Remote Battery Bank Monitoring System

(Project Report)

by

Shri Chitransh Lodha (31989650)

Operations Officer

Under the guidance of

Shri Rakesh Ramesh Wagh

Senior Station Manager – Pipeline (I/C)

HINDUSTAN PETROLEUM CORPORATION LIMITED

MANGALORE HASSAN MYSORE BANGALORE LPG P/L

MYSORE RECEIVING STATION

MYSORE – 570016, KARNATAKA, INDIA

ACKNOWLEDGMENT

I express my sincere gratitude towards Shri Rakesh Ramesh Wagh -Senior Station Manager Pipeline (I/C) MHMBPL Mysore and Shri M S Karthi - Sr. Manager Operations (MHMBPL MDS) for their valuable guidance and support through all the stages of the project without which the completion of this project was not possible.

I would also like to extend my sincere thanks to Shri Devesh Choudhary (Manager Operations- MHMBPL MRS) and Shri Sidda Raghavendra (Manager Operations – MHMBPL MRS) for their constant support and co-operation in completing my project.

Chitransh Lodha

Operations Officer
Mysore Receiving Station
MHMBPL
Hindustan Petroleum Corporation Limited

TABLE OF CONTENTS

S.No	Title	Pages
1.	Philosophy	4-5
2.	System Overview	6
3.	Introduction - Batteries	7-10
4.	Temperature Effect on Battery	11-12
5.	Sensors	13-16
	1. Voltage Sensing	13
	2. Current Sensing	14
	3. Temperature Sensing	15-16
6.	Arduino Mega	17-18
7.	NodeMCU	19
8.	ThingSpeak Platform	20
9.	Liquid Crystal Display	21
10.	Results and Conclusion	22-24
11.	References	25
12.	Appendix	26-32

PHILOSOPHY

Mangalore Hassan Mysore Bangalore LPG Pipeline (MHMBPL) was commissioned in the year 2016 for feeding Mysore LPG plant (MLP) and Yediyur LPG plant (YLP). Total length of the Pipeline is 357 Km, 256 KM from Mangalore to Yediyur with a tap off at Hassan and a spur line of 101 Km from Hassan I.P Station (HIPS) to Mysore Receiving Station (MRS). The cross country pipeline has Main dispatch station at Mangalore (MDS),2 receiving stations MLBP and YLBP,2 Intermediate Pigging stations at Neriya (NIPS) and Hassan(HIPS) and 18 sectionalizing valve stations (SV) enroute the 357 Km pipeline. SV stations are provided to sectionalize the pipeline during any emergency to reduce the release of product to the atmosphere.

All the main stations as well as SV stations have been provided with state-of-the-art equipment and infrastructure for smoothly carrying out the pipeline transfer (PLT) operation. To ensure uninterrupted power supply to the equipment and devices at these stations sufficient number of battery banks have been provided at each of the SV station and main stations with details given below:

MRS Main Station			
Battery Bank	**Rating**	**Number of cells**	**Battery Type**
Mini UPS Battery Bank	120 V, 120 Ah	10 Nos of cells rated 12V each	Lithium Ion
Main UPS Battery Bank	330 V, 200 Ah	144 Nos of cells rated 2.25V each	Tubular VRLA
Telecom Bank	48 V, 150 Ah	25 Nos of cells rated 2.25V each	Tubular VRLA
Typical SV Station			
Battery Bank	**Rating**	**Number of Cells**	**Battery Type**
Primary Bank	370V, 120 Ah	163 Nos of cells rated 2.25V each	Tubular VRLA
Secondary Bank	370V, 120 Ah	163 Nos of cells rated 2.25V each	Tubular VRLA

Table 1: Battery Bank Details of MRS and SV stations

To ensure battery health and as per guidelines in IMS manuals, present battery bank maintenance activity at all main stations as well as SVs is being carried out on a monthly basis by physically visiting each individual station and checking the individual battery health parameters like cell voltage, electrolyte level, specific gravity and visual checks for corrosion in connecting links etc. If undervoltage is detected for any battery or any damage is found, then the cell is replaced and further taken for maintenance and repair. But considering the large number of cells in each bank and spread across a huge geography, it poses a serious logistical and manpower challenge to maintain and ensure healthiness of this critical asset. Presently, there is no means to remotely monitor the individual battery health parameters like battery voltage, temperature and current from the control room. Each cell costs in the range of 10000-12000 INR and once a cell is damaged it also damages the other cells connected in series to it and thus degrading the overall performance of the battery bank. This can lead to costly downtimes and pose a risk to the safety of the pipeline and other stake holders too.

To install a remote battery monitoring system the UPS-OEM vendor was approached and has quoted the cost as Rs 18 Lakhs per SV station. The cost was very high compared to the original cost of the battery bank itself. The average cost of a 100 Ah Tubular VRLA battery is Rs 10,000 and once a cell is damaged it also affects the performance of other cells connected in series and hence degrading the overall performance of the battery bank.

Hence, in this project a low cost IOT based remote battery monitoring system is proposed with the purpose of remotely monitoring critical battery parameters like voltage, temperature and current for each cell and transmit the data to the cloud and finally make it available to the control room for remote monitoring. IOT or internet of things is a technology of the fifth industrial revolution and enables devices to be connected to the internet for transmitting and receiving information using various communication protocols.

As part of the project, a working prototype of the battery monitoring system for 10 cells was also developed to demonstrate the concept and the results of which have been attached in the report. The approximate cost of developing the prototype was just Rs 5,000.

SYSTEM OVERVIEW

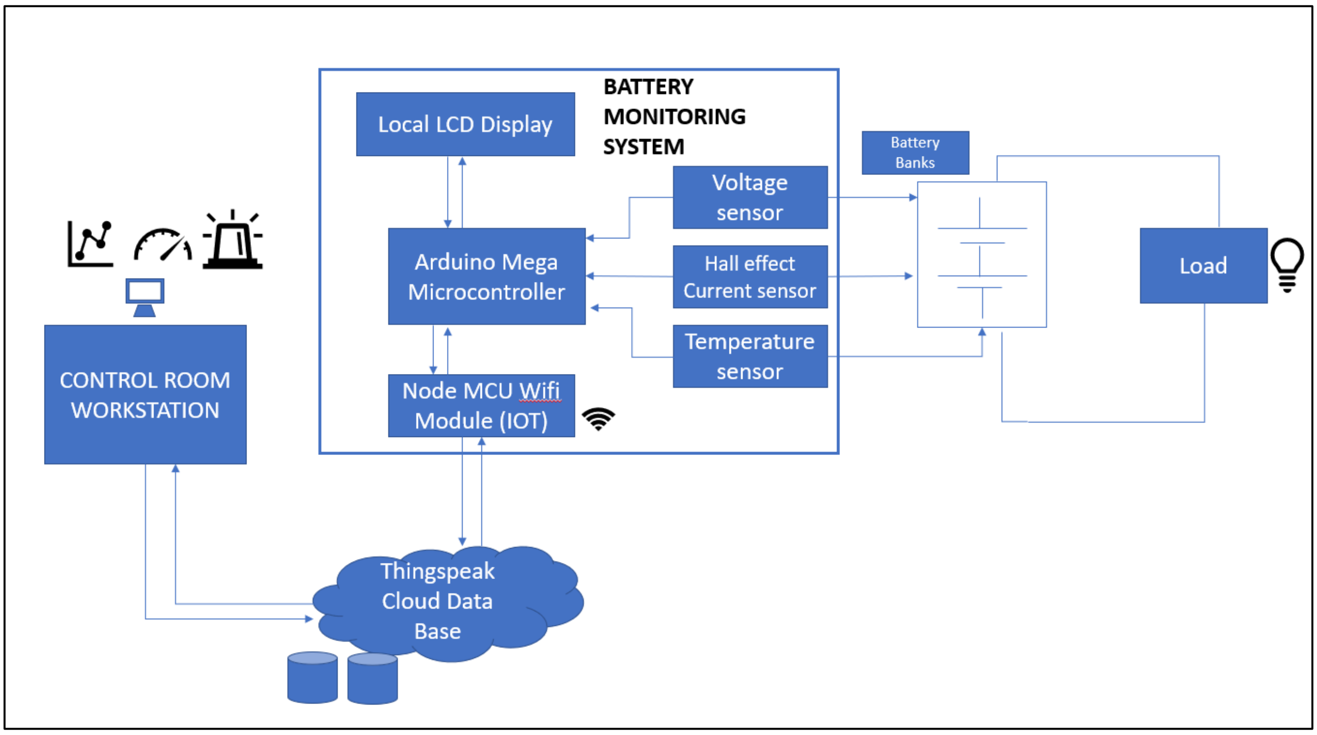

BATTERY MONITORING SYSTEM
Local LCD Display
Arduino Mega Microcontroller
Node MCU Wifi Module (IOT)
Voltage sensor
Hall effect Current sensor
Temperature sensor
Battery Banks
Load
CONTROL ROOM WORKSTATION
Thingspeak Cloud Data Base

Figure 1: Block diagram of Battery Monitoring System

The above block diagram shows the overall structure of the proposed battery monitoring system.

The system consists of various sensors to read the physical parameters of the battery and the data from these sensors is acquired and then further filtered for noise by the Arduino Mega microcontroller.

The Arduino Mega then transmits this data using serial communication to NodeMCU board in JSON encoded format. The NodeMCU has WiFi capabilities and is connected to a locally available WiFi Network with an active internet connection. The NodeMCU then sends the received data to the ThingSpeak server which then allows the data to be accessed from anywhere on the World Wide Web. Thus, the battery health can be remotely monitored conveniently and suitable corrective actions taken as and when required.

INTRODUCTION - BATTERIES

A cell is an electro-chemical device capable of supplying the energy that results from an electro chemical reaction to an external electric circuit.

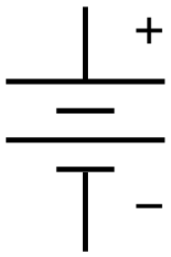
A battery is composed of one or more cells, connected either in parallel or series combination to obtain a required current/voltage capability.

When a battery is supplying electric power, its positive terminal is the cathode and its negative terminal is the anode. The terminal marked negative is the source of electrons that will flow through an external electric circuit to the positive terminal. When a battery is connected go an external electric load, a redox reaction converts high-energy reactants to lower-energy products, and the free energy difference is delivered to the external circuit as electrical energy.

Batteries are mainly classified into two types- Primary (Single use) batteries and Secondary (rechargeable) batteries.

As the name suggests, primary batteries once used are discarded as the electrode material is irreversibly changed during discharge. A common example is the alkaline battery used in flashlights and other portable electronic devices.

A secondary battery can be discharged and recharged multiple times using an applied electric current, and the original composition of the electrode can be restored by reverse current. Common examples include the lead-acid batteries used in vehicles and lithium-ion batteries for portable electronics like laptops and mobile phones.



1. Electronic symbol of a battery

2. Various cells and batteries

Figure 2: A typical battery symbol and visuals

Batteries come in all sizes and capacities based on the application, ranging from miniature cells used in wristwatches, to small thin cells in smartphones, large lead acid batteries in vehicles and huge battery banks requiring separate rooms that provide standby or emergency power for computer data centres and telephone exchanges.

Principle of Operation

Batteries convert chemical energy directly to electrical energy. In most cases, the electrical energy released is the difference in the cohesive or bond energies of the metals, oxides, or molecules undergoing the electrochemical reaction. Batteries are designed so that the energetically favourable redox reaction can occur only when electrons move through the external part of the circuit.

A battery consists of voltaic cells which further contains two half-cells connected in series by a conductive electrolyte containing metal cations. One half-cell includes electrolyte and the negative electrode, the electrode to which anions migrate, the other half-cell includes electrolyte and the positive electrode, to which cations migrate.

Each half-cell has an electromotive force (emf E, measured in volts) relative to a standard and the net emf of the cell is the difference between the emfs of its half-cells.

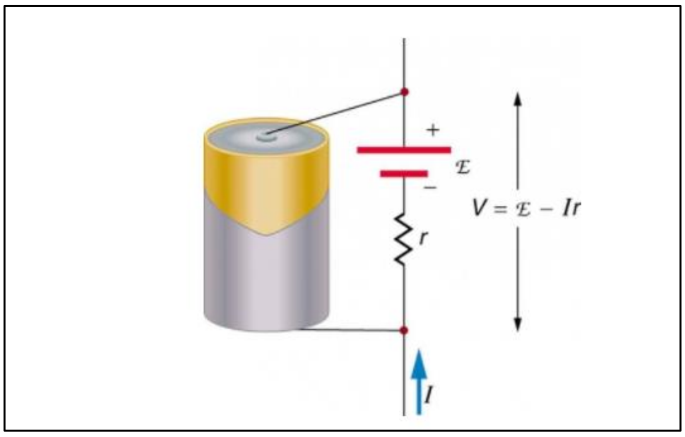


Figure 3: Terminals of a cell

The electrical driving force across the terminals of a cell is known as the terminal voltage (V) measured in volts. The terminal voltage of a cell that is neither charging or discharging is called the open-circuit voltage and equals the emf of the cell.

Because of the internal resistance (r), the terminal voltage of a cell that is discharging is smaller in magnitude than the open circuit voltage and the terminal voltage of a cell that is charging exceeds the open-circuit voltage. An ideal cell has negligible internal resistance, so it would maintain a constant terminal voltage until exhausted and then dropping to zero.

