Journal of Engineering Science and Technology
Vol. 13, No. 12 (2018) 4131 - 4144

IMPLEMENTATION OF AN INDUSTRIAL AUTOMATION SYSTEM MODEL USING AN ARDUINO

ARSHEEN MIR[1,*], R. SWARNALATHA[2]

[1]Electronics and Instrumentation Engineering, Birla Institute of Technology and Science, Pilani Dubai Campus, Academic City, Dubai, United Arab Emirates
[2]Electrical and Electronics Department, Birla Institute of Technology and Science, Pilani, Dubai Campus, Academic City, Dubai, United Arab Emirates
*Corresponding Author: mirarsheen@gmail.com

Abstract

This paper discusses a model of an industrial system on a small scale, which performs sequential operations using relays. The system shuts down automatically once it detects high water or temperature levels, which are hazardous for industrial operation. This model is implemented using an Arduino microcontroller as it proves to be the most viable alternative. Being an open source platform with the minimal cost it helps prototype the system with ease. Its compatibility with sensors helps provide the required feedback and the Arduino controls the system by taking corrective measures. The status of the system is depicted using devices like the Liquid Crystal Display (LCD), buzzer, etc. The sensor data is derived and stored in a Data Acquisition System (DAS), this data can then be used for control and monitoring purposes. Thus, the designed system helps replicate an industrial application and reduce the cost of setting up and maintaining an actual Distributed Control System (DCS), Programmable Logic Controller (PLC) or Supervisory Control and Data Acquisition (SCADA) in viable industrial environments. It also provides flexibility, with rapid prototyping and ease in error rectification. However, the additional hardware, certifications and technical support required to implement the design in an industrial environment when compared to a conventional industrial system can prove to be a drawback.

Keywords: Arduino, Data acquisition, Industrial automation, Temperature sensor, Water sensor.

1. Introduction

With the recent developments in technology, all processes are being automated. Apart from industrial automation, automation is prevalent in the domestic domain making homes more smart and secure. It has also helped reduce human effort enabling the control of devices/appliances with great ease while being energy efficient [1]. Various home automation models have been implemented incorporating Android platforms [1], Global System for Mobile communications (GSM) modules [2], Wi-Fi-based systems [3], etc., into the fundamental microprocessor, sensor and actuator network [4, 5]. The wide application of microprocessors is not confined to domestic/home automation applications but can also be further extended into the industrial environment.

In the industrial domain, a fundamental automation system model is a basic control system, which includes an input/sensor, a controller and an output/actuator. This model can help implement any industrial application with appropriate hardware and software selection. The application discussed in this paper is level and temperature control during a continuous sequential switching operation.

A repetitive sequential switching operation replicates a traditional production line. Sequential operations are widely used in industries for packaging, production and similar activities. A simple example of such an operation would be the production of a batch of screws in a factory. The sequence of the process flow for the mentioned activity would be cutting, heading, lathing, threading, heat treatment, electroplating, and packaging. This repetitive sequential nature of activities is widely observed in all industries and can be easily replicated on a small scale for small factories.

Level determination and monitoring systems are used in wastewater treatment plants, oil and gas industries, chemical and food processing industries, etc., for several applications including liquid storage, monitoring and control. These systems use sensors based on different working principles namely magnetic, ultrasonic, and Radiofrequency technology. The controllers used are mostly PLC's or conventional Proportional-Integral (PI)/Proportional-Integral-Derivative (PID) controllers, which communicate between the sensor and the output device (usually a pump) to assist the inflow or outflow of fluid as required.

Temperature control and monitoring systems find its application in a wide range of process control industries. The input is derived from contact temperature sensors like thermistors, thermocouples or resistance temperature detectors or non-contact sensors based on infrared or similar technologies. These devices provide the required input to the system and trigger output as determined by the controller. Depending on the entire system either conventional controllers like the PI/PID controller or an ON/OFF controller is used. The latter is used if the temperature is not a very critical parameter in the system.

As evident, industrial processes are mostly based on PLC, DCS or SCADA systems. These systems prove to be complex for small-scale industries and also expensive because of their high initial cost of setup. As commented by Darandale and Gunjal [6] and Sobota et al. [7], Arduino being an open source hardware platform enables us to replicate any industrial process and experiment with it on a small scale at a very minimal cost, with a workflow like industrial controllers. Using PLC's and conventional systems requires a lot of wiring and technical

expertise, thus, an Arduino helps reduce assembly, manufacturing and maintenance cost in such scenarios. With all these facts stated, an Arduino cannot completely replace the existing systems but can be a viable option for small-scale industries and initial startups as it helps rapidly prototype, code and interface different types of sensors, devices and modules and is also widely available.

