Heliyon

Received:
3 May 2018
Revised:
28 August 2018
Accepted:
21 November 2018

Cite as: Sufyan Samara, Emad Natsheh. Modeling the output power of heterogeneous photovoltaic panels based on artificial neural networks using low cost microcontrollers. Heliyon 4 (2018) e00972. doi: 10.1016/j.heliyon.2018.e00972

Modeling the output power of heterogeneous photovoltaic panels based on artificial neural networks using low cost microcontrollers

Sufyan Samara*, **Emad Natsheh**

Department of Computer Engineering, An-Najah National University, Nablus, Palestine

* Corresponding author.

E-mail address: sufyan_sa@najah.edu (S. Samara).

Abstract

Many implementations of artificial neural networks have been reported in scientific papers. However, few of these implementations allow the direct use of off-line trained networks. Moreover, no implementation reported the use of relatively small network adequate to run on low cost microcontroller. Hence, this work, which presents a small artificial neural network, which models the output power of heterogeneous photovoltaic panel. In addition, the work discuss the hardware implementation that allows such network to run on low cost microcontroller. The hardware implementation has the ability to model heterogeneous photovoltaic panel's output power with very high accuracy and fast response time. Feedforward back propagation has been used because of its high resolution and accurate activation function. Real-time measured parameters can be used as inputs for the developed system. The resulting hardware data is tested with data from real photovoltaic panels; to confirm that it can efficiently implement the models prepared off-line with Matlab. The comparison revealed the robustness of the proposed heterogeneous photovoltaic model system at different conditions. The proposed heterogeneous photovoltaic model system offer a proper and

https://doi.org/10.1016/j.heliyon.2018.e00972

efficient tool that can be used in monitoring photovoltaic panels, such as the ones used in smart-house applications.

Keywords: Electrical engineering, Energy engineering, Computer science

1. Introduction

Many developing countries depend on other countries for more than 80% of their electricity import. Taking the Palestinian Territories as an example; although they have the lowest total energy consumption in the region (0.79 MW h/inhabitant) but they have the highest energy cost in the Middle East. On the other hand, the Palestinian Territories, as many of other developing counties, have high solar energy potential, about 3000 sunshine hours per year. This encourages the research to take advantage of the solar energy for different applications [1, 2].

Photovoltaic (PV) panels are one of the main sources for power generation from solar energy. For solar electricity providers, it is essential to accurately predict solar power generation for demand planning in an electrical grid. However, this prediction is a challenging task because solar power generation is weather dependent. So, with the incremental power generation from solar energy, the prediction of PV panel's power generation has gained big attention [3].

In the literature, different numerical models have been developed to forecast weather and solar power. Among them, Heinemann et al. [4] presented different approaches to solar irradiance forecasting on different time scales. Maini et al. [5] developed and implemented a system for forecasting maximum and minimum temperatures for 12 locations in India based on the perfect prog method (PPM) approach. However, these models require a robust computing system and afterwards they proved to be unstable in handling the various changes in weather conditions.

Besides the conventional methods [4, 5, 6], there are some more advanced techniques like the genetic algorithms, and the artificial neural networks (ANNs), which has the ability to represent the nonlinear systems with high efficiency [7, 8]. With neural networks (NNs), any continuous nonlinear function can be approximated with one or more hidden layers. More, NNs have a parallel architecture that is composed of many processing elements with a simple structure, which is useful for implementation using embedded systems.

This give rise to the usage of NNs to forecast various criteria [9]. In [10, 11] the authors developed an ANN model to predict solar radiation. The results showed the feasibility of this approach for forecasting solar radiation. Chen et al. [12] amended the ANN to forecast the power generated from solar panels. Devi et al. [13] applied the ANN for temperature forecasting, and find out that this model can be used in

https://doi.org/10.1016/j.heliyon.2018.e00972

complex modeling for different factors. The authors in [14, 15] mentioned that feed-forward neural network (FFNN) is the best model of neural network for real-time forecasting due to the least training time and fast response. In [16, 17] the authors applied four types of neural networks for prediction of photovoltaic power output. Their research was based on FFNN, general regression neural network (GRNN), recursive neural network (RNN), and a gamma memory (GM). Based on their research, the FFNN has shown a better performance comparing with the other topologies.

Although there has been research into the prediction of some parameters such as temperature and solar radiation [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21], there has not been comprehensive research into the hardware implementation of the neural network in PV-systems [22, 23, 24]. Mekki et al. [25] implemented a PV panel based on ANNs and VHDL language. They used two hidden layers during their implementation. The used ANN network is not small, which make it unsuitable to run on low cost microcontroller as this will increased the execution time. The implementation using VHDL assumes the use of an FPGA. Although FPGAs are known for their speed due to parallel execution, their cost and power consumption is considered high when compared to low cost microcontrollers. Baptista et al. [26], also used FPGA for implementing ANNs to predict the energy production of a photovoltaic system. The results show that the PV panel can be accurately modeled based on data from a nearby meteorological installation and the hardware implementation produces precise results. Algarín et al. [27], implemented a low-cost maximum power point tracking system based on neural network inverse model controller. Results demonstrated that the proposed implemented model tracks the maximum power point of a PV panel in a more efficient way than the traditional P&O algorithm. Their main novelty was the use of the inverse neural network based on the low cost platform with a buck converter as a control device.

