A PROJECT REPORT ON

DATA ACQUISITION SYSTEM USING MICROCONTROLLER AND RTOS

SUBMITTED IN
PARTIAL FULFILLMENT OF

DIPLOMA IN EMBEDDED SYSTEM DESIGN (PG-DESD)

BY

Miss. NIKITA MALI	**230944230049**
Miss. RANI PARMAR	**230944230081**
Miss. ADITI RAUT	**230944230003**
Miss. MRUNMAYEE KHATGAONKAR	**230944230059**

AT

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, HINJEWADI, PUNE

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, HINJEWADI

CERTIFICATE

This is to certify that the project

DATA ACQUISITION SYSTEM USING MICROCONTROLLER AND RTOS

Has been submitted by

Miss. NIKITA MALI	**230944230049**
Miss. RANI PARMAR	**230944230081**
Miss. ADITI RAUT	**230944230003**
Miss. MRUNMAYEE KHATGAONKAR	**230944230059**

In partial fulfilment of the requirement for the Course of PG Diploma in Embedded System Design (PG-DESD SEP 2023) as prescribed by The CDAC ACTS, PUNE.

Place: Hinjewadi **Date:** 22nd February 2024

Mr.
Project Guide **Authorized Signature**

ACKNOWLEDGEMENT

It is indeed a matter of great pleasure and proud privilege to be able to present this project on **“Data Acquisition System Using Microcontroller and RTOS”.**

The completion of this project work is great experience in student’s life and its execution is inevitable in hands of guide. We are highly indebted to the Project Guide Mr.   for his invaluable guidance and appreciation for giving form and substance to this project. It is due to his enduring efforts, patience and enthusiasm which has given sense of direction and purposefulness to this project and alternately made it a success.

I also wish to express my sincere thanks to the CEO of Sunbeam Institute of Information Technology Pune **Mr. Nitin Kudale** for their immense planning, continuous guidance. Also, I am grateful to all the faculty members specially **Mr. Devendra Dhande**, **Mr. Nilesh Ghule**, for their consistent support and assistance.

Thanks for all your encouragement!

CONTENTS

ABSTRACT

The project proposes the development of a sophisticated Data Acquisition System, integrating a microcontroller, real-time operating system (RTOS), and a host computer to facilitate real-time monitoring and analysis of environmental parameters. The chosen microcontroller is the STM32F407VG, implementing FreeRTOS for efficient task management. The system employs various sensors, including temperature (DHT22), gas (MQ5), infrared (HW201), and sound sensors (KY038), collectively providing a comprehensive understanding of the surrounding environment. The central focus of the project is on real-time Analog-to-Digital Conversion (ADC) of sensor data through FreeRTOS on the STM32F407VG microcontroller. The ADC values, representing the diverse environmental parameters, are then transmitted to a connected computer via UART communication. On the computer side, FreeRTOS is ported to manage the received data efficiently. Graphical representation and visualization of the acquired data is achieved through the integration of the ThingSpeak library. The ThingSpeak application reads the data from the UART and dynamically plots the ADC values on a graph. This graphical representation provides a user-friendly and intuitive means of interpreting the real-time variations in temperature, gas concentration, infrared radiation, and sound levels. The diverse range of sensors contributes to the versatility of the data acquisition system, making it suitable for applications such as environmental monitoring, industrial automation, and research. The temperature sensor captures ambient temperature, the gas sensor detects gas concentrations the infrared sensor detects infrared radiation, and the sound sensor captures acoustic levels in the environment. The integration of FreeRTOS ensures that the microcontroller efficiently manages the acquisition tasks, enabling real-time responses to environmental changes. The use of UART communication facilitates seamless data transfer between the microcontroller and the computer. FreeRTOS on the computer side ensures the timely and organized processing of the received sensor data, enhancing the overall responsiveness of the system. In summary, this project presents a comprehensive and versatile Data Acquisition System, leveraging the capabilities of FreeRTOS on both the microcontroller and the computer. The inclusion of diverse sensors and the graphical representation of data through ThingSpeak application enhance the system's applicability in real-time monitoring scenarios, offering insights into environmental parameters for a wide range of potential applications.

INTRODUCTION

In the rapidly evolving landscape of technological advancements, real-time data acquisition systems play a pivotal role in monitoring and analyzing diverse environmental parameters. A data acquisition system is a system that comprises sensors, measurement devices, and a computer. A data acquisition system is used for processing acquired data, which involves collecting the information required to understand electrical or physical phenomena.

This project explores the development of a sophisticated data acquisition system, integrating a microcontroller and a real-time operating system (RTOS) to facilitate the concurrent monitoring of temperature, pressure, sound, and infrared parameters. The chosen microcontroller, the STM32 Discovery board, serves as the central processing unit, while FreeRTOS and handle task management on the microcontroller and computer sides, respectively. The motivation behind this project stems from the growing demand for comprehensive sensor-based monitoring solutions across various domains. Industries, research institutions, and even home automation systems require efficient real-time data acquisition to ensure timely decision-making and analysis. The integration of multiple sensors, each measuring distinct environmental variables, enhances the system's versatility and applicability.

The fundamental objective is to design a robust system that not only acquires data in real-time but also ensures seamless communication between the microcontroller and a computer. The selected sensors—DHT22 for temperature, MQ5 for gas, KY038 for sound, and an infrared sensor HW201 —contribute to a diverse set of data sources, enabling the system to cater to a wide array of applications. The chosen microcontroller, the STM32 Discovery board, provides a powerful and versatile platform for embedded systems. FreeRTOS, a real-time operating system, is ported onto the microcontroller to manage concurrent tasks associated with each sensor. This ensures efficient data collection without compromising system responsiveness.

On the computer side, the FreeRTOS framework is employed to receive, manage, and process the data transmitted by the microcontroller. This real-time framework enhances the precision and predictability of data handling, crucial for applications where timing is paramount.

