Annals of Emerging Technologies in Computing (AETiC)
Vol. 3, No. 1, 2019

Research Article

Pattern Recognition using a Neural Network on a Microcontroller with I2C Ultrasonic Sensors

Fazal Noor

Islamic University of Madinah, Madinah, Saudi Arabia
mfnoor@gmail.com

Received: 28th November 2018; Accepted: 18th December 2018; Published: 1st January 2019

Abstract: Ultrasonic sensors have been used in a variety of applications to measure ranges to objects. Hand gestures via ultrasonic sensors form unique motion patterns for controls. In this research, patterns formed by placing a set of objects in a grid of cells are used for control purposes. A neural network algorithm is implemented on a microcontroller which takes in range signals as inputs read from ultrasonic sensors and classifies them in one of four classes. The neural network is then trained to classify patterns based on objects' locations in real-time. The testing of the neural network for pattern recognition is performed on a testbed consisting of Inter-Integrated Circuit (I2C) ultrasonic sensors and a microcontroller. The performance of the proposed model is presented and it is observed the model is highly scalable, accurate, robust and reliable for applications requiring high accuracy such as in robotics and artificial intelligence.

Keywords: ***Ultrasonic Sensors; Time of Flight; Echo; Transmitter; Receiver; I2C; Neural Network; Pattern Recognition; SONAR***

1. Introduction

Hand gestures are used in a variety of applications for control purposes such as, turning devices on and off (e.g. lamps), opening and closing (e.g. doors or windows), controlling the operation of computers, playing musical sounds and many other devices [1]. Different hand gestures (different motions) form unique patterns and are used in conjunction with ultrasonic sensors to form a variety of controls. There are different types of sensors available in the market to measure range values to objects. Infrared, LIDAR, RADAR and Ultrasonic sensors are the ones most commonly used in numerous applications. Ultrasonic sensors are easily available in the market and are cheap, however, they have poor resolution compared with LIDAR. LIDAR is expensive compared to the others, but provides a very fine beam and has an excellent resolution [2].

Like hand gestures, placement of a set of objects in a grid of cells creates a pattern which may be used to provide robust and reliable unique codes. This will be called objects' position-based gesture (OPBG) in this paper. As ultrasonic sensors are used in hand gesture 'control, likewise, they can be used to generate unique codes from measuring distances of objects' placements. These OPBG codes then may be used for controls. Depending on the accuracy and precision of a sensor's readings one may devise many classes (clusters of range readings). Using multiple ultrasonic sensors in parallel can form a grid of cells. Objects placed in cells of a grid can form different patterns and hence unique codes. Since an

Fazal Noor, "Pattern Recognition using a Neural Network on a Microcontroller with I2C Ultrasonic Sensors", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp 9-19, Vol. 3, No. 1, 1st January 2019, Published by International Association of Educators and Researchers (IAER), DOI: 10.33166/AETiC.2019.01.002, Available: http://aetic.theiaer.org/archive/v3/v3n1/p2.html.

object may be placed anywhere in a cell, sensor range readings will be different for various positions. In other words, a cluster of sensor range readings of an object in a particular cell will form a class. The problem that arises here is the separation of clusters (range values) into cells and their boundaries. The solution presented is by using Neural networks. Neural networks have been used in many fields and have provided good results. A Neural network is first trained with clusters of range data, then validation and testing are done and loss performance is measured. In this work, a testbed is formed by implementing a neural network on an Arduino microcontroller with Inter-Integrated Circuit (I2C) ultrasonic sensors used for range readings to the objects. The OPBG system proposed in this paper can be augmented by hand gestures and provide a robust gesture recognition system.

2. Literature Review

There are variety of range measuring sensors based on ultrasonic, infrared, RADAR and LiDAR technologies. One company named Chirp Microsystems founded in 2013 is producing miniature ultrasonic sensors (millimeter in size) that can calculate range to objects very precisely (approx. 16 feet) and are used to read hand gestures, track movements of the eye and track people in a room [15]. Their sensors operate on 1.8 V supply and have I2C interface [15]. Sensors based on ultrasound are being used in smartphones and variety of other devices such as autonomous cars and robots in warehouses to monitor surrounding areas [15]. In [16], hand gesture recognition is performed using Deep Learning approach. They use Time of Flight (ToF) for mid-air hand gesture recognition on mobile devices [16]. In [17], micro hand gesture recognition system using ultrasonic sensing is proposed. The hand gestures pattern recognition is performed by machine learning and achieve accuracy in range of 90% to 96.32% [17].