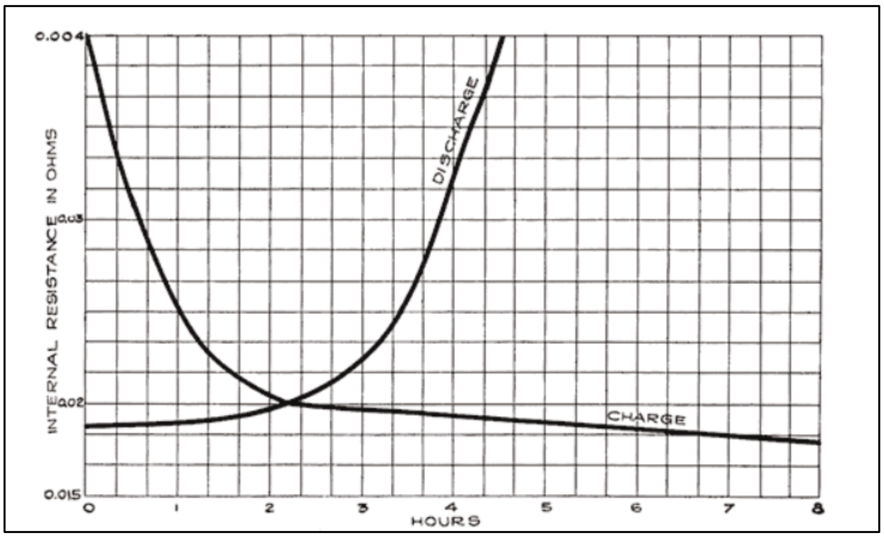


Figure 4: Variation of internal resistance with charging/discharging

In an actual cell, the internal resistance increases under discharge and the open-circuit voltage also decreases under discharge. Also, under charging the internal resistance of the battery decreases over time and hence the open circuit voltage increases under charging.

Battery Capacity

A battery's capacity is the amount of electric charge it can deliver at the rated voltage. The more electrode material contained in the cell, the greater its capacity. A small cell has less capacity than a larger cell with the same chemistry, even though they may develop the same open-circuit voltage.

Battery capacity is measured in units such as ampere-hour (Ah) or mili-ampere-hours (mAh).

For example: A battery rated at 100 Ah can deliver 5 A over a 20 Hr period at room temperature.

The fraction of the stored charge that a battery can deliver depends on multiple factors, including battery chemistry, the rate at which the charge is delivered (current), the required terminal voltage, the storage period, ambient temperature and other factors.

The higher the discharge rate the, the lower the capacity. The relationship between current, discharge time and capacity of a lead acid battery is approximated by Peukert's law

$$t = \frac{Q_P}{I^k}$$

Where

Q_P is the capacity when discharged at a rate of 1 A.

I is the current drawn from battery (A).

t is the amount of time (in hours) that a battery can sustain.

k is a constant around 1.3

Batteries that are stored for a long period or that are discharged at a small fraction of the capacity lose capacity due to the presence of generally irreversible side reactions that consume charge carriers without producing current. This phenomenon is known as **internal self-discharge.**

C-rate

The C-rate is a measure of the rate at which a battery is being charged or discharged. It is defined as the current through the battery divided by the theoretical current draw under which the battery would deliver its nominal rated capacity in one hour.

Because of internal resistance loss and the chemical processes inside the cells, a battery rarely delivers nameplate rated capacity in only one hour.

Typically, maximum capacity is found at a low C-rate, and charging or discharging at a higher C-rate reduces the usable life and capacity of a battery. Manufacturers often publish datasheets with graphs showing capacity versus C-rate curves. C-rate is also used as a rating on batteries to indicate the maximum current that a battery can safely deliver in a circuit.

C-rate is the rate at which a battery is discharged relative to is maximum capacity. For a fully charged battery rated at 1 Ah, rated at 1C then it will provide 1A for one hour. The same battery discharging at 0.5C or C/2 should provide 500mA for two hours, and at 2C it delivers 2A for 30 minutes.

$$C - rate = \frac{Current\ of\ charge\ or\ discharge\ (Amps)}{Rated\ battery\ capacity\ (Ah)}$$

Smaller batteries are commonly rated at 1C rating, which is also the one-hour rate. For example: If a battery is labelled 3000 mAh at the one-hour rate, then the 1C rating is 3A i.e., it is rated to supply 3 A for 1 hour before it fully gets discharged. Lead acid batteries on the other hand are rated at a very low discharge rate 0.05C or C20 which is a 20-hr rate. This means it is designed to supply rated current for 20hrs before it fully gets discharged. If it is discharged at a higher rate then its performance may decrease. On the other hand, lithium-ion batteries have much higher discharging C-rates.

C-rate	**Time**
5C	12 min
2C	30 min
1C	1h
0.5C or C/2	2h
0.2C or C/5	5h
0.1C or C/10	10h
0.05C or C/20	20h

Table 2 : C-rate and service times when charging and discharging batteries of 1Ah

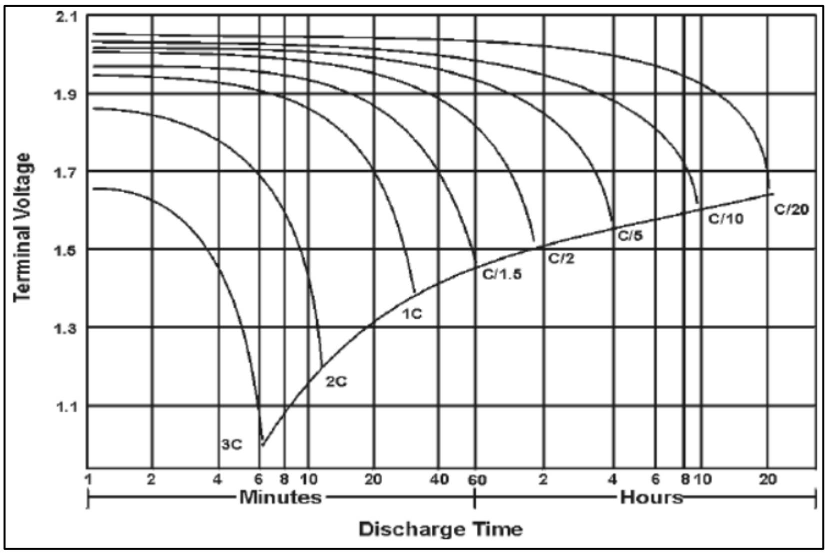


Figure 5: Typical discharging curves of a Tubular VRLA battery

TEMPERATURE EFFECT ON BATTERY

With the advent of global climate change, there are increasingly chances of extreme temperatures (both hot and cold) being recorded. This change can significantly effect the performance capabilities of batteries by effecting battery lifecycle, depth of discharge, performance and safety capabilities. Hence, it becomes all the more important to have an effective monitoring mechanism to prevent and avoid

The ambient temperature of the cells is a major factor in the lifetime of any battery system.

Effects of high temperature

When temperatures increase this affects the chemical reactions that occur inside a battery. As the temperature of the battery increases the chemical reactions inside the battery also quicken. The anticipated battery life is specified by the manufacturer for batteries installed in an environment at or near the reference temperature of 77°F (25°C) , above this temperature the battery life is reduced. The chief aging mechanism is accelerated viz- corrosion of the positive plates, grid structure, and strap, which increases exponentially as a function of temperature. The chart below defines the reduction in lifetime through operating of elevated temperatures.

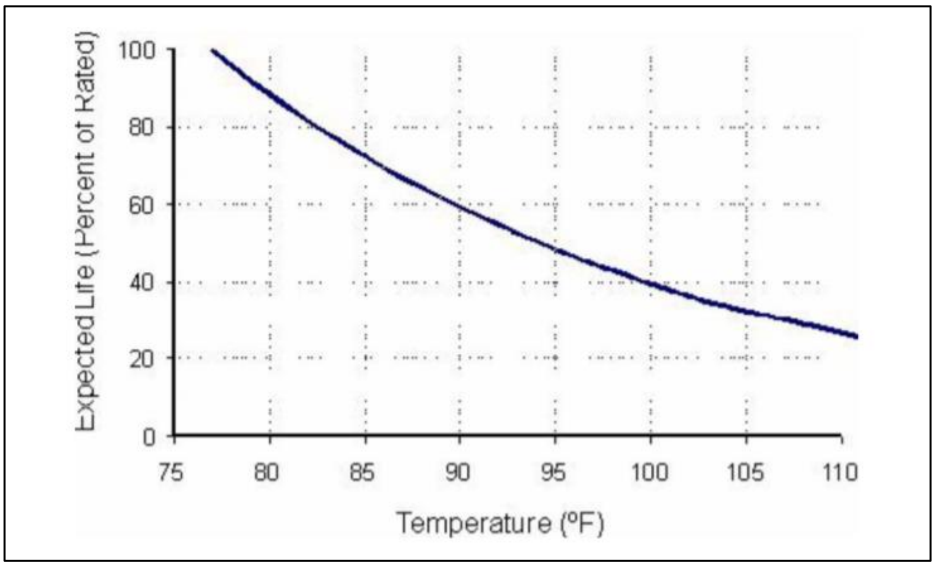


Figure 6: Battery Life with change in temperature

A general rule of thumb for a lead-acid battery is that the battery life is halved for every 15°F (9.3 °C) above 77 °F (25 °C). Thus, a battery rated for 5 years of operation under ideal conditions at 77°F (25°C) might only last 2.5 years at 95°F (35°C).

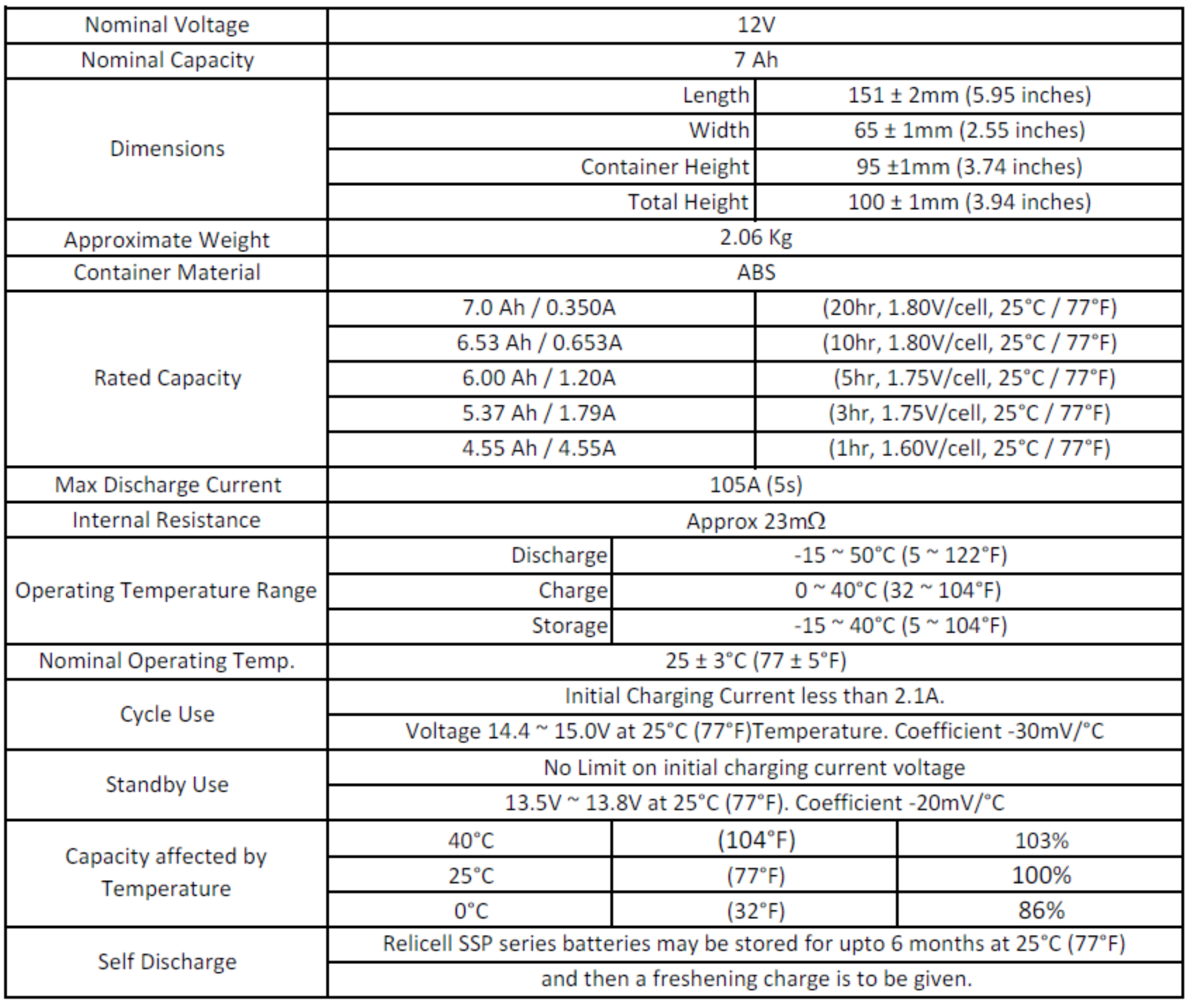
Effects of low temperature

Prolonged exposure to cold temperatures also has a big impact on battery performance and safety. When temperatures drop the internal resistance of the battery is increased. This means that it requires more effort by the battery to charge, in turn lowering the capacity. However, it is important to note that the loss of capacity also depends on the charge and discharge rates and the effect of the cold weather is different for batteries made with different chemistries. For example, a lead-acid battery may provide just half the nominal capacity at 0 °F.

It’s very important to be aware of the charging temperatures that a battery can accommodate. If batteries don’t operate at the accepted temperature, charge acceptance will be decreased because ion combination will be slower. Forcing high current can build up pressure causing explosions of sealed batteries.

Hence, it is important to devise and place a efficient temperature monitoring system for the batteries to prolong and maximize battery life.

For the prototype, we are using four numbers of 12V 7Ah Relicell batteries. The technical specifications of the battery are given below for reference.