Microprocessors like the ATmega 328P (present on the Arduino), can make a number of logical control decisions by mere programming in C/C++ languages. Arduino UNO is compatible with a number of software's like Matrix Laboratory (MATLAB), Parallax Data Acquisition (PLX-DAQ) tool apart from its main software, which is the Arduino IDE. The applications of this board can be extended even further by using compatible shields and external modules like GSM, Bluetooth, etc. [2]. With its affordable price, it thus, becomes a feasible option for a wide range of applications like Access Control, Data Logging and Automation [6].

2. Method and Implementation

2.1. Overview and basic block diagram

The designed system model as discussed in the introduction can be seen in Fig. 1. Arduino UNO acts as the controller; it receives inputs from the water level sensor and temperature sensor continuously. These monitored input variables determine if the system should run smoothly or be stopped. The output of the system is depicted by relays, their continuous sequential working indicating a smooth operation and the additional relay indicating an alert.

A relay is used to power many high voltage devices, like lamp loads, motors and pumps. In this design, regular Light Emitting Diode (LED)'s is used to demonstrate the logical functioning of the relay, which is a real-time system can be replaced by other actuators as required. The relay activates the loads (LED's) in the following sequence: relay 1- relay 2- relay 3 - relay 4 - relay 1 on a loop. Thus, the relay is an actuator in the model, which helps produce the required output.

As discussed, the repetitive sequence of operation continues until the system encounters any disruption in the form of increased water level in the tank or temperature within the plant, by the sensors. These sensors are thus, the inputs required by the system to take corresponding actions.

Once the system stops due the above-mentioned disturbances, the switching on of the relay 5 signifies the operation of a pump to take charge of the exceeding water level. An external power supply and a regulator is required to meet the high-power needs of a large number of components in the circuit as shown in Fig. 1. The status of the system is displayed continuously with the help of the LCD module. Apart from the process above, there is continuous logging of data from the sensors to a Data Acquisition System, which is the PLX-DAQ.

2.2. Hardware Implementation

Various blocks of the system as seen in Fig. 1 were implemented in hardware in Fig. 2, following the circuitry in Fig. 3. The various components that make this circuit include (1) Water and temperature sensor, (2) Arduino UNO board, (3) Relay Module, (4) External power supply & Voltage Regulation block, (5) LCD,

Buzzers and LED's (Loads), (6) Real Time Clock (RTC) Module and (7) PLX-DAQ. The details of each are listed below.

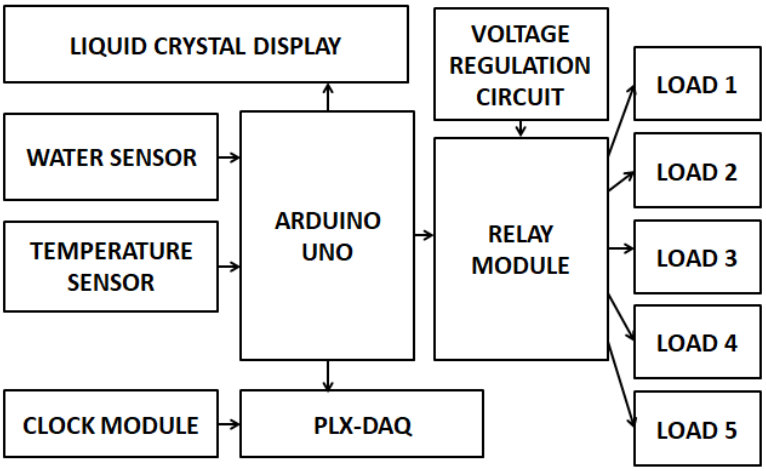


Fig. 1. Basic block diagram of the system.

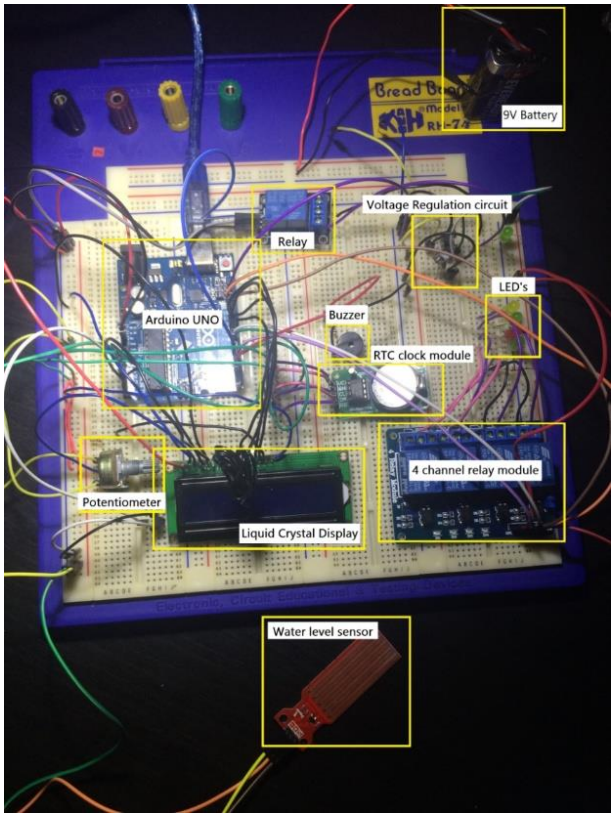


Fig. 2. Hardware Implementation.