In summary, all previous studies have shown that most of the implementation tryouts tackled only one type of PV panels with no regard to different manufactured PV panels. In addition, these tryouts uses expensive systems, such as FPGAs. In this paper, we present a PV panel's output power model based on small artificial neural network. We also present a real and efficient hardware implementation of the model on low cost microcontroller. Moreover, we show that the used approach has the ability to model heterogeneous PV (HPV) panels' output power with very high accuracy and fast response time. This allows the hardware implementation to be used as a module in a real-time PV panels monitoring system. The paper is organized as follows: section 2 discusses the mathematical model of the PV panel and presents the database used in this study. Section 3 provides the ANN architecture used to model the HPV panels; also, it describes the implementation of the HPV-ANN topology using the ATmega2560 microcontroller. Results and discussion are presented in section 4. Finally, section 5 concludes the work.

https://doi.org/10.1016/j.heliyon.2018.e00972

2. Theory/Calculation

To design and evaluate the performance of a photovoltaic panel, an accurate PV model should predict reliable P-V characteristic curves under different operating conditions [28, 29]. Hence, the model used in this study is based on the dynamic PV model developed and validated in our previous work [28, 29, 30]. The circuit, see Fig. 1, is mainly consisted of a photocurrent, diode, parallel resistor, and a series resistor. The photocurrent ($\boldsymbol{I_{gc}}$) is proportional to the light falling on the cell. The solar cell is not active, during darkness; it works as a diode [31].

The current of the PV cell is model as shown in Eq. (1) [30]:

$$I_{pv} = I_{gc} - I_o\left[\exp\left(\frac{V_{pv} + I_{pv}R_s}{V_t}\right) - 1\right] - \frac{V_{pv} + I_{pv}R_s}{R_P} \tag{1}$$

Where $\boldsymbol{V_{pv}}$ and $\boldsymbol{I_{pv}}$ are the output voltage (V) and current (A) of the solar cell, respectively, $\boldsymbol{I_o}$ is the diode saturation currents (A), $\boldsymbol{I_{gc}}$ is the photocurrent (A), $\boldsymbol{R_p}$ is the shunt resistance (U), $\boldsymbol{R_s}$ is the series resistance (U), and $\boldsymbol{V_t}$ is the diode thermal voltage that can be given as shown in Eq. (2) [30]:

$$V_t = \frac{aKBT_C}{q} \tag{2}$$

Where $\boldsymbol{KB}$ is Boltzmann's constant ($1.38065e^{-23}$ J/K), $\boldsymbol{a}$ is the diode ideality factor that represents the components of diffusion current, $\boldsymbol{q}$ is the electron charge ($1.60217e^{-19}$ C), and $\boldsymbol{T_C}$ is the cell temperature (K). The proposed model [28, 29, 30] is implemented as shown in Fig. 2.

As mentioned previously, the performance of a PV model can be described as a function of solar radiation and ambient temperature. The daily average ambient temperature and solar radiation data for Nablus were collected from An-Najah Energy Research Center [32].

The distribution of solar irradiation and ambient temperature throughout the day for Nablus is illustrated in Fig. 3.

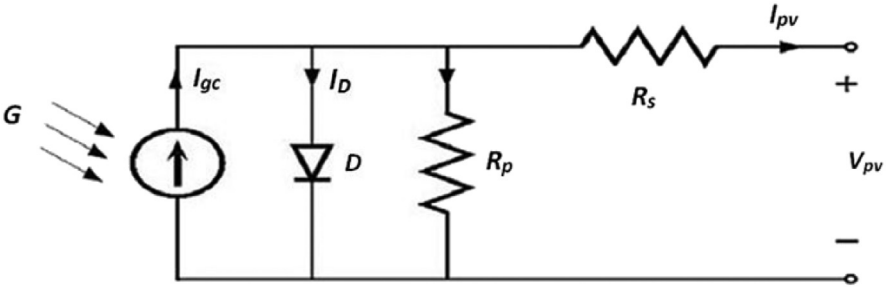

Fig. 1. Single diode PV cell equivalent circuit.

