**Design of a virtual sensor using machine learning imputation techniques in a wireless sensor network.**

Final Report

**M. Matusowsky**

11093804

Submitted as partial fulfilment of the requirements of Project EPR402

in the Department of Electrical, Electronic and Computer Engineering

University of Pretoria

November 2018

Study leader: Mr D. Ramotsoela

Copy 1 of 4

|  |
| --- |
| **Part 1. Preamble** |

This report describes the work that I did in designing a virtual sensor for a wireless sensor network using a machine learning technique.

*Project proposal and technical documentation*

This main report contains a copy of the approved Project Proposal (Part 3 of the report) and technical documentation (Part 5 of the report). The latter provides details of the schematics, stripboard layout, and software code. I have included this appendix on a CD or DVD that accompanies this report.

*Project history*

This project does not build on any projects that were completed in previous years. Stripboard design software as well as the circuit error checking software was provided by Fritzing.org, an opensource initiative. The stripboard sensor circuit was designed and soldered by myself. One of the equations I used was adapted from Steinhart [1]. I used the Tkinter python library to implement the graphical user interface. All other work in this report was my own and completed from first principles.

*Language editing*

This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was Mr Joseph Henry Mervitz

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Language editor signature Date*

*Declaration*

I, Michael Matusowsky understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else.

Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

M. Matusowsky Date

**TABLE OF CONTENTS**

**Part 2. Summary** vi

What has been done vi

What has been achieved vi

Findings vi

Contribution vi

**Part 3. Project identification: approved Project Proposal** viii

1. Problem statement viii
2. User requirements x
3. Functional analysis xi
4. Specifications xv
5. Deliverables xvii
6. References xix

**Part 4. Main report**

1. Literature study 1

1.1 Background and context of the problem 1

1.1.1 Digital modulation techniques 1

2. Approach 4

2.1 Design alternatives 4

2.2 Preferred solution 6

3. Design and implementation

3.1 Theoretical analysis and modelling 8

3.2 Simulation study and optimization 12

3.3 Hardware design 16

3.3.1 Design alternatives 16

3.3.1 Design procedure followed 17

3.3.2 Design equations and design calculations 17

3.3.3 Final design schematics 19

3.4 Hardware implementation

3.4.1 PC board layout and manufacturing process 22

3.4.2 Hardware integration and testing 25

3.5 Software design 27

3.5.1 Design alternatives 27

3.5.2 Design procedure followed 27

3.5.3 Design calculations 29

3.6 Software implementation 30

3.7 Final system integration and testing 32

3.8 Design summary

4. Results 34

4.1 Summary of results achieved 34

4.2 Qualification tests 34

4.2.1 Qualification test 1: testing of communication distance 34

4.2.1.a Qualification test

*Objectives of test/experiment* 34

*Equipment used* 34

*Experimental parameters and setup* 35

*Experimental protocol (experimental steps)* 36 4.2.1b Results and observations

objective description

*Measurements*  37

*Description of results* 38

*Statistical analysis* 39

4.2.2 Qualification test 2: measurement of bit error rate 40

4.2.2.a Qualification test

*Objectives of test/experiment* 40

*Equipment used* 40

*Experimental parameters and setup* 40

**Very important section - read notes in study guide**

*Experimental protocol (experimental steps)* 42 4.2.2b Results and observations

*Measurements*  43

*Description of results* 45

*Statistical analysis* 46

5. Discussion (subjective interpretation of the implications of the results)

5.1 Interpretation of results 50

5.2 Aspects to be improved in current design 51

5.3 Strong points of current design 51

5.4 Under which circumstances will the current system fail? 52

…

6. Conclusion 55

6.1 Summary of the work 56

6.2 Summary of the results and conclusions 58

6.3 Suggestions for future work 59

7. References 60

**Part 5. Technical documentation** 61

**LIST OF ABBREVIATIONS**

**TDD**  Test-driven development

**PIC32** Peripheral interface controller 32-bit

**MCU** Microcontroller Unit

**TCP/IP** Transmission control protocol/Internet protocol

**PC** Personal computer

|  |
| --- |
| **Part 2. Summary** |

This report describes work carried out on the design of a virtual sensor using machine learning in a wireless sensor network with the objective of using the virtual sensor to impute sensor values of a single sensor node, if a sensor node in the network fails, using the other sensor nodes that are still actively transmitting.