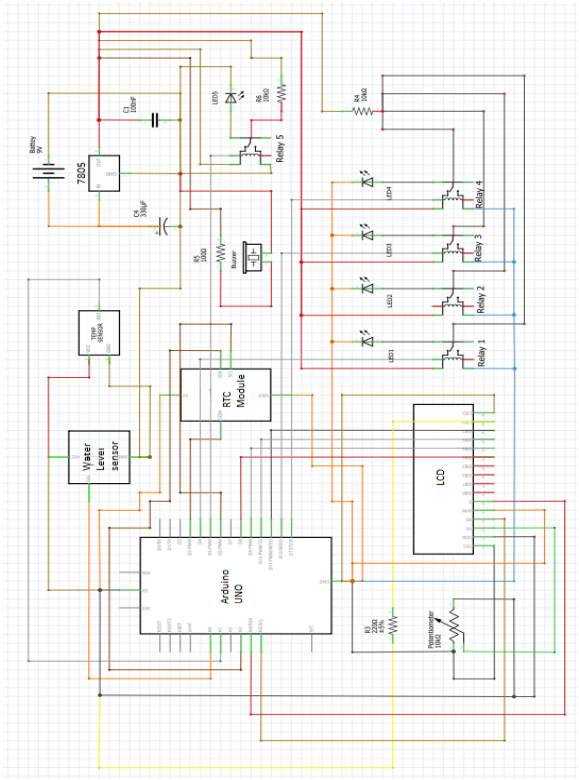


Fig. 3. Proposed circuit.

2.2.1. Water and temperature sensor

A Keyes Water Level Sensor as seen in Fig. 4 is used for Level Monitoring. It comprises 3 pins, for analogue output, supply and ground. It measures the size of the water droplets through the lines with the help of the parallel wires exposed and determines the water level. For monitoring temperature, an LM35 temperature sensor is used. As seen in Fig. 4 it consists of 3 pins, 2 for input voltage and ground and the central for analogue output [4, 8]. It is used for temperature monitoring [4] and measurement applications like Vehicle Traffic Congestion Control [9], Automated Irrigation system [10], Automatic chicken feeder [11], greenhouse control system [12] and fire alarm system [13]. The output voltage produced by this sensor varies linearly with temperature in the range of -55 to 150 degree Celsius.

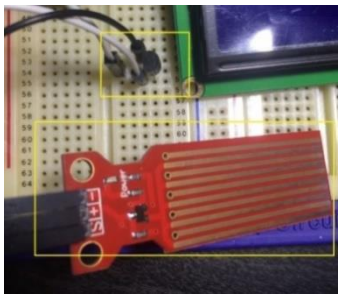


Fig. 4. Keyes water sensor and LM35 temperature sensor.

2.2.2. Arduino UNO

Arduino UNO board comprises of the ATmega 328P microcontroller and supports digital/analogue inputs and outputs as shown in Fig. 5. According to Priya et al. [14], it consists of 14 digital I/O pins, out of which, 6 can be used as Pulse Width Modulation (PWM), 6 analogue pins, a reset button, in-circuit serial programming (ICSP) header and a power jack. When connected through the USB port it communicates through Serial and loads the program into the microcontroller. As explained by Darandale and Gunjal [6], in addition to this, it has an inbuilt voltage regulation circuitry, which makes it ideal for use in such applications.

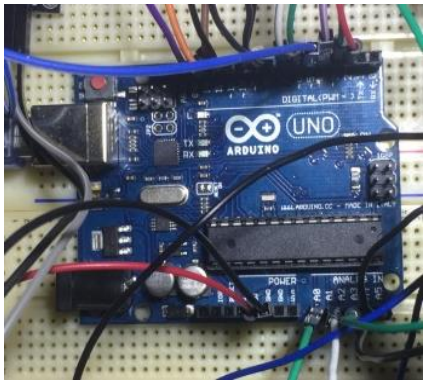


Fig. 5. Arduino UNO.

2.2.3. Relay module

Based on a study by Mashilkar et al. [15], the relay is an electrically operated switch. It uses an electromagnet to mechanically operate a switch using a low power signal. Relay board with 4 relays has been used for carrying out the repetitive sequential action and an additional relay to drive the pump in case the water level exceeds as in Fig. 6.

