2018 International Conference on Multidisciplinary Research
Track 1: STEM (Science, Technology, Engineering & Mathematics)
https://doi.org/10.26803/MyRes.2018.10

Design and Implementation of an Automatic Electrical Motor Protection

Busiso Mtunzi,Tinashe Shelton Gavhu, Zedekia Madumbu Nyathi, Reginald Gonye and Fidelis Nhenga Mugarsanwa
National University of Science and Technology,
Bulawayo, Zimbabwe

Abstract
The purpose of this paper was to design and implement an automatic motor protection system prototype. The system designed made use of sensors to measure and provide protection to the motors against temperature, overcurrent, overvoltage and under voltage using an arduino as the controller. Units to monitor temperature, overcurrent, overvoltage and under-voltages were designed and integrated. The temperature and current tolerance of the motor were set at 50^{o}C and 0.26A respectively. The system could respond accordingly to temperature and current changes by way of stopping the motor when set conditions were exceeded. The data measured was stored on a memory card and could be retrieved at any time to analyse the induction motor behaviour. The system if incorporated on motors, could minimize potential damages to electric motors' windings, hence reducing electric motor maintenance costs for companies.

Keywords: Arduino, GSM Module; Induction Motor; Over Voltage; Under Voltage.

Introduction

This paper looks at the design and implementation of an automatic electrical motor protection and data logging system using a microcontroller as the heart of the system. According to Craig and Multilin (2016), over 1 billion motors are in service in North America alone and these, use over 70% of the energy used in the manufacturing sector. These motors' failure rate have been found to be as high as 12% and, 36% of this failure rate is related to the environmental conditions in which the motor is operating and 33% is due to electrical related issues. This would apply to all motors in use in general. It has been found that a 10^{o}C increase in operating temperature shortens motor life by about half and hence regularly checking of the operating temperature of critical motors would prevent unexpected shutdowns and extend motor life (Bishop, 2013). The most efficient operation and long motor life is only achievable if motors are operated close to the rated voltages and far from the outer limits (Cowern, 2000).

Monitoring the operating conditions like temperature under which the motors operate and also the current and voltage levels of the motors to make sure they do not operate outside the prescribed ranges will minimise motor failure rate in industry and in turn reduce maintenance costs for companies.

Electromechanical protection method is the mainstream form of technology used to protect electric motors in industry. These electromechanical type of motor protection methods prevent internal thermal damage to the electric motor and their main drawback is that they do not provide any advanced warning that an event is about to occur, and they do not provide any information about why the event occurred (Rockwell Automation [RA], 2016).

There is a need for consolidating many motor protection methods into a single electronic device so as to cut on installation costs, component costs, panel size, and maintenance time. Using a microcontroller based motor protection device can also provide communications capabilities to alert for possible motor problems and record on why the motor stopped

Research Methodology

The motor that was used in designing the motor protection system was a single phase induction motor. Induction motors are currently the most widely used motors in industry, hence their choice in this research activity. The block diagram illustrating the motor protection system was as shown in figure 1 below:

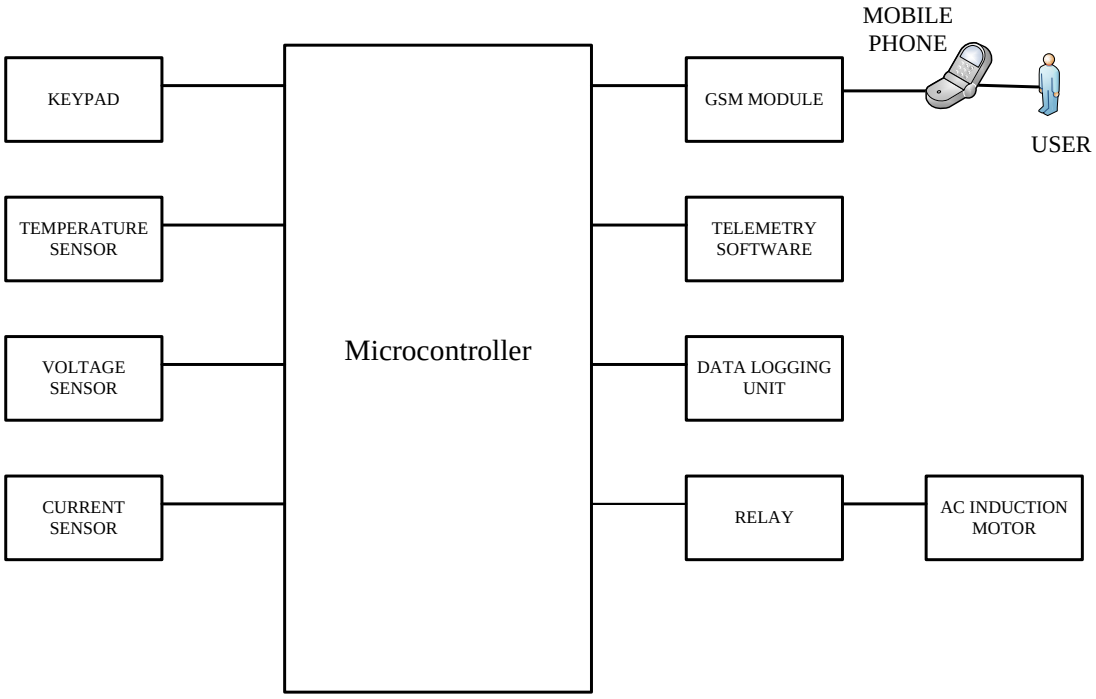

Figure 1 - System Block Diagram

As can be noted in figure 1, the system consisted of several functional units put together. These included the microcontroller, which provided the brains of the system, sensor units (temperature, voltage and current sensors), and output units which included a data logging unit, the AC induction motor (which responded accordingly if the sensed quantity was exceeded).

