

**Inexpensive Method For Data Acquisition  
And Monitoring In Vibrating Structures  
Using Microcontrollers**

*Thesis submitted in partial fulfilment of the  
requirement for the degree of*

**Master of Mechanical Engineering**

*By*

**DIPANKAR DAS**

Department of Mechanical Engineering  
Faculty of Engineering & Technology

Jadavpur University

Kolkata – 700032

May, 2016



**Inexpensive Method For Data Acquisition And Monitoring In  
Vibrating Structures Using Microcontrollers**

*Thesis submitted in partial fulfilment of the requirement for the degree of*

**Master of Mechanical Engineering**

*By*

**DIPANKAR DAS**

Exam Roll No. – M4MEC1607

Registration No. – 129384 of 2014-2015

*Under the Guidance of*

**DR. ARGHYA NANDI**

**DR. SUMANTA NEOGY**

Department of Mechanical Engineering

Faculty of Engineering & Technology

Jadavpur University

Kolkata – 700032

May, 2016

**FACULTY OF ENGINEERING & TECHNOLOGY  
JADAVPUR UNIVERSITY**

**CERTIFICATE OF APPROVAL\***

*This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.*

Committee on final examination

for the evaluation of the thesis:

-----

(Signature of the Examiners)

\* Only in case the thesis is approved

**FACULTY OF ENGINEERING AND TECHNOLOGY  
JADAVPUR UNIVERSITY**

**THESIS CERTIFICATE**

*We hereby recommend that the thesis presented under our supervision by Mr. Dipankar Das (Roll No.- M4MEC1607) entitled "Inexpensive Method For Data Acquisition And Monitoring In Vibrating Structures using Microcontrollers" be accepted in partial fulfillment of the requirements for the degree of Master of Mechanical Engineering.*

**Countersigned**

-----  
Head of the Department  
Mechanical Engineering  
Jadavpur University

-----  
**Dr. ARGHYA NANDI**  
Thesis Advisors

-----  
Dean of Faculty of  
Engineering and Technology  
Jadavpur University

-----  
**Dr. Sumanta Neogy**

# ACKNOWLEDGEMENT

*At the very outset, I acknowledge, with earnest appreciation, the generous and arduous assistance and inspiration provided to me by my honorable guide, **Dr. Arghya Nandi**, whose foresight and experience in the field of micro-controller worked as elixir of encouragement for me and will do so for many to come and without whom my thesis could not have reached this stage, and my advisor, **Dr. Sumanta Neogy** whose support and guidance gave me a significant direction during my research at Jadavpur University. Words of thanks fall short to express my gratitude to them in proper dimensions for their kind co-operation and methodical guidance in course of the thesis work.*

*My sincere thank also goes to the Mr. Pradip Das of Applied Mechaics Department for his tireless and enthusiastic support in providing various laboratory made equipment during my thesis work.*

*I am very thankful to my friends and seniors of Mechanical engineering department, who have helped me by providing continuous menatl encouragement and supported me for working in a field which is not so familiar among the varsities of applied mechanics.*

*Finally, I must acknowledge my parents, brother and friends, without whose compassionate voices and support from behind the scenes, this work would have remained incomplete.*

Place: Jadavpur  
Date: 15/05/2016

(Dipankar Das)  
Roll no. M4MEC1607

# ***LIST OF FIGURES***

<b>Figure</b>	<b>Description</b>	<b>Page No.</b>
1. Fig.1.1	Block diagram of a very simple Data Acquisition System.	2
2. Fig.1.2	Block diagram of a very simple Monitoring System.	5
3. Fig.2.1	COM on device manager	13
4. Fig.2.2	Putty login information	13
5. Fig.2.3	User name and password for edison02 login	14
6. Fig.2.4	Two USB serial ports connected between computer and Edison	14
7. Fig.2.5	Various connections to Intel Edison for performing data acquisition	15
8. Fig.2.6	Connections of Q8_USB	16
9. Fig.2.7	Simulink diagram for data acquisition	17
10. Fig.2.8:	Two scopes for checking generated and input signal to Edison via Q8_USB	18
11. Fig.2.9	Python programme for acquiring data in Edison	18
12. Fig.2.10	Acquired data rendering on putty terminal	19
13. Fig.2.11	Python programme for acquiring and saving data in Edison	20
14. Fig.2.12	Four steps to check and access saved data	21
15. Fig.2.13	Matlab programme 'man1.m.' for imparting equal interval of saved data file	22
16. Fig.2.14	MATLAB programme 'fft_calc.m' for Fast Fourier Transformation	23
17. Fig.2.15	Output signal from saved data file after modifying by MATLAB	24
18. Fig.2.16	FFT of saved data file of Edison	24
19. Fig.2.17	Simulink diagram for combination of frequency.	25
20. Fig.2.18	Output signal of combined frequency from saved data file after modifying by Matlab.	25
21. Fig.2.19	FFT of the combined frequency signal.	25
22. Fig.2.20	Python programme for sending data in another host via PySerial	26
23. Fig.2.21	Python programme at another host for receiving data from Edison	27
24. Fig.2.22	Real time graph using python	27

25. Fig.2.23	Programme at another host for receiving, saving and plotting of data from Edison.	28
26. Fig.2.24	Location where data will be stored in another host.	28
27. Fig.3.1,3.2	Changes on SPP-loopback.py to get SPP-blink.py programme	30-31
28. Fig.3.3	Searching Bluetooth device on phone	31
29. Fig.3.4	Connecting with edison02	31
30. Fig.3.5	Message on putty terminal as soon a Bluetooth connection was established	32
31. Fig.3.6,3.7	Blinking LED by sending command over Bluetooth	32
32. Fig.3.8	Changes on SPP-loopback.py to get SPP-MaxVal_QUARC_4.9V.py programme.	33
33. Fig.3.9,3.10	Results displaying on android phone over Bluetooth	34
34. Fig.3.11	Results displaying on serial terminal of putty.	34
35. Fig.4.1	The input and output response	36
36. Fig.4.2	Simulink model.	37
37. Fig.4.3	Arduino programme which was uploaded in MCP4921 device	37
38. Fig.4.4	The breadboard view of model with jumpers connected to requisite pin.	38
39. Fig.4.5	The schematic view of our model which is shown in Fig.4.4	38
40. Fig.4.6	MCP4921 connected to input and output terminals via two crocodile connector	39
41. Fig.4.7	The DAQ device connected to two terminals	39
42. Fig.4.8	Input Output Signal	40
43. Fig.4.9&4.10,	10Hz&DIV4 and 20Hz&DIV4	41
44. Fig.4.11&4.12,	20Hz&DIV2 and Fig.4.12:40Hz&DIV2	42
45. Fig:5.1	Cantilever beam and support frame.	45
46. Fig:5.2	Actuation system for inducing vibration in cantilever.	45
47. Fig:5.3	Permanent South Pole magnet	46
48. Fig:5.4	Hall sensor used in experiment	46
49. Fig:5.5	Amplitude voltage of beam	46
50. Fig:5.6	Side way Hall Displacemrnt sensor	47



51. Fig:5.7	Working principle of Hall Sensor	47
52. Fig:5.8	Output voltage proportional to Magnetic Flux	49
53. Fig:5.9	Strain gauge and strain measuring instrument	50
54. Fig:5.10	Oscilloscope to see voltage graph	50
55. Fig:5.11	Structure of Strain Gage	51
56. Fig:5.12	Wheatstone Bridge	52
57. Fig:5.13	WB with two Gage system	52
58. Fig:5.14	Gages are connected to the opposite sides of the beam	52
59. Fig:5.15	Block diagram of INA125P connected with other peripheral	53
60. Fig:5.16	ATmega 328 Pin Mapping	54
61. Fig:5.17	Complete Experimental setup	55
62. Fig:5.18	Speed controller of DC motor	56
63. Fig:5.19	Actuation system and accessories to count RPM	57
64. Fig:5.20	Arduino programme for speed control of DC motor	57
65. Fig:5.21	Tachometer	57
66. Fig:5.22	Three-Components of Vibration meter	58
67. Fig:5.23	Two-Components of Vibration meter	58
68. Fig:5.24	Vibration meter with Arduino and LCD display	59
69. Fig:5.25	The arduino programme which was uploaded while performing this experiment.	59
70. Fig:5.26	LCD reading at the beginning	60
71. Fig:5.27	LCD reading during vibration	60
72. Fig:5.28	Vibrating by an exciter	60
73. Fig:5.29	Arduino Serial Monitor	60
74. Fig:5.30	HC06 module	61
75. Fig:5.31	Arduino with BT HC06 module	61
76. Fig:5.32	Arduino programme for BT vibration meter	62
77. Fig:5.33	Experimental setup for BT vibration meter	62
78. Fig:5.34	BT reading when beam was vibrating.	63
79. Fig:5.35	BT reading when beam was stopped	63
80. Fig:5.36	Laboratory equipped device for strain measurement	64

81. Fig:5.37	Block diagram of Laboratory equipped device for displaying strain data on Oscilloscope.	66
82. Fig: 5.38	Oscilloscope Reading of strain voltage from strain gage.	66
83. Fig:5.39	Mounting of ATmega micro-controller on arduino for uploading programme.	67
84. Fig:5.40	Procedure of uploading arduino programme on arduino	67
85. Fig:5.41	Arduino programme for wireless strain meter	68
86. Fig:5.42	Block diagram of Laboratory equipped device for sending strain data over Bluetooth	69
87. Fig:5.43	Tachometer showing critical speed	69
88. Fig:5.44	Strain data on phone via Bluetooth	69

## ***LIST OF TABLES***

<b>Table no.</b>	<b>Description</b>	<b>Page No.</b>
1. Table-1	Some common sensors	1
2. Table 2.1	Some important points regarding Intel Edison.	10
3. Table-5.1	Technical Specification of ATmega 328P	54

## **ABSTRACT**

ATMEGA328 is Atmel 8-bit AVR RISC-based microcontroller with 32KB ISP flash memory. Arduino UNO, the most used and documented board of the whole Arduino & Genuino family employs this microcontroller as its workhorse. ATMEGA328 can be programmed using the Arduino IDE. On the other hand, Intel Edison is a microcontroller with 32 bit dual core Intel Atom processor. The module is equipped with a Linux OS based on YOCTO and can run python, C/C++ programs.

The above-mentioned microcontrollers are widely used all over the world for multi-various applications. However, reports on their use in the field of vibration data acquisition and monitoring are comparatively scanty. The present thesis makes an attempt to assess these inexpensive devices in this direction.

In the first phase of the work, known voltage signals are generated from a state of the art, expensive data acquisition and control card. These signals are acquired in the microcontrollers and saved in SD card. The acquired data are then matched with the known input signals.

Next, a vibrating beam with an eccentric mass exciter is developed. The vibration signals are picked up by two different types of sensors – Hall Effect based displacement sensor and an electric wire resistance (EWR) strain gauge. The Hall sensor directly converts the displacement at a point of the beam into voltage. The EWR strain gauges generate small voltage signals corresponding to the resistance change in the strain bridge circuit. This small voltage signal is amplified by the known instrumentation amplifier IC INA125P.

The sensor data is acquired using the microcontrollers. The data is now displayed in a LCD module. However, wireless data acquisition is of more interest in this project. ATMEGA328 requires a separate Bluetooth module for wireless data transmission. HC06 Bluetooth module is used for this purpose. Intel Edison has its own Bluetooth hardware. Using Arduino code in ATMEGA328 and python code in Edison, the vibration data, be it displacement or strain now be acquired in a Bluetooth device. In the present work an Android mobile phone with BlueTerm+ app acquires the data.

It is observed during this project work that both ATMEGA328 and Edison cannot acquire data in real time. However, under normal conditions, data can be acquired with an average of a little more than 1ms interval. In order to compute Fast Fourier Transform (FFT), an interpolation of the data is performed to make it uniformly spaced in time. For high frequency vibration data the maximum and minimum can easily be computed and displayed.

# CONTENTS

<b>Subject</b>	<b>Page No.</b>
<b>CERTIFICATE OF APPROVAL</b>	<b>ii</b>
<b>THESIS CERTIFICATE</b>	<b>iii</b>
<b>ACKNOWLEDGEMENT</b>	<b>iv</b>
<b>LISTS OF FIGURES</b>	<b>v</b>
<b>LISTS OF TABLES</b>	<b>viii</b>
<b>ABSTRACT</b>	<b>ix</b>
<b>CONTENTS</b>	<b>X</b>

## Chapter-1

### Introduction

1.1) Introduction .....	1
1.2) Sensor .....	1
1.2.1) Some Common Sensors .....	1
1.3) Data acquisition .....	1
1.3.1) DAQ Device .....	2
1.3.1.1) Key Measurement Components of a DAQ Device .....	3
1.3.1.2) Computer's Role in a DAQ System .....	3
1.3.1.3) Different Software Components in a DAQ System .....	4
1.4) Monitoring .....	4
1.5) Brief history of Arduino .....	5
1.5.1) Advantages Of Arduino .....	6
1.6) Literature Review .....	7
1.7) Aim and scope of this work .....	9

# Chapter-2

## Data acquisition using Intel Edison

2.1) Introduction	10
2.2) Intel Edison	10
2.2.1) Key features	10
2.3) Introduction to Quanser – a control card with software	12
2.3.1) Data acquisition from Quanser	
2.3.1.1) Experimental Procedure	13
2.3.1.2) Results and Discussion	19
2.3.2) Data storage in SD card	
2.3.2.1) Experimental Procedure	20
2.3.2.2) Results and Discussion	21
2.4) Post processing of data and FFT in MATLAB	
2.4.1) Data Validation	22
2.4.2) Experimental Procedure	22
2.4.3) Results and Discussion	23
2.5) Edison and PySerial	
2.5.1) PySerial – A python library	26
2.5.2) Experimental Procedure	26
2.5.3) Results and Discussion	27

# Chapter-3

## Wireless vibration meter using Intel Edison

3.1) Introduction	29
3.2) Using Bluetooth in Intel Edison	
3.2.1) Enable Bluetooth in Edison	29
3.2.2) Pairing the Bluetooth Device	29
3.3) Blinking a LED with command sending over Bluetooth	

3.3.1) Checking the Bluetooth module	.....30
3.3.2) Experimental Procedure	.....30
3.3.3) Results and Discussion	.....32
3.4) Making a Bluetooth Vibration Meter	.....33
3.4.1) Experimental Procedure	.....33
3.4.2) Results and Discussion	.....34

## Chapter-4

### Performance of MCP 4921 as SPI based Digital-Analog Converter

4.1) Introduction	.....35
4.2) Definition of SPI	.....35
4.3) Serial Data Transfer through SPI Bus	.....35
4.3.1) MCP4921ADAC EXPERIMENT	
4.3.1.1) The goal of the experiment	.....36
4.3.1.2) Experimental procedure	.....36
4.3.1.3) Performance Test of MCP4921 Analog to Digital converter.	40
4.3.1.4) CONCLUSION	.....43

## Chapter-5

### Displacement and Strain Data Acquisition from a Vibrating Beam

5.1) Introduction	.....44
5.2) Description of the experimental set-up	.....44
5.2.1) Cantilever Beam	.....44
5.2.2) Actuation system	.....45
5.2.3) Sensing arrangement	
5.2.3.1) Sensors for Measuring Displacement Data	.....46
5.2.3.1.1) HALL Displacement Sensor	.....47

5.2.3.2) Sensors for Measuring Strain Data	
5.2.3.2.1) STRAIN GAGE	50
5.2.3.2.2) INA125P as a Strain Amplifier	53
5.2.3.2.3) Technical Specification of ATmega328P	53
5.2.3.2.4) ATmega 328 Pin Mapping	54
5.3) Complete Experimental setup	55
5.4) Speed control of DC motor with help of microcontroller	56
5.4.1) Brief description about the controller	56
5.4.2) Experimental Procedure	57
5.4.3) Results	57
5.5) Vibration meter with Arduino and LCD display	58
5.5.1) Hardware Required	58
5.5.2) Software Required	58
5.5.3) Experimental Procedure	59
5.5.4) Results and Discussion	60
5.6) Arduino BT vibration meter	61
5.6.1) HC06 Bluetooth Module	61
5.6.2) Experimental Procedure	61
5.6.3) Results And Discussion	63
5.7) ATmega-328P and INA125P with IC-7660 as vibrating strain meter	64
5.7.1) Brief description of the circuitry	65
5.7.2) Experiment procedure	65
5.7.3) Schematic of laboratory equipped instrument	66
5.7.4) Results and Discussion	66
5.8) ATmega-328P With BT as Wireless Strain Meter	67
5.8.1) Experimental procedure	67
5.8.2) Schematic of laboratory equipped instrument	69
5.8.3) Results and Discussion	69
5.9) Arduino, Firmata protocol, PySerial and Matplotlib	70
<b>FUTURE SCOPE OF THE WORK</b>	<b>71</b>
<b>REFERENCES</b>	<b>72</b>

---

## Introduction

---

**1.1. Introduction:** The aim of the present work is to develop an inexpensive actuation, sensing and data acquisition system for vibrating structures. Low cost sensors and micro controllers are now-a-days available in the market. Though these instruments are widely utilised in multi-various systems, their application in the field of vibration appears to be scanty. This chapter briefly introduces sensing and data acquisition systems. It also presents a brief historical background, literature review accompanied by objectives and scope of the present work.

**1.2. Sensor:** The measurement of a physical phenomenon, such as the temperature of a room, the intensity of a light source, or the force applied to an object, begins with a sensor. A sensor, also called a transducer, converts a physical phenomenon into a measurable electrical signal. Depending on the type of sensor, its electrical output can be a voltage, current, resistance, or another electrical attribute that varies over time. Some sensors may require additional components and circuitry to properly produce a signal that can accurately and safely be read by a DAQ device.

### 1.2.1. Some Common Sensors:

Sensor	Phenomenon
Thermocouple, RTD, Thermistor	Temperature
Photo Sensor	Light
Microphone	Sound
Strain Gage, Piezoelectric Transducer	Force and Pressure
Potentiometer, LVDT, Optical Encoder	Position and Displacement
Accelerometer	Acceleration
pH Electrode	pH

Table-1: Some common sensors

**1.3. Data acquisition:** Data acquisition (DAQ) is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition (DAQ) is the process of measuring an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound with a computer. Compared to traditional measurement systems, PC-based DAQ



Systems exploit the processing power, productivity, display, and connectivity capabilities of industry-standard computers providing a more powerful, flexible, and cost-effective measurement solution. DAQ typically converts analog waveforms into digital values for processing. The components of data acquisition systems include.

- a) Sensors, to convert physical parameters to electrical signals.
- b) Signal conditioning circuitry, to convert sensor signals into a form that can be converted to digital values.
- c) Analog-to-digital converters, to convert conditioned sensor signals to digital values.<sup>[1.1]</sup>

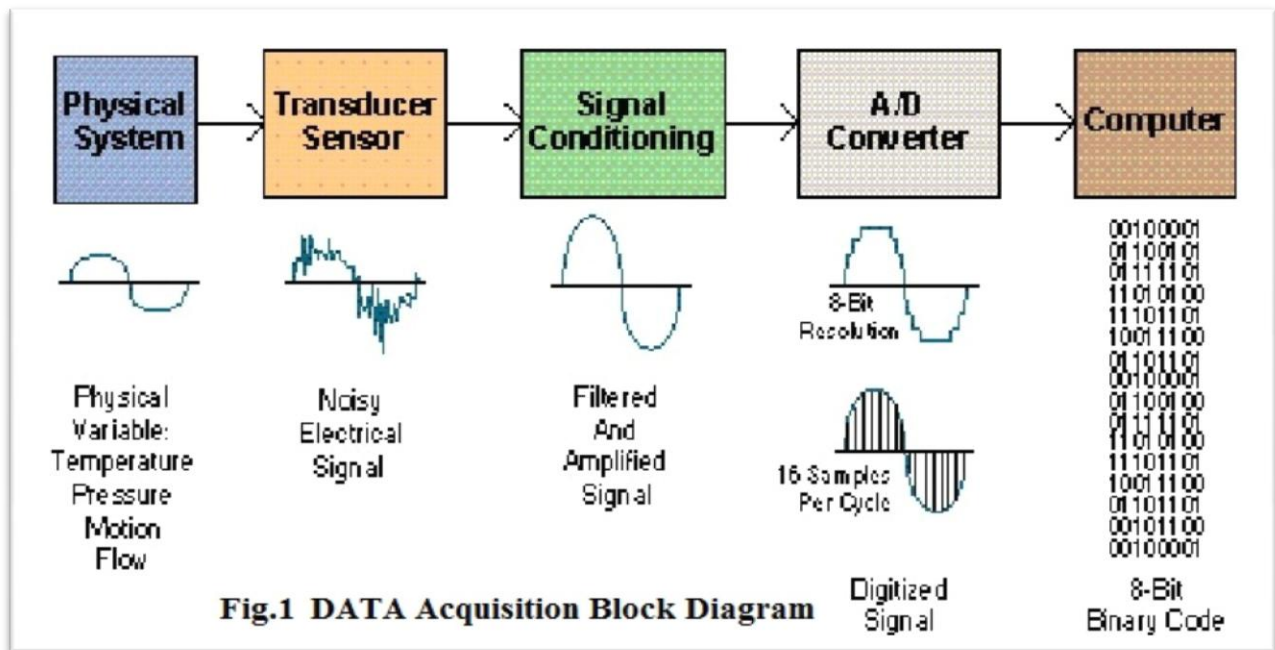


Fig.1.1: Block diagram of a very simple Data Acquisition System

**1.3.1. DAQ Device:** DAQ hardware acts as the interface between a computer and signals from the outside world. It primarily functions as a device that digitizes incoming analog signals so that a computer can interpret them. The three key components of a DAQ device used for measuring a signal are the signal conditioning circuitry, analog-to-digital converter (ADC), and computer bus. Many DAQ devices include other functions for automating measurement systems

and processes. For example, digital-to-analog converters (DACs) output analog signals, digital I/O lines input and output digital signals, and counter/timers count and generate digital pulses.

### **1.3.1.1. Key Measurement Components of a DAQ Device:**

**Signal Conditioning:** Signals from sensors or the outside world can be noisy or too dangerous to measure directly. Signal conditioning circuitry manipulates a signal into a form that is suitable for input into an ADC. This circuitry can include amplification, attenuation, filtering, and isolation. Some DAQ devices include built-in signal conditioning designed for measuring specific types of sensors.

**Analog-to-Digital Converter (ADC):** Analog signals from sensors must be converted into digital before they are manipulated by digital equipment such as a computer. An ADC is a chip that provides a digital representation of an analog signal at an instant in time. In practice, analog signals continuously vary over time and an ADC takes periodic “samples” of the signal at a predefined rate. These samples are transferred to a computer over a computer bus where the original signal is reconstructed from the samples in software.

**Computer Bus:** DAQ devices connect to a computer through a slot or port. The computer bus serves as the communication interface between the DAQ device and computer for passing instructions and measured data. DAQ devices are offered on the most common computer buses including USB, PCI, PCI Express, and Ethernet. More recently, DAQ devices have become available for 802.11 Wi-Fi for wireless communication. There are many types of buses, and each offers different advantages for different types of applications.

### **1.3.1.2. Computer’s Role in a DAQ System:**

A computer with programmable software controls the operation of the DAQ device and is used for processing, visualizing, and storing measurement data. Different types of computers are used in different types of applications. A desktop may be used in a lab for its processing power, a laptop may be used in the field for its portability, or an industrial computer may be used in a manufacturing plant for its ruggedness.

### **1.3.1.3. Different Software Components in a DAQ System:**

**Driver Software:** Driver software provides application software the ability to interact with a DAQ device. It simplifies communication with the DAQ device by abstracting low-level hardware commands and register-level programming. Typically, DAQ driver software exposes an application programming interface (API) that is used within a programming environment to build application software.

**Application Software:** Application software facilitates the interaction between the computer and user for acquiring, analyzing, and presenting measurement data. It is either a prebuilt application with predefined functionality, or a programming environment for building applications with custom functionality. Custom applications are often used to automate multiple functions of a DAQ device, perform signal-processing algorithms, and display custom user interfaces.<sup>[1,2]</sup>

**1.4. Monitoring:** Monitoring is the regular observation and recording of activities taking place in a project or programme. It is a process of routinely gathering information on all aspects of the project. To monitor is to check on how project activities are progressing. It is observation; — systematic and purposeful observation. Monitoring also involves giving feedback about the progress of the project to the donors, implementers and beneficiaries of the project.

Reporting enables the gathered information to be used in making decisions for improving project performance.

