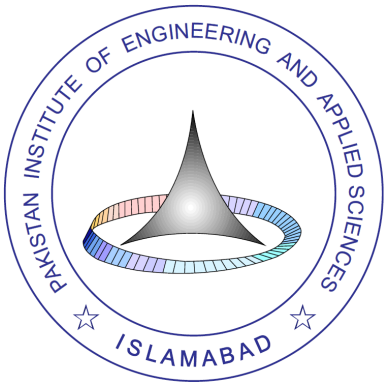

EE-311 Microprocessors Interfacing

Complex Engineering Problem

Title: Condition Monitoring System of a motor over IoT

Group No: 18

Section: A

Date Of Submission: 22-05-2024

Submitted by

Muhammad Ali Athar

Muhammad Fahad

Department of Electrical Engineering
Pakistan Institute of Engineering and Applied Sciences

Abstract

The purpose of this project is to develop a condition monitoring system for motor, using IoT to provide real-time data. The system bases on an STM32F103C8T6 microcontroller, an ACS712 current sensor, an LM335 temperature sensor, an ADXL345 accelerometer, and an ESP8266 Wi-Fi module. The main goal is to monitor motor parameters such as vibration, temperature, current, and sound levels, to make sure the motor doesn't fail unexpectedly. The code is based on FREERTOS middleware, which uses a memory management scheme to make it look like the tasks are running in parallel, but actually it is utilizing osDelay functions to make sure the delay time doesn't go to waste and the controller does some other task in meantime. More than 1500 samples are collected every 20 seconds, and sent to the internet using the esp-01 module, to "thingspeak.com", by MATHWORKS, which has capability to receive data and plot it to show user-friendly graphs, where the motor can be analyzed for faults.

Contents

Chapter 1

Introduction

In real-life industrial environment, keeping motors running smoothly is very important. If a motor breaks down unexpectedly, it can cause delays and reduce efficiency. Manually checking the motor is a discrete process, and cant be done continuously, and even if done regularly, it can miss early signs of problems. To solve this issue, we developed a condition monitoring system using Internet of Things (IoT) technology. This system provides real-time monitoring and helps in predicting maintenance needs. This project aims to improve motor maintenance by providing a detailed, real-time view of the motor's condition. By utilizing a device to do the work over internet, is an efficient method and can be preferred over manual inspection.

1.1 STM32 Microcontroller

An Arm cortex M3 microcontroller, with rich set of peripherals consisting those required for the project including USART, I2C, SPI, and ADCs. Moreover, running on 72 MHz, it provides decent enough speeds, to sample data. The device has 20KB SRAM which can handle bulk data extraction from small sensors.

1.2 ADXL345 Accelerometer

An accurate accelerometer which provide three axis acceleration data,in 12 bit accuracy, in the units of gravitational constants. It can provide accuracy down to 2g and up to 16g. the acceleration data, cn then be procesed in to vibration data. It can be used both with I2C and SPI mode.

1.3 Analog Sensors

A simple microphone sensor is used here. KY-038, provides an analog output. Biased at half the Vcc and providing somewhat usable sound output. We use the data to simply detect any knocking

sounds in the motor. An ACS-712 hal effect current sensor measures current. LM335 Temperature sensor is used to sense motor temperature.

1.4 Wifi Module

ESP-01 Wifi module is used. We send AT commands in string format through Uart at 115200 bps. It simply does all the work after that. The Commands to send the data are also sent in string format, therefore rendering it simple and easy-to-use.

Chapter 2

Equipments and Procedure

2.1 Equipments

Following equipmenst have been used for designing the circuit:

- STM32F103C8T6 Arm Microcontroller
- ACS-712 5B current sensor
- KY-038 Microphone module
- ADXL345 Accelerometer
- LM335 Temperature Sensor
- ESP-01 Wifi Module
- STLink-V2

2.2 Procedure

The implementation of the project was done in hardware and software in the following manner:

2.2.1 Hardware

1. Connect the ESP-01 to the USART1 TX pin, which is A9 Pin.
2. Connect ADXL345 SDA, SDO and SCK to SP1 pins of controller, which are A5,A6 and A7.
3. Connect the analog Senors to the pins A0-A2.