https://doi.org/10.1016/j.heliyon.2018.e00972

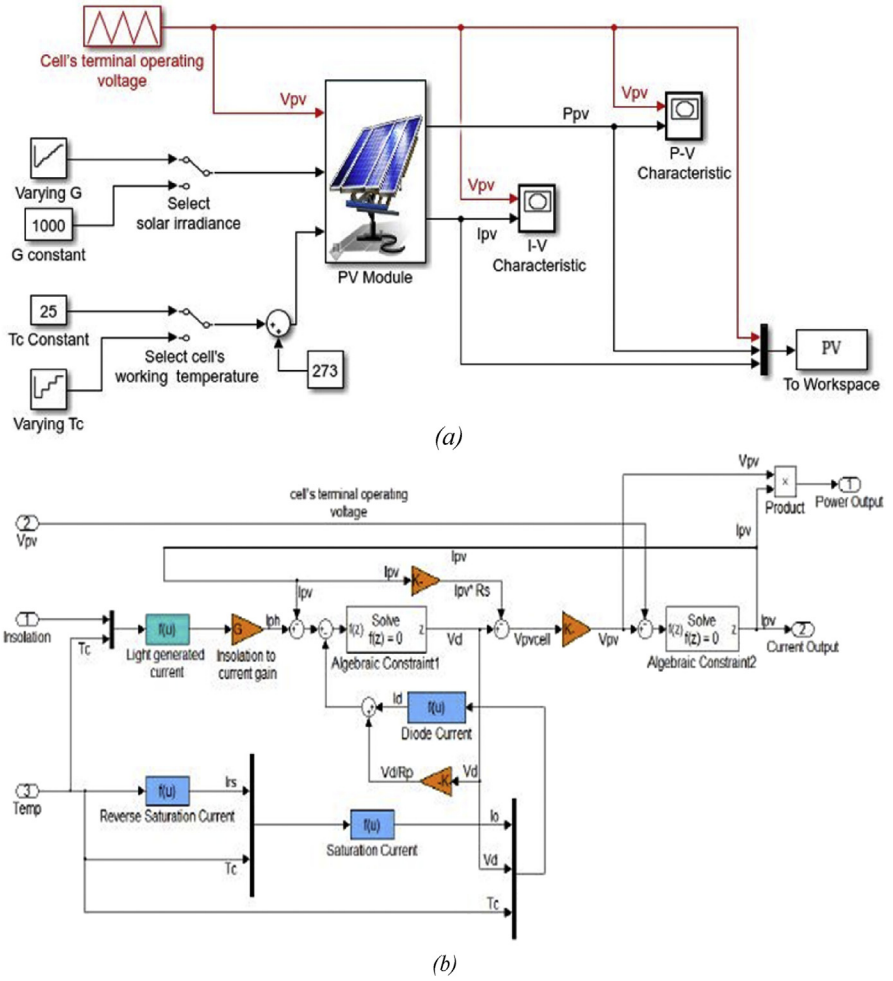

Fig. 2. (a) Implementation of the PV model (b) Subsystem implementation of the PV model.

3. Model

The conversion efficiency of solar energy is related both to the early fault detection and optimal design of PV system. In this section, a simple, accurate, and fast evolutionary model is proposed for the power energy output forecasting of a heterogeneous PV panel based on artificial neural network using low cost microcontroller. This make the model adequate to be used as a module in heterogeneous PV panels power planning and real-time monitoring systems.

3.1. The proposed ANN topology

An artificial neural network is an adaptive system that changes its structure based on the information that flows through the network. Their ability to learn from experimental data makes neural networks very powerful and flexible than any other

https://doi.org/10.1016/j.heliyon.2018.e00972

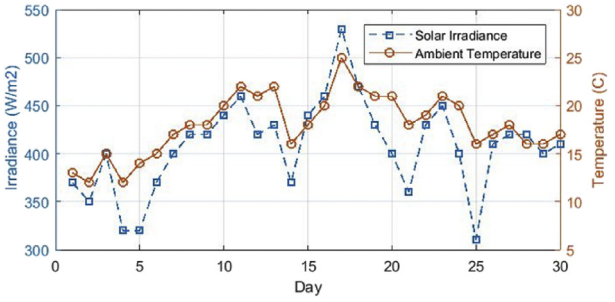

Fig. 3. Solar radiation and temperature distributions during March 2016.

parametric techniques. Therefore, ANN have been used widely for solving classification problems in different application fields [17].

The feedforward neural network was the first and simplest type of neural network developed. It consists of many connected processing nodes known as neurons. All these neurons carry out the same operation, as shown in Fig. 4. First, the neuron computes the weighted sum of the input signals (see Equation 3). Then they apply the result to the activation function.

$$X_j = \left[\sum_{i=1}^{N} x_{ij} w_{ij}\right] - b_j \tag{3}$$

The activation function limits the output of the neuron, usually between the values [-1, +1] or [0, 1]. Each layer in a feedforward neural network has its own specific function. The most commons activate functions are the step function, the linear combination, and the sigmoid function [33]. The topology of the proposed FFNN is shown in Fig. 5. The FFNN is used to model a heterogeneous PV panel output power and approximate the generated power.

As illustrated in Fig. 5, the proposed model composed of three layers. The first layer (input layer) consists of three input neurons, which are the cell temperature ($\boldsymbol{T_C}$), the

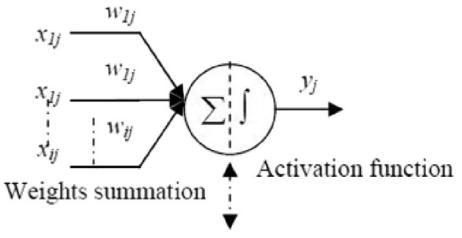

Fig. 4. Schema of artificial neuron.

https://doi.org/10.1016/j.heliyon.2018.e00972

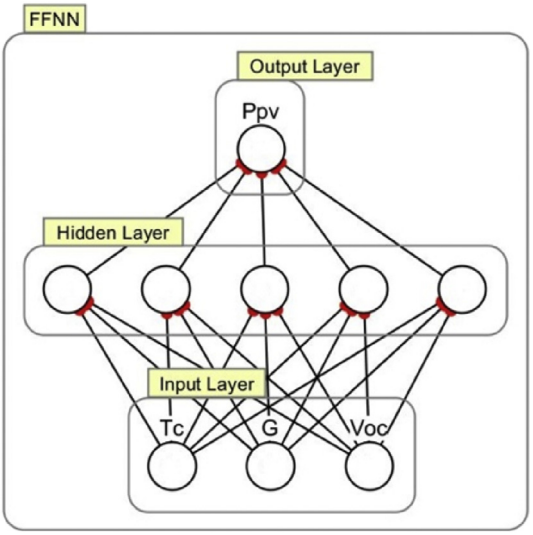

Fig. 5. Architecture of FFNN.

solar radiation ($\boldsymbol{G}$), and the cell's open circuit voltage ($\boldsymbol{V_{OC}}$) at a 1 kW/m^2 and 25 °C. The middle layer (hidden layer) consists of five neurons, whose activation function is of the sigmoid type. The output layer consists of one neuron that is the PV power output, whose activation function is the linear function.

Once the supervised FFNN model is constructed, the training data are gathered and fed into the model. The FFNN is trained, by Levenberg Marquardt (LM) training algorithm [28,33], to find the relationships between the training parameters. In LM algorithm, the processing element's thresholds and the interlayer connection weight are first initialized at small random values. Then a training patterns set is presented to the network. Each set consists of one output ($\boldsymbol{P_{pv}}$), and three inputs ($\boldsymbol{T_C}$, $\boldsymbol{G}$, and $\boldsymbol{V_{OC}}$). A MatLab code is implemented to generate the PV power, by analyzing the non-linear characteristics of a validated PV model [28, 29, 30]. The temperature and solar irradiance data are gathered from An-Najah Energy Research Center [32]. Before getting start with training the FFNN, a training data set of 1960 cases were collected from two different PV panels; at different conditions. The training data were given repeatedly to the proposed model, and an adjustment was made after each iteration; if the real output is different from the desired output. The cell's $\boldsymbol{V_{OC}}$ is used as a reference parameter to select between the two PV panels. Table 1 shows the manufacturer specifications for two different PV panels under standard testing condition (STC).

