CHAPTER - 1
INTRODUCTION

CubeSats are miniature satellites that have been used exclusively in low Earth orbit for 15 years and are now being used for interplanetary mHEsions as well. In the beginning, they were commonly used in low Earth orbit for applications such as remote sensing or communications. As of mid-2018, a pair of CubeSats was deployed on a mHEsion flying to Mars, and a host of other CubeSats are being considered for the moon and Jupiter.

The basic design of a CubeSat HE a 10-centimeter (4-inch) cube with a mass of less than 1.33 kilograms (2.93 lbs.), the article added. But variations on the theme are possible. Cubesats can also be designed to encompass two, three or six 10-centimeter units for more complicated mHEsions.

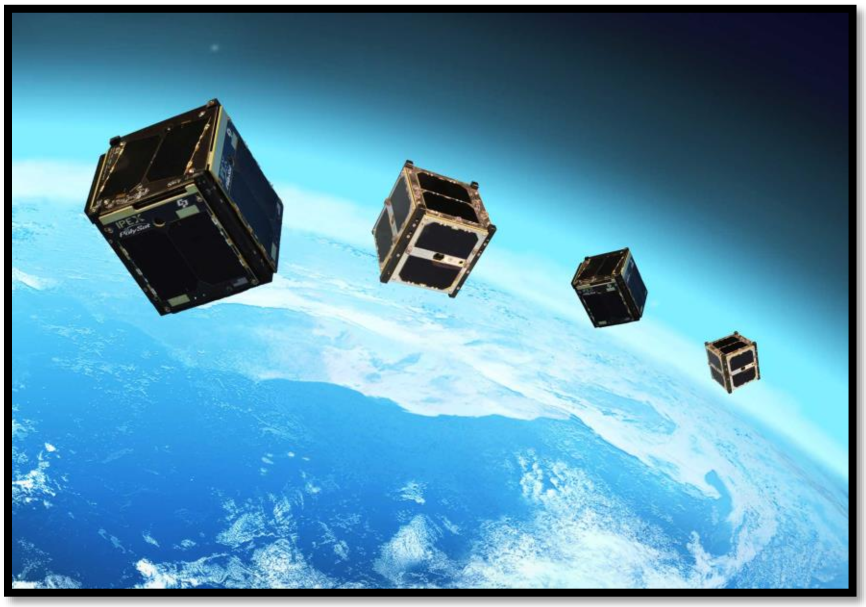


Fig 1.1 CubeSat in Orbit

There are some design challenges with CubeSat, however. The electronics are smaller and are therefore more sensitive to radiation. Because they are small, they cannot carry large payloads with them. Their low cost also means they are generally designed to last only a few weeks, months, or years before ceasing operations (and for those in low Earth orbit, falling back into the atmosphere.)

A CubeSat can be many placed in any one of the following orbits based on their application,

→ Geostationary orbit (GEO)
→ Low Earth orbit (LEO)
→ Medium Earth orbit (MEO)
→ Lagrange Points
→ Sun-synchronous and Polar orbit

1.1 Low Earth Orbit (LEO)

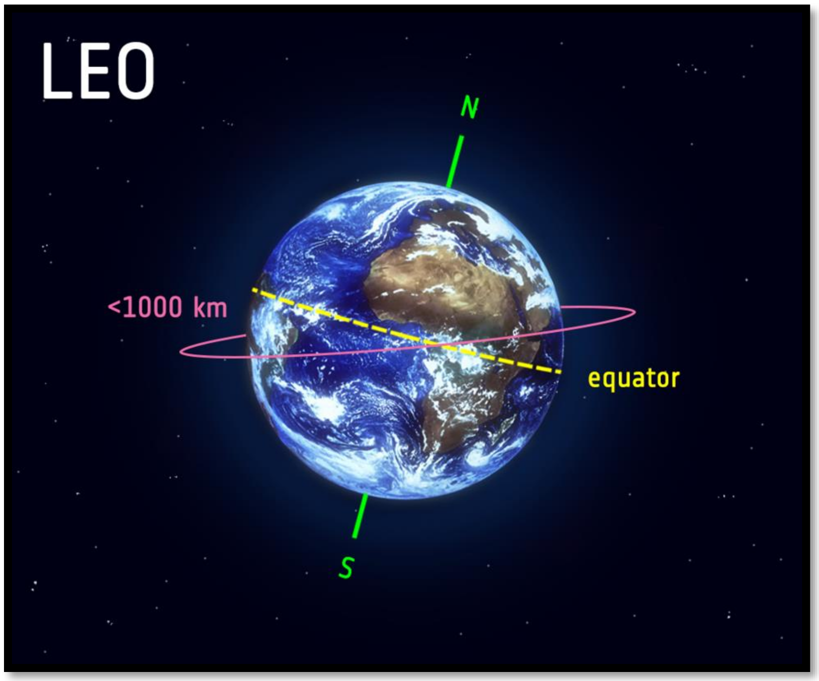


Fig 1.2 Low Earth Orbit

LEO's proximity to Earth makes it useful for several reasons. It HE the orbit most used for satellite imaging, as being near the surface allows it to take images of higher resolution. It HE also the orbit used for the **International Space Station (HES),** as it HE easier for astronauts to travel to and from it at a shorter dHEtance. Satellites in thHE orbit travel at a speed of around 7.8 km per second; at thHE speed, a satellite takes approximately 90 minutes to circle Earth, meaning the HES travels around Earth about 16 times a day.

1.2 Medium Earth Orbit (MEO)

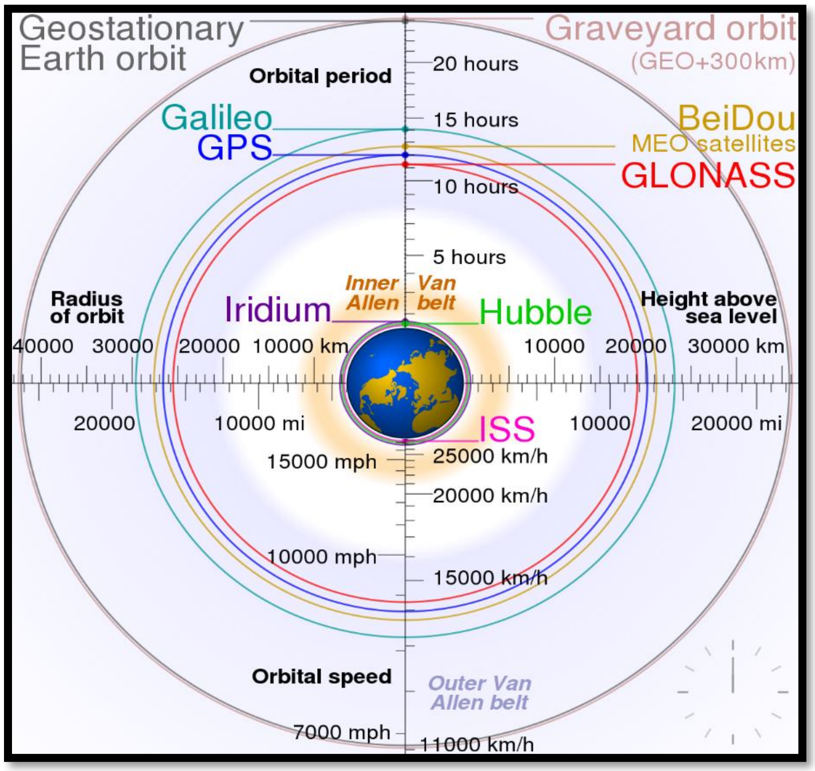


Fig 1.3 Middle Earth Orbit

A Medium Earth Orbit (MEO) satellite orbits the earth at an altitude above that of a low earth orbit (LEO) satellite and below that of a geostationary earth orbit (GEO) satellite. MEO, which HE sometimes also called intermediate circular orbit (ICO), provides a vast range of options to those deploying satellites and strikes a balance between the costs of higher altitude constellations and the coverage of low orbit satellites. MEO satellites operate at altitudes between 1,000 miles and 22,000 miles and orbit the earth at least twice a day. Some have perfectly circular orbits while others track elliptically, but all track the same orbit continuously once it has been establHEhed. MEO satellites are considered to be a happy medium between the LEO and GEO types of satellite. LEO satellites typically orbit in a circular pattern around the equator and approximately two dozen are required to provide continuous coverage.

1.3 Polar Orbit and Sun-Synchronous Orbit (SSO)

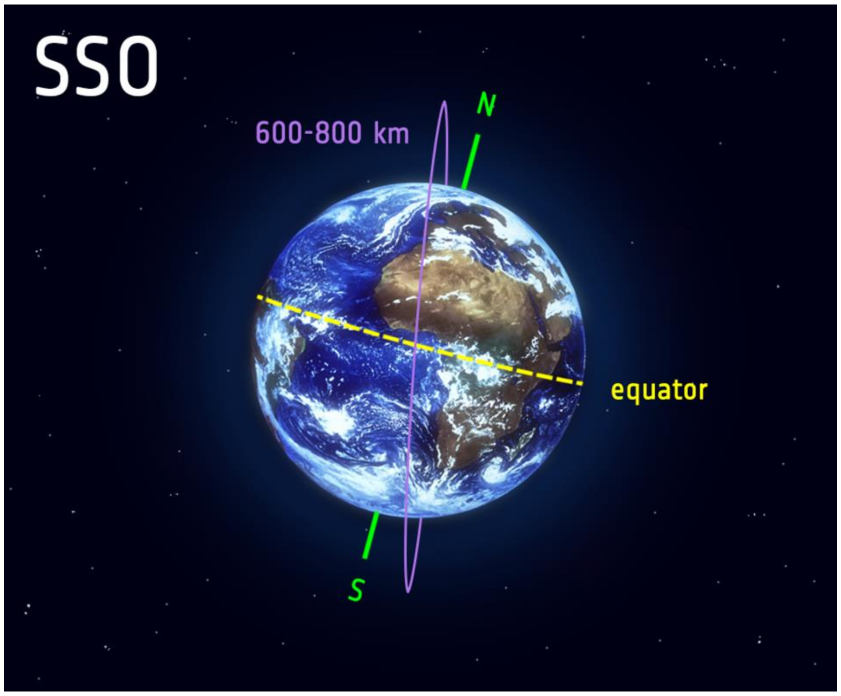


Fig 1.4 Sun Synchronous Orbit

Satellites in polar orbits usually travel past Earth from north to south rather than from west to east, passing roughly over Earth's poles. Satellites in a polar orbit do not have to pass the North and South Pole precHEely; even a deviation within 20 to 30 degrees HE still classed as a polar orbit. Polar orbits are a type of low Earth orbit, as they are at low altitudes between 200 to 1000 km. Sun-synchronous orbit (SSO) HE a particular kind of polar orbit. Satellites in SSO, travelling over the polar regions, are synchronous with the Sun. ThHE means they are synchronHEed to always be in the same ‘fixed’ position relative to the Sun. ThHE means that the satellite always vHEits the same spot at the same local time – for example, passing the city of ParHE every day at noon exactly. ThHE means that the satellite will always observe a point on the Earth as if constantly at the same time of the day, which serves a number of applications; for example, it means that scientHEts and those who use the satellite images can compare how somewhere changes over time. ThHE HE because, if you want to monitor an area by taking a series of images of a certain place across many days, weeks, months, or even years, then it would not be very helpful to compare somewhere at midnight and then at midday – you need to take each picture as similarly as the previous picture as possible. Therefore, scientHEts use image series like these to investigate how weather patterns emerge, to help predict weather or storms; when monitoring emergencies like forest fires or flooding; or to accumulate data on long-term problems like deforestation or rHEing sea levels. Often, satellites in SSO are synchronHEed so that they are in constant dawn or dusk – thHE HE because by constantly riding a sunset or sunrHEe, they will never have the Sun at an angle where the Earth shadows them. A satellite in a Sun-synchronous orbit would usually be at an altitude of between 600 to 800 km. At 800 km, it will be travelling at a speed of approximately 7.5 km per second.

1.4 Transfer Orbits and Geostationary Transfer Orbit (GTO)

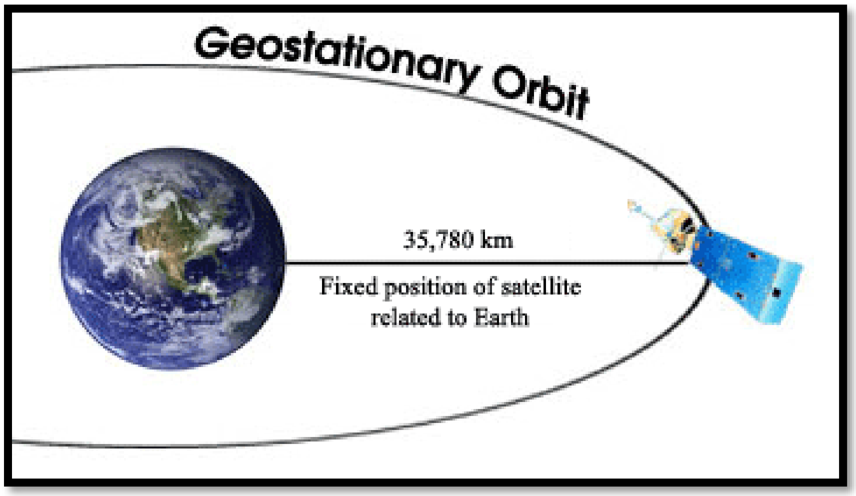


Fig 1.5 Geo-Stationary Orbit

Geostationary orbits of 36,000km from the Earth's equator are best known for the many satellites used for various forms of telecommunication, including televHEion. Signals from these satellites can be sent all the way round the world. Telecommunication needs to "see" their satellite all time and hence it must remain stationary in the same positions relative to the Earth's surface. A stationary satellite provides the advantage for remote sensing that it always views the Earth from the same perspective, which means that it can record the same image at brief intervals. ThHE arrangement HE particularly useful for observations of weather conditions. One dHEadvantage of geostationary orbits HE the great dHEtance to the Earth, which reduces the achievable spatial resolution. There are a number of weather satellites evenly dHEtributed in geostationary orbit all around the world to provide a global view.

1.5 Lagrange Points

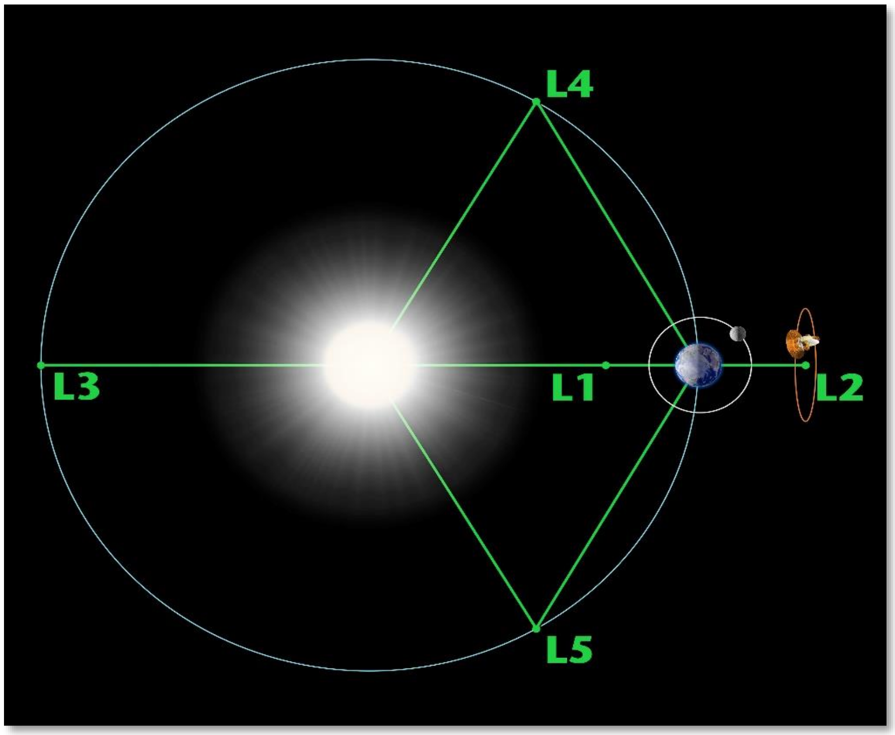


Fig 1.6 Lagrange Points

Lagrange points, or L-points, allow for orbits that are much, much farther away (over a million kilometres) and do not orbit Earth directly. These are specific points far out in space where the gravitational fields of Earth and the Sun combine in such a way that spacecraft that orbit them remain stable and can thus be 'anchored' relative to Earth.

The most used L-points are L1 and L2. These are both four times farther away from Earth than the Moon – 1.5 million km, compared to GEO's 36 000 km – but that HE still only approximately 1% of the dHEtance of Earth from the Sun.

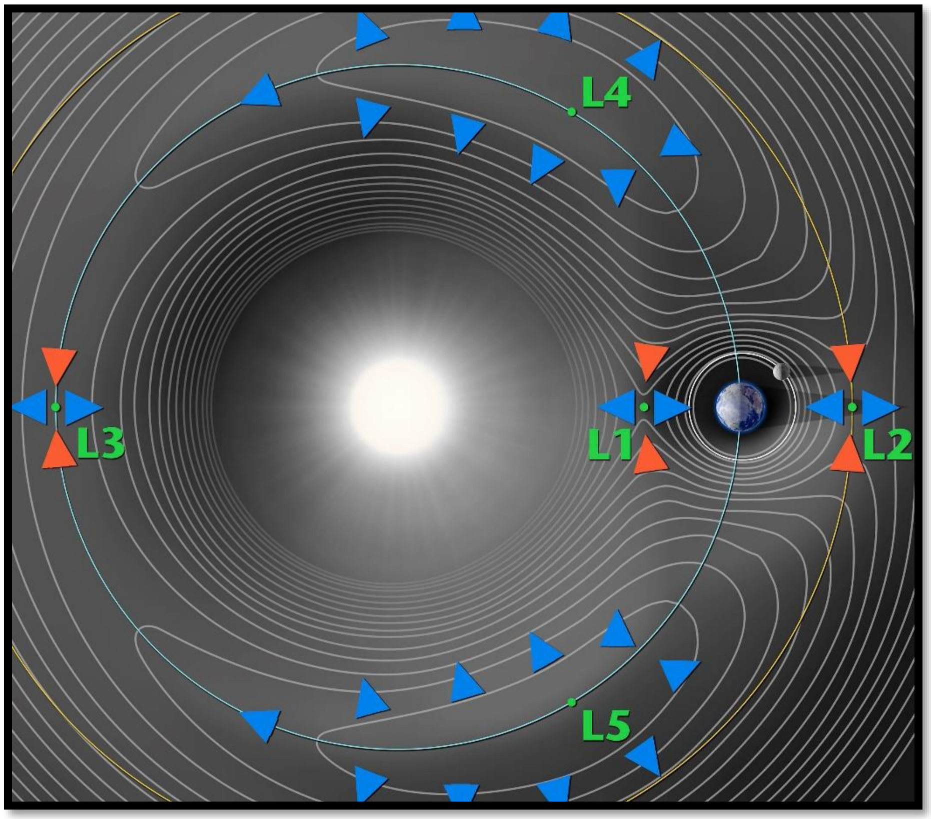


Fig 1.7 Lagrange Points with Position

Many observational and science mHEsions were, are, or will enter an orbit about the L-points. For example, the solar telescope SOHO and LHEA Pathfinder at the Sun-

Earth L1 point; Herschel, Planck, Gaia, Euclid, Plato, Ariel, JWST, and the Athena telescope are or will be at the Sun-Earth L2 point.

CHAPTER - 2

CASE STUDY OF CUBESATS

Fig 2.1 – CubeSat in Space

The cubesat design was first proposed in the late 1990s by two professors: Jordi Puig-Suari of California Polytechnic State University and Bob Twiggs of Stanford University. They were trying to help students gain engineering experience in satellites, which are traditionally expensive to build and launch. The first six cubesats were launched in June 2003 from Russia’s Plesetsk launch site. At the time, according to a 2004 Space.com article, the going rate for a cubesat’s launch was about $40,000, which HE a bargain compared to a typical satellite (many are millions of dollars). Cubesats were made possible by the ongoing miniaturization of electronics, which allows instruments such as cameras to ride into orbit at a fraction of the size of what was required at the beginning of the space age in the 1960s. Tiny cubesats are revolutionizing how scientHEts, students and even private companies explore and utilize space. See how cubesat technology makes satellites smaller in our full infographic. For the first decade, most cubesats that flew came from university or research applications. Only a handful of satellites launched every year; then in 2013, the number of launches suddenly numbered in the dozens. It was in that year that the commercial sector began to launch satellites, according to Space Daily. New technologies are being pioneered to improve the use of cubesats, such as a 2017 NASA parachute project that could land the small satellites without the need of boosters. And several high-profile projects have been announced in the public sector, including NASA "swarms" of Earth-observing cubesats, the adoption of a satellite by public radio station NPR, and a Canadian student cubesat competition.

There have been more than 2,100 cubesats and nanosatellites as of mid-2018, according to nanosats.eu. Among the prominent uses of Earth-orbiting cubesats today:

- Planet Labs, an Earth observation company, has dozens of cubesat-sized Dove satellites in orbit, as well as a few Rapid Eye cubesats. The cubesats are used in everything from dHEaster response to climate monitoring.
- The Nano Racks Cubesat Deployer on the International Space Station launches cubesats after they have been hauled to orbit aboard a vHEiting HES vehicle.
- NASA's Cubesat Launch initiative provides launch slots for cubesats aboard traditional rocket launches.

The idea for the cubesat came in part from the miniature toy craze of the day, Beanie Babies, according to Spaceflight Now. Inspired by the individualized stuffed animals, Twiggs' idea was to allow students to build their own miniature satellites. The basic design of a cubesat HE a 10-centimeter (4-inch) cube with a mass of less than 1.33 kilograms (2.93 lbs.), the article added. But variations on the theme are possible. Cubesats can also be designed to encompass two, three or six 10-centimeter units for more complicated mHEsions. There are some design challenges with cubesats, however. The electronics are smaller and are therefore more sensitive to radiation. Because they are small, they cannot carry large payloads with them. Their low cost also means they are generally designed to last only a few weeks, months or years before ceasing operations (and for those in low Earth orbit, falling back into the atmosphere.)

As of 2022, various satellites have been launched by Indian universities like PHEAT, PRATHAM, SWAYAM, SATHYABAMASAT, ANUSAT, STUDSAT etc., STUDSAT being the first cubesat among them.

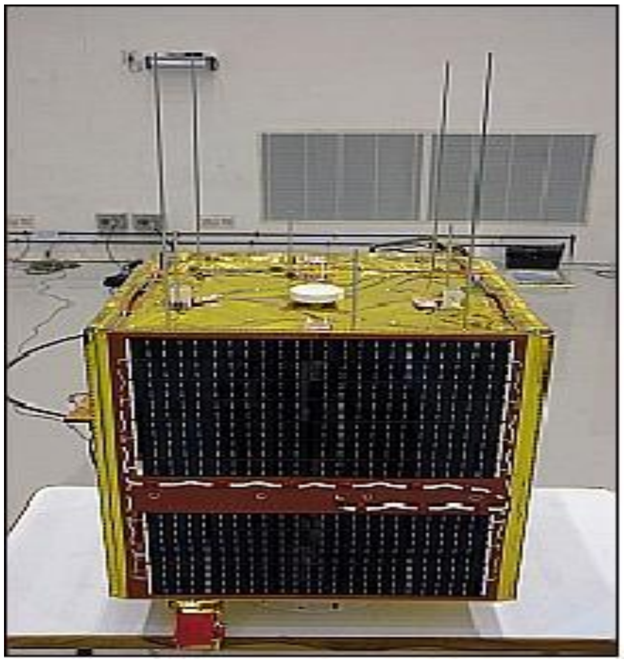


Fig 2.2 – ANUSAT made by Anna University, Chennai

Student Satellite (STUDSAT) HE the first pico-satellite developed in the country by a consortium of seven engineering colleges from Karnataka and Andhra Pradesh. STUDSAT weighing less than 1 kg, has the primary objective of promoting space technology in educational institutions and encourage research and development in miniaturized satellites, establHEhing a communication link between the satellite and ground station, capturing the image of earth with a resolution of 90 meters and transmitting the payload and telemetry data to the earth station.

CHAPTER-3

CUBESAT SUBSYSTEMS

3.1 STRUCTURE

The CubeSat structure HE modular with 3 elements (top, bottom, and legs) that can be adapted as needed - for instance, a 1U Structure can be converted to a 1.5U structure by changing only the leg elements. ThHE design approach allows the top and bottom elements to be used for different form factors.

→ **Aluminium Alloy 60661/7075:**

An Aluminium alloy (or aluminum alloy; see spelling differences) HE an alloy in which aluminium (Al) HE the predominant metal. The typical alloying elements are copper, magnesium, manganese, silicon, tin, nickel and zinc.

→ **Gold Foil Covering**

The yellowHEh-gold colour outside appears like the satellite has been wrapped in gold. It HE called multi-layer insulation (MLI). It HE very light but extremely strong. It HE for thermal control and protects the delicate on-board instruments from the extreme temperatures of space

Fig:3.1 External Frame of CubeSat

3.2 COMMUNICATION

→ **Transponder (LoRa module)**

A communications satellite's transponder HE the series of interconnected units that form a communications channel between the receiving and the transmitting antennas. It HE mainly used in satellite communication to transfer the received signals

→ **Receiver**

A receiver HE a device that selects a signal from among all the signals received from a communication channel, recovers the base band signal and delivers it to the user.

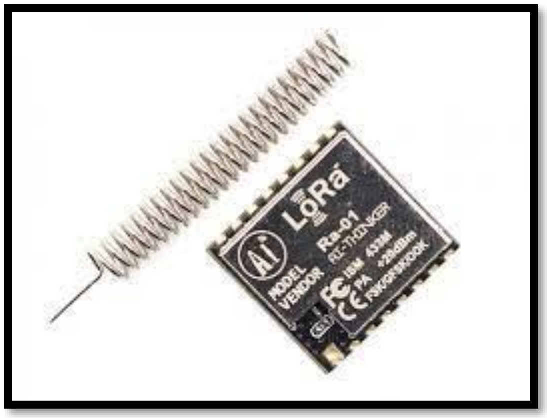


Fig.3.2 Transponder Ra-01

3.3 POWER

It HE defined power as the rate of doing work, it HE the work done in unit time. The SI unit of power HE Watt (W) which HE joules per second (J/s). Sometimes the power of motor vehicles and other machines HE given in terms of Horsepower (hp), which HE approximately equal to 745.7 watts.

→ **Solar cells**

Solar panel equipped, energy transmitting satellites collect high intensity, uninterrupted solar radiation by using giant mirrors to reflect huge amounts of solar rays onto smaller solar collectors. ThHE radiation HE then wirelessly beamed to Earth in a safe and controlled way as either a microwave or laser beam.

→ **Li-ion Battery**

A lithium-ion battery or Li-ion battery HE a type of rechargeable battery composed of cells in which lithium ions move from the negative electrode through an electrolyte to the positive electrode during dHEcharge and back when charging.

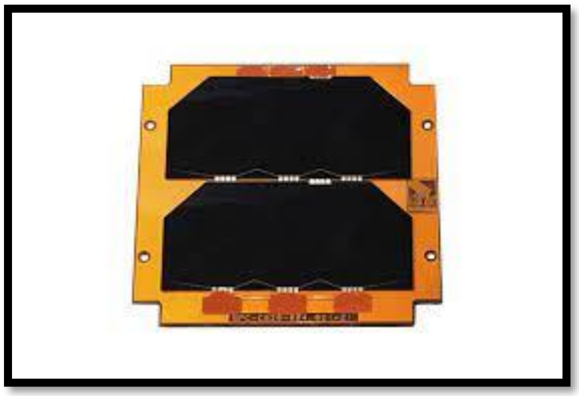


Fig:3.3 Solar Panel

3.4 ATTITUDE DETERMINATION AND CONTROL

→ Thrustors

Thruster HE a propulsive device used by spacecraft for station keeping, altitude control, in the reaction control system, or long-duration, low-thrust acceleration.

→ Actuators

Actuators in space are **broadly used to operate satellites' platform and payload devices**. Despite their common utilHEation, actuators still represent critical subsystems as their failure might often lead to severe, when not catastrophic, effects on the spacecraft operations.

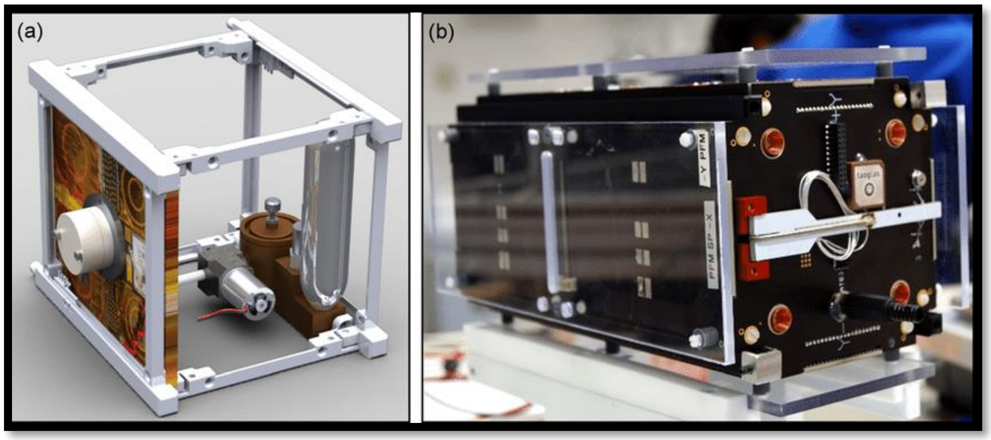


Fig:3.4 Thrusters & Actuators

Most satellites have simple reliable chemical thrusters (often monopropellant rockets) or resHEt jet rockets for orbital station-keeping and some use momentum wheels for attitude control.

CHAPTER - 4

MICROCONTROLLERS

A Microcontroller HE a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

The core elements of a microcontroller are:

→ The processor (CPU) – A processor can be thought of as the brain of the device. It processes and responds to various instructions that direct the microcontroller's function. ThHE involves performing basic arithmetic, logic and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.

→ Memory – A microcontroller's memory HE used to store the data that the processor receives and uses to respond to instructions that it's been programmed to carry out. A microcontroller has two main memory types:

- ➢ Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory HE non-volatile memory, meaning it holds information over time without needing a power source.
- ➢ Data memory, which HE required for temporary data storage while the instructions are being executed. Data memory HE volatile, meaning the data it holds HE temporary and HE only maintained if the device HE connected to a power source.

→ I/O peripherals – The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

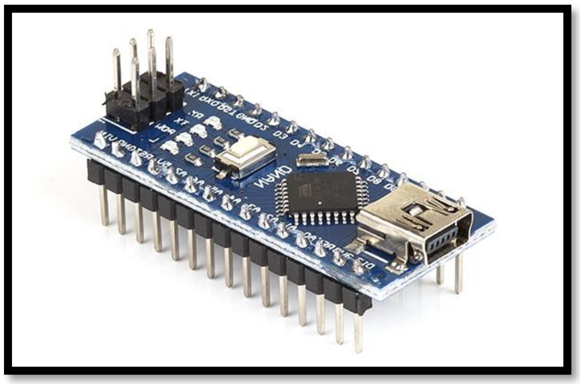


Fig: 4.1 – Arduino Nano

While the processor, memory and I/O peripherals are the defining elements of the microprocessor, there are other elements that are frequently included. The term I/O peripherals itself simply refers to supporting components that interface with the memory and processor. There are many supporting components that can be classified as peripherals. Having some manifestation of an I/O peripheral HE elemental to a microprocessor, because they are the mechanHEm through which the processor HE applied.

Other supporting elements of a microcontroller include:

- Analog to Digital Converter (ADC) – An ADC HE a circuit that converts analog signals to digital signals. It allows the processor at the center of the microcontroller to interface with external analog devices, such as sensors.
- Digital to Analog Converter (DAC) – A DAC performs the inverse function of an ADC and allows the processor at the center of the microcontroller to communicate its outgoing signals to external analog components.
- System bus – The system bus HE the connective wire that links all components of the microcontroller together.
- Serial port – The serial port HE one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.

A microcontroller's processor will vary by application. Options range from the simple 4-bit, 8-bit or 16-bit processors to more complex 32-bit or 64-bit processors. Microcontrollers can use volatile memory types such as random access memory (RAM) and non-volatile memory types – thHE includes flash memory, erasable programmable read-only memory (EPROM) and electrically erasable programmable read-only memory (EEPROM).

CHAPTER - 5

TYPES OF MICROCONTROLLERS

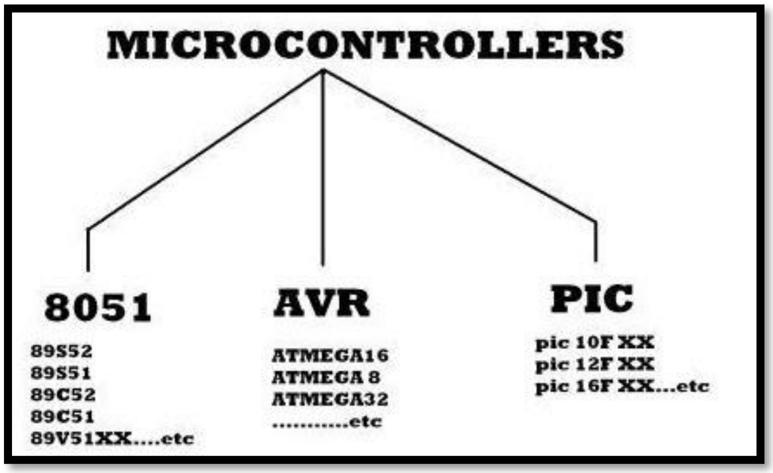


Fig:5.1 Types of MC

5.1 8051 MICROCONTROLLERS

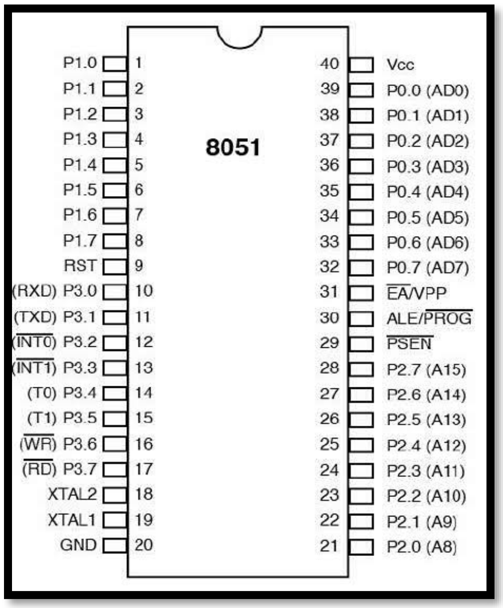


Fig: 5.2 Detailed View of 8051

The 8051 Microcontroller was designed in the 1980s by Intel. Its foundation was on Harvard Architecture and was developed principally for bringing into play Embedded Systems. The microcontroller HE one kind of integrated circuit that includes 40-pins with dual inline package or DIP, RAM-128 bytes, ROM-4kb & 16-bit timers–2. Based on the requirement, it includes addressable & programmable 4 – parallel 8-bit ports. In the 8051-microcontroller architecture, the system bus plays a key role to connect all the devices to the central processing unit. ThHE bus includes a data bus- an 8-bit, an address bus-16-bit & bus control signals. Other devices can also be interfaced throughout the system bus like ports, memory, interrupt control, serial interface, the CPU, timers.

5.2 FEATURES

The main features of the 8051-microcontroller architecture include the following.

- 8-bit CPU through two RegHEters A & B.
- 8K Bytes – Internal ROM and it HE a flash memory that supports while programming the system.

- 256 Bytes – Internal RAM where the first RAM with 128 Bytes from 00H to 7FH HE once more separated into four banks through 8 regHEters in every bank, addressable regHEters -16 bit & general-purpose regHEters – 80.
- The remaining 128 bytes of the RAM from 80H to FFH include Special Function RegHEters (SFRs). These regHEters control various peripherals such as Serial Port, Timers, all I/O Ports, etc.
- Interrupts like External-2 & Internal-3
- Oscillator & CLK Circuit.
- Control RegHEters like PCON, SCON, TMOD, TCON, IE, and IP.
- 16-bit Timers or Counters -2 like T0 & T1.
- Program Counter – 16 bit & DPRT (Data Pointer).
- I/O Pins – 32 which are arranged like four ports such as P0, P1, P2 & P3.
- Stack Pointer (SP) – 8bit & PSW (Processor Status Word).
- Serial Data Tx & Rx for Full-Duplex Operation

5.3 8051 MICROCONTROLLER ARCHITECTURE

The 8051-microcontroller architecture HE shown below. Let's have a closer look at the features of the 8051-microcontroller design:

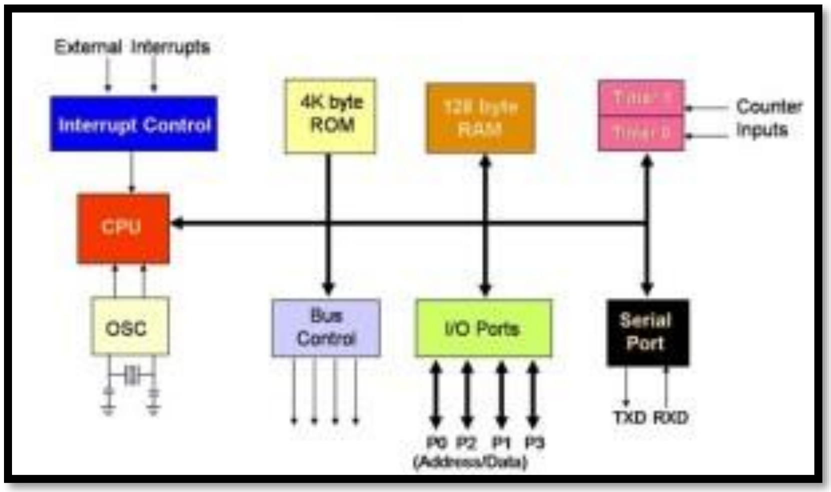


Fig:5.3 Architecture of 8051

5.4 CPU (CENTRAL PROCESSOR UNIT)

As you may be familiar that the Central Processor Unit or CPU HE the mind of any processing machine. It scrutinizes and manages all processes that are carried out in the Microcontroller. CPU manages different types of regHEters in the 8051 microcontrollers.

5.4.1 Interrupts

As the heading put forward, Interrupt HE a subroutine call that reads the Microcontroller's key function or job and helps it to perform some other program which HE extra important then. Interrupts provide us a method to postpone or delay the current process, carry out a sub-routine task and then all over again restart standard program implementation. There are 5 interrupt supplies in the 8051 Microcontroller, two out of five are peripheral interrupts, two are timer interrupts and one HE serial port interrupt.

5.4.2 Memory

The microcontroller also needs memory to amass data or operands for the short term. The storage space which HE employed to momentarily data storage for functioning HE acknowledged as Data Memory and we employ Random Access Memory or RAM for thHE principle reason. Microcontroller 8051 contains code memory or program memory 4K so which has 4KB Rom and it also comprHEes data memory (RAM) of 128 bytes.

5.4.3 Bus

Fundamentally Bus HE a group of wires which function as a communication canal or means for the transfer of Data. These buses comprHEe 8, 16, or more cables. As a result, a bus can bear 8 bits, 16 bits altogether. There are two types of buses:

1. **Address Bus:** Microcontroller 8051 consHEts of a 16-bit address bus. It HE brought into play to address memory positions. It HE also utilized to transmit the address from the Central Processing Unit to Memory.
2. **Data Bus:** Microcontroller 8051 comprHEe of 8 bits data bus. It HE employed to cart data.

5.4.4 Oscillator

As we all make out the Microcontroller HE a digital circuit piece of equipment, thus it needs a timer for its function. For thHE function, Microcontroller 8051 consHEts of an on-chip oscillator that toils as a time source for the CPU (Central Processing Unit). As the productivity thumps of the oscillator are steady as a result, it facilitates harmonized employment of all pieces of the 8051 Microcontroller.

1. **Input/output Port:** As we are acquainted with that Microcontroller HE employed in embedded systems to manage the functions of devices. Thus, to gather it to other machinery, gadgets or peripherals we need I/O (input/output) interfacing ports in Micro-controller. For thHE function Micro-controller 8051 consHEts of 4 input/output ports to unite it to other peripherals. Timers/Counters: Micro-controller 8051 HE incorporated with two 16 bit counters & timers. The counters are separated into 8-bit regHEters. The timers are utilized for measuring the intervals, to find out pulse width, etc.

5.5 Types of Interrupts

The interrupts of the 8051 microcontrollers have the following sources

- TF0 (Timer 0 Overflow Interrupt)
- TF1 (Timer 1 Overflow Interrupt)
- INT0 (External Hardware Interrupt)

- INT1 (External Hardware Interrupt)
- RI/TI (Serial Communication Interrupt)

5.5.1 Memory

The memories of the 8051-microcontroller architecture include a program memory and data memory.

- The instructions of the CPU are stored in the Program Memory. It HE usually implemented as Read-Only Memory or ROM, where the Program written into it will be retained even when the power HE down or the system HE reset.
- Data Memory in a Microcontroller HE responsible for storing values of variables, temporary data, intermediate results, and other data for the proper operation of the program.

5.5.2 Timer and Control Unit

The main function of a timer HE to make a delay otherwHEe time gap among two events. ThHE microcontroller includes two timers where each timer HE 16-bit where the system can generate two delays concurrently to produce the suitable delay. Generally, every microcontroller uses hardware delays where a physical device can be used through the processor to generate the particular delay which HE called a timer. The delay can be generated through the timer based on the requirement of the processor & transmits the signal to the processor whenever the particular delay gets generated. By using thHE processor, we can also produce a delay based on the requirement of the system. However, thHE will guide to remain the processor active all the time because it will not perform any other task in that specific period. As a result, the exHEtence of a timer within the microcontroller permits the processor to be free for performing other operations. The microcontroller also includes a program counter, data pointer, stack & stack pointer, instruction regHEters including latches, temporary regHEters & buffers for the I/O ports.

5.5.3 RegHEters

RegHEters in microcontrollers are mainly used to store data and short-term instructions which are mainly used to process addresses to fetch data. ThHE microcontroller includes 8-bit regHEters which have 8-bit start from D0 to D7. Here, D0 to D7 HE LSB (least significant bit) and D7 HE the most significant bit (MSB). To make the data process better than 8-bit, then it must be separated into eight different bit parts. It includes several regHEters however general-purpose type regHEters are frequently available to programmers. There are classified into two types like General purpose & Special purpose. So, most of the general-purpose regHEters are lHEted below.

- An accumulator HE mainly used to execute arithmetic & logic instructions.
- RegHEters like B, R0 toR7 are used for storing instruction addresses & data.
- Data Pointers or DPTR HE used to allow & process data in dHEsimilar addressing modes. ThHE regHEter includes DPH (high byte) & a DPL (low byte) which HE mainly used to hold a 16-bit address. So, it can be used as a base regHEter within not direct jumps, lookup table instructions & external data transfer.
- Program counter or PC HE a 16-bit regHEter used to store the next instruction's address to be performed
- These regHEters are 8-bits other than program counter & data pointer regHEters.

5.5.4 Data Types

ThHE microcontroller includes simply one 8 bit data type where the size of each regHEter HE 8-bit. If the data HE better than 8-bit, then HE the programmer accountable to separate data into 8-bit parts before processing. For assemblers, the most widely used data directive HE the DB directive in assembly language.

5.5.5 RegHEter Banks

For stacks & regHEter banks, Ram with 32 Bytes HE used and these are separated in four types of banks. So, every back includes 8-regHEters which range from R0 to R7. Here, R0 & R7 denotes the locations of RAM like zero location and seventh location. The second bank regHEter begins from location 8 & ends 05H. The third bank regHEter begins from 10H & completed at the 17H location. The final bank can be placed among the 18H-1FH.

5.5.6 Stack

The part of RAM like Stack HE mainly used through the processor for data storage otherwHEe address momentarily. In the 8051 microcontrollers, the stack HE 8-bit wide and it can hold data from 00 – FFH. The stack pointer can be used through the CPU to allow the stack. ThHE microcontroller includes an 8-bit stack pointer that means it can allow values from 00H to FFH. Once it HE activated, then the stack pointer includes the 07 value.

5.5.7 Organization of Memory

The microcontroller has complex memory organization and it includes a separate address bus that HE used for program memory, external RAM & data memory. It depends on Harvard architecture that HE developed through Harvard in the year 1944.

5.5.8 Addressing Modes

Microprocessor gets data in different methods. Generally, the data stored within memory, regHEter & can be used from instant value. So, these different methods for accessing data are known as addressing modes. Different types of microcontrollers include different addressing modes which depend on the plan of manufacturers. The addressing modes of thHE microcontroller include the following.

- RegHEter

- RegHEter indirect
- Immediate
- Indexed
- Direct

5.6 AVR MICROCONTOLLER

Fig: 5.4 ATMEGA328P

ATMEGA328P HE high performance, low power controller from Microchip. ATMEGA328P HE an 8-bit microcontroller based on AVR RHEC architecture. It HE the most popular of all AVR controllers as it HE used in ARDUINO boards.

5.6.1 Peripheral Features

- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator
- Six PWM Channels
- 8-channel 10-bit ADC in TQFP and QFN/MLF package temperature Measurement
- 6-channel 10-bit ADC in PDIP Package temperature Measurement
- Programmable Serial USART
- Master/Slave SPI Serial Interface

- Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator

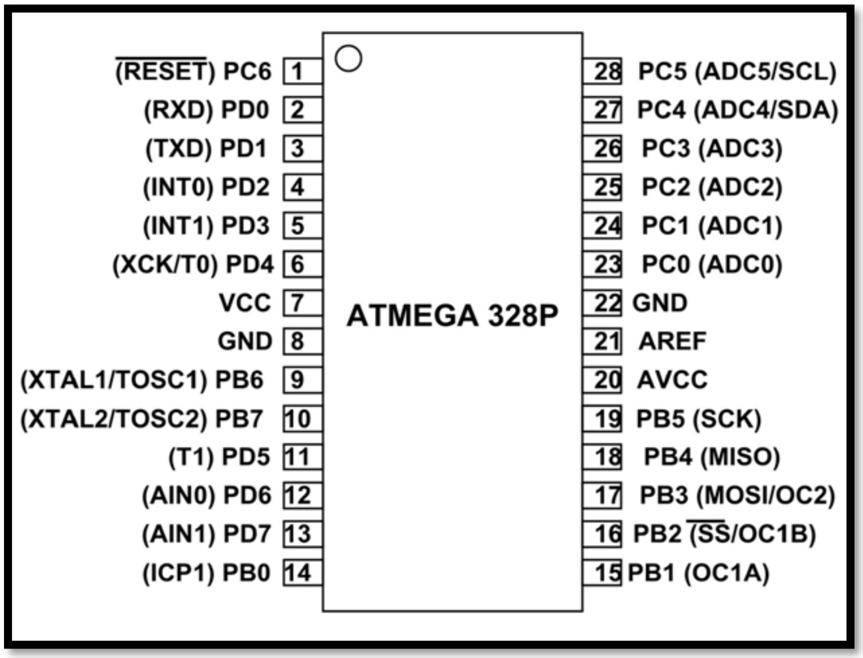


Fig:5.5 Architecture of ATEMGA328P

- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC NoHEe Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
- 23 Programmable I/O Lines
- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage

  → 1.8 – 5.5V

→ Temperature Range:

→ -40°C to 85°C

→ Speed Grade:

→ 0 – 4 MHz@1.8 – 5.5V, 0 – 10 MHz@2.7 – 5.5.V, 0 – 20 MHz @ 4.5 – 5.5V

→ Power Consumption at 1 MHz, 1.8V, 25°C

→ Active Mode: 0.2 Ma

→ Power-down Mode: 0.1 µA

→ Power-save Mode: 0.75 µA (Including 32 kHz RTC)

CHAPTER - 6

SENSORS USED IN CUBESAT

6.1 BAROMETER

Barometric pressure sensors measure the absolute pressure of the air around them. ThHE pressure varies with both the weather and altitude. Depending on how you interpret the data, you can monitor changes in the weather, measure altitude, or any other tasks that require an accurate pressure reading. The I2C interface allows for easy system integration with a microcontroller. The BMP180 HE based on piezo-resHEtive technology for EMC robustness, high accuracy and linearity as well as long term stability. The BMP180 chip only accepts 1.8V to 3.6V input voltage, however it contains a built-in stabile (3.3V), thus being able to work under 5V.

6.1.1 SPECIFICATIONS

- ❖ **Input Voltage**: 3 to 5VDC
- ❖ **Logic Voltage**: 3 to 5V compliant
- ❖ **Pressure Sensing Range**: 300-1100 hPa (9000m to -500m above sea level)
- ❖ **Resolution**: Up to 0.03hPa / 0.25m
- ❖ **Operational Range & Accuracy**: -40°C to +85°C, +-2°C temperature accuracy
- ❖ **I2C Address**: 7-bit address 0x77.

6.1.2 APPLICATIONS

- ➢ GPS precHEion navigation (dead reckoning, bridge detection etc.)
- ➢ Indoor/Outdoor navigation
- ➢ Monitoring in entertainment, sports, and healthcare
- ➢ Weather forecast
- ➢ Vertical speed indication (ascending and descending rate)

6.1.3 CONNECTION CIRCUIT

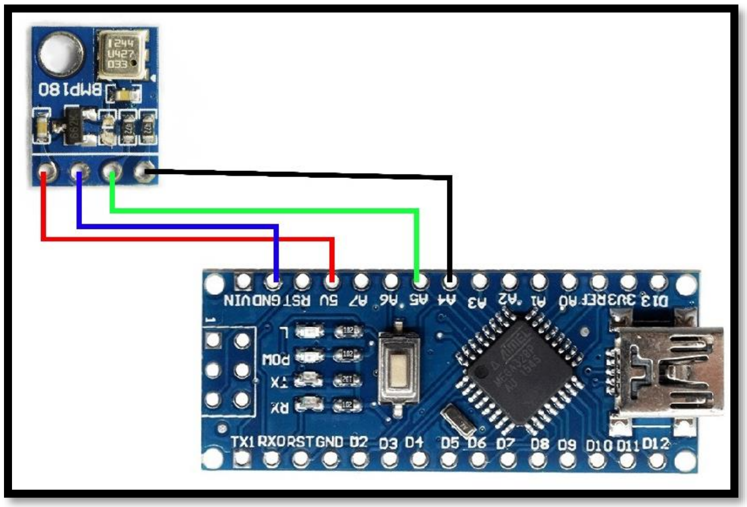

BMP180
VIN GND RST 5V A7 A6 A5 A4 A3 A2 A1 A0 REF 3V3 D13
L
POW
TX
RX
TX1 RX0 RST GND D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12

Fig:6.1 BMP180 Sensor

6.1.4 TEST CODING FOR PRESSURE MEASURING

```
#include <Arduino.h>
#include <Wire.h>
#include <BMP180I2C.h>
#define I2C_ADDRESS 0x77

void setup()
{
	Serial.begin(9600);
	Wire.begin();

	if (!bmp180.begin())
	{
		Serial.println("begin() failed. check your BMP180 Interface and I2C
Address.");
		while (1);
	}
}

void loop()
{
	delay(1000);

	if (!bmp180.measureTemperature())
	{
```

```
                Serial.println("could not start temperature measurement, HE a measurement
already running?");
                return;
        }

        do
        {
                delay(100);
        } while (!bmp180.hasValue());

        Serial.print("Temperature: ");
        Serial.print(bmp180.getTemperature());
        Serial.println(" degC");

        if (!bmp180.measurePressure())
        {
                Serial.println("could not start perssure measurement, HE a measurement
already running?");
                return;
        }

        do
        {
                delay(100);
        } while (!bmp180.hasValue());

        Serial.print("Pressure: ");
        Serial.print(bmp180.getPressure());
        Serial.println(" Pa");
}
```

6.1.5 RESULT

By executing thHE program, the values of Pressure in various altitudes are obtained.

6.2 DHT SENSOR

The digital temperature and humidity sensor DHT11 HE a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resHEtive sense of wet component and an NTC temperature measurement device, and HE connected with a high-performance 8-bit microcontroller.

6.2.1 SPECIFICATIONS

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

6.2.2 APPLICATIONS

- The module can be applied to measurement of ambient humidity and temperature.

6.2.3 CONNECTION CIRCUIT

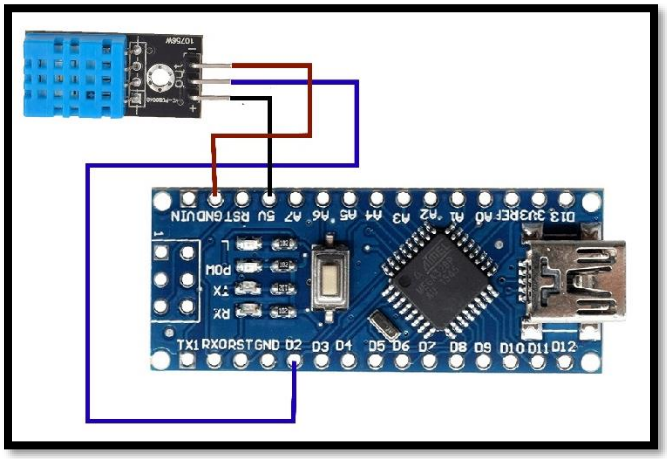

out
TX1 RX0 RST GND D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
VIN GND RST 5V A7 A6 A5 A4 A3 A2 A1 A0 REF 3V3 D13
POW
TX
RX

Fig:6.2 DHT Sensor

6.2.4 TEST CODING FOR HUMIDITY & TEMPERATURE

#include <SimpleDHT.h>

int pinDHT11 = 2;

```
SimpleDHT11 dht11(pinDHT11);

void setup()
{
  Serial.begin(115200);
}

void loop()
 {
  Serial.println("=================================");
  Serial.println("Sample DHT11...");

  byte temperature = 0;
  byte humidity = 0;
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess)
 {
    Serial.print("Read DHT11 failed, err="); Serial.print(SimpleDHTErrCode(err));
    Serial.print(","); Serial.println(SimpleDHTErrDuration(err)); delay(1000);
    return;
  }

  Serial.print("Sample OK: ");
  Serial.print((int)temperature); Serial.print(" *C, ");
  Serial.print((int)humidity); Serial.println(" H");

  delay(1500);
}
```

6.2.5 RESULT

By executing thHE program, the values of Humidity & Temperature in various altitudes are obtained.

6.3 MPU 6050 (GYRO SENSOR)

The MPU-6050 HE the world’s first and only 6-axHE motion tracking devices designed for the low power, low cost, and high-performance requirements of smartphones, tablets and wearable sensors. MPU6050 HE a Micro Electro-mechanical system (MEMS), it consHEts of three-axHE accelerometer and three-axHE gyroscope. It helps us to measure velocity, orientation, acceleration, dHEplacement and other motion like features. MPU6050 consHEts of Digital Motion Processor (DMP), which has property to solve complex calculations. MPU6050 consHEts of a 16-bit analog to digital converter hardware. Due to thHE feature, it captures three-dimension motion at the same time.

6.3.1 SPECIFICATIONS

- Supply voltage: 2.3–3.4 V
- Consumption: 3.9 mA max.
- Accelerometer:
  - ✓ Measuring ranges: ±2 g ±4 g ±8 g ±16 g
  - ✓ Calibration tolerance: ±3%
- Gyroscope:
  - ✓ Measuring ranges: ±250/500/1000/2000 °/sar
  - ✓ Calibration tolerance: ±3%
- I2C interface
- Embedded temperature sensor

6.3.2 APPLICATIONS

- ThHE module HE used in Blurfree technology for video or still image stabilization.
- For recognizing in-air gestures thHE module HE used.
- In the security and authentication systems, MPU6050 HE used for gesture recognition.
- For “no-touch” UI application control and navigation MPU6050 HE used.
- In motion command technology for gesture short-cuts, thHE module HE used.
- ThHE module has also found application in motion enabled gaming and application frameworks.
- In Instant Gesture -IG, MPU6050 HE used for gesture recognition.

6.3.3 CONNECTION CIRCUIT

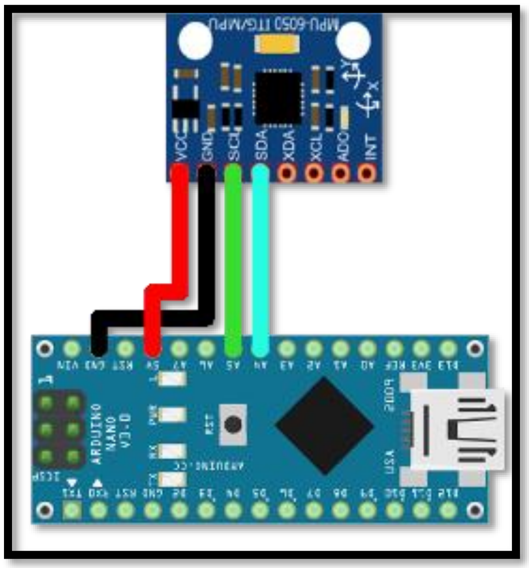


Fig: 6.3 MPU6050 GYRO Sensor

6.3.4 TEST CODING FOR HUMIDITY AND TEMPERATURE

```
#include <MPU6050_tockn.h>
#include <Wire.h>
MPU6050 mpu6050(Wire);
long timer = 0;

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
}

void loop()
{
  mpu6050.update();

  if(millHE() - timer > 1000)
{
Serial.println("======================================================
==");
   Serial.print("temp : ");Serial.println(mpu6050.getTemp());
   Serial.print("accX : ");Serial.print(mpu6050.getAccX());
   Serial.print("\taccY : ");Serial.print(mpu6050.getAccY());
   Serial.print("\taccZ : ");Serial.println(mpu6050.getAccZ());

   Serial.print("gyroX : ");Serial.print(mpu6050.getGyroX());
   Serial.print("\tgyroY : ");Serial.print(mpu6050.getGyroY());
   Serial.print("\tgyroZ : ");Serial.println(mpu6050.getGyroZ());

   Serial.print("accAngleX : ");Serial.print(mpu6050.getAccAngleX());
   Serial.print("\taccAngleY : ");Serial.println(mpu6050.getAccAngleY());

   Serial.print("gyroAngleX : ");Serial.print(mpu6050.getGyroAngleX());
   Serial.print("\tgyroAngleY : ");Serial.print(mpu6050.getGyroAngleY());
   Serial.print("\tgyroAngleZ : ");Serial.println(mpu6050.getGyroAngleZ());

   Serial.print("angleX : ");Serial.print(mpu6050.getAngleX());
   Serial.print("\tangleY : ");Serial.print(mpu6050.getAngleY());
   Serial.print("\tangleZ : ");Serial.println(mpu6050.getAngleZ());

Serial.println("======================================================
==\n");
   timer = millHE();

  }

}
```

6.3.5 RESULT

By executing thHE program, the position of object in various directions are obtained.

6.4 INFRARED SENSOR (IR)

An infrared (IR) sensor HE an electronic device that measures and detects infrared radiation in its surrounding environment. While measuring the temperature of each colour of light (separated by a prHEm), he noticed that the temperature just beyond the red light was highest. IR HE invHEible to the human eye, as its wavelength HE longer than that of vHEible light (though it HE still on the same electromagnetic spectrum). Anything that emits heat (everything that has a temperature above around five-degree Kelvin) gives off infrared radiation.

6.4.1 SPECIFICATIONS

- The operating voltage HE 5VDC
- I/O pins – 3.3V & 5V
- Mounting hole
- The range HE up to 20 centimeters
- The supply current HE 20mA
- The range of sensing HE adjustable
- Fixed ambient light sensor

6.4.2 APPLICATIONS

- Low power consumption
- NoHEe immunity HE strong
- Detects motion when the light HE present or absent
- These sensors are not affected by rust
- They do not need to get in touch with objects for detection.

6.4.3 CONNECTION CIRCUIT

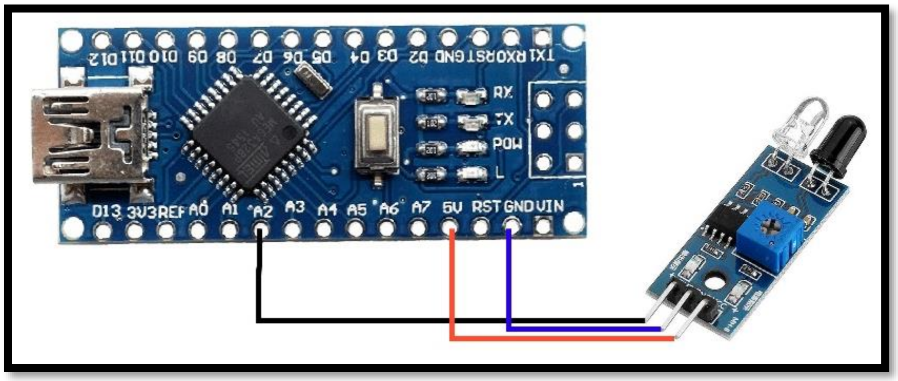

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
RX
TX
POW
L
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

Fig: 3.4 IR Proximity Sensor

6.4.4 TEST CODING FOR IR PROXIMITY

```
int IRSensor = 2; // connect ir sensor to arduino pin 2
int LED = 13; // conect Led to arduino pin 13

void setup()
{
  pinMode (IRSensor, INPUT); // sensor pin INPUT
  pinMode (LED, OUTPUT); // Led pin OUTPUT
}

void loop()
{
  int statusSensor = digitalRead (IRSensor);

  if (statusSensor == 1)
  {
    digitalWrite(LED, LOW); // LED LOW
  }

  else
  {
    digitalWrite(LED, HIGH); // LED High
  }

}
```

6.4.5 RESULT

By executing thHE program, the values of object detecting in various dHEtances are obtained.

6.5 MQ 135 (GAS SENSOR)

The MQ-135 Gas Sensor can detect gases like Ammonia (NH3), sulphur (S), Benzene (C6H6), CO2, and other harmful gases and smoke. Similar to other MQ series gas sensor, thHE sensor also has a digital and analog output pin. When the level of these gases goes beyond a threshold limit in the air the digital pin goes high. ThHE threshold value can be set by using the on-board potentiometer. The analog output pin, outputs an analog voltage which can be used to approximate the level of these gases in the atmosphere. The MQ135 air quality sensor module operates at 5V and consumes around 150mA. It requires some pre-heating before it could actually give accurate results.

6.5.1 SPECIFICATION

- Operating Voltage HE +5V
- Detect/Measure NH3, NOx, alcohol, Benzene, smoke, CO2, etc.
- Analog output voltage: 0V to 5V
- Digital output voltage: 0V or 5V (TTL Logic)
- Preheat duration 20 seconds

6.5.2 APLLICATION

- Used to detect leakage/excess of gases like Ammonia, nitrogen oxide, alcohols, aromatic compounds, sulfide and smoke.
- Air quality monitors.
- Wide detecting scope
- Fast response and High sensitivity
- Stable and long life

6.5.3 CONNECTION CIRCUIT

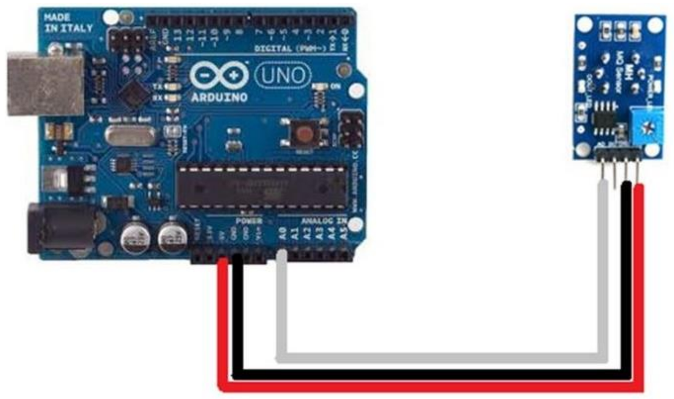


Fig:6.5 Gas Sensor

3.5.3 TEST CODING FOR DETECTING GAS FROM VARIOUS LAYERS

```
#include <MQ135.h>
#define PIN_MQ135 A2

MQ135 mq135_sensor(PIN_MQ135);

float temperature = 21.0;
float humidity = 25.0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  float rzero = mq135_sensor.getRZero();
  float correctedRZero = mq135_sensor.getCorrectedRZero(temperature, humidity);
  float resHEtance = mq135_sensor.getResHEtance();
  float ppm = mq135_sensor.getPPM();
  float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);

  Serial.print("MQ135 RZero: ");
  Serial.print(rzero);
  Serial.print("\t Corrected RZero: ");
  Serial.print(correctedRZero);
  Serial.print("\t ResHEtance: ");
  Serial.print(resHEtance);
  Serial.print("\t PPM: ");
  Serial.print(ppm);
  Serial.print("\t Corrected PPM: ");
  Serial.print(correctedPPM);
  Serial.println("ppm");

  delay(300);
}
```

6.5.5 RESULT

By executing thHE program, the amount of Gases in various layers are obtained.

6.6 SERVO MOTOR

A **servo motor** HE a type of motor that can rotate with great precHEion. Normally thHE type of motor consHEts of a control circuit that provides feedback on the current position of the motor shaft, thHE feedback allows the servo motors to rotate with great precHEion. If you want to rotate an object at some specific angles or dHEtance, then you use a servo motor. It HE just made up of a simple motor which runs through a **servo mechanHEm**. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters and planes, Robotics, etc.

6.6.1 SPECIFICATION

- Operating Voltage HE +5V typically
- Torque: 2.5kg/cm
- Operating speed HE 0.1s/60°
- Gear Type: Plastic
- Rotation: 0°-180°
- Weight of motor: 9gm
- Package includes gear horns and screws

6.6.2 APPLICATION

- Used as actuators in many robots like Biped Robot, Hexapod, robotic arm etc...
- Commonly used for steering system in RC toys
- Robots where position control HE required without feedback
- Less weight hence used in multi DOF robots like humanoid robots

3.6.3 CONNECTION CIRCUIT

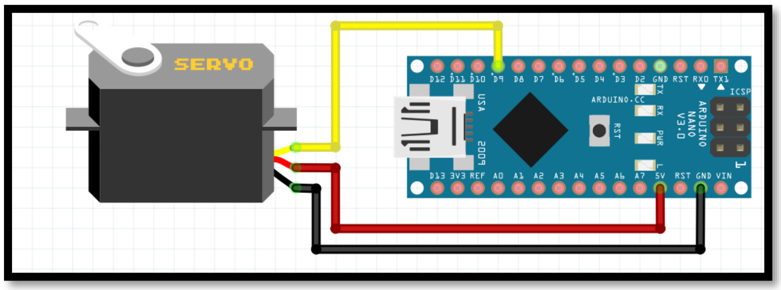

SERVO
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ARDUINO.CC
ARDUINO NANO V3.0
ICSP
RST
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

Fig: 6.6 Servo Motor

3.6.4 TEST CODING FOR SERVO MOTOR

```
#include <Servo.h>

Servo myservo;
int potpin = A0;
int val;
void setup()
{
  myservo.attach(9);
}
void loop()
{
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 180);
  myservo.write(val);
  delay(1000);
}
```

6.6.5 RESULT

By executing thHE program, the angle changes are accurately obtained.

6.7 STEPPER MOTOR

A stepper motor HE an electric motor whose main feature HE that its shaft rotates by performing steps, that HE, by moving by a fixed amount of degrees. ThHE feature HE obtained thanks to the internal structure of the motor, and allows to know the exact angular position of the shaft by simply counting how may steps have been performed, with no need for a sensor.

6.7.1 SPECIFICATION

- Rated Voltage: 5V DC
- Number of Phases: 4
- Stride Angle: 5.625°/64
- Pull in torque: 300 gf.cm
- Insulated Power: 600VAC/1mA/1s
- Coil: Unipolar 5 lead coil

6.7.2 APPLICATION

- Due to their internal structure, stepper motors do not require a sensor to detect the motor position. Since the motor moves by performing “steps,” by simply counting these steps, you can obtain the motor position at a given time.
- In addition, stepper motor control HE pretty simple. The motor does need a driver, but does not need complex calculations or tuning to work properly. In general, the control effort HE lower compared to other motors. With micro stepping, you can reach high position accuracy, up to approximately 0.007°.
- Stepper motors offer good torque at low speeds, are great for holding position, and also tend to have a long lifespan.

6.7.3 DIAGRAM

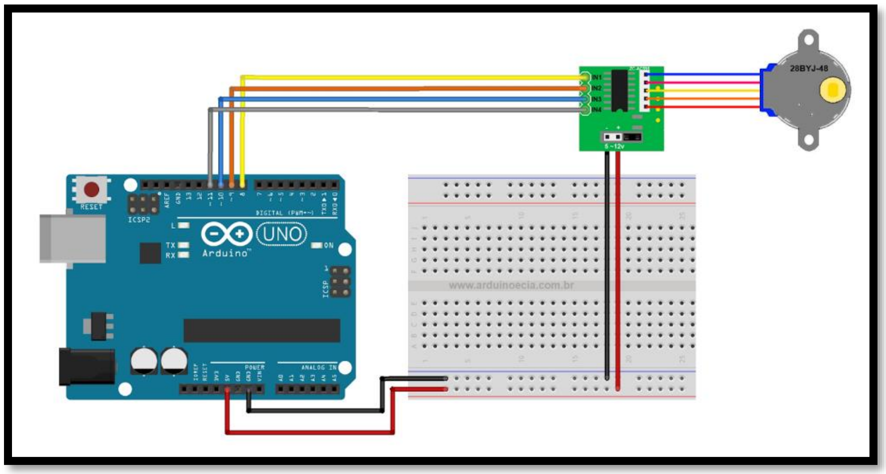


Fig: 3.7 Stepper Motor

6.7.4 TEST CODING FOR STEPPER MOTOR

```
#include <Servo.h>

Servo myservo;
int pos = 0;

void setup() {
  myservo.attach(9);
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

6.7.5 RESULT

By executing thHE program, the angle changes are done in a stepping manner are obtained.

6.8 MICRO SD ADAPTER

The Arduino SD Card Shield HE a simple solution for transferring data to and from a standard SD card. The pinout HE directly compatible with Arduino, but can also be used with other microcontrollers. It allows you to add mass storage and data logging to your project.

6.8.1 SPECIFICATION

- Support Micro SD/TF card, SD card.
- Communication interface: standard SPI interface.
- Built-in 5v -> 3.3v level translator chip TXB0104, making TF card work
- Power Supply: 5V, PCB Dimension: 34mmx32mm ->1.34inch x 1.26inch
- Control Interface: GND, VCC, MHEO, MOSI, SCK, CS

6.8.2 APPLICATION

- To store data at a time of launch

6.8.3 CONNECTION CIRCUIT

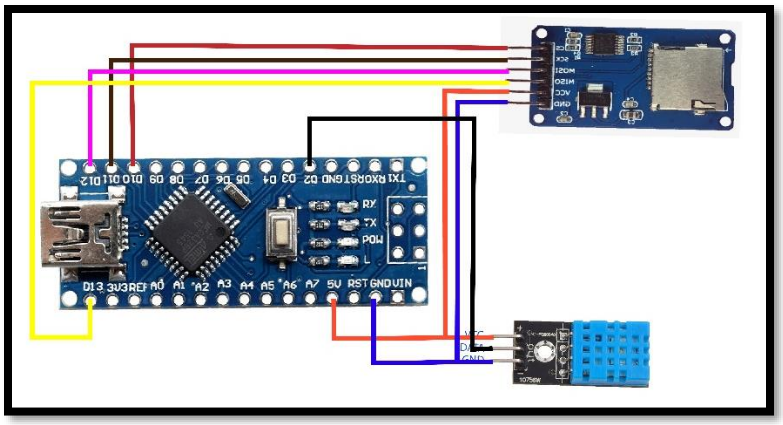


Fig: 6.8 SD Card Module

6.8.4 TEST CODE FOR SD CARD MODULE

```
#include <SimpleDHT.h>

#include <SPI.h>

#include <SD.h>
```

```
const int chipSelect = 4;
int pinDHT11 = 2;
SimpleDHT11 dht11(pinDHT11);
void setup() {
  Serial.begin(9600);
  Serial.begin(9600);
  while (!Serial) {
    ;
  }
  Serial.print("Initializing SD card...");
    if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while (1);
  }
  Serial.println("card initialized.");
}
void loop() {
  String dataString = "";
  for (int analogPin = 0; analogPin < 3; analogPin++) {
    int sensor = analogRead(analogPin);
    dataString += String(sensor);
    if (analogPin < 2) {
      dataString += ",";
    }
  }
File dataFile = SD.open("datalog.txt", FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
    // print to the serial port too:
    Serial.println(dataString);
  }
   else {
    Serial.println("error opening datalog.txt");
  }
  Serial.println("=================================");
  Serial.println("Sample DHT11...");

 byte temperature = 0;
  byte humidity = 0;
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
    Serial.print("Read DHT11 failed, err="); Serial.print(SimpleDHTErrCode(err));
    Serial.print(","); Serial.println(SimpleDHTErrDuration(err)); delay(1000);
    return;
  }
    Serial.print("Sample OK: ");
  Serial.print((int)temperature); Serial.print(" *C, ");
  Serial.print((int)humidity); Serial.println(" H");
  delay(1500);
}
```

6.8.5 RESULT

By executing thHE program, to store the data at a time of launch are obtained and that will be used for further analysHE.

6.9 ESP32 (CAMERA MODULE)

The ESP32 CAM WiFi Module Bluetooth with OV2640 Camera Module 2MP for video representation has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 40 x 27 mm; a deep sleep current of up to 6mA and HE widely used in various IoT applications. It HE suitable for home smart devices, industrial wireless control, wireless monitoring, and other IoT applications.

The ESP32-CAM HE a small size, low power consumption camera module based on ESP32. It comes with an OV2640 camera and provides onboard TF card slot. The ESP32-CAM can be widely used in intelligent IoT applications such as wireless video monitoring, WiFi image upload, QR identification, and so on.

6.9.1 SPECIFICATION

- Low power 32-bit CPU can also serve the application processor
- Up to 160MHz clock speed, summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, built-in flash lamp

6.9.2 APPLICATION

- RIP motion detection
- Face detection
- Video streaming

6.9.3 DIAGRAM

Fig: 6.9 ESP 32 CAM

6.9.4 TEST CODING FOR SD CARD MODULE

```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
#include "camera_pins.h"

const char* ssid = "Vaayusastra 5G";
const char* password = "changetheworld";
void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif
```

```
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  sensor_t * s = esp_camera_sensor_get();
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  startCameraServer();
  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}


void loop() {
  delay(10000);
}
```

6.9.5 RESULT

By executing thHE program, to capture the video using Wi-Fi Module are obtained.

CHAPTER - 7

PRINTED CIRCUIT BOARD ASSEMBLY

Printed circuit board assembly (PCBA) design brings your electronic circuits to life in the physical form. Using layout software, the PCB design process combines component placement and routing to define electrical connectivity on a manufactured circuit board.

Here we designing a PCB board with proper internal connections with sensors to Arduino nano board given below

1. DTH 11
2. Power Supply
3. SD Card Module
4. Gyro
5. Barometer

7.1 CONNECTION PINS

7.1.1 For DHT 11

- ✓ GND-GND
- ✓ DATA-D2
- ✓ VCC-3V3

7.1.2 For Power Supply

- ✓ GNG-GND
- ✓ VIN-5V

7.1.3 For SD Card Module

- ✓ GND-GND
- ✓ VCC-5V
- ✓ MHEO-D2
- ✓ MOSI-D11
- ✓ SCK-D13
- ✓ CS-D2

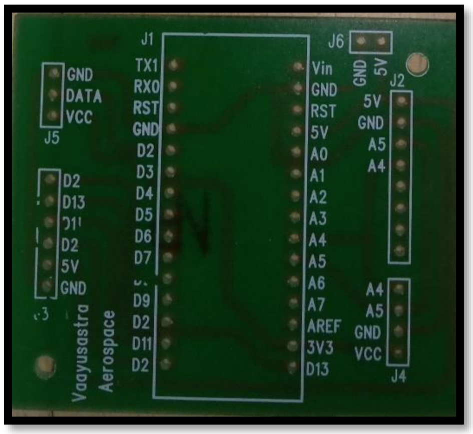
7.1.4 For Gyroscope

- ✓ VCC-5V
- ✓ GND-GND
- ✓ SCL-A5
- ✓ SDA-A4

7.1.5 For Barometer Sensor

- ✓ VCC-3V3
- ✓ GND-GND
- ✓ SCL-A5
- ✓ SDA-A4

7.1.6 DIAGRAM OF PCBA



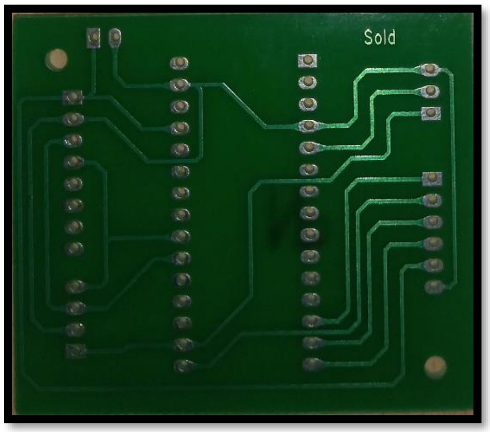


Fig: 7.1 PCBA Board without Connections

CHAPTER - 8

SOLDRING

Soldering HE a process used for joining metal parts to form a mechanical or electrical bond. It typically uses a low melting point metal alloy (solder) which HE melted and applied to the metal parts to be joined and thHE bonds to the metal parts and forms a connection when the solder solidifies. It HE different to welding in that the parts being joined are not melted and are usually not the same material as the solder.

The connecting pins were soldered to the PCB board using a pen type Soldering iron. The ports were first coated with a flux called Rosin, to prevent oxidation. The lead wire HE composed of lead and tin and has melting point of 188 degree Celsius, the iron was heated to a temperature of 315-350 degree Celsius. Also, a part of the wire was first heated with the tip. The soldered port should finally achieve a pyramid-like shape and shouldn’t be like a bubble. Also, make sure the lead stays in the designated port in order to avoid shorting of the circuit board.

8.1 DIAGRAM

Fig: 8.1 Soldering Process of PCBA

CHAPTER - 9

CASING FOR CUBESAT

The Prototype case was made out of coroplast. Coroplast, also called pp plate sheet ("Fluted Polypropylene Sheet"), HE lightweight (hollow structure), non-toxic, waterproof, shockproof, long-lasting material that resHEts corrosion. Compared with cardboard, Coroplast has the advantages of being waterproof and coroplast.

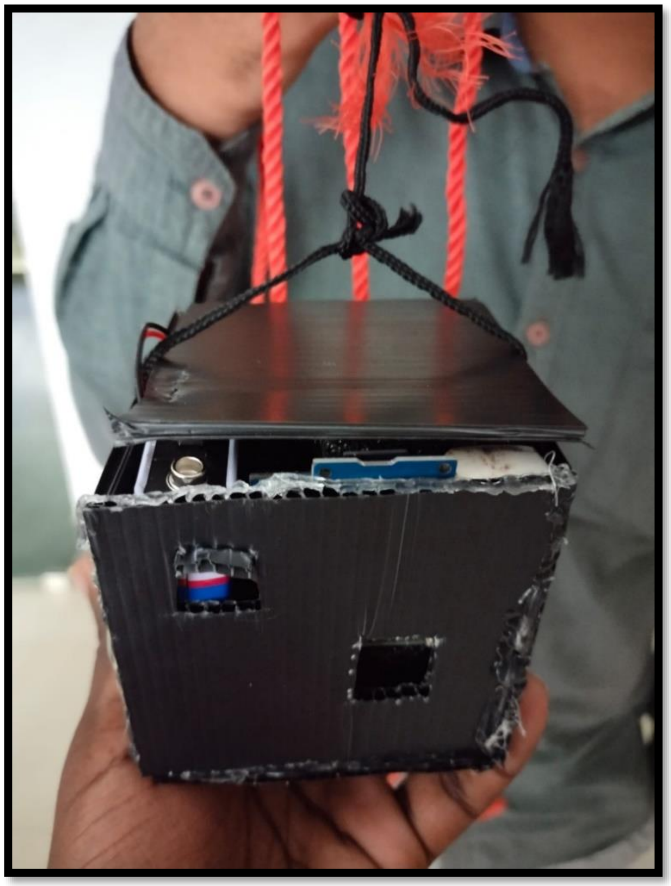


Fig: 9.1 Cubesat Casing

CHAPTER - 10

LAUNCHING OF CUBESAT

By using the above-mentioned sensors and casing finally, we launch the satellite using the Helium Balloon in IITM Research Park Campus and obtained the values with the help of the SD Card Module. We programmed DHT & BMP180 Sensors for measuring the values of Humidity, Temperature and Pressure in some altitude level.

10.1 LAUNCH DETAILS

Launch Date & Time	:	02.08.2022 & 03:00 PM of HET
Launch Place	:	IITM Research Park, Chennai, Tamilnadu, India.
Payload	:	150gms (Approximately)
Reached Altitude	:	140mts (Approximately)
Lifetime	:	15-20mins

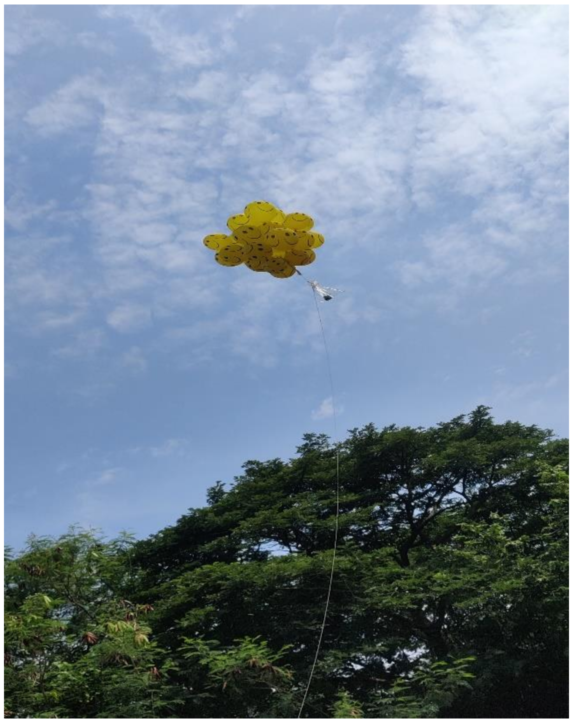
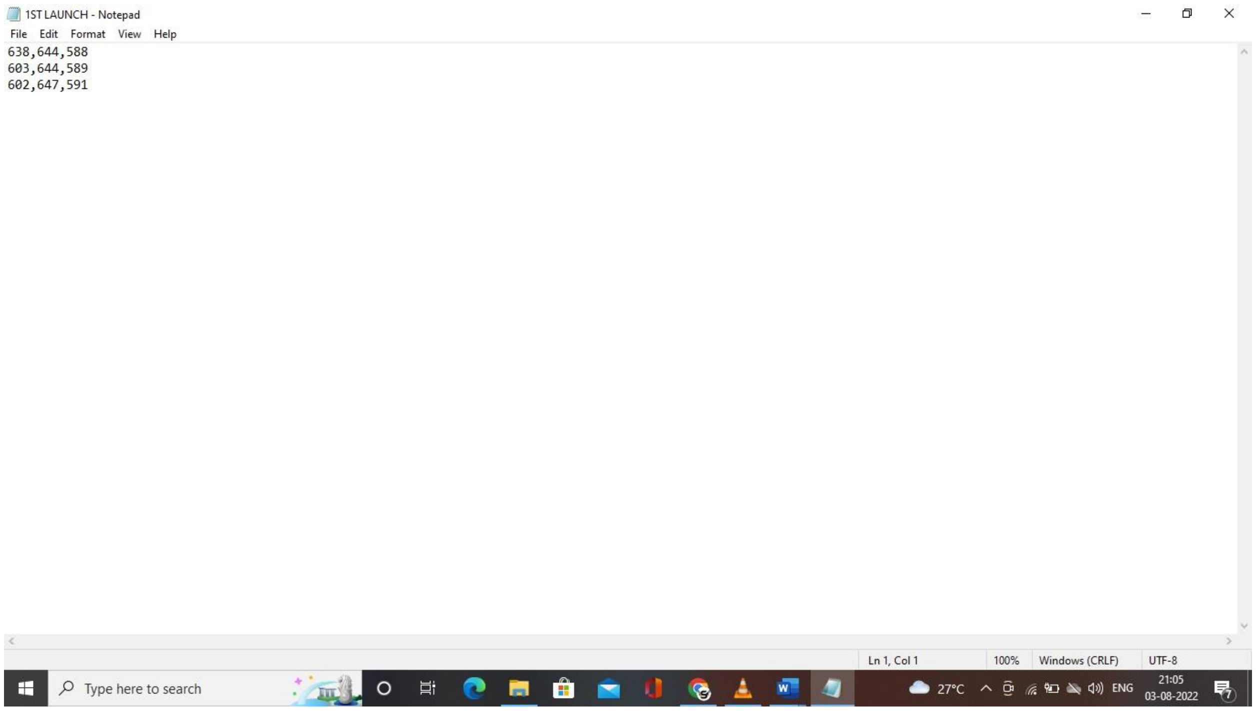


Fig 10.1 Sat in Altitude

Fig: 10.2 Satellite inside casing

10.2 OBTAINED RESULT POST LAUNCH

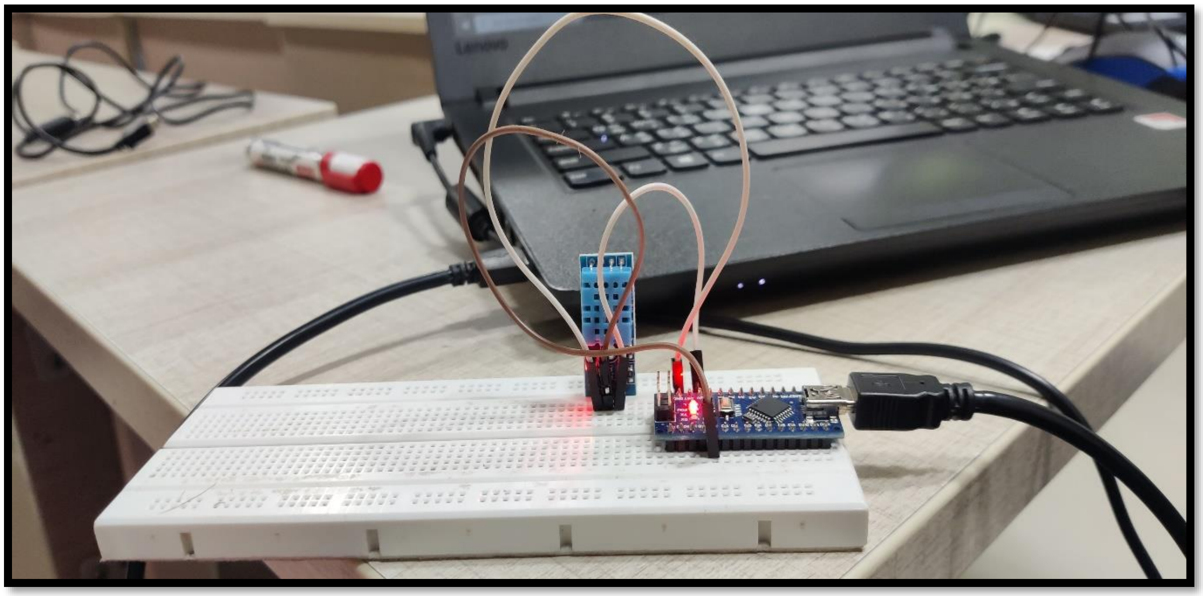


```
638,644,588
603,644,589
602,647,591
```

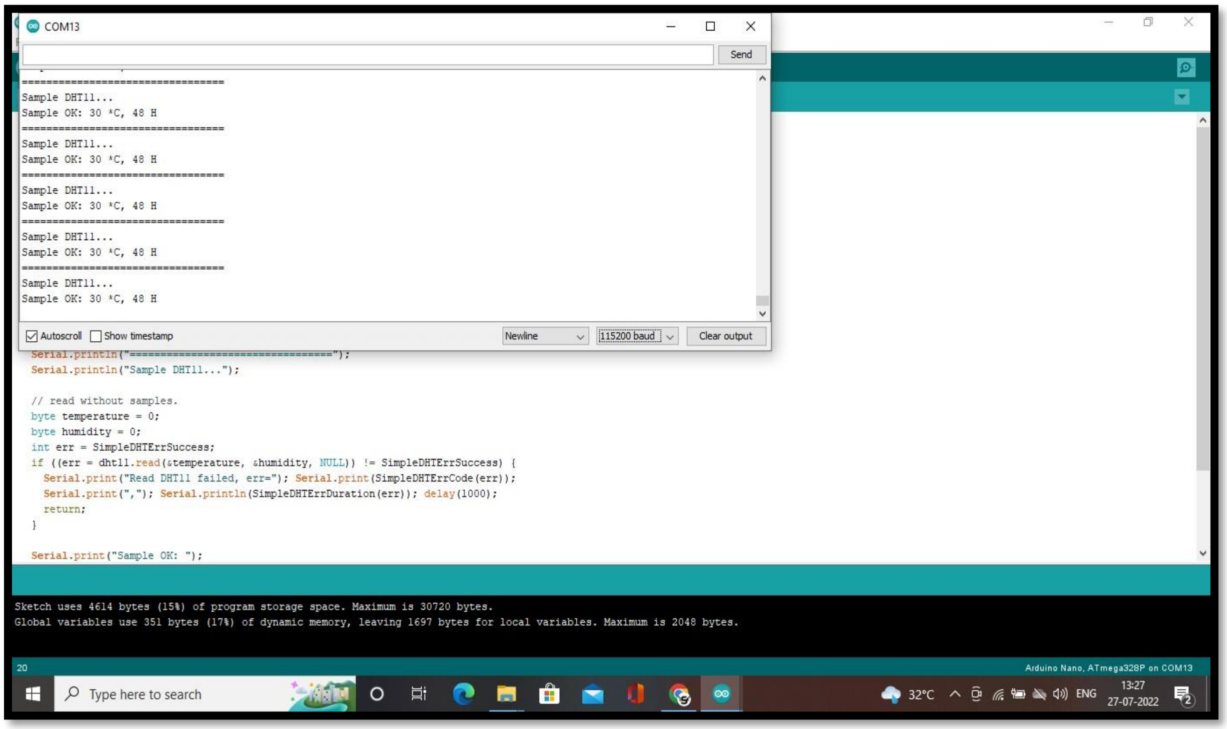
Fig:10.3 DHT & BMP 180 Sensors Result

APPENDIX

DHT 11 SENSOR (Humidity & Temperature)



SENSOR ASSEMBLY



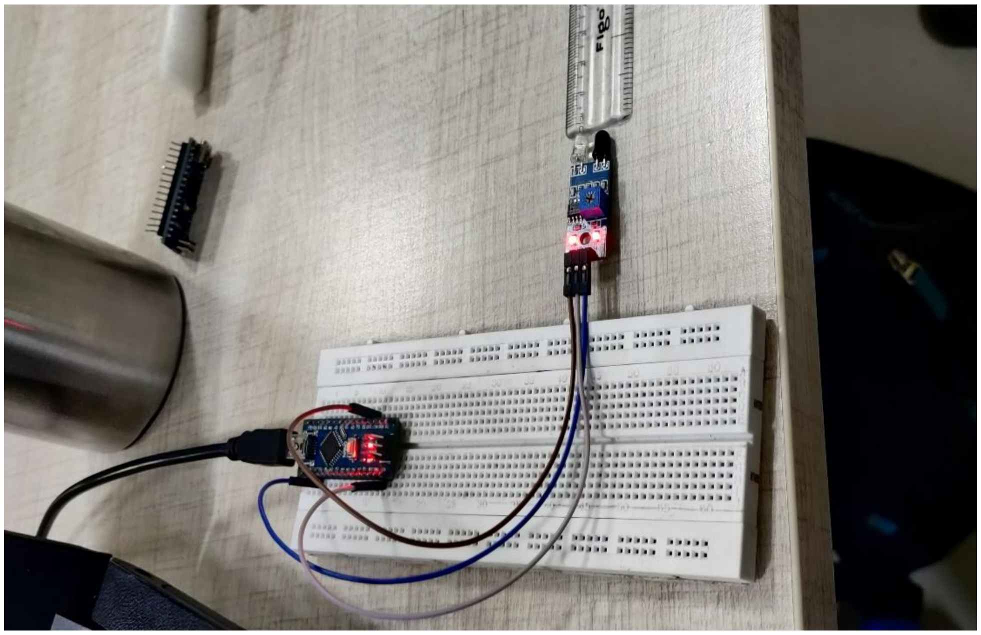
```
==================================
Sample DHT11...
Sample OK: 30 *C, 48 H
==================================
Sample DHT11...
Sample OK: 30 *C, 48 H
==================================
Sample DHT11...
Sample OK: 30 *C, 48 H
==================================
Sample DHT11...
Sample OK: 30 *C, 48 H
==================================
Sample DHT11...
Sample OK: 30 *C, 48 H
```

```
  Serial.println("==================================");
  Serial.println("Sample DHT11...");

  // read without samples.
  byte temperature = 0;
  byte humidity = 0;
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
    Serial.print("Read DHT11 failed, err="); Serial.print(SimpleDHTErrCode(err));
    Serial.print(","); Serial.println(SimpleDHTErrDuration(err)); delay(1000);
    return;
  }

  Serial.print("Sample OK: ");
```

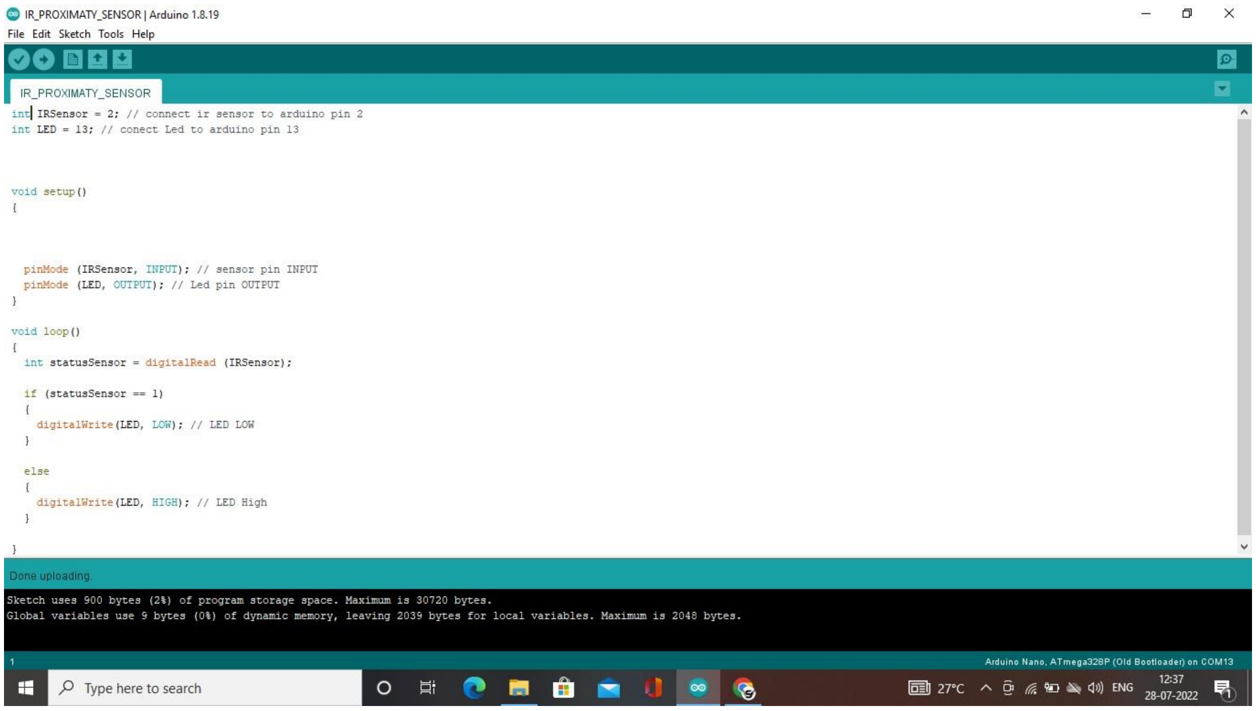
```
Sketch uses 4614 bytes (15%) of program storage space. Maximum is 30720 bytes.
Global variables use 351 bytes (17%) of dynamic memory, leaving 1697 bytes for local variables. Maximum is 2048 bytes.
```

CODING OUTPUT

IR PROXIMATY SENSOR (WEATHER MONITORING)



SESNOR ASSEMBLY



IR_PROXIMATY_SENSOR | Arduino 1.8.19
File Edit Sketch Tools Help
IR_PROXIMATY_SENSOR

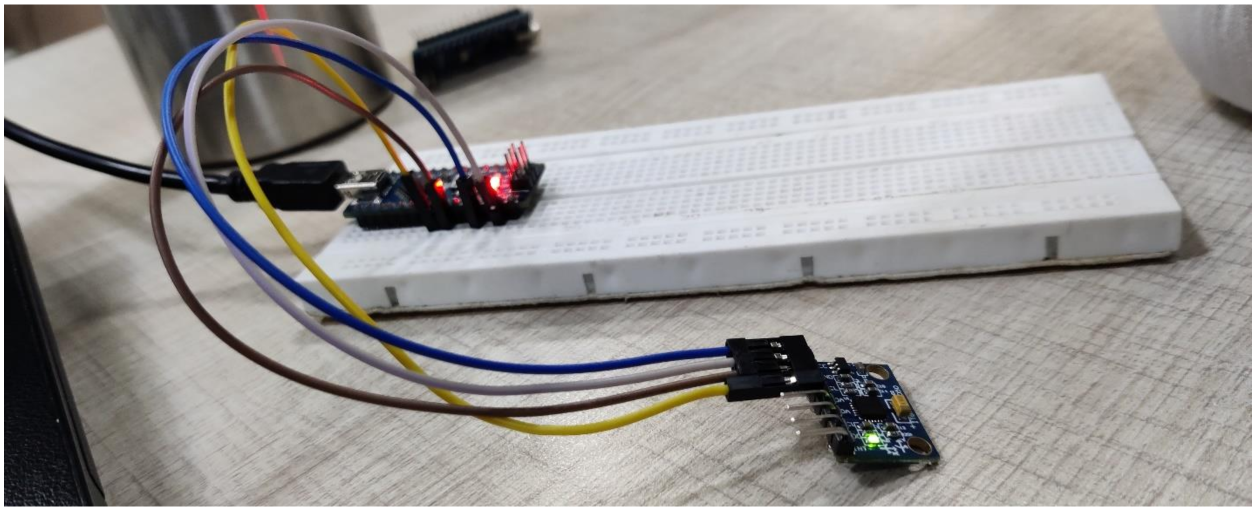
```
int IRSensor = 2; // connect ir sensor to arduino pin 2
int LED = 13; // conect Led to arduino pin 13



void setup()
{



  pinMode (IRSensor, INPUT); // sensor pin INPUT
  pinMode (LED, OUTPUT); // Led pin OUTPUT
}

void loop()
{
  int statusSensor = digitalRead (IRSensor);

  if (statusSensor == 1)
  {
    digitalWrite(LED, LOW); // LED LOW
  }

  else
  {
    digitalWrite(LED, HIGH); // LED High
  }

}
```

Done uploading.

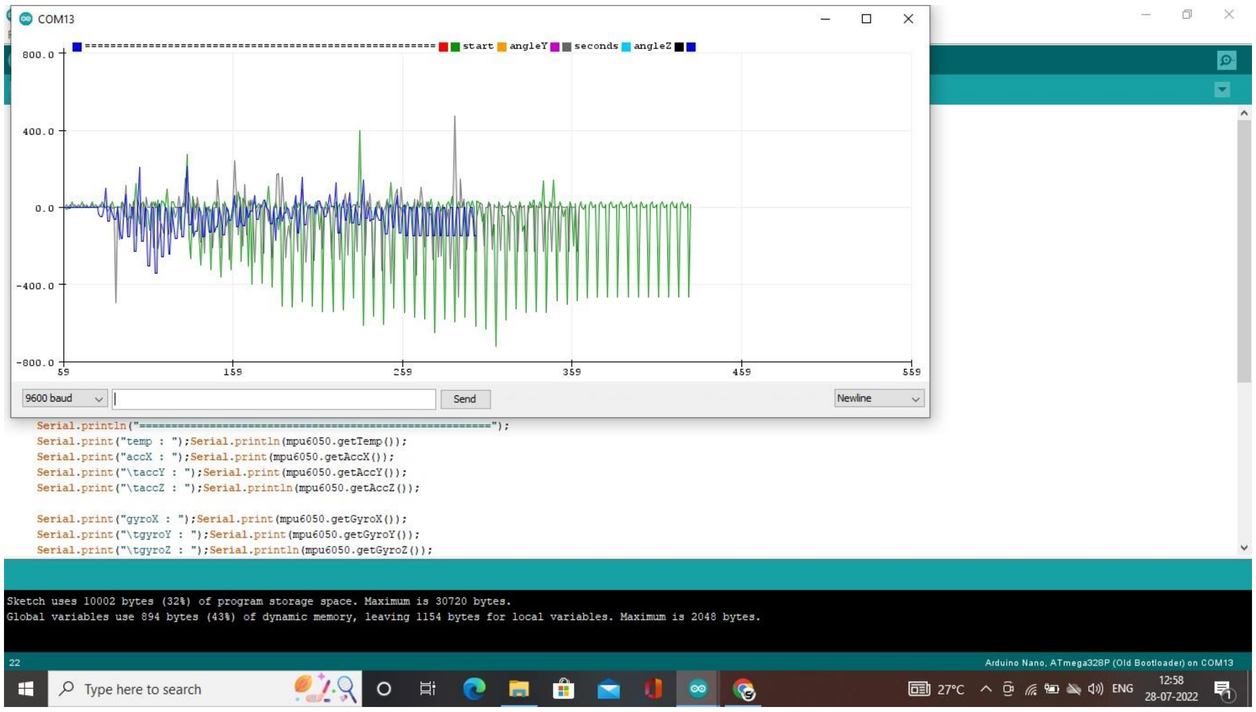
```
Sketch uses 900 bytes (2%) of program storage space. Maximum is 30720 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

CODING OUTPUT

GYRO MPU6050 SENSOR (DETECTING POSITION)



SESNOR ASSEMBLY

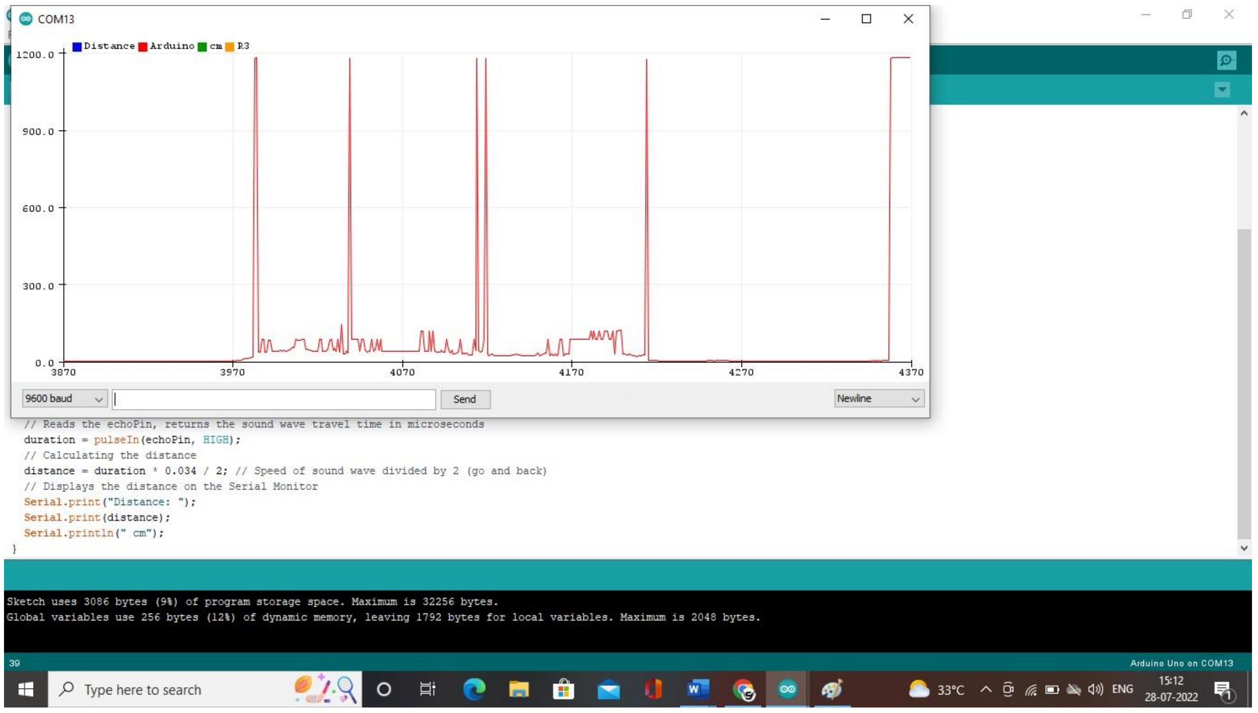

COM13
== start angleY seconds angleZ
800.0
400.0
0.0
-400.0
-800.0
59
159
259
359
459
559
9600 baud
Send
Newline
22
Arduino Nano, ATmega328P (Old Bootloader) on COM13
Type here to search
27°C
ENG
12:58
28-07-2022

```
    Serial.println("==========================================================");
    Serial.print("temp : ");Serial.println(mpu6050.getTemp());
    Serial.print("accX : ");Serial.print(mpu6050.getAccX());
    Serial.print("\taccY : ");Serial.print(mpu6050.getAccY());
    Serial.print("\taccZ : ");Serial.println(mpu6050.getAccZ());

    Serial.print("gyroX : ");Serial.print(mpu6050.getGyroX());
    Serial.print("\tgyroY : ");Serial.print(mpu6050.getGyroY());
    Serial.print("\tgyroZ : ");Serial.println(mpu6050.getGyroZ());
```

```
Sketch uses 10002 bytes (32%) of program storage space. Maximum is 30720 bytes.
Global variables use 894 bytes (43%) of dynamic memory, leaving 1154 bytes for local variables. Maximum is 2048 bytes.
```

CODING OUTPUT

ULTRASONIC SENSOR

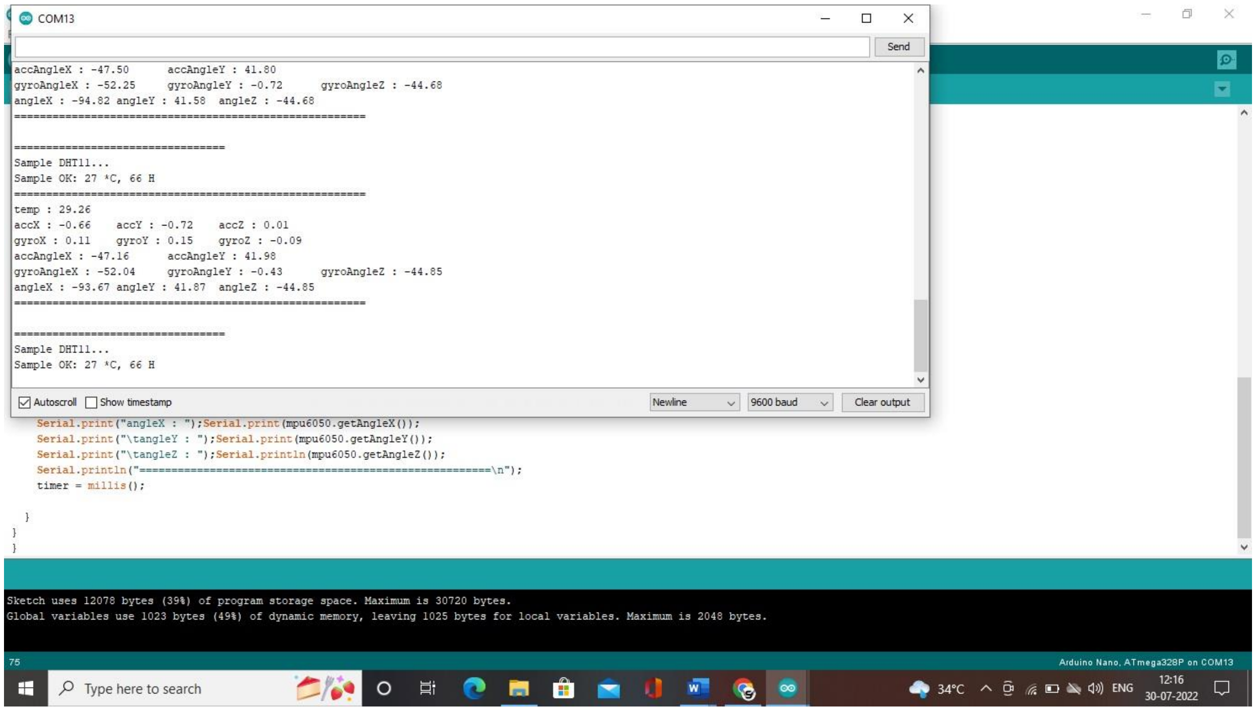


```
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

```
Sketch uses 3086 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 256 bytes (12%) of dynamic memory, leaving 1792 bytes for local variables. Maximum is 2048 bytes.
```

CODING OUTPUT

COMBINED GYRO & DHT SENSOR



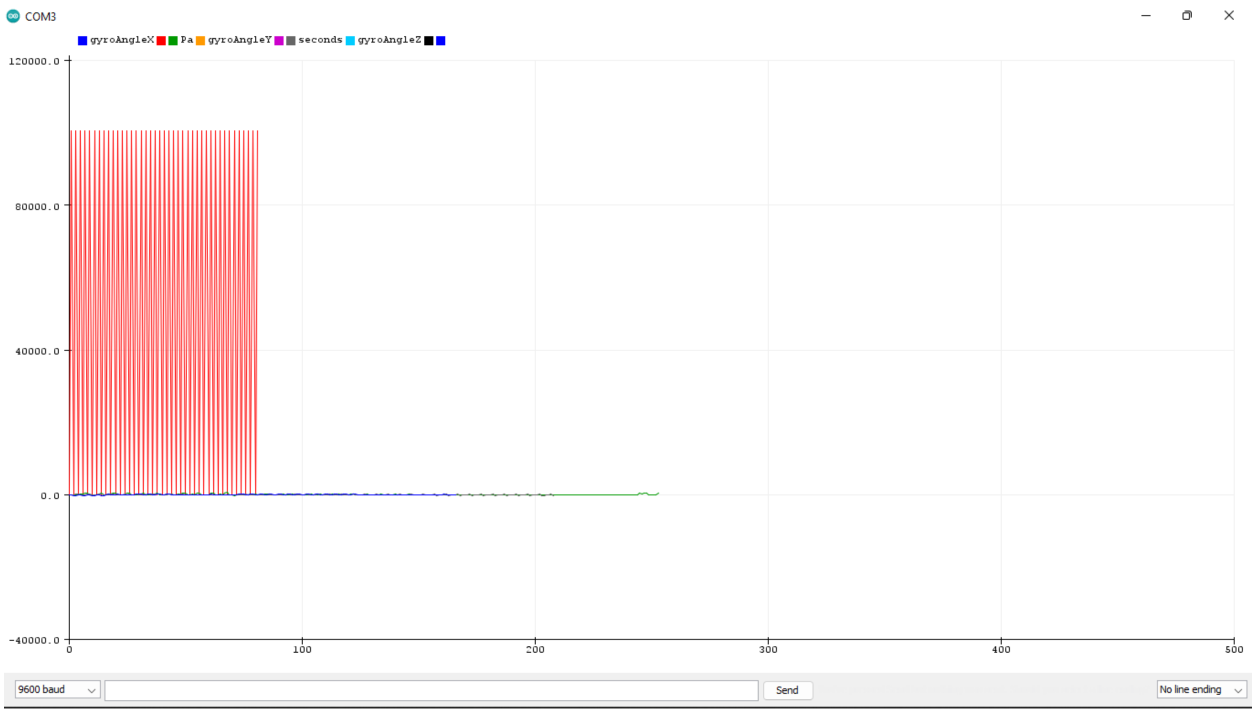
```
accAngleX : -47.50      accAngleY : 41.80
gyroAngleX : -52.25     gyroAngleY : -0.72      gyroAngleZ : -44.68
angleX : -94.82 angleY : 41.58  angleZ : -44.68
=======================================================

=================================
Sample DHT11...
Sample OK: 27 *C, 66 H
=======================================================
temp : 29.26
accX : -0.66    accY : -0.72    accZ : 0.01
gyroX : 0.11    gyroY : 0.15    gyroZ : -0.09
accAngleX : -47.16      accAngleY : 41.98
gyroAngleX : -52.04     gyroAngleY : -0.43      gyroAngleZ : -44.85
angleX : -93.67 angleY : 41.87  angleZ : -44.85
=======================================================

=================================
Sample DHT11...
Sample OK: 27 *C, 66 H
```

```
    Serial.print("angleX : ");Serial.print(mpu6050.getAngleX());
    Serial.print("\tangleY : ");Serial.print(mpu6050.getAngleY());
    Serial.print("\tangleZ : ");Serial.println(mpu6050.getAngleZ());
    Serial.println("=======================================================\n");
    timer = millis();

  }
}
}
```

```
Sketch uses 12078 bytes (39%) of program storage space. Maximum is 30720 bytes.
Global variables use 1023 bytes (49%) of dynamic memory, leaving 1025 bytes for local variables. Maximum is 2048 bytes.
```

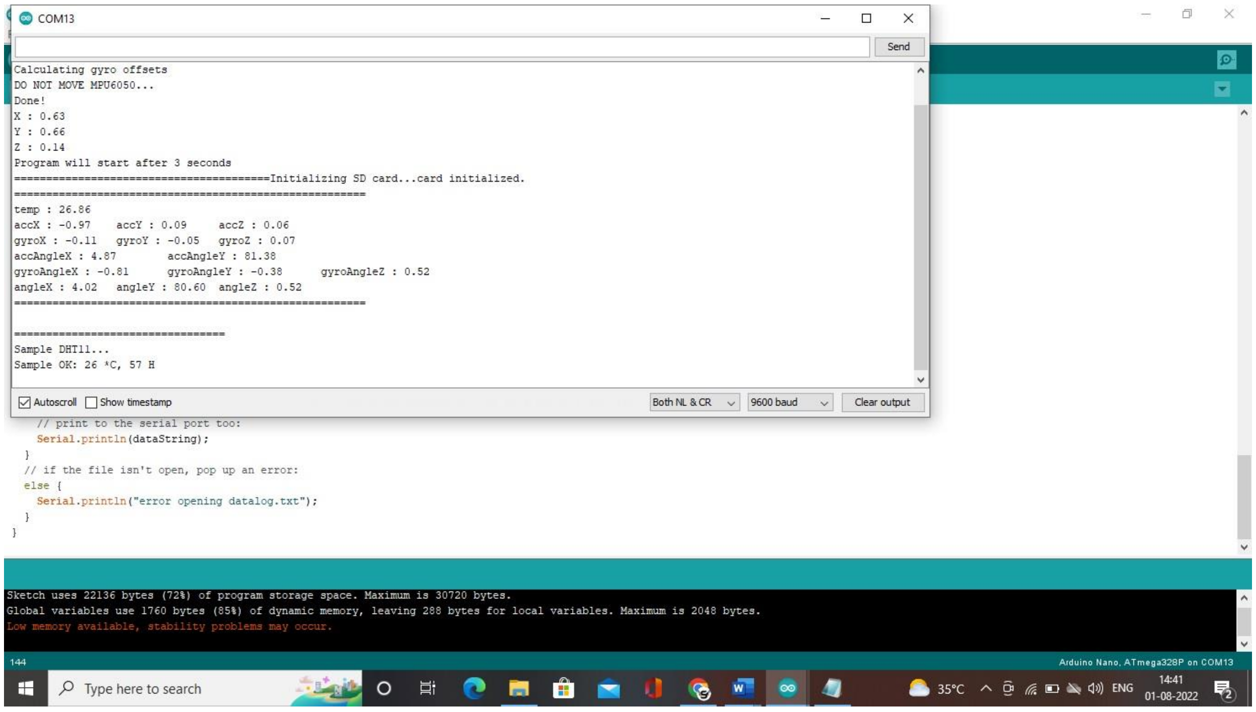
CODING OUTPUT

COMBINED GYRO & BMP



CODING OUTPUT

SD CARD MODULE (GYRO & DHT)



```
Calculating gyro offsets
DO NOT MOVE MPU6050...
Done!
X : 0.63
Y : 0.66
Z : 0.14
Program will start after 3 seconds
========================================Initializing SD card...card initialized.
=======================================================
temp : 26.86
accX : -0.97    accY : 0.09     accZ : 0.06
gyroX : -0.11   gyroY : -0.05   gyroZ : 0.07
accAngleX : 4.87        accAngleY : 81.38
gyroAngleX : -0.81      gyroAngleY : -0.38      gyroAngleZ : 0.52
angleX : 4.02   angleY : 80.60  angleZ : 0.52
=======================================================

=================================
Sample DHT11...
Sample OK: 26 *C, 57 H
```

Autoscroll [✓] Show timestamp [ ] Both NL & CR 9600 baud

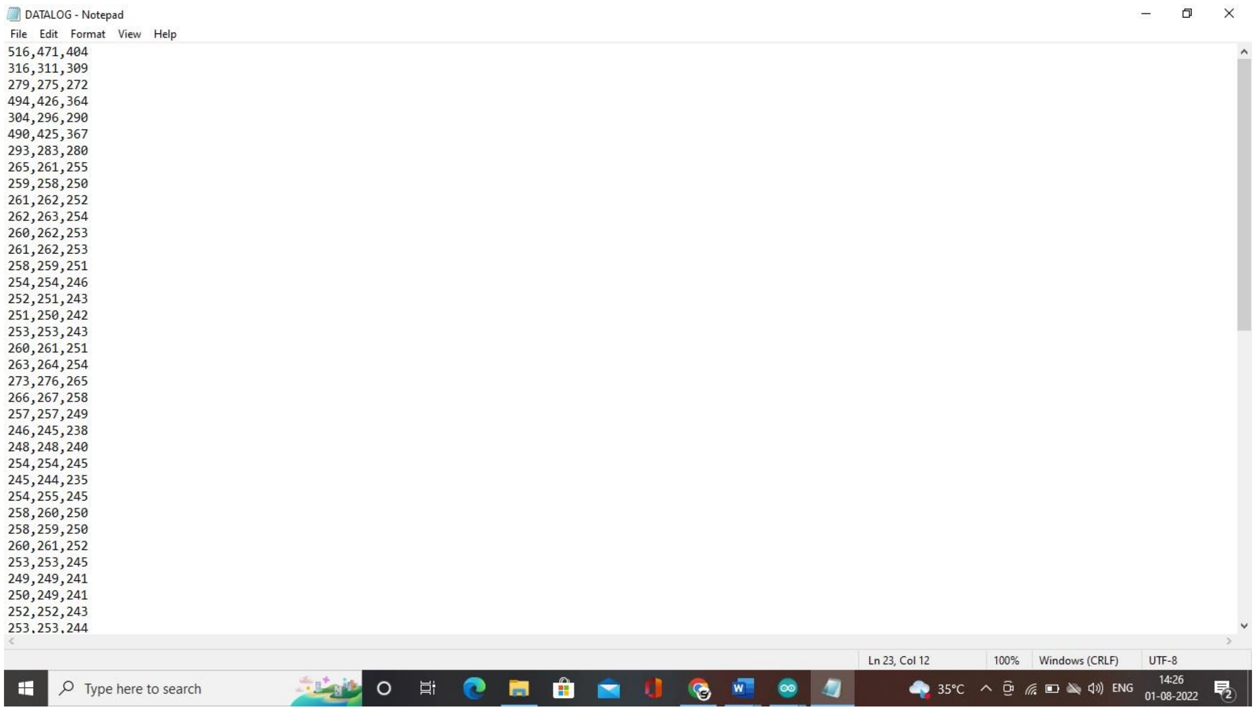
```
    // print to the serial port too:
    Serial.println(dataString);
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println("error opening datalog.txt");
  }
}
```

```
Sketch uses 22136 bytes (72%) of program storage space. Maximum is 30720 bytes.
Global variables use 1760 bytes (85%) of dynamic memory, leaving 288 bytes for local variables. Maximum is 2048 bytes.
Low memory available, stability problems may occur.
```

144 Arduino Nano, ATmega328P on COM13

CODING OUTPUT

SD CARD OUTPUT (GYRO & DHT)


DATALOG - Notepad

```
516,471,404
316,311,309
279,275,272
494,426,364
304,296,290
490,425,367
293,283,280
265,261,255
259,258,250
261,262,252
262,263,254
260,262,253
261,262,253
258,259,251
254,254,246
252,251,243
251,250,242
253,253,243
260,261,251
263,264,254
273,276,265
266,267,258
257,257,249
246,245,238
248,248,240
254,254,245
245,244,235
254,255,245
258,260,250
258,259,250
260,261,252
253,253,245
249,249,241
250,249,241
252,252,243
253,253,244
```

SERIAL MONITOR OUTPUT