FPGA Based Meteorological Monitoring Station

Víctor Asanza[1], Rebeca Estrada Pico[1,2], Danny Torres[2], Steven Santillan[1], Juan Cadena[1]
[1] Facultad de Ingeniería en Electricidad y Computación, FIEC
[2] Centro de Tecnologías de Información, CTI
Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, 090150 Guayaquil, Equator
{vasanza, restrada, daaltorr, steisant, jpcadena}@espol.edu.ec

***Abstract*— In this paper, we propose to implement a meteorological monitoring station using embedded systems. This model is possible thanks to different sensors that enable us to measure several environmental parameters, such as i) relative humidity, ii) average ambient temperature, iii) soil humidity, iv) rain occurrence, and v) light intensity. The proposed system is based on a field-programmable gate array device (FPGA). The proposed design aims at ensuring high-resolution data acquisition and at predicting samples with precision and accuracy in real-time. To present the collected data, we develop also a web application with a simple and friendly user interface.**

Keywords—Sensor, Meteorological Station, Humidity, Temperature, Soil Moisture, Rain, Embedded System, FPGA, Microprocessor, Nios-II, ARM, Linux Web Server, DHT11 Sensor, FC-28 Hygrometer, FC-37 Sensor, LDR Sensor

I. INTRODUCTION

In general, the lack of new state-of-art technology in developing countries such Ecuador makes difficult to automate certain type of systems. The area of meteorology is not an exception, owing to the fact that there is still some obsolete equipment and procedures to obtain data and to carry out its respective analysis. To address this problem, National Institute of Meteorology and Hydrology (INAMHI) launched a program to replace conventional weather stations using new technology that allows the capture of information and its management [1]. Nevertheless, this renewal process requires considerable time to be implemented throughout the entire country.

According to [1], Equator has 519 weather stations, of which 41% are rain gauges, 29% are used for climatology and, only 2% are used for agro-meteorology. This means that approximately 13 stations are only available for the area of our interests. It should be noticed that the large meteorological stations are located in the southern part of the country. The vast majority of them are out of service because of either their obsolete devices or insufficient technological and human resources to monitor and control them.

On the other hand, there are various types of solutions for climate monitoring [2]. The use of static stations is the most classic solution. Each static station collects the information and then sends it to the main station, which is responsible for handling that information [3]. This methodology implies constant maintenance and monitoring. Another solution focuses on the use of wireless sensor networks (WSN). Each sensor node is responsible for capturing and reporting the information to the main station. Currently, these static stations are becoming manual and automated stations that are easy to operate. [4, 5]

For real-time monitoring applications, recent works have enhanced up to 5 times the read and write time of data in FPGA SRAM memory vs. DDR3 memory with an ARM processor [12]. Reducing the memory accessing times can help to reduce the processing response time of complex prediction algorithms [13]. On the other hand, certain FPGAs (e.g. Cyclone V) have some limitations in terms of the number of configurable logical blocks (CLB) and do not allow us taking advantage of all the benefits of Deep Learning algorithms. These FPGAs are mainly for academic purposes. We believe that their use may be justified for processes that require fast response time and classification/ prediction algorithms requiring reduced memory access.

In this paper, we present a system based on Field-Programmable Gate (FPGA) matrix. We build the system through block implementation and low-level programming to carry out the processing and analysis of different meteorological variables. These variables are measured and collected by means of sensors connected to an ARDUINO UNO and its serial interface sends them to the FPGA. By doing so, we ensure high-resolution when capturing and processing the information, which is stored and then it can be displayed using a web application.

Designing a micro meteorological station with a simple and practical implementation can bring other benefits such as acceptable cost, portability, usability, speed-up the processing the time for monitor and management, among others.

This paper is organized as follows: Section II describes the related work; section III explains the data acquisition procedure; section IV describes the methodology and system model; section V shows the results obtained using the proposed method. Finally, Section VI presents the conclusions and future work.

II. RELATED WORK

Nowadays, the use of technology in the industrial and production area is essential, especially in agriculture. Technology has been widely used in this area not only to improve productivity but also to reduce human efforts. Some research focused on the use of technology to propose automated processes or systems for farmers to improve their production yield. The research is still on and, in this section; we revise some of the prior implementations.