In previous work [3], a study was performed on an object's width and gap measurements and gap detection between objects using ultrasonic sensors. Ultrasonic sensors have good range precision but poor angular resolution. It was seen the performance depends on sensor type, sensor capabilities and limitations. This work uses the results in [3] to place ultrasonic sensors [4] with optimum gaps in between such that no overlap and no crosstalk occurs. Crosstalk is undesired and occurs when more than one sensor transmits a signal and the echoes of one sensor are received by the other sensors. Figures 1 and 2 show how angular and distance resolution decrease when a sensor is placed further away. A gap between objects is not distinguishable when a sensor is placed far from it. Also, in [3] sensitivity to an object's face angle was studied and was seen to give false readings when an object was not placed perpendicular to the sensor. In this work, a testbed setup was formed and care was taken for both sensors' and objects' placements as to minimize false readings due to either crosstalk or angles.

Inspired by the human brain, Warran S. McCulloch and Walter Pitts in 1943 developed a model of an artificial neural neuron with inputs, processor and generation of an output [5]. Pattern recognition such as character recognition or handwritten or facial recognition are few common applications of artificial neural networks[6, 10]. Applications of neural networks are many such as in aerospace, automobile, military, electronics, financial, industrial, medical, speech, telecommunications, transportation, software, control, signal processing and others. In general, a neural network is an adaptive system where its internal structure consists of neurons (having activation functions) connecting to other neurons by links and its weights are adjusted based on error performance at the output. Once the desired output behavior is achieved the neural system is said to have been trained and ready for testing it on unseen input data.

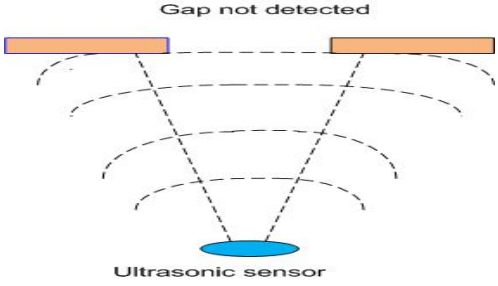

Figure 1. Gap between 2 objects not detected

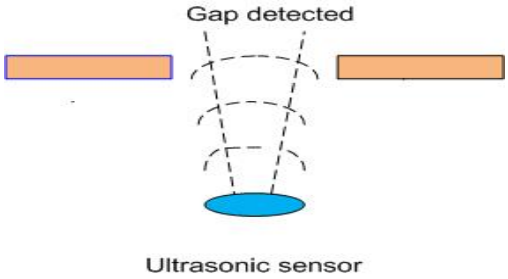

Figure 2. Gap between 2 objects detected

3. Methods

3.1 Neural Networks

A Neural network is based on mimicking the brain cells. The brain of a human has a complex interconnection of nerve cells called neurons and a human brain has on the order of 1011 neurons [3]. A neuron connects to approximately ten of thousands of other neurons by axons. The figure 3 below shows dendrites, neurons and axons. The dendrites receive signals from the sensory organs as electric impulses which are processed by neurons and which in turn send messages to other neurons through connections called axons [3].

In a Neural network the neuron node has an activation function which is evaluated and output is passed to a neuron connected to it. A weight is associated with each link.

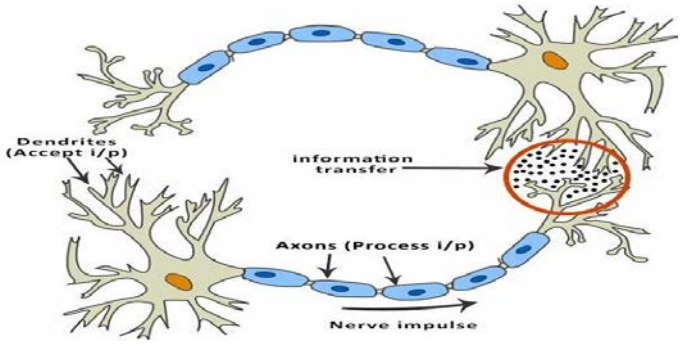

Figure 3. A Typical Brain Cell.

There are 2 types of Neural networks topologies, one is the feedforward and the other is with feedback. In a feedforward Neural network the flow is one way and there is no feedback loop. The feedforward Neural networks are useful in pattern generation, pattern classification and pattern recognition problems.

Two neurons are connected with an arrow indicating the flow of information. A weight is assigned to each connection or link and a function is assigned to each neuron. The network weights are adjusted incrementally to produce a desired output. There are several methods for neural networks to learn, namely, supervised learning, unsupervised learning and reinforcement learning. In supervised learning a teacher guides the NN with the right answers and corrections are made by adjusting the weights depending on a loss function. In unsupervised learning, there is no teacher to guide with an example data set. In reinforcement learning, the ANN observes the environment and makes a decision and adjusts the weights if the observation is incorrect.

Back propagation method is a feedforward with feedback learning algorithm and learns using examples and adjusts the weights according to derivatives of a loss function [7]. The back propagation is useful in applications requiring pattern recognition and mapping tasks [8]. The algorithm is summarized below in algorithm 1 and the block diagram is presented in figure 4.