Nominal Voltage	12V		
Nominal Capacity	7 Ah		
Dimensions	Length	151 ± 2mm (5.95 inches)	
	Width	65 ± 1mm (2.55 inches)	
	Container Height	95 ±1mm (3.74 inches)	
	Total Height	100 ± 1mm (3.94 inches)	
Approximate Weight	2.06 Kg		
Container Material	ABS		
Rated Capacity	7.0 Ah / 0.350A	(20hr, 1.80V/cell, 25°C / 77°F)	
	6.53 Ah / 0.653A	(10hr, 1.80V/cell, 25°C / 77°F)	
	6.00 Ah / 1.20A	(5hr, 1.75V/cell, 25°C / 77°F)	
	5.37 Ah / 1.79A	(3hr, 1.75V/cell, 25°C / 77°F)	
	4.55 Ah / 4.55A	(1hr, 1.60V/cell, 25°C / 77°F)	
Max Discharge Current	105A (5s)		
Internal Resistance	Approx 23mΩ		
Operating Temperature Range	Discharge	-15 ~ 50°C (5 ~ 122°F)	
	Charge	0 ~ 40°C (32 ~ 104°F)	
	Storage	-15 ~ 40°C (5 ~ 104°F)	
Nominal Operating Temp.	25 ± 3°C (77 ± 5°F)		
Cycle Use	Initial Charging Current less than 2.1A.		
	Voltage 14.4 ~ 15.0V at 25°C (77°F)Temperature. Coefficient -30mV/°C		
Standby Use	No Limit on initial charging current voltage		
	13.5V ~ 13.8V at 25°C (77°F). Coefficient -20mV/°C		
Capacity affected by Temperature	40°C	(104°F)	103%
	25°C	(77°F)	100%
	0°C	(32°F)	86%
Self Discharge	Relicell SSP series batteries may be stored for upto 6 months at 25°C (77°F)		
	and then a freshening charge is to be given.		

Table 3: Relicell Battery technical specifications

SENSORS

Various sensors are deployed to acquire the battery health data viz. voltage, temperature and current and are discussed in detail in the section below.

1. VOLTAGE SENSING

A resistive voltage divider circuit is used to read the voltage level of each battery. The battery nominal voltage is 12V and since the batteries are connected in series the terminal voltages will be in the multiples of 12V which are 24V,36V, 48V etc. But, the Analog pins of Arduino Mega can read a maximum of 5V.
Hence, we use a voltage divider which steps down the voltage being measured to within the range of the Arduino analog inputs (0-5V). Suitable resistor combinations are used to step down the voltage based on the maximum voltage at the terminal and safety margin of atleast 14% is also considered for voltage surges/short circuit etc. Resistors are chosen which are easily available and widely used in the industry.

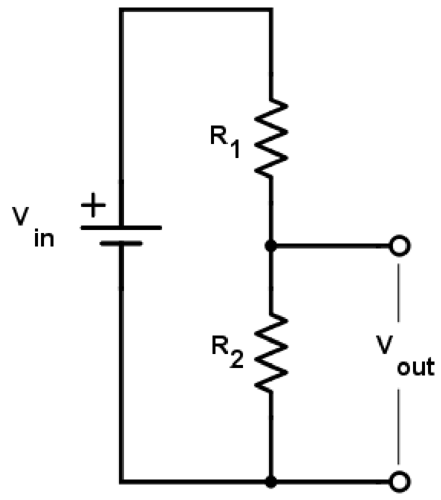


Figure 7: Standard Voltage Divider Circuit

$$V_{out} = V_{in}\frac{R_2}{R_{1+}R_2} = V_{in} * \beta$$

Voltage Terminal	$V_{in\,max}$	R_1	R_2	β	$V_{out\,max}$	**Safety Margin**
Battery 1	12V	100 kΩ	10 kΩ	0.091	1.092V	3.57
Battery 2	24V	100 kΩ	10 kΩ	0.091	2.184V	1.29
Battery 3	36V	100 kΩ	10 kΩ	0.091	3.276V	0.53
Battery 4	48V	100 kΩ	10 kΩ	0.091	4.368V	0.14

Table 4: Selection of resistor values

The code in Arduino sketch is then used to compute the actual voltage being measured. Further, to provide reverse current protection to batteries and Arduino, we use P-N junction Diodes (IN4007). They block passage to any reverse current and thus effectively protect the microcontroller and batteries.

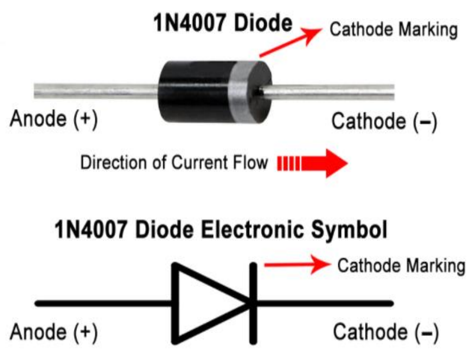


Figure 8: I4007 (P-N Junction diode) and Ceramic Capacitor

Ceramic capacitors of 0.1 µF are connected across resistor output to avoid any ripple in DC voltage output to the microcontroller.

2. CURRENT SENSING

Battery current is required to understand the charging or discharging profile of the battery bank.
Hall Effect based current sensor ACS712 (Range of -5A to +5A) is used to measure load current. The Allegro ACS712 provides economical and precise solutions for AC and DC current sensing in industrial, commercial and communication systems. It consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

DS18B20 Sensor Specifications

- Current sensing range of ±5A
- 80kHz bandwidth
- 66 to 185 mV/A output sensitivity
- The low-noise analog signal path
- Device bandwidth is set through the new FILTER pin
- 1.2 mΩ internal conductor resistance
- Stable output offset voltage.
- Near zero magnetic hysteresis

It also provides isolation from the load as it is based on the hall effect principle. It outputs Analog voltage (0-5V) based on the current flowing through the wire. It has two terminal connectors with mounting screws through which the wire which carries current is to be connected. The sensor needs to be connected in series to the load current. There are three other pins where the Vcc is connected to +5V to power the module and ground is connected to the ground of the MCU (system). The analog voltage given out by the ACS712 module is read using any analog pin on the Microcontroller. The scale factor for 5A sensor is 185 mV/A.

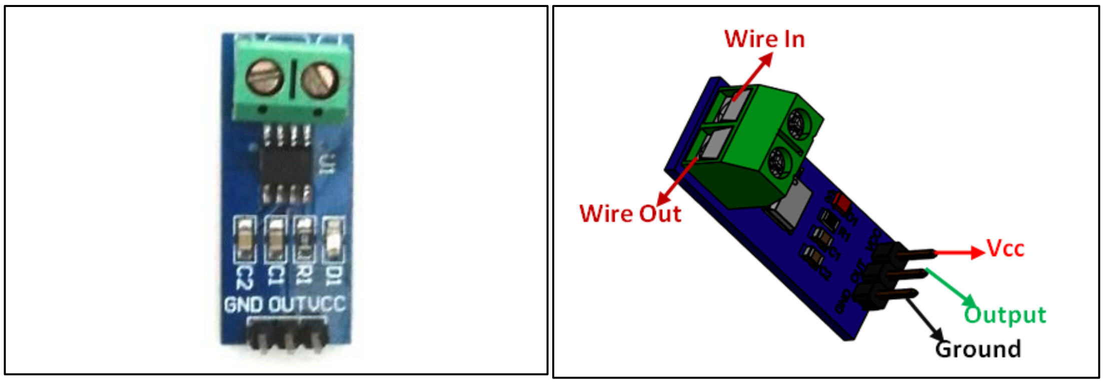


Figure 9: ACS712 wiring diagram

3. TEMPERATURE SENSING

Temperature data of a battery is very critical as explained previously. To measure the temperature of the battery surface and especially at the terminals, we use a rugged and sturdy sensor which can withstand harsh situations also. The DS18B20 is a one-wire programmable temperature sensor from maxim integrated. It is widely used to measure temperature in hard environments like in chemical solutions, mines or soil etc. The construction is rugged and waterproof.

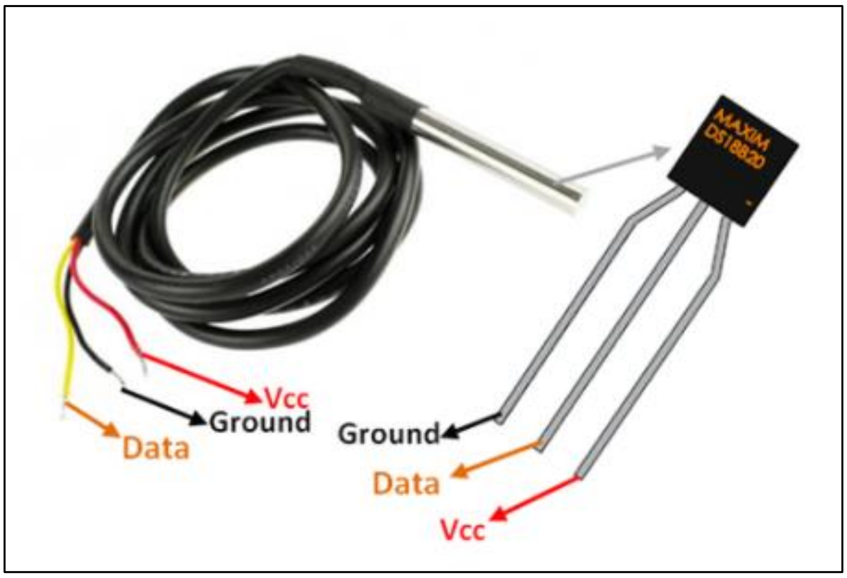


Figure 10: DS18B20 Temperature Sensor diagram

DS18B20 Sensor Specifications

- Programmable Digital Temperature Sensor

- Communicates using 1-Wire method
- Operating voltage: 3V to 5V
- Temperature Range: -55°C to +125°C
- Accuracy: ±0.5°C
- Output Resolution: 9-bit to 12-bit (programmable)
- Unique 64-bit address enables multiplexing
- Conversion time: 750ms at 12-bit
- Programmable alarm options
- Available as To-92, SOP and even as a waterproof sensor

It can measure a wide range of temperature from **-55°C to +125°C** with a decent accuracy of **±0.5°C.** Each sensor has a unique address and requires only one pin of the MCU to transfer data so it is the desirable option for measuring temperature at multiple points without blocking many digital pins on the microcontroller.

The sensor works with the method of one-wire communication. It required only the data pin connected to the microcontroller with a pull up resistor and the other two pins are used for power.

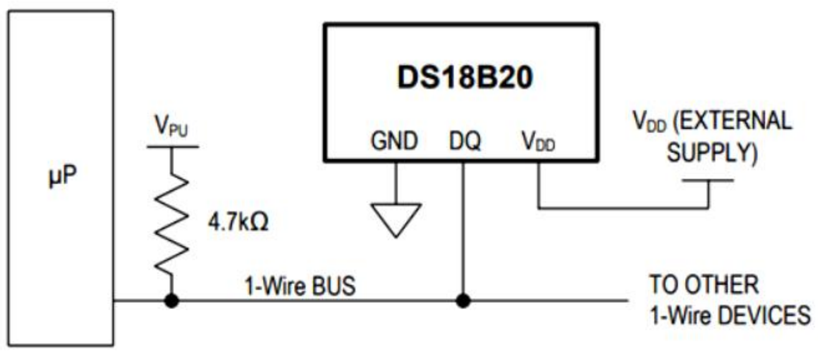


Figure 11: DS18B20 Interface diagram with Microprocessor

The pull-up resistor is used to keep the line in high state when the bus is not in use. The temperature value measured by the sensor will be stored in a 2-byte register inside the sensor. This data can be read by the using the 1- wire method by sending in a sequence of data.

ARDUINO MEGA

The heart of any smart device is the microcontroller and for the battery monitoring system also we need a powerful yet economical MCU.

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (Recommended)	7-12V
Input Voltage (Limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current Per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Table 5: Technical Specifications of Arduino Mega

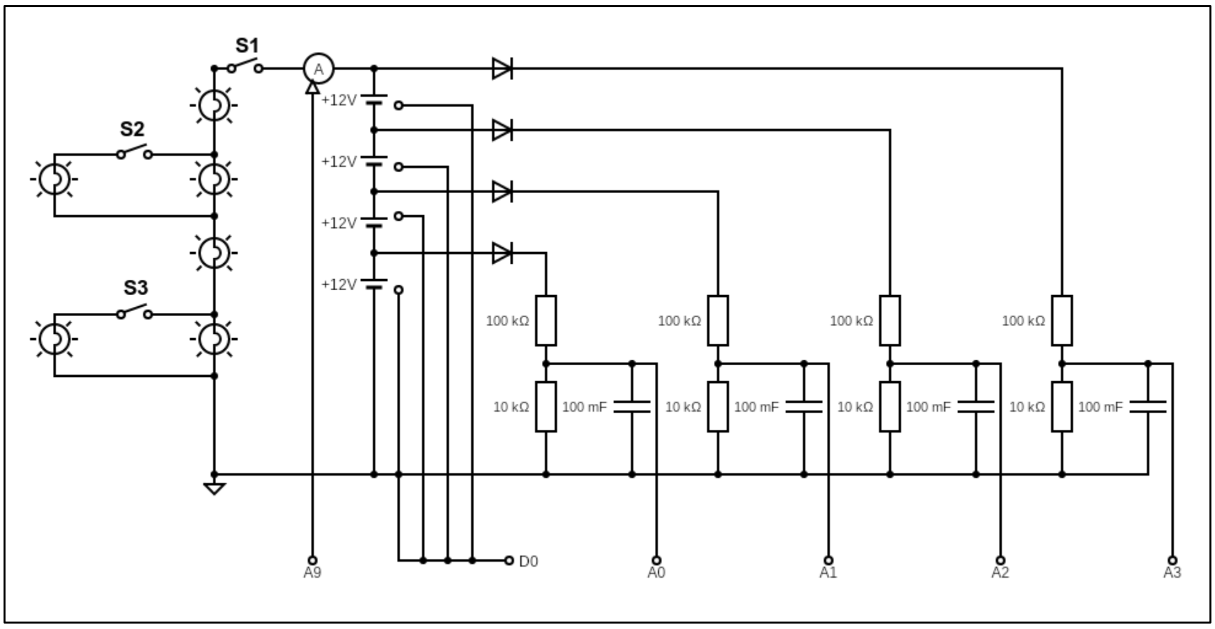


Figure 12: Arduino Sensor interface diagram along with DC Lighting loads and cells