Monitoring provides information that will be useful in:

- Analysing the situation in the community and its project
- Determining whether the inputs in the project are well utilized
- Identifying problems facing the community or project and finding solutions
- Ensuring all activities to be carried out properly in time by the right people
- Using lessons from one project experience on to another and

- Determining whether the way the project was planned is the most appropriate way of solving the problem at hand.<sup>[1, 3]</sup>

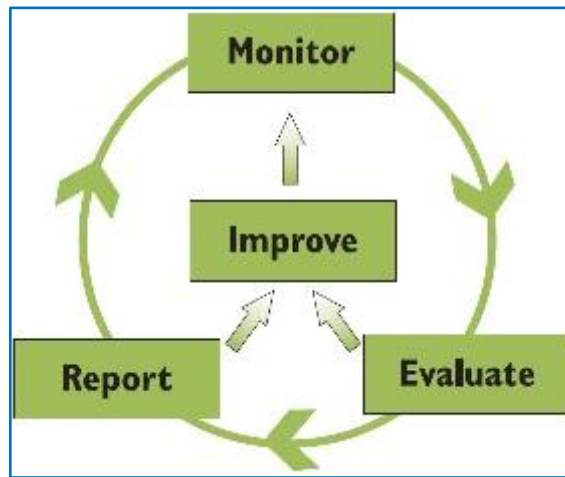


Fig. 1.2: Block diagram of a very simple Monitoring System.

For performing data acquisition and Monitoring, low cost Arduino and Intel Edison serve the purpose of this thesis. Arduino is a microcontroller based tinkering device for performing various micro-controller powered work which are cost justified and meet expectation in our daily life. In the next section evolution of Arduino is stated and in the next chapter a brief description of Intel Edison will be expressed.

### **1.5. Brief history of Arduino:**

Colombian student Hernando Barragán created the development platform wiring as his Master's thesis project in 2004 at the Interaction Design Institute Ivrea in Ivrea, Italy. Massimo Banzi and Casey Reas (known for his work on Processing) were supervisors for his thesis. The goal was to create low cost, simple tools for non-engineers to create digital projects. The Wiring platform consisted of a hardware PCB with an ATmega128 microcontroller, an integrated development environment (IDE) based on Processing and library functions to easily program the microcontroller.

In 2005, Massimo Banzi, with David Mellis (then an IDII student) and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work

on Wiring, they forged (or copied) the Wiring source code and started running it as a separate project, called Arduino.

The Arduino's initial core team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis.

The name *Arduino* came from a bar in Ivrea, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

Following the completion of the Wiring platform, its lighter, lower cost versions were created and made available to the open-source community. Associated researchers, including David Cuartielles, promoted the idea.<sup>[1.4]</sup>

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. One can tell Arduino board what to do by sending a set of instructions to the microcontroller on the board. To do so one use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

**1.5.1. ADVANTAGES OF ARDUNIO:** Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino UNO modules cost less than Rs. 2000 rupee.
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage as well. For teachers, it is conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people willing to understand the technical details can make the leap from Arduino to the AVR C programming language on which it is based. Similarly, AVR-C code can be added directly into Arduino programs, if needed.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so that experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and to save money. <sup>[1.5]</sup>

## 1.6. Literature Review:

**Sheikh Ferdoush, Xinrong Li**<sup>[1.6]</sup> described a wireless sensor network system that was developed using open-source hardware platforms, Arduino and Raspberry Pi. The system is low-cost and highly scalable both in terms of the type of sensors and the number of sensor nodes, thereby making it well suited for a wide variety of applications related to environmental monitoring.

**Luis J. Claros-Marfil, J. Francisco Padial, Benito Lauret**<sup>[1.7]</sup> showed a new and inexpensive open source data acquisition and controller for solar research: Application to a water-flow glazing.

**Varun Kumar**<sup>[1.8]</sup> built a surveillance car which can be controlled over Internet or any private network. His objective was to do, easily and cost effectively, something that can be used for security purposes. His car uses DTMF module which takes its input from mobile phone attached to the car; that mobile is used not only to generate DTMF tones but also for the surveillance by providing Live Footage from its camera, over the network using AirDroid.

**Jason Poel Smith**<sup>[1.9]</sup> described the process of controlling an Arduino with a TV Remote. This project uses a multi-protocol infrared remote library that was developed by Ken Shirriff. This library allows the Arduino to both decode and transmit the infrared signals that are used in most commercial remote control systems.

**Jens Christoffersen**<sup>[1.10]</sup> has shown us how to use the internal EEPROM of an ATmega 328P microcontroller. He had used a USB-to-serial converter, and an LM35 temperature sensor. The circuit will measure from 2 different sensors, and store them in its memory.

LM35 Temperature Sensor with Data-logging on SD card on Intel Edison was described by **vishal1502**<sup>[1.11]</sup>. His work demonstrates the use of an LM35 sensor on Intel Edison to measure temperature over extended periods ranging from several hours to weeks and logs the temperature readings from the sensor to an SD card inserted on-board; all necessary things such as time interval between two consecutive readings, reading temperature either in degrees Celsius or in Fahrenheit can be varied from this code itself.

**Shachindra\_1992**,<sup>[1.12]</sup> in his work titled Sensor Data Monitoring with Edison (Intel IoT), instructed for integrating the grove sensors and actuators with Intel Edison by NodeJS, and for monitoring the sensor data such as air quality, Sound, Temperature, Light, Touch and LED, Buzzers.

**TwitterPlotBot on Edison by NavinB**<sup>[1.13]</sup> has demonstrated an experiment to build something that is going to bring together social networking and data analytics(sort of). He has created a twitter bot that tweets plots of temperature (or any user defined data). This bot creates the plots using the "gnuplot" program and uses Twython Python library to tweet the generated plots. This

bot can log data and tweet plots at configurable intervals. The data that has to be logged and plotted subsequently is user configurable and hence we can log any data that we wish to record and plot. Also it is possible to plot multiple data sources.

**1.7. Aim and scope of this work:** ATMEGA328 is Atmel 8-bit AVR RISC-based microcontroller with 32KB ISP flash memory. Arduino UNO, the most used and documented board of the whole Arduino & Genuino family, employs this microcontroller as its workhorse. ATMEGA328 can be programmed using the Arduino IDE. On the other hand, Intel Edison is a microcontroller with 32 bit dual core Intel Atom processor. The module is equipped with a Linux OS based on YOCTO and can run python, C/C++ programs.

The above-mentioned microcontrollers are widely used all over the world for multi-various applications. However, reports on their use in the field of vibration data acquisition and monitoring are comparatively rare. The present thesis makes an attempt to assess the performance of these inexpensive devices in this direction. For any real life problem, where failure is to be checked and data needs to be monitored for maintaining certain standard references, these devices could be as useful as any sophisticated monitoring system with an exception that it would come at a much lower cost.



---

## Data acquisition using Intel Edison

---

**2.1. Introduction:** As it was stated in the first chapter, data acquisition helps a system operator to monitor a system's performance and storing the same data helps in analysing and predicting system's behaviour in particular environment. Intel Edison is a low cost System on Chip that helps engineer to acquire data and store it for monitoring and analyzing purpose.

### 2.2. Intel Edison:

The Intel<sup>®</sup> Edison module is a SoC (System on Chip) that operates on Yocto Linux and includes an Intel<sup>®</sup> Atom™ 500MHz dual-core, dual-threaded CPU and a 32-bit Intel<sup>®</sup> Quark™ 100MHz microcontroller. Table 2.1 shows some important points regarding Intel Edison.

Operating Voltage	3.3V/5V
Input Voltage	7-15V
Digital I/O Pins	20 (of which 6 provide analog input and 4 provide PWM output, and rest are digital output)
Flash Storage	4 GB eMMC
RAM	1 GB LPDDR3 POP
Clock Speed	500 MHz (Intel <sup>®</sup> Atom™ CPU) 100 MHz (Quark™ microcontroller)
Length	35.5mm (Edison), 127mm (Kit for Arduino)
Width	25mm (Edison), 72mm (Kit for Arduino)
Height	4mm (Edison) 12mm (Kit for Arduino)

Table- 2.1: Some important points regarding Intel Edison

#### 2.2.1. Key features:

- Integrated Wi-Fi, Bluetooth 4.0 LE
- Support for Yocto Linux, Python, Node.js and Wolfram <sup>[2.1]</sup>

Atom is Intel's family of x86 and x86-64 processors that are optimized for small computing devices, such as smart-phones and mobile Internet devices (MIDs). Most notebooks on the market today run on Atom. An Atom processor performs at a level about half that of an equivalent Pentium chip. The trade-off between power, consumption and performance is intentional because the target devices for the processor are rather than compute-intensive activities like gaming.<sup>[2.2]</sup>

A dual-core CPU has two central processing units. So, it appears to the operating system as two CPUs. A different process can be using each core at the same time. This speeds up the system, because the computer can do multiple things at once.<sup>[2.3]</sup>

500 MHz dual-core means the maximum number of clock cycles per second that the RAM operates on.  $500 \text{ MHz clock rate} \times 2 \text{ (for Dual-core DDR, 1 for SDR)} \times 8 \text{ Bytes} = 8000 \text{ MB/s bandwidth or } 500 \times 2 = 1000 \text{ MT/s}$ . RAM chips are not named based on frequency at all, but on data rate, which is measured in millions of transfers per second (MT/s).<sup>[2.4]</sup>

The original Pentium 4 had just a single CPU core, so it could only do one thing at a time — but hyper-threading attempted to make up for that. A single physical CPU core with hyper-threading appears as two logical CPUs to an operating system. The CPU is still a single CPU, so it's "cheating" a bit — while the operating system sees two CPUs for each core, the actual CPU hardware only has a single set of execution resources for each core. The CPU pretends it has more cores than it has, and it uses its own logic to speed up program execution. Hyper-threading allows the two logical CPU cores to share physical execution resources. This can speed things up somewhat — if one virtual CPU is stalled and waiting, the other virtual CPU can borrow its execution resources. Hyper-threading can help speed the system up, but it's nowhere near as good as having additional cores.<sup>[2.3]</sup>

Hyper-threading is now an added advantage. While the original consumer processors with hyper-threading only had a single core that masqueraded as multiple cores, modern Intel CPUs now have both multiple cores and hyper-threading technology. The dual-core CPU with hyper-threading appears as four cores to the operating system, while the quad-core CPU with hyper-threading appears as eight cores. Hyper-threading is no substitute for additional cores, but a dual-

core CPU with hyper-threading should perform better than a dual-core CPU without hyper-threading.<sup>[2.3]</sup>

**The Intel Edison has a single core microcontroller in addition to its dual core processor.**

The single-core micro controller (Intel Quark, SoC) runs a real-time operating system which manages the hardware resources on device. A microcontroller can do work that a full-blown CPU running Linux might struggle with. This includes tasks with tight requirements around: Timing and Power.

**Timing:** One can tweak microcontroller behaviour down to the cycle level, and push data around at a low level without worrying about overheads or OS interference.

**Power:** A microcontroller usually runs at a lower current, can put itself into sleep mode, and wakes on interrupts. The switching between waking up and going back to sleep mode happens very quickly, thereby reducing power consumption drastically in some applications.

The trade off is that it becomes much more difficult to write microcontroller software as per the growing complexity of the requirements. For instance, talking to a sensor might be easier with a microcontroller, but talking to a server via HTTPS would be a tough work to be accomplished. A combination of the two, each doing what they're best at, can be a great combination.<sup>[2.5]</sup>

**2.3. Introduction to Quanser – a control card with software:** Quanser's QUARC software adds powerful tools to MATLAB<sup>®</sup> and Simulink<sup>®</sup> to make the development and deployment of sophisticated real-time mechatronics and control applications easier. QUARC generates real-time code directly from Simulink-designed controllers and runs it in real-time on the Windows target - all without digital signal processing or without writing a single line of code.<sup>[2.6]</sup>

## 2.3. 1. Data acquisition from Quanser:

### 2.3. 1.1. Experimental Procedure:

Following steps had been taken for data acquisition.

- 1) The Edison was connected to a system having Quanser and a SD card was attached with the Edison.
- 2) The USB serial port was accessed through the path: My Computer>Manage Computer >Device Manager>Ports (COM & LPT)>USB Serial Port(here it is COM9 as shown in Fig.2.1).

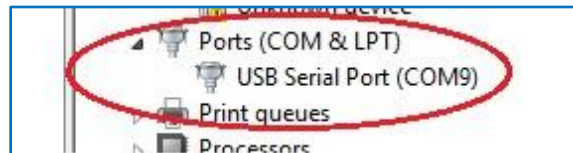


Fig.2.1: COM on device manager

- 3) after opening putty configuration following information was inserted as shown in Fig.2.2.

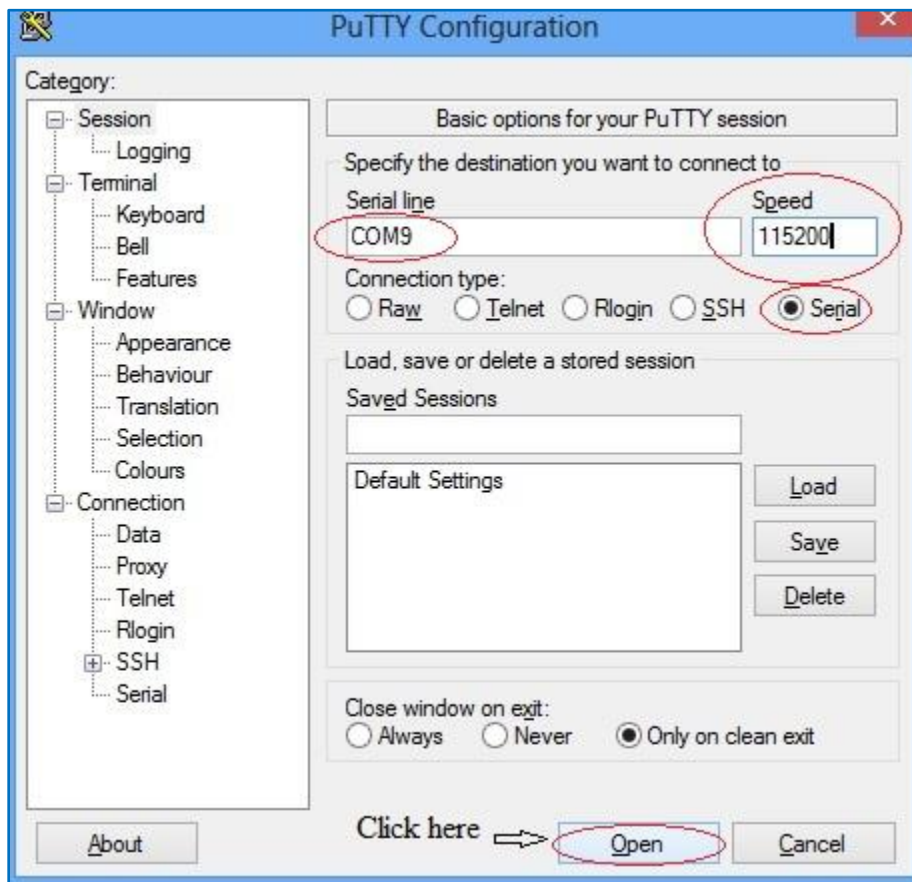
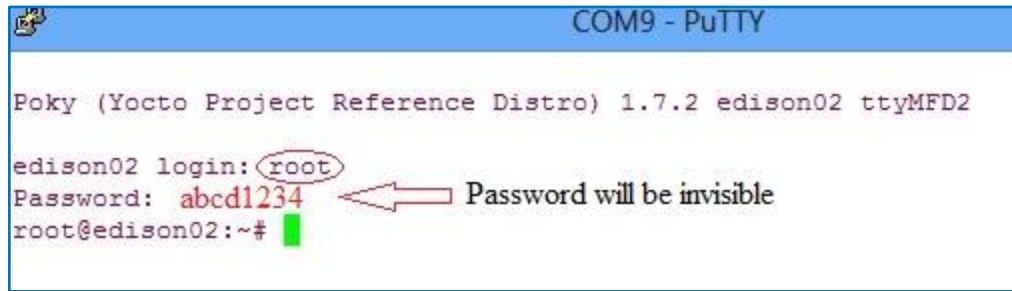


Fig.2.2: Putty login information

4) after logging in 'enter' was pressed and then username and password were entered as shown in Fig.2.3.



```
COM9 - PuTTY
Poky (Yocto Project Reference Distro) 1.7.2 edison02 ttyMFD2
edison02 login: root
Password: abcd1234
root@edison02:~#
```

Fig.2.3: User name and password for edison02 login

**read\_analogData\_Quanser.py** was now located and the execution of the programme was temporarily postponed until some other action was performed.

5) Now Edison was connected to computer using two USB cables and powered on using a 12V DC power supply as shown in Fig.2.4. The Edison must be connected to 12V DC supply, otherwise it will not show up in computer.



Fig.2.4: Two USB serial ports connected between computer and Edison

6) Below is a snipe illustrating all the connections one needs to be perform for DATA acquisition from computer to Edison via a Q8\_USB device.

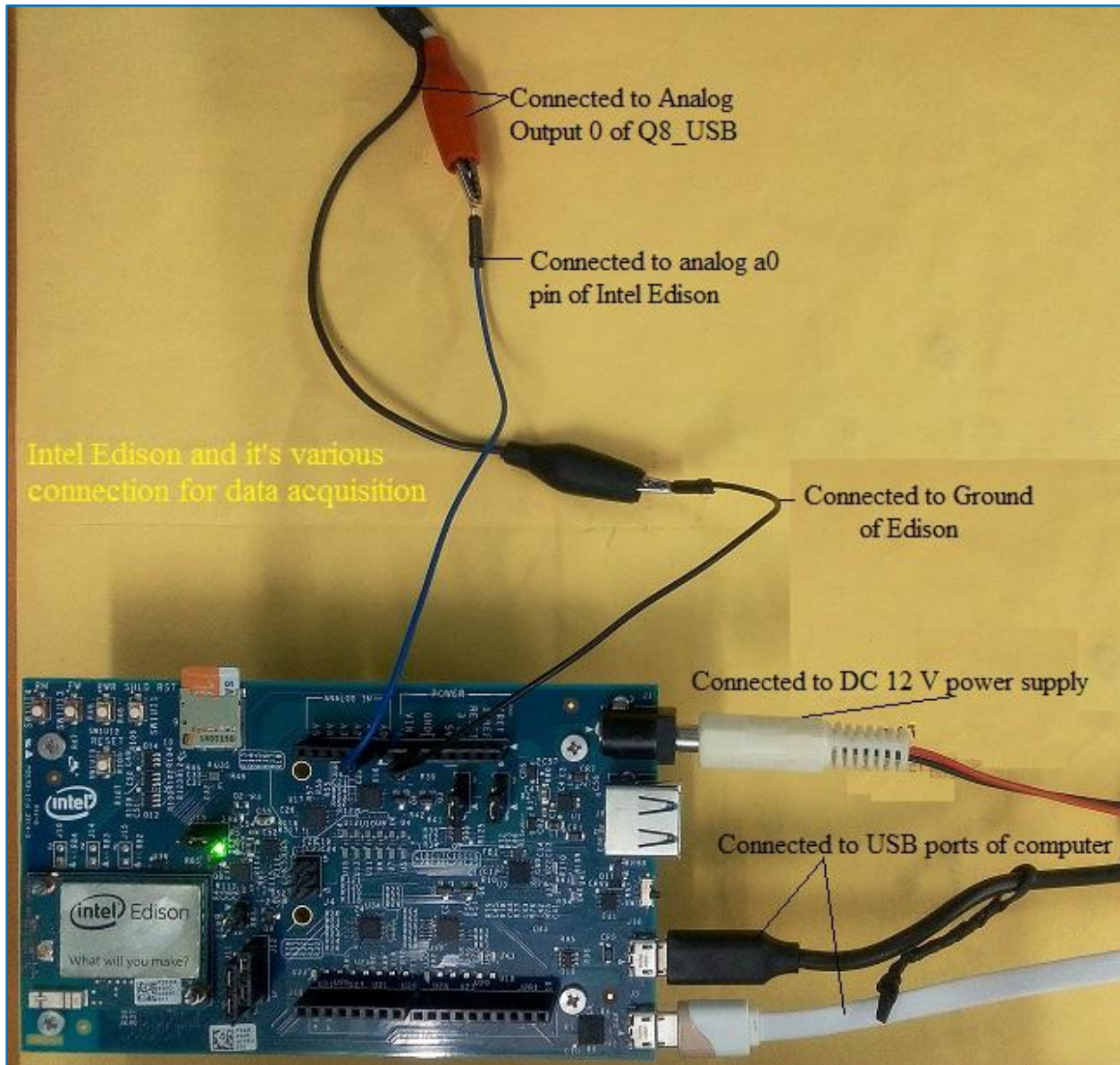


Fig.2.5: Various connections to Intel Edison for performing data acquisition



7) The below figure.2.6 demonstrates the design of Q8\_USB device after the required connections were done to Edison.



Fig.2.6: Connections of Q8\_USB

Now in MATLAB with integrated Quanser, a Simulink diagram was created for this experiment and named single\_sinusoidal\_5volt\_edison.

All the connections shown here must be made strictly. In Simulink diagram as shown in Fig.2.7, it is illustrated how to generate a sinusoidal signal and subsequently convert it to an analog form using Q8\_USB. The same analog signal was then received at Edison using two crocodile connector. Later on the voltage and corresponding time were captured using a python programme which was developed using the firmware existing in the Edison. The experimental data, thus generated in Simulink, was now appeared on putty serial monitor and was getting displayed continuously until the programme was stopped. Putty was hosted in another computer. It was used to access Edison's firmware.

8) The sketch was built and connected to target. Now before clicking on RUN, it was necessary to connect Edison in between the **Simulink sketch** and **read\_analogData\_Quanser.py** (which can be accessed via putty) as shown in Fig.2.5.

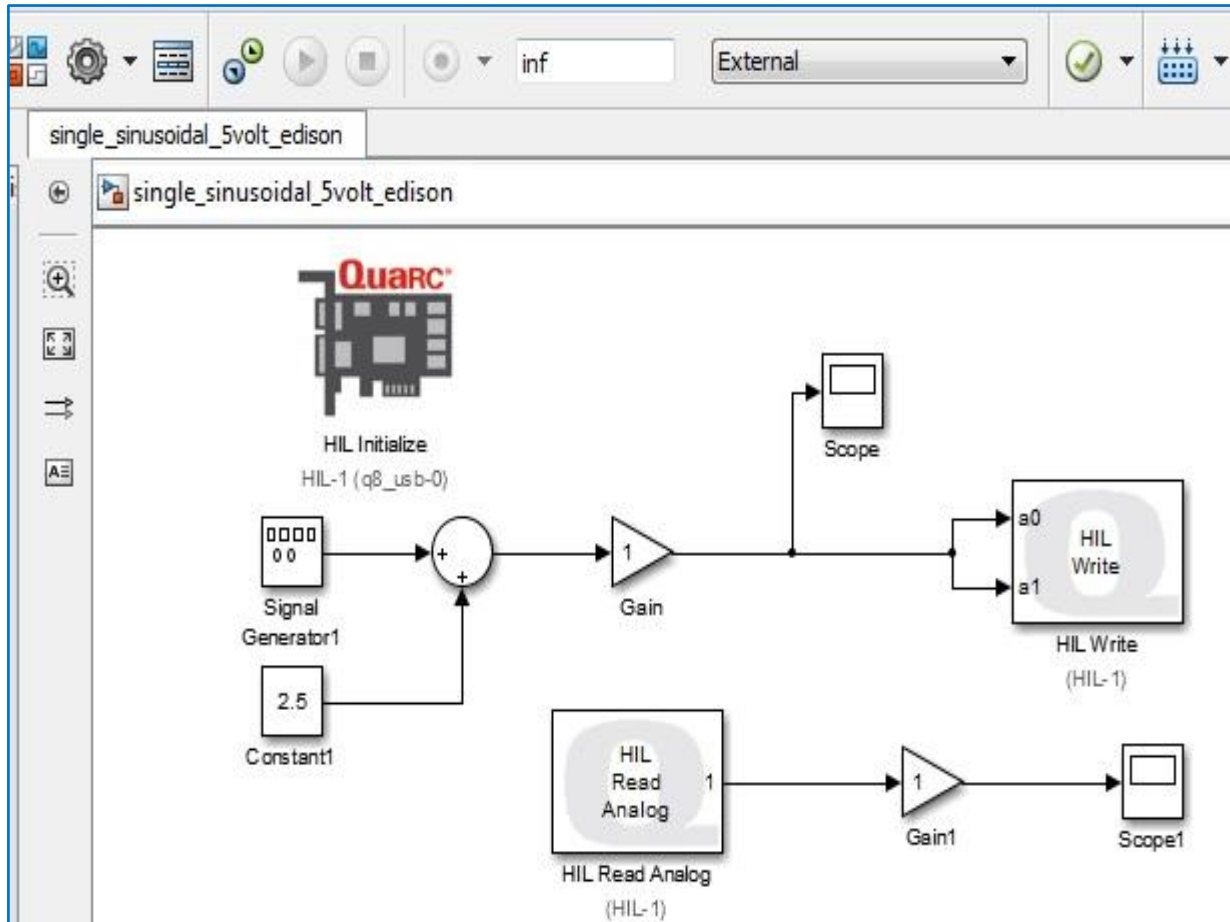


Fig.2.7:Simulink diagram for data acquisition

9) Before the signal could be generated on Simulink and sent to Edison from Simulink model, it was needed to be sure that a signal, not more than 5V, was generated for Edison to function properly. For that purpose, scope was provided in the Simulink so that it could check the signal generated by Simulink model. Now it was necessary to validate that all such input signals got fed into the Edison. Hence, scope1 was there (Fig.2.8) to check the output signal from Q8\_USB or input signal to Edison.