Algorithm 1. Neural Network with Back Propagation

1. Initialization; Set weights matrix to random numbers.
2. Forward propagate by using weights and activation functions.

3. Error Function: Compute the error function sum of squares of absolute error
4. Optimize weights using differentiation of the loss error function or use optimization technique.
5. Perform back-propagation of errors to the hidden layers.
6. Update the weights.
7. Iterate steps 2 to 6 until convergence

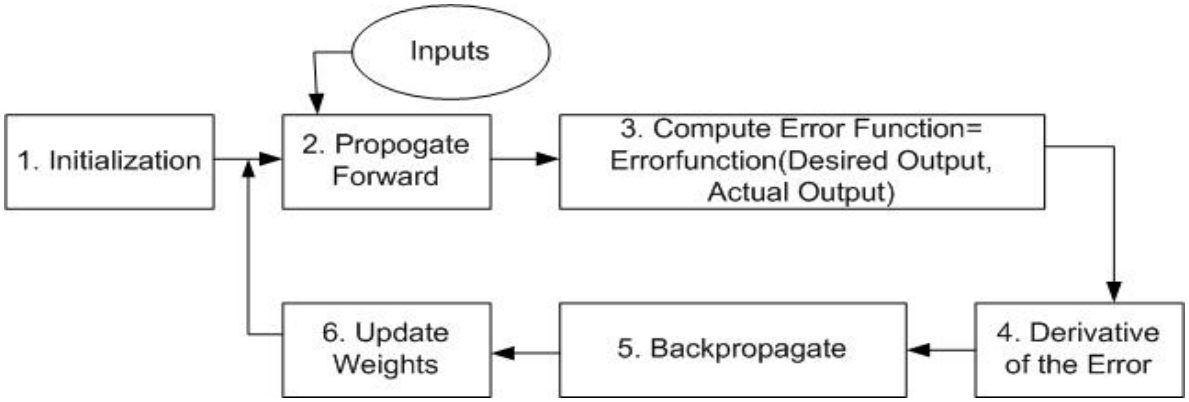

Figure 4. Feedforward with backpropagation block diagram.

The main problem is to design a neural network for classifications (having extremely low errors) of an object placed at an arbitrary point in a column of 2-dimensional space into one of four classes. Since ultrasonic sensors are used to find the range to the object, it is assumed that the object is placed perpendicular to the sensor beam thus forming a narrow column of 4 classes (row 1 to row 4). In other words, a grid of 4 cells in a column shape are formed as shown in figure 9. The number of cells in a column formed is arbitrary, however in this study 4 cells are chosen, 1 for each class. One class represents a cluster of sensor range values. The cell closest to the sensor represents class 1 and the furthest one represents class 4. Each cell or class will represent one bit of the code. Four sensors placed in a linear fashion will produce a total of 4^4 = 256 unique patterns or codes.

For the training of the Neural network (NN), 4 clusters of data per sensor are used. The NN consists of 2 inputs per sensor (i.e. x1, x2 where x1 is x value and x2 is the range value measured by a sensor), 4 neurons in the hidden layer and 4 outputs (classes).

Let X_{ij}^{L} represent the input, where L is the layer number, i represents data and j represents input.

$$X^1 = \begin{bmatrix} X_{11}^1 & X_{12}^1 \\ \vdots & \vdots \\ X_{N1}^1 & X_{N2}^1 \end{bmatrix} \quad (1)$$

Let W^1 represent the weight matrix at layer 1 where the notation W_{qr} represents the link weight from node q to node r.

$$W^1 = \begin{bmatrix} W_{11} & \cdots & W_{14} \\ \vdots & \ddots & \vdots \\ W_{21} & \cdots & W_{24} \end{bmatrix} \quad (2)$$

Let V be the value matrix obtained by multiplying the inputs X by the weights W matrix, $V_1^1 = XW$

$$V_1^1 = [Xi1W11 + Xi1W21 \quad Xi2W12 + Xi1W22 \quad Xi1W13 + Xi2W23 \quad Xi1W14 + Xi2W24] \quad (3)$$

Let F(v) be the function matrix, where the sigmoid function is used at each node, $f(v) = 1/(1 + e^{-v})$ [9]

$$F^1(v) = \begin{bmatrix} f_1(v1) & f_1(v2) & \cdots & f_1(v4) \\ & \vdots & \ddots & \vdots \\ f_N(v1) & f_N(v2) & \cdots & f_N(v4) \end{bmatrix} \quad (4)$$

Let Y^2 be the output obtained by function matrix F(v) multiplied by the weight matrix W^2 at layer 2.

$$Y^2 = F^2(v)\, W^2 = \begin{bmatrix} Y_{11}^2 & Y_{12}^2 & \cdots & Y_{14}^2 \\ & \vdots & \ddots & \vdots \\ Y_{N1}^2 & Y_{N2}^2 & \cdots & Y_{N4}^2 \end{bmatrix} \quad (5)$$