2.2.2 Software

1. Enable Clock of the needed peripherals. configure the pins for the usage of Alternate functions, or analog Input.
2. Send the necessary AT commands to ESP-01 to start the working on IoT.
3. Start Taking out the sensor Data, and perform the necessary calculations, using the formulae provided in datasheets of the respective sensors.
4. Send the gathered and processed data to the IoT using the string AT commands in ESP-01.

Chapter 3

Literature Survey

In literature survey, we studied various aspects of what factors to include in the design of a motor's condition monitoring. We came across a company, that designed a similar product on a commercial scale, ABB motors and generators. They consider many such properties that cant be measured by the instruments that are available to us. Although, we chose 4 viable conditions that were in our level of comprehension. Overall Vibration, skin temperature, Current draw and knocking sounds, are the readings we start with. For the consultation of calibration of the sensors and calculation formulae, we consulted the respective datasheets. For the programming aid, we used '• The STM32F103 Arm Microcontroller and Embedded Systems' by the authors 'Sepehr Naimi, Sarmad Naimi, Muhammad Ali Mazidi'. The book provides the concept of bit-level working of each peripheral.

Chapter 4

Results And Discuccions

4.1 Block Diagram

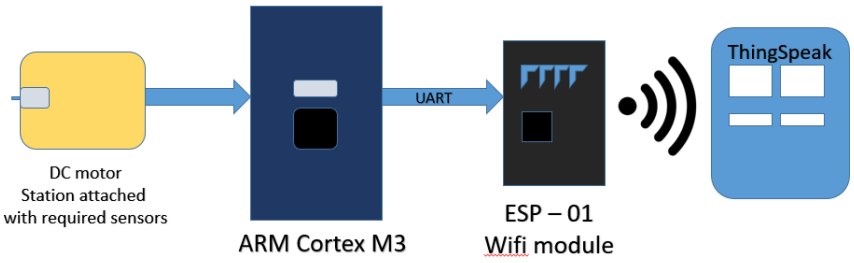

Figure 4.1: Block Diagram

4.2 Results

4.2.1 Hardware Implementation

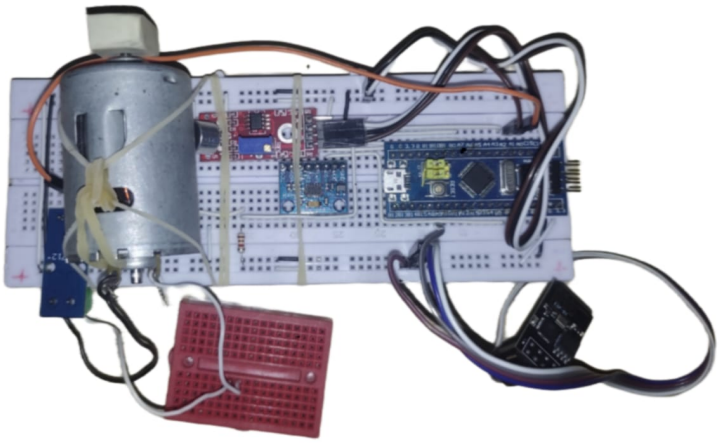

Figure 4.2: Top View of Circuit

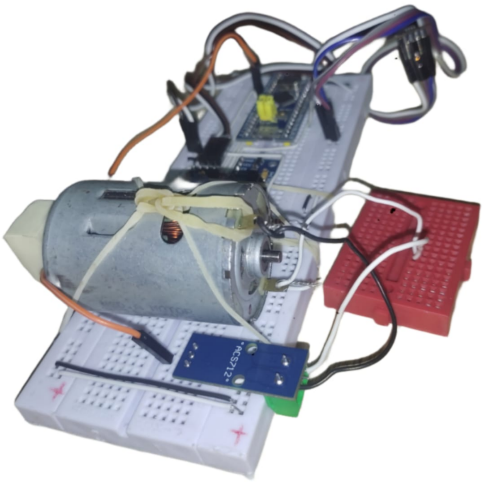

Figure 4.3: Side View

4.2.2 Output on IoT

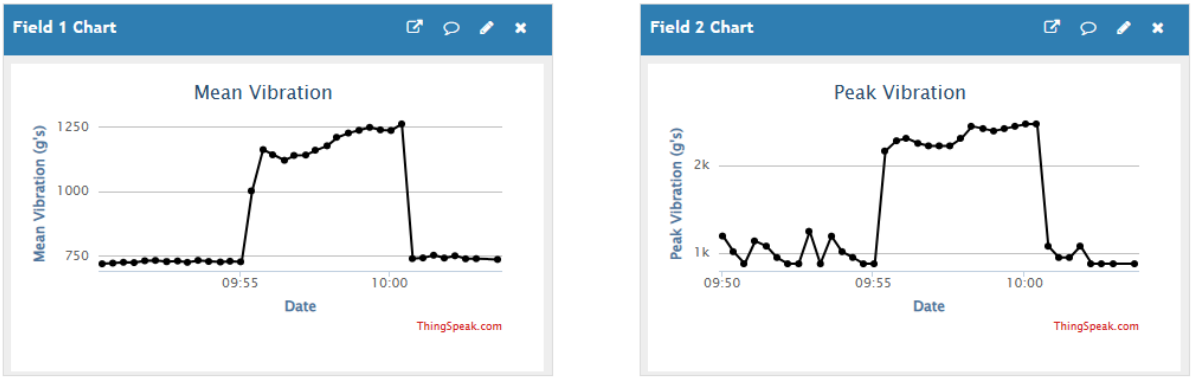

Figure 4.4: Mean and Peak Vibration

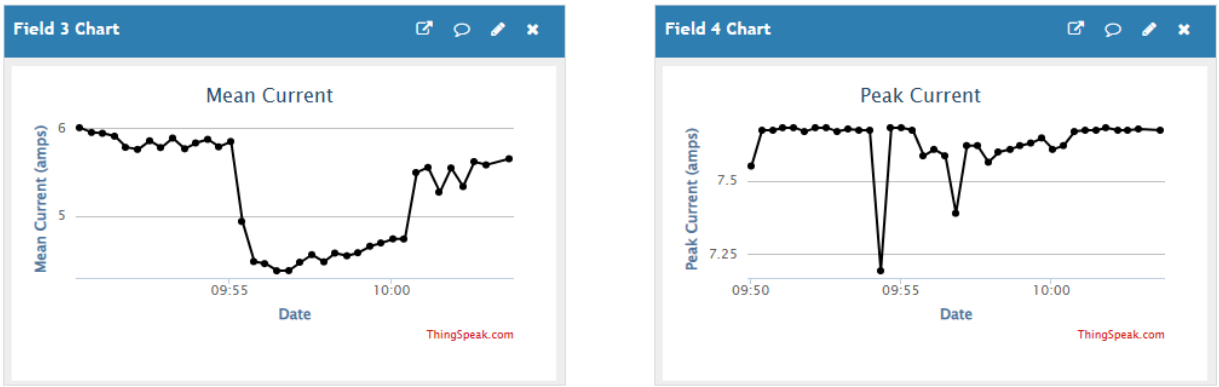

Figure 4.5: Current in the motor