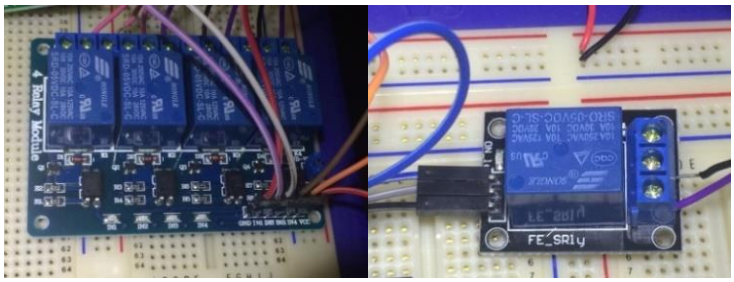


Fig. 6. Relay module and an additional relay.

2.2.4. External power supply and voltage regulation block

Since a large number of components are connected to the Arduino board and they derive a lot of power, an additional battery is needed to drive the components. This voltage must be regulated to 5V for smooth operation and thus, a voltage regulator IC LM7805 is used in addition to a few capacitors to get the required voltage. The circuit diagram can be seen in Fig. 7 [16].

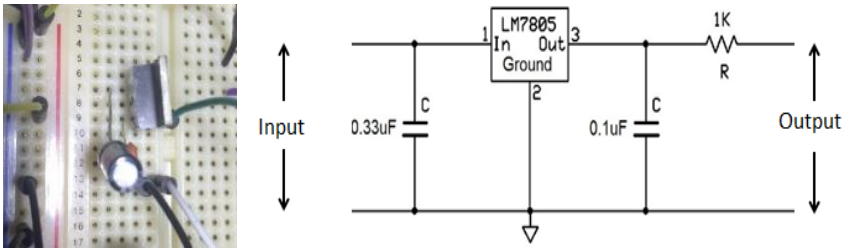


Fig. 7. Voltage regulation circuit (hardware and circuit).

2.2.5. LCD, buzzers and LED's

A 16X2 Liquid Crystal Display always provides the status of the operation [15]. The LCD works on 5V and 1mA current and requires a total of 11 connections comprising of 8 data and 3 control pins [10]. In addition to the LCD, a buzzer rings in case the water level or temperature exceeds the operating range. LED's are used to indicate the relay switching on/off as shown in Figs. 8 and 9.

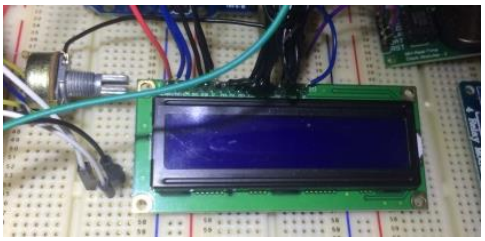
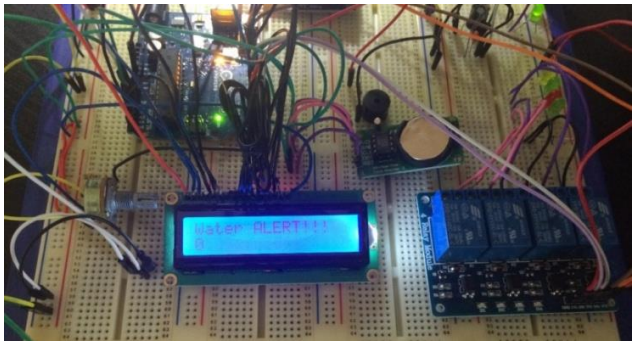


Fig. 8. Liquid crystal display.

(a) Process start status.

(b) Process in operation.



(c) Process disturbance alert.

Fig. 9. LCD statuses.

2.2.6. RTC clock module

The DS1302 in the RTC clock module functions as a real-time clock and calendar as shown in Fig. 10. It has 31 bytes of Static Random-Access Memory (SRAM) and communicates with the processor using serial communication. According to Zeebaree and Yasin [2], it makes automatic adjustments based on the number of months and also handles leap years. In this design, it helps to keep an accurate record of data for the PLX-DAQ.

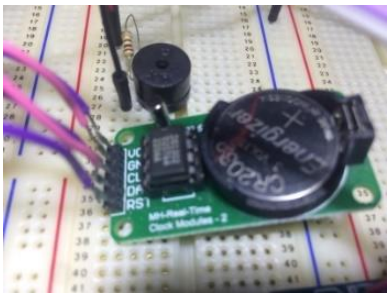


Fig. 10. RTC clock module.

2.2.7. PLX-DAQ

The PLX-DAQ or the Parallax Data Acquisition System is an add-on to excel, which helps us acquire data from any microcontroller from up to 26 channels. It puts the incoming data into separate columns. This available data can be used for analysis, monitoring and control [17] as in Fig 11.