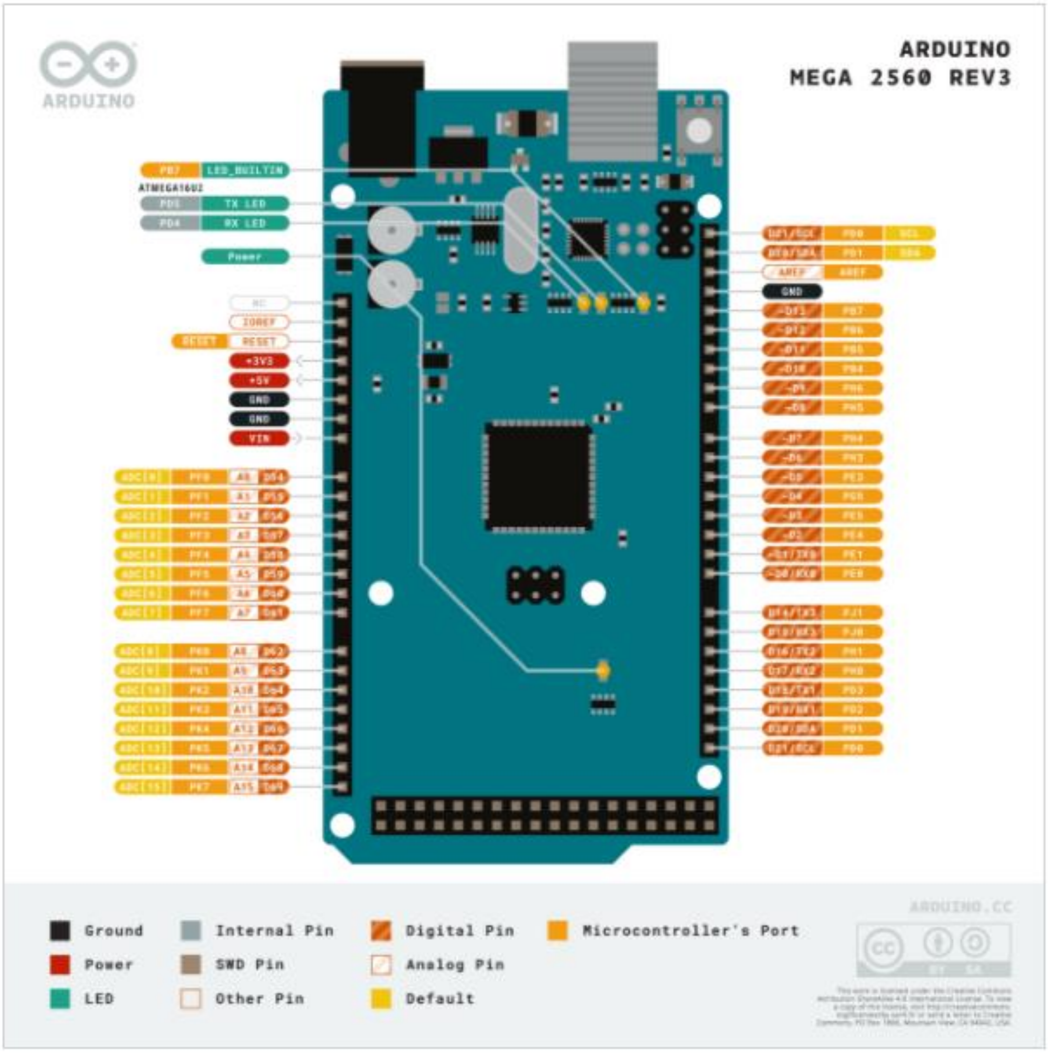


Figure 13: Arduino Mega Pin Out Diagram

The Arduino is interfaced with all the sensors as shown in Figure. 12 The code is attached in the appendix for reference.

The Arduino acquires the data from the sensors and then filters it and finally sends it in JSON encoded format to the NodeMCU for transmitting it to the cloud.

NODEMCU

Once the data from the sensors is acquired, we need to send to be able to access this data remotely as we may not be always be physically present near the device. Hence, we transmit this data to the cloud using NodeMCU.

NodeMCU is an open source Lua based firmware for the ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. NodeMCU is implemented in C and is layered on the Espressif NON-OS SDK. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for the Internet of Things (IoT) projects of all kinds.

Microcontroller	ESP-8266 32-bit
Operating Voltage	3.3V
Input Voltage (Recommended)	4.5V-10V
Digital I/O Pins	11
Analog Input Pins	1
Size	49mm x 26mm
Clock Speed	80 MHz
USB Connector	Micro USB
ADC Range	0-3.3V
UART/SPI/I2C	1/1/1
WiFi Built-In	802.11 b/g/n
Temperature Range	-40°C to +125°C

Table 6: Technical Specifications of NodeMCU

The NodeMCU acquires data from Arduino Mega using Serial communication and it is connected to a Local WiFi network. It then transmits this data to the ThingSpeak web server.

The code for NodeMCU is also attached in the appendix for reference.

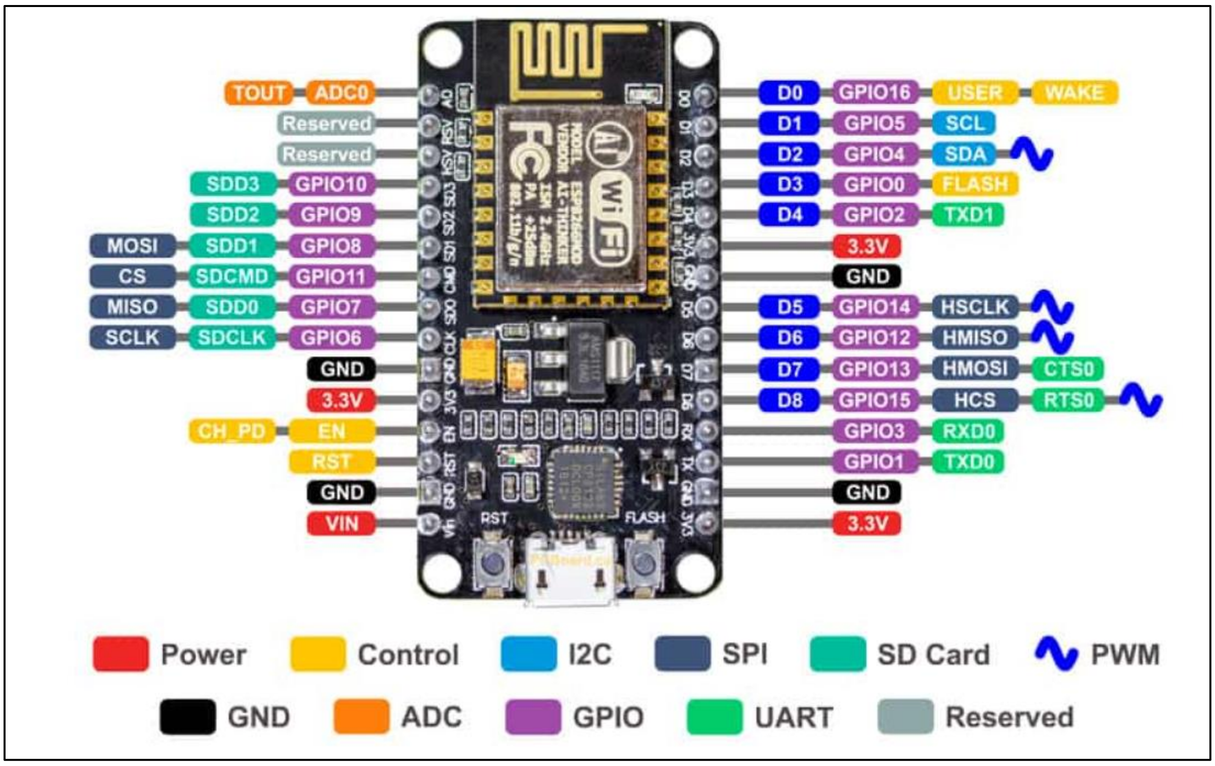


Figure 14: Pin Diagram of NodeMCU

THINGSPEAK

The data sent to the cloud is stored in a open source web server provided by ThingSpeak. ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

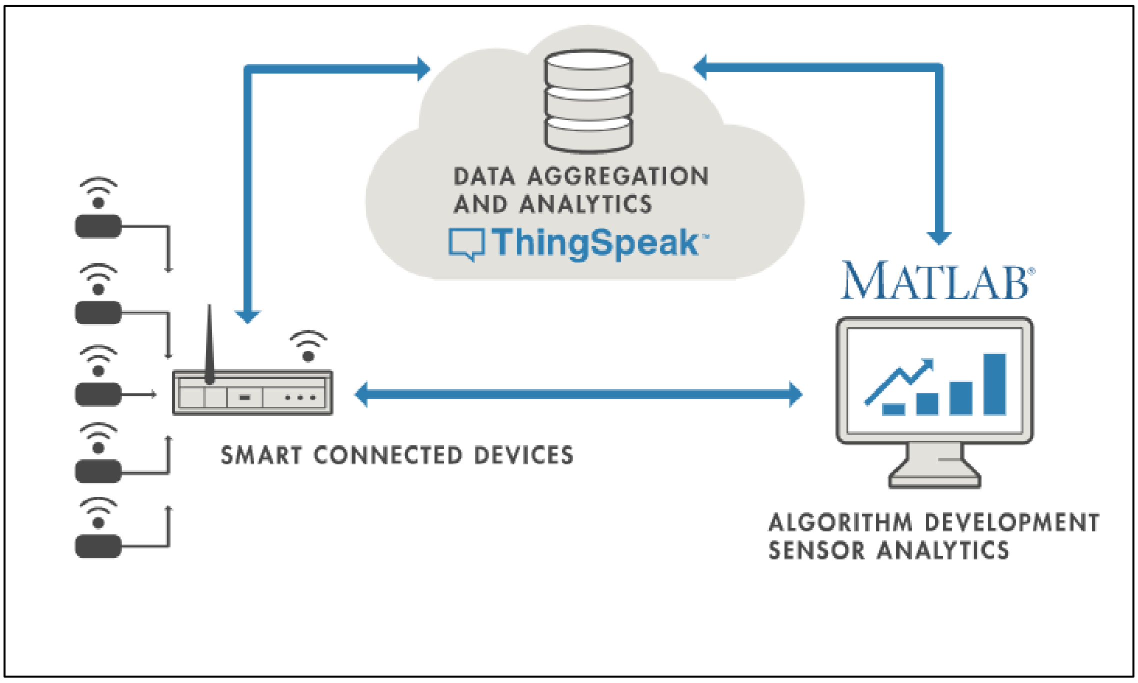


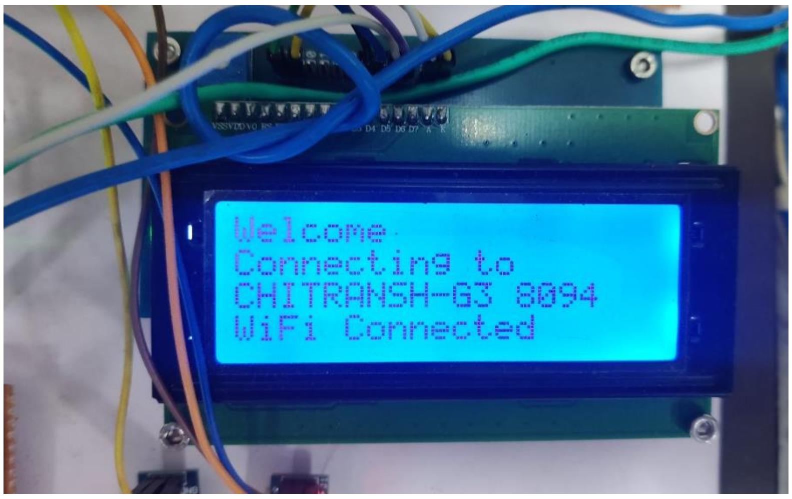
Figure 15: ThingSpeak working Overview

Some features of ThingSpeak are:

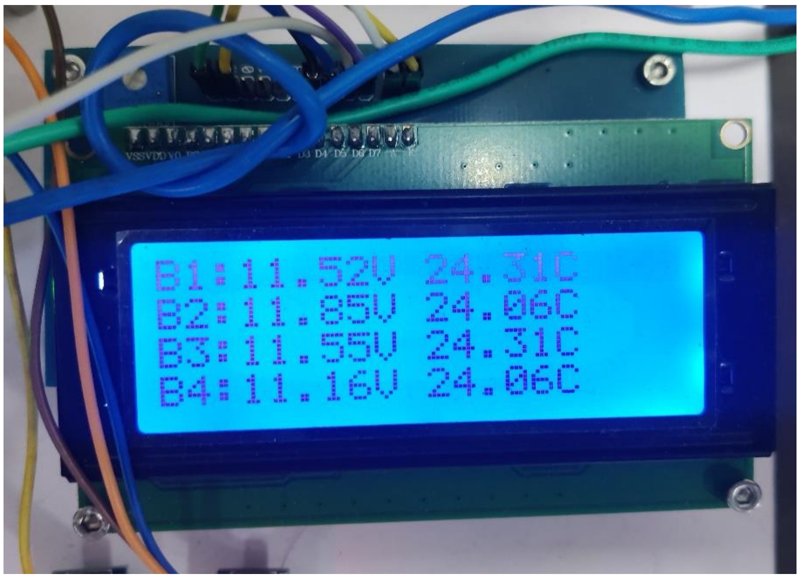
- Collect data in private channels
- Share data with public channels
- RESTful and MQTT APIs
- MATLAB® analytics and visualizations
- Event scheduling
- Alerts
- App integrations

LIQUID CRYSTAL DISPLAY

For local display and visualization, a 16x4 LCD is also added to the system and is interfaced with the NodeMCU board displaying the sensor readings as well as the status messages of the system viz any errors in connecting to WiFi or receiving data from the Arduino. This makes the device more interactive and user friendly



```
Welcome
Connecting to
CHITRANSH-G3 8094
WiFi Connected
```



```
B1:11.52V 24.31C
B2:11.85V 24.06C
B3:11.55V 24.31C
B4:11.16V 24.06C
```

Figure 16: LCD Displaying system status messages and Battery voltage and temperature parameters

RESULTS AND CONCLUSION

The complete setup was assembled and tested at varying load combinations to get clear understanding of the performance parameters of the batteries under test.