M. Haefke et al. proposed an intelligent detection system based on ZigBee, to monitor variables such as temperature, pressure, sunlight intensity, and humidity. Their proposed intelligent weather station has design features such as low cost, the flexibility of handling, the accuracy of the

information, low consumption among others [6]. Similarly, G. Mendez et al. developed a system that monitored the same variables and added the air pressure. Their proposal uses Wi-Fi to transfer, to store these environmental variables in several nodes, and to exchange information between them. This intelligent system is low cost, with little wiring and equipped with the possibility of powering through solar panels [7]. Other researchers proposed to implement a similar system based on FPGA considering humidity and light intensity as their main variables in order to maximize the production while guaranteeing the accuracy by means of neural networks and other factors environmental [8, 9].

Some researchers used machine learning to predict the futures values of the analyzed variables, such as the design of an Artificial Neural Network, ANN, based on FPGA for the prediction of Solar Radiation Data from sunlight duration and temperature. This architecture was developed with VHDL, and as a result, the embedded system allows predicting the solar radiation using the average temperature and daylight duration [10].

Similarly, several researchers carried out other implementations with the same objective. For instance, solar power prediction model based on various satellite images and a support vector machine (SVM) learning scheme was proposed in [11]. This model was compared to the conventional time-series model and an artificial neural network (ANN) model in terms of prediction accuracy, using variables as the amount of clouds and irradiance. Nevertheless, the SVM Model does not work very well for dataset if target classes overlap. In addition, it is not suitable for large data sets due to the long training time. For this reason, it underperforms when the number of features for each data point exceeds the number of training data samples.

III. High-Resolution Data Acquisition

To generate our high-resolution dataset, we used four different sensors to acquire data from three analog signals and two digital signals. The signals' values are sampled each 30 seconds and saved as CSV file with the Unicode Transformation Format of 8-bit bytes (UTF-8). The collected information corresponds to weather activity of a plant placed in the backyard of a house in Guayaquil. These measurements were collected during a period of the three days (i.e Dec 6 to Dec 8).

The sensors were connected to an Arduino UNO in order to send the data through a serial connection to a Workstation running Windows 10 and open-source software named Tera Term [15] during the two days. On the third day, we replaced the workstation with a 4GB RAM Raspberry Pi 4 B running Debian-based OS Raspbian Buster and other open-source graphical software to acquire the data [16]. Table I presents the details of the variables in our dataset.

A. Sensors

We used four sensors to collect the data and they are briefly described as follows:

- DHT-11 is a digital sensor that allows us to obtain two variables environment temperature and relative humidity. This sensor provides floating values of temperature by its embedded thermistor and humidity by a capacitive humidity sensor. It can get new data every 2 seconds and it has good accuracy for both variables.
- FC-28 Soil Hygrometer measures soil moisture, working as a variable resistor in terms of conductivity that depended on the amount of water in the soil of the plant. The output comes from an analog pin that works as another column from the dataset.
- FC-37 Rain Sensor is set up by an electronic board built-in a variable resistor by its conductivity in terms of how much water contact it gets, and its output is referenced by a voltage that gets mapped by two thresholds that define its state: rain (0), drizzle (1) and no rain (2).
- LDR Lux meter built from resistor and Photo resistor that establish the intensity of light that receives the plant a moment of the day. The analog output is a voltage that is processed and converted by electrical and optical equations into a float value representing the quantity of light over the plant.

B. Web Application

To access our web application, we should be connected within the same network. Through the web application, we can visualize in real-time the variables measured for the monitored environment.

TABLE I. Dataset

Variable	Description	Range
Relative Humidity	Sensor (DHT11) is used to measure the environment humidity using oneWire.communication.	[42 – 69]
Environment Temperature	Sensor (DHT11) is used to measure the environment temperature using oneWire.communication.	[27.1 - 32.9]
Soil moisture	Sensor (FC28) generates an analog signal as a function of the soil moister. Sampled signal is digitized with 10 bits.	[434 – 1000]
Light intensity	Light dependent resistor (LDR) generates an analog signal proportional to the intensity of incident light. Sampled signal is digitized with 10 bits.	[119.76-9582]
Rain Occurrence	Digital sensor (FC-37) detects if there is rain (0), drizzle (1) and no rain (2).	[0 - 2]

IV. Methodology and System Model

The proposed architecture aims at building a basic embedded system, which incorporates components such as System ID, Arria V/Cyclone V Hard Processor System, NIOS II Processor, Interval Timer, Avalon, and On-Chip Memory. The proposed architecture is shown in Figure 1.

