement was in the first quadrant (Fig. 2).

\[ \theta_1 = \cos^{-1} \left( \frac{Y1 - Y2}{PQ} \right) \]

where, PQ was the calculated distance between points P and Q. The gestures the system was developed to interpret can be seen below:

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Code Outputted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fn:</td>
<td>where “n” depends on distance moved by the finger</td>
</tr>
<tr>
<td>Fn&gt;:</td>
<td>where “n” depends on distance moved by the finger</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Bn&gt;:</td>
<td>where “n” depends on distance moved by the finger</td>
</tr>
<tr>
<td>Bn:</td>
<td>where “n” depends on distance moved by the finger</td>
</tr>
<tr>
<td>Bn&lt;:</td>
<td>where “n” depends on distance moved by the finger</td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>Fn&lt;:</td>
<td>where “n” depends on distance moved by the finger</td>
</tr>
</tbody>
</table>

By using Pythagoras’s theorem and trigonometric functions (Fig. 1), the distance between points PQ and QR could be calculated and the angular displacements. For example to calculate the distance between points PQ using Pythagoras’s theorem:

III. IMPLEMENTATION

In this project, we proposed a reliable means via which gestures made by the finger on a touch screen, can be adequately interpreted to control the movement of a mobile robot. The implementation of the project can be seen below:

A. Hardware

A block diagram of physical architecture of the system can be seen in Fig. 3. It consisted of the microcontroller, a resistive touch screen, a wireless digital link working on the
433MHz range, and finally the mobile robot which had been built to be used in testing the system.

**B. Touch Screen**

For this project, a five wire resistive touch screen which uses a mechanical contact to output an analogue voltage was selected. This was used because it was the most stable among the different types (4, 8 wire and so on), thereby making it less susceptible to temperature changes [7]. The position of the touch point was determined by the outputted analogue voltages from it; representing the position of the finger along the X and Y axis depending on configuration.

**C. Micro Controller**

A standalone Microchip® microcontroller (PIC18F2520) was used; it also had a lot of inbuilt peripherals needed to execute the project like the Analogue to Digital converter (ADC), power conservation mode, inbuilt oscillator (8MHz max), Universal Synchronous Asynchronous Receive Transmit (USART), Low voltage power capabilities, internal EEPROM (256 bytes) [8] and so on.

**D. MOSFET Drivers**

When the current drawn by the touch screen was measured independently with a supply voltage of 5 volts, it was noticed it drew a current more than what the PIC18F2520 microcontroller could supply which was limited to 25mA [8]. For this reason a MOSFET driver which could deliver enough current had to be used. The Microchip® TC4468 Quad AND MOSFET driver was selected. This was because it could supply a current of about 300mA from each of its pins, and 500mA from the total chip as a whole [8].

**E. USB to Serial Converter**

It was required that the serial protocol be used, for the microcontroller to communicate with the PC for testing purposes, and especially when giving calibration instructions to the user; but in most systems this port does not exist. Also on the other hand, it was desired that the converter be able to draw power from the USB host; for this will help in reducing the amount of power drawn from the system’s independent power supply. The MM232R USB to serial converter was chosen to be used for the task [10].

**F. Radio Frequency Transceiver Module**

The mobile robot in Fig. 3.3 was to be controlled over a wireless link and therefore an RF transceiver module was needed to carry out the linking function. An RF transceiver is an electronic module capable of both receiving and transmitting. The ER400TRS was selected as the transceiver module to be used in setting up the link, as it could cover up to 10m in distance [9], and it operated within the licence-free frequency range of 433MHz. Lastly it could also support serial communication to and from the microcontroller.

**G. Mobile Robot**

A pre-constructed mobile robot which could be controlled using the predefined codes and functions shown in Table I sent over the wireless channel was selected (Fig. 5). The mobile robot’s direction and distance of movement could be controlled but not its speed. The mobile robot used can be seen below:

**H. Software**

After the hardware had been developed, the software which was to be programmed into the microcontroller was then developed. The flowchart diagram of the system, upon which the software was built, can be seen in Fig. 6.
The software was developed using a mixture of both the C and assembler language, to aid in code efficiency; and the algorithm for the software that was implemented can be broken down into different tasks as outlined below:

A. Touch detection 
B. Configuration of the Sampled Axis 
C. Conversion of the Analogue Voltage into Digital 
D. System Calibration 
E. Reading of the Touch Point 
F. Finger Motion and Gesture Analysis 
G. Outputting of the Control Codes based on gestures

I. Touch Detection

This task was designed to be executed continuously, and was used to check if the touch screen was being touched by the user. Kit was a task which returned a binary value of either “1” or “0” representing it being touched or not respectively.