3.2. Implementation on low cost microcontroller

The Heterogeneous PV ANN (HPV-ANN) topology was implemented using the ATmega2560 microcontroller. This microcontroller can be found on many kits

https://doi.org/10.1016/j.heliyon.2018.e00972

Table 1. PV panels specification (1 kW/m^2, 25 °C).

Sharp's NUS0E3E		**Astronergy CHSM6610P-225**	
Maximum power (P_m)	180 W	*Maximum power (P_m)*	225 W
Voltage at *Pm* (V_{amp})	23.7 V	Voltage at *Pm* (V_{amp})	29.76 V
Current at *Pm* (I_{amp})	7.6 A	Current at *Pm* (I_{amp})	7.55 A
Open circuit voltage (V_{oc})	30 V	Open circuit voltage (V_{oc})	36.88 V
Short circuit current (I_{sc})	8.37 A	Short circuit current (I_{sc})	8.27 A
Temp coefficient for V_{oc}	−104 mV/°C	Temp coefficient for V_{oc}	−0.129 V/°C
Temp coefficient for I_{sc}	+0.053%/°C	Temp coefficient for I_{sc}	+0.052%/°C

including DIY Bare Minimum Mega 2560 and the famous Arduino mega 2560. The ATmega2560 [34] is an 8-bit AVR RISC-based microcontroller which combines 256KB ISP flash program memory, 8KB SRAM, 4KB EEPROM, and 86 general purpose I/O lines. Many AVR microcontrollers including the ATmega2560 allows the usage of the program memory to store and access data. Although the ATmega2560 microcontroller is used in this implementation, other microcontrollers can be used even with less specifications. This is possible because the implementation methodology used in this paper does not require a powerful microcontroller but it requires a 64KB memory. However, even if such memory is not included inside a microcontroller, an external relatively cheap serial 64KB flash memory can be easily connected to virtually any microcontroller. We used the Atmel Studio 7.0 software to carry out our implementation. In the following subsections we will discuss how the HPV-ANN explained in section 3.1 was implemented using the ATmega2560 microcontroller.

3.2.1. The HPV-ANN layers implementation

The HPV-ANN, shown in Fig. 5, consists of three layers. The first layer is just an input layer to the middle layer. The middle layer consists of five neurons each performing the same operation but with different values, see Fig. 4. The neurons in the middle layer are the most processing extensive neurons in the HPV-ANN. Each neuron in the middle layer performs the calculations defined by the equations given by Eqs. (4) and (5).

$$Y_j = \frac{2}{1+e^{-2X_j}} - 1, \qquad where\, j = 1,2,3,4,5.\, and\, X_j\, is\, given\, by \tag{4}$$

$$X_j = \left[\sum_{i=1}^{3} x_{ij} w_{ij}\right] - b_j\,, \quad where\, j = 1,2,3,4,5. \tag{5}$$

https://doi.org/10.1016/j.heliyon.2018.e00972

Using the training phase, which was carried out using MatLab, see section 3.1, we calculated the constants w_{ij} and b_j, which are shown in Table 2.

Having in mind low cost microcontrollers, the middle layer was implemented according to the block diagram shown in Fig. 6.

The implementation in Fig. 6 uses an optimized memory mapped sigmoid method, discussed in subsection 3.2.2, which was developed solely for the sake of this implementation. This optimized method allows for fast calculation of P_{pv}. The last layer in the HPV-ANN uses the values Y_1 to Y_5 calculated by the middle layer to calculate using the equation shown in Eq. (7).

$$Z = \left[\sum_{j=1}^{5} Y_j V_j\right] + C_1 \tag{6}$$

$$P_{pv} = C_2\left(\frac{Z}{2}\right) + C_3 \tag{7}$$

In the Eqs. (6) and (7) the constants V_1, V_2, V_3, V_4, V_5, C_1, C_2, and C_3 are found using the training phase, which was carried out using MatLab, see section 3.1. The values of these constants are shown in Table 3.

3.2.2. The optimized sigmoid memory mapped implementation

Most of the calculations done inside the neurons, except for the sigmoid function, are simple functions involving simple arithmetic operations. For the HPV-ANN to be implemented on a low cost microcontroller, the sigmoid function need to be either simplified, estimated, or otherwise optimized in a way to allow for an

Table 2. Values for weights and biases in the hidden layer.

Neuron	Weight (w_{ij})	Bias (b_j)
1	$w_{11} = -1.1763$ $w_{21} = 0.3765$ $w_{31} = 3.6974$	$b_1 = -4.2404$
2	$w_{12} = 0.4883$ $w_{22} = 0.4777$ $w_{32} = 2.6601$	$b_2 = 2.2941$
3	$w_{13} = 0.7049$ $w_{23} = 0.2171$ $w_{33} = 2.1340$	b3 = 1.4885
4	$w_{14} = 0.5709$ $w_{24} = 0.1614$ $w_{34} = -3.107$	$b_4 = -2.6729$
5	$w_{15} = 0.3174$ $w_{25} = -0.037$ $w_{35} = 4.9545$	$b_5 = -4.8791$