Materials and Procedure

The microcontroller that was used for the system design was the Arduino Mega 2560 shown below in figure 2.

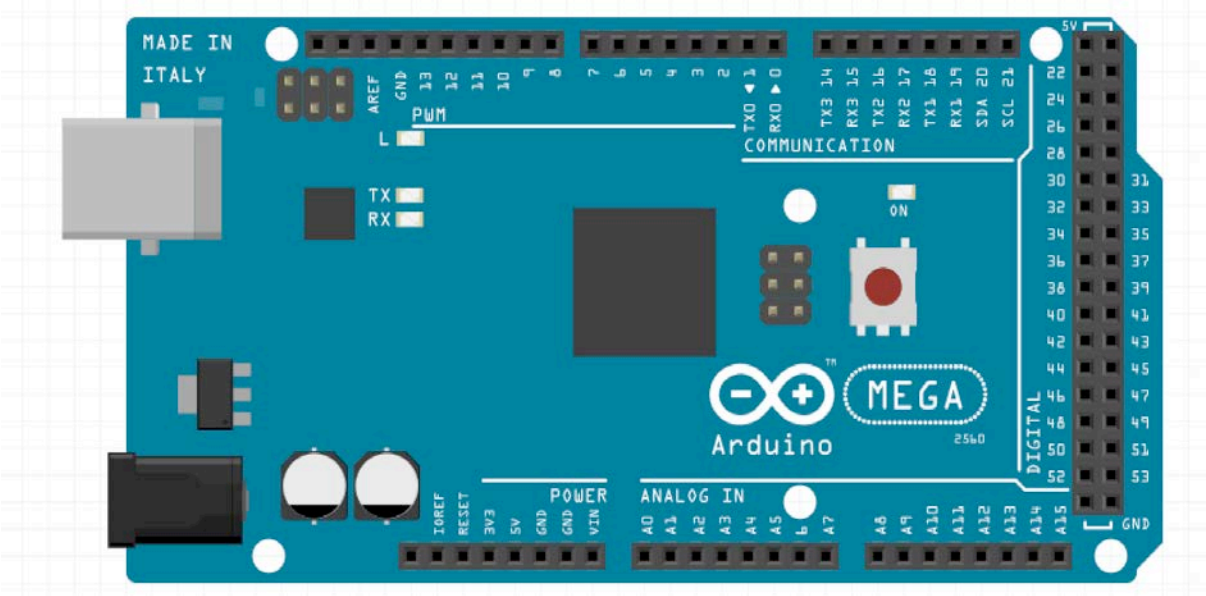

Figure 2 - Arduino Mega 2560

The Mega 2560 is a microcontroller board based on the ATmega2560 which has 54 digital input/output pins with 15 pins which can be used as pulse width modulation (PWM) outputs, 16 analogue inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button (ARDUINO AG, 2012).

The output unit of this system consisted of a relay which was used to stop and start the motor, a data logging terminal for storing all the measured operating parameters and a GSM module to notify engineers and technicians on the motor sensed quantities.

Motors need to be protected from high ambient temperatures and from high temperature rise due to the motor itself. High temperatures bring about damage to the wiring and insulation. The temperature protection mechanism of the system was set up to measure both the ambient temperature and motor temperature. Manufacturers usually specify the range of temperatures not to be exceeded on motors. They follow the National Electrical Manufacturers Association (NEMA) rates which gives the insulation and temperatures accommodated by electric motors. Class A insulation has a recommended temperature limit of 105°C, Class B 130°C, Class F 155°C, and Class H 180°C. The motor that was used in this research activity was a class A motor and had a power rating of 6 Watt. An infrared temperature

sensor MLX90614 designed for non-contact temperature sensing was used. It has an internal 17-bit analogue to digital converter and a powerful digital signal processor that was used for temperature sensing. The sensor could measure both the ambient and object temperature from a distance.

According to the sensor datasheet, the sensor could measure -40 to 85°C for the ambient temperature and -70 to 382.2°C for the object temperature. Figure 3 shows the temperature sensor and its connections;