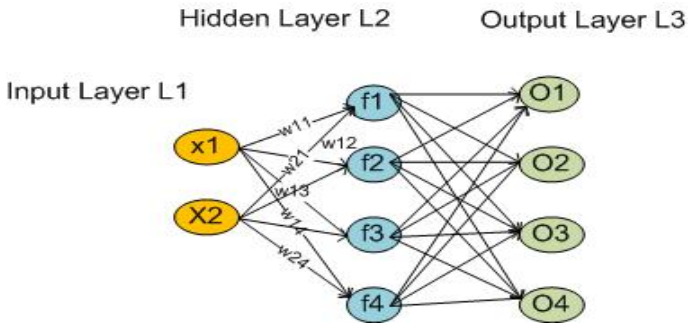

Figure 5. Diagram of a Neural Network for one Ultrasonic Sensor

Either the squared error function

$$E = -\sum_{z=1}^{4} 0.5\ x\ (\ target_z\ - actual_z\ )^2 \quad (6)$$

or the cross entropy can be used to determine the performance of the neural network

$$CE = CE\ (target, output) = -\sum_{z=1}^{z=4} t_{iz}\ \log(\ o_{iz}) \quad (7)$$

where tiz is the target and oiz is the actual output of trained neural network.

In the backpropagation method, the optimal weights are determined by the gradient descent of total error, E, by taking the partial derivative of E with respect to the weights as given by,

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial z_k}\ x\ \frac{\partial z_k}{\partial f}\ x\ \frac{\partial f}{\partial w_i} \quad (8)$$

For the cross entropy, CE, taking the partial derivative of CE with respect to the weights as given by,

$$\frac{\partial CE}{\partial w_i} = \frac{\partial CE}{\partial z_k}\ x\ \frac{\partial z_k}{\partial f}\ x\ \frac{\partial f}{\partial w_i} \quad (9)$$

Once the partial derivatives are taken and evaluated, the old weights are adjusted accordingly such

$$W_{new} = W_{old} - \text{delta} * \nabla_W CE \quad (10)$$

Where delta is a small step value. Note: the new value of weights does not guarantee it is better than the old weight, it might be necessary to re-adjust weights according to error convergence criteria.

3.2 Ultrasonic Sensors

The testbed consists of 4 Maxbotix I2C ultrasonic sensors and an Arduino microcontroller [13, 14]. The sensors operate in open air environment and emit high frequency sound waves in the environment to detect an object's range. The range is determined by measuring the time of flight of the high frequency sound signal which is transmitted to and reflected back from the detected object [3, 14]. Figure 6 below shows the connections to the sensor SCL and SDA lines. The I2C bus is a two-way interface consisting of a data line (SDA) and a clock line (SCL) and connected to a pull-up resistor. Note: one pull-up resistor per bus is required connected to the SCL and SDA lines and not for each sensor as shown in the figure. When connecting multiple sensors, it is necessary to program first each sensor with a different address. Then specific sensor based on its address can be called to get the range data of an object.

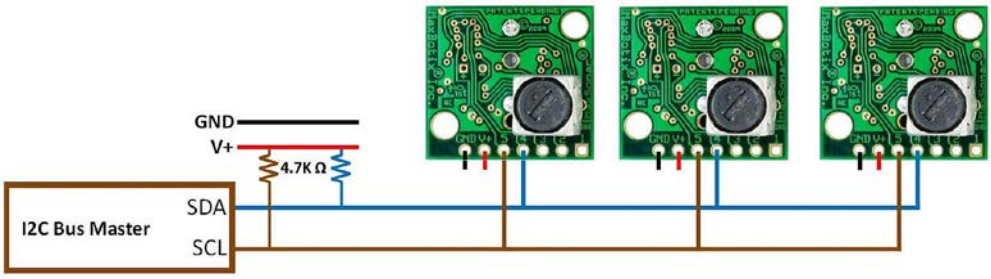

Figure 6. Diagram of I2C sensors connected to bus master [4].

The Maxbotix I2CXL sensors have the following features: resolution of 1 cm, I2C interface, reading rate of up to 40Hz, min distance 20 cm range, operational temp from 0 C to 65 C, resistance to high noise and real-time automatic calibration of humidity, noise and voltage. The sensors only operate as I2C slave. The default address of the sensor is a 7-bit address of 0x70.

3.2.1 Operation of I2CXL Maxsonar sensors

The sensor remains in an idle state listening to I2C bus signals after the power up cycle. It operates as an I2C slave only and never as a bus master. The sensor waits for a read or write instruction from a bus master having a default address of 0x70 (decimal 112). The write address has hexadecimal value of E0 and read address value of E1. The address may be altered to any other value. In total a single I2C bus supports up to 127 sensors. The reliable operating clock speed is up to 100 kHz. Sampling at speeds faster than 10 Hz in high noisy environment may provide false data readings.