The graphical user interface (GUI) implemented using the ThingSpeak Library serves as the visual representation of the acquired data. Through a seamless integration of UART and communication protocols, the system transmits ADC values to the computer, where ThingSpeak plots the data on a graph. This graphical representation enhances the interpretability of the data, allowing users to glean insights at a glance.

This project entails the development of a real-time data acquisition system using an STM32 Discovery board with FreeRTOS for microcontroller task management and computer-side real-time processing. The system achieves efficient communication via UART, enabling the transfer of ADC values to the computer. The graphical representation of data using ThingSpeak provides a user-friendly interface. Applications include environmental monitoring, industrial automation, smart home systems, and research.

BASIC COMPONENTS OF DATA ACQUISITION SYSTEM

The physical phenomena or physical characteristics to be measured comes first in the data collecting process. Temperature, light intensity, vibration, gas pressure, fluid movement, and force are a few examples of factors often considered in a DAQ system. No matter what kind of physical property has to be measured, the physical state must first be unified into a form that a data acquisition system can sample.

These alterations are carried out by sensors. An ensemble of software and hardware known as a data acquisition system enables the measurement or control of physical properties of objects in the real world. A full data acquisition system consists of DAQ hardware, sensors, actuators, signal conditioning gear, and a computer running DAQ software. Furthermore, an independent timing system must be used if timing is important (for example, in event mode DAQ systems).

Sensors: - Sensors or transducers serve the purpose of interacting with the subject measured. These tools convert the physical values to produce an output of electrical signals.

Transmission / Signal Conditioners: - Signal conditioning is the process of improving electrical signals obtained from sensors by addressing issues such as noise, interference, and signal weakness. This involves using additional circuitry, known as a signal conditioner, to enhance the quality and reliability of the signals, making them suitable for accurate measurement and analysis by a data acquisition system.

Data Acquisition Hardware: - Data acquisition hardware is the hardware that is connected between the sensors and the computer. The data acquisition hardware serves to take in the signals from the sensors and then convert them into digital signals that are readable by the computer.

Analog-to-Digital Converters: - This component of the DAQ system serves to convert analog signals into digital signals. This component is at the core of all data acquisition systems. This chip serves to take data from the environment and convert it into separate levels that can be interpreted by a processor.

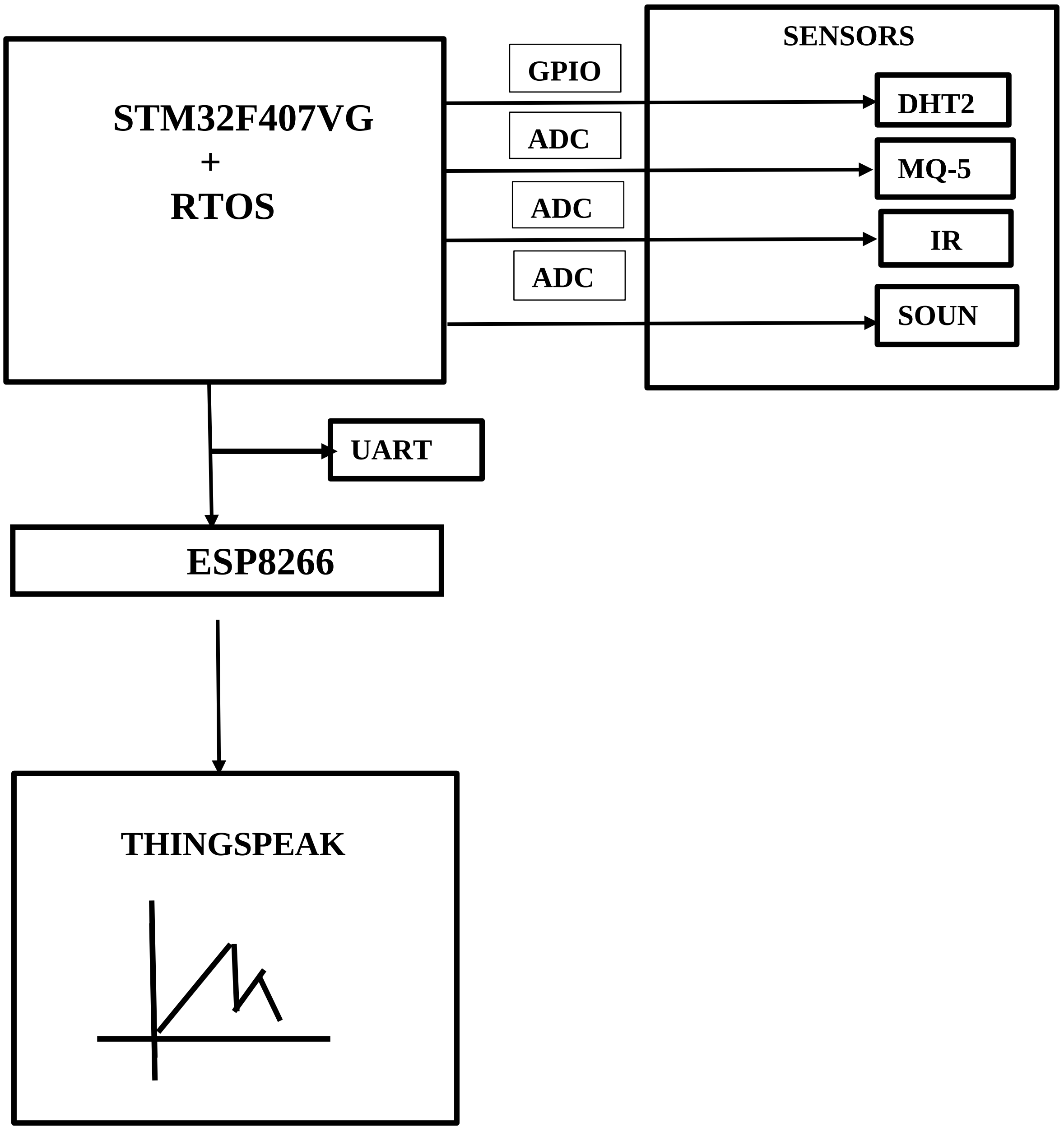
Advantages: -

1. High efficiency and Reliability of Processes
2. Faster Analysis and Resolution of Problems
3. Reduced Data Redundancy
4. Cost Effectiveness
5. Quality Control