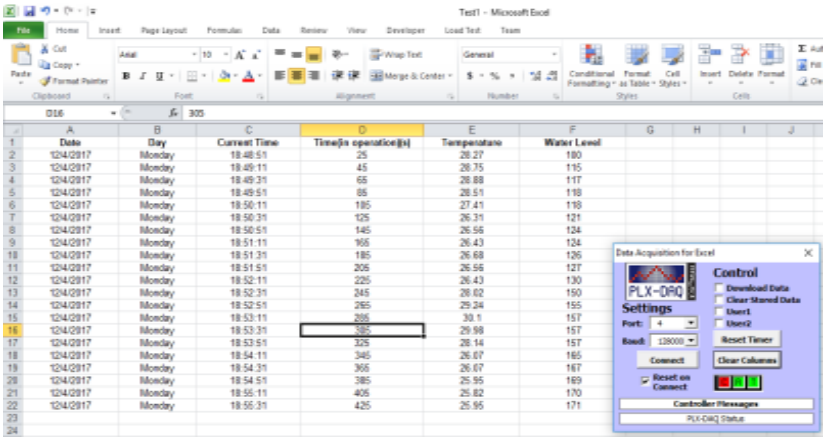


Fig. 11. PLX-DAQ.

2.3. Software implementation

Arduino IDE 1.0 and PLX-DAQ were the two major software's used for the implementation of this design. Apart from this, 2 libraries for the LCD and clock module were incorporated into Arduino IDE. These are the 'LiquidCrystal.h' and 'DS1302.h' for the LCD and the RTC module respectively. The logic can be seen in Fig. 12 in the form of a flowchart.

As depicted in this chart, the relay sequence continues unless and until it experiences a disturbance in the form of a water level or temperature hike. The Arduino code is composed of 2 modules, namely setup and loop [6]. The setup function initializes input/output pins, variables and serial ports for serial communication. The loop function performs the repetitive action of monitoring the water level and temperature, controlling the relay sequence accordingly and recording the readings in the PLX-DAQ system. An additional set of code is required to initialize the date and time on the RTC module, which can be then be commented after its first use. The Arduino IDE interface can be seen in Fig. 13.

Program code

The code is composed of 2 modules, namely setup and loop [6]. The setup function initializes input/output pins, variables and serial ports for serial communication. The loop function performs the repetitive action of monitoring the water level and temperature, controlling the relay sequence accordingly and recording the readings in the PLX-DAQ system. An additional set of code is required to initialize the date and time on the RTC module, which can be then be commented after its first use. The code is outlined in *Appendix A.*

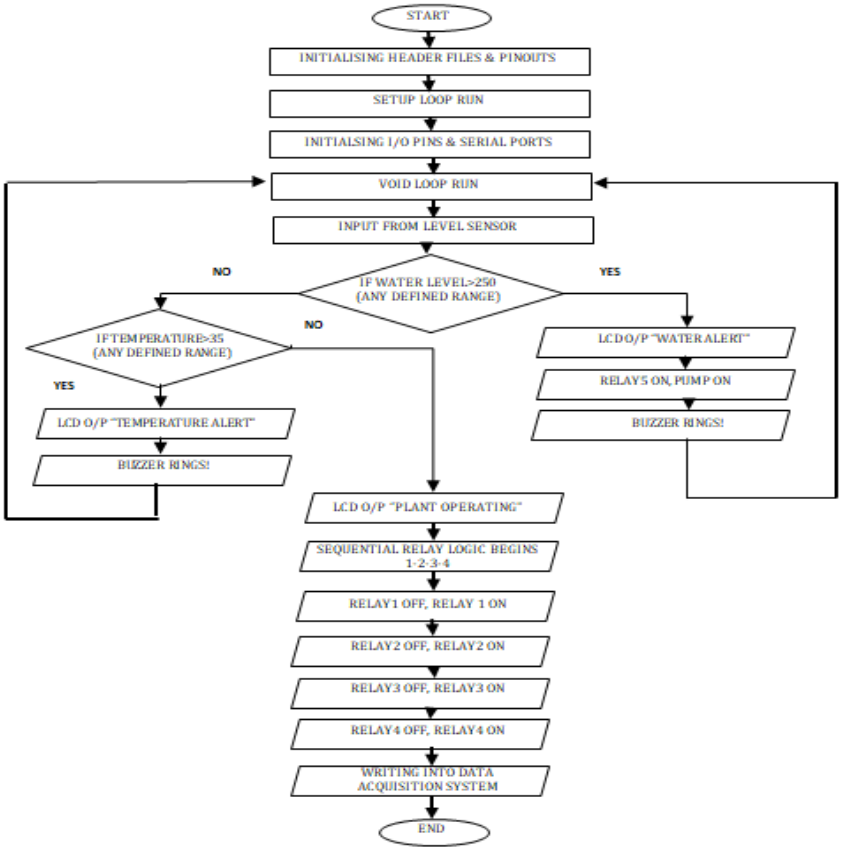


Fig. 12. Program logic.