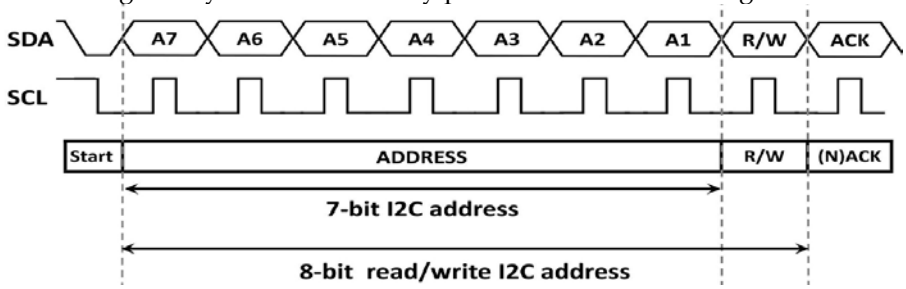

Figure 7. Diagram of address read or write [4].

Figure 7 explains the I2C addressing. Each message sent over the bus has 8 bits, the 7 bits are for the address and the last bit indicates a write to or read from the slave. The ninth bit sent back by the slave indicates the request is acknowledged or not acknowledged.

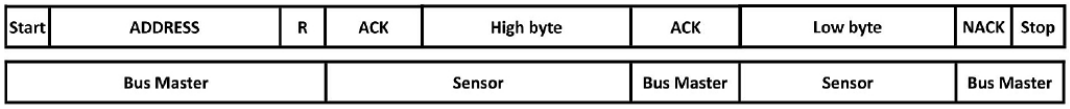

Figure 8. Diagram of bus master communication with sensor [4].

Figure 8 above shows the I2C command diagram. The steps are as follows bus master sends a signal to perform range reading command. Sensor sends ack for address match and the high byte. Bus master sends an ack for receiving the high byte and sensor sends low byte. The readings sent by the sensor are measured in the units of centimeters. The stop signal indicates the bus master will not communicate further.

The following is a code for Arduino to gather the ultrasonic sensor's range values to objects and on how to change the initial default addresses of the Maxbotix ultrasonic sensors in order to function properly on an I2C bus.

Code 1. Code to change default address and take readings.

```
/* Code for Arduino Uno Each sensor starts with a default address and has to be changed one by
one so the 4 sensors each will have a different address. Wire connection of sensors would be as
follows: Pin 4 to SDA; Pin 5 to SCL; Pin 6 to 5 V; Pin 7 to GND SCL and SDA to work reliably
require pull-ups to 5V */
#include "Wire.h"
#define Saddress byte(0x70)
#define DistInstr byte(0x51)                    // The sensors ranging command has a value of 0x51
#define AddressChngInstr1 byte(0xAA)     // These are two commands that must be sent in
#define AddressChngInstr2 byte(0xA5)     //    in sequence to change the sensor address
void setup() {
    Serial.begin(9600);
    Wire.begin();        }                    // Initiate Wire library for I2C communications
```

```
void loop(){
   PerformDistRead();                    // Instruct sensor to perform a distance cycle
   delay(100);                           // Give time for sensor to finish
   word Dist = requestDist();            // Send a request for Distance from sensor
   Serial.print("Dist: ");   Serial.println(Dist);   }          // Print the distance
void PerformDistRead(){//   Instructs sensor to perform a Distance reading
    Wire.beginTransmission(Saddress);        //   Begin addressing
    Wire.write(DistInstr);                   //    send Distance instruction
    Wire.endTransmission(); }                //    End transmission
word requestDist(){         //Returns last Distance sensor took in its last distance cycle in cm
     Wire.requestFrom(Saddress, byte(2));       // Returns 0 if no comm.
     if (Wire.available() >=   2) {             // Sensor replied with two bytes
        byte HighByte = Wire.read();            // Read high byte
        byte LowByte = Wire.read();             // Read low byte
    word Dist = word(HighByte, LowByte);    // Form a 16-bit word out of two bytes for distance
    return Dist;    }
else { return word(0); }   }                    //   otherwsie nothing was received, send 0
/* Instructs sensor at oldAddress to change address to newAddress oldAddress must be the 7-bit
form address is used by Wire 7 BitHuh determines whether newAddress is given as the new 7
bit version or the 8 bit version of the address If true, then if is 7 bit version, if false, it is the 8 bit
version */
void changeAddress(byte oldAddress, byte newAddress, boolean SevenBitHuh){
    Wire.beginTransmission(oldAddress);        //Begin addressing
    Wire.write(AddressChngInstr);              //Send first change address command
    Wire.write(AddressChngInstr2);             //Send second change address command
byte temp;
 if (SevenBitHuh){ temp = newAddress << 1; } //The new address must be written to the sensor
    else { temp = newAddress; }              // in the 8bit form, so this handles automatic shifting
       Wire.write(temp);                     // Send the new address to change to
       Wire.endTransmission(); }
```