For the safety reason of Edison a**4.8V**, sinusoidal wave signal was sent to Edison.

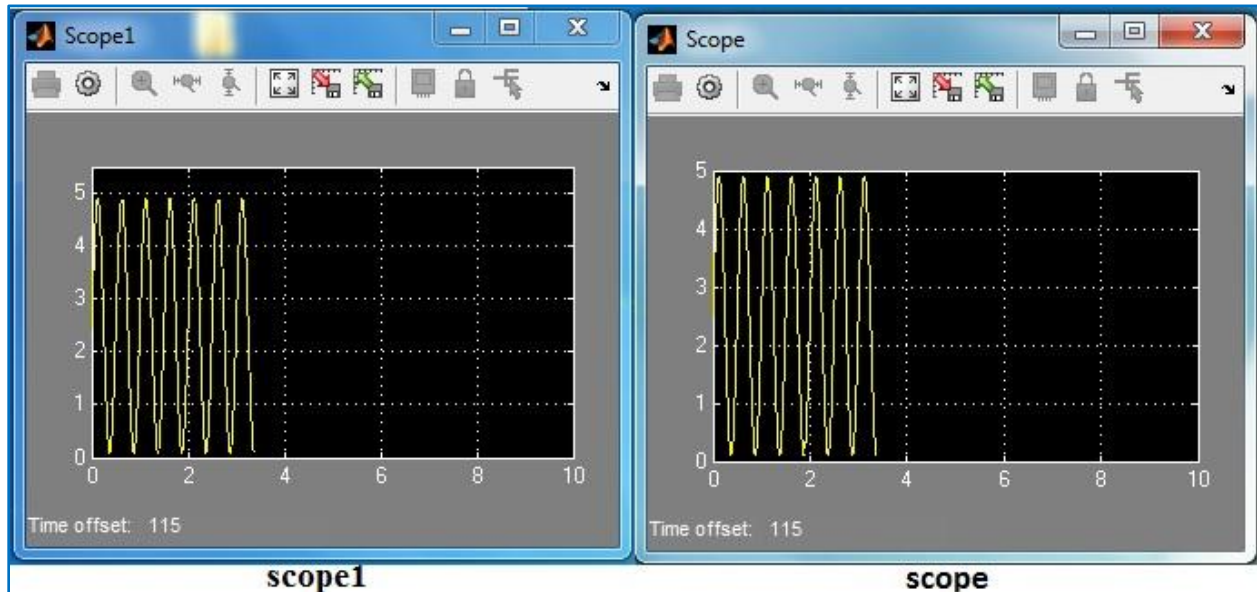


Fig.2.8: Two scopes for checking generated and input signal to Edison via Q8\_USB Scope shows the very signal that was generated using Simulink diagram as shown in Fig.2.7. Below Fig.2.9 shows the screen shot of python programme **read\_analogData\_Quanser.py**.

```
COM9 - PuTTY
GNU nano 2.2.6 File: read_analogData_Quanser.py

import time
from datetime import datetime
import mraa

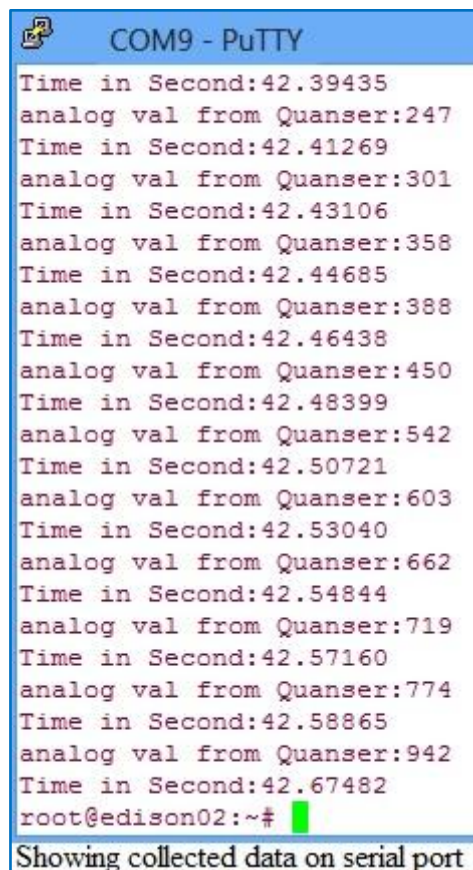
a0 = mraa.Aio( 0)
a0. setBit( 10)
myTime = 0
now1 = datetime.now()
print(now1)
try:
    for x in range (1,10001):
        val = a0. read()
        myTime = datetime.now()
        currentSecond = myTime.second
        currentMicrosecond = myTime.microsecond
        print "analog val from Quanser:"+str(val)
        time_sec =(str(currentSecond)+ "." +str(currentMicrosecond))
        print "Time in Second:"+str(time_sec)
        time.sleep(0.001)
except KeyboardInterrupt:
    print "done"
```

Fig.2.9: Python programme<sup>[7]</sup> for acquiring data in Edison

### 2.3.1.2. Results and Discussion:

From basic knowledge of python language and Nano editor, read\_analogData\_Quanser.py programme was developed. Here is a screen shot from putty serial terminal in Fig.2.10 showing how data will be continuously displayed on putty screen. The data was collected after an interval of one millisecond. For a 5V sinusoidal signal on Simulink model, in general, voltage value continuously changes from '0' to '1024' in Edison.

Fig.2.10 shows a screen shot of voltage values that were observed on putty serial monitor during our experiment. The range of the voltage values fell in the range of '0' to '983' which was expected considering the 4.8V sinusoidal signal.



```
COM9 - PuTTY
Time in Second:42.39435
analog val from Quanser:247
Time in Second:42.41269
analog val from Quanser:301
Time in Second:42.43106
analog val from Quanser:358
Time in Second:42.44685
analog val from Quanser:388
Time in Second:42.46438
analog val from Quanser:450
Time in Second:42.48399
analog val from Quanser:542
Time in Second:42.50721
analog val from Quanser:603
Time in Second:42.53040
analog val from Quanser:662
Time in Second:42.54844
analog val from Quanser:719
Time in Second:42.57160
analog val from Quanser:774
Time in Second:42.58865
analog val from Quanser:942
Time in Second:42.67482
root@edison02:~#
Showing collected data on serial port
```


Fig.2.10: Acquired data rendering on putty terminal

## 2.3.2. DATA STORAGE IN SD CARD:

In this section it will be explained how to store the real-time data in an external SD card by Edison. As it was stated in previous section that acquired data helps an engineer in a great way to analyse the behaviour and performance of a system in a particular environment and thus those analysis could be used in reducing cost of the prototype or enhancing system's performance.

**2.3.2.1. Experimental Procedure:** All the processes are same as illustrated in previous section titled 'Data acquisition from Quanser'.

1) Python programme had to be changed a little bit to save and store data in an external SD card. Following Fig.2.11 shows the changes which were made in our python programme `read_analogData_Quanser.py` and renamed `save_analogData_Quanser.py`.



```
COM9 - PuTTY
GNU nano 2.2.6 File: save analogData Quanser.py

import time
from datetime import datetime
import mraa

a0 = mraa.Aio( 0)
a0. setBit( 10)
myTime = 0
myList=[]
now1 = datetime.now()
print(now1)
for x in range (1,10001):
    val = a0. read()
    myTime = datetime.now()
    currentSecond = myTime.second
    currentMicrosecond = myTime.microsecond
    print "analog val from Quanser:"+str(val)
    time_sec =(str(currentSecond)+ "." +str(currentMicrosecond))
    print "Time in Second:"+str(time_sec)
    var = (str(currentSecond)+ " " +str(currentMicrosecond)+ " "+str(val))
    myList.append(var+" "\n")
    time.sleep(0.001)
fileOut = open('/media/sdcard/save_Data_Edison.txt','w')
fileOut.writelines(myList)
now2 = datetime.now()
print(now2)
```

Fig.2.11: Python programme<sup>[7]</sup> for acquiring and saving data in Edison

### 2.3.2.2. Results and Discussion:

After the programme was run for 1000 data, the data file was generated and stored in a location. By following this path `media/sdcard/save_Data_Edison.txt` the data file was checked and accessed. After the programme was stopped, it was needed to check whether the data got saved in SD card. For that purpose it was necessary to dismount the SD card from Edison and insert the same into a card reader as shown in Fig.2.12.



Step-1

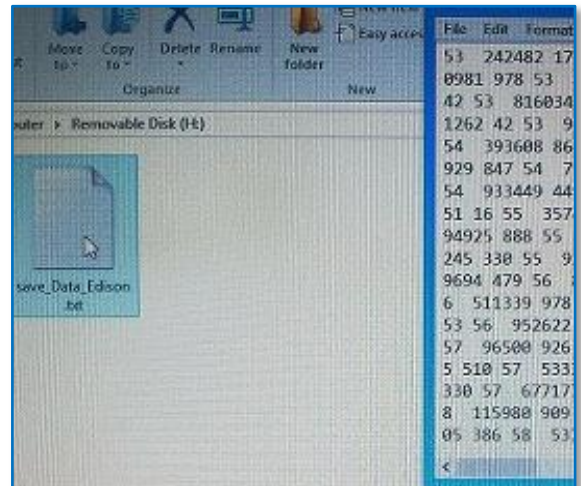


Step-2

Then it was plugged in a USB port of a computer and finally it was open in a windows explorer as shown in 4 steps in Fig.2.12. Thus one can access the generated data file whenever required.



Step-3



Step-4

Fig.2.12: Four steps to check and access saved data



## 2.4. POST PROCESSING OF DATA AND FFT IN MATLAB:

**2.4.1. Data Validation:** After having the data file in the SD card, it was necessary to check whether the data stored was correct i.e. whether Edison provided the same sinusoidal signal which was fed during data acquisition. To validate the same, it was required to use MATLAB software.

The data file which was collected in previous step was now loaded to MATLAB. Then we tried to obtain a sinusoidal graph, since a sinusoidal signal was fed to Edison in data acquisition step. After performing the above step, it was observed that the time intervals were not equal. It was important to correct the graph from the data file. In this section, it is explained how to modify the data file and get a uniformly spaced signal.

**2.4.2. Experimental Procedure:** Following steps were required to be performed.

- 1) Saved data file (in this case **save\_Data\_Edison.txt**) was placed in MATLAB directory containing **man1.m** and **fft\_calc.m** mat file as shown in Fig.2.13.
- 2) Since Edison did not provide equal interval data, the saved data file was processed through a MATLAB operation which, in turn, generated an intermediate file, **results1.txt**, with the help of **man1.m**.

```
clear all
abc=load('save_Data_Edison.txt');
sum=0;

abct=zeros(size(abc,1),1);
abct(1)=abc(1,1)*1e6+abc(1,2);
for i=2:size(abc,1)
    if abc(i,1)==0 && abc(i-1,1)==59;sum=sum+60;end
    abct(i)=(sum+abc(i,1))*1e6+abc(i,2);
end
abct=abct-abct(1);

abctt=linspace(0,abct(length(abct)),length(abct));
yy=interp1(abct,abc(:,3),abctt,'linear');
plot(abctt,yy)
hold
plot(abct,abc(:,3),'y.')
xxx=[abctt'*1e-6 yy'];
save results1.txt xxx -ascii
```

Fig.2.13:MATLAB programme 'man1.m.'for imparting equal interval to saved data file

- 3) The man1.m file was executed and the following sinusoidal frequency (Fig.2.15) was appeared.
- 4) After results1.txt was obtained (in the same directory), it was time to run fft\_calc.m as shown in Fig.2.14 to get the result of Fourier transformation as shown in Fig.2.16.

```

clear all
n_points=4096;
mskip=1;
load results1.txt
[m,n]=size(results1);
yyy=results1(m-mskip*n_points+1:mskip:m,2);
%yyy=results1(m-n_points+1:m,3);
yyy=yyy.*barthannwin(n_points);
zzz=abs(fft(yyy));
t1=results1(2,1);
t2=results1(3,1);
delt=mskip*(t2-t1);
%delt=(t2-t1);
fmax=1./delt;
zmax=max(zzz);

zzz=20*log10(zzz/zmax);
%zzz=zzz/zmax;
f=0:fmax/(n_points-1):fmax;
plot(f,zzz)
axis([0,fmax/2,-50,0])
%axis([0,1000,0,1])
grid on

```

Fig.2.14: MATLAB programme 'fft\_calc.m' for Fast Fourier Transformation

### 2.4.3) Results and Discussion:

The Fig.2.15 is a replica of same sinusoidal signal which was fed to Edison from Simulink as shown in Scope of Fig.2.8. From the Fig.2.16, it could be said that post processing of data in MATLAB rectified data file and gave the same signal (in this case 10 Hz frequency).

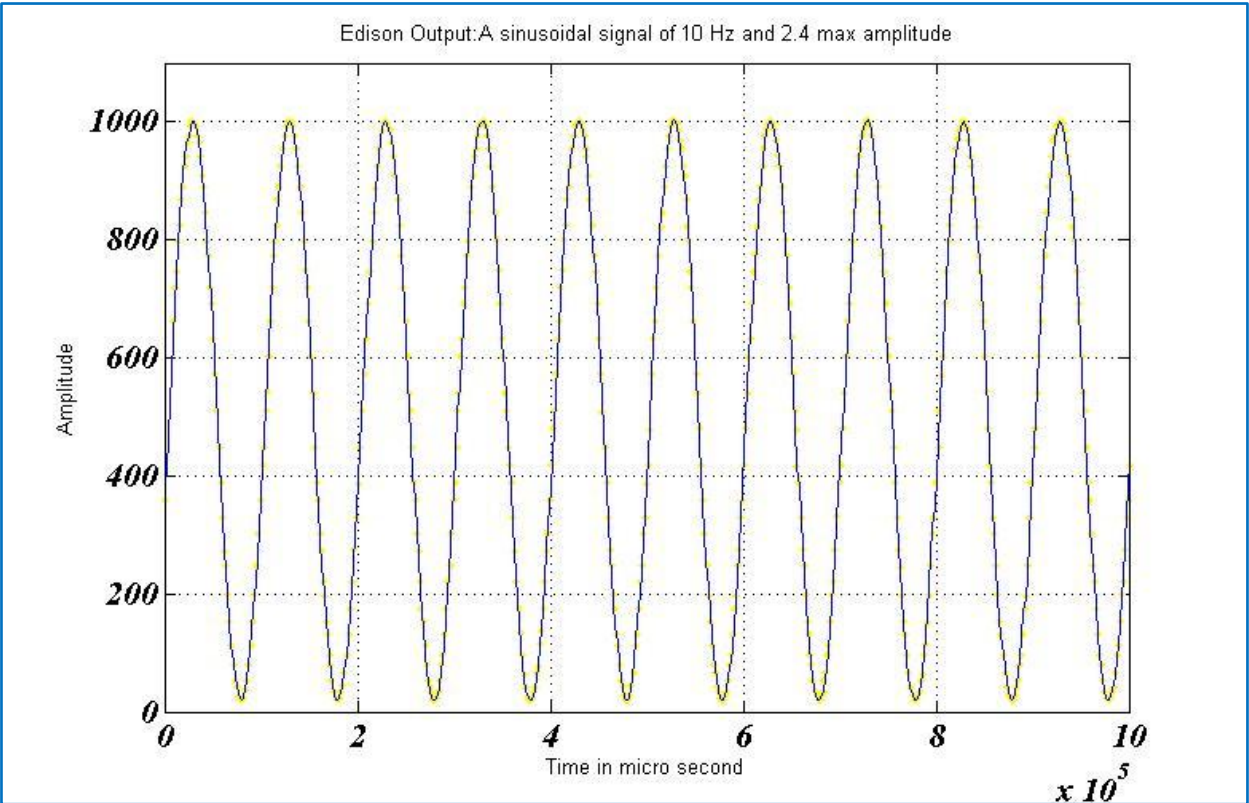


Fig.2.15: Output signal from saved data file after modifying by MATLAB.

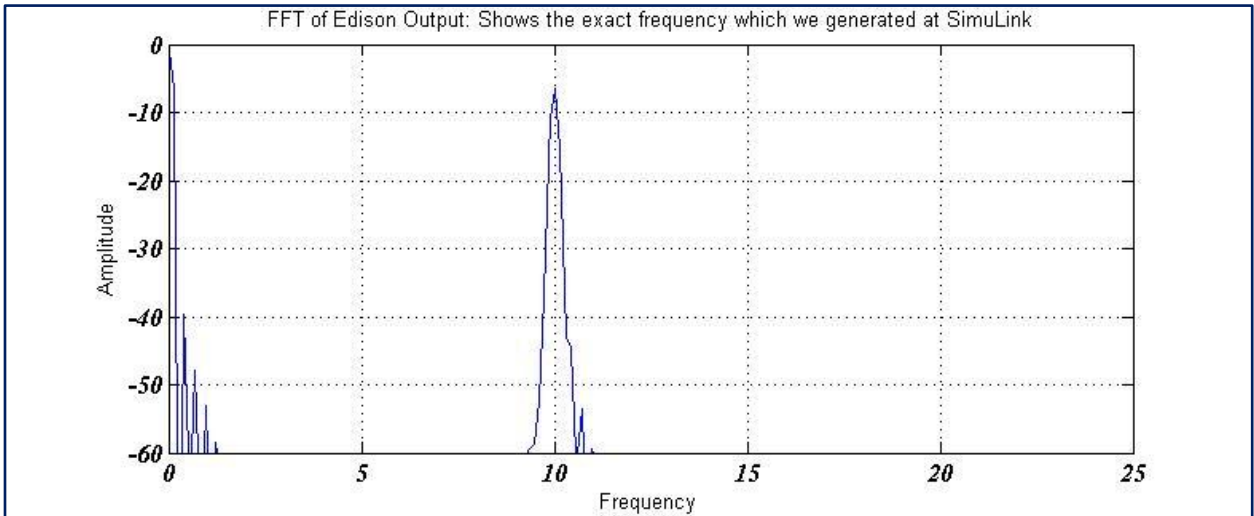


Fig.2.16: FFT of saved data file of Edison

From the graph of Fourier transformation as shown in Fig.2.16, it can be concluded that one can get back the same signal with same frequency and amplitude as it was previously generated in Simulink during data acquisition from Quanser.

Similarly, one can test Edison's output with a combination of any other frequency less than 50 Hz. The following Fig.2.17 shows Simulink diagram for a combination of 20 and 40 Hz signal.

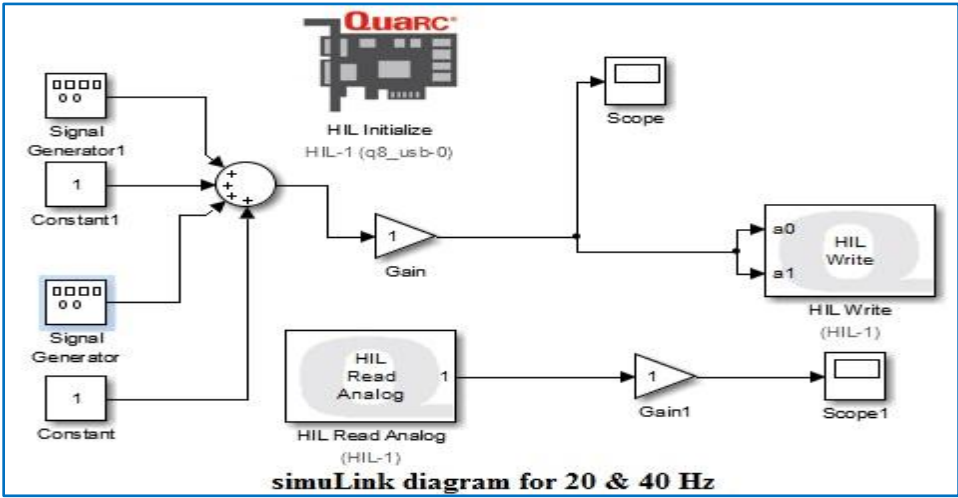


Fig.2.17: Simulink diagram for combination of frequency

Fig.2.18 depicts a signal which is a combination of 20 and 40 Hz frequency. And its FFT is shown in Fig.2.19.

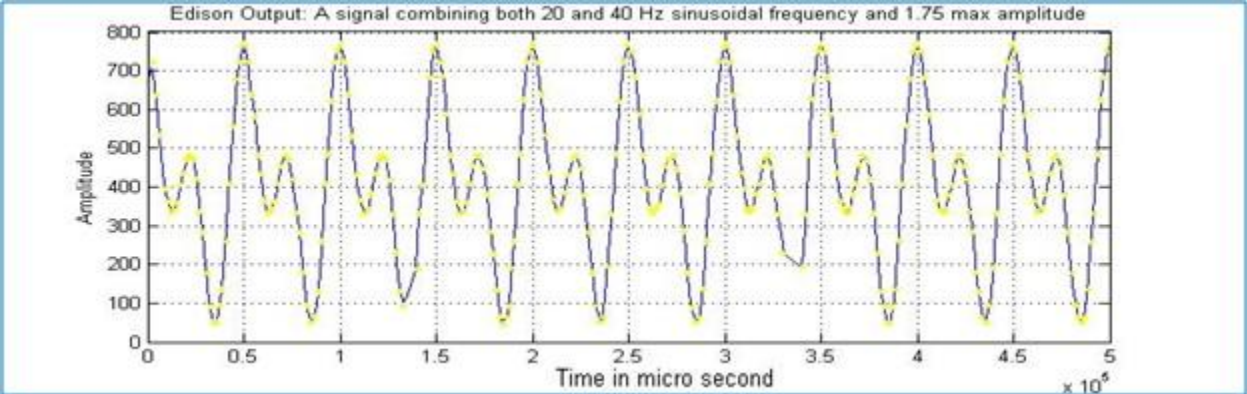


Fig.2.18:Output signal of combined frequency of saved data file after modifying in MATLAB

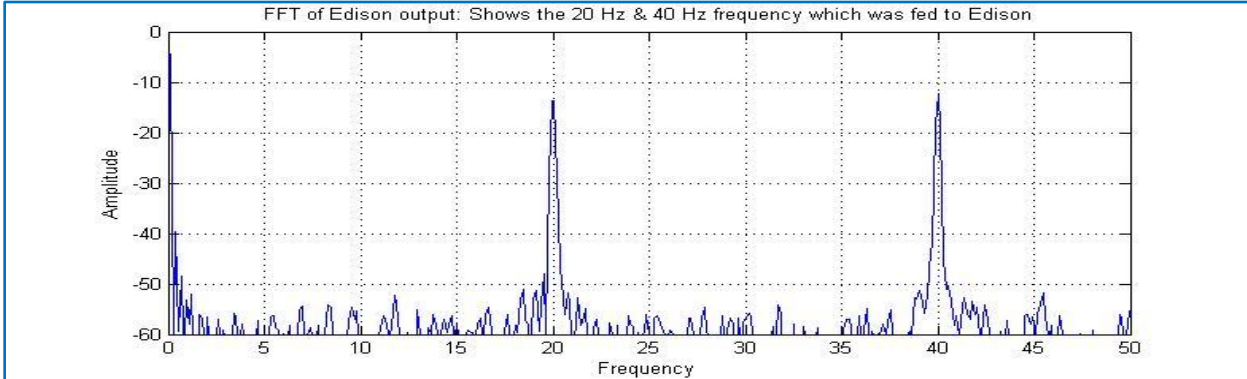


Fig.2.19:The FFT of the combined frequency signal



## 2.5. EDISON AND PYSERIAL:

**2.5.1. PySerial – A python library:** PySerial is a Python API module to access the serial port. It provides a uniform API across multiple operating systems, including Windows, Linux, and BSD. With the use of PySerial it is possible to plot real time graph at the computer end and at the same time one can save the data as a text file, as it was done in the previous section, for future use. This real time graph plotting could be handy for performance analysis of any model system.

**2.5.2. Experimental Procedure:** Following procedure needed to be followed to draw the dynamically updating plot.

- 1) The Quanser, Edison and computer back end were connected as it was done before during the data storing procedure.
- 2) Now a 10 Hz and 2 amplitude signal was fed to the Edison.
- 3) The Edison was accessed from putty and the following programme as shown in Fig.2.20 was run.

```
import serial
import time
from datetime import datetime
import mraa
serialport = '/dev/ttyMFD2'
ser = serial.Serial(serialport, 115200)
a0 = mraa.Aio(0)
a0.setBit(10)
now1=datetime.now()
print(now1)
try:
    while True:
        val = a0.read()
        print str(val)
        time.sleep(0.001)
except KeyboardInterrupt:
    print "done"
Quarc_Edison_pySerial.py
```

Fig.2.20: Python programme<sup>[7]</sup> for sending data in another host via PySerial

- 4) Then putty screen had to be closed before another python programme could be run from backend because keeping putty might cause an access error for a same COM port.

5) Now following python programme as shown in Fig.2.21 was executed from backend.

```
import serial
from time import sleep
import numpy as np
from matplotlib import pyplot
s=serial.Serial('COM9',115200)
#Initialize interactive mode
pyplot.ion()
pData=[0]*1000
fig = pyplot.figure()
pyplot.title('Real-time Edison reading')
ax1 = pyplot.axes()
l1, = pyplot.plot(pData)
pyplot.ylim([0,1000])
print "real-time plotting loop"

try:
    while True:
        sleep( 0.001)
        pData.append(float(s.readline()))
        pyplot.ylim([0,1000])
        del pData[0]
        l1.set_xdata([i for i in xrange(1000)])
        l1.set_ydata(pData) #update the data
        pyplot.show(block=False)#update the plot
        pyplot.pause(0.001)
except KeyboardInterrupt:
    pass

print "done" Edison_Live_graph.py
```

Fig.2.21: Python programme at another host for receiving data from Edison

**2.5.3. Results and Discussion:** The following graph was generated as shown in Fig.2.22 and it got updated during run time.

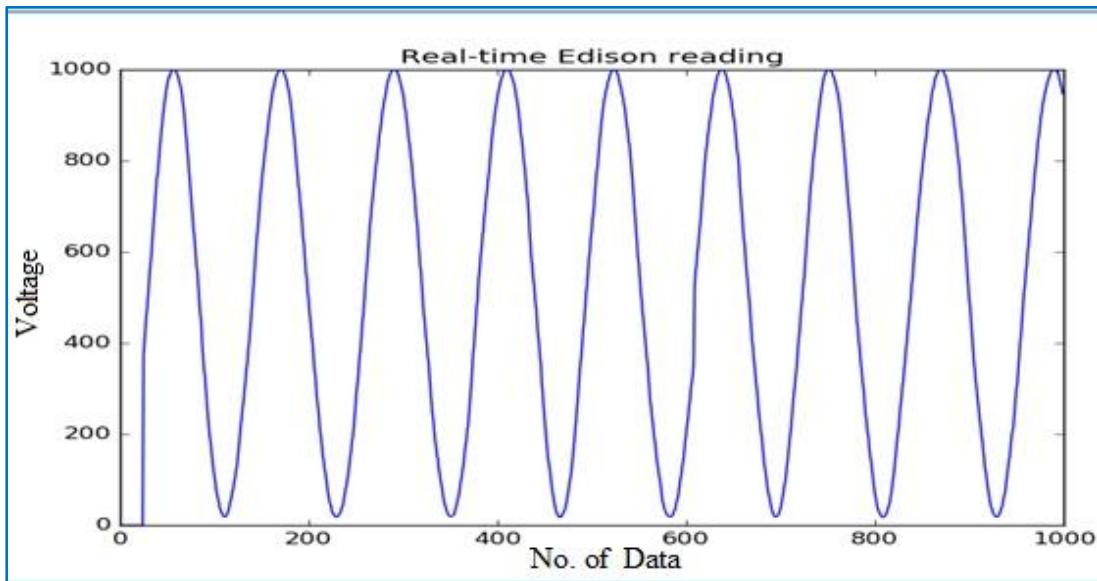


Fig.2.22: Real time graph using python

It is also possible to save the data file as a text file as it was done with the Edison. By using PySerial one can save the data file to avoid the complexity of transferring and the possibility of any data loss. Edison\_live\_graph.py was modified to Edison\_live\_data\_store.py to serve this purpose (Fig.2.23).

```

print "real-time plotting loop"
mI = 0
mList = []
for x in range (1,1001):
    sleep(0.001)
    pData.append(float(s.readline()))
    pyplot.ylim([0,1000])
    del pData[0]
    l1.set_xdata([i for i in xrange(1000)])
    l1.set_ydata(pData) #update the data
    pyplot.show(block=False)#update the plot
    pyplot.pause(0.001)
    val = s.readline()
    mI = datetime.now()
    cS = mI.second
    cMs = mI.microsecond
    time_sec = (str(cS)+"."+str(cMs))
    var = (str(cS)+"."+str(cMs)+" "+str(val))
    mList.append(var+" "+"")
    time.sleep(0.001)

fileOut = open('C:/Users/Prof. Arghya Nandi/Desktop/graph4mtextfile/Edison_data.txt','w')
fileOut.writelines(mList)

```

**Edison Live data store and real time plotting**

Fig.2.23: Programme at another host for receiving, saving and plotting of data from Edison

In order to retrieve the data file, the path mentioned in the programme was accessed as shown in Fig.2.24.

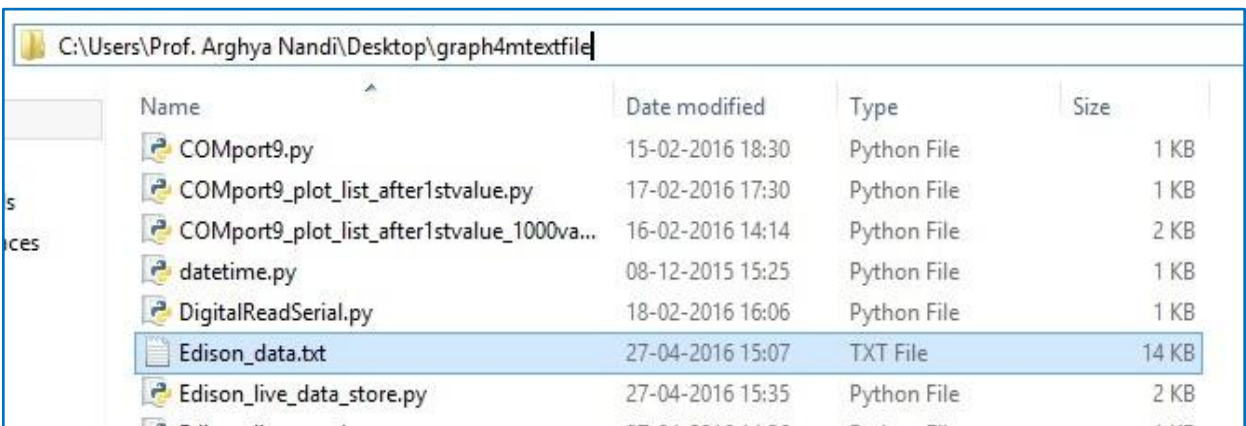


Fig.2.24: Location where data will be stored in another host

---

## Wireless vibration meter using Intel Edison

---

### 3.1. INTRODUCTION:

In the previous section, it has been discussed that it is possible to use Edison as data logger. However, it is also possible to use Edison as a wireless device to monitor and store vibration data. In this section, it will be detailed how Edison can be used as a wireless data logger. Also a vibration meter using Edison will be developed.

### 3.2. Using Bluetooth in Intel Edison:

**3.2.1. Enable Bluetooth in Edison:** It is known from chapter-2 that Intel Edison comes with Integrated Wi-Fi, Bluetooth 4.0 LE (BLE). By default, Intel Edison disables this feature. To enable BLE, the following command was entered in terminal/putty.

```
#rfkill unblock Bluetooth
```

Edison boots in a Bluetooth-disabled mode in order to save power. Every time Edison is powered on, it needs to issue the rfkill command to instantiate the Bluetooth again.<sup>[3.1]</sup>

**3.2.2. Pairing the Bluetooth Device:** This step needs to be done only once for each device; the pairing will remain forever so long as one does not delete the paired Bluetooth devices from the paired android phone or Intel Edison.

*bluetoothctl* command was entered on Edison terminal/putty. It showed the MAC address of Edison. The command *help* was entered to see all the Bluetooth configuration options available in *bluetoothctl*. On android phone, Bluetooth discovery of device was enabled by going to Settings → Bluetooth, after confirming that Bluetooth was set to **on**, and then the device was set to discoverable. Once the phone was made discoverable, Bluetooth scanning was started on Intel Edison to find the phone's MAC address.<sup>[3.1]</sup> This took a few seconds to search:

```
[bluetooth]# scan on Discovery started
[CHG]Controller98:4F:EE:02:F2:D2 Discovering:yes
[NEW] Device 74: 04: 2B:C7:BC: 2F(Lenovo A6000MAC address)
```

Once the MAC address of phone was obtained, scanning was disabled and the Edison was paired with android phone, after making sure that the phone was unlocked and active:

```
[bluetooth]# scan off          [CHG] Device A8: 66: 7F:AB:AE: 01 RSSI is nil
[CHG] Controller 98: 4F:EE: 02: F2: D2 Discovering: no Discovery stopped
```

Following command was typed in order to pair Edison with the phone.

```
[bluetooth]# pair74:04:2B:C7:BC:2F
Attempting to pair with 74:04:2B:C7:BC:2F
```

After the command was issued, a pairing message was popped up on the phone. Pairing was selected so that the two devices could connect. Upon making the connection, the status was indicated in bluetoothctl. Future connections would forget this process, Intel Edison was set to discoverable and permission to trust the Smartphone was too set for indefinitely.<sup>[3.1]</sup>

```
[bluetooth]# discoverable on
[bluetooth]# trust 74:04:2B:C7:BC:2F
```

Exit from bluetoothctl:[bluetooth]# quit

```
[DEL] Controller 98: 4F:EE: 02: F2: D2 edison [default]
```

### 3.3. Blinking a LED with command sending over Bluetooth:

**3.3.1. Checking the Bluetooth module:** Before a wireless vibration meter could be built, it was necessary to check that Bluetooth was working properly. In this section, a LED was blinked on and off using Edison's Bluetooth feature by sending command from an android handset.

#### 3.3.2. Experimental Procedure:

1) Below is the SPP-blink.py programme which was run at the background. The SPP-loopback.py was modified in two places to get our own SPP-blink.py. SPP-loopback.py was made available in Github by renowned scientist and technologist Pranav Mistry.<sup>[3.2]</sup>

```
try:
    from gi.repository import GObject
except ImportError:
    import gobject as GObject

# Initiate LED stuff
import mraa
ledPin = 13          # led pin number for mraa library
led = mraa.Gpio(ledPin)
led.dir(mraa.DIR_OUT)
```

Fig.3.1: Changes on SPP-loopback.py to get SPP-blink.py



Fig.3.1 and Fig.3.2 show the two major places where original programme was modified to get this version of SPP-blink.py. This was executed to initiate a Bluetooth connection.

```

server_sock = socket.fromfd(self.fd, socket.AF_UNIX, socket.SOCK_STREAM)
server_sock.setblocking(1)
server_sock.send("This is Edison's LED Control.\nSend 0 to turn
off and send 1 to turn on.\nPlease start:\n")

try:
    while True:
        data = server_sock.recv(1024)
        if data == '1':
            led.write(1)
            stringy = "LED is on\n"
        elif data == '0':
            led.write(0)
            stringy = "LED is off\n"
        else:
            stringy = "Unrecognized Command. Please send
either 0 or 1 to toggle the LED.\n"
            server_sock.send(stringy)
    except IOError:
        pass

server_sock.close()
print("all done")

```

Fig.3.2: Changes on SPP-loopback.py to get SPP-blink.py programme

2) Once Edison was paired with android Phone, it remembered this device for future connection too. After that Edison had to be connected with phone as shown in Fig.3.3.

3) After searching for all Bluetooth devices on android phone, when Edison showed up in the list, 'edison02' was clicked in the Bluetooth setting of the phone and it showed that Edison was connected(Fig.3.3).

4) BlueTerm+ app<sup>[3.3]</sup> was explored. Menu was opened and a connection was established with Edison. Below screen was appeared on the BlueTerm+ app(Fig.3.4).

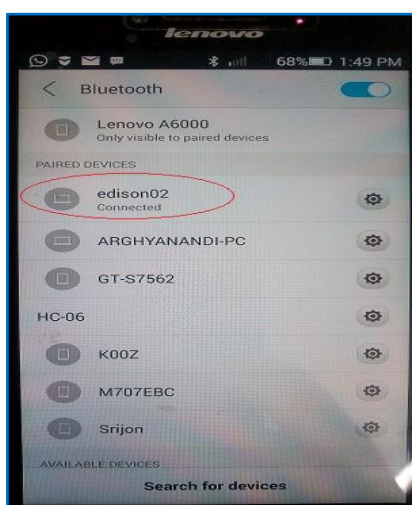


Fig.3.3: Searching Bluetooth device on phone



Fig.3.4: Connecting with edison02

5) The moment Bluetooth connection was established from the phone, the following message was appeared in the putty<sup>[3,4]</sup> terminal as shown in Fig.3.5.

```
root@edison02:~# python SPP-blink.py
NewConnection (/org/bluez/hci0/dev_74_04_2B_C7_BC_2F, 13)
```

Fig.3.5: Message on putty terminal as soon a Bluetooth connection was established

### 3.3.3. Results and Discussion:

When everything was done properly, upon clicking on ediosn02 from android phone, a message “This is Edison’s LED Control. Send 0 to turn off and send 1 to turn on LED” was appeared on android app. In the below figures whole experimental connection is shown. Positive or long leg of LED was connected to pin 13 and shorter or negative leg was connected to Ground of Edison. It was seen that when ‘1’ was pressed in android phone, LED was turned on as shown in Fig.3.6 and when ‘0’ was pressed in android phone, LED was turned off as shown in Fig.3.7.



Fig.3.6: Turning LED on by sending command '1' over Bluetooth.



Fig.3.7: Turning LED off by sending command '0' over Bluetooth.

### 3.4. Making a Bluetooth Vibration Meter:

It was known that voltage is proportional to displacement, so we made a ‘Vibration Meter’ which helped us to know the displacement and consequently the vibration. A sinusoidal signal from Quarc was sent to Edison and optimum (maximum and minimum) voltage values were captured over the Bluetooth after a fixed number of iteration. To do so we had copied the SPP-blink.py or SPP-loopback.py and edited some of its logic to get an output according to our requirement.

#### 3.4.1. Experimental Procedure:

1) Below is the change (Fig.3.8) which was implemented on SPP-MaxVal\_QUARC\_4.9V.py to get the maximum and minimum voltage from QUARC. To assure safety of Edison and to avoid error in the captured output, we had considered a 4.9V sinusoidal signal instead of 5V. This 4.9V signal was sent to Edison and from there it was accessed in android phone over Bluetooth using **BlueTerm+ app**.

Here is some change in logic. The part encircled in **Red** happened to be the most important point.

```
try:
    from gi.repository import GObject
except ImportError:
    import gobject as GObject

# Initiate analog read stuff
import time
import mraa
a0 = mraa.Aio(0)
a0.setBit(10)
```

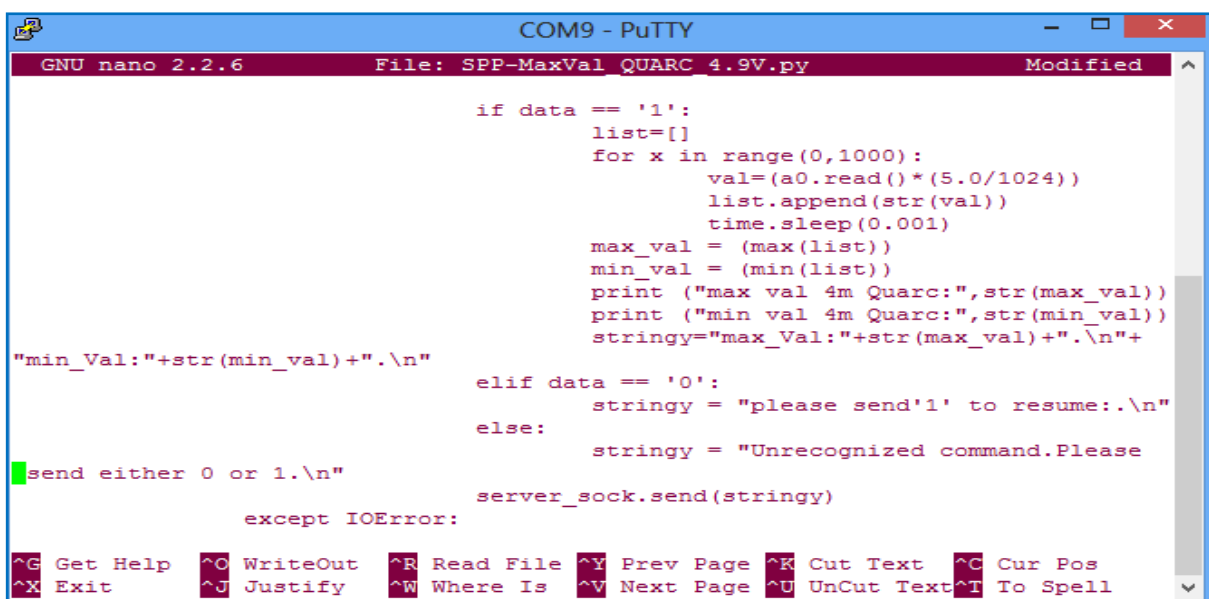


Fig.3.8: Changes on SPP-loopback.py to get SPP-MaxVal\_QUARC\_4.9V.py programme



2) At that time, as stated before, 'rfkill unblock bluetooth' was executed on putty terminal to turn the Bluetooth on. Then MaxVal\_QUARC\_4.9V.py programme was executed to establish a connection. Lastly, Bluetooth was turned on. Then BlueTerm+ app was opened to established a connection with Edison.

4) Thereafter, Feeding5volt\_2\_edison Simulink diagram was opened in MatLab and from signal generator, frequency of 20Hz, amplitude of 2.45, and a constant of 2.5 were set. Then the simulink model was built, connected to target and run as stated in chapter-2. This process had generated a sinusoidal frequency of 4.9V which was fed directly to Edison and from there, it was sent to android phone over Bluetooth.

**3.4.2. Results and Discussion:** Now after a moment, when BlueTerm+ app was connected with Edison, a message asking “Let us start” was appeared on app as shown in Fig.3.9. Once '1' was pressed, maximum and minimum values of Simulink signal were appeared on BlueTerm+ app as shown in Fig.3.10. When '0' was pressed, a message asking “please send '1' to resume” was appeared as per the design of the programme.



Fig.3.9: At the beginning of setting connection.

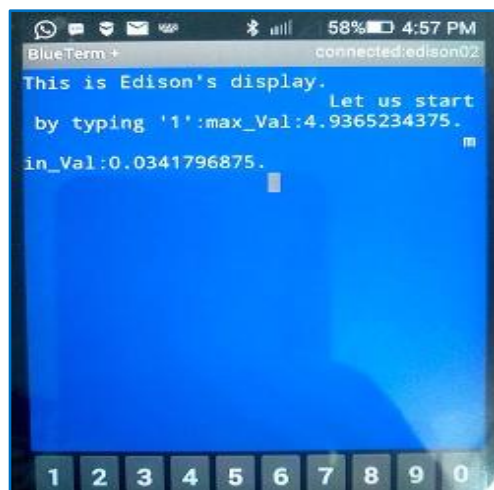


Fig.3.10: When command '1' was sent over bluetooth

Following messagees as in Fig.3.11 were appeared on the terminal. This is how Edison was used as a vibration meter.

```

root@edison02:~# python SPP-MaxVal_QUARC_4.9V.py
NewConnection(/org/bluez/hci0/dev_74_04_2B_C7_BC_2F, 13)
max val 4m Quarc: 0.009765625 When no voltage had fed to Edison
min val 4m Quarc: 0.0
max val 4m Quarc: 4.5458984375 when 4.5 volt signal was fed
min val 4m Quarc: 0.4833984375
max val 4m Quarc: 4.9365234375 when 4.9 volt signal was fed
min val 4m Quarc: 0.0341796875

```

Fig.3.11: Results displaying on serial terminal of putty

## Performance of MCP4921 as SPI based DAC

---

**4.1. Introduction:** It is known that Arduino Uno has 6 analog input pins of 10 bits. However, Uno does not have any analog output. It gives only Pulse Width Modulation (PWM) signals as output. The frequency of the PWM signal is approximately 490 Hz. Only the pins 5 and 6 have a frequency of approximately 980 Hz. If one intends to use Uno for the purpose of control then in many cases, it is advantageous to get an analog output signal. The 12 bit Digital to Analog Converter (DAC) MCP4921 can serve this purpose. This DAC exchanges data with Uno on its Serial Peripheral Interface (SPI).

This chapter depicts the experiment of generating a harmonic analog signal in a well-known sophisticated control card. Arduino Uno reads the signal through its analog input. Uno then converts the signal to a digital one which again gets converted to analog by MCP4921. The final analog signal is compared with the input signal using the same sophisticated control card. The frequency of the input signal is slowly increased to check the distortion of the output signal, if any.

**4.2. Definition of SPI:** Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device one wish to talk to.<sup>[4.1]</sup>

**4.3. Serial Data Transfer through SPI Bus:** An analog signal is fed to analog input of an Arduino from a Data Acquisition (DAQ) Device. The MCP4921DAC device is to be connected to the digital PWM output of the same Arduino Uno which will provide the necessary digital signal to MCP4921 to convert it to an analog signal.

An Arduino Uno device takes analog signal through its input terminal from a DAQ device and converts it into a digital signal. Again this digital signal is converted to an analog signal through the MCP4921 and ultimately goes back to analog DAQ device.

In short, DAQ will receive the same analog signal which was fed to an Arduino by it.

### 4.3.1. MCP4921DAC EXPERIMENT

#### 4.3.1.1. The goal of the experiment:

By passing a known analog signal through the device and getting back same signal after reconversion through it will help us to predict the performance of MCP4921 device for future application.

#### 4.3.1.2. Experimental procedure:

1. MCP4921 device was connected to a DAQ device through its terminal points. An Arduino was connected between MCP4921 and DAQ through its terminal points as shown in Fig.4.6.
  2. The required program (Fig.4.3) was uploaded into MCP4921 device.
  3. The DAQ device was connected to a computer with Windows operating system, as outlined in DAQ manual. In our case, it was a Q8\_USB DAQ device.
  4. A work space was created in MATLAB and MCP4921A2D.slx MAT file was built.
  5. The same MAT file was executed for minimum of 5 seconds.
  6. The plot\_volt\_graph.m MAT script was executed to generate the MAT figure (Fig.4.1).
  7. First at clock speed 4, the response graphs of MCP4921 @10Hz and 20Hz were taken.
  8. Then at clock speed 2, the response graphs of MCP4921 @20Hz and 40Hz were taken.
- These graphs are shown in Fig.4.9, 4.10, 4.11, 4.12.

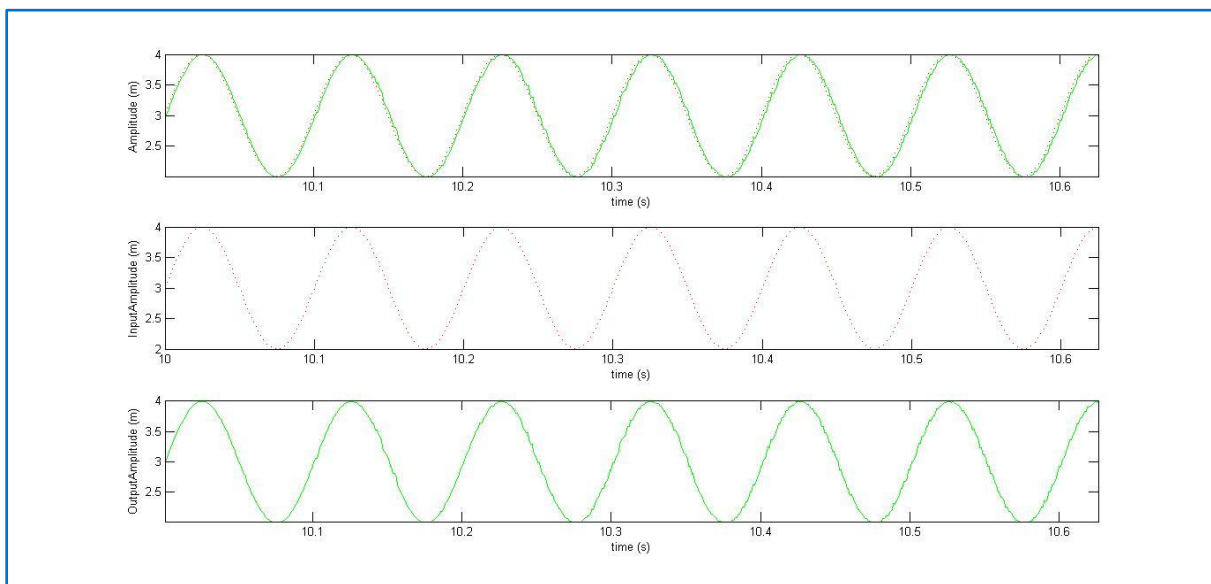


Fig.4.1: The input and output response

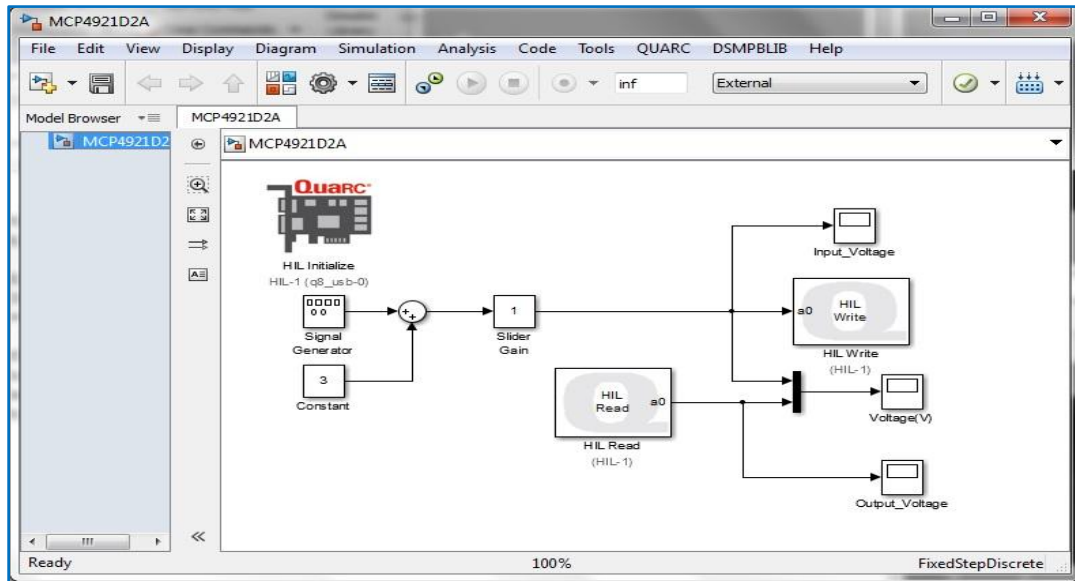


Fig.4.2: Simulink model

```

mcp4921a
#include <SPI.h>
const int chipSelectPin = 10;
const int analogInPin = A0;
int sensorValue = 0;
int outputValue = 0;
byte outputValueByte0 = 0;
byte outputValueByte1 = 0;

void setup()
{
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  SPI.setClockDivider(SPI_CLOCK_DIV4); // SPI_CLOCK_DIV2
  pinMode(chipSelectPin, OUTPUT);
  digitalWrite(chipSelectPin, HIGH);
}

void loop()
{
  sensorValue = analogRead(analogInPin);
  outputValue = map(sensorValue, 0, 1023, 0, 4095);
  outputValueByte0 = byte(outputValue);
  outputValue = outputValue >> 8;
  outputValueByte1 = byte(outputValue | 0b00110000);

  digitalWrite(chipSelectPin, LOW);
  SPI.transfer(outputValueByte1);
  SPI.transfer(outputValueByte0);
  digitalWrite(chipSelectPin, HIGH);
}

```

Fig.4.3: The Arduino programme which was uploaded in MCP4921 device.

The performance of MCP4921 was noted down in @clock speed 4 and 2, and the output graph was compared to get a clear idea of how MCP4921 might behave when it would be implemented in a specific application.

9) All the required connections are shown in Fig.4.4 which is a breadboard view. Fig.4.5 shows a schematic view in fritzing. Fig.4.6 shows experimental setup. Fig.4.7 shows DAQ device connected to Arduino and MCP4921.

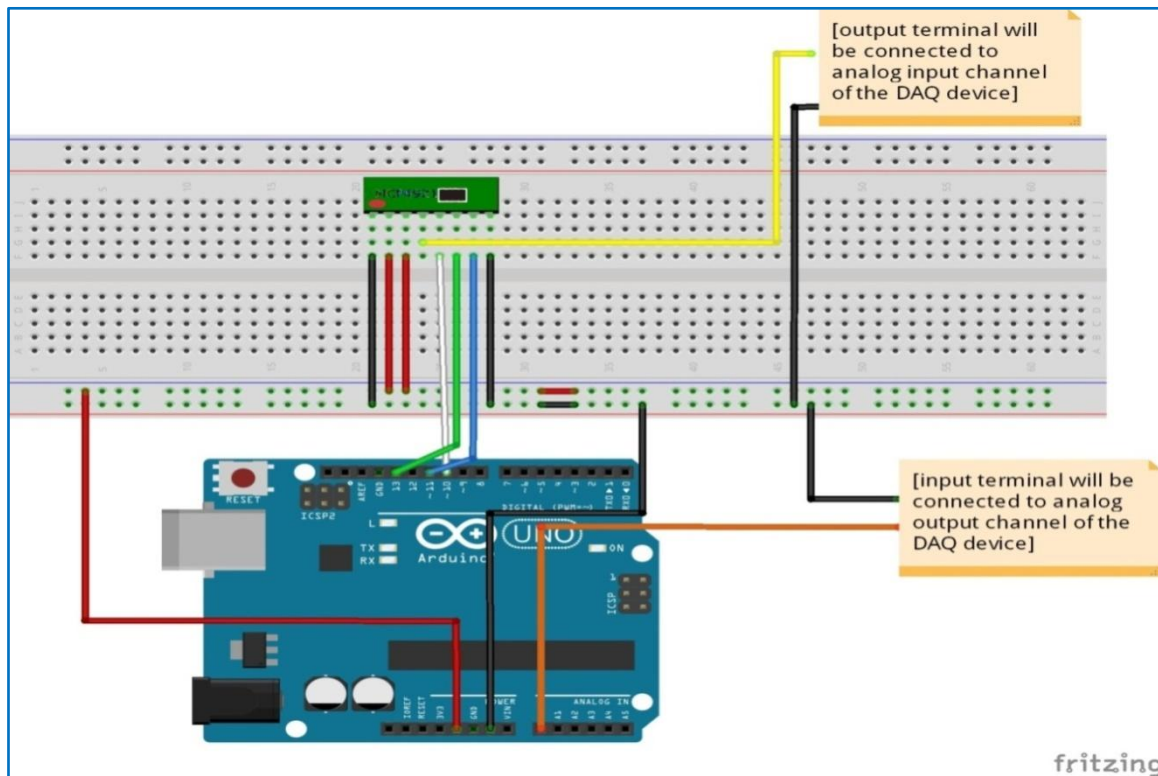


Fig.4.4: The breadboard view of model<sup>[4.2]</sup> with jumpers connected to requisite pin

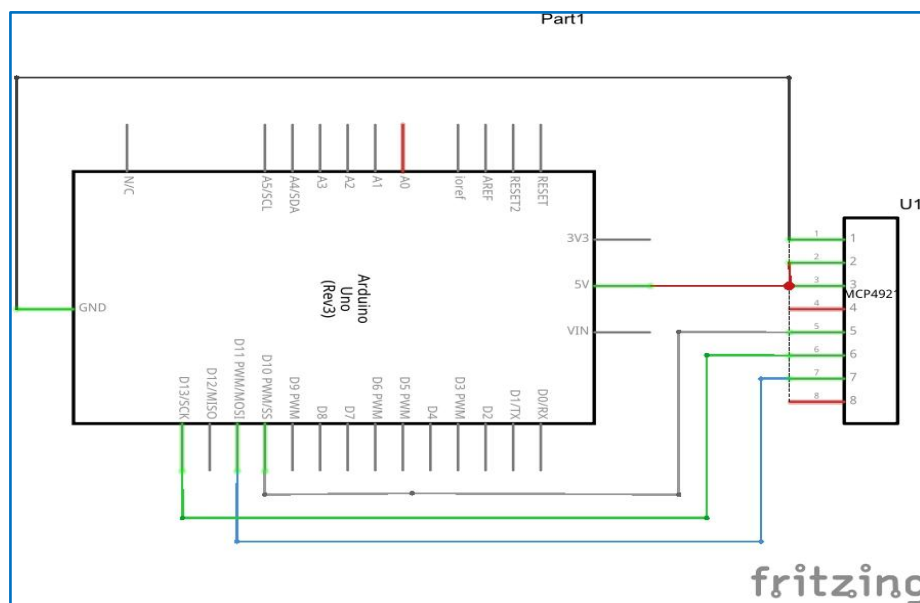


Fig.4.5: The schematic view<sup>[4.2]</sup> of our model which is shown in Fig.4.4



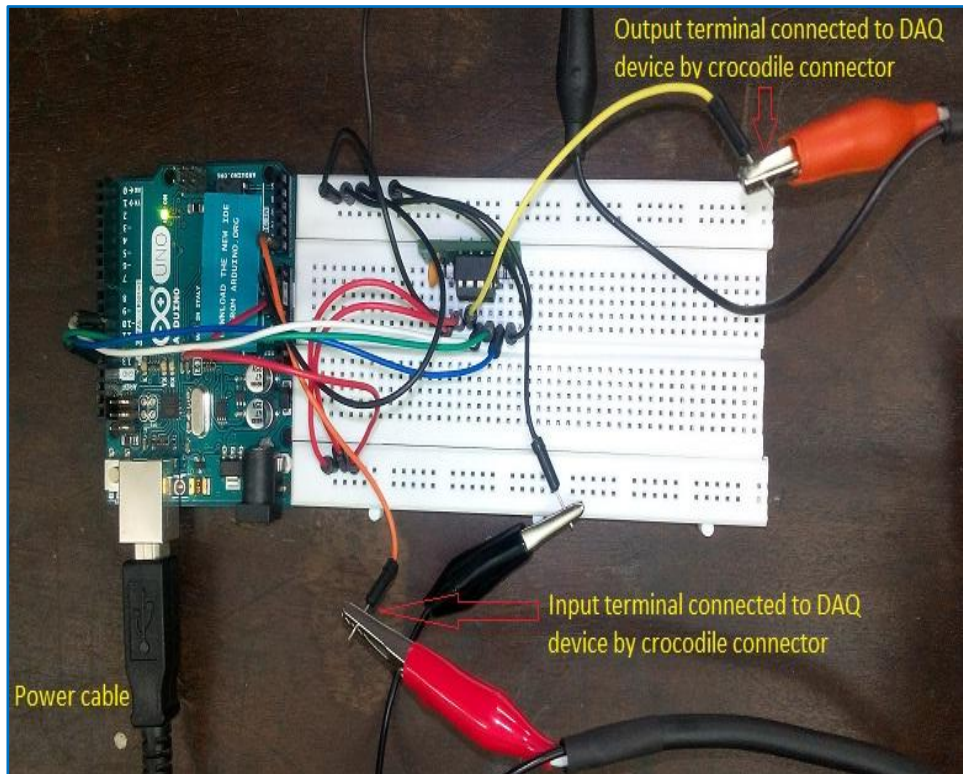


Fig.4.6: MCP4921 connected to input and output terminals via two crocodile connector



Fig.4.7: The DAQ device, connected to two terminals

### 4.3.1.3. Performance Test of MCP4921 Analog to Digital converter:

- A. The performance test of MCP4921 gave same signal as shown in figure 4.8.
- B. The middle one among the graphs of Fig.4.8 represents the analog input signal which was fed to Arduino.
- C. The bottom one among the graphs of Fig.4.8 represents the analog output signal from the MCP4921 device.
- D. The top graph in Fig.4.8 represents the both signal on the same plot.
- E. From the upper graph it is quite clear that @10 Hz the output deviated significantly from the input at clock speed 4.
- F. When we changed the clock speed to 2, the output deviated from input @20 Hz. All those plots are shown in Fig.4.9, 4.10, 4.11, 4.12.

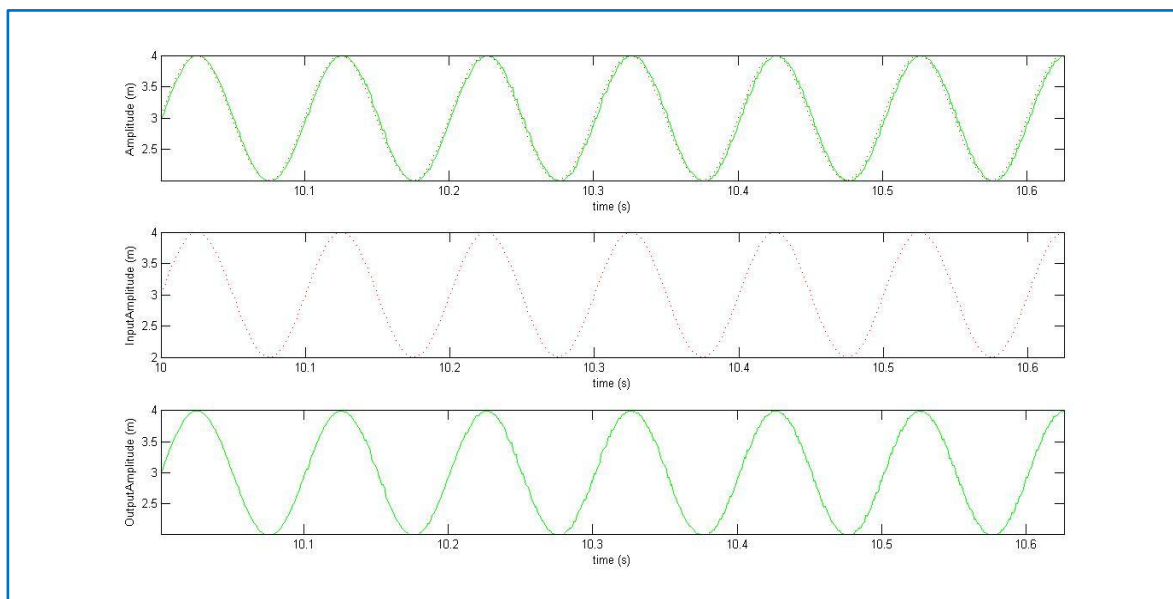


Fig.4.8: Input Output Signal

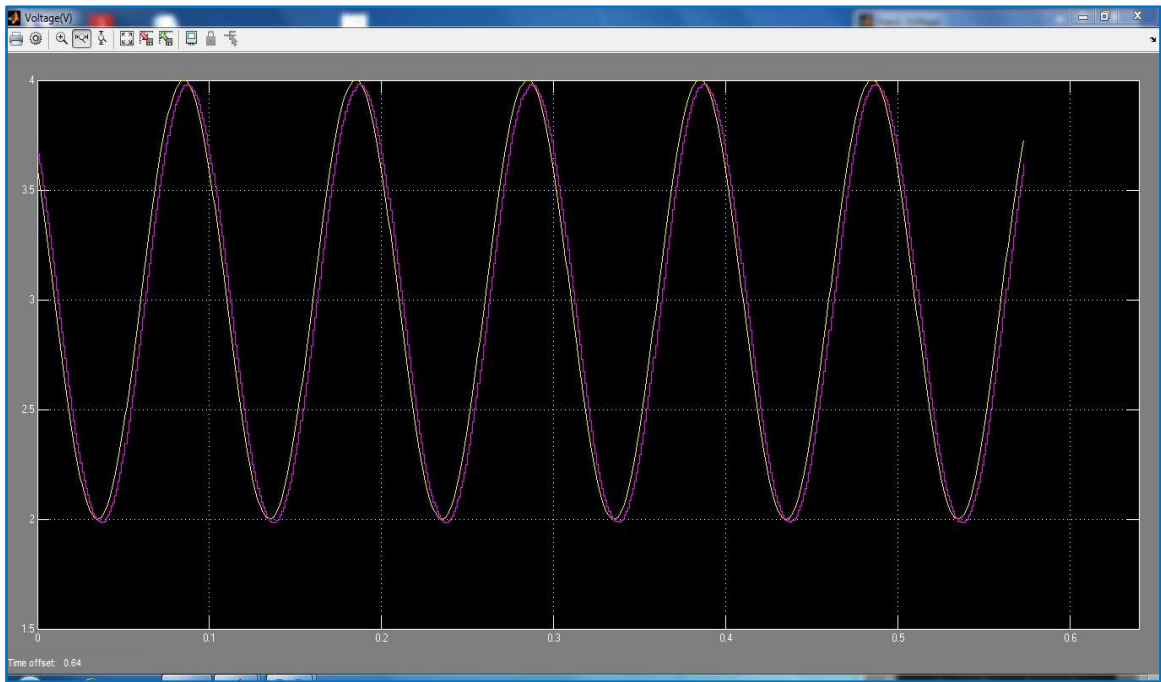


Fig.4.9:10Hz&DIV4

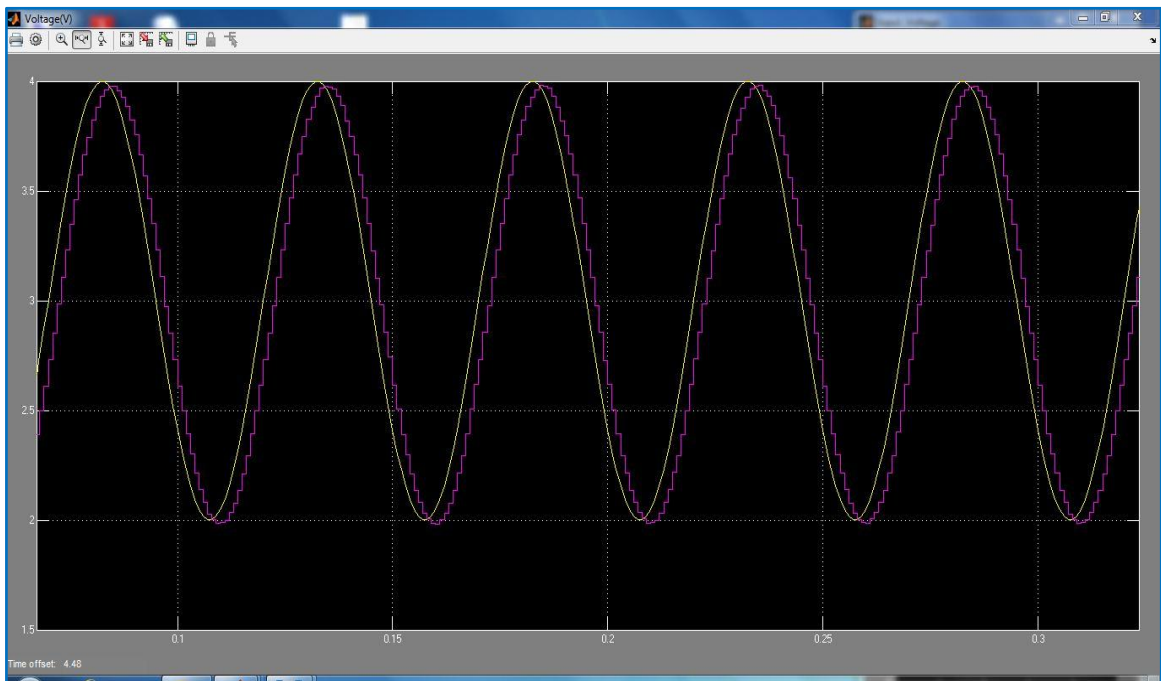


Fig.4.10:20Hz&DIV4



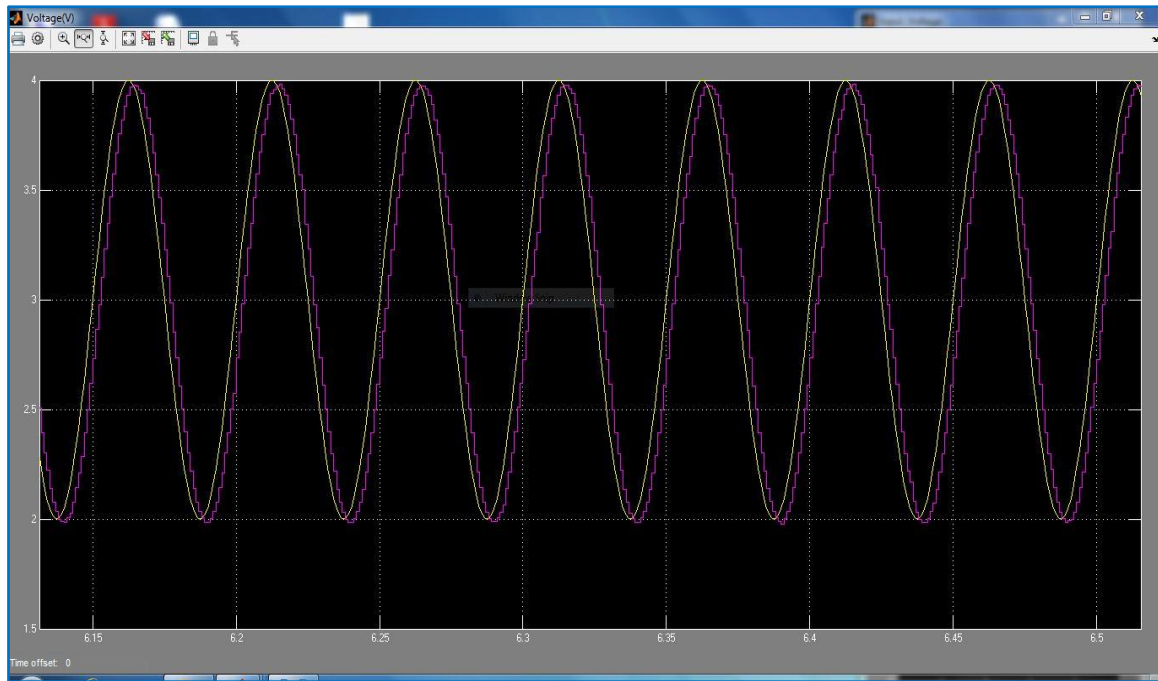


Fig.4.11:20Hz&DIV2

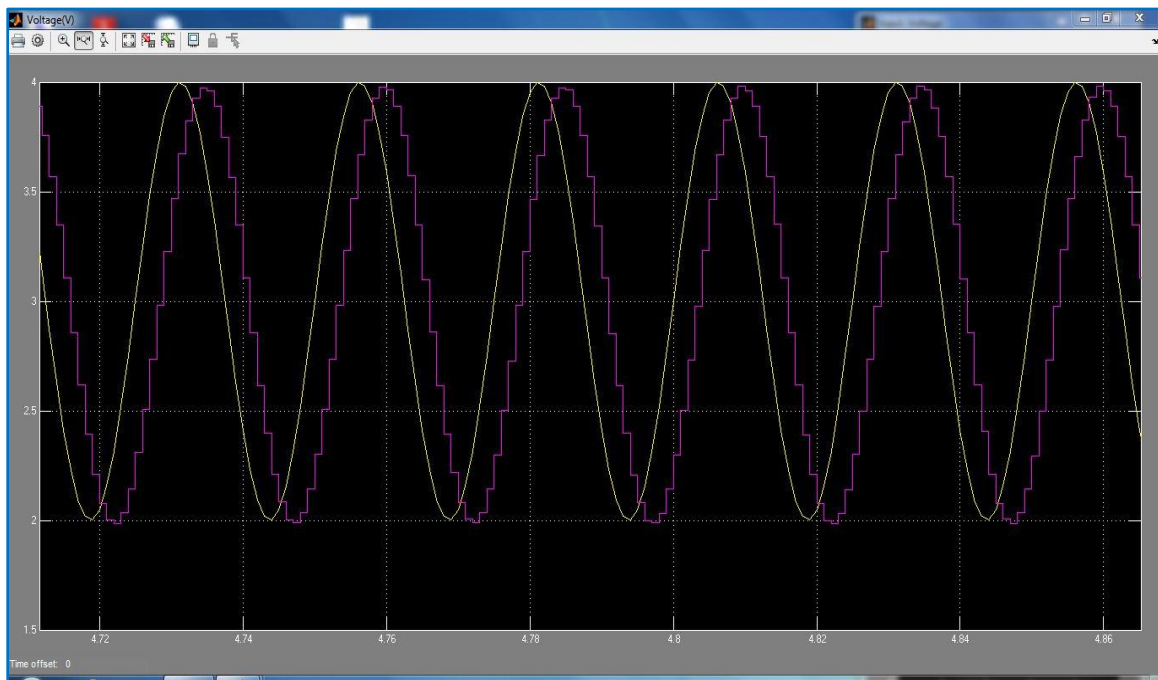


Fig.4.12:40Hz&DIV2

#### **4.3.1.4. CONCLUSION:**

We presented how an analog signal was retrieved from Arduino Uno with the use of MCP4921 which gave an analog shape to the digital output of Uno. We compared the final analog signal, thus produced, with the input signal using the control card. The comparison brought forth some phase lag. The graphs exhibited in the figures shown in the above section clearly depict the lag in terms of response time associated with our device. We went on increasing the frequency from 1 Hz and checked the graph at every instant, and we found that the phase separation between input and response was clearly visible almost @10 Hz for clock speed 4 and @20 Hz for clock speed 2. Thus we can conclude that though it was possible to obtain the same analog signal with the same magnitude, there was some phase lag associated with the signal obtained after reconversion from MCP4921 as clock speed and frequency were gradually increased.

## **Displacement and Strain Data Acquisition from a Vibrating Beam.**

---

### **5.1. Introduction:**

The objective of the present chapter is to evaluate the performance of low cost sensors and ATMEGA328P based data acquisition system in a vibrating test set-up. A simple cantilever beam was used for this purpose with an eccentric mass actuator. The eccentric mass generates a harmonic force on the beam with a frequency proportional to the speed of rotation. Amplitude resonance can be detected simply by varying the speed of the motor. Two types of sensors were used for measurement of vibration: a non-contact Hall sensor in order to detect the displacement of the beam close to its tip and a strain gage sensor to pick up the dynamic strain. Both displacement and strain signal were monitored using a display unit. The most important part of this work was to develop a non-contact strain sensor. The maximum and minimum strain could then be acquired in an Android phone with Bluetooth.

### **5.2. Description of the experimental set-up:**

As mentioned before the total experimental set-up consisted of a cantilever beam, an actuation system and two different sensing arrangements. The components of the actuation system and the sensing arrangements are as follows:-

Actuation system – motor with eccentric mass, motor speed controller, motor speed measurement system, LCD Display

Sensing arrangement –

1. Hall sensor, Arduino UNO, LCD Display
2. Half bridge strain gage, INA125P based amplifier, ATMEGA328P with HC04 Bluetooth module

#### **5.2.1. Cantilever Beam:**

A cantilever beam made of aluminium of length 46.5 cm, width 38.18 mm and thickness 6.02 mm was used as a vibrating structural element. At the free end a clamp was attached as a holder of a dc motor.

A permanent magnet was placed near the free end. Thus a magnetic field was generated which was changing with vibration and was sensed by a Hall sensor.



Fig.5.1: Cantilever beam and support frame

### 5.2.2. Actuation system:

An actuation system consists of motor with eccentric mass, motor speed controller, motor speed measurement system, LCD Display. From motor speed controller a modulated pulse of 12V went into the motor (1000 rpm). By rotating a potentiometer knob, the speed of the motor was controlled and a particular speed was reached where amplitude of cantilever beam attained a maximum value; this was the critical speed for beam material. The critical speed was directly measured by an oscilloscope. The motor speed and input voltage were seen on a LCD Display integrated with the speed controller. Fig.5.2 shows a glimpse of actuating system. Speed controller is not shown here. It will be discussed later.



Fig.5.2: Actuation system for inducing vibration

### 5.2.3. Sensing arrangement:

Sensing elements are like nerve in the human system. In this low-cost experiment, there were two major sensing arrangements which acted as receptors to collect data from vibrating beam.

#### 5.2.3.1. Sensors for Measuring Displacement Data:

Hall sensor, Arduino UNO, LCD Display together acted as a unit for sensing displacement data from a fluctuating magnetic field of the cantilever beam. Then the unit sent those data to an Arduino and consequently displayed maximum and minimum value of those data on a LCD. The below figure 5.3 shows a permanent south pole magnet which was used in our experiment. This was attached to the cantilever beam, which when vibrated, induced a magnetic field due to the moving magnet. Then a Hall sensor as shown in Fig 5.4 was placed in that magnetic field with the help of a piece of bread board under the magnet in a suitable position. It gave an output as voltage which was directly proportional to the strength of the magnetic field. The displacement data can be read on Arduino and LCD unit as shown in Fig. 5.5



Fig.5.3: Permanent South Pole magnet



Fig.5.4 Hall sensor used in experiment



Fig.5.5: Amplitude voltage of beam

### 5.2.3.1.1. HALL Displacement sensor:

Magnetic sensors are solid state devices that are becoming more and more popular because they can be used in many different types of application such as sensing position, velocity or directional movement. They are also popular choice of sensor for the electronics designer due to their non-contact wear free operation, low maintenance, and robust design. And, sealed Hall Effect devices are also immune to vibration, dust and water. [5.1]

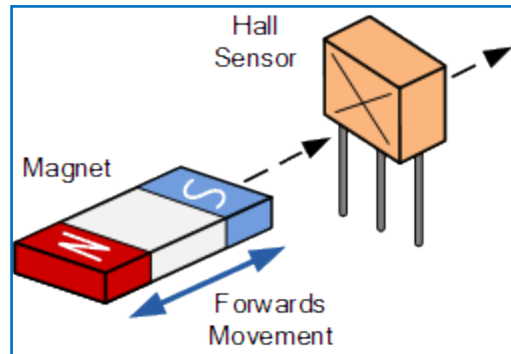


Fig.5.6: Side way Hall sensor

**Hall Effect Sensors** are devices which are activated by an external magnetic field. We know that a magnetic field has two important characteristics flux density, (B) and polarity (North and South Poles). The output signal from a Hall Effect sensor is the function of the magnetic field density around the device. When the magnetic flux density around the sensor exceeds a certain pre-set threshold, the sensor detects it and generates an output voltage called the **Hall Voltage,  $V_H$** . Consider the diagram below. [5.1]

### Hall Effect Sensor Principals

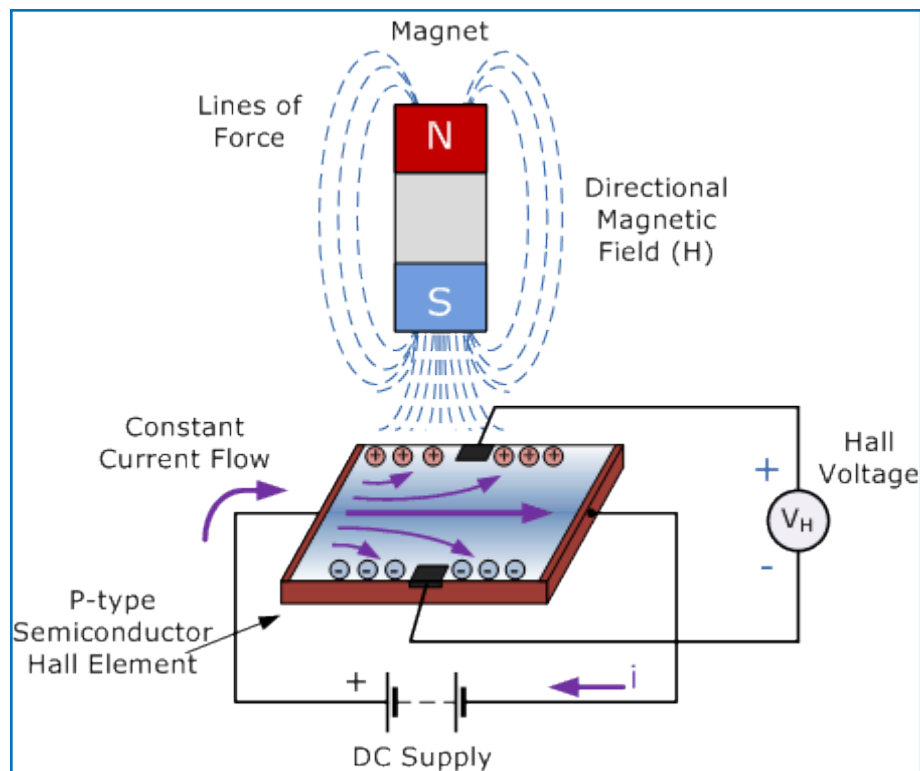


Fig.5.7: Working principle of Hall Sensor

**Hall Effect Sensors**<sup>[5.1]</sup> comprise basically of a thin piece of rectangular p-type semiconductor material such as gallium arsenide (GaAs), indium antimonide (InSb) or indium arsenide (InAs) passing a continuous current through itself. When the device is placed within a magnetic field, the magnetic flux lines exert a force on the semiconductor material which deflects the charge carriers, electrons and holes, to either side of the semiconductor slab. This movement of charge carriers is a result of the magnetic force they experience passing through the semiconductor material.

As these electrons and holes move sideward, a potential difference is produced between the two sides of the semiconductor material by the build-up of these charge carriers. So the movement of electrons through the semiconductor material is affected by the presence of an external magnetic field which is at right angles to it and this effect is greater in a flat rectangular shaped material.

The effect of generating a measurable voltage by using a magnetic field is called the **Hall Effect**, named for Edwin Hall who discovered it back in the 1870s with the basic physical principle underlying the Hall Effect being Lorentz force. To generate a potential difference across the device the magnetic flux lines must be perpendicular ( $90^\circ$ ) to the flow of current and must be of the correct polarity, generally a south pole.

The Hall Effect provides information regarding the type of magnetic pole and magnitude of the magnetic field. For example, a south pole would cause the device to produce a voltage output while a north pole would have no effect. Generally, Hall Effect sensors and switches are designed to be in the “OFF” (open circuit condition) state when there is no magnetic field present. They are turned “ON” (closed circuit condition) only when subjected to a magnetic field of sufficient strength and polarity.

The output voltage, called the Hall voltage ( $V_H$ ) of the basic Hall Element, is directly proportional to the strength of the magnetic field passing through the semiconductor material (output  $\propto H$ ). This output voltage can be quite small, only a few micro-volts even when subjected to strong magnetic fields. Therefore, the most commercially available Hall Effect devices are manufactured with built-in DC amplifiers, logic switching circuits and voltage regulators to improve the sensitivity of sensors; hysteresis and output voltage. This also allows the Hall Effect sensor to operate over a wider range of power supplies and magnetic field conditions.<sup>[5.1]</sup>



There are two basic types of digital Hall Effect sensor, **Bipolar** and **Unipolar**. Bipolar sensors require a positive magnetic field (South Pole) to operate them and a negative field (North Pole) to release them while Unipolar sensors require only a single magnetic south pole to both operate and release them as they move in and out of the magnetic field.

**Hall Effect Sensors** <sup>[5.1]</sup> are available with either linear or digital outputs. The output signal for linear (analogue) sensors is taken directly from the output of the operational amplifier with the output voltage being directly proportional to the magnetic field passing through the Hall sensor. This output Hall voltage is given as:

$$V_H = R_H \left( \frac{I}{t} \times B \right)$$

- $V_H$  is the Hall Voltage in volts
- $R_H$  is the Hall Effect co-efficient
- $I$  is the current flow through the sensor in amps
- $t$  is the thickness of the sensor in mm
- $B$  is the Magnetic Flux density in Teslas Density

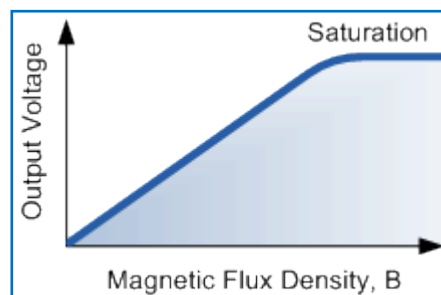


Fig.5.8: Output voltage proportional to Magnetic Flux



### 5.2.3.2. Sensors for Measuring Strain Data:

Half bridge strain gage, shown in left side of Fig 5.9, was tightly bonded to two surfaces of beam and the latter half was on the laboratory equipped electronic board along with the INA125P based amplifier, ATMEGA328P with HC04 Bluetooth module. All these together acting as a unit, sensed the strain data from vibrating beam, and then sent the data to oscilloscope (Fig.5.10) and android phone via micro-controller based electronic device and HC06 BT device respectively.

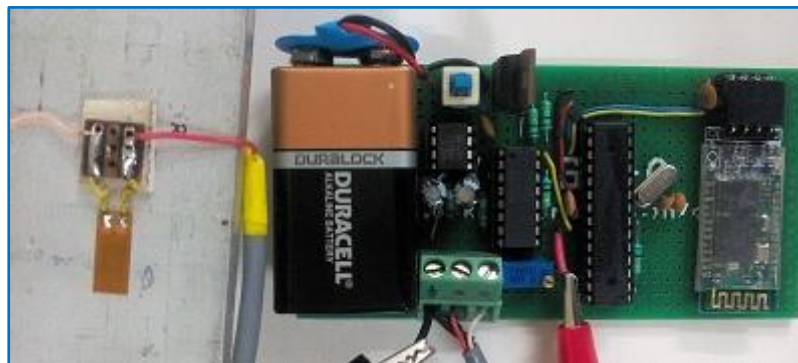


Fig.5.9: Strain gage and strain measuring instrument



Fig.5.10: Oscilloscope to see voltage graph

**5.2.3.2.1. STRAIN GAGE:** A strain gage is a strain measuring device, tightly bonded to a strain measuring object so that the sensing element (metallic resistive foil) may elongate or contract according to the strain borne by the measuring object.

**Principle of Strain Gage:** When bearing mechanical elongation or contraction, most metals undergo a change in electric resistance. The strain gage applies this principle to strain measurement through the resistance change. Generally, the sensing element of the strain gage is made of a copper-nickel alloy foil. The alloy foil has a rate of resistance change proportional to strain with a certain constant.

Let's express the principle as follows:

$$\Delta R/R = K \cdot \epsilon$$

where, R: Original resistance of strain gage,  $\Omega$  (ohm)

$\Delta R$ : Elongation- or contraction-initiated resistance change,  $\Omega$  (ohm)

K: Proportional constant (called gage factor)

$\epsilon$ : Strain

**Structure of Strain Gage:** There are many types of strain gages. Among them, a universal strain gage has a structure such that a grid-shaped sensing element of thin metallic resistive foil (3 to 6 $\mu\text{m}$  thick) is put on a base of thin plastic film (15 to 16 $\mu\text{m}$  thick) and is laminated with a thin film.

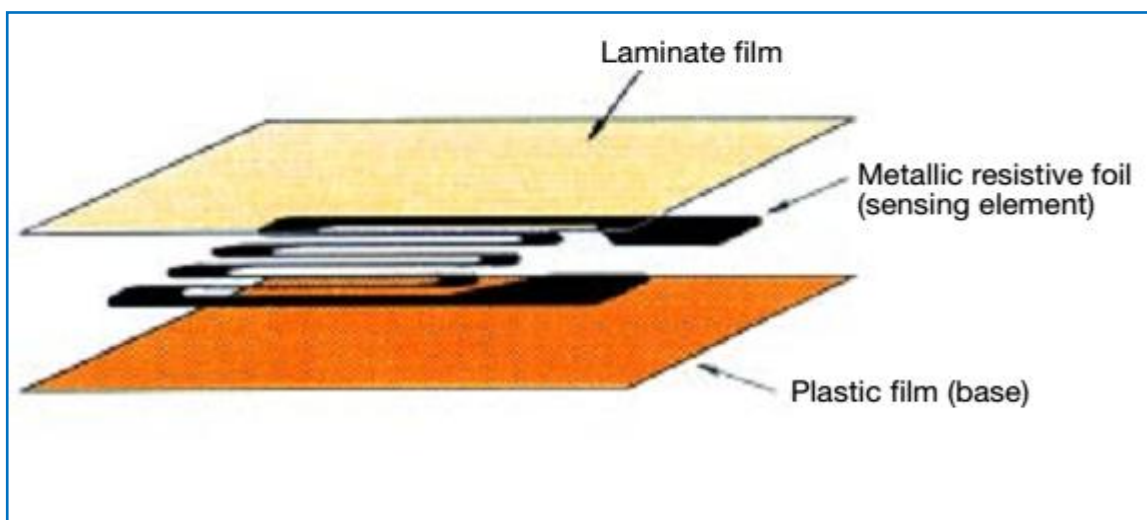


Fig.5.11: Structure of Strain Gage

### Wheatstone bridge:

The Wheatstone bridge is an electric circuit suitable for detection of minute resistance changes. It is, therefore, used to measure resistance changes of a strain gage. The bridge is configured by combining four resistors as shown in Fig. 5.12

Suppose:  $R_1 = R_2 = R_3 = R_4$ , or  $R_1 \times R_3 = R_2 \times R_4$

Then, whatever voltage is applied to the input, the output, e, is zero.

Such a bridge status is called “balanced”. When the bridge loses the balance, it outputs a voltage corresponding to the resistance change.

We employed a 2-gage system as shown in Fig.5.13.

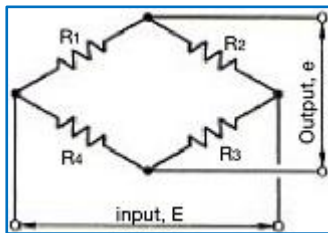


Fig.5.12: Wheatstone Bridge

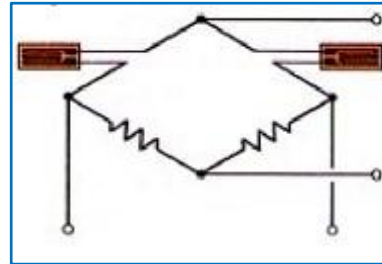


Fig.5.13: WB with two Gage system

**Output voltage of 2-gage system:** Two sides among the four initiate resistance change. Thus, the 2-gage system in the case of Fig.5.13 provides the following output voltage:

$$e = \frac{1}{4} \left( \frac{\Delta R_1}{R_1} - \frac{\Delta R_2}{R_2} \right) E$$

or,

$$e = \frac{1}{4} K (\epsilon_1 - \epsilon_2) E$$

These two gages are connected to adjacent or opposite sides of the bridge (Fig5.14), and the bending or tensile strain can be measured separately. That is, gage 1 senses the tensile (plus) strain and gage 2 senses the compressive (minus) strain. The absolute strain value is the same irrespective of polarities, provided that the two gages are at the same distance from the end of the cantilever.<sup>[5.2]</sup>

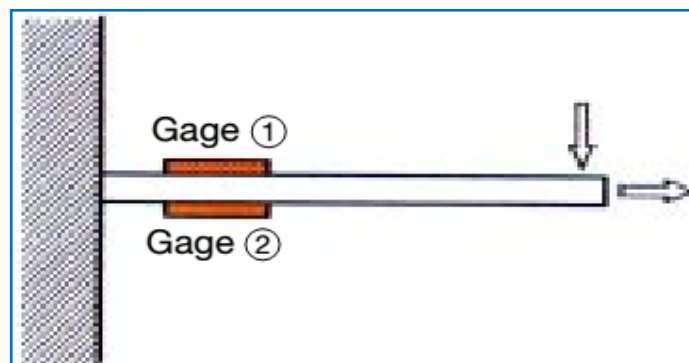


Fig.5.14: Gages are connected to the opposite sides of the beam

**5.2.3.2.2. INA125P as a Strain Amplifier:** The INA125 is a low power, high accuracy instrumentation amplifier with a precision voltage reference. It provides complete bridge excitation and precision differential-input amplification on a single integrated circuit. A single external resistor sets any gain from 4 to 10,000. The INA125 is laser-trimmed for low offset voltage (250 $\mu$ V), low offset drift (2 $\mu$ V/ $^{\circ}$ C), and high common-mode rejection (100dB at G=100). It operates on single (+2.7V to +36V) or dual ( $\pm$ 1.35V to  $\pm$ 18V) supplies. The voltage reference is externally adjustable with pin-selectable voltages of 2.5V, 5V, or 10V, allowing to use it with a variety of transducers. The reference voltage is accurate to  $\pm$ 0.5% (max) with  $\pm$ 35ppm/ $^{\circ}$ C drift (max). Sleep mode allows shutdown and duty cycle operation to save power. The INA125 is available in 16-pin plastic DIP and SO-16 surface-mount packages, and is specified for the  $-40^{\circ}$ C to  $+85^{\circ}$ C industrial temperature range Fig.5.15.<sup>[5.3]</sup>

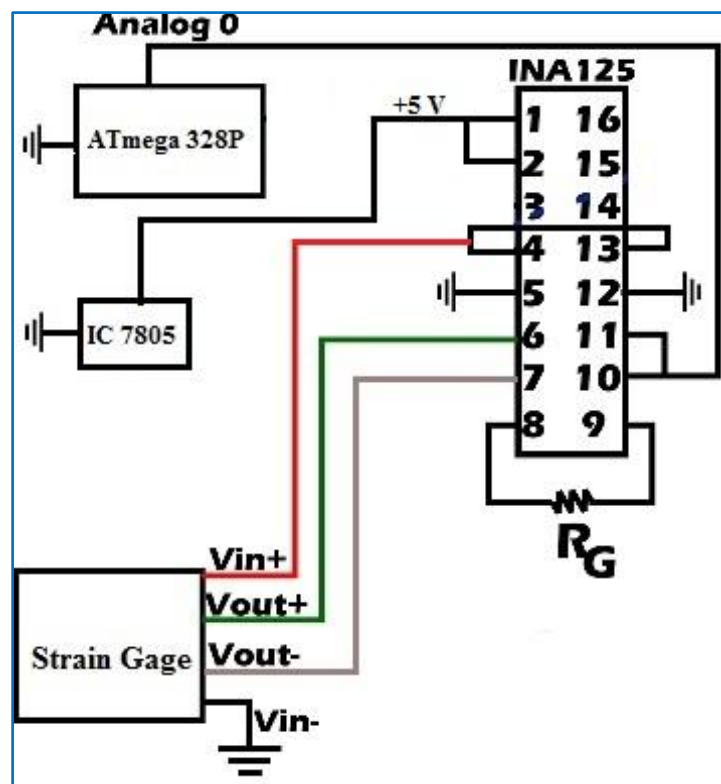


Fig.5.15: Block diagram<sup>[5.4]</sup> of INA125P connected with other peripheral

### 5.2.3.2.3. Technical Specification of ATmega328P:

The Atmel 8-bit AVR RISC-based microcontroller combines 32 kB ISP flash memory with read-while-write capabilities, 1 kB EEPROM, 2 kB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and

QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz.<sup>[5.5]</sup>

**Key Parameters:**

Parameter	Value
CPU type	8-bit AVR
Performance	20 MIPS at 20 MHz[2]
Flash memory	32 kB
SRAM	2 kB
EEPROM	1 kB
Pin count	28-pin PDIP, MLF, 32-pin TQFP, MLF[
Maximum operating frequency	20 MHz
Number of touch channels	16
Hardware QTouch Acquisition	No
Maximum I/O pins	26
External interrupts	24
USB Interface, USB Speed	No, No

Table-5.1: Technical Specification of ATmega328P

**5. 2.3.2.4. ATmega 328 Pin Mapping:**

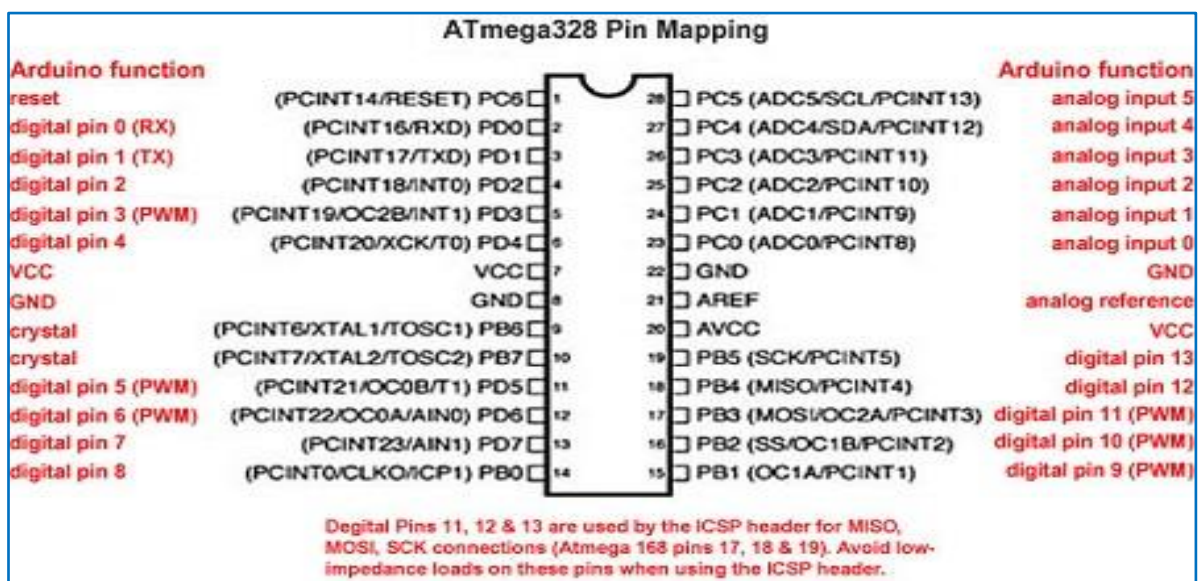


Fig.5.16: ATmega 328 Pin Mapping<sup>[5.6]</sup>



### 5.3. Complete Experimental setup:

Fig.5.16 shows how **Complete Experimental setup** looked like. First, 12V DC power was supplied to the device '4'. Then it started the motor which, in turn, rotated the actuator (7) and as a result beam (6) started to vibrate. By controlling voltage of a potentiometer on device '4', this vibration was controlled. When beam vibrated, the magnet too did and it resulted in a magnetic field. A Hall sensor(8) sensed this field and gave an output to an Arduino-LCD display unit(3), which then sent it over BT. Also a strain gage(9), fitted on the beam, generated a signal which was seen on an oscilloscope(1). Oscilloscope helped to determine the natural frequency of the beam. The same strain gage was used to measure the strain with the help of device '2' and strain values were sent over BT and read on BlueTerm+ android app. This was how the whole experiment was conducted.

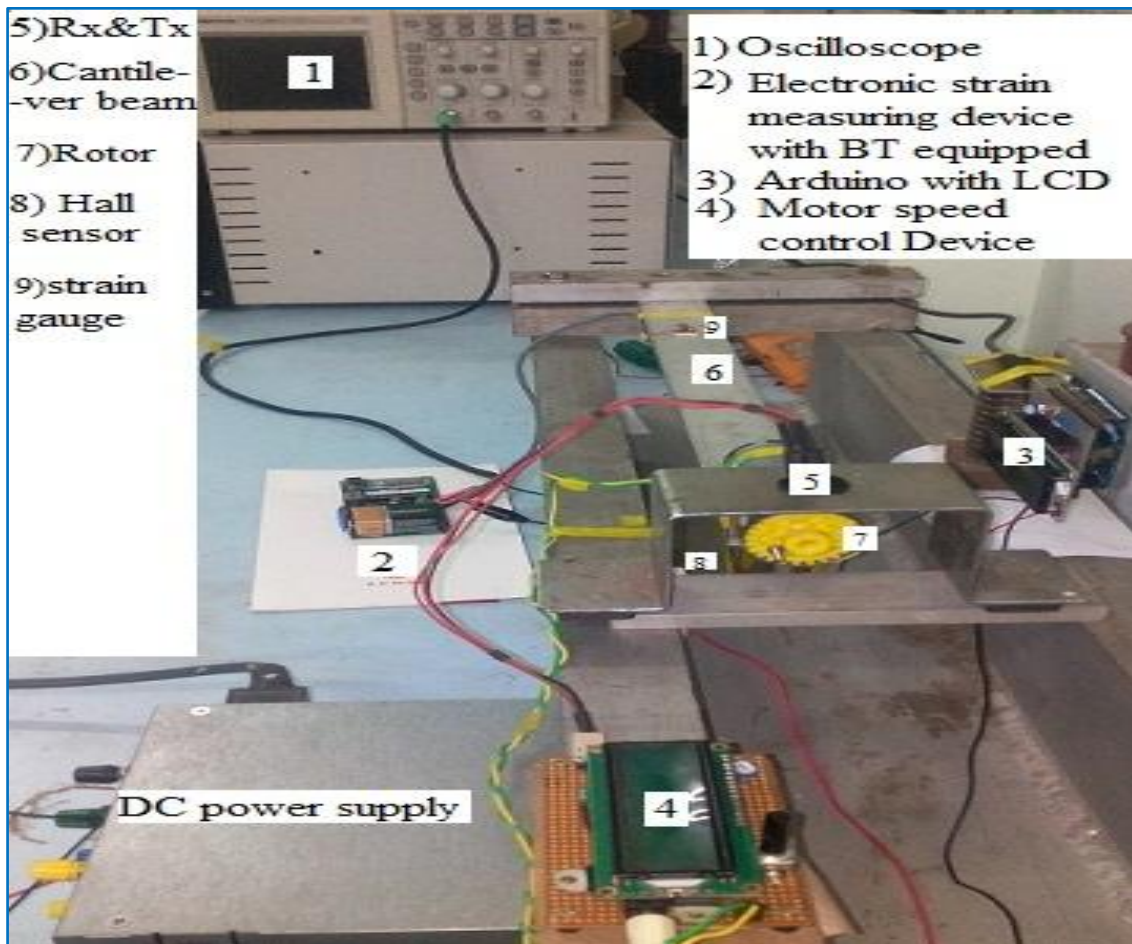


Fig.5.17: Complete Experimental setup

This is just an overview. More discussion could be found in the upcoming sections.

## 5.4. Speed control of DC motor with help of microcontroller:

For inducing vibration in the cantilever beam, the speed of eccentrically loaded rotor must be adjusted and controlled. In this section we have discussed how critical vibration of the beam was induced by controlling the potentiometer knob. Once the critical speed of the beam was reached, corresponding rotor speed and voltage value was recorded in a LCD. To achieve this we had designed an electronic board which looked like the figure shown in Fig.5.18.

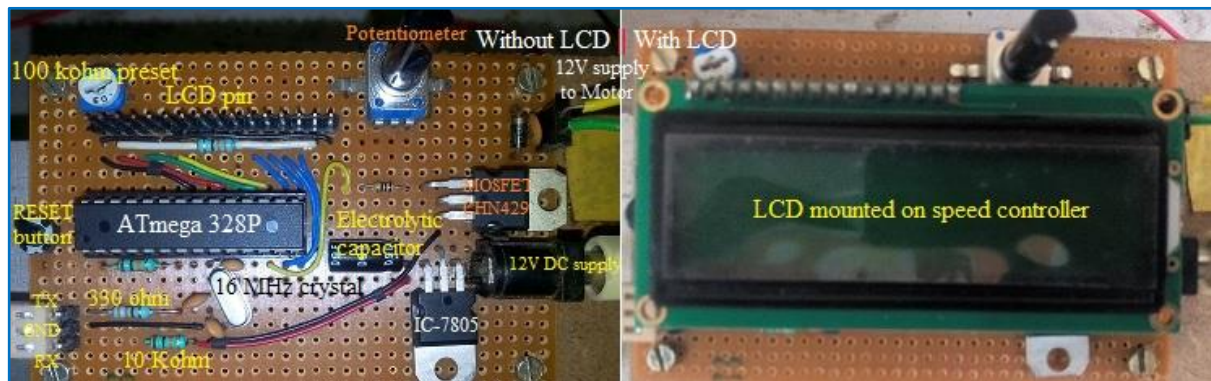


Fig.5.18: Speed controller of DC motor

### 5.4.1. Brief description about the controller:

When this controller was connected to 12V DC main power supply, IC-7805 regulated it to a 5V DC supply suitable for microcontroller. An electrolytic capacitor filtered the signal and made it free from any noise. Then this 5V supply reached to ATmega 328P-PU. Through a Potentiometer (10Kohm), this 5V DC supply was made to vary. From there, PWM pin6 sent a modulated pulse to an on-board MOSFET and at the same time MOSFET took a 12V main supply which, combined with 5V, varied modulated pulse signal from potentiometer and made it to a 12V PWM and fed it to the motor (1000 rpm). As potentiometer voltage was varied, the PWM signal to the motor was varied too. This combined signal was sent to an analog Ao pin of microcontroller. A 10Kohm resistor was connected to pin0 (RESET pin) of microcontroller and 5V supply with a push switch to ground; it helped us to RESET the microcontroller, if needed to get rid of any disturbance. The Tx and Rx were attached to the controller board. Tx (IR diode) was placed between 5V and ground through 330 ohm resistor. Rx (IR photo diode) was placed between ground and 5V through 10Kohm resistor (Fig.5.19). Interrupt-0 pin of microcontroller was connected to Rx and only rising edge of every PWM was considered for counting the number of interruption and rotor speed. On the LCD, it was seen that when potentiometer voltage was set to 39%, the beam reached its critical speed and the speed of rotor became 769 rpm.

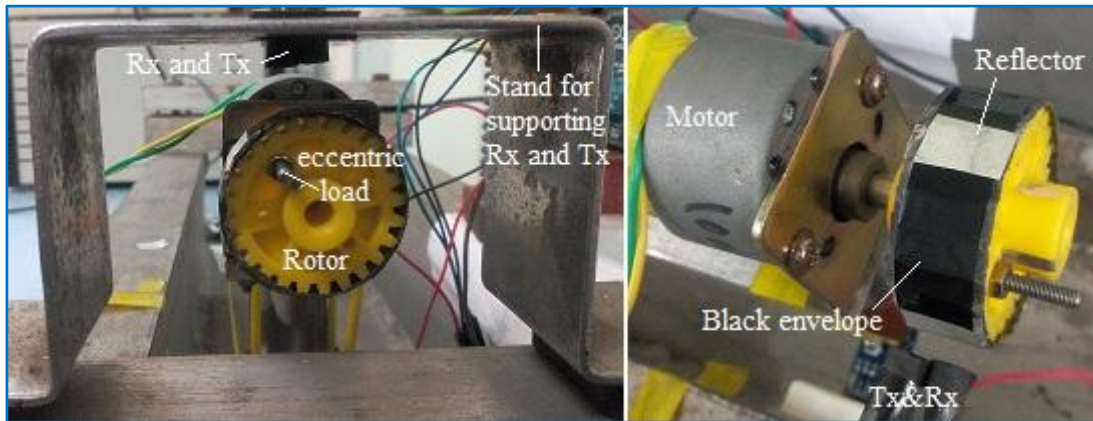


Fig.5.19: Actuation system and accessories to count RPM

**5.4.2. Experimental Procedure:** A black cello tape was wrapped around the rotor wheel and four reflectors were added for counting the number of revolution of the rotor as shown in Fig.5.19, and accordingly an Arduino programme was designed and uploaded. The below Fig.5.20 shows the programme for a motor speed controller and frequency display Unit which was uploaded on ATmega microcontroller to perform this experiment.

<pre> motorSpeedControl_frequencyDisplayUnit  #include &lt;LiquidCrystal.h&gt; LiquidCrystal lcd(12, 11, 10, 9, 8, 7); int pwm=6; int pot=A0; float value=0; int percent; float rev=0; int rpm; int oldtime=0; int time; void isr() //interrupt service routine { rev++; } void setup() { lcd.begin(16, 2); // (column, row) Initialize LCD lcd.setCursor(0,1); delay(3000); attachInterrupt(0,isr,RISING);} </pre>	<pre> void loop() { delay(1000); detachInterrupt(0); time=millis()-oldtime; rpm=(rev/time)*15000; oldtime=millis(); rev=0; value=analogRead(pot); value=value/4; analogWrite(pwm,value); percent=(value/255)*100; lcd.clear(); lcd.setCursor(0,0); lcd.print("__TACHOMETER__"); lcd.setCursor(0,1); lcd.print(rpm); lcd.print(" RPM"); lcd.print(" "); lcd.print(percent); lcd.print("%"); attachInterrupt(0,isr,RISING);} </pre>
---	---

Fig.5.20: Arduino programme for speed control of DC motor

**5.4.3. Result:** Fig.5.21 shows rotor speed and potentiometer voltage in percentage on the LCD mounted on the Motor speed controller board.



Fig.5.21: Tachometer



## 5.5. Vibration meter with Arduino and LCD display:

From previous section it is known that vibrating beam with permanent magnet induces a magnetic field. In this section, it will be discussed how to acquire data from magnetic field in terms of Hall voltage with the help of an Arduino as well as how to display those data on an on-board Liquid Crystal Display (LCD) for our convenience. The Liquid Crystal Library allowed us to control LCD display that was compatible with the Hitachi HD44780 driver. There are many LCDs out there and we can usually instruct them by the 16-pin interface.

**5.5.1. Hardware Required:** Arduino Uno Board, LCD Screen (compatible with Hitachi HD44780 driver), pin headers to solder to the LCD display pins, 10k ohm potentiometer, 220 ohm resistor, hook-up wires, Vero-board.

**5.5.2. Software Required:** A computer running the Arduino Software (IDE).

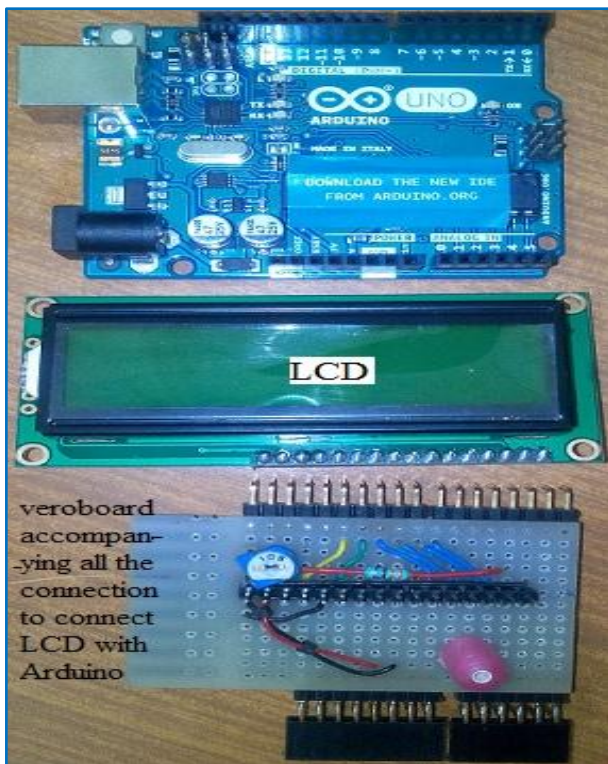


Fig.5.22:Three-Components of Vibration meter

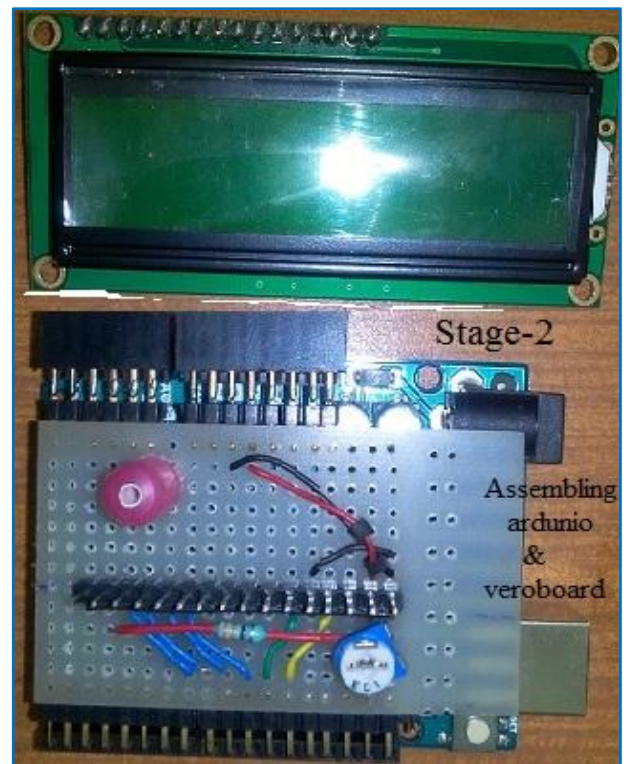


Fig.5.23:Two-Components of Vibration meter

**5.5.3. Experimental Procedure:** We had designed a Vero-board which accommodated all the pins of LCD, by which an Arduino output was displayed over the LCD. The Vero-board, LCD setup, in turn, was mounted on an Arduino. After assembling the device as shown in Fig.5.22, Fig.5.23 and Fig.5.24, the Arduino was connected to a computer via USB cable. Then following programme (Fig.5.25) was uploaded and the actuators were turned on.

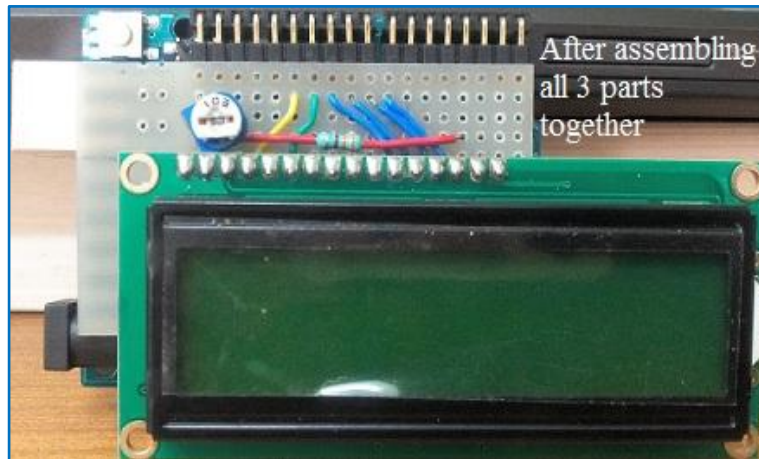


Fig.5.24: Vibration meter with Arduino and LCD display

```
vibratingBeam_HallSensor_Arduino_LCD_serialMonitor_Voltage $
// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2); // (column, row)
  // initialize the serial communications:
  Serial.begin(9600);}
void loop() {
  int Max_Val = 0;
  int Min_Val = 10000;
  for (int i=0; i <100; i++){
    int NewSensorValue = analogRead(A0);
    if (NewSensorValue > Max_Val){
      Max_Val = NewSensorValue;}
    if (NewSensorValue < Min_Val){
      Min_Val = NewSensorValue;}
    delay(1000); //delay 1000 milliseconds before the next reading:
    Serial.println(Max_Val);
    Serial.println(Min_Val);
    lcd.setCursor(0, 0); // (column, row)
    lcd.print("mxHalVlt:");
    lcd.print(Max_Val*(5.0/1024));
    lcd.setCursor(0, 1); // (column, row)
    lcd.print("mnHalVlt:");
    lcd.print(Min_Val*(5.0/1024));
  }
}
```

Fig.5.25: The Arduino programme which was uploaded while performing this experiment



Fig.5.26: LCD reading at the beginning



Fig.5.27: LCD reading during vibration

Fig.5.26 shows LCD reading when Arduino was powered on and Fig.5.27 shows LCD reading when the beam started vibrating at its critical speed.

**5.5.4. Results and Discussion:** The Fig.5.28 depicts the beam vibrating by an exciter mounted on it and that vibration created a magnetic field which was sensed by Hall sensor and sent to vibration meter for evaluation. Also the same data was seen over a computer on Arduino serial port as shown in Fig.5.29. The data was appeared alternatively as maximum and minimum, as it was designed in the Arduino programme. These voltage values were nothing but the output of Hall sensor.



Fig.5.28: Vibrating by an exciter

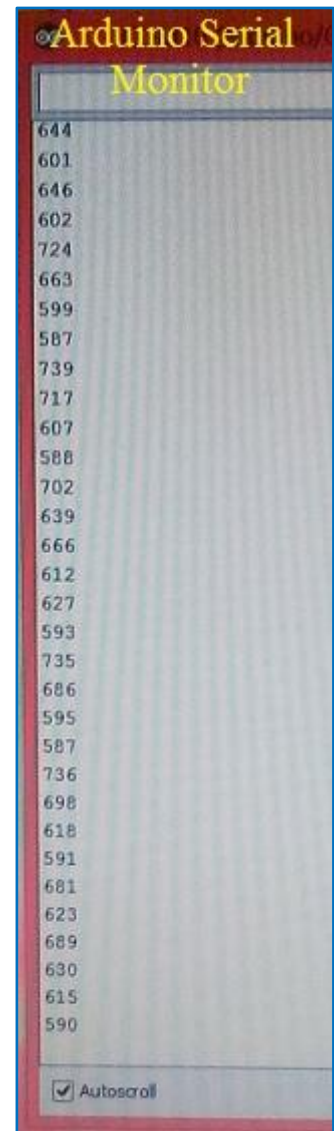


Fig.5.29: Arduino Serial Monitor



## 5.6. Arduino BT vibration meter:

In this section, the above experiment was performed again and this time we used Bluetooth powered android phone for showing output data of Hall Sensor. For that purpose a HC06 Bluetooth module was used for enabling Bluetooth on Arduino and data was collected using a BlueTerm+ android app.

**5.6.1. HC06 Bluetooth Module:** Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices and is also used for building personal area networks (PANs). Range is approximately 10 Meters (30 feet). HC-05 is a more capable module that can be set to be either Master or Slave. HC-06 is a Slave only device (Fig.5.30). These small (3 cm long) modules run on 3.3V power with 3.3V signal levels; they have no pins and usually solder to a larger board. The module has two modes of operation, Command Mode where we can send AT commands to it and Data Mode where it transmits and receives data to another Bluetooth module.



Fig.5.30:HC06..module

The default mode is DATA Mode, and this is the default configuration that may work fine for many applications:<sup>[5,7]</sup>

- Baud Rate: 9600 bps, Data : 8 bits, Stop Bits: 1 bit, Parity : None, Handshake: None
- Passkey: 1234
- Device Name: HC-06

## 5.6.2. Experimental Procedure:

BT modules Tx and Rx were connected to Arduino D10pin and D11 pin. VCC was connected to 3.5V. After the connection of GND of HC06 to GND of Arduino Uno was done properly, it looked like the figure as shown in adjacent Fig.5.31.

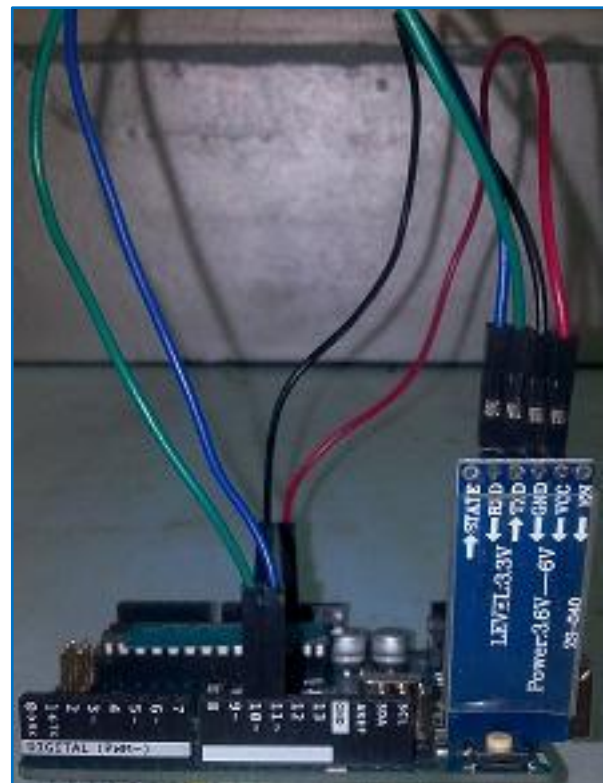


Fig.5.31: Arduino with BT HC06 module

```
vibratingBeam_HallSensor_Arduino_BT_Voltage $
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11);
// creates a "virtual" serial port UART
// connect BT module TX to D10
// connect BT module RX to D11
// connect BT Vcc to 5V, GND to GND
void setup() {
// set the data rate for the SoftwareSerial port
  BT.begin(9600);
// Send test message to other device
  BT.println("Hello from Arduino");}
//stores incoming character 4m android device
char a;
|
void loop() {
  int Max_Val = 0;
  int Min_Val = 10000;
  for (int i=0; i <100 ; i++){
  int NewSensorValue = analogRead(A0);
  if (NewSensorValue > Max_Val){
  Max_Val = NewSensorValue;}
  if (NewSensorValue < Min_Val){
  Min_Val = NewSensorValue;} }

  delay(1000);
  BT.println("Max_Val");
  BT.println(Max_Val*(5.0/1024));
  BT.println("Min_Val");
  BT.println(Min_Val*(5.0/1024));}
```

Fig.5.32: Arduino programme for BT vibration meter

Fig.5.32 shows the programme "vibratingBeam\_HallSensor\_Arduino\_BT\_Voltage" which was uploaded into Arduino for performing this experiment. The pin should be connected as instructed in the programme by comment line.

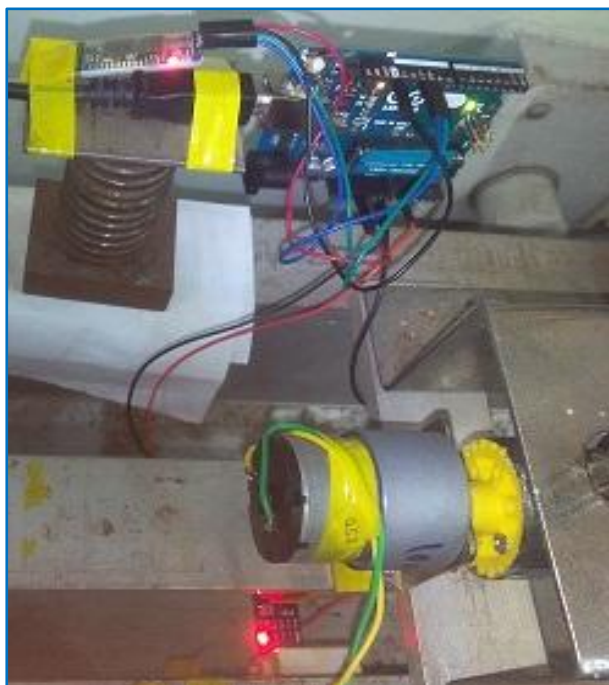


Fig.5.33: Experimental setup for BT vibration meter

Fig.5.33 shows setup for this experiment. Connect **VCC** pin of Hall sensor to **5 V DC** power supply. Connect **GND** pin of Hall sensor to the **Ground** of Arduino. Connect **AOUT** pin of Hall sensor to the analog **A0** pin of Arduino. Also, connect common ground of Arduino, BT device, Hall sensor, DC power supply.

Then Arduino programme was uploaded to start the experiment. Android phone was connected to HC06 BT device as per the description provided in the last chapter. After that, the data was appearing gradually on BlueTerm+ app.

These were the screen shots of readings which were seen on an android handset.

**5.6.3. Results and Discussion:** Fig.5.34 shows BT reading, when beam was vibrating and Fig.5.35 shows BT reading on android phone, when the beam was made to stop vibrating.

Thus a vibrating meter was developed using an Arduino, Hall sensor and HC-06 BT module.

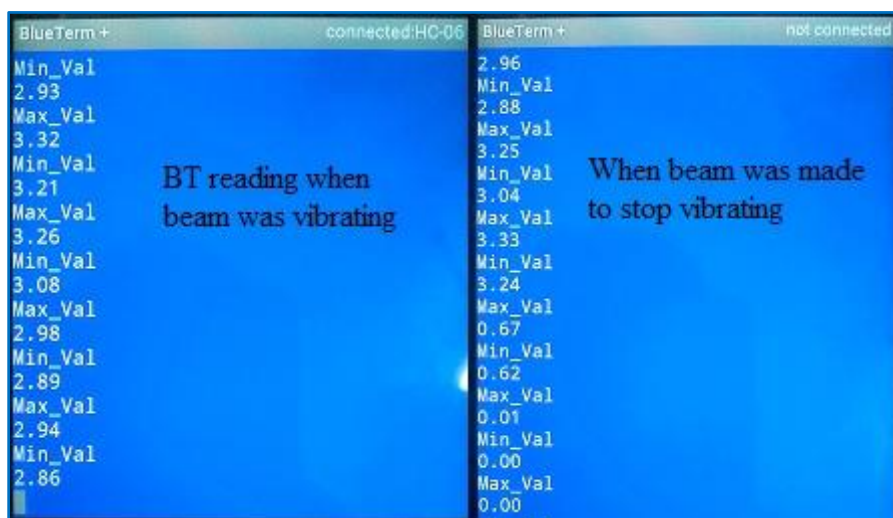


Fig.5.34: BT reading when beam was vibrating. Fig.5.35: BT reading when beam was stopped



### 5.7. ATmega-328P and INA125P with IC-7660 as vibrating strain meter:

In last section, it was discussed how to use Arduino with LCD and how a HC06 was integrated together as a vibration meter for measuring displacement. In this section, our aim was to measure strain with the help of output voltage of a strain gage and then to determine the natural frequency of the cantilever beam. Strain measuring device is shown in Fig.5.36.

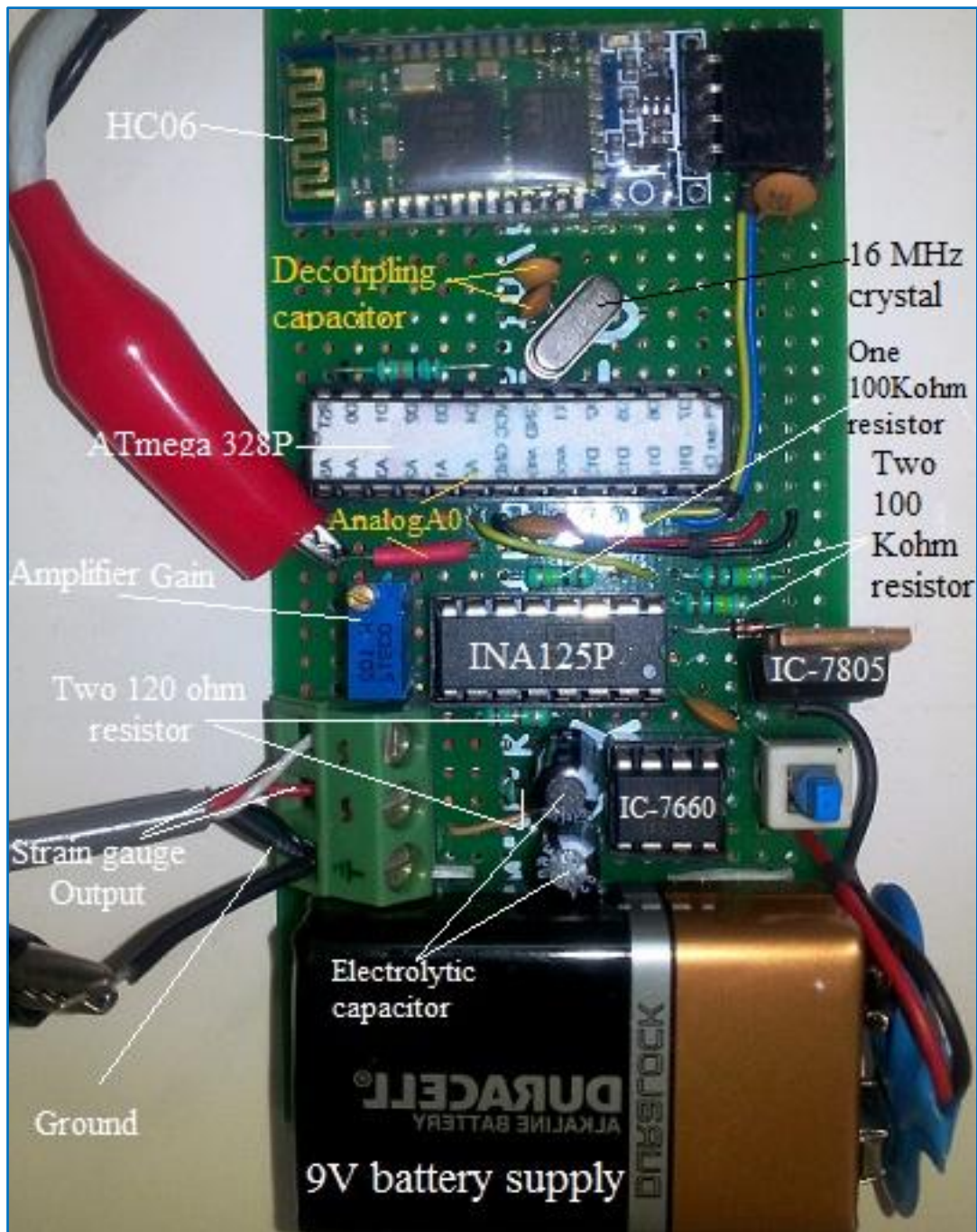


Fig.5.36: Laboratory equipped device for strain measurement

With the help of Atmega-328P, INA125P and IC-7660 we had built an electronic device which determined the natural frequency of the cantilever beam after measuring vibrating strain by the help of a strain gage. Fig.5.36 shows the device.

INA125P has single reference voltage 1.25V, 2.5V, 5V and 10V. It also has a differential amplifier. We had used single 1.25V reference volts in our device and another IC-7660 to generate a negative 5V supply which was fed to INA125P in order to operate on dual power supply.

### **5.7.1. Brief description of the circuitry:**

IC-7805 regulates 9V battery power to +5V. This 5V DC power must be smooth and free from any disturbance. IC-7660 converts this positive power supply to a negative supply and thus a dual power supply  $\pm 5V$  was fed to INA-125P amplifier. Two 10microF electrolytic capacitors present as peripheral to IC-7660. Among the four reference voltage present in INA-125P, we had used 1.25V reference signal, which, with two 120 ohm resistors, acted as one half of wheatstone bridge and other half of wheatstone bridge was completed by two strain gages, each having resistance of 120 ohm and attached to either surface of the beam. As beam started to vibrate, its upper surface and bottom surface started to expand and contract successively which, in turn, changed the resistance of wheatstone bridge and a very low signal was generated. This was sent to instrumental amplifier (INA-125P). Now amplifier amplified that signal after adding gain and sent it to ATmega microcontroller. But as this signal was a dual phase and micro-controller can only operate on positive supply, it was necessary that ground voltage of analog A0 pin should be raised to 2.5 volts. It was achieved after dividing 5V with two 100 Kohm resistor. Now, when amplified signal entered into microcontroller via a 100 Kohm resistor it was unable to damage microcontroller. A decoupling capacitor of 22pF was fitted along with a 16MHz crystal beside the ATmega-328P microcontroller. This decoupling capacitor protects the 5V power supply from 16MHz crystal noise. A HC-06 Bluetooth device also presents on the board which helped us to start a bluetooth connection with an android device and thus data was sent and read on android phone.

**5.7.2. Experiment procedure:** A wire having a pair of crocodile connector was connected at one end to analog A0 pin and Ground pin as shown in above Fig.5.36 and other end of the wire was connected to an Oscilloscope for plotting strain gage signal on it. On oscilloscope, at channel-1, all the parameters were set as shown in Fig.5.38.



### 5.7.3. Schematic of laboratory equipped instrument:

Following Fig.5.37 is a hand drawing schematic of INA125P and its interaction with strain gage and ATmega 328P-pu etc.

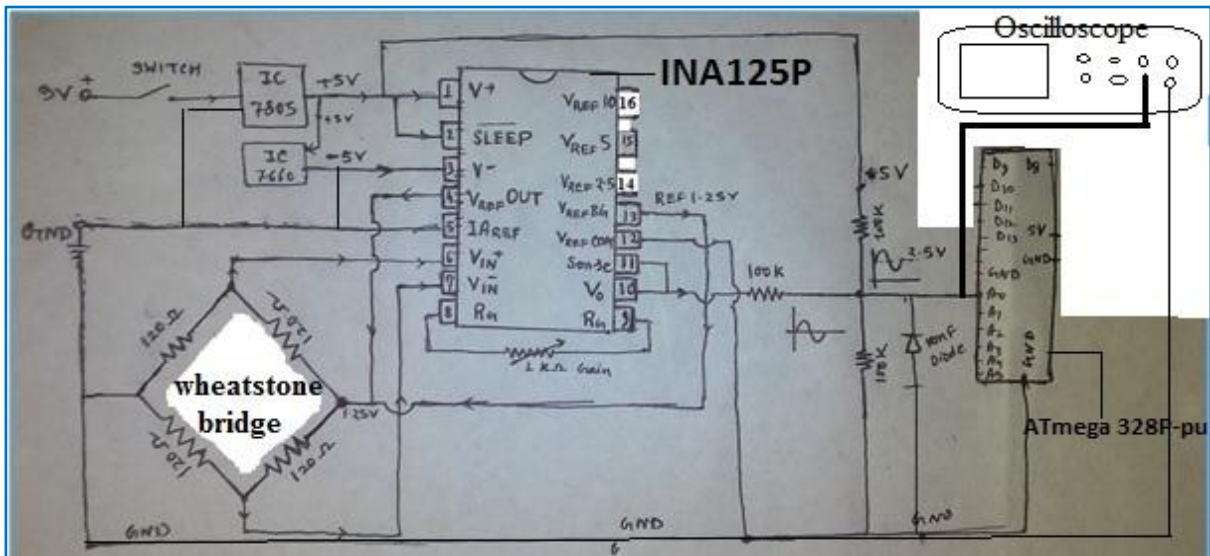


Fig.5.37: Block diagram of Laboratory equipped device for displaying strain data on Oscilloscope.

**5.7.4. Results and Discussion:** After the beam started to vibrate, a sinusoidal wave of vibrating beam appeared on the Oscilloscope. These data were amplified by the amplifier and these were the output voltage of strain gage. From the graph as shown in Fig.5.38, it could be seen that the frequency of the beam was 10.3630 Hz.

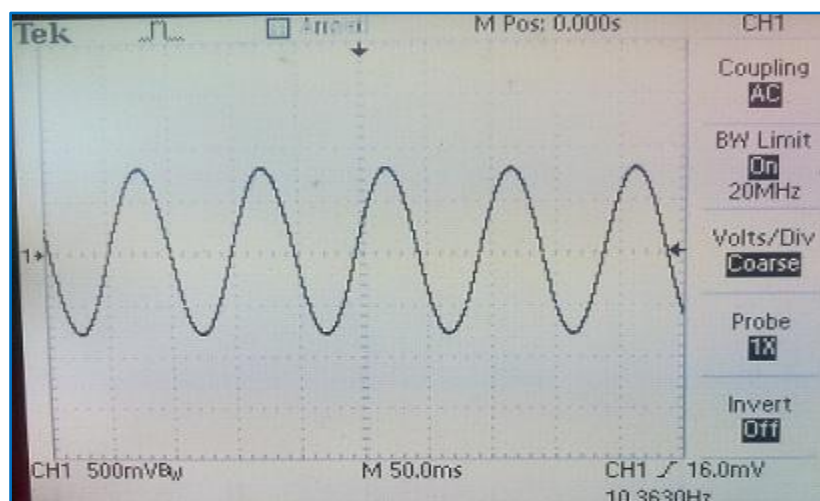


Fig.5.38: Oscilloscope Reading of strain voltage from strain gage

## 5.8. ATmega-328P with BT as Wireless Strain Meter:

The electronic device which was employed in the last experiment was remained same throughout this experiment; with only exception that the ATmega micro-controller with HC06 BT module came into play for this experiment. The device was powered on as before. An android handset was connected with the Bluetooth network available by HC06 on the strain measuring device. Now we were ready to receive the strain data over Bluetooth.

**5.8.1. EXPERIMENTAL PROCEDURE:** First, we had dismantled the ATmega from strain measuring device and then mounted it on an Arduino platform (Fig.5.39) for uploading the programme named StrainGage\_amplifier\_ATmega\_HC06BT\_phone.ino as shown in Fig.5.41



Fig.5.39: Mounting of ATmega microcontroller on Arduino for uploading programme. When the LED attached to digital pin 13 on Arduino board blinked (Fig.5.40), we got the indication that Arduino programme was uploaded.



Fig.5.40: Procedure of uploading Arduino programme on Arduino

Fig.5.41 shows programme which had to be uploaded. After uploading was complete, it was necessary to dismount the microcontroller again from Arduino platform and mount it on our strain measuring electronic instrument.

As we increased the speed of motor, the amplitude of vibration of cantilever beam increased and, at a certain speed of motor (784 rpm) and a supply voltage (39%) the beam frequency attained a critical value. This was the natural frequency of the beam which was measured from the oscilloscope as described in the previous experiment.

```
StrainGauge_amplifier_ATmega_HC06BT_phone
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11);
// creates a "virtual" serial port/UART
// connect BT module TX to D10
// connect BT module RX to D11
// connect BT Vcc to 5V, GND to GND
void setup() {
  Serial.begin(9600);
  // set the data rate for the SoftwareSerial port
  BT.begin(9600);
  // Send test message to other device
  BT.println("Hello from Arduino");
}
void loop() {
  int Max_Val = 0;
  int Min_Val = 10000;
  for (int i=0; i <1000; i++){
    int NewGaugeValue = analogRead(A0);
    if (NewGaugeValue > Max_Val){
      Max_Val = NewGaugeValue;}
    if (NewGaugeValue < Min_Val){
      Min_Val = NewGaugeValue;} }
  delay(1000);
  BT.println("Max_Val");
  BT.println(Max_Val);
  BT.println("Min_Val");
  BT.println(Min_Val );}
```

Fig.5.41: Arduino programme for wireless strain meter

### 5.8.2. Schematic of laboratory equipped instrument:

Following (Fig.5.42) is a hand drawing schematic of INA125P and its interaction with strain gage, ATmega 328P-pu, HC06 and other.

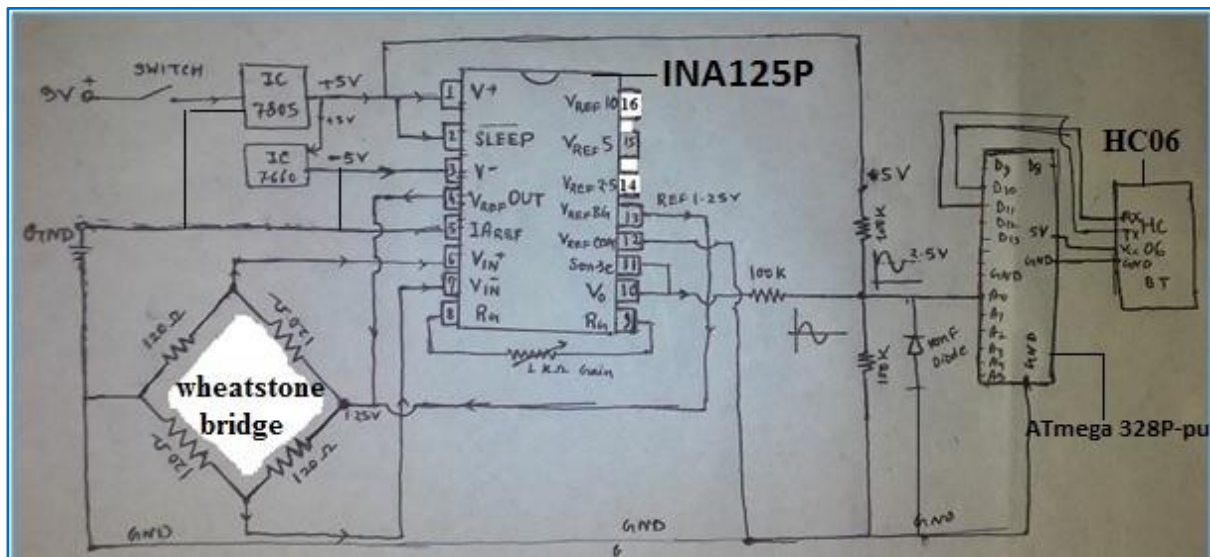


Fig.5.42: Block diagram of laboratory equipped device for sending strain data over Bluetooth

In this diagram, a HC06 Bluetooth module is also shown after the ATmega micro-controller. PIN-D10 and PIN-D11 are connected to Rx and Tx on HC06 module respectively.

**5.8.3. Results and Discussion:** We observed that as the beam was undergoing prolonged vibrating motion, its stiffness got reduced, so the natural frequency of the cantilever beam also came down. Once our device was powered on by pressing push button with the help of 9V DC battery supply, the HC06 BT module started blinking its red LED. We had connected our android handset with the wireless network available from strain measuring device and consequently the reading was showed as shown in Fig.5.44 and corresponding RPM and voltage were displayed as shown in Fig.5.43.

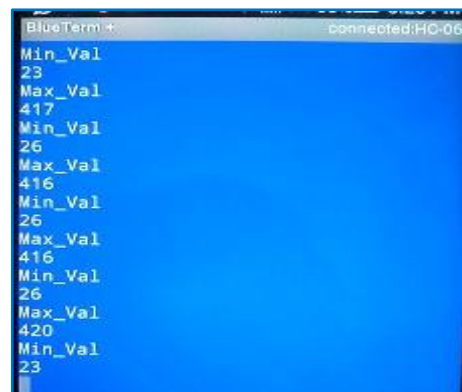


Fig.5.43: Tachometer showing critical speed Fig.5.44: Strain data on phone via Bluetooth

## 5.9. Arduino, Firmata protocol, PySerial and Matplotlib:

Before Arduino, the domain of microcontroller-based applications was limited to hardware programmers. Arduino made it simple for developers that came from other software fields and even for the non-coding community to develop microcontroller-based hardware applications. Arduino consists of a simple hardware design with a microcontroller and I/O pins to interface external devices. If one can write an Arduino sketch that can transfer the control of the microcontroller and these pins to an external software mechanism, then it will reduce one's efforts to upload Arduino sketches for every modification. This process can be performed by developing such an Arduino program that can then be controlled using a serial port. There exists a protocol called Firmata, which does exactly that. <sup>[5.9]</sup>

Firmata is a generic protocol that allows communication between the microcontroller and the software that is hosted on a computer. Any software from any computer host that is capable of serial communication can communicate with the microcontroller using Firmata. Firmata gives complete access of Arduino directly to the software and eliminates the processes of modifying and uploading Arduino sketches.

Although the Firmata protocol helps us to develop complex applications from our computer without modifying the Arduino sketch, we are not yet ready to start coding these applications.

The first step towards writing these complex applications is to provide an interface between our programming environment and the Arduino via a serial port.

Writing our own library, which includes implementation of functions and specifications to enable communication on a serial protocol, is an inconvenient and time consuming process.

The PySerial library enables communication with Arduino by encapsulating the access for the serial port. This module provides access to the serial port settings through Python properties and allows us to configure the serial port directly through the interpreter.

PySerial will be the bridge for any future communication between the Python and Arduino.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI tool kits like wxPython, Qt, or GTK+. <sup>[5.9]</sup>

## **FUTURE SCOPE OF THE WORK**

Monitoring is the regular observation and recording of activities taking place in a project or programme. It is a process of routinely gathering information on all aspects of the project. Inexpensive methods studied in this thesis can be utilized in those situations where continuous monitoring of any data is very significant. The device can be placed in a remote inaccessible or hazardous location and data can be monitored using Bluetooth or Wi-Fi. The same low-priced micro-controller based mechanism could also be utilized to track temperature and pressure of any place. From the knowledge of strain, stresses can be determined in critical components like piping systems, turbine blade, generator shaft etc. and proper precautionary steps can be taken. It is also possible to measure level of radioactivity of any place by Arduino with appropriate module and preventive measure can be adopted before it crosses safety level.



---

# References

---

- 1.1) What Is Data Acquisition? 2016  
<http://www.ni.com/data-acquisition/what-is/>(Accessed 18-05-2016)
- 1.2) Data acquisition . 2016  
[https://en.wikipedia.org/wiki/Data\\_acquisition](https://en.wikipedia.org/wiki/Data_acquisition). 2016 (Accessed 13-04-2016)
- 13) Definition and purpose of monitoring. by Phil Bartle, PhD. 2011  
<http://cec.vcn.bc.ca/cmp/modules/mon-wht.htm>(Accessed 13-04-2016)
- 1.4) Arduino - Wikipedia the free encyclopedia.2016.  
<https://en.wikipedia.org/wiki/Arduino>(Accessed 06-04-2016)
- 1.5) Why Arduino. 2016  
<https://www.arduino.cc/en/Guide/Introduction>(Accessed 06-04-2016)
- 1.6) What is Arduino. 2016  
<https://www.arduino.cc/en/Guide/Introduction>(Accessed 06-04-2016)
- 1.7) Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications. by Sheikh Ferdoush, Xinrong Li.2014  
<http://www.sciencedirect.com/science/article/pii/S1877050914009144>  
(Accessed 06-05-2016).
- 1.8) A new and inexpensive open source data acquisition and controller for solar research: Application to a water-flow glazing. by Luis J. Claros-Marfil, J. Francisco Padial, Benito Lauret. 2016  
<http://www.sciencedirect.com/science/article/pii/S0960148116301380>.  
(Accessed 06-05-2016).
- 1.9) Surveillance Car Controlled via DTMF by varun kumar .2015  
<http://www.engineersgarage.com/contribution/surveillance-car-controlled-dtmf>.  
(Accessed 06-05-2016).
- 1.10) Control an Arduino With a TV Remote.2015  
<http://www.allaboutcircuits.com/projects/how-to-control-an-arduino-with-a-tv-remote/>  
(Accessed 06-05-2016).
- 1.11) An ATmega328-based Data Acquisition System. 2015  
<http://www.allaboutcircuits.com/projects/how-to-use-the-internal-eprom-in-atmega/>  
(Accessed 06-05-2016).



- 1.12) LM35 Temperature Sensor with Data logging on SD card on Intel Edison.2016  
<http://www.instructables.com/id/LM35-Temperature-Sensor-With-Datalogging-on-SD-Car/>  
 (Accessed 06-05-2016).
- 1.13) Sensor Data Monitoring with Edison (Intel IoT). 2015  
<http://www.instructables.com/id/Sensor-Data-Monitoring-with-Edison-Intel-IoT/>.  
 (Accessed 06-05-2016).
- 1.14) TwitterPlotBot on Edison. 2015  
<http://www.instructables.com/id/TwitterPlotBot-on-Edison/>.(Accessed 06-05-2016).
- 2.1) Intel® Edison. 2016  
<https://www.arduino.cc/en/ArduinoCertified/IntelEdison>. (Accessed 11-04-2016)
- 2.2) Intel Atom- Margaret Rouse.  
<http://internetofthingsagenda.techtarget.com/definition/Intel-Atom>(Accessed 11-04-2016)
- 2.3) Multi-Core CPUs  
<http://www.howtogeek.com/194756/cpu-basics-multiple-cpus-cores-and-hyper-threading-explained/>. (Accessed 11-04-2016)
- 2.4) What does the MHz of RAM really mean? Axel Kennedal - TechTutor. 2014.  
<http://superuser.com/questions/736381/what-does-the-mhz-of-ram-really-mean>. (Accessed 11-04-2016)
- 2.5) Why does the Intel Edison have a single core microcontroller in addition to its dual core processor? David Perry. Apr2015.  
<https://www.quora.com/> (Accessed 18-04-2016)
- 2.6) QUARC® Real-Time Control Software. 2016  
<http://www.quanser.com/Products/quarc>(Accessed 19-04-2016)
- 3.1) Getting Started with Intel Edison. Nov,2015 by Stephanie Moyerman
- 3.2) <https://github.com/explore>. Github's programme by pranav mistry, 2015.
- 3.3) <https://play.google.com/store/apps/details?id=de.jentsch.blueterm&hl=en>. Jan27,2014
- 3.4) <http://www.putty.org/> (Accessed 06-05-2016).
- 4.1) Serial peripheral interface (SPI) March, 2011.  
[whatis.techtarget.com/definition/serial-peripheral-interface-SPI](http://whatis.techtarget.com/definition/serial-peripheral-interface-SPI), (Accessed 06-05-2016).
- 4.2) [fritzing.org](http://fritzing.org). (Accessed 06-12-2015)
- 5.1) Table showing Technical specifications of arduino. 2016  
<https://www.arduino.cc/en/Main/ArduinoBoardUno>(Accessed 06-04-2016)

- 5.1) Hall Effect Sensor. 2014  
<http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html> (Accessed 10-05-2016)
- 5.2) What's a strain gauge. 2015  
<http://www.kyowa-ei.com>(Accessed 12-05-2016)
- 5.3) INA125P- Strain amplifier. Brown corporation.1998  
[www.ti.com/lit/ds/symlink/ina125.pdf](http://www.ti.com/lit/ds/symlink/ina125.pdf)(Accessed 12-05-2016)
- 5.4) Block diagram of INA125P, 2015  
<http://forum.arduino.cc/index.php?topic=306118.0> (Accessed 12-05-2016)
- 5.5) Technical Specification of ATmega328P. Atmel, 2015  
<http://www.atmel.com/devices/atmega328p.aspx>(Accessed 12-04-2016)
- 5.6) ATmega 328 Pin Mapping. 2014  
<http://www.jameco.com/Jameco/workshop/JamecoBuilds/arduinocircuit.html?CID=arduinobuild>(Accessed 20-05-2016)
- 5.7) HC06 Bluetooth Module. 2015  
<https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To>(Accessed 10-05-2016)
- 5.9) Python programming for arduino. Pratik Desai, PhD, 2014  
<http://www.it-ebooks.info>. (Accessed 14-05-2016)
- 6) Getting Started with Intel Edison Sensors, Actuators, Bluetooth, and Wi-Fi on the Tiny Atom-Powered Linux Module Stephanie Moyerman, Nov,2015
- 7) The Hands-on Intel Edison Manual Lab Kindle Edition, by Agus Kurniawan .Dec,2014.