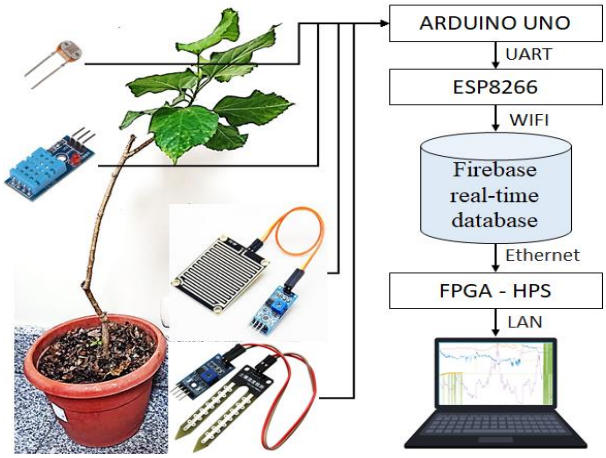

Fig. 1: Architecture of embedded system

The system consists of several stages as follows: 1) data acquisition, 2) data processing, 3) web application requests and responses, 4) file handling, 5) variable values prediction, and 6) graphical interfaces. The digital and analog sensors are considered as a single node in a WSN. Every 5 seconds in the same way the data set was collected for the previous phase. The flowchart is illustrated in Figure 2.

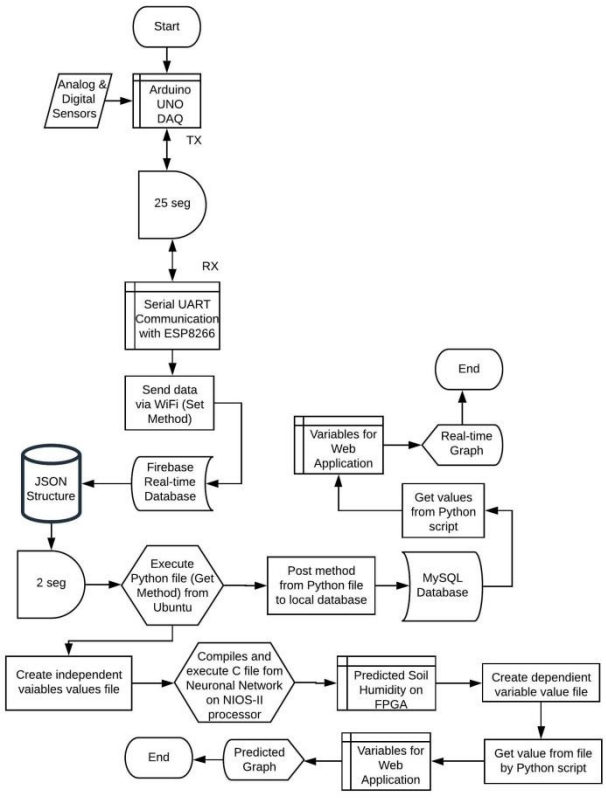

Fig. 2: System flowchart.

The Arduino Uno acts as a transmitter and sends the information through the Serial UART on TX/RX pins to the receiver ESP8266 board every 25 seconds. Thus, the L106 32-bit RISC microprocessor forwards the data through WIFI to a Firebase real-time database.

In Google's platform, data is handled in a JSON structure, which can be read by running a Python script on Ubuntu's 13.06 on HPS after 2 seconds. This data array is posted to a local MySQL database, where another Python script obtains the variables values to present a real time graph in the web application.

In the FPGA, the first Python script creates a file with the independent variable values, humidity, temperature, rain occurrence, and light. Meanwhile, other C script executes, compiles the Multilayer Perceptron (MLP) Neural Network (NN), and returns a predicted value of Soil humidity. This forecasted value is stored in another file, which can be displayed later on the web application as a NN Prediction graph.