4. Results and Analysis

4.1 Testbed setup

Figure 9 below shows 4 sensors with 4 objects placed in grid of cells. Each cell in the grid may have only one object placed in it. Each sensor forms an independent column and may contain only one object in one of 4 classes. With the arrangement shown in figure 9 the total number of patterns formed are 4*4 = 256 patterns. In general, given N sensors arranged linearly, a grid of size Mrows x Ncols can be formed, where M is the number of rows (or classes) of approximately equal length in the grid. Knowing the maximum and minimum range of the sensor measurement capability and resolution, M can easily be formed. Letting L=max – min indicate the total length of the grid. Then let M = L / r represent the number of rows formed in a grid, where r is the length of each cell. For example, if the min and max range of a sensor is 6 cm and 36 cm, respectively. Then L = 30 cm and M = 30 / 6 = 5 rows, each of length 6 cm. The total number of patterns formed are M^N. In this case, 5^4 = 625 patterns.

The number of cells formed is a function of sensor's properties such as angular resolution, max-min ranges and precision in range measurement. The object's width has an affect on sensor's detection

capability, in [3] a method was proposed to approximately measure an object's width or a gap between two objects. It was observed the accuracy depends on the type of sensor's used. With the setup shown in the figure there are 256 patterns. Number of patterns formed by multiple ultrasonic sensors placed linearly. In fig 9 a setup is shown using 4 sensors and 4 rows of equal spacing forming a grid of 16 cells. The sensors are placed with a minimum gap between each other in a linear fashion such that there will be no beam overlap occurring from its neighbours. In other words, each sensor will have a column of a clear path to an object to measure object's range. An object may be placed anywhere in a cell. For example, two objects placed in two different columns but same row may have different distance readings by the sensors yet belong to the same class. The neural network is trained by a cluster of patterns having a variety of distance readings. Once the neural network is trained by approximately 75% of the patterns, the other 25 % of the patterns are used for validation and testing the neural network for performance.

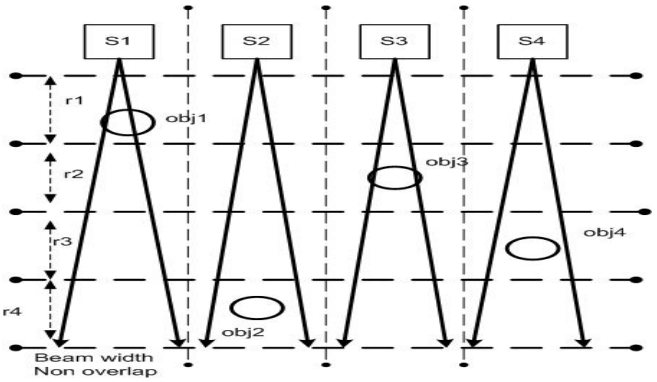

Figure 9. Testbed consisting of 4 ultrasonic sensors.

Table 1 shows sensors' range readings and used for training the neural network with its classes. The output of the neural network is in binary (i.e. 1000 -> class 1; 0100 -> class 2; 0010 -> class 3 and 0001 -> class 4).

Table 1. Testbed consisting of 4 ultrasonic sensors.

Objects' Distances in centimetres								
No.	**Sensor 1**	**Class**	**Sensor 2**	**Class**	**Sensor 3**	**Class**	**Sensor 4**	**Class**
1	4.6	1	4.2	1	14.3	3	19.8	4
2	15.3	3	9.8	3	10.4	3	10.5	3
3	18.2	3	19.4	3	8.1	2	8.4	2
….	….		….		….		….	
….	….		….		….		….	
500	14.1	3	10.2	3	8.4	2	9.3	2

4.2 Performance

Each sensor's range reading depends on an object's location within a cell. Figure 9 shows object's location may fall in one of the 4 classes (i.e. rows 1 to 4). The epochs indicate the number of times NN is trained. Figure 10 shows the graphs of training, validation and testing phases. The 3 graphs depict a downward trend till epoch 17, after which the validation graph shows a slight up and downward trend. It is seen the best validation performance is 0.0027 at epoch 17. Validation is a good indicator of network behaviour. As the value goes up after epoch 17 it indicates the network is over trained and it may become unstable with time. At this point the network training is stopped. The gradient value is 0.001238 and validation checks is 6 (maximum number of allowed failures) at epoch 23 as shown in figure 11.