**What has been done**

A literature survey was completed on modern wireless communication systems as well as different machine learning implementations. The hardware and software for the sensor nodes and server was then designed and implemented from first principles. communication system was then designed from first principles. At the core of the hardware system is three PIC32 (Peripheral interface controller 32-bit) microcontroller units (MCU) with an ESP8266 WiFi module that includes a transmission control protocol/internet protocol (TCP/IP) framework, and all additional hardware was designed and implemented. A Python program was developed to simulate the system, as well as C code for the PIC32 processor. The system was implemented and tested throughout the development phase using a test-driven development (TDD) process. Four WiFi modules were used in a star network topology whereby one WiFi module acted as the TCP/IP server on the desktop personal computer (PC) and the other three modules acted as TCP/IP clients on the sensor nodes. The neural network for each sensor node was implemented on the server as well as on the corresponding sensor node’s MCU. The main results from one of the test cases is shown in the figures below.

(Add a figure or figures here – this will probably be a figure that also appears in section 4 of the report).

**What has been achieved**

Successful imputation using machine learning was achieved in a wireless sensor network that was deployed in multiple locations. The accuracy of the imputation depended highly on the amount of training data available where it was discovered that a minimum of 5 days of data was required before a sensor node would be adequately trained. Figure X shows a virtual sensor trained with 2 days of data while figure Y shows a virtual sensor trained with 5 days of data.

**Findings**

It was found, while analysing the weights of the neural networks, that the time input in the neural network was much more important than the sensor inputs with regards to imputing temperature data. Another important discovery was that due to glitches within the hardware, the sensed value by the thermistor would sometimes return a ground value and thus a filtering algorithm had to be implemented to account for the glitches. Such an occurrence is shown in figure X below.

**Contribution**

New software that had to be mastered to complete this project was Fritzing. Fritzing is an opensource PCB/stripboard/circuit designing software that undergraduate students would not usually be aware of and has not been covered in any undergraduate module. The Fritzing software was used specifically for the stripboard module to plan, test and implement the circuitry on a stripboard. Further software that needed to be learned was the MPLAB Harmony Framework for PIC32 MCU’s which was extremely useful in configuring the right clock speed as well as the Universal Synchronous/Asynchronous Receiver/Transmitter (USART) and analog-to-digital converter (ADC) modules.

Due to the limited amount of program memory in the PIC32 and the requirement for string handling, some functions usually provided by the standard library for C had to be re-written from first principles due to the overhead in program memory introduced when using functions such as sprint() which would allow data to be converted from integers and floats to strings and then written to the USART buffer.

A server was developed using Python on a PC connected by USB with a WiFi module. The server would act as the communication center to periodically request and transmit readings from the sensor nodes as well as determine if a sensor node is offline. The code for the serial communication was implemented by the student with the use of the pyserial library.

Code for the neural network, including the training algorithms, was developed by the student without reliance on any existing libraries in python. A framework for training neural networks with a minimum of 3 inputs, theoretically unlimited hidden nodes and layers, and a maximum of 1 output was developed from first principles. A friend in a postgraduate course helped provide some clarity on the chosen training method that is used as it was complex and completely new to the student since no undergraduate module covers evolutionary models for training in neural networks. The study leader provided no assistance with regards to the training but did provide some advice regarding the output layer of the neural network as the student originally wanted to have the output be a binarized output layer with multiple output nodes. The study leader advised to use a single output node in the output layer.

There was a strong reliance on an existing python library called csv to deal with reading and writing from the local database.

|  |
| --- |
| **Part 3. Project identification: approved Project Proposal** |

|  |
| --- |
| 1. **Problem Statement** |

In this section the problem that is being addressed by the project proposal is described in terms of the motivation, context, technical challenges and limitations of the proposed project.

**Motivation.** Sensors are devices used in everyday objects such as mobile phones, touch-sensitive elevators and self-driving cars. There are innumerable amounts of applications where sensors are used to gather data about the environment and then used in other electronics to monitor or react to the conditions being sensed such as the lighting in a room being adjusted based on the required illumination level. A problem arises in wireless sensor networks (WSN) where, due to the nature of the wireless communication, data can be lost or corrupted during the transmission phase due to external factors such as solar radiation corrupting data from satellites or congestion in a network causing packet loss when transmitting data to other devices. Furthermore, the cost of implementing or needing to replace many physical sensors in a network can become prohibitively expensive.

This project will look at designing and implementing a wireless sensor network that will make use of data imputation methods and machine learning to realise virtual sensors that can completely replace physical sensor nodes and give accurate substituted data in place of nodes with failed sensor modules.