The NN consists of ten layers with five input features and is based on a linear model to predict the next soil humidity value of the plant, as illustrated in Figure 3. The NN is used as a regressor and was trained with 70% of the dataset using the Bayesian Regularization training algorithm.

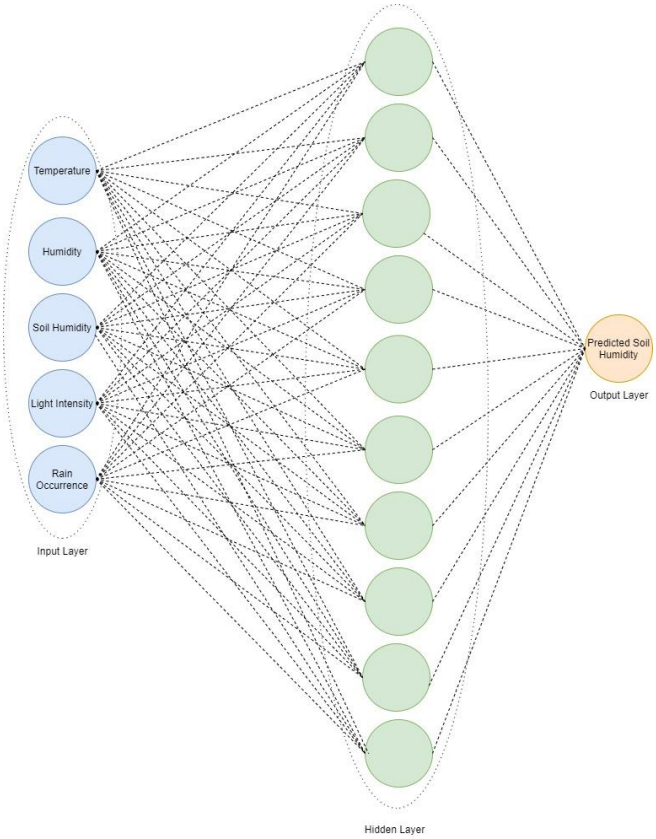

Fig. 3: Diagram of the neural net pattern recognition

The dataset was distributed as follows: 15% for validation, 15% for testing, and 70% for training. The performance of the regression model was measured by calculating the several performance metrics known for evaluation of regression models.

Through the Nginx web server, a python script "uWSGI" run and full-features are displayed in real-time, or

by means of buttons, the user can select them individually through the interface.

To validate the use of the neural network, we carried out several test by means of statistical tools such as multiple variable linear regression. The latter can predict the next value of the variable of interest (i.e. soil moisture) and as a regression model and it is given as:

$$y = B_0 + B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4 \quad (1)$$

where 'Xn' variables corresponds to the temporal series obtained for the Light intensity, relative humidity, rain occurrence and environment temperature. The coefficients B_0, B_1, B_2, B_3 and B_4 were estimated using the Matlab Regression Learner Toolbox with a linear model and are shown in Table II.

Table II. Coefficients of linear regression with their typical error

Multiple Linear Regression	Coefficients (B_n)	Typical Error
Interception (X_0)	2797.6580	222.5496
Relative Humidity (X_1)	-14.4528	0.9428
Ambient Temperature (X_2)	-44.5125	6.2064
Light Intensity (X_3)	0.0238	0.0023
Rain Occurrence (X_4)	20.5883	8.4108

V. Numerical Results

In this section, we present the normalized acquired data in a graph and the numerical results of the proposed prediction models for the soil moisture variable. The dataset is available at http://ieee-dataport.org/4469 [14].

A. Normalized Data

Figure 4 presents the normalized sampled data obtained from the sensors. One can observe five curves in this figure, blue line represents the relative humidity, red line de environment temperature, orange line is the soil moisture, purple line is the light intensity and the dotted line corresponds to the rain occurrence.

We water the plant the first day, which explains the low values from soil moisture between 0 and 500. It is important to remark that during the winter season in Equator, there is a lot of raining nights, which explains the leaps of soil moisture. Also, due to the cloudy days, the light intensity seems to be highly reduced during the experiment.