https://doi.org/10.1016/j.heliyon.2018.e00972

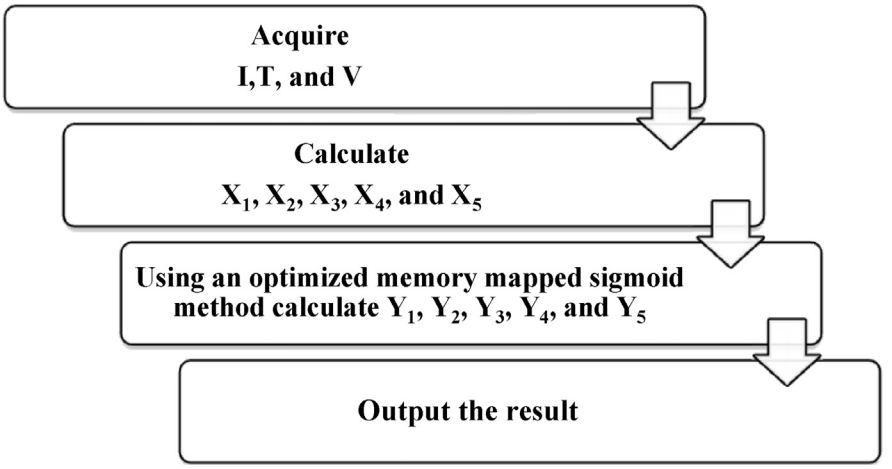

Fig. 6. Block diagram shows how the HPV-ANN middle layer was implemented.

Table 3. Constant values for the last HPV-ANN layer.

Constant	Value	Constant	Value	Constant	Value
V_1	0.2294	V_4	1.8414	C_2	260.6061
V_2	0.1583	V_5	2.0996	C_3	0.03815
V_3	0.5460	C_1	−1.7459		

implementation on a low cost microcontroller. Moreover, the calculation must be efficient with an acceptable fast response. This will enable the HPV-ANN hardware implementation to be included as a module in a real time hardware monitoring system for heterogeneous PV panels.

In our HPV-ANN implementation, the sigmoid function was analyzed for optimization. First, we see that the sigmoid function, shown in Fig. 7, is mirrored around the axis Y = -X. This allows us to implement only one part, namely the positive part, of the function. Any value from the other part of the function, the negative part, can be obtained by simply negate its corresponding mirrored positive value.

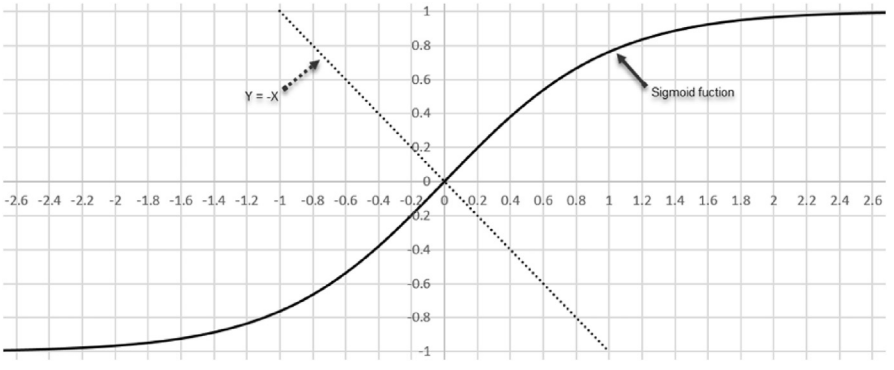

Fig. 7. The sigmoid function.

https://doi.org/10.1016/j.heliyon.2018.e00972

Second, we see that the output values of the positive part of the sigmoid function rage from 0 to 1 for input values range from 0 to 10. To fast evaluate the function on a low cost microcontroller, we implemented the positive part of the sigmoid function using memory mapped or look up table approach. The memory mapped approach requires storing the result of the function, $\boldsymbol{Y_j}$ in Eq. (4), in memory locations addressed by the input of the function, $\boldsymbol{X_j}$ in Eq. (4). However, the sigmoid function output is of type float. According to the IEEE-754 standard [35], each floating point number requires 32-bit. Because the sigmoid function output is between 0 and 1, we need only to store fraction numbers. After performing some testing, based on the MatLab simulation and testing in section 3.1, we noticed that the fraction number needs not to be more than four digits. This means that the output of the sigmoid function is a number that ranges from 0.0000 to 0.9999. Having that in mind, we can store the fraction without the decimal point in memory; that is store the numbers from 0000 to 9999. Whenever a result is obtained, we put back the decimal point. This way we reduce the bits needed to store each number from 32-bits to 14-bits. Noting that each memory location is 8-bit wide in most memories, two memory locations are required to store each number. However, for the sigmoid function the input value may range from 0.0000 to 9.9999. This requires to store 100000 number. Luckily, because we only need the output to have a precision of four digits, the sigmoid function output will converge to 1 when the input value is greater than 2.3999. This allows us to round the sigmoid function output to 1 for any input value greater than 2.3999. By doing that, the required memory size needed to store the output of the function is reduced to 48KB. The sigmoid function is further optimized by noticing that the output of the function is nearly linear when the input value is close to zero. Also, by noticing that many input values are mapped to the same output value. Moreover, to allow for fast access, the sigmoid function outputs are divided and stored in more than one bank. The algorithm used to calculate the sigmoid function is shown in Fig. 8.

4. Results & discussion

4.1. Multi-layer network

A dataset, of 2800 cases, was obtained from two different PV panels, namely: Sharp's-NUS0E3E, and Astronergy-CHSM6610P. Each case is composed of three inputs (G, T_C, V_{OC}), and one output (P_{pv}). The collected dataset was split randomly into three subsets. The size of each set is a percentage value of 70% for training, 15% for validation (to minimize overfitting), and 15% for testing.