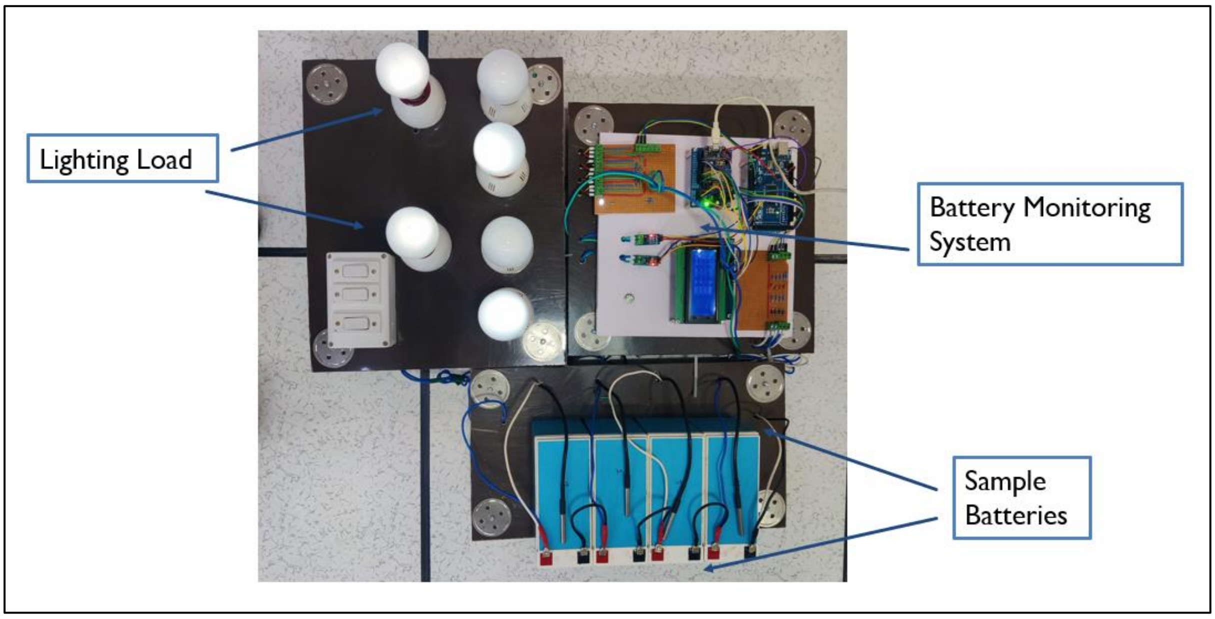


Figure 17: Prototype Developed at MRS

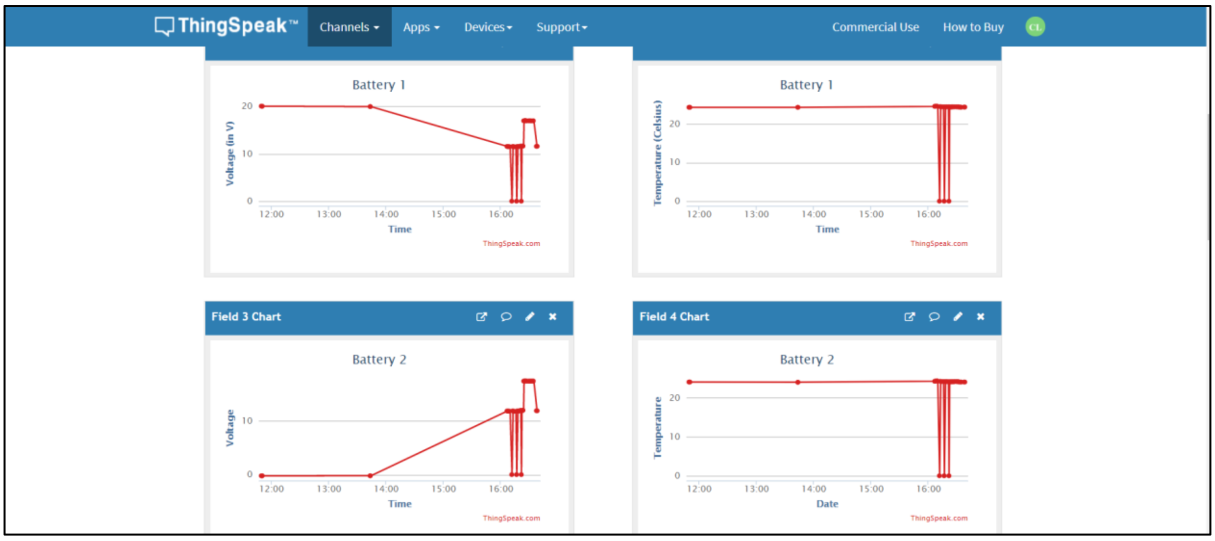


Figure 18: ThingSpeak Dashboard with graphical visualization

A lighting load consisting of DC lamps of 3W,12V rating is used. Four lamps are connected in series and 2 lamps in parallel to simulate load variation conditions for the battery. Three switches are available as shown in Figure 12 , Switch 1 is the main load switch, Switch 2 turns on one lamp and Switch 3 turns on one more additional lamp.

The battery parameters are observed for 5 hours under varying load conditions. The results obtained from ThingSpeak server are tabulated and plotted below for reference.

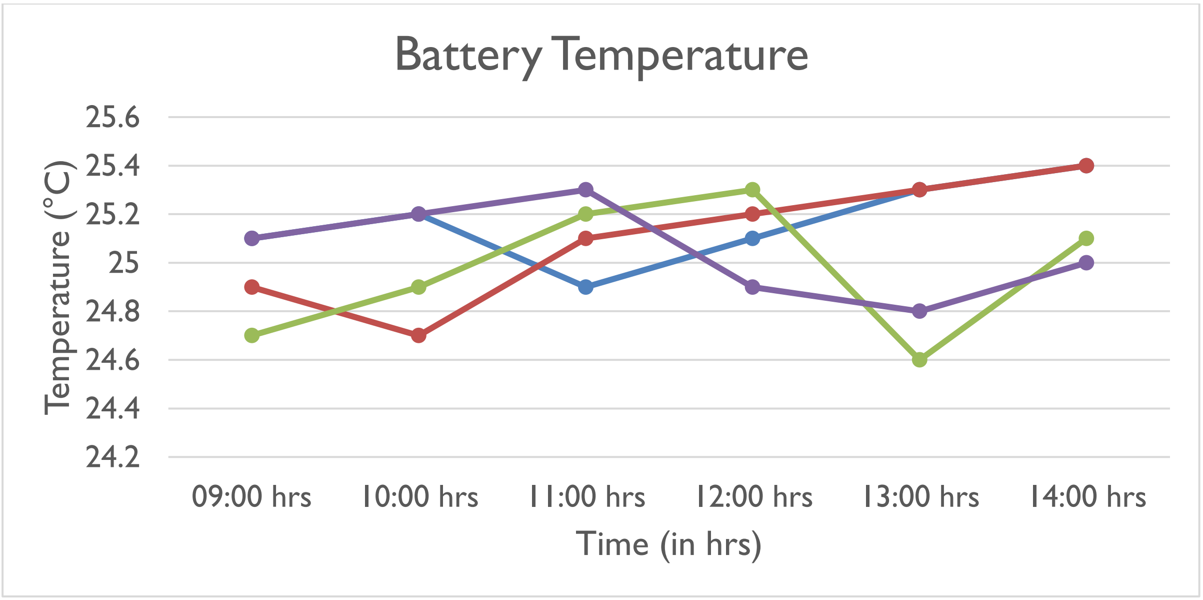


Figure 19: Battery Temperature v/s Time for the cells downloaded from ThingSpeak server

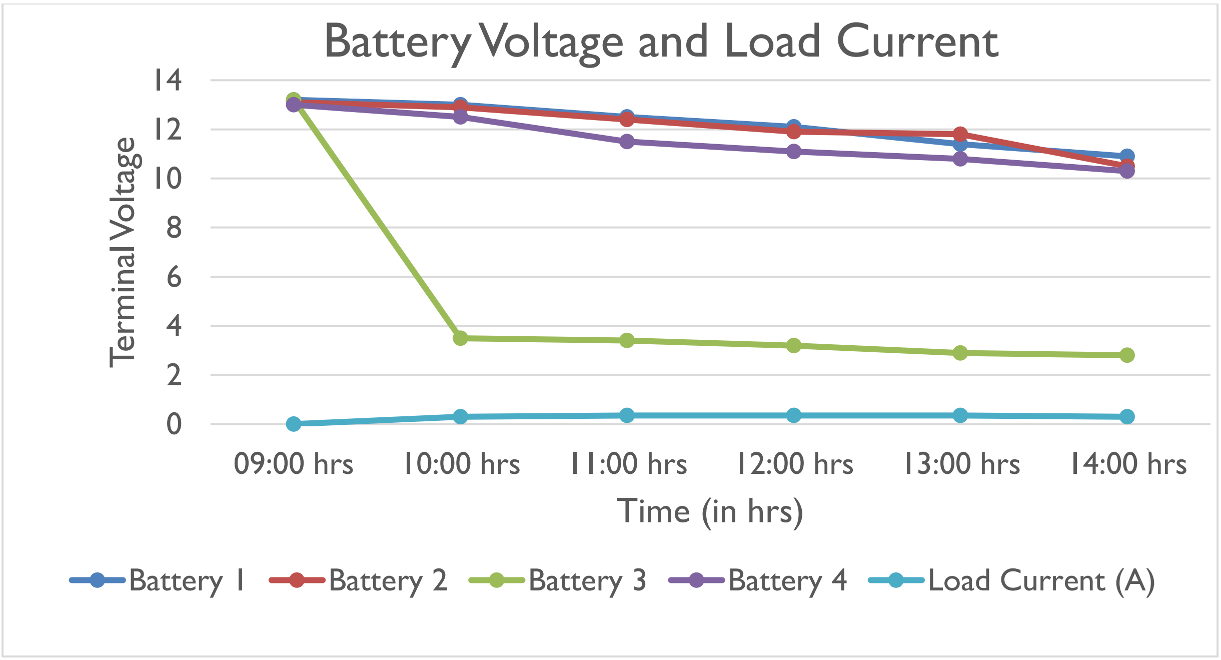


Figure 20: Battery Voltage and Load current v/s Time for the cells downloaded from ThingSpeak server

	Battery 1	Battery 2	Battery 3	Battery 4	Load Current
09:00 hrs	13.2V, 25.1 °C	13.1V, 24.9 °C	13.2V, 24.7 °C	13.0V, 25.1 °C	0 A
10:00 hrs	13.0V, 25.2 °C	12.9V, 24.7 °C	3.5V , 24.9 °C	12.5V, 25.2 °C	0.3 A
11:00 hrs	12.5V, 24.9 °C	12.4V, 25.1 °C	3.4V , 25.2 °C	11.5V, 25.3 °C	0.35 A
12:00 hrs	12.1V, 25.1 °C	11.9V, 25.2 °C	3.2V , 25.3 °C	11.1V, 24.9 °C	0.35 A
13:00 hrs	11.4V, 25.3 °C	11.8V, 25.3 °C	2.9V , 24.6 °C	10.8V, 24.8 °C	0.35 A
14:00 hrs	10.9V, 25.4 °C	10.5V, 25.4 °C	2.8V , 25.1 °C	10.3V, 25.0 °C	0.3 A

Table 7: Sensor health data downloaded from ThingSpeak web server

Some of the benefits envisaged:

- Financial benefits in terms of maximizing battery shelf life and avoiding unplanned shutdown of pipeline operation due to battery backup failure.
- Enhanced safety of the pipeline
- Measures battery parameters such as individual cell voltage, string voltage, battery current, cell terminal / ambient (room) temperature.
- Transmits data to a centralized control room/ remote location through Ethernet/RS485 or using IOT (Cloud).
- Protected against accidental mis-wiring and over voltage on all channel inputs
- The system can be further modified to provides Battery Ah, State of Charge, Residual life of the battery, time duration a battery can discharge safely by measuring, calculating and predicting battery parameters
- Provides warnings of impending malfunction based on set limits of voltage, current and temperature
- This system can also be used for Battery Load test to get accurate battery discharge characteristic curves and better analyse health of the battery.