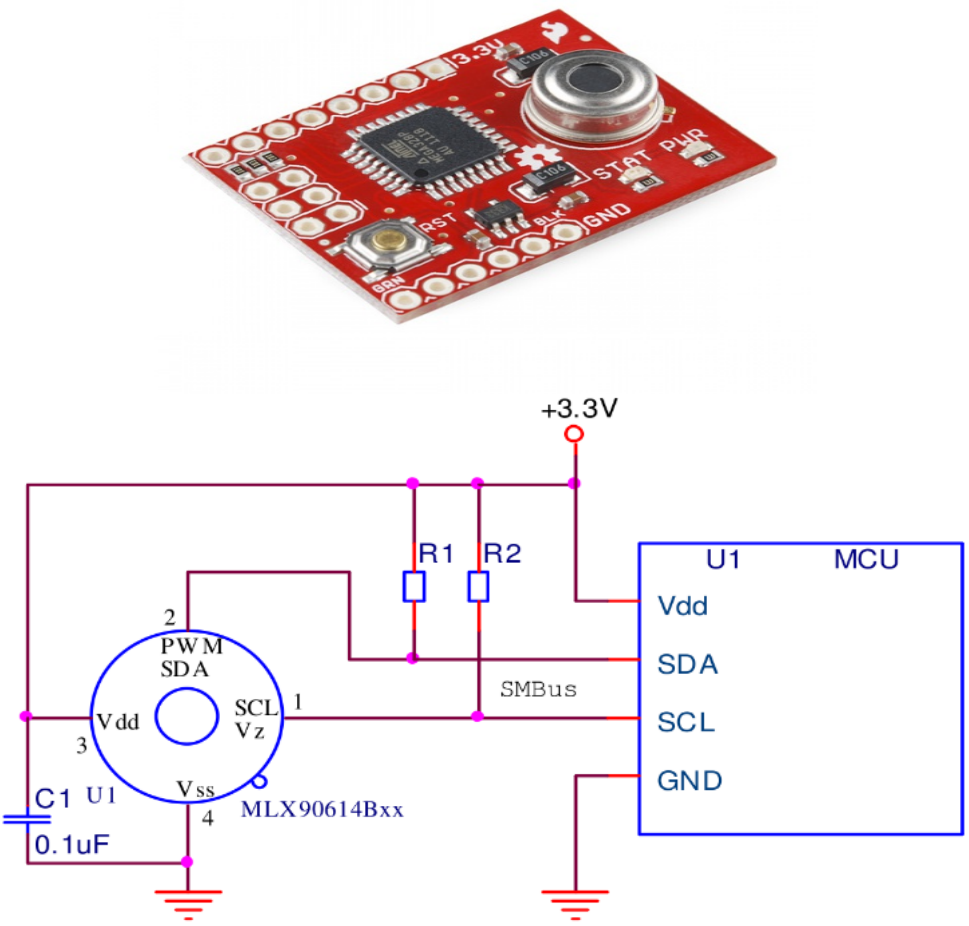

Figure 3 - Infrared Temperature Sensor (type: - MLX90614)

The temperature sensor in figure 3 was calibrated using an ordinary mercury thermometer. A code for the Arduino was developed and used to acquire the temperature readings.

The flowchart illustrating the temperature protection algorithm was as shown in figure 4 in the next page.

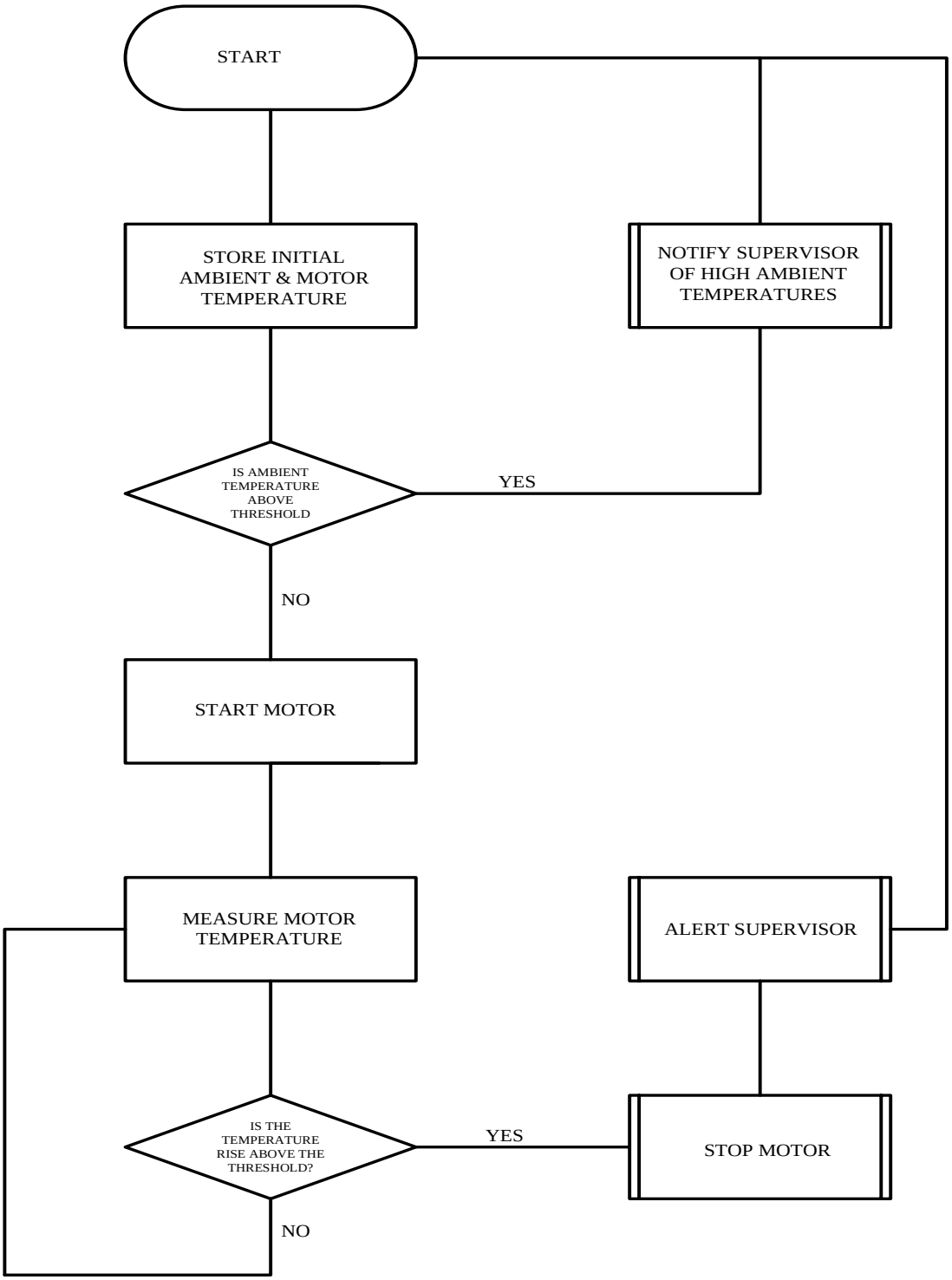

Figure 4 -3 Temperature Protection Flowchart

The flowchart gives the steps that were followed in the temperature protection algorithm of the system.