During the training, a trial for finding the number of nodes in the hidden layer is evaluated. Table 4 represents the training MSE and regression value for different numbers of hidden nodes for FFNN. Moreover, Fig. 9 shows a graphical visualization of MSE and regression based on the number of nodes in the hidden layer.

https://doi.org/10.1016/j.heliyon.2018.e00972

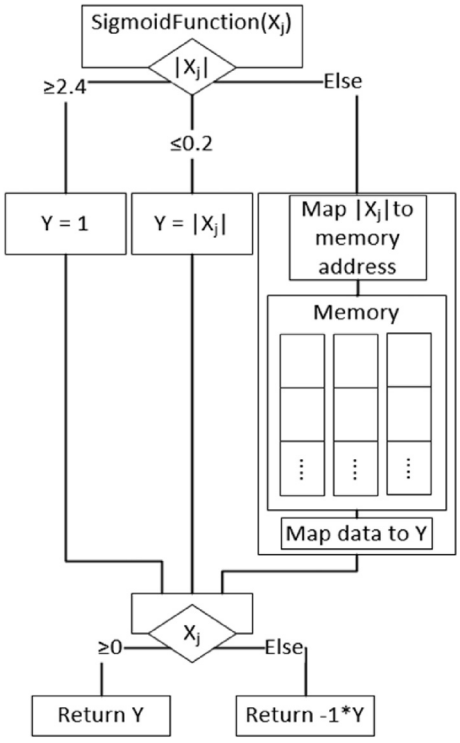

Fig. 8. Memory mapped sigmoid function implementation.

According to the above figure and table (Fig. 9, Table 4), the best validation performance is when the hidden layer has five nodes.

After the training of the FFNN, discussed in Section 3.1, the next step was to test the network for performance and to determine whether the actual results agree with the predicted ones. Using a set of actual 420 cases, the model input parameters were entered consecutively for each case and the predicted output was compared to the corresponding actual obtained PV power. Fig. 10 shows a brief comparison between the predicated and real data under different condition.

As illustrated in Fig. 10, the level of accuracy is very high. Fig. 11 presents the regression of the calculated data and the real data.

Table 4. MSE and regression for FFNN.

Number of neurons in hidden layer	MSE train	Regression train
2	0.25	0.999157
3	0.23	0.999005
4	0.195	0.999778
5	0.179	0.999987
6	0.181	0.999978

https://doi.org/10.1016/j.heliyon.2018.e00972

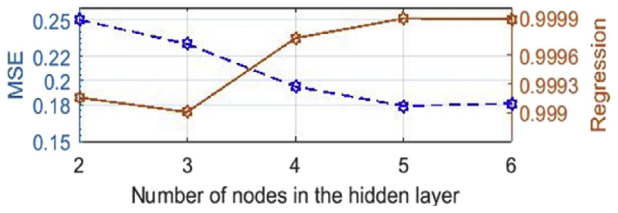

Fig. 9. MSE and regression based on the number of nodes in the hidden layer.

4.2. Implementation on microcontroller

The implementation was tested using the same input data set against the real data obtained from the PV panels, Fig. 12. The input data was entered using a mobile phone application connected through Bluetooth module to the ATmega2560 microcontroller. The results was probed back along with the processing time through the Bluetooth to the mobile application. Depending on the input data, the calculation time required to obtain a result varies from 0.8 ms to 1.1 ms. The variation in calculation time is due to the optimized implementation and of the sigmoid function, which is discussed in section 3.2.2. For example if a specific input to the sigmoid function is located on the linear part of the sigmoid function, the output is calculated very fast. On the other hand, if that input is located on memory mapped part of the sigmoid function, more time will be needed depending on which bank and which location that output resides on. In either cases the best and worst time to obtain a result varies from 0.8 ms to 1.1 ms respectively.

Fig. 12 shows the power obtained from real PV panels for 24 sample sets of I, T, and V. The x-axis shows the sample set number, the y-axis on the left shows the power

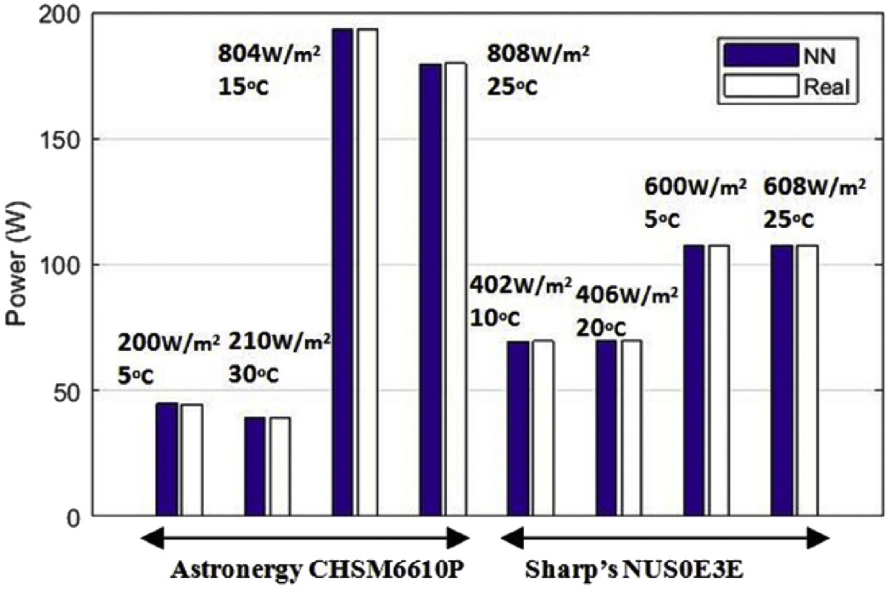

Fig. 10. Comparison between the predicated and real data at different conditions.