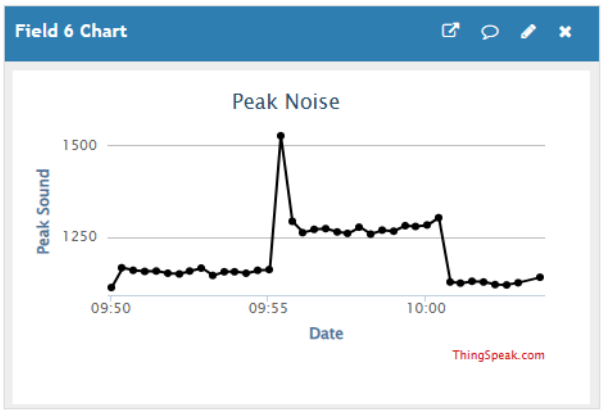

Figure 4.6: Noise of the Motor

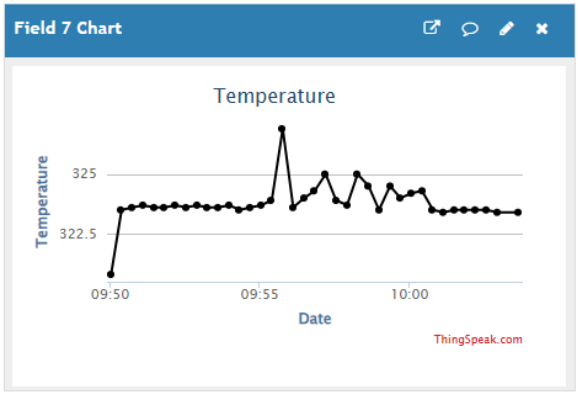

Figure 4.7: Skin Temperature of the motor

4.3 Discussion

We used FREERTOS for the realization of our design. It is a memory management scheme which incorporates the systick timer to time the tasks when called. The delays are replaced with osDelays, which blocks the current function for the given time and runs other functions meanwhile. Hence it creates an effect of running multiple functions in parallel. We used two functions, one for sending data to IoT, blocked for 20 seconds and normal OS priority. The other function collects sensor data with 10 milli-seconds block, and low OS priority.

The challenge we took on to ourselves, was not to use any kind of HAL library support except for the RTOS middle-ware. In several proto-types, we wrote code to run the vibration sensor using the I2C, which was left behind due to hardware bugs which we couldnt figure out the reasons for. So we had to use a little support of HAL Library for taking the vibration sensor data out, by incorporating the SPI HAL functions. But the aim was satisfied by writing the rest of the code in pure embedded C.

The failure we encountered was failing to use the one-wire protocol, to use the DS18B20 sensor to measure temperature which is many time more accurate than currently used LM335. But the problem we encountered due to our goal not to use library support. We wrote several prototypes for the sensor to work, but it didn't work out in the end, and we were left with using the LM335.

In sending the data to ESP-01, we incorporated the DMA peripheral support. It just reduced some lines of code. what it does is simply transferring the data from memory to the uart peripheral, in our case, and the plus point being, it doesn't take up the CPU resources. It wasnt that useful in our case though, since we just kept the program stuck in a loop till the whole data was transferred. The most it did was helping us understand the coding of DMA.