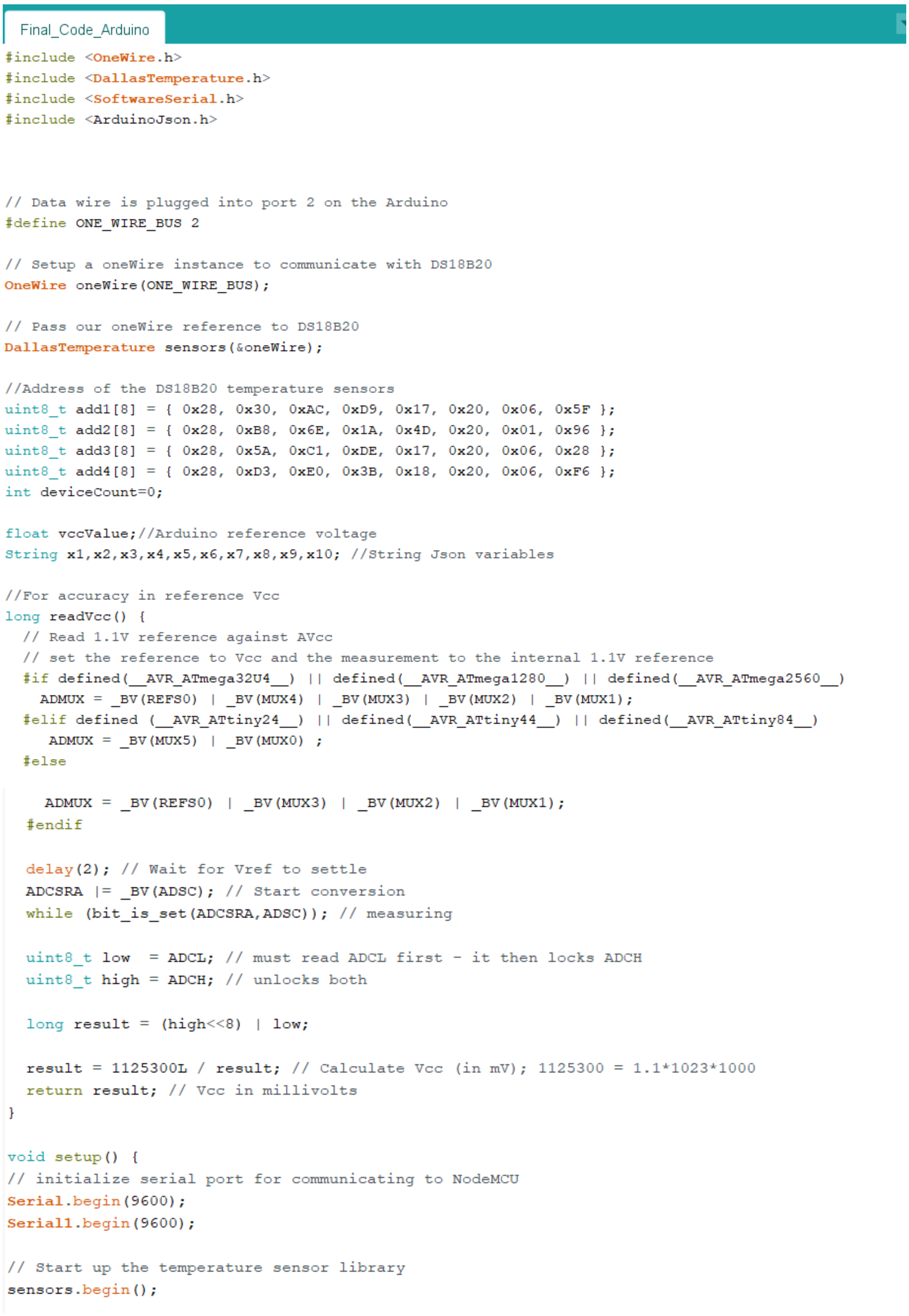
Thus, from a remote battery monitoring system in place we can effectively monitor the health parameters from our control stations and plan effective action in advance and extend the life of the battery banks.

REFERENCES

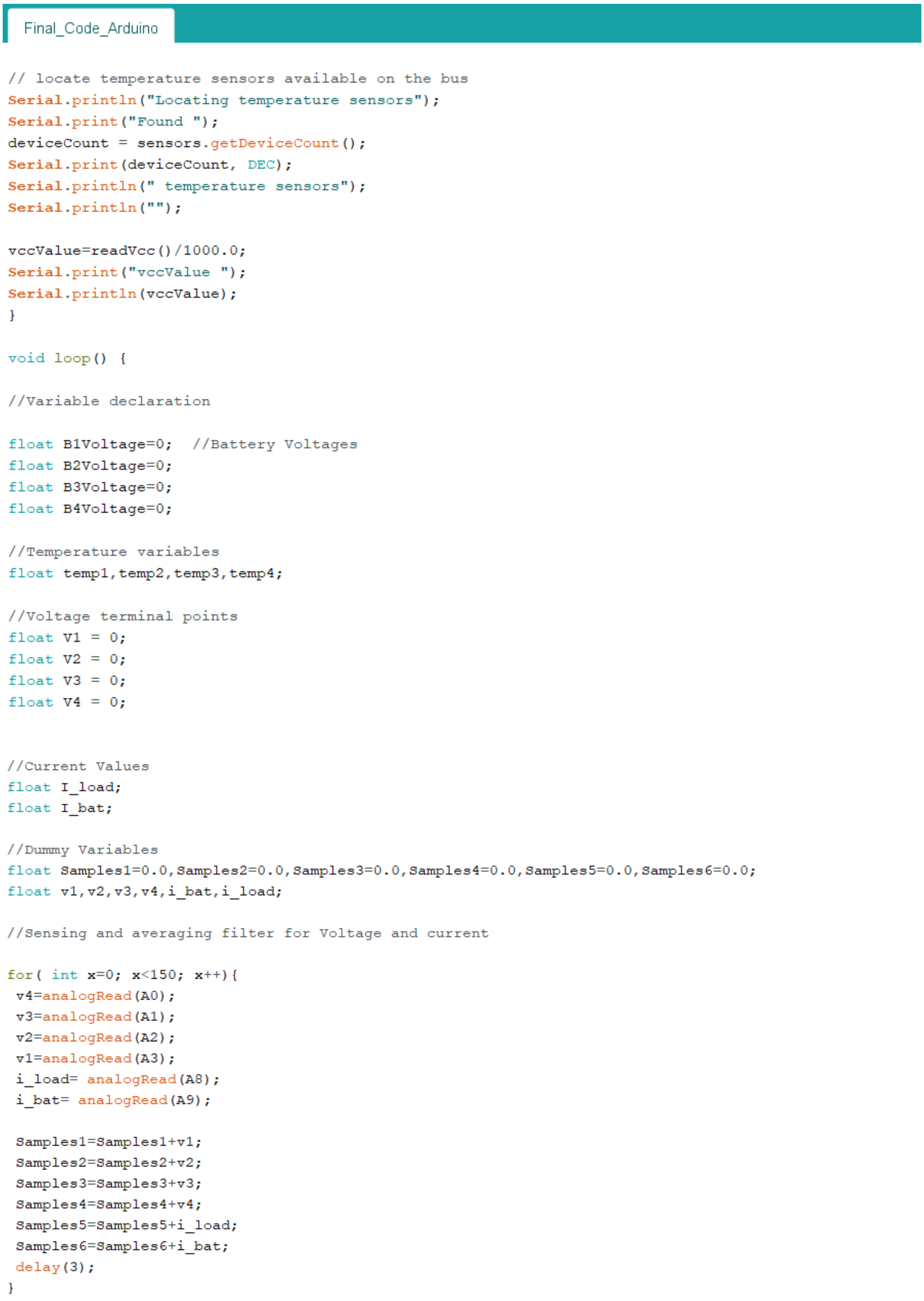
1. The Effect of temperature on VRLA reaction rates and the determination of Battery State of Charge Part 2- Fundamental Considerations- F.J. Vaccaro, J. Rhoades and B. Le (Power Battery Company, Inc.25 McLean Blvd, Paterson, NJ 07514-1507 U.S.A.
2. W. Kevin Siagian, D. Fernando Purba, A. Sipahutar and I. Hartarto Tambunan, "Design and Implementation of Battery Management System for On-Grid System," 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2018, pp. 179-183, doi: 10.1109/ICCEREC.2018.8712088.
3. S. M. Salamati, C. S. Huang, B. Balagopal and M. Chow, "Experimental battery monitoring system design for electric vehicle applications," 2018 IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES), 2018, pp. 38-43, doi: 10.1109/IESES.2018.8349847.
4. Kyung-Sung Lee, Chae-Joo Moon, Tae-Gon Kim, Moon-Seon Jeong, Sang-Man Kim and Byeong-Ju Park, "A development of battery monitoring and management system," 2012 IEEE Vehicle Power and Propulsion Conference, 2012, pp. 428-430, doi: 10.1109/VPPC.2012.6422579.
5. E. A. Dahl and J. Decorpro, "Automatic Battery Monitoring System," INTELEC '85 - Seventh International Telecommunications Energy Conference, 1985, pp. 615-624.
6. W. Kevin Siagian, D. Fernando Purba, A. Sipahutar and I. Hartarto Tambunan, "Design and Implementation of Battery Management System for On-Grid System," 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2018, pp. 179-183, doi: 10.1109/ICCEREC.2018.8712088.
7. S. A. Mathew, R. Prakash and P. C. John, "A smart wireless battery monitoring system for Electric Vehicles," 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), 2012, pp. 189-193, doi: 10.1109/ISDA.2012.6416535.
8. The Truth About Batteries- POWERTHRU White Paper
9. Battery Management System Using Arduino – N Scharich, B Schniter, A Herbert and Md. S Islam, Saginaw Valley State Univ. Saginav, MI 48710
10. Design and Specifications for Safe and Reliable Battery Systems for Large UPS – White Paper 2017, Schneider Electric
11. https://thingspeak.com/pages/learn_more
12. https://store.arduino.cc/products/arduino-mega-2560-rev3
13. https://www.nodemcu.com/index_en.html
14. https://create.arduino.cc/projecthub/SurtrTech/measure-any-ac-current-with-acs712-70aa85
15. https://lastminuteengineers.com/ds18b20-arduino-tutorial/
16. https://www.hbl.in/product-view-7-power-electronics-battery-monitoring-system.html