https://doi.org/10.1016/j.heliyon.2018.e00972

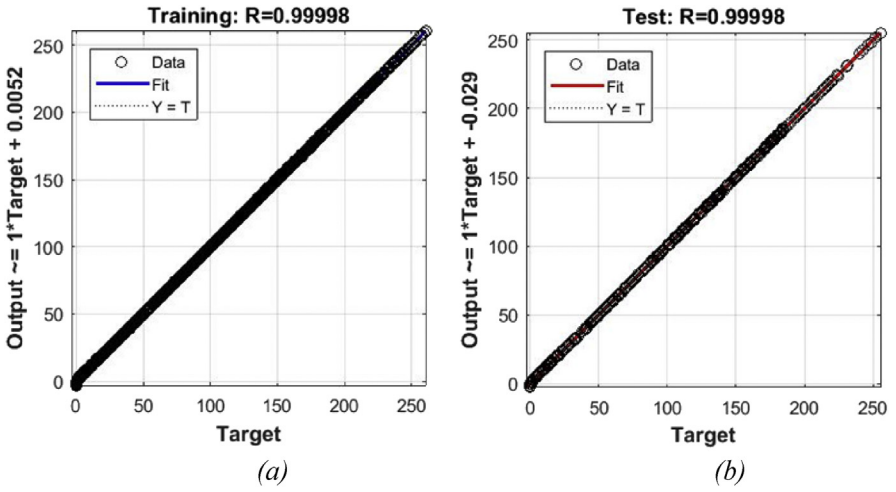

Fig. 11. FFNN regression of error (a) Training dataset (b) Testing dataset.

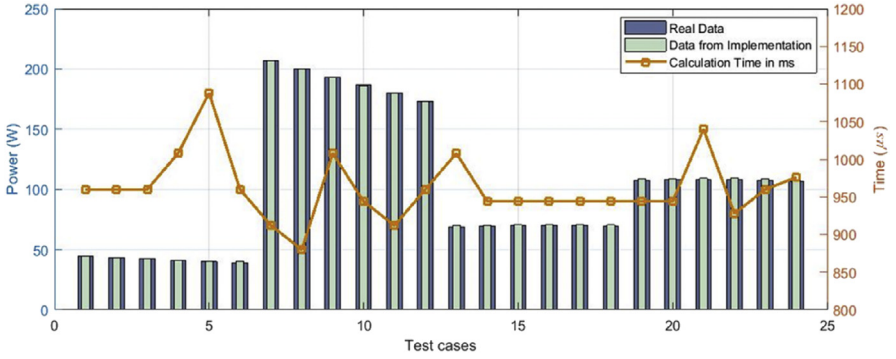

Fig. 12. PV panels Power obtained from real PV panels against Power obtained from the microcontroller implementation.

obtained for each sample set. The right y-axis shows the time, in microsecond, required to get the result of each sample set using the module, which was implemented on ATmega2560 microcontroller. We can see that the results obtained by the implemented module are very close, if not the same, to the results obtained from the real PV panels. Each result requires an average calculation time of 0.9 ms with an error of less that $\pm 0.1\%$.

5. Conclusions

Power prediction for PV panels is needed for accurate power planning and monitoring. Hence, in this paper, we presented an efficient hardware implementation that models the output power of heterogeneous PV panels. The hardware implementation was made possible by developing a very small artificial neural network, which accurately modeled the output power of heterogeneous PV panels. The presented

https://doi.org/10.1016/j.heliyon.2018.e00972

hardware model showed high accuracy and fast response time, which enables it to be included as a module in a real-time PV panels monitoring system.

The power of PV panels have been generated from an applied MatLab code, which analyses the output P-V characteristics of the validated PV model. Afterward, the estimated power and meteorological data have been used for training the proposed FFNN.

Simulation results were obtained by the implementation of the HPV-ANN topology using the ATmega2560 microcontroller. Result shows that the proposed module exhibits excellent performance under variable atmospheric conditions. The use of ANN allows the system to be easily updated to adapt for future PV systems. This is possible by updating the new weights after recalculating and offline training.

Declarations

Author contribution statement

Sufyan Samara, Emad Natsheh: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Funding statement

This work was supported by An-Najah National University (ANNU379MoHE-1819-Sc005).

Competing interest statement

The authors declare no conflict of interest.

Additional information

No additional information is available for this paper.

References

[1] A. Juaidi, F.G. Montoya, I.H. Ibrik, F. Manzano-Agugliaro, An overview of renewable energy potential in Palestine, Renew. Sustain. Energy Rev. 65 (2016) 943−960.

[2] E.M. Natsheh, Power generation of solar PV systems in Palestine, Appl. Sol. Energy 52 (2016) 193−196.

https://doi.org/10.1016/j.heliyon.2018.e00972

[3] L. Martin, L.F. Zarzalejo, J. Polo, A. Navarro, R. Marchante, M. Cony, Prediction of global solar irradiance based on time series analysis: application to solar thermal power plants energy production, Sol. Energy 84 (2010) 1772−1781.

[4] D. Heinemann, E. Lorenz, M. Girodo, Forecasting of solar radiation, Proc. IEEE Int. Symp. Ind. Electron. (2008) 1537−1541.

[5] P. Maini, A. Kumar, L.S. Rathore, S.V. Singh, Forecasting maximum and minimum temperatures by statistical interpretation of numerical weather prediction model output, Weather Forecast. 18 (2003) 938−952.

[6] R. Perez, K. Moore, D. Wilcox, S. Renńe, A. Zelenka, Forecasting solar radiation-preliminary evaluation of an approach based upon the national forecast database, Sol. Energy 81 (2007) 809−812.

[7] S. Singh, P. Bhambri, J. Gill, Time series based temperature prediction using back propagation with genetic algorithm technique, IJCSI Int. J. Comput. Sci. Issues 8 (2011) 28−32.