For overcurrent protection an Allegro ACS712 current sensor was used to measure the current being drawn by the motor. According to National Electrical Manufacturers Association (NEMA), motors with one horsepower output have a service factor of 1.15. A service factor of 1.15 was adopted for the motor which had a rated current of 0.15 A. The motor current protection unit was then designed not to

allow the motor current to go beyond 1.15 times of the rated current. In this case the current was capped at 0.1725 A. To avoid nuisance tripping of the motor when an overcurrent condition is detected, the current protection system was designed to delay for 5 seconds and if the overcurrent condition persisted then the motor would be disconnected. Figure 5, below shows the Allegro ACS712 (Allegro Microsystem, LLC, 2017).

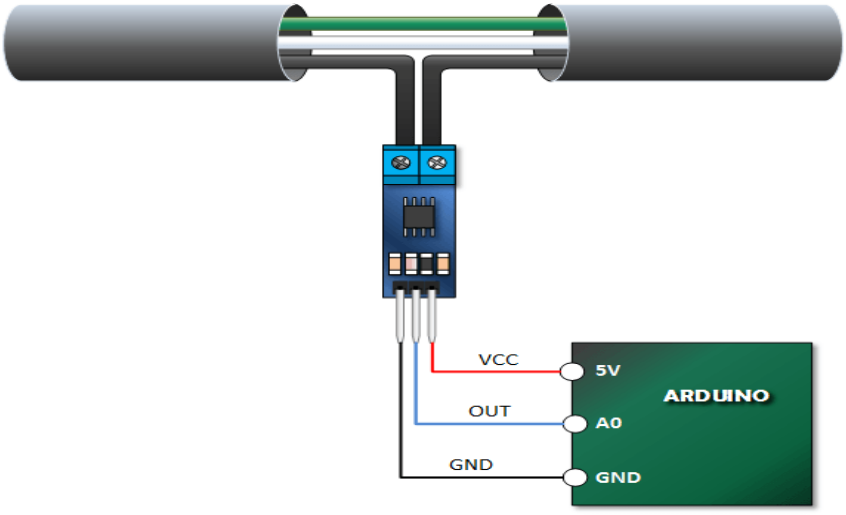

Figure 5 - Allegro Current Sensor (type:- ACS712)

The ACS712 current sensor measures AC current by calculating an RMS current value from the device readings.

For overvoltage and under voltage protection, a voltage sensor was used. Voltage limits were set and whenever these voltage levels where above or below set limits, the motor would be switched off through a relay. The voltage sensor used was a ZMPT101B voltage transformer shown below in figure 6.

Figure 6 - Voltage Sensor (type :-ZMPT101B)

The voltage sensor was connected to the Arduino and the voltage levels set and included in the code. Any voltages below or above the set limit would cause the motor to be switched off.

All the measured parameters; voltage, temperature and current, were stored for later analysis in a Micro SD storage card via the Arduino. The real time clock provided a

timestamp each time data was written to the card. The micro SD card shield was as shown below in figure 7.

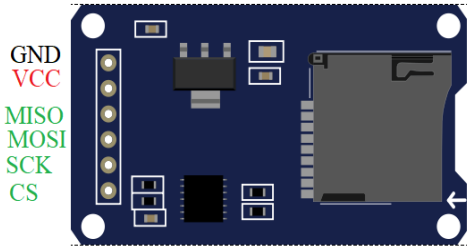

Figure 7 - Micro SD card Shield

The shield communicates with the Arduino via the serial peripheral interface (SPI) which is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers. With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. There are three lines common to all the devices:

- MISO (Master in Slave Out) - The Slave line for sending data to the master,
- MOSI (Master Out Slave in) - The Master line for sending data to the peripherals,
- SCK (Serial Clock) - The clock pulses which synchronize data transmission generated by the master

There is also a line specific to each device called the Slave Select (SS) / Chip Select (CS) which can be used by the master to enable and disable specific devices. When the SS/CS pin of a device is low, it communicates with the master and when it is high, it ignores the master. This allows one to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines (ARDUINO AG, 2017).

The real time clock (RTC) used to provide time stamps was as shown below in figure 8.

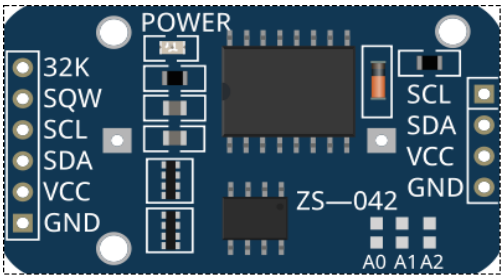

Figure 8 - Real Time Clock

The real time clock used was a DS1307 RTC which communicates with the microcontroller using I²C communication. I²C uses two lines to communicate with the slave devices; they are called Serial Clock (SCL) and Serial Data (SDA). The SCL line is the clock signal which synchronizes the data transfer between the devices on the I²C bus and it is generated by the master device and the SDA line carries the data. Since each device has a unique address, the master specifies the address of the device it wants to communicate with (Nedelkovski, .2015).

The circuit diagram of the data logging unit is shown below in figure 9.