MOTIVATION

The motivation behind selecting this project lies in the recognition of the pressing need for advanced, real-time data acquisition systems in today's technological landscape. As industries and research sectors increasingly rely on data-driven insights, the development of a robust system capable of seamlessly collecting and processing information from a variety of sensors becomes crucial. The project not only addresses this demand but also serves as an educational endeavor, providing a hands-on opportunity to explore and apply concepts related to microcontroller programming, real-time operating systems, and sensor integration. The integration of open-source frameworks like FreeRTOS adds a layer of innovation and collaboration to the project, aligning with the ethos of community-driven development. Furthermore, the inclusion of diverse sensors, such as temperature, pressure, sound, and infrared, enhances the project's versatility and prepares individuals for interdisciplinary applications. Ultimately, the motivation stems from the desire to contribute to technological advancements, foster practical skills development, and align with the evolving needs of industries relying on cutting-edge data acquisition solutions.

BLOCK DIAGRAM



COMPONENTS USED IN THE PROJECT

A. STM32 Microcontroller

The STM32F20x family is based on the high-performance Arm® Cortex®-M4 32-bit RISC core operating at a frequency of up to 120 MHz. The family incorporates high-speed embedded memories (Flash memory up to 1 Mbyte, up to 128 Kbytes of system SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, three AHB buses and a 32-bit multi-AHB bus matrix. All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers. a true number random generator (RNG). They also feature standard and advanced communication interfaces.

Features:

a. Up to three I2Cs
b. Three SPIs
c. To achieve audio class accuracy, the I2S peripherals can be clocked via a dedicated internal audio PLL or via an external PLL to allow synchronization.
d. Four USARTs and two UARTs
e. A USB OTG high-speed with full-speed capability
f. A second USB OTG (full-speed)
g. Two CANs

The STM32F205xx and STM32F207xx devices operate in **the –40 to +105 °C** temperature range from a **1.8 V to 3.6 V power supply**. STM32F205xx and STM32F207xx devices are offered in various packages, ranging from 64 to 176 pins.

These features make the STM32F205xx and STM32F207xx microcontroller family suitable for a wide range of applications:

- Motor drive and application control
- Medical equipment
- Industrial applications: PLC, inverters, circuit breakers
- Alarm systems, video intercom, and HVAC
- Home audio appliances

B. FreeRTOS

FreeRTOS is a collection of C libraries comprised of a real-time kernel and a set of modular libraries that implement complementary functionality. The FreeRTOS kernel is ideally suited to deeply embedded real-time applications that run on microcontrollers or small microprocessors. This type of application typically includes a mix of both hard and soft real-time requirements. Soft real-time requirements state a time deadline and Hard real-time requirements state a time deadline.

Why to use an RTOS?

1. Abstracting away timing information: The RTOS is responsible for execution timing and provides a time-related API to the application and this will allow the structure of the application code to be more straightforward and will lead to smaller code size.
2. Maintainability/Extensibility: Abstracting away timing details results in fewer interdependencies between modules and allows the software to evolve in a controlled and predictable way.
3. Modularity: Tasks are independent modules, each of which should have a well-defined purpose.

4. Easier testing: Tasks that are well-defined independent modules with clean interfaces are easier to test in isolation.
5. Code reuse: Code designed with greater modularity and fewer interdependencies is easier to reuse.
6. Improved efficiency: Application code that uses an RTOS can be completely event-driven. Countering the efficiency gained from being event driven is the need to process the RTOS tick interrupt and switch execution from one task to another.
7. Idle time: The automatically created Idle task executes when there are no application tasks that require processing.
8. Power Management: The efficiency gains that result from using an RTOS allow the processor to spend more time in a low power mode. Power consumption can be decreased significantly by placing the processor into a low power state each time the Idle task runs.

There are two separate FreeRTOS ports for the Cortex-M3:

a. FreeRTOS-MPU: includes full Memory Protection Unit (MPU) support. In this version, tasks can execute in either User mode or Privileged mode. Also, access to Flash, RAM, and peripheral memory regions can be tightly controlled, on a task-by-task basis.

b. FreeRTOS (the original port) This does not include any MPU support. All tasks execute in the Privileged mode and can access the entire memory map.

Resources Used by FreeRTOS: FreeRTOS makes use of the Cortex-M3 SysTick, PendSV, and SVC interrupts. FreeRTOS has a very small footprint. A typical kernel build will consume approximately 6K bytes of Flash space and a few hundred bytes of RAM. Each task also requires RAM to be allocated for use as the task stack.

FreeRTOS uses a modified GPL license. The modification is included to ensure:

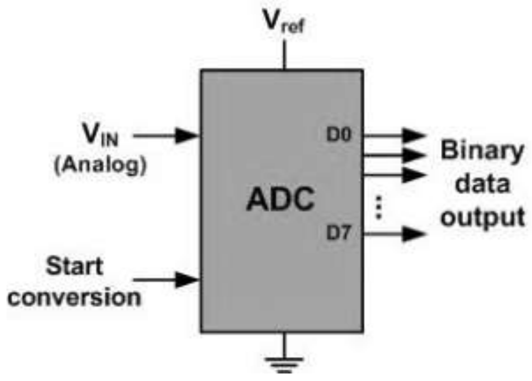
1. FreeRTOS can be used in commercial applications.

2. FreeRTOS itself remains open source.

3. FreeRTOS users retain ownership of their intellectual property.

C. Analog – to – Digital Converter:

An ADC (Analog-To-Digital) converter is an electronic circuit that takes in an analog voltage as input and converts it into digital data, a value that represents the voltage level in binary code. The ADC samples the analog input whenever you trigger it to start conversion. It performs a process called quantization to decide on the voltage level and its binary code that gets pushed into the output register.