[8] A. Mellita, A.M. Pavan, A 24-h forecast of solar irradiance using artificial neural network: application for performance prediction of a grid-connected PV plant at Trieste, Italy, Sol. Energy 84 (2010) 807−821.

[9] Z. Wang, F. Wang, S. Su, Solar irradiance short-term prediction model based on BP neural network, Energy Proc. 12 (2011) 488−494.

[10] E.A. Ahmed, M.E.-N. Adam, Estimate of global solar radiation by using artificial neural network in Qena, Upper Egypt, J. Clean Energy Technol. 1 (2013) 148−150.

[11] T. Khatib, A. Mohamed, K. Sopian, M. Mahmoud, Assessment of artificial neural networks for hourly solar radiation prediction, Int. J. Photoenergy 2012 (2012) 1−7.

[12] C. Chen, S. Duan, T. Cai, B. Liu, Online 24-h solar power forecasting based on weather type classification using artificial neural network, Sol. Energy 85 (2011) 2856−2870.

[13] C.J. Devi, B.S.P. Reddy, K.V. Kumar, B.M. Reddy, N. Raja, ANN approach for weather prediction using back propagation, Int. J. Eng. Trends Technol. 3 (2012) 19−23.

[14] K.O. Alawode, M.O. Oyedeji, A comparison of neural network models for load forecasting in nigerian power system, Int. J. Renew. Energy Technol. 2 (2013) 218−222.

 https://doi.org/10.1016/j.heliyon.2018.e00972

[15] A. Linares-Rodríguez, J.A. Ruiz-Arias, D. Pozo-Vázquez, J. Tovar-Pescador, Generation of synthetic daily global solar radiation data based on ERA-Interim reanalysis and artificial neural networks, Energy 36 (2011) 5356−5365.

[16] A. Saberian, H. Hizam, M.A.M. Radzi, M.Z.A. Ab-Kadir, M. Mirzaei, Modelling and prediction of photovoltaic power output using artificial neural networks, Int. J. Photoenergy 2014 (2014) 1−10.

[17] V.L. Brano, G. Ciulla, M.D. Falco, Artificial neural networks to predict the power output of a PV panel, Int. J. Photoenergy 2014 (2014) 1−12.

[18] L. Miloudi, D. Acheli, M. Kesraoui, Application of artificial neural networks for forecasting photovoltaic system parameters, Appl. Sol. Energy 53 (2017) 85−89.

[19] S. Hamid Oudjana, A. Hellal, I. Hadj Mahammed, Neural network based photovoltaic electrical forecasting in south Algeria, Appl. Sol. Energy 50 (2014) 273−277.

[20] M. Awad, I. Qasrawi, Enhanced RBF neural network model for time series prediction of solar cells panel depending on climate conditions (temperature and irradiance), Neural Comput. Appl. (2016).

[21] R. Muhammad Ehsan, S.P. Simon, P.R. Venkateswaran, Day-ahead forecasting of solar photovoltaic output power using multilayer perceptron, Neural Comput. Appl. 28 (2017) 3981−3992.

[22] J.P. Shankarrao, FPGA Implementation of Maximum Power Point Tracking Algorithm for PV System (master's thesis), National Institute Of Technology Rourkela, India, 2013.

[23] N. Khaldi, H. Mahmoudi, M. Zazi, Y. Barradi, Implementation of a MPPT neural controller for photovoltaic systems on FPGA circuit, WSEAS Trans. Power Syst. 9 (471) (2014) 478.

[24] H. Mekki, A. Mellit, S.A. Kalogirou, A. Messai, G. Furlan, FPGA-based implementation of a real time photovoltaic module simulator, Prog. Photovoltaics 18 (2010) 115−127.

[25] H. Mekki, A. Mellit, H. Salhi, K. Belhout, Modeling and simulation of photovoltaic panel based on artificial neural networks and VHDL-language, in: Proceedings of the 4th International Conference on Computer Integrated Manufacturing (CIP '07), 2007, pp. 1−5. November 2007.

[26] D. Baptista, S. Abreu, C. Travieso-González, F. Morgado-Dias, Hardware implementation of an artificial neural network model to predict the energy production of a photovoltaic system, Microprocess. Microsyst. 49 (2017) 77−86.

https://doi.org/10.1016/j.heliyon.2018.e00972

[27] C.R. Algarín, D.S. Hernández, D.R. Leal, A low-cost maximum power point tracking system based on neural network inverse model controller, Electronics 7 (2018) 1−17.

[28] E.M. Natsheh, A.R. Natsheh, A.H. Albarbar, An automated tool for solar power systems, Appl. Sol. Energy 50 (2014) 221−227.

[29] E.M. Natsheh, A. Albarbar, Photovoltaic model with mpp tracker for standalone/grid connected applications, in: IET Conference on Renewable Power Generation (RPG 2011), 2011, pp. 1−6. Sept. 2011.

[30] E.M. Natsheh, A.R. Natsheh, A.H. Albarbar, Intelligent controller for managing power flow within standalone hybrid power systems, IET Sci. Meas. Technol. 7 (2013) 191−200.

[31] G. Walker, Evaluating MPPT converter topologies using a MATLAB PV model, J. Electr. Electron. Eng. 21 (2001) 49−56.

[32] Energy Research Center, 2017. https://www.najah.edu/en/community/scientific-centers/.

[33] M. Negnevitsky, Artificial Intelligence: a Guide to Intelligent Systems, Addison Wesley, Essex, 2004.

[34] The Atmega2560 datasheet. http://www.microchip.com/wwwproducts/en/atmega2560.

[35] 754-2008 - IEEE Standard for Floating-Point Arithmetic, 2008 online.

https://doi.org/10.1016/j.heliyon.2018.e00972