APPENDIX

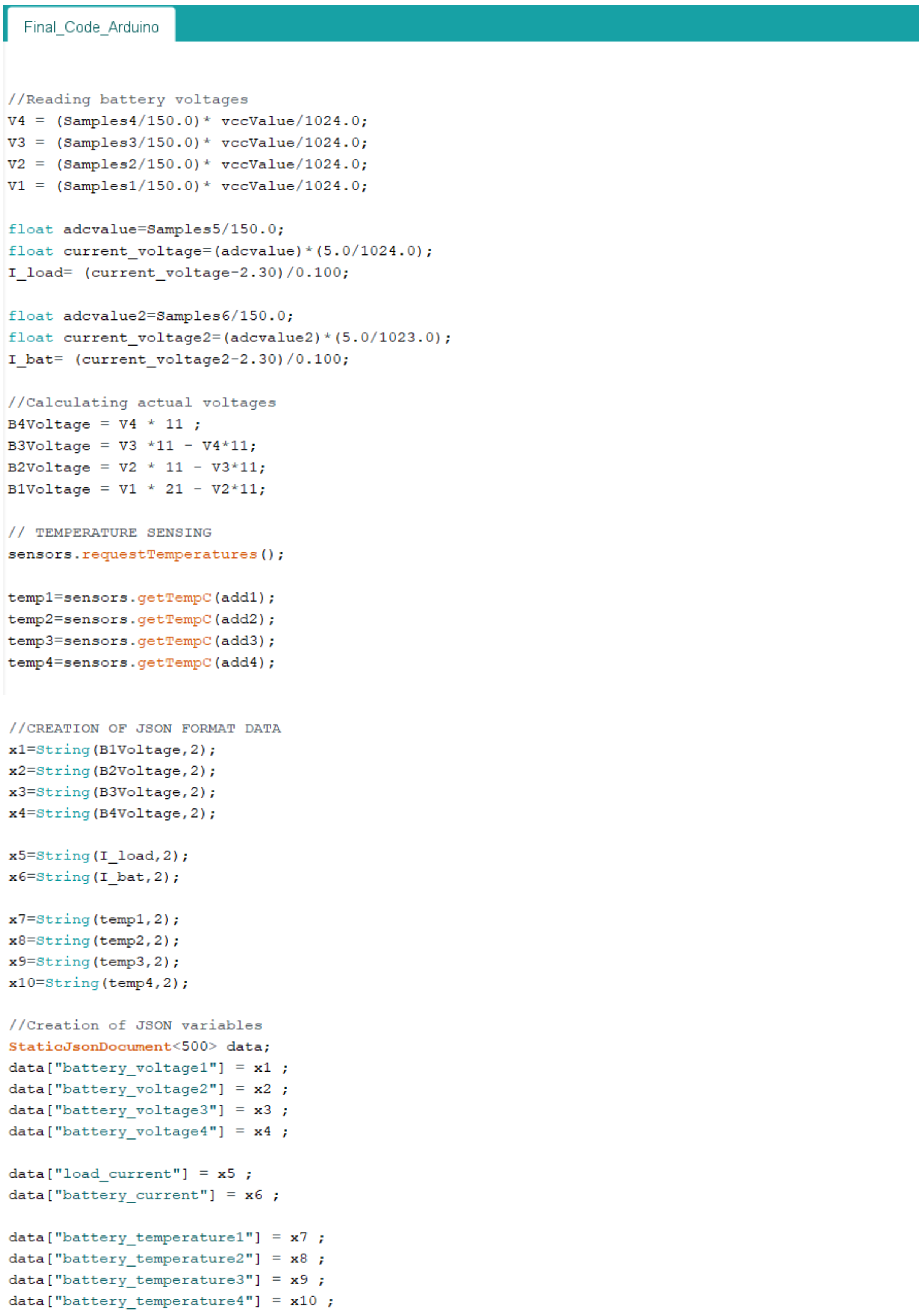
1. Arduino Mega Code



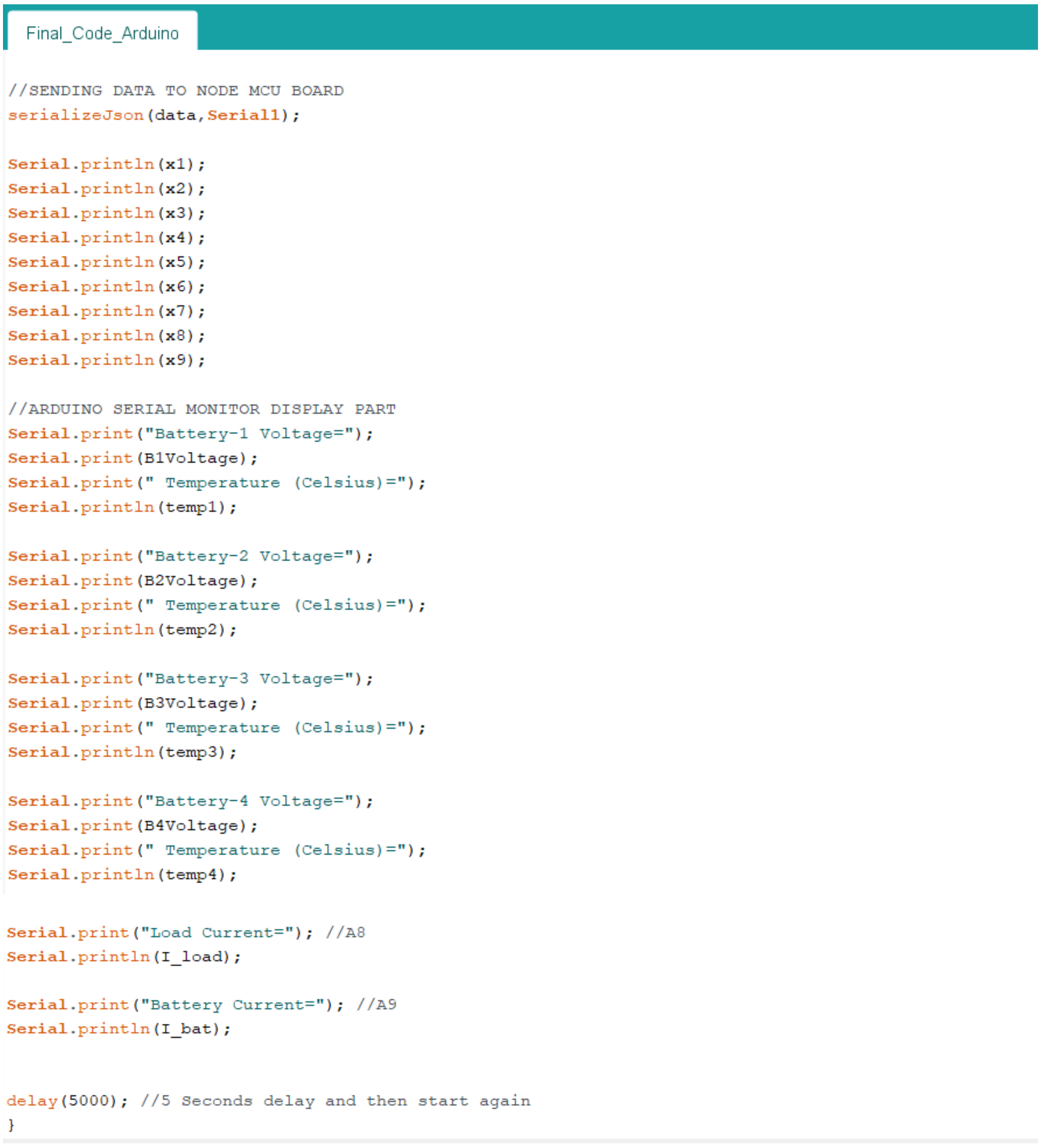
```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>



// Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with DS18B20
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to DS18B20
DallasTemperature sensors(&oneWire);

//Address of the DS18B20 temperature sensors
uint8_t add1[8] = { 0x28, 0x30, 0xAC, 0xD9, 0x17, 0x20, 0x06, 0x5F };
uint8_t add2[8] = { 0x28, 0xB8, 0x6E, 0x1A, 0x4D, 0x20, 0x01, 0x96 };
uint8_t add3[8] = { 0x28, 0x5A, 0xC1, 0xDE, 0x17, 0x20, 0x06, 0x28 };
uint8_t add4[8] = { 0x28, 0xD3, 0xE0, 0x3B, 0x18, 0x20, 0x06, 0xF6 };
int deviceCount=0;

float vccValue;//Arduino reference voltage
String x1,x2,x3,x4,x5,x6,x7,x8,x9,x10; //String Json variables

//For accuracy in reference Vcc
long readVcc() {
  // Read 1.1V reference against AVcc
  // set the reference to Vcc and the measurement to the internal 1.1V reference
  #if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
    ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
     ADMUX = _BV(MUX5) | _BV(MUX0) ;
  #else

    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #endif

  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA,ADSC)); // measuring

  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both

  long result = (high<<8) | low;

  result = 1125300L / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
  return result; // Vcc in millivolts
}

void setup() {
// initialize serial port for communicating to NodeMCU
Serial.begin(9600);
Serial1.begin(9600);

// Start up the temperature sensor library
sensors.begin();
```


Final_Code_Arduino

```
// locate temperature sensors available on the bus
Serial.println("Locating temperature sensors");
Serial.print("Found ");
deviceCount = sensors.getDeviceCount();
Serial.print(deviceCount, DEC);
Serial.println(" temperature sensors");
Serial.println("");

vccValue=readVcc()/1000.0;
Serial.print("vccValue ");
Serial.println(vccValue);
}

void loop() {

//Variable declaration

float B1Voltage=0;  //Battery Voltages
float B2Voltage=0;
float B3Voltage=0;
float B4Voltage=0;

//Temperature variables
float temp1,temp2,temp3,temp4;

//Voltage terminal points
float V1 = 0;
float V2 = 0;
float V3 = 0;
float V4 = 0;


//Current Values
float I_load;
float I_bat;

//Dummy Variables
float Samples1=0.0,Samples2=0.0,Samples3=0.0,Samples4=0.0,Samples5=0.0,Samples6=0.0;
float v1,v2,v3,v4,i_bat,i_load;

//Sensing and averaging filter for Voltage and current

for( int x=0; x<150; x++){
 v4=analogRead(A0);
 v3=analogRead(A1);
 v2=analogRead(A2);
 v1=analogRead(A3);
 i_load= analogRead(A8);
 i_bat= analogRead(A9);

 Samples1=Samples1+v1;
 Samples2=Samples2+v2;
 Samples3=Samples3+v3;
 Samples4=Samples4+v4;
 Samples5=Samples5+i_load;
 Samples6=Samples6+i_bat;
 delay(3);
}
```

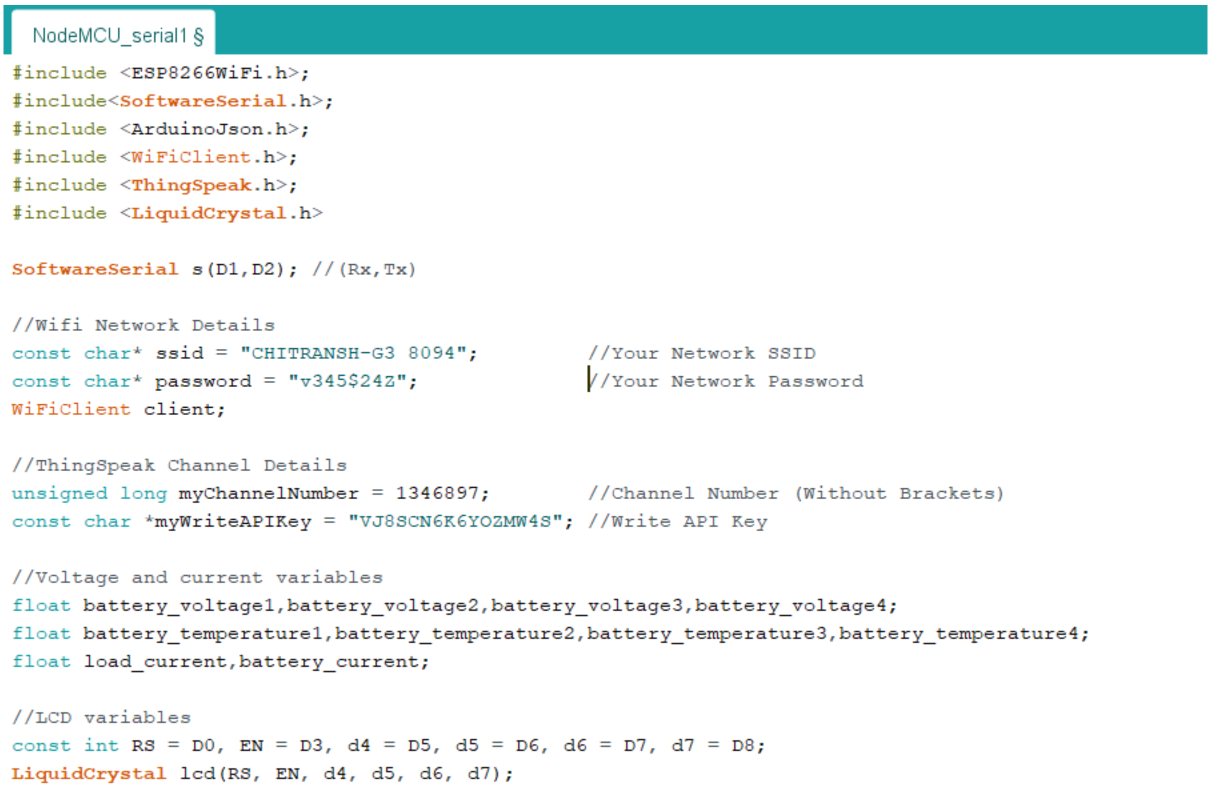


```
//Reading battery voltages
V4 = (Samples4/150.0)* vccValue/1024.0;
V3 = (Samples3/150.0)* vccValue/1024.0;
V2 = (Samples2/150.0)* vccValue/1024.0;
V1 = (Samples1/150.0)* vccValue/1024.0;

float adcvalue=Samples5/150.0;
float current_voltage=(adcvalue)*(5.0/1024.0);
I_load= (current_voltage-2.30)/0.100;

float adcvalue2=Samples6/150.0;
float current_voltage2=(adcvalue2)*(5.0/1023.0);
I_bat= (current_voltage2-2.30)/0.100;

//Calculating actual voltages
B4Voltage = V4 * 11 ;
B3Voltage = V3 *11 - V4*11;
B2Voltage = V2 * 11 - V3*11;
B1Voltage = V1 * 21 - V2*11;

// TEMPERATURE SENSING
sensors.requestTemperatures();

temp1=sensors.getTempC(add1);
temp2=sensors.getTempC(add2);
temp3=sensors.getTempC(add3);
temp4=sensors.getTempC(add4);


//CREATION OF JSON FORMAT DATA
x1=String(B1Voltage,2);
x2=String(B2Voltage,2);
x3=String(B3Voltage,2);
x4=String(B4Voltage,2);

x5=String(I_load,2);
x6=String(I_bat,2);

x7=String(temp1,2);
x8=String(temp2,2);
x9=String(temp3,2);
x10=String(temp4,2);

//Creation of JSON variables
StaticJsonDocument<500> data;
data["battery_voltage1"] = x1 ;
data["battery_voltage2"] = x2 ;
data["battery_voltage3"] = x3 ;
data["battery_voltage4"] = x4 ;

data["load_current"] = x5 ;
data["battery_current"] = x6 ;

data["battery_temperature1"] = x7 ;
data["battery_temperature2"] = x8 ;
data["battery_temperature3"] = x9 ;
data["battery_temperature4"] = x10 ;
```

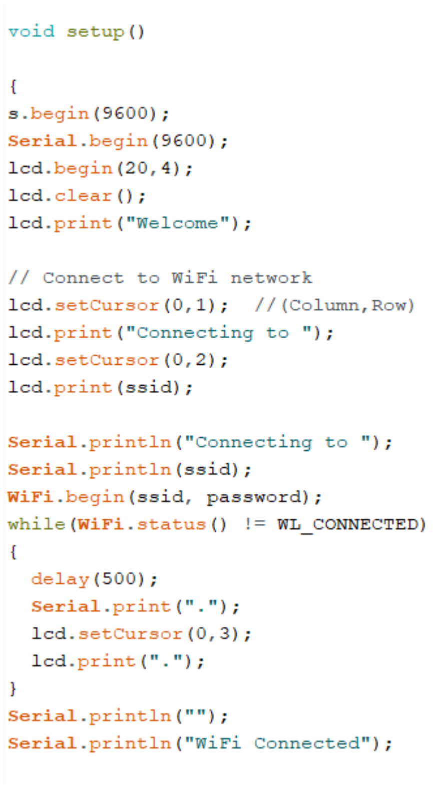


```
//SENDING DATA TO NODE MCU BOARD
serializeJson(data,Serial1);

Serial.println(x1);
Serial.println(x2);
Serial.println(x3);
Serial.println(x4);
Serial.println(x5);
Serial.println(x6);
Serial.println(x7);
Serial.println(x8);
Serial.println(x9);

//ARDUINO SERIAL MONITOR DISPLAY PART
Serial.print("Battery-1 Voltage=");
Serial.print(B1Voltage);
Serial.print(" Temperature (Celsius)=");
Serial.println(temp1);

Serial.print("Battery-2 Voltage=");
Serial.print(B2Voltage);
Serial.print(" Temperature (Celsius)=");
Serial.println(temp2);

Serial.print("Battery-3 Voltage=");
Serial.print(B3Voltage);
Serial.print(" Temperature (Celsius)=");
Serial.println(temp3);

Serial.print("Battery-4 Voltage=");
Serial.print(B4Voltage);
Serial.print(" Temperature (Celsius)=");
Serial.println(temp4);


Serial.print("Load Current="); //A8
Serial.println(I_load);

Serial.print("Battery Current="); //A9
Serial.println(I_bat);


delay(5000); //5 Seconds delay and then start again
}
```