The error histogram shown in figure 12 provides important information of errors occurrence in the Neural network. The Error frequency and its value are two important elements in the histogram. It is seen the variance is minimal in this model as seen from the graph and centred around 0 error. This is the case when with training data have minimum overlap and network is able to differentiate the 4 classes with a close to 100% accuracy. This is confirmed with confusion matrix which provides valuable information about the accuracy and precision of network's outputs. Figure 13 shows the four types of

confusion matrix. The training confusion matrix shows 66 of class 1, 65 of class 2, 75 of class 3 and 74 of class 4 have been correctly classified and none being misclassified. The Neural network proposed classifies training samples with 100% accuracy. The validation and test matrices also indicate 100% and 98.3% accuracy, respectively. The proposed network is easily scalable. The overall performance is 99.8% as indicated by the all confusion matrix. Similar results are obtained from other 3 sensors in the testbed. Figure 14, shows the performance of the NN based on error squared function. It is seen there is drastic downward trend in the error graph after epoch 30. The best error performance occurs at epoch 40 with a value of 9.2 x 10-14,

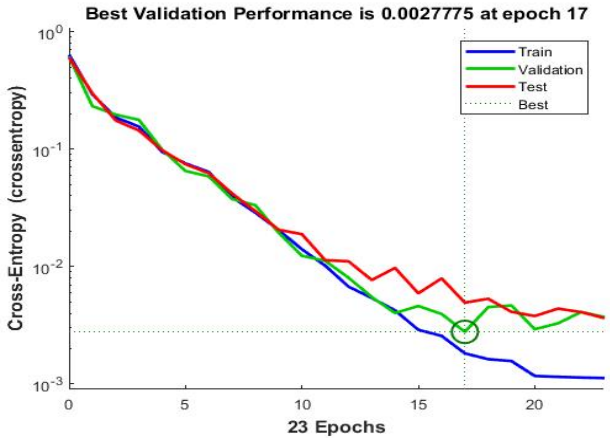

Figure 10. Performance

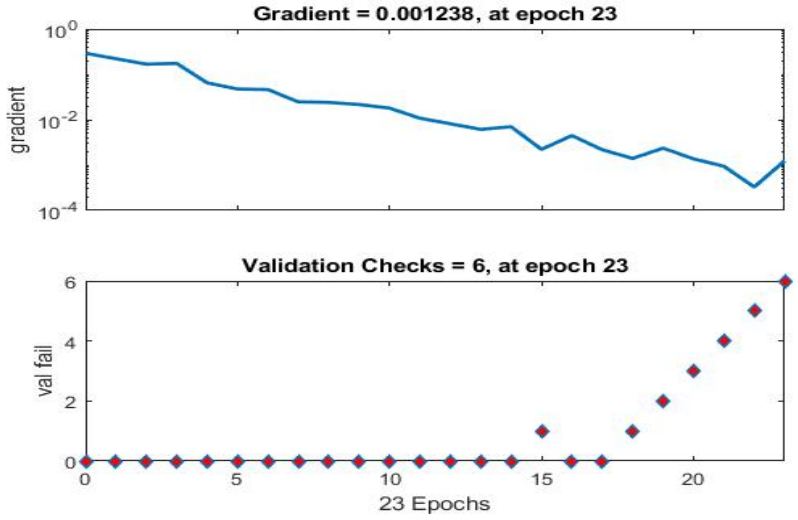

Figure 11. Gradient and Validation Checks.