Major Characteristics of ADC:

a. Resolution: The ADC has n-bit resolution, where n can be 8, 10, 12, 16, or even 24 bits. Higher-resolution ADCs provide a smaller step size, where step size is the smallest change that can be discerned by an ADC.
b. Vref: Input voltage used for the reference voltage. The voltage connected to this pin, along with the resolution of the ADC chip, determine the step size. For an 8-bit ADC, the step size is V_{ref} / 256 because it is an 8-bit ADC, and 2 to the power of 8 gives us 256 steps.
c. Conversion Time: the time it takes the ADC to convert the analog input to a digital number.
d. Digital Data Output: In an 8-bit ADC we have an 8-bit digital data output of D0–D7, while in the 10-bit ADC the data output is D0–D9. To calculate the output voltage, we use the following formula:

$$D_{OUT} = V_{IN} / \text{Step size}$$

where D_{out} = digital data output (in decimal),

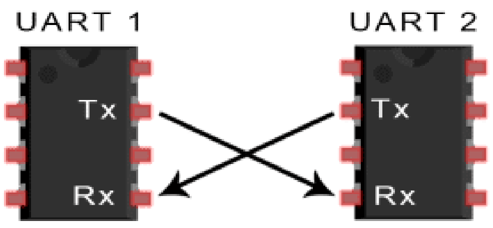
V_{in} = analog input voltage

e. Analog Input Channels: Many data acquisition applications need more than one analog input for ADC. For this reason, we see ADC chips with 2, 4, 8, or even 16 channels on a single chip.

D. UART (Universal Asynchronous Receiver Transmitter)

The STM32F20x devices embed four universal synchronous/asynchronous receiver transmitters (USART1, USART2, USART3 and USART6) and two universal asynchronous receiver transmitters (UART4 and UART5). The USART1 and USART6 interfaces are able to communicate at speeds of up to 7.5 Mbit/s. The other available interfaces communicate at up to 3.75 Mbit/s. One of the best things about UART is that it only uses two wires to transmit data between devices.

In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART:

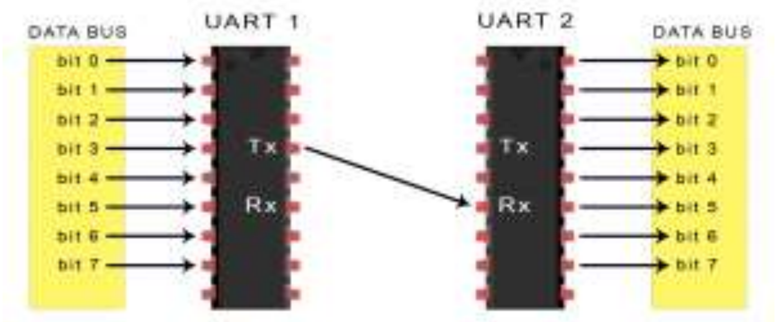


UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits.

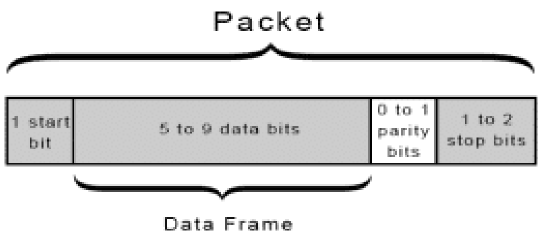
When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off.

Working of UART:

The UART that is going to transmit data receives the data from a data bus. The data bus is used to send data to the UART by another device like a CPU, memory, or microcontroller. Data is transferred from the data bus to the transmitting UART in parallel form. After the transmitting UART gets the parallel data from the data bus, it adds a start bit, a parity bit, and a stop bit, creating the data packet. Next, the data packet is output serially, bit by bit at the Tx pin. The receiving UART reads the data packet bit by bit at its Rx pin. The receiving UART then converts the data back into parallel form and removes the start bit, parity bit, and stop bits. Finally, the receiving UART transfers the data packet in parallel to the data bus on the receiving end:



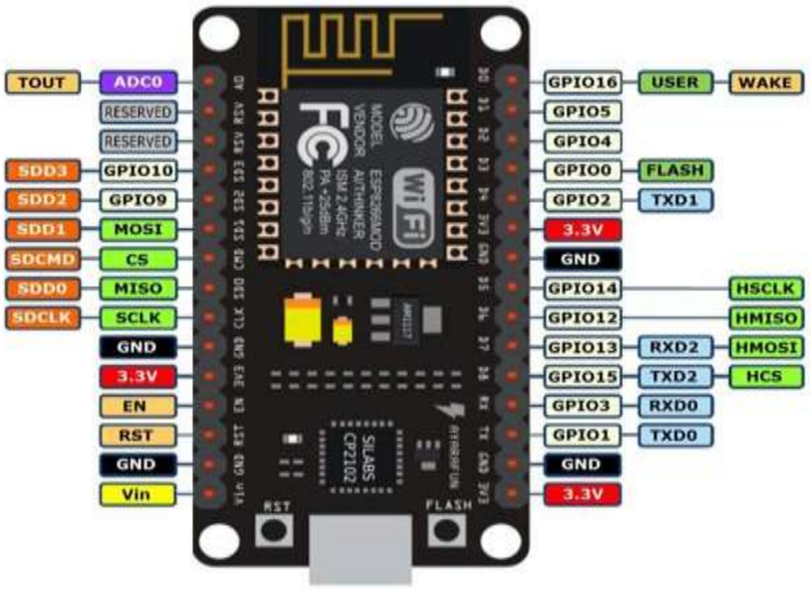
UART transmitted data is organized into packets. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional parity bit, and 1 or 2 stop bits:



Advantages:

- Only uses two wires.
- No clock signal is necessary.
- Has a parity bit to allow for error checking.

E. NODEMCU ESP8266



The NodeMCU (***N***ode ***M***icro***C***ontroller ***U***nit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port.

POWER PINS:

There are four power pins. **VIN** pin and three **3.3V** pins.

- **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the **VIN** pin
- **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.

GND are the ground pins of NodeMCU/ESP8266

I2C Pins:

are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins:

NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channels:

The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins:

NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins:

NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO

PWM Pins:

The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).