**Context.** The integrity of received information is an important issue in the modern age. Sensors play a pivotal role in electronic devices of all shapes, sizes and function and especially more so in wireless sensor networks where data loss is an expected occurrence [1]. Data imputation and virtual sensors are tools that allow a system to counter-act the effect of this data loss by accurately substituting values that real sensors would most likely return [2]. This then allows the system to still make use of incomplete data rather than completely discarding affected data entries.

The main function of this project will be to ensure the robustness of a wireless sensor network by making sure that damaged nodes in a wireless sensor network can be replaced by virtual sensors using imputation techniques that make use of machine learning algorithms which will allow the system to remain robust in terms of the provided sensor data even when a node malfunctions or is removed from the network thus allowing the system to continue running in real-time while gathering data that closely resembles the affected sensor nodes would-be data.

K-Nearest Neighbours (KNN, lazy learning), multi-layered perceptrons (MLP, supervised learning) and self-organizing maps (SOM, unsupervised learning) are three popular machine learning methods that have been used to great effect in data sets that do not deal with time-series analysis [3,4,5,6] where KNN, MLP and SOM outperform traditional imputations techniques , such as hot-swapping, to significant degrees on multiple data sets including but not limited to a breast cancer detection data set, a seed classifying data set and sonar imaging data set.

**Technical Challenges.** The technical challenges for this project are: (i) Designing and implementing an imputation technique using machine learning algorithms on individual sensor nodes that can act as a virtual sensor. (ii) The algorithm should be robust to ensure the integrity of the data that is being substituted by the virtual sensors. (iii) The algorithm should be efficient enough so that congestion does not occur due to computations taking place which in itself would then cause loss of more data.

**Limitations.** The availability of bandwidth in the network will limit how many readings can be transferred and received per time interval between all nodes and the server. The second limitation is the processing speed and power of the processing unit at each sensor node. Another limitation is the cost of developing the product thus cost-effective hardware is a necessity as well as putting a financial limit on the amount of nodes that can be physically implemented for this project.

|  |
| --- |
| 1. **Project requirements** |

The main aim of the project will be to create a wireless sensor network that makes use of the machine learning imputation techniques to ensure the integrity and robustness of the data that is transmitted and received over the network.

**2.1 Mission requirements of the product**

The system will need to fulfil the following requirements.

* The system must use machine learning algorithms to implement virtual sensors in the wireless sensor network.
* Communication between nodes in the network must be done using wireless communication.
* The algorithm implemented must be efficient enough so as not to introduce computational congestion in the system.
* The system must detect a malfunctioning node and replace it with a virtual sensor.
* The virtual sensor must be able to replicate data accurately as if they were real sensors.
* The data read by all the sensors must be stored in a database.
* The virtual sensors must be implemented on every corresponding node as well as having a copy of every trained virtual sensor on the server.
* Each sensor node should consist of a power unit, a sensor module and a processing unit.

**2.2 Student tasks**

The following tasks will need to be completed.

* An investigation of the current machine learning imputation techniques in literature must be done.
* An investigation of where and how virtual sensors have been implemented must be done.
* An investigation of wireless communication interfaces must be done to decide on the best communication interface for the product must be done.
* An investigation for the causes of lost data in wireless sensor networks must be done.
* The imputation algorithms must be designed and implemented with a focus on robustness.
* The algorithms must be trained and tested in MATLAB or Python using data that has been collected from the environment being sensed by the hardware.
* The control algorithms on the processing units for each sensor node must be implemented.
* The circuitry for the sensor nodes must be implemented on stripboard.
* The software and GUI for the server must be designed and implemented on a PC.

|  |
| --- |
| 1. **Functional analysis** |

This section contains information on the subsystems and each function unit of the system separated into the hardware and software components.

The hardware component, shown in Figure 1, will consist of a sensor module (FU 1) which will sense the characteristics of the environment and be attached to a processing unit (FU 2). These two functional units cover a single sensor node. The processing unit will, at regular intervals, take environmental readings from the attached sensor(s) and package the data for transmission via wireless communication (FU 3) with the other sensor nodes as well as a central server (FU 4) which will contain the database that will store historical data for later use in data modelling and then training the virtual sensors of each sensor node as well as general storage for historical data of the network.

**Figure 1. Overview of the hardware component**



**Figure 2. Overview of the server software component.**

The server software component, shown in Figure 2, will consist of the database and all the software of the server of the system.