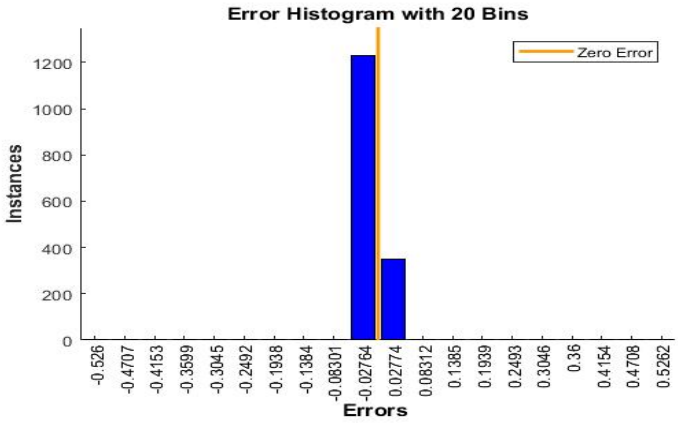

Figure 12. Errors = Targets - Outputs

Figure 13. Confusion matrix

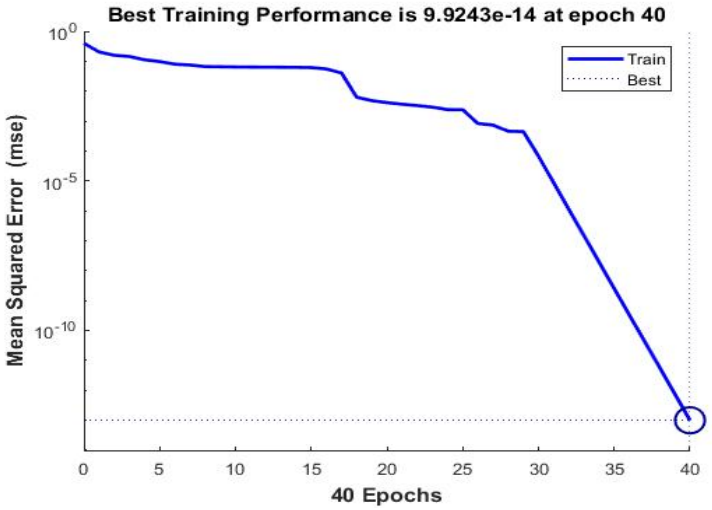

Figure 14. Mean Squared Error Performance

5. Conclusions

A neural network was implemented on an Arduino microcontroller and I2C ultrasonic sensors were used to read in range values of OPBG. A set of objects placed in cells formed unique patterns which may be used for control purposes. A cluster of range readings forms cells (classes). Due to the limitation of the Arduino memory, the training was performed in batches. Once the neural network was trained, it was seen the performance to be near to 99% accurate. The OPBG system may be useful in a number of applications, where pattern recognition is used for controls, such as in smart homes, robotics, and gaming. The scalability of the Neural network is easily achieved due to each sensor being independently operated. A total of 127 sensors maybe added on an I2C bus. For each additional sensor added to the testbed a parallel set of 4 classes are added. As a result, the patterns formed are Msensors hence allowing for more devices or tasks to be controlled. The high level of Neural network's classification accuracy and reliability makes it very suitable for control applications (i.e. using the patterns of objects as a code for control). Increasing the number of classes to be classified is a function of sensor's properties and range capabilities and is a subject for future research.

Acknowledgment

The author is grateful to the Faculty of Computer Science and Information Systems at Islamic University of Madinah for their support in this research.

References

[1] Harpreet Kaur and Jyoti Rani, A review: Study of various techniques of Hand gesture recognition, 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), 4-6 july 2016.

[2] Marek Kulawiak and Zbigniew Lubniewski, Processing of LIDAR and Multibeam Sonar Point Cloud Data for 3D Surface and Object Shape Reconstruction, 2016 Baltic Geodetic Congress (BGC Geomatics), pp. 187-190, 2016.

[3] Fazal Noor, et. Al, "A Method to Detect Object's Width with Ultrasonic Sensor", IEEE International Conference on Computing, Electronics & Communications Engineering 2018, ICCECE '18 Conference, University of Essex, Southend, UK, August 2018.

[4] MaxSonar Datasheet. Available: https://www.maxbotix.com/documents/I2CXL-MaxSonar-EZ_Datasheet.pdf, Accessed September 2018.

[5] Martin T. Hagan and Howard B. Demuth, Neural Network Design, 2nd Edition, eBook

[6] R.P. Lippmann, Pattern Classification using neural networks, IEEE Communications Magazine, vol 27, issue 111, Nov pp. 47-50, 1989.

[7] D. Zipser, R. A Andersen, "A Back Propagation Programmed Network that Simulates Response Properties of a Subset of Posterior Parietal Neurons", *Nature*, vol. 331, pp. 679-684, 1988.

[8] L. W. Chan, F. Fallside, "An Adaptive Training Algorithm for Back-Propagation Networks", *Comp. Speech and Language*, vol. 2, pp. 205-218, 1987.

[9] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control Signals and Systems*, vol. 2, no. 4, 1989.

[10] G. Nagy, "Candide's Practical Principles of Experimental Pattern Recognition", *IEEE Trans. on Pattern Anal. and Machine Intel.*, vol. PAMI-5, no. 2, pp. 199-200, 1983.

[11] M. F. Tenorio, W. T. Lee, "Self Organizing Neural Networks for the Identification Problem" in Advances in Neural Information Processing Systems, CA, San Mateo: Morgan Kauffman, vol. 1, pp. 57-64, 1989.

[12] Xiaolei Ma, et.al, Parallel Architecture of Convolutional Bi-Directional LSTM NeuralNetworks for NetworkWide Metro Ridership Prediction, IEEE Transaction on Intelligent Transportation Systems, pp 1-11, 2018.

[13] F. Noor and M. Alhaisoni, Distinguishing moving objects using Kalman Filter and Phase Correlation methods, 17th IEEE International Multi Topic Conference 2014, pp 299-304, 2014.

[14] F. Noor, Modeling Bat's Sonar System using a Microcontroller, International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 5, No6, December 2015.

[15] James Mora, Dec 19, 2017, https://www.electronicdesign.com/embedded-revolution/chirp-microsystems-sticks-sonar-chips-detect-world-around-them, Accessed Nov 2018.

[16] Thomas Kopinski, Fabian Sachara, Uwe Handmann, A Deep Learning Approach to Mid-air Gesture Interaction for Mobile Devices from Time of Flight Data, Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services, pp 1-9, Hiroshima, Japan, Nov 28 to Dec 1, 2016.

[17] Yu Sang, Laixi Shi, and Yimin Liu, Micro Hand Gesture Recognition System Using Ultrasonic Active Sensing, IEEE Access, Vol 6, pp. 49339 – 49347, 2018.

© 2019 by the author. Published by Annals of Emerging Technologies in Computing (AETiC), under the terms and conditions of the Creative Commons Attribution (CC BY) license which can be accessed at http://creativecommons.org/licenses/by/4.0.