Control Pins:

are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

- **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- **RST:** RST pin is used to reset the ESP8266 chip.
- **WAKE:** Wake pin is used to wake the chip from deep-sleep

F. ThingSpeak

What is ThingSpeak?

ThingSpeak is IoT Cloud platform where you can send sensor data to the cloud. You can also analyze and visualize your data with MATLAB or other software, including making your own applications.

The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your existing MathWorks Account.

ThingSpeak is free for small non-commercial projects.

ThingSpeak includes a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications. It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists) But it should work with all kind of Programming Languages, since it uses a REST API and HTTP.

Application of ThingSpeak

Environmental Monitoring: ThingSpeak can be used to monitor environmental parameters such as temperature, humidity, air quality, and pollution levels. This data can be collected from various sensors deployed in different locations and analyzed to gain insights into environmental conditions.

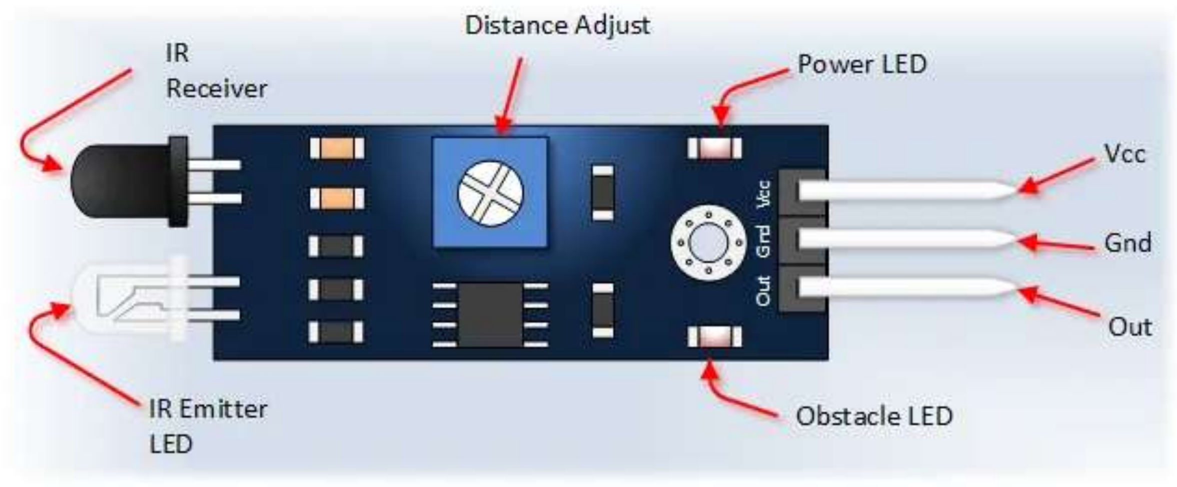
Smart Agriculture: In agriculture, ThingSpeak can be utilized to monitor soil moisture, temperature, and humidity levels in farms. Farmers can use this data to optimize irrigation schedules, monitor crop health, and improve overall farm productivity.

Home Automation: ThingSpeak can be integrated with home automation systems to monitor and control various devices such as lights, thermostats, and security cameras. Users can remotely monitor and control their home appliances using the ThingSpeak platform.

Industrial Monitoring and Control: In industrial settings, ThingSpeak can be used for monitoring and controlling various parameters such as machine temperature, pressure, and vibration levels. This helps in predictive maintenance, optimizing equipment performance, and ensuring safety

SENSORS USED

1. IR SENSOR



- IR sensor is an electronic device, that emits the light in order to sense some object of the surroundings.

- An IR sensor can measure the heat of an object as well as detects the motion.

- Usually, in the infrared spectrum, all the object radiates some form oof thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

- The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR Photodiode.

- Photodiode is sensitive to IR light of the same wavelength which is emitted by the IR LED.

-When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.

- IR Working Sensor Principle:

An IR Sensor consists of an IR LED and an IR Photodiode, together they are called as PhotoCoupler or OptoCoupler.

- IR transmitter is a light emitting diode (LED) which emits radiations called as IR LED's.

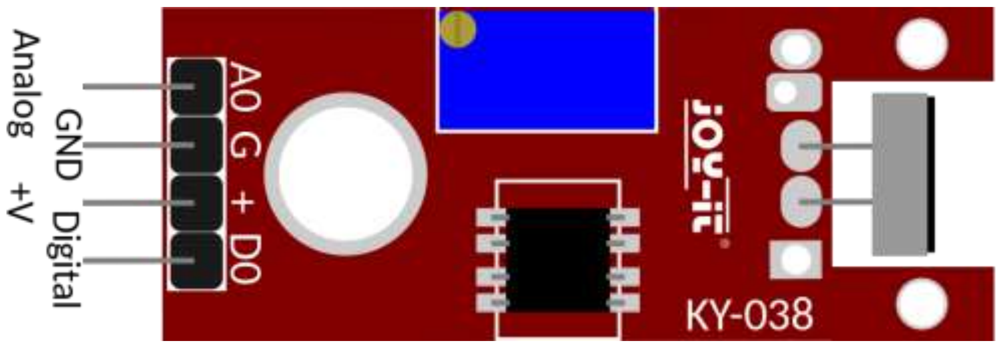
- Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to human eye.

SPECIFICATIONS:

- Operating Voltage: 3.0V – 5.0V
- Detection range: 2cm – 30cm (Adjustable using potentiometer)
- Current Consumption at 3.3V: ~23 mA, at 5.0V: ~43 mA
- Active output level: Outputs Low logic level when an obstacle is detected
- Onboard Obstacle Detection LED indicator.

2. SOUND SENSOR

- AO: analog output sensor
- GND: ground
- VCC: Power supply input range: 3 to 5 Volts.
- DO: Digital Output (comparator output)
- Two red LED indication: POWER and SENSOR. POWER: Power is off.