When the system is functioning in a manner that all the sensor nodes are active with functioning physical sensor nodes, the server will simply store the sensor readings received from the sensor nodes through a wireless communication link (FU 3) in the local database. This data will then be modelled (FU 5) so that I may be used as offline training (FU 6) data for the machine learning algorithms that will be deployed as the virtual sensors on both the server as well as the processing unit of the sensor nodes.

When the system is functioning in a manner that a sensor node in the network has been taken offline, the incoming sensor data to the server from the remaining sensor nodes will be stored in the database and modelled (FU 5) to be used as an input for the virtual sensor (FU 7) contained on the server that will be activated once it is determined that a sensor node has stopped communicating with the server.

When the system is functioning in a manner that a sensor node is online (i.e. it is communicating with the server) but does not have a functioning sensor module attached, the server will continue to receive readings from the remaining sensor nodes that do have a functioning sensor module attached. These readings will be stored in the database and forwarded to the sensor node without a functioning sensor module to be used as the input for the virtual sensor contained on the sensor module.



**Figure 3. Overview of the sensor node software component.**

The sensor node software component, shown in Figure 3, will consist of two modes of operation.

The first and default mode of operation will consist of the sensor node taking environmental readings using the attached sensor module (FU 1) and using the digital filtering algorithms to reduce noise from the read data (FU 8). This data is then uploaded to the server using a wireless communication link (FU 3) to be stored in the server’s database.

The second mode of operation will consist of the sensor node imputing environmental readings by activating the virtual sensor (FU 7), which itself will use the machine learning imputation method, if the node detects that no sensor module is attached to the processing unit. Readings from the other sensors in the network will be passed to the sensor node by the server through the wireless communication link (FU 3) and then modeled so that the data can be used as input data for the virtual sensor. The imputed sensor data from the virtual sensor is then uploaded to the server for storage in the database.

One of the potential methods of the virtual sensor involves the usage of a self-organising map, an unsupervised neural network, where the neurons are organised in a connected 2-D grid as shown in figure 5 and the correlation in data is learned using the steps in Figure 4.



**Figure 4. The steps of a self-organising map.**



**Figure 5. A self-organising map neural network after 301 iterations.**

The self-organising map initially randomises the positions of all the neurons. A data point is then selected randomly to be used as the reference point to match the neuron that is the closest in proximity to the point. The neuron is then moved closer to the data point. The neighbouring neurons in a certain radius of that neuron are then also moved closer based on how far they were positioned from the data point, with further neighbouring neurons moving less. The moving rate and the radius for the chosen neuron, also known as the learning rate, is then decreased on each iteration. These steps are repeated until the positions of all the neurons has stabilised in a way that significant neuron movement does not occur as seen in Figure 5.

|  |
| --- |
| 1. **Target specifications** |

This section contains the main specifications of the proposed project.

**4.1 Mission-critical system specifications**

The mission critical systems specifications are given in Table 1 below.

|  |  |  |
| --- | --- | --- |
| **SPECIFICATION**  (in measurable terms) | **ORIGIN**  **(**or motivation of this specification) | **VERIFICATION**  (how will you confirm that your system complies with the specification?) |
| The virtual sensor using the MLP neural network imputation method should impute values, at minimum, within 30% accuracy of the actual observed values for 90% of the data. | [2] and [3] both show that the imputed values from MLP neural network methods are within 30% of the actual observed values on average and within 15% in an absolute best-case scenario. | The imputed values from the virtual sensors will be compared to the physical sensor readings. |
| The virtual sensor using the MLP neural network imputation method should not stray further than 1 standard deviation from the mean values that would be sensed by a physical sensor. | A standard deviation greater than 1 would indicate that the virtual sensor has been improperly trained and will not be of any practical use. [7] suggests a standard deviation of not higher than 1 would provide a good indication of a well-trained system. | Each virtual sensor, both on the server and on the node, will be tested by turning nodes ability to physically sense the environment as well as completely turning off a sensor node to ensure that the server can activate the corresponding virtual sensor. If the virtual sensor can impute data that would be sensed by the physical sensor within the specified standard deviation from the mean, it would be deemed functioning correctly. |
| Imputation of a single value should take no longer than 7 seconds when using the MLP neural network algorithm while running on the server once the physical sensor data from the functioning sensor nodes has been input into the virtual sensor. | Imputation of a single value takes up to 7 seconds in [1] when using the MLP neural network imputation technique and this is due to the computations being done by the processor. To prevent congestion due to server processes taking place, some time must be given for these computations to take place before new values can be imputed. | A timer will be setup between the time that the virtual sensor has been given the input and the time that the virtual sensor gives an output. This will determine the time it takes to impute a value on the server for the given algorithm. |