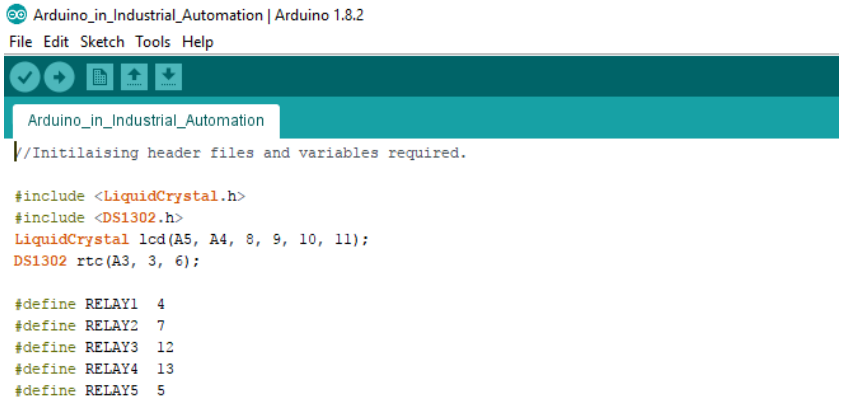


Fig. 13. Arduino IDE interface.

3. Results and Discussions

As witnessed, the designed hardware helps replicate a sequential logical operation on a small scale using a micro-controller based system. This design can be implemented in an industrial environment, with appropriate measures taken. Continuous monitoring of data is done through the LCD display as seen in section 2.2.7. With PLX-DAQ this data is compiled for analysis and record purposes. Experimental data from PLX-DAQ is shown in Table 1 and Fig. 14.

Based on an analysis by Reneker [18] with the current designed system on implementation of flow control using an Arduino and PLC independently, the results can be summarised as follows:

- Conventional controllers prove to be expensive for small industries due to the high initial cost of setup compared to a basic microcontroller system. An Arduino thus, with its minimal cost proves to be a feasible option
- Arduino being an open source platform, can perform any control function and replicate any industrial process by mere C/C++ programming language eliminating the need of wiring and technical expertise.
- In addition to the initial setup cost, the added cost of assembly, manufacturing and maintenance activities is also minimised.
- The use of Arduino can be extended further, by incorporating compatible shields, external modules, sensors and actuators like Humidity sensor, Bluetooth module, motors, etc.
- The Arduino platform is also compatible with many software's like MATLAB, PLX-DAQ, etc., through the Arduino IDE further increasing its applications.
- An Arduino provides flexibility, helps rapidly prototype and rectify errors with ease.

Despite the above-mentioned advantages, the existing industrial systems like DCS, SCADA, etc., are still irreplaceable in a number of domains. Where an Arduino can perform similar functions, its use is still limited to certain applications and industrial environments due to industry standard and certification requirements. In rugged environment conditions like high temperatures, the use of an external protection shield becomes essential. Since

PLC's and conventional industrial systems are built for the purpose, they have built-in protocols and provide extensive resources to aid set-up and operation. An Arduino in comparison would require additional hardware, programming skills and technical support.

Table 1. Data from PLX-DAQ.

Date	Day	Current time	Time (in operation)	Temperature	Water level
12/4/2018	Monday	18:48:51	25	28.75	100
12/4/2018	Monday	18:49:11	45	28.27	115
12/4/2018	Monday	18:49:31	65	28.88	117
12/4/2018	Monday	18:49:51	85	28.51	118
12/4/2018	Monday	18:50:11	105	27.41	118
12/4/2018	Monday	18:50:31	135	26.31	121
12/4/2018	Monday	18:50:51	155	26.56	124
12/4/2018	Monday	18:51:11	175	26.43	124

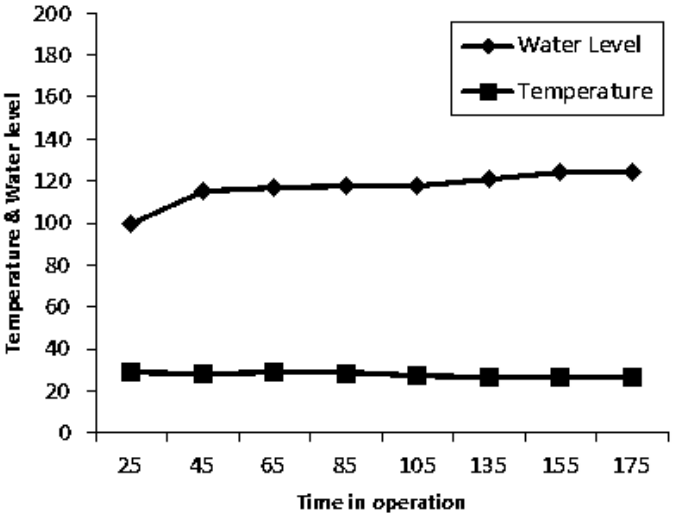


Fig. 14. Temperature and water level profile.