When the microphone senses sound reaches a certain value, this LED light. This sensor is ideally suited for threshold measurement. This means that the sensor emits a digital high signal as soon as a threshold value set by the user is exceeded. However, this also means that the analog measured values are not suitable for conversions, as the analog signal is also influenced by the rotary potentiometer.

Digital output: Via the potentiometer, a limit value for the received sound can be set, at which the digital output should switch.

Analog output: Direct microphone signal as voltage level

LED1: Shows that the sensor is powered

LED2: Indicates that a noise has been detected

FUNCTION OF THE SENSOR

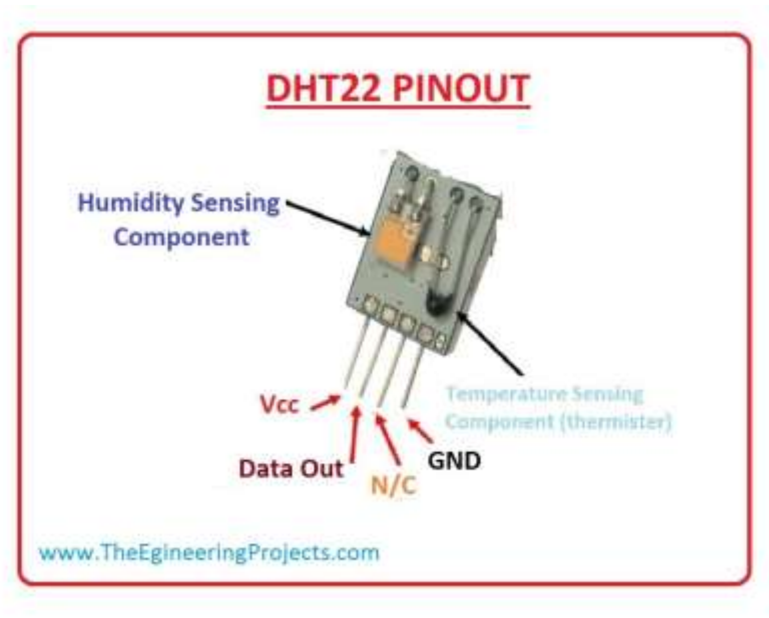
This sensor has three functional components on its circuit board: The front sensor unit, which physically measures the environment and outputs it as an analog signal to the second unit, the amplifier. This amplifies the signal depending on the resistance set on the rotary potentiometer and sends it to the analog output of the module.

Here it is to be noted: The signal is inverted. If a high value is measured, this results in a lower voltage value at the analog output.

The third unit represents a comparator, which switches the digital output and the LED when the signal falls below a certain value. This value (and thus the sensitivity of the module) can be adjusted via the rotary potentiometer:

As you can see from the image, a condenser microphone consists of two charged metal plates. The first plate is called the diaphragm and the second plate is the backplate of the microphone. These two plates together form a capacitor. When a sound wave hits the diaphragm of the microphone the diaphragm starts to vibrate, and the distance between the two plates changes. The movement of the diaphragm and the change in spacing produces the electrical signal that corresponds to the sound picked up by the microphone and this signal then gets processed by the onboard op-amp. This module also has two built-in onboard LEDs, one of which lights up when power is applied to the board and the other one lights up when the incoming audio signal exceeds the threshold value set by the potentiometer.

3. Temperature Sensor


DHT22 PINOUT
Humidity Sensing Component
Temperature Sensing Component (thermister)
Vcc
Data Out
N/C
GND
www.TheEgineeringProjects.com

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer. Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory. Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

Model-DHT22

Power supply: 3.3-6V DC

Output signal digital signal via single-bus

Sensing element Polymer capacitor

Pin sequence number: 1 2 3 4 (from left to right direction). Pin Function 1 VDD----power supply 2 DATA--signal 3 NULL 4 GND

Applications of DHT Sensor

1. DHT sensors can be used as a compensator with ultrasonic sensors to determine the distance more precisely.

2. DHT sensor is used in various applications such as measuring humidity and temperature values in heating, ventilation and air conditioning (HVAC) systems.

3. This sensor can be used in warehouse as level of humidity in air affects various physical, chemical and biological processes.

4. Weather stations also use these sensors to predict weather conditions.

4. Gas Sensor

The Simple MQ-5 Methane LPG Liquid Propane Gas Sensor Module is a great way to add basic gas sensing to your project at a reasonable price. TheMQ-5 Methane LPG Liquid Propane Gas Sensor Module particulate sensor uses a Buzzer to sense any detected material that is even present in the air!
This MQ-5 Methane LPG Liquid Propane Gas Sensor Module is widely used in gas leakage detecting pieces of equipment in family and industry, are suitable for detecting LPG, natural gas, town gas, avoid the noise of alcohol and cooking fumes and cigarette smoke
The MQ5 (MQ-5) is used in gas leakage detecting equipment in consumer and industry applications, this sensor is suitable for detecting LPG, natural gas, coal gas. Avoid the noise of alcohol, cooking fumes, and cigarette smoke. The sensitivity can be adjusted by the potentiometer.

The sensitive material of the MQ-5 gas sensor is tin Oxide ($SnO2$), which with lower conductivity in clean air. When the target combustible gas exists, the sensor’s conductivity is higher along with the gas concentration rising. Please use a simple electro circuit, Convert the change of conductivity to the corresponding output signal of gas concentration.

MQ-5 gas sensor has a high sensitivity to Methane, Propane, and Butane, and could be used to detect both Methane and Propane. The sensor could be used to detect different combustible gas especially Methane, it is with low cost and suitable for different application.