**Table 1. Mission-critical system specifications**

**4.2 Field conditions**

|  |  |
| --- | --- |
| **REQUIREMENT** | **SPECIFICATION**  (in measurable terms) |
| The sensor nodes must be directly connected to the server. | The connection will need to work over WiFi or Bluetooth |
| The server must run on a computer capable of processing large streams of data for training purposes. | Any moden computer with a CPU speed of at least 2GHz. |

**Table 2. Field conditions**

|  |
| --- |
| 1. **Deliverables** |

**5.1 Technical deliverables**

Table 3 shows the technical deliverables that will be required to complete the project.

|  |  |  |
| --- | --- | --- |
| **DELIVERABLES** | **DESIGNED AND IMPLEMENTED BY STUDENT** | **OFF-THE-SHELF** |
| Atmospheric (barometric or temperature or humidity) sensor modules for the processing units. |  | X |
| 16-bit or 32-bit microcontroller for the sensor nodes. |  | X |
| Wireless communication modules. |  | X |
| Power supply for the sensor nodes. |  | X |
| Sensor module interface with the microcontroller. | X |  |
| Wireless communication module interface with the microcontroller. | X |  |
| Desktop PC for the server. |  | X |
| Desktop monitor. |  | X |
| Software libraries for accessing the database. | X |  |
| Processing unit sensor control software. | X |  |
| Processing unit communication protocol. | X |  |
| Database on the server to store the data received from the physical sensors. | X |  |
| Server communication protocol. | X |  |
| Imputation algorithm using MLP neural network technique in MATLAB or Python for simulation. | X |  |
| Imputation algorithm using MLP neural network technique on the sensor node processing units. | X |  |
| Imputation algorithm on the server. | X |  |
| Data modelling software on the server. | X |  |
| Data modelling software on the sensor nodes. | X |  |
| Graphical user interface for the PC application. | X |  |

**Table 3. Deliverables**

**5.2 Demonstration at the examination**

1. All the software will be preloaded onto the sensor nodes and the server before the demonstration begins.
2. The physical system will then be demonstrated by switching on all the sensor nodes and the server and activating the system.
3. The graphical user interface on the PC will be used to show the current outputs of the sensor nodes in the network.
4. A software command will be submitted to one of the sensor nodes from the PC to activate the virtual sensor using the MLP neural network algorithm to impute data.
5. The sensor node will acknowledge this command and return the imputed value as well as the sensor reading to compare it values. This will confirm the virtual sensor is functioning correctly using the MLP neural network imputation technique.
6. A software command will then be entered on the server to activate a virtual sensor using the MLP neural network imputation technique on the server that will impute values for the chosen sensor node. The values will again be compared to the values sensed by the sensor node. This will confirm that the virtual sensor on the server is functioning correctly using the MLP neural network imputation technique.

|  |
| --- |
| 1. **References** |

[1] Y. Li and L. Parker, “Nearest neighbour imputation using spatial-temporal correlations in wireless sensor networks,” *Information Fusion*, vol. 15, pp. 64-79, 2014.

[2] J. Jerez, I. Molina, P. Garcia-Laencina, E. Alba, N. Ribelles, M. Martin, and L. Franco, “Missing data imputation using statistical and machine learning methods in a real breast cancer problem,” *Artifical Intelligence in Medicine*, vol. 50, no. 2, pp. 105-115, 2010.

[3] P. Garcia-Laencina, J. Sancho-Gomez, A. Figuieras-Vidal, and M. Verleysen, “K nearest neighbours with mutual information for simultaneous classification and missing data imputation,” *Neurocomputing*, vol. 72, no. 7-9, pp. 1483-1493, 2009.

[4] L. Li, J. Zhang, F. Yang, and B. Ran, “Robust and flexible strategy for missing data imputation in intelligent transportation system”, *Intelligent Transport System*, vol. 12, no. 2, pp. 151-157.

[5] K. K. Nalanda, “Using fuzzy c means and multi layer perceptron for data imputation: Simple v/s complex dataset,” in *3rd International Conference on Recent Advances in Information Technology*. IEEE, March 2016.

[6] Y. Li and L. Parker,” Spatial-temporal imputation technique for classification with missing data in a wireless sensor network,” in *Proceedings of IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2008.