B. Multiple linear regression model

The multiple linear regression model allowed us to obtain the hyperplane equation (2), with four explanatory variables. To predict the value of soil moisture, this equation establishes a function to obtain the soil moisture value from the other four parameters: Light intensity, relative humidity, rain occurrence and environment temperature.

$$y = 2797.66 - 14.45 * X_1 - 44.51 * X_2 + 0.023 * X_3 + 20.59 * X_4 \quad (2)$$

Table III presents the statistical values of the regression: multiple correlation coefficients, coefficient of determination, R^2 Tight equation (3), typical error and number of observations. The variability ratio of our variable of interest has a very small value as shown by the coefficient of determination. In addition, the adjusted coefficient of determination (R^2 Tight), which is our reference value, is very close to 0. Therefore, our variable 'y' is very little explained by our regression model and would have little representation of the real variation.

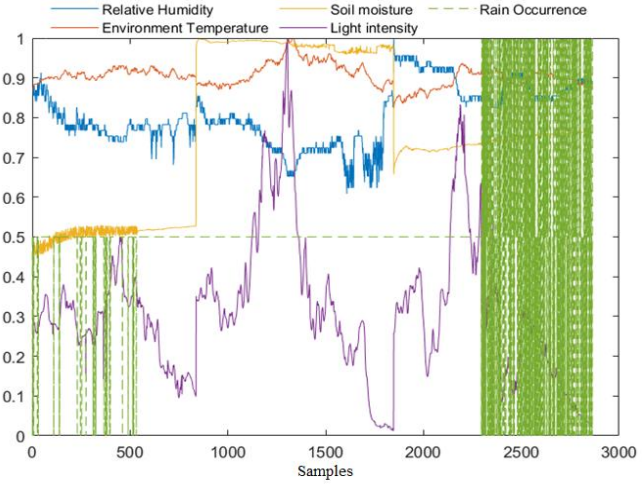

Fig. 4. The five variables measured during the weekend from 9am-6pm.

TABLE III. REGRESSION STATISTICS

Regression Statistics	
Multiple correlation coefficient	0.3481
Coefficient of determination R^2	0.1212
R^2 Tight	0.1199
Typical error	178.5566
Observations	2864

Table IV shows the percentage of correlation that exists between our variable of interest and the remaining variables. By doing this, we can be confirmed that the contribution to our variable of interest from that the other variables is low. Only 11.99% of the average of the variability of the variable 'y' can be explained by the adjusted regression model.

TABLE IV. CORRELATION VALUES

Correlation Matrix	Soil Moisture
Relative Humidity	-0.3068
Ambient Temperature	0.1850
Light Intensity	0.2130
Rain Occurrence	0.0489
Soil Moisture	1

For this reason, we develop and implement a neural network as explained in Section IV. Using this approach, we could visualize that the predicted values were very close to expected values. It is important to remark that the value passed per parameter should be in the range of previously trained data. The prediction of the real soil moisture with the neural network algorithm is shown in Figure 5 together with the real values.

To perform the neural network tests, 10-fold cross validation of 30% of the dataset was used. This was possible thanks to the multiple tests carried out with experimental data from our graph.

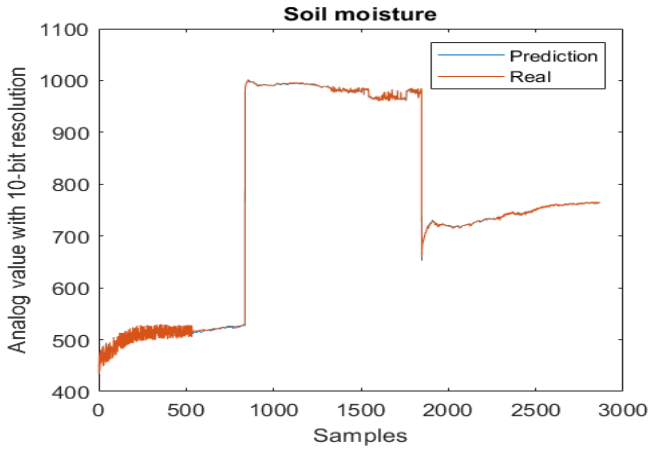

Fig. 5. Prediction of the next value of soil moisture

To evaluate the model, we use three commonly metrics for model evaluation in regression:

- R-Squared (R^2) is a statistical measure used to determine how similar in percentage the prediction is to the output. This parameter is given as follows:

$$R^2 = \frac{\sum_{t=1}^{n}(\hat{Y}_t - \bar{Y})^2}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2} \quad (3)$$

where *n* represents the number of data points, (Y_i, $\hat{Y}_i$) are the observed and predicted values respectively.