4. Conclusion

In this paper, a simple industrial system was implemented on a small scale. This design can be modified as per needs and utilized in many small applications. It depicts how simple control functions can be carried out using just a microcontroller and a few sensors and actuators. This design could perform repetitive sequential actions with great ease and could also take corrective actions in case of water level or temperature disturbances. The LCD, Buzzer depicts the status of the process and the PLX-DAQ helps with data logging.

Acknowledgement

The authors would sincerely like to thank Prof. R. N. Saha, Director, BITS Pilani, Dubai Campus for his constant encouragement and support.

Abbreviations

DAS	Data Acquisition System
DCS	Distributed Control System
GSM	Global System for Mobile Communications
ICSP	In Circuit Serial Programming
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MATLAB	Matrix Laboratory
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PLC	Programmable Logic Controller
PLX-DAQ	Parallax Data Acquisition Tool
PWM	Pulse Width Modulation
RTC	Real Time Clock
SCADA	Supervisory Control and Data Acquisition
SRAM	Static Random-Access Memory

References

1. Javale, D.; Mohsin, M.; Nandanwar, S.; and Shingate, M. (2013). Home automation and security system using android ADK. *International Journal of Electronics Communication and Computer Technology* (*IJECCT*), 3(2), 382-385.
2. Zeebaree, S.R.M.; and Yasin, H.M. (2014). Arduino based remote controlling for home: power saving, security and protection. *International Journal of Scientific and Engineering Research*, 5(8), 266-272.
3. ElShafee, A.; and Hamed, K.A. (2012). Design and implementation of a WiFi based home automation system. *International Journal of Computer and Information Engineering*, 6(8), 1074-1080.
4. Bharathkumar. V; Irshad, S.M.; Gowtham, S.; and Geethamani, R. (2017). Microcontroller based digital meter with alert system using GSM. *Proceedings of the 11th International Conference on Intelligent Systems and Control* (*ISCO*). Coimbatore, India, 444-448.
5. David, N.; Chima, A.; Ugochukwu, A.; and Obinna, E. (2015). Design of a home automation system using Arduino. *International Journal of Scientific and Engineering Research*, 6(6), 795-801.
6. Darandale, D.C.; and Gunjal, B.L. (2013). Development of web-based SCADA like application using Arduino platform. *International Journal of Emerging Technology and Advanced Engineering*, 3(8), 172-176.
7. Sobota, J.; Pisl, R.; Balda, P.; and Schlegel, M. (2013). Raspberry Pi and Arduino boards in control education. *IFAC Proceedings Volume*, 46(17), 7-12.
8. Liu, C.; Ren, W.; Zhang, B.; and Lv, C. (2011). The application of soil temperature measurement by LM35 temperature sensors. *Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology*. Harbin, China, 1825-1828.
9. Abhimane, A.; Limkar, P.; Barate, B.; and Puranik, V.G. (2017). Iot based vehicle traffic congestion control and monitoring system. *Proceedings of the*

2nd *International Conference for Convergence in Technology (I2CT)*. Mumbai, India, 1220-1223.

10. Srilikhitha, I.; Saikumar, M.M.; Rajan, N.; Neha, M.L.; and Ganesan, M. (2017). Automatic irrigation system using soil moisture sensor and temperature sensor with microcontroller AT89S52. *Proceedings of the International Conference on Signal Processing and Communication (ICSPC)*. Coimbatore, India, 186-190.
11. Soh, Z.H.C.; Ismail, M.H.; Otthaman, F.H.; Safie, M.K.; Zukri, M.A.A.; and Abdullah, S.A.C. (2017). Development of automatic chicken feeder using Arduino Uno. *Proceedings of the International Conference on Electrical, Electronics and System Engineering (ICEESE)*. Kanazawa, Japan, 120-124.
12. Enokela, J.A.; and Othoigbe, T.O. (2015). An automated greenhouse control system using Arduino prototyping platform. *Australian Journal of Engineering Research*, 2(2), 1-13.
13. Bahrudin, M.S.; Kassim, R.A., and Buniyamin, N. (2013). Development of fire alarm system using Raspberry Pi and Arduino Uno. *Proceedings of the International Conference on Electrical, Electronics and System Engineering (ICEESE)*. Kuala Lumpur, Malaysia, 43-48.
14. Priya, V.K.; Vimaladevi, V.; Pandimeenal, B.; and Dhivya, T. (2017). Arduino based smart electronic voting machine. *Proceedings of the International Conference on Trends in Electronics and Informatics (ICEI)*. Tirunelveli, India, 641-644.
15. Mashilkar, B.; Khaire, P.; and Dalvi, G. (2015). Automated bottle filling system. *International Research Journal of Engineering and Technology* (*IRJET*), 2(7), 771-776.
16. Mir, A.; and Gaidhane, V.H. (2017).Deriving energy from far field RF signal. *Proceedings of the International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. Ras Al Khaimah, United Arab Emirates, 1-4.
17. Cavalcante, M.M.; de Souza Silva, J.L.; Viana, E.C., Dantas, J.R. (2014). The Arduino platform for didactic purposes. A case study with data collection from PLX-DAQ. *Proceedings of the XXXIV Congress of the Brazillian Computer Society*. Porto Alegre, Brazil, 1687-1696.
18. Reneker, D. (2017). PLC vs. Arduino for industrial control. Retrieved June 28, 2018, from https://www.controldesign.com/articles/2017/arduino-vs-plc-for-industrial-control/.