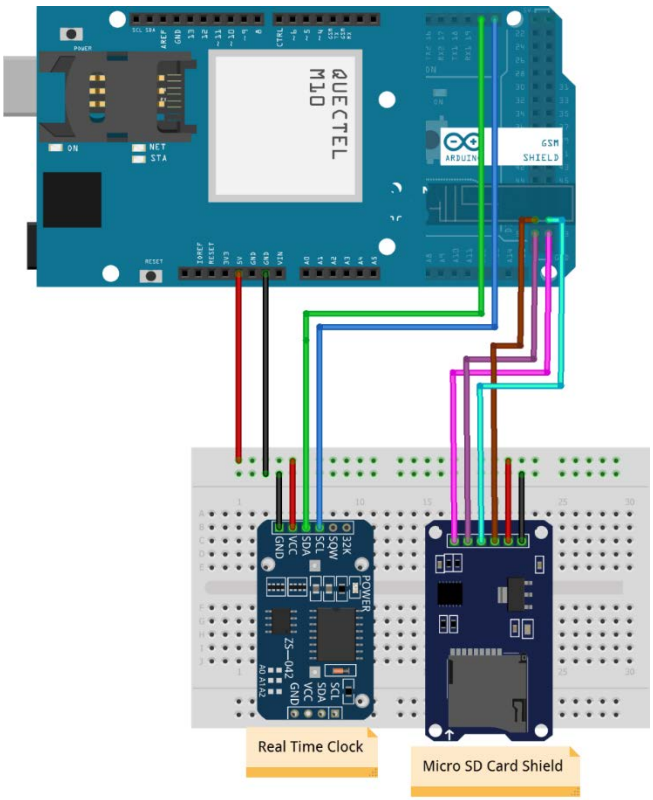

Figure 9 - Data Logging Unit

The data logging and function of the real time clock were verified when all the units were brought together to see if the values for voltage, current and temperature were saved in the memory card with timestamps. A code was then written for the full circuit.

The functional units were consolidated to give the full circuit of the system which was as shown in figure 10.

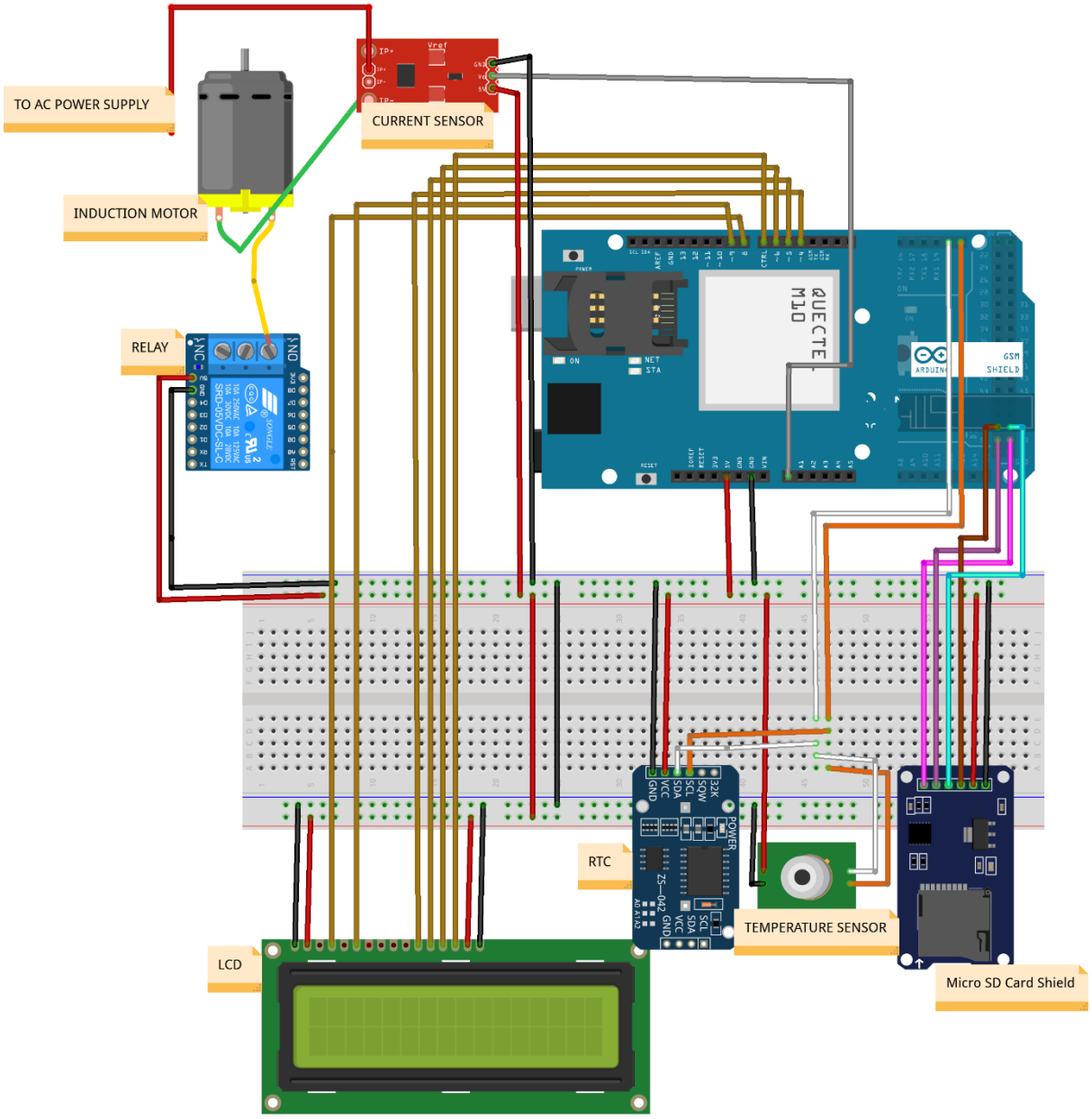

Figure 10 - Full Circuit Diagram

Figure 10 shows a full circuit diagram illustrating the interconnections of the system. The functional units of the system were integrated including the data logging function to see how they work together. The setup was as shown in figure 11.