- Mean Squared Error (MSE) is one way to determine the prediction error, which is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \quad (4)$$

- Mean Absolute Error (MAE) is similar to MSE but it takes the sum of absolute value of error, and is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\left(Y_i - \hat{Y}_i\right)\right| \quad (5)$$

In our case, we use the Neural Network Toolbox from Matlab to generate Figure 6, which presents that the best training performance occurred at the iteration 383. Figure 7 presents the obtained values for the metrics R^2, MSE and MAE for the proposed model.

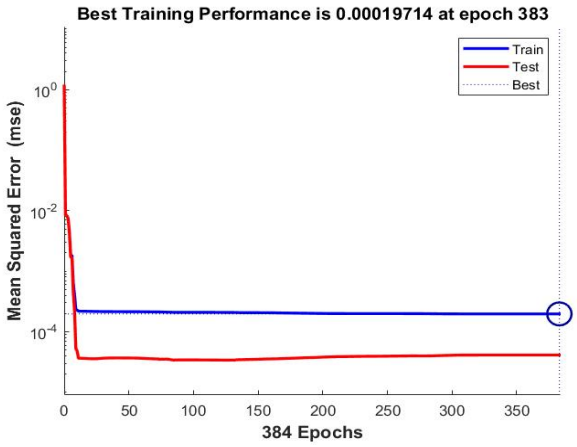

Fig. 6: Best Training Performance

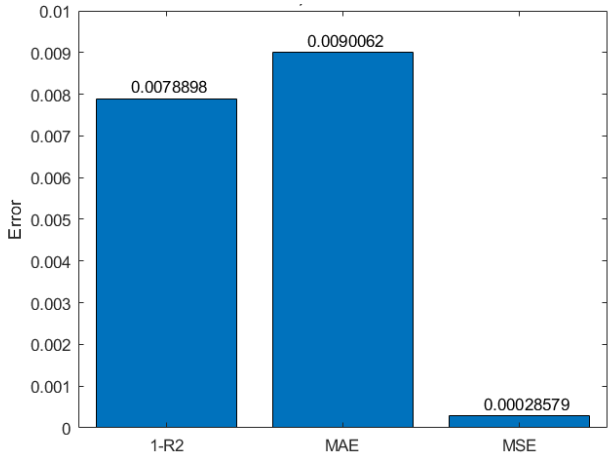

Fig. 7 Performance Metrics for the model evaluation in regression

C. *Computational Resources Usage*

We present the resource usage for each service in Table 5. In other words, Table V presents the percentage of CPU and RAM in the FPGA used by the different services in order to analyze and visualize the data.

As we expected, the requirement of computational resources is very small. Nevertheless, one can observed that Python service requires the highest percentage of processing while the database service requires the highest percentage of memory.

TABLE V. USED RESOURCES

Processes	% CPU	% MEM
My SQL	2.30	7.50
Network Manager	0.01	1.40
N GIN X	0.01	1.70
Python	4.70	0.80
PHP	0.01	2.30
Total	7.03	13.70

VI. CONCLUSIONS

By using statistical tools and neural network concepts, our model is able to predict the next value of our variable of interest with the performance metrics R^2 = 0.9921, MSE = 0.00028 and MAE = 0.0090. Using simple multiple linear regression model does not provide a viable solution due to the low percentage of the variability of the variable of interest. Regarding the neural network, it is important to select the appropriate variables that allow the adequate training taking into account the limited hardware resources in such way that the NN will have low consumption of memory and processing. Moreover, it is recommended to use language such as 'C' and 'Python' for analyzing and displaying the data respective, which use only 7.03% of CPU and 13.70% of RAM, resulting in little stress to hardware and low energy consumption. As future works, we are planning to test linear regression algorithms and some variant of recurring neural network or use classical statistical predictors to compare them with the results obtained with a neural network and the accessibility of the application and to develop a mobile application for remote access to perform the management and visualization of information from their mobile devices.