[7] A. Wang, Y. Chen, N. An, J. Yang, L. Li, and L. Jiang, “Microarray missing value imputation: A regularized local learning method,” in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. IEEE, 2018.

|  |
| --- |
| **Part 4. Main report** |

**1. Literature study**

* 1. **BACKGROUND AND CONTEXT OF THE PROBLEM**

With the widespread proliferation of sensors in all different types of environments a new use case has come up in recent times called the Internet of Things (IoT) where sensor devices are interconnected to form wireless sensor networks (WSN). WSN have applications in many different fields including but not limited to the mining and aeronautic industry.

Different WSN may use different technologies to achieve the required goals, be it networks that may require low-power usage (LoRa), networks that need to communicate over vast distances at a low cost (Sigfox) or networks that need fast and reliable communication (ZigBee).

Two problems arise in all these WSN that may cause loss of important data. The first problem deals with the noise level in the environment. If the noise level is high then it may result in a degraded signal strength and thus performance in the WSN degrades to a point where data packets are lost (packet loss) or corrupted packets being transmitted across the network. These lost or corrupted packets may then cause errors further down the WSN pipeline if the application requires corrective action.

The second problem deals with dead sensor nodes in the network. Dead sensor nodes are sensor nodes in a network that have failed or stopped responding and thus the data from these nodes is no longer being collected.

There are different systems in place that deal with the above problems, one of which is called data imputation where empty values are substituted with values that should be the same or close to the original values. There are two main methods of data imputation called statistical imputation and imputation using machine learning.

* + 1. ***Statistical imputation***

Using the existing data, missing values can be imputed using statistical analysis and applying different methods like hot-swapping, cold-decking, mean substitution and regression.

*1.1.1.1 Hot-swapping imputation*

Hots-swapping used to be a common data imputation technique where missing values would be imputed randomly from a randomly selected record that had other similar inputs. A common way to use this method would be to simply use the previous record to impute the record with the missing data. An issue arises with this method in that if a sensor goes offline for a n unspecified amount of time, then the method assumes that the values are not varying, and this increases the bias of the imputed values.

*1.1.1.2 Cold-decking imputation*

Cold-decking uses a similar principal to hot-swapping with the only difference being that instead of using the current dataset to randomly select values it uses a donor from a different dataset with the same fields. The issue is clear right away that using a dataset from a different source may have wildly different values than the dataset which requires imputation.

*1.1.1.3 Mean substitution imputation*

Mean substitution makes use of averaging of similar data then replacing the missing values with the imputed data. This does however cause the attenuation of any correlations involving the variables that are being imputed. This means that mean substitution is problematic in multivariate analysis but may be quite attractive for univariate analysis.

*1.1.1.4 Regression imputation*

Regression imputation involves predicting values based on other variables in the dataset and then fitting a curve to the dataset. The model is then used to impute values in cases where variables are missing. The issue with this method is that the estimates will fit perfectly along the regression line without any variance which causes the relationship to be over identified and suggests greater precision in the imputed values than is warranted. In other words, the model predicts the most likely value but does not supply any uncertainty about the value.

* + 1. ***Imputation using machine learning***

Using the existing data, machine learning algorithms may be implemented to learn the relationship between all the variables in a dataset such that if there are missing values then the chosen method of machine learning can impute the values based on the learned relationship. The three main methods of machine learning involve lazy using K-Nearest Neighbours algorithm (KNN) learning, supervised learning using a multiple layer perceptron (MLP) and unsupervised learning using a self-organising map (SOM).

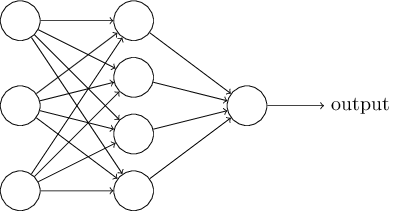
*1.1.2.1 Lazy learning imputation*

The KNN algorithm is one of the popular lazy learning algorithms in the field of machine learning. By utilising an existing dataset, the KNN algorithm is able to impute values in time-series datasets by comparing the incoming values and matching them to the *k* nearest values that closely resemble the similar multivariate values in the dataset as shown in insert figure here.