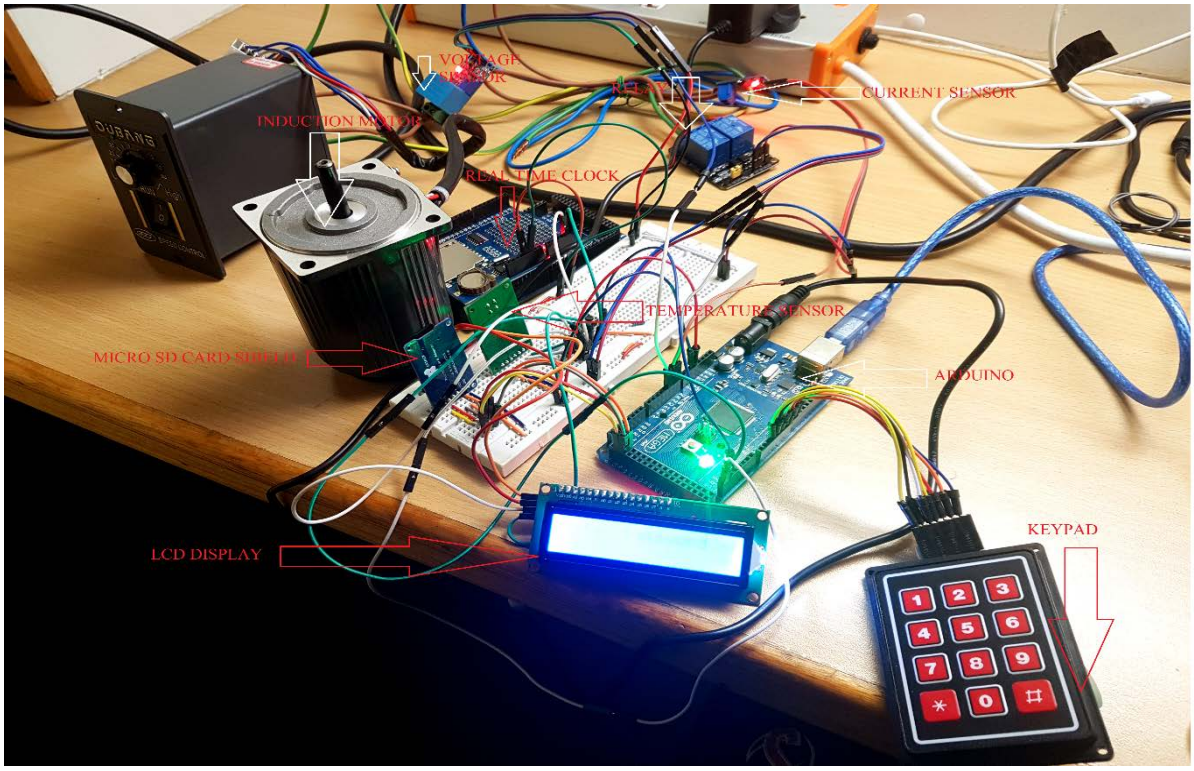

Figure 11 - Fully Integrated System

The system was tested to check on its response on the current protection unit, temperature protection unit and the overall system with all the units integrated. The motor used for the prototype was a single phase 220 Volts, 6W induction motor. In the case of overcurrent, overvoltage and under-voltage, when detected, the system was designed to have a delay of up to 5 seconds and after these seconds, it would check if the condition still exists and if so, then the motor would trip. The value of the delay time could be changed to suit different practical conditions.

The Arduino Integrated Development Environment (IDE) serial monitor was also used to view the system prompts and to view the data being acquired by the system as well as its outputs.

Results and discussion

Some of the current and temperature monitoring results taken were considered. Figure 12 below shows the response of the system on the Serial Monitor due to an overcurrent condition.

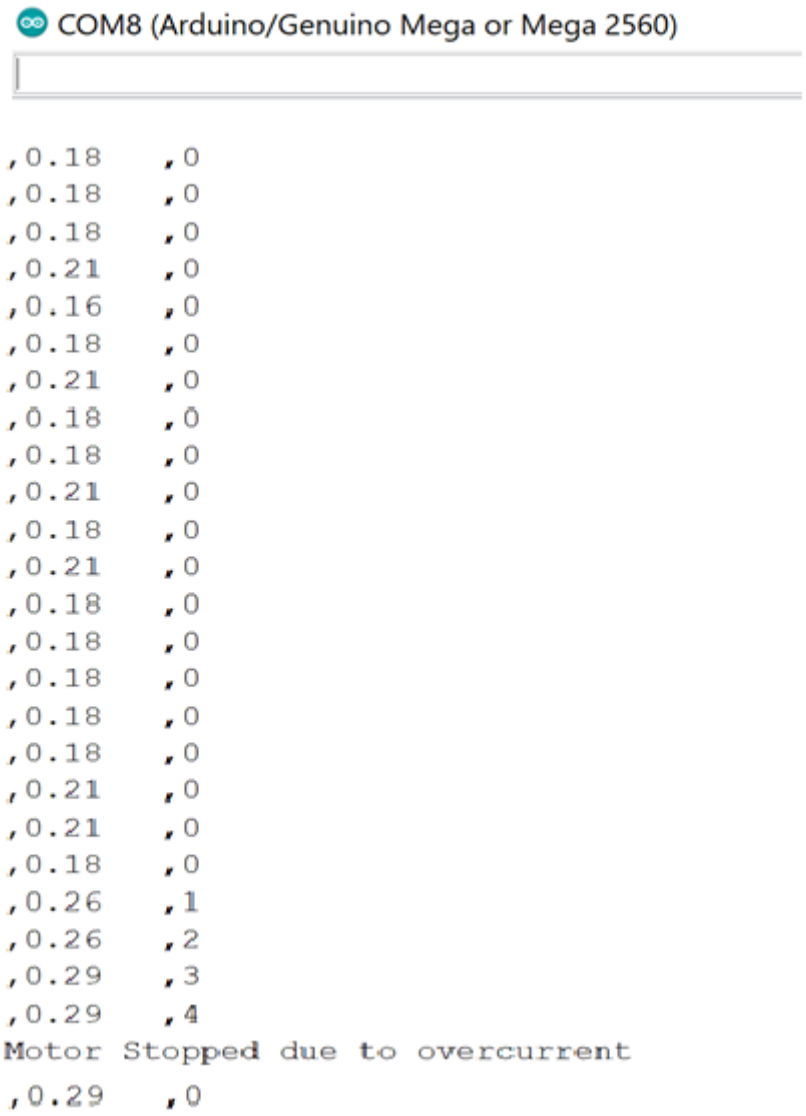

Figure 12 - Overcurrent Condition Shown On Serial Monitor

As shown in figure 12, the serial monitor showed the current level in Amps and the time in seconds from the system. The same was logged on the system. The overcurrent rose from 0.26 A to 0.29A, and occurred for 4 seconds and thereafter the motor was stopped on the 5th second as per the design.