The code and related files are sourced here:
rb.gy/vjox66

Chapter 5

Summary And Conclusion

5.1 Summary

A motor condition monitoring system was designed by the help of ARM Cortex M3 Microcontroller, and the data was displayed on the IoT Platform. The program was written in Keil uVision 5. It incorporated FREERTOS for better use of the hardware and CPU resources.

5.2 Conclusion

We successfully designed a motor's condition monitoring system which helped us strengthen our understanding of the application of microprocessors in real-world scenarios. It helped us set a base for embedded system programming, and learn the interfacing of sensors. This CEP helped us efficiently utilize the resources given to us in the form factor or microcontroller. We collectd data from sensors and displayed the results on ThingSpeak.com website, after doing some calculations. The values provided by the system are not super-accurate, so they are not effective to use in industry, but it provides a good insight of the general behavior of each of the quantity. It gives the trends followed by circuit. We see how turning on the motor causes the vibration to raise to a certain level, and maintain there until the motor is turned of. Similar trends were seen in the rest of the quantities too. Observing overall, most of the goals of the project were somewhat met, the project was completed and accurate results were obtained.

References

1. The STM32F103C8 Arm microcontroller and embedded systems, by Sepehr Naimi, Sarmad Naimi, Muhammad Ali Mazidi
2. ESP-8266 user manual, by Espressif
3. Nicerlands.com website, by Sepehr Naimi
4.