REFERENCES

[1] Zambrano, X. H. V., Cedeno, A. C., Alava, J. M., Cedeno, J. M., Sinichenko, E. K., & Gritsuk, I. I. (2020, November). Mathematical model to determine the runoff coefficient based on precipitation and curve number data, in the Manabi hydrographic demarcation, Ecuador. In Journal of Physics: Conference Series (Vol. 1687, No. 1, p. 012036). IOP Publishing.

[2] Kaewwongsri, K., & Silanon, K. (2020, June). Design and implement of a weather monitoring station using CoAP on NB-IoT network. In 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) (pp. 230-233). IEEE.

[3] Stefano Abbate, Marco Avvenuti, Luca Carturan, and Daniel Cesarini. Deploying a communicating automatic weather station on an alpine glacier. The 4th International Conference on Ambient Systems, Networks and Technologies, the 3rd International Conference on Sustainable Energy Information Technology.

[4] Beşikçi, E. B., Kececi, T., Arslan, O., & Turan, O. (2016). An application of fuzzy-AHP to ship operational energy efficiency measures. Ocean Engineering, 121, 392-402.

[5] Rocha, S. N. G., Cristobal-Salas, A., & Santes, N. I. L. (2015, October). REMMA citizen observatory: a monitoring and an alerting real-time system. In 2015 International Conference on Computing Systems and Telematics (ICCSAT) (pp. 1-5). IEEE.

[6] M. Haefke, S. Mukhopadhyay and H. Ewald, “A Zigbee Based Smart Sensing Platform for Monitoring Environmental Parameters”, IEEE Conference on Instrumentation and Measurement Technology, pp.1–8

[7] G. R. Mendez, M. A. Yunus and Dr. S. C. Mukhopadhyay, “A Wi-Fi based Smart Wireless Sensor Network for an Agricultural Environment”, Fifth International Conference on Sensing Technology, pp. 405–410.

[8] I. Aziz, M. Hasan, M. Ismail, M. Mehat and N. Haron, “Remote Monitoring in Agricultural Greenhouse using Wireless Sensor and Short Message Service”, International Journal of Engineering Technology, vol. 9, pp. 1–12

[9] M. Dinesh and P. Saravanam, “FPGA Based Real Time Monitoring System for Agricultural Field”, International Journal of Electronics and Computer Science Engineering, pp. 1514–1519

[10] SHAARI, S., et al. FPGA-based artificial neural network for prediction of solar radiation data from sunshine duration and air temperature. In 2008 IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering. IEEE, 2008. p. 118-123

[11] Jang, H. S., Bae, K. Y., Park, H. S., & Sung, D. K. (2016). Solar power prediction based on satellite images and support vector machine. IEEE Transactions on Sustainable Energy, 7(3), 1255-1263

[12] V. A. Armijos, N. S. Chan, R. Saquicela and L. M. Lopez, "Monitoring of system memory usage embedded in FPGA," 2020 International Conference on Applied Electronics (AE), 2020, pp. 1-4, doi: 10.23919/AE49394.2020.9232863

[13] Asanza V., Sanchez G., Cajo R., Peláez E. (2021) Behavioral Signal Processing with Machine Learning Based on FPGA. In: Botto-Tobar M., Zamora W., Larrea Plúa J., Bazurto Roldan J., Santamaría Philco A. (eds) Systems and Information Sciences. ICCIS 2020. Advances in Intelligent Systems and Computing, vol 1273. Springer, Cham. https://doi.org/10.1007/978-3-030-59194-6_17

[14] Juan Cadena, Steven Santillan, Víctor Asanza, Rebeca Estrada. (2021). Weather Monitoring Station For Farms And Agriculture. IEEE Dataport. https://dx.doi.org/10.21227/mdfs-ya42

[15] Teranishi, T. (1994). Tera Term Open Source Project. Retrieved 6 December 2019, from https://ttssh2.osdn.jp/index.html.en

[16] Neundorf, A. (2004). CuteCom. Retrieved 8 December 2019, from http://cutecom.sourceforge.net/.