Figure 13 shows the temperature values and a high motor temperature condition being displayed on the serial monitor.

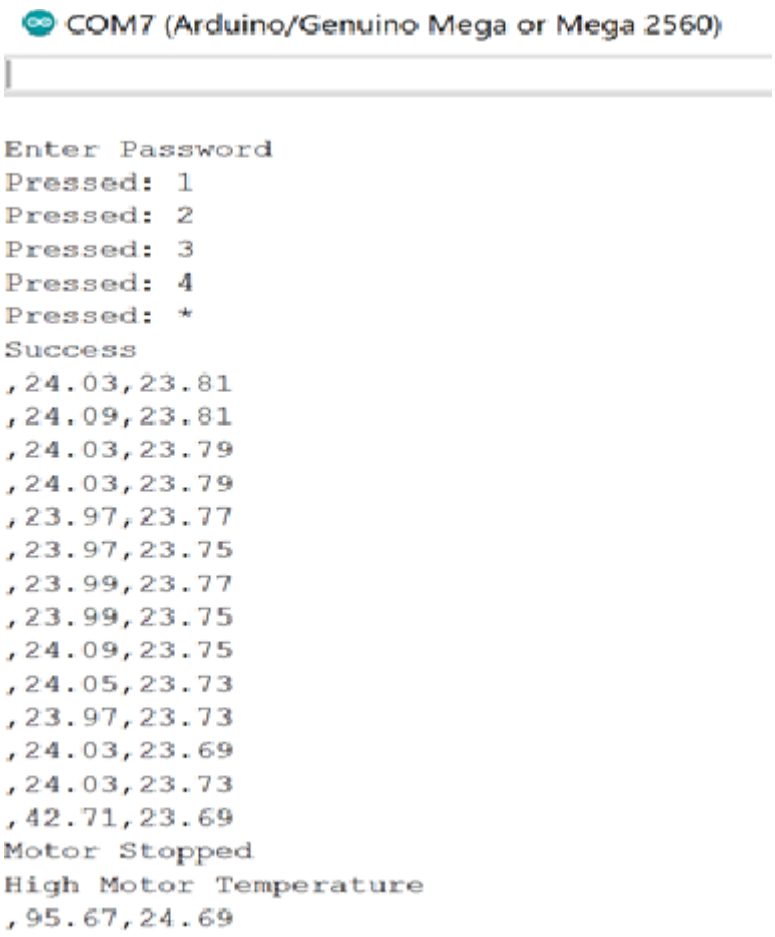

Figure 13 - Temperature Condition Shown On Serial Monitor

Figure 13 shows the motor and ambient temperature on the serial monitor. The left column shows the motor temperature and the right column shows the ambient temperature. The maximum ambient and motor temperatures were set at 50°C for this test and as illustrated the motor temperature rose from 24.03°C then to 42.71°C and the to 95.67°C, which was a high motor temperature and automatically brought the motor to a stop. This high temperature condition was also displayed on the LCD as shown in figure 14.

Figure 14 - High Temperature Condition On LCD

Data for both, temperature variations and current variations could also be displayed on the serial monitor. Table 1 below shows results displayed for a High Motor Temperature Condition test in the Integrated System as displayed in the serial monitor;

Table 1 - High Motor Temperature Condition tested in the Integrated System

Time(s)	Voltage	Current	Motor Temperature	Ambient Temperature	Event Seconds
12	211.73	0.21	22.35	24.75	0
14	213.24	0.18	22.35	24.77	0
15	208	0.21	22.37	24.77	0
16	209.58	0.18	22.35	24.75	0
17	211.22	0.18	22.41	24.77	0
19	211	0.21	22.43	24.75	0
20	212.95	0.21	22.47	24.77	0
21	206.79	0.21	22.43	24.77	0
22	210.61	0.21	22.43	24.77	0
24	211.29	0.21	22.19	24.81	0
25	211.92	0.18	22.1	24.77	0
26	208.22	0.18	22.37	24.77	0
27	212.29	0.18	22.51	24.75	0
29	208.12	0.21	22.43	24.77	0
30	209.17	0.21	22.29	24.77	0
31	210.03	0.18	22.43	24.77	0
33	207.55	0.18	76.99	25.09	Motor Stopped

Table 1, shows currents' flowing through the motor, ambient and motor temperature the motor is exposed to and a high temperature of 76.99°C that was detected by the

system. This high temperature caused the motor to be brought to a stop. The table further shows the temperature, current and voltage levels across the motor.

Graphically, the variations of the motor current when the current increases beyond the set limit is shown in figure 15.