Connections

VCC ↔ *2.5V~5.0V*

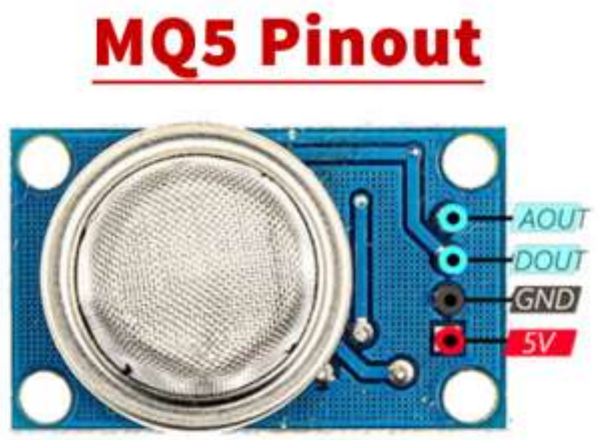
GND ↔ *ground*

Aout ↔ *MCU.IO (analog-output)*

Dout ↔ *MCU.IO (digital-output)*

Features:

1. High sensitivity to LPG, natural gas, town gas.
2. Small sensitivity to alcohol, smoke.
3. Stable and long life.
4. Simple drive circuit.
5. Fast response.



Hardware and Software Requirements

Hardware Requirements:

a) Microcontroller: STM32F407VG

b) Sensors:

Temperature Sensor: DHT22

Gas Sensor: MQ5

Infrared Sensor: HW201

Sound Sensor: KY038

c) Interfacing Components:

Analog-to-Digital Converter (ADC) for interfacing with sensors

Universal Asynchronous Receiver-Transmitter (UART) for communication with the host computer

d) Host Computer:

Capable of running FreeRTOS

Suitable connectivity ports for UART communication

e) Power Supply:

Power source for the microcontroller and sensors

Software Requirements:

A. Microcontroller Side:

FreeRTOS for task management on STM32F407VG

Device drivers for sensor interfacing and UART communication

ADC driver for analog-to-digital conversion

Code for sensor data acquisition and transmission over UART

B. Host Computer Side:

FreeRTOS ported for managing received data efficiently

Device drivers for UART communication

ThingSpeak library for graphical representation and visualization of acquired data

Code for receiving data from UART, processing, and displaying it using ThingSpeak library

C. Development Environment:

Integrated Development Environment (IDE) suitable for STM32F407VG development (e.g., STM32CubeIDE, Keil µVision, etc.)

UART terminal software for monitoring data transmission during development and testing

D. Operating System Compatibility:

Ensure compatibility of FreeRTOS with the host computer's operating system

Compatibility of development tools and drivers with the host computer's operating system

C. Programming Languages:
C or C++ for microcontroller firmware development

Possibly Python or other scripting languages for host computer software development

D. Documentation and Support:
Documentation for the STM32F407VG microcontroller, sensors, and related components

Support resources for FreeRTOS, ThingSpeak, and other libraries or tools used in the project

System Workflow:

Microcontroller Side:

FreeRTOS manages tasks including sensor data acquisition via ADC, real-time processing, and UART transmission.

ADC values from sensors (temperature, gas, infrared, sound) are acquired and converted to digital data.

The microcontroller sends ADC values via UART to the connected computer.

Host Computer Side:

FreeRTOS manages tasks for receiving and processing data from the microcontroller.

UART communication is established to receive ADC values from the microcontroller.

ThingSpeak library is utilized to dynamically plot ADC values on a graph for graphical representation.

Functionality and Features:

Real-time Monitoring: The system provides real-time monitoring of environmental parameters including temperature, gas concentration, infrared radiation, and sound levels.

Data Visualization: ThingSpeak library enables graphical representation of acquired data, facilitating intuitive interpretation of environmental variations.

Versatility: The integration of diverse sensors makes the system suitable for various applications such as environmental monitoring, industrial automation, and research.

Efficient Task Management: FreeRTOS ensures efficient task management on both the microcontroller and host computer, enabling real-time responses to environmental changes and organized data processing

Applications:

Environmental Monitoring

Industrial Automation

Research and Development

Smart Cities

Agricultural Monitoring

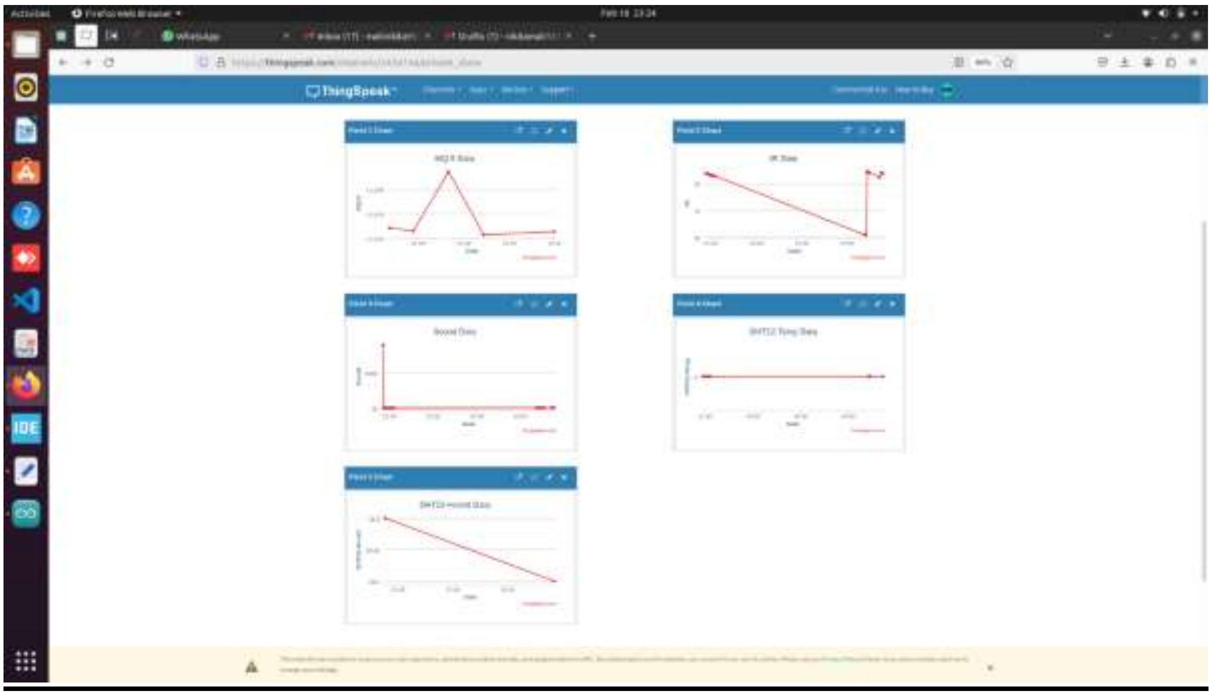
Health Monitoring (e.g., indoor air quality)