J. Configuration of the Sampled Axis

This task was used to configure the touch screen to take the required axis to be read; since the different axis (y and x) could only be read one at a time. A table can be seen below, showing the pin configuration used based on the touch screen’s datasheet for the X and Y axis value to be taken:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Axis</th>
<th>Y axis</th>
<th>X axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper left</td>
<td>Low</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Lower left</td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Upper right</td>
<td>Low</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Lower right</td>
<td>High</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

K. Conversion of the Analogue Voltage into Digital Values

This was a task designed to be executed continuously as long as a touch was detected. It was used to check the point of contact along the X and Y axis. To achieve this; the pin used to sense the analogue voltage was converted to an analogue input then three readings of the inputted analogue voltage was taken and then added together. Finally the average of these readings was taken and the value stored.

L. System Calibration

This was a task designed only to be executed if at start up or system reset, the touch screen was being touched by the user. It was used to calibrate the system to ensure that the touch points were properly read and analysed. It was done by first checking if the touch screen was being touched, and if touched give instructions to the user on the points to touch as shown in Fig. 3.5 below. It then calculated the “slope” of the touch screen's outputted data based on the linearity of the screen by using the formula shown below in Fig. 7. Next the data obtained was then stored in the system’s EEPROM for usage.

\[
X \text{ axis slope} = \frac{X_{\text{max}}-X_{\text{min}}}{\text{Length of X axis (obtained from the touch screen)}}
\]

\[
Y \text{ axis slope} = \frac{Y_{\text{max}}-Y_{\text{min}}}{\text{Length of Y axis (obtained from the touch screen)}}
\]
M. Reading of the Touch Point

This task was responsible for analyzing the touch point, and then outputting the touch point in millimetres which was to be used for the touch analysis. Once a touch was detected the system then read the analogue voltage sensed on each of the axis depending on the one configured to be read.

N. Finger Motion and Gesture Analysis

This task was used to detect when a motion was made on the touch screen. To compensate for any possible errors, the system ensured a minimum distance was moved along the X or Y axis before the motion analysis was done. In other to recognise if a motion had been properly made across the touch screen, the system was designed to take samples every 50mSecs; then based on the read samples, the system could determine the magnitude of the motion. Next the system analysed the gestures made on it using the displacement algorithm method (as outlined above) coupled with pattern spotting algorithm.

The pattern spotting algorithm worked by analyzing the “shape” of the obtained gesture based on the various angles of the and then

O. Outputting of the Control Codes based on Gestures

After the gestures have been analyzed, the next step was to send out the required code as specified in the Table I. A look-up table was used to check what code was to be sent out, for a particular gesture. The look-up table was designed using arrays which have the ASCII representatives, and addressing them to be sent out when needed. A screen shot can be seen below, showing the codes that can be sent over the serial port when the gestures were being interpreted in Fig. 8.

IV. Results

This section of the paper presents the obtained results of the project after it was completed. The various gestures as outlined in Table I was tested and the output were studied to check if the right codes were transmitted. The results from the ISIS simulation of the hardware design can be seen below via a screen dump in Fig. 10. A test code written to output data via the virtual terminals depending on the inputted signal was used in the test.

The system was tested to be able to properly analyze the angle of movement of the user’s finger across the touch screen from point to point and made to output its analyzed angle via the COM port. A screen dump below showing what was outputted when the user’s finger was on a spot, moved left and right along the X axis in Fig. 11.

A snapshot of the hyper terminal window, which was used to view the transmitted data between both the touch screen and the robot can be seen below in Fig. 12. The mobile robot was capable of sending over the air how close it was to the nearest object at its left, right and centre.
V. CONCLUSION

A two dimensional touch interface, which is capable of controlling a remote mobile robot based on finger gestures made over the interface, was developed in this paper. The use of a resistive touch screen made it a two dimensional system; then by using the angular displacement method the system could adequately determine the movement of the finger within a predefined time interval. The combination of the angular displacement technique used in capturing then pattern spotting for analyzing the gesture was well suited for the system. The intuitiveness of the control system by using gestures was tested across 20 different users, with 95% of them learning to control the robot in less than 30 seconds. This method allowed for an effective and efficient way of interpreting the gestures, using them in controlling a mobile robot.

ACKNOWLEDGMENT

The authors express thanks to their parents whose continuous support, inspiration and love facilitated them reaching this stage successfully. O. Ejale and N. B. Siddique are intensely grateful to their supervisor Dr. Richard Seals from the University of Greenwich whose proper guidance, suggestion and inspiration in time made them accomplish this project successfully. They would also like to have the privilege of giving their heartiest thanks to all the helpful lab technicians specially Mr. Dodson and all the staffs of School of Engineering of the University of Greenwich for their kind cooperation. N.B. Siddique would also like to thank his beloved fiancée Dr. Fariha Hussain for performing a careful proof reading of this paper.

REFERENCES


[8] Microchip©, PIC18F2420/2520/4420/4520 Data Sheet (DS39631E), Microchip, 2008, pg. 1