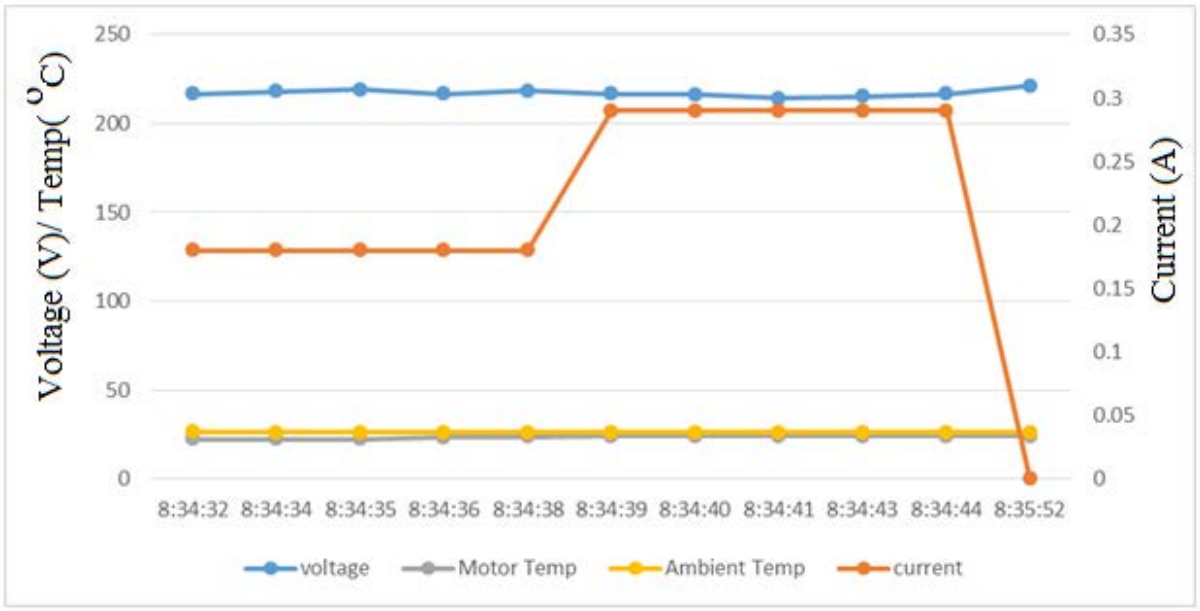

Figure 15 - High motor current condition.

From figure 15 it can be noted that when the current drawn by motor rose to 0.29A, the motor current dropped to zero, indicating a high current flow through the motor.

The ambient temperature, motor temperature and the input supply voltage remained constant due to the fact that there were no disturbance with regards these parameters on the motor.

For high temperature variations, figure 16 illustrates the response of the system.

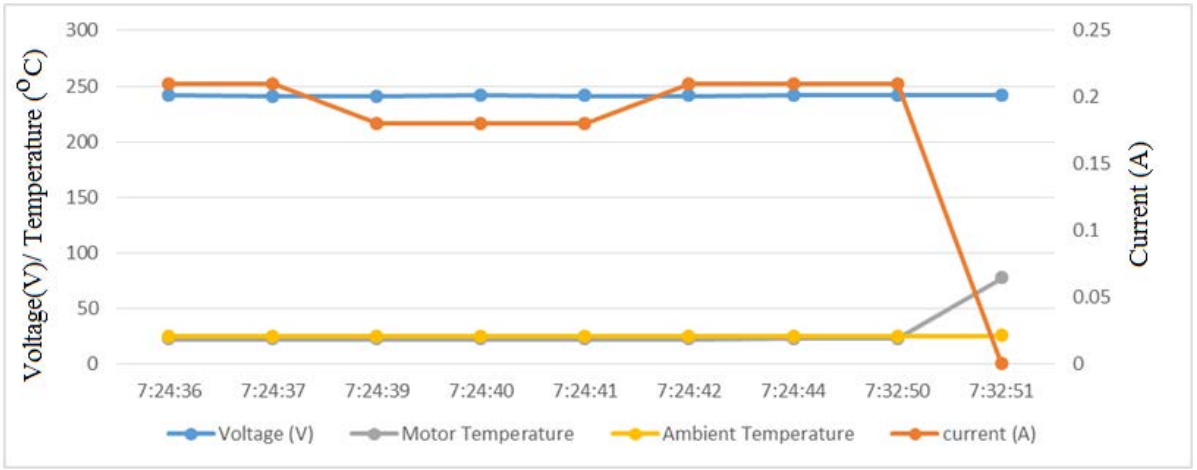

Figure 16 - Current response of motor to temperature changes

The moment the temperature starts to rise beyond the set value, the current supply to the motor is immediately cut through the use of a relay getting a command from the code stored in the arduino.

.

Conclusion

An automatic motor protection system was designed. The system could stop the motor when set conditions were exceeded. It is evident that electromechanical systems can be protected from overcurrent and high temperature levels when in use. Such systems would be able to save lots of capital in terms of purchasing new motors by industries if put into use.

References

Allegro Microsystems. (2017). *ACS712: Fully Integrated, Hall-Effect-Based Linear Current Sensor IC with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor.* Retrieved from: http://www.allegromicro.com/en/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712.aspx

ARDUINO AG. (2012). *Arduino MEGA 2560 & Genuino MEGA 2560.* Retrieved from https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

ARDUINO AG. (2017). *SPI library.* Retrieved from: https://www.arduino.cc/en/reference/SPI.

Bishop T. H. (2013). *Increase of motor reliability by monitoring and reducing operating temperature:* Retrieved from http://www.plantengineering.com/single-article/increase-motor-reliability-by-monitoring-and-reducing operatingtemperature/8fbd6b07e6090125cb576f9e667d8ca4.html

Craig. W and Multilin. GE. (2010). *Motor Protection Principles.* Retrieved from https://www.l-3.com/private/ieee/Motor%20Protection%20Principles.pdf.

Cowern E. (2000). *The Highs and Lows of Motor Voltage.* Retrieved from http://ecmweb.com/design/highs-and-lows-motor-voltage.

Nedelkovski, D. (2015). *How I2C Communication Works and How To Use It with Arduino. How to Mechatronics.* Retrieved from http://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/.

Rockwell Automation. (2016).*Electromechanical Temperature Controls*. Retrieved from https://ab.rockwellautomation.com/Sensors-Switches/Temperature-Sensors/Electromechanical.