Appendix A

Program Code

```
//Initilaising header files and variables required.
#include <LiquidCrystal.h>
#include <DS1302.h>
LiquidCrystal lcd(A5, A4, 8, 9, 10, 11);                DS1302 rtc(A3, 3, 6);
#define RELAY1  4
#define RELAY2  7
#define RELAY3  12
#define RELAY4  13
#define RELAY5  5
int level = A0;                         int buzzer = 2;             int tempPin = A1;           int t=0;
int val;                                float temp;  float voltage;
void setup()
{ //Initilaising input and output pins.
  pinMode(level, INPUT);
  pinMode(RELAY1, OUTPUT);    pinMode(RELAY2, OUTPUT);
  pinMode(RELAY3, OUTPUT);    pinMode(RELAY4, OUTPUT);
  pinMode(RELAY5, OUTPUT);
  digitalWrite(RELAY5,LOW);   digitalWrite(RELAY1,LOW);
  digitalWrite(RELAY2,LOW);   digitalWrite(RELAY3,LOW);
  digitalWrite(RELAY4,LOW);
//Initialising the LCD
  lcd.clear();   lcd.begin(16, 2);          lcd.setCursor(0, 0);
  lcd.print(" Setup running! ");            delay(5000);
//Setting up the DAS - PLX-DAQ
Serial.begin(128000);                   Serial.println("CLEARDATA");      Serial.println("CLEARLABEL");

  Serial.println("LABEL,Date,Day,Current Time,Time(in operation)(s),Temperature,Water Level");
//Setting up the RTC-clock module
 rtc.halt(false);                 rtc.writeProtect(false);}
void loop()
{  int w=0;       int flag1=1;
    w = analogRead(level); //Taking input from the water sensor
    //Taking input from the temperature sensor
  val = analogRead(tempPin);
  voltage = 5 * val / 4095.0;       temp = (0.5-voltage) * 100;
   if(w>250)                //Checking the water level
   {    flag1=0;
   lcd.clear();                 lcd.setCursor(0, 0);        lcd.print("Water ALERT!!!");
   lcd.setCursor(0, 1);         lcd.print(flag1);
   digitalWrite(RELAY5,HIGH);
   tone(buzzer, 1000);       //Buzzer Alert
   delay(1000);                 noTone(buzzer);             delay(2000);    }
  else if(temp>35)          //Checking the temperature
   {    flag1=0;                lcd.clear();                lcd.setCursor(0, 0);        lcd.print("Temperature ALERT");
   lcd.setCursor(0, 1);         lcd.print(flag1);
   tone(buzzer, 1000);       //Buzzer Alert
   delay(1000);                 noTone(buzzer);
   delay(2000);    }
  else
   {    //Operation in normal operating conditions
   lcd.clear();                 lcd.setCursor(0, 0);        lcd.print("Plant Operating!");
   lcd.setCursor(0, 1);         lcd.print(flag1);
   //RELAY sequence
   digitalWrite(RELAY1,HIGH);           delay(5000);
   digitalWrite(RELAY1,LOW);
   digitalWrite(RELAY2,HIGH);           delay(5000);
   digitalWrite(RELAY2,LOW);
  digitalWrite(RELAY3,HIGH);            delay(5000);
   digitalWrite(RELAY3,LOW);
    digitalWrite(RELAY4,HIGH);          delay(5000);
   digitalWrite(RELAY4,LOW);    }
     long int milli_time = millis();  //Calculating the time since the start of operation
  t=milli_time/1000;
  //The following code is used to initialise the RTC module
  //Commented once used
  /*Serial.println(rtc.getTimeStr());    Serial.println(rtc.getDOWStr());
    Serial.println(rtc.getDateStr());
    Serial.println("LABEL,Date,Day,Current Time,Time(in operation),Temperature,Water Level"); */
  //Printing into the Data Aquistion System
  Serial.print("DATA,DATE,");             Serial.print(rtc.getDOWStr());
  Serial.print(",");                      Serial.print(rtc.getTimeStr());
  Serial.print(",");                      Serial.print(t);
  Serial.print(",");                      Serial.print(temp);
  Serial.print(",");                      Serial.print(w);
  Serial.println(); }
```