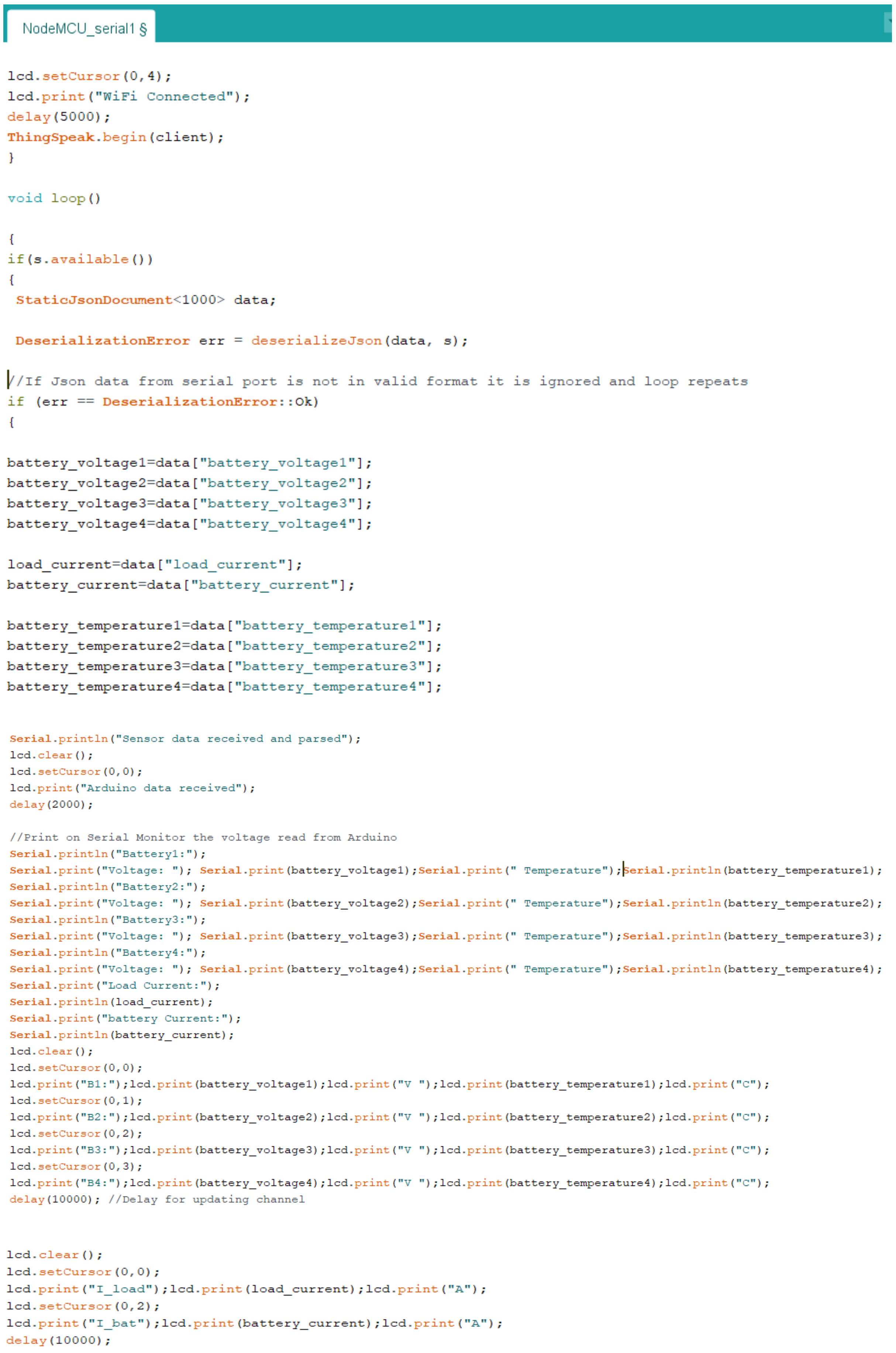
1. NodeMCU Code



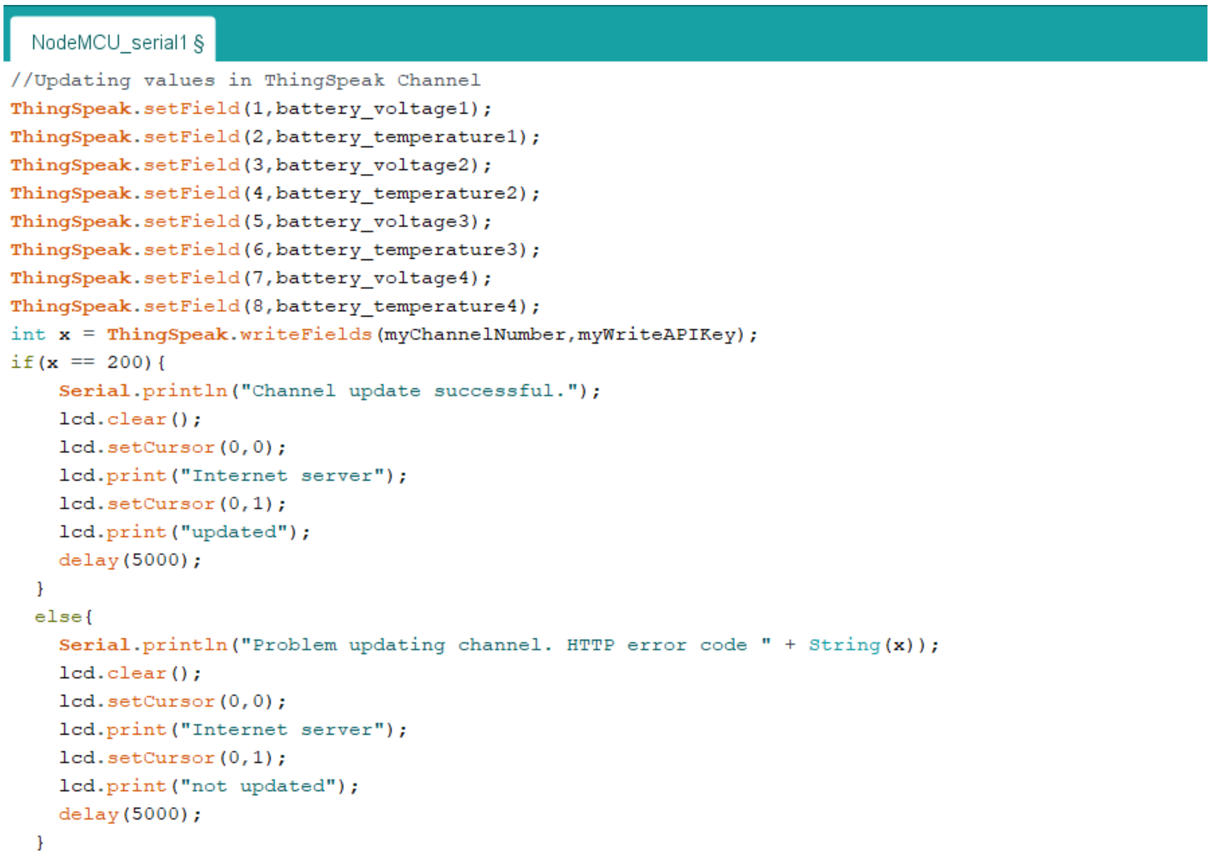
```
#include <ESP8266WiFi.h>;
#include<SoftwareSerial.h>;
#include <ArduinoJson.h>;
#include <WiFiClient.h>;
#include <ThingSpeak.h>;
#include <LiquidCrystal.h>

SoftwareSerial s(D1,D2); //(Rx,Tx)

//Wifi Network Details
const char* ssid = "CHITRANSH-G3 8094";         //Your Network SSID
const char* password = "v345$24Z";              //Your Network Password
WiFiClient client;

//ThingSpeak Channel Details
unsigned long myChannelNumber = 1346897;        //Channel Number (Without Brackets)
const char *myWriteAPIKey = "VJ8SCN6K6YOZMW4S"; //Write API Key

//Voltage and current variables
float battery_voltage1,battery_voltage2,battery_voltage3,battery_voltage4;
float battery_temperature1,battery_temperature2,battery_temperature3,battery_temperature4;
float load_current,battery_current;

//LCD variables
const int RS = D0, EN = D3, d4 = D5, d5 = D6, d6 = D7, d7 = D8;
LiquidCrystal lcd(RS, EN, d4, d5, d6, d7);
```



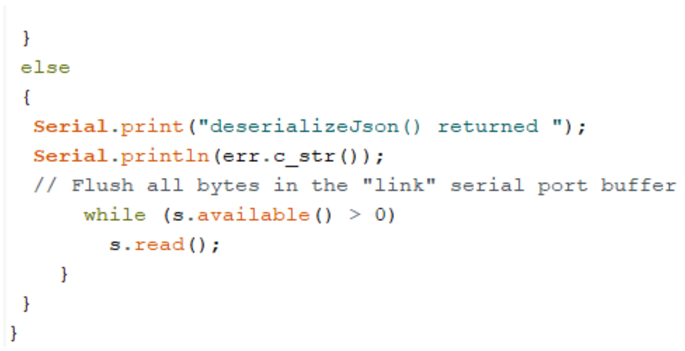
```
void setup()

{
s.begin(9600);
Serial.begin(9600);
lcd.begin(20,4);
lcd.clear();
lcd.print("Welcome");

// Connect to WiFi network
lcd.setCursor(0,1);  //(Column,Row)
lcd.print("Connecting to ");
lcd.setCursor(0,2);
lcd.print(ssid);

Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while(WiFi.status() != WL_CONNECTED)
{
  delay(500);
  Serial.print(".");
  lcd.setCursor(0,3);
  lcd.print(".");
}
Serial.println("");
Serial.println("WiFi Connected");
```



```
lcd.setCursor(0,4);
lcd.print("WiFi Connected");
delay(5000);
ThingSpeak.begin(client);
}

void loop()

{
if(s.available())
{
 StaticJsonDocument<1000> data;

 DeserializationError err = deserializeJson(data, s);

//If Json data from serial port is not in valid format it is ignored and loop repeats
if (err == DeserializationError::Ok)
{

battery_voltage1=data["battery_voltage1"];
battery_voltage2=data["battery_voltage2"];
battery_voltage3=data["battery_voltage3"];
battery_voltage4=data["battery_voltage4"];

load_current=data["load_current"];
battery_current=data["battery_current"];

battery_temperature1=data["battery_temperature1"];
battery_temperature2=data["battery_temperature2"];
battery_temperature3=data["battery_temperature3"];
battery_temperature4=data["battery_temperature4"];


Serial.println("Sensor data received and parsed");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Arduino data received");
delay(2000);

//Print on Serial Monitor the voltage read from Arduino
Serial.println("Battery1:");
Serial.print("Voltage: "); Serial.print(battery_voltage1);Serial.print(" Temperature");Serial.println(battery_temperature1);
Serial.println("Battery2:");
Serial.print("Voltage: "); Serial.print(battery_voltage2);Serial.print(" Temperature");Serial.println(battery_temperature2);
Serial.println("Battery3:");
Serial.print("Voltage: "); Serial.print(battery_voltage3);Serial.print(" Temperature");Serial.println(battery_temperature3);
Serial.println("Battery4:");
Serial.print("Voltage: "); Serial.print(battery_voltage4);Serial.print(" Temperature");Serial.println(battery_temperature4);
Serial.print("Load Current:");
Serial.println(load_current);
Serial.print("battery Current:");
Serial.println(battery_current);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("B1:");lcd.print(battery_voltage1);lcd.print("V ");lcd.print(battery_temperature1);lcd.print("C");
lcd.setCursor(0,1);
lcd.print("B2:");lcd.print(battery_voltage2);lcd.print("V ");lcd.print(battery_temperature2);lcd.print("C");
lcd.setCursor(0,2);
lcd.print("B3:");lcd.print(battery_voltage3);lcd.print("V ");lcd.print(battery_temperature3);lcd.print("C");
lcd.setCursor(0,3);
lcd.print("B4:");lcd.print(battery_voltage4);lcd.print("V ");lcd.print(battery_temperature4);lcd.print("C");
delay(10000); //Delay for updating channel


lcd.clear();
lcd.setCursor(0,0);
lcd.print("I_load");lcd.print(load_current);lcd.print("A");
lcd.setCursor(0,2);
lcd.print("I_bat");lcd.print(battery_current);lcd.print("A");
delay(10000);
```



```
//Updating values in ThingSpeak Channel
ThingSpeak.setField(1,battery_voltage1);
ThingSpeak.setField(2,battery_temperature1);
ThingSpeak.setField(3,battery_voltage2);
ThingSpeak.setField(4,battery_temperature2);
ThingSpeak.setField(5,battery_voltage3);
ThingSpeak.setField(6,battery_temperature3);
ThingSpeak.setField(7,battery_voltage4);
ThingSpeak.setField(8,battery_temperature4);
int x = ThingSpeak.writeFields(myChannelNumber,myWriteAPIKey);
if(x == 200){
    Serial.println("Channel update successful.");
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Internet server");
    lcd.setCursor(0,1);
    lcd.print("updated");
    delay(5000);
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Internet server");
    lcd.setCursor(0,1);
    lcd.print("not updated");
    delay(5000);
  }
```



```
 }
 else
 {
  Serial.print("deserializeJson() returned ");
  Serial.println(err.c_str());
  // Flush all bytes in the "link" serial port buffer
      while (s.available() > 0)
        s.read();
    }
 }
}
```