As the database increases, the speed with which the KNN algorithm can impute values begins to degrade and becomes computationally expensive thus making it unfeasible to implement on microcontrollers with limited computational capacity. Cite uses the KNN algorithm to great effect on a breast cancer dataset

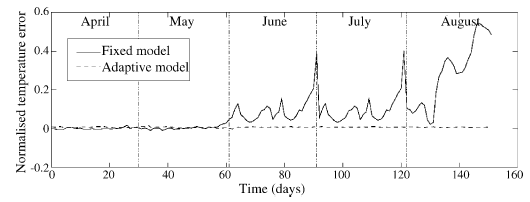
*1.1.2.2 Multiple layer perceptron imputation*

Artificial neural networks are a network of simple elements called artificial neurons which receive one or more inputs and then are passed through an activation function to change the internal state to produce an output based on the input and activation function. The multi-layered perceptron (MLP) neural network consists of a minimum three layers of neurons namely the input layer, the hidden layer (of which there may be more than one in the case of deep neural networks) and the output layer which are fully connected and each connection having its own weight. There are various methods to train the MLP weights with two methods being popular today namely the backpropagation algorithm [1] and genetic algorithms [2] . A typical representation of neural network is shown in **Figure 1** below showing that every layer is fully connected to the next layer.



**Figure 1: A basic neural network** [3]**.**

There have been multiple implementations of neural networks with regards to weather prediction and room temperature sensing. Devi et al [4] implemented a feed-forward neural network that would be used to predict weather patterns given multiple environmental inputs. They use backpropagation as the training method due to the ability of the algorithm to capture the complex relationship between the many factors that might influence the environmental temperature which may include but not be limited to: Atmospheric pressure, wind speed, wind direction, humidity, dew point temperature and elevation above sea level. The model was able to infer a relationship between the given inputs and outputs of historical data suggesting that a neural network can be used to predict some weather patterns however the system was not deployed in a real-time environment as the research was focused on the viability of a neural network prediction model. Hayati and Mohebi [5] applied a 3-layer neural network to design a short-term temperature forecasting system for Kermanshah city, Iran. Ten years of weather forecasting data from 1996-2006 was used to train the MLP, specifically 6-hour average soil temperature at various depths up to 100 cm, wet and dry temperatures [6], humidity, pressure, hours of sunshine and radiation. Training was done using the scaled conjugate gradient method, a numerical optimization technique, described by [Møller](https://www.sciencedirect.com/science/article/pii/S0893608005800565" \l "!) [7]. The appropriate number of hidden neurons used was determined through an iterative trial and error process where hidden neurons were added and the MLP re-initiated to a random state after a certain amount of epochs until a few MLP models emerged as likely candidates for predictive weather forecasting. The downside to this is that a lot of time is needed to run through the iterative process; however, the training method used converges much faster than the backpropagation algorithm making it a still viable method of determining the optimal MLP [8]. The optimal hidden neurons was found to be six, after 2000 epochs. The mean absolute error minimum and maximum of 0.0079 and 1.2916 show that the MLP is very accurate in terms of predicting future temperature based on recent historical data, never deviating more than 3.5 ° C from the actual measures temperatures. Ruano et al [9] implemented a MLP system in a school for smart energy systems to save on electrical costs. The costs involved expensive on-site meteorological tools and equipment and indoor sensors placed throughout the school. The type of sensors involved air temperature, humidity, atmospheric pressure sensors, the state of the doors and windows (open or closed) and air-conditioner power consumption. It is important to note that all data was acquired without human noise or noise from equipment i.e. a perfect environment. The MLP is trained using the Levenberg-Marquardt method [10, 11] due to it regressing faster than backpropagation and a second attempt is made using genetic algorithms due to the inability of the algorithm to fully explore the model space. Using the genetic algorithm, 1000 MLPs were initiated and 100 generations were run where 100 candidate models per generation would repopulate the population for the following generation. The MLP performs quite well when used in the period of the year that the training data was gathered (April-May) but decreased in accuracy as the weather changed from autumn to summer where no previous data was collected to train the MLP. An illustration of this error is shown in **Figure 2**.



**Figure 2: Neural network error increase as the year proceeds from the trained period [9].**

**2. Approach**

Commence this section on a new page

Your approach study, described in the study guide, goes here.

**2.1 Design alternatives**

**2.2 Preferred solution**

**3. Design and implementation**

Commence this section on a new page

**3.1 Theoretical analysis and modelling**

…

Very important table in this section.

The required content is described in the study guide. What appears here, is an example.

**3.7 Design summary**

This section summarises the project tasks and how they were implemented (see table 1).

|  |  |  |
| --- | --- | --- |
| **Task** | **Implementation** | **Task completion** |
| Design of a PCB for the main electronics | The PCB design was completed, using the PCBCAD package. This was done from first principles. | completed |
| Development of optimization routine | Optimization was completed in Matlab, but the Optimization Toolbox was used, and while some code was developed from first principles, numerical methods for optimization were taken off the shelf. | incomplete |

**Table 1**

**Design summary.**

Ensure that your table lists all of your tasks (technical design tasks, not including things like “write report”, or “bind report”). These would include student tasks in sections 2.3 and 2.5 of the Project Proposal.

**4. Results**

Commence this section on a new page

**Super important! The compulsory table mentioned in Appendix 4 of the study guide should appear here.**

**The following is just an example.**

**4.1 Summary of results achieved**

|  |  |  |
| --- | --- | --- |
| **Description of requirement or specification**  **(intended outcome)** | **Actual outcome** | **Location in report** |
| **Mission requirements of the product** | | |
| The system should provide continuous AC power to a household | The system could provide continuous power during daytime hours when grid power failed. | Section 4.2.6 |
| Motor speed should be controlled. | Stable feedback control of the motor speed was achieved. | Section 4.2.2 |
| Fuel consumption must be kept to a minimum. | Fuel consumption was slightly higher than expected. | Section 4.2.7 |
| Bit error rate (BER) should be low. | The measured BER was high. | Section4.2.1 |
| Delivered power should be adequate for the load. | The system could not deliver the required power into the load. | Section 4.2.3 |
| **Field conditions** | | |
| The system should supply power and actual environmental conditions (sunshine or rain; day or night) | The system was never tested under rainy conditions. The system could not supply power under any conditions other than bright sunlight. | Section 4.2.6 |
| The system should use actual real-time data, corrupted by noise, arriving over a noisy wireless link. | The system could work error-free for at least one hour under these actual field conditions. | Section 4.2.1 |
| **Specifications** | | |
| BER should be below 1E-6. | The BER was measured as 10 bit errors in 1000 bits. | Section 4.2.1 |
| The motor should reach 50 rpm. | 46 rpm was measured. | Section 4.2.2 |
| 2 kW should be delivered to the load. | The system could deliver 800 Watts into the load before overheating. | Section 4.2.3 |
| **Deliverables** | | |
| DC-DC convertors had to be designed and implemented by the student. | The student completed the design and implementation. | Section 4.2.8 |
| The inverter had to be designed and implemented by the student. | The student did not complete the inverter. The design was completed and simulated, but the implementation in hardware did not work correctly. | Section 4.2.8 |

**TABLE 2**

**Summary of results achieved.**

**4.2 Qualification tests**

**4.2.1 Qualification test 1: testing of communication distance**

4.2.1.a Qualification test

*Objectives of test/experiment*

*Equipment used*

*Experimental parameters and setup*

*Experimental protocol (experimental steps)*

4.2.1b Results and observations

*Measurements*

*Description of results*

*Statistical analysis*

**4.2.2 Qualification test 2: measurement of bit error rate**

4.2.2.a Qualification test

*Objectives of test/experiment*

*Equipment used*

*Experimental parameters and setup*

*Experimental protocol (experimental steps)*

4.2.2b Results and observations

*Measurements*

*Description of results*

*Statistical analysis*

**5. Discussion**

**See the study guide – this section is very important. You need to show that you can stand back and be critical of your own work. The worst possible thing that you can write here is “everything works perfectly”. There is no perfect design, and you as (aspiring) engineer should be able to point out the shortcomings of your design and/or results.**

**Tip: Use the headings in the study guide. These are *not* compulsory, but will help you to organize your thoughts, and the headings actually tell you some of the things that you are required to comment on.**

Commence this section on a new page

**6. Conclusion**

Commence this section on a new page

**6.1 Summary of the work**

**…**

**7. References**

**Here you need to be extremely honest about what you achieved, and did not achieve.**

**Conclusions need to be technical and MAY NOT relate to your personal experience (e.g. “I learnt a lot” would be a good example of what NOT to write).**

|  |
| --- |
| **Part 5. Technical documentation** |

This main report is supplemented with technical documentation. This provides more detail on the software that was used in the experiments, including program listings, a user guide and circuit designs. This section appears on the electronic medium that accompanies this report.

The CD (or DVD, or flash disk) is organized into the directories listed below.

*Main report*

**Use this text**

*Part 5: Technical documentation*

*Software*

*References*

*Datasheets*

*Author*

**indicate the actual directories on your CD here. The minimum required directories are listed in Appendix 4 to the study guide.**

*Datasets/Raw*

*Datasets/Final*