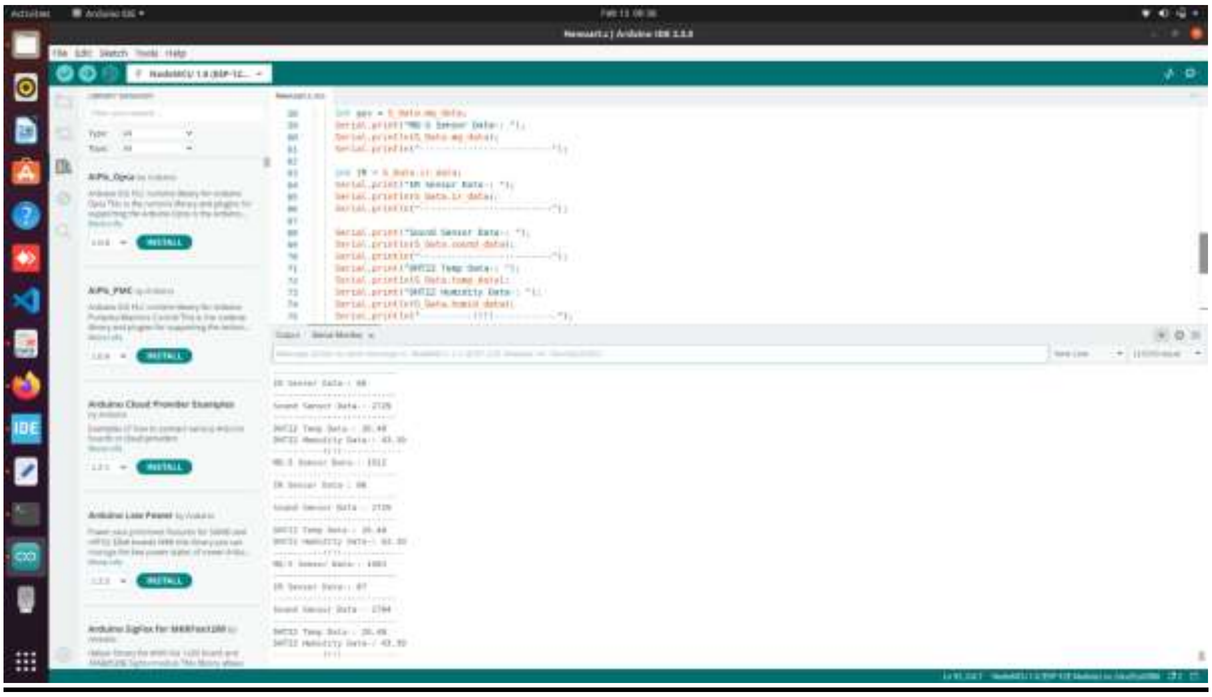
Benefits:

Comprehensive Understanding: Provides a comprehensive understanding of the surrounding environment through the integration of multiple sensors.

Real-time Responses: Enables real-time responses to environmental changes due to efficient task management and data processing.

OUTPUT





```
int aqv = S_Data.mq_data;
Serial.print("MQ 3 Sensor Data:- ");
Serial.println(S_Data.mq_data);
Serial.println("------------------------------");

int IR = S_Data.ir_data;
Serial.print("IR Sensor Data:- ");
Serial.println(S_Data.ir_data);
Serial.println("------------------------------");

Serial.print("Sound Sensor Data:- ");
Serial.println(S_Data.sound_data);
Serial.println("------------------------------");
Serial.print("DHT22 Temp Data:- ");
Serial.println(S_Data.temp_data);
Serial.print("DHT22 Humidity Data:- ");
Serial.println(S_Data.humid_data);
Serial.println("------------||||-------------");
```

```
------------------------------
IR Sensor Data:- 68
------------------------------
Sound Sensor Data:- 2126
------------------------------
DHT22 Temp Data:- 26.48
DHT22 Humidity Data:- 43.30
------------||||-------------
MQ 3 Sensor Data:- 1512
------------------------------
IR Sensor Data:- 68
------------------------------
Sound Sensor Data:- 2126
------------------------------
DHT22 Temp Data:- 26.48
DHT22 Humidity Data:- 43.30
------------||||-------------
MQ 3 Sensor Data:- 1483
------------------------------
IR Sensor Data:- 67
------------------------------
Sound Sensor Data:- 2744
------------------------------
DHT22 Temp Data:- 26.48
DHT22 Humidity Data:- 43.30
------------||||-------------
```

CONCLUSION

In conclusion, this project presents an integrated system that seamlessly amalgamates diverse sensor networks, leveraging the capabilities of FreeRTOS for real-time task scheduling and coordination. The purpose of this endeavor is to create a robust and efficient data acquisition system capable of acquiring, processing, and transmitting data from multiple sensors, namely the MQ-5 gas sensor, IR sensor, and sound sensor. The overarching objective is to facilitate real-time monitoring and analysis of environmental parameters, enabling applications in various domains such as industrial automation, IoT, and medical instrumentation.

The system architecture is meticulously designed to ensure modularity, scalability, and responsiveness. By employing FreeRTOS, the project attains a level of flexibility and task management crucial for handling concurrent operations of diverse sensors. The utilization of an STM32 microcontroller, along with FreeRTOS, provides a reliable and efficient platform for data acquisition.

Motivated by the imperative to enhance data acquisition capabilities, the inclusion of sensors like the MQ-5 gas sensor addresses concerns related to environmental safety and pollution monitoring. The IR sensor contributes by enabling proximity detection and object recognition, while the sound sensor captures acoustic data for further analysis.

The FreeRTOS operating system plays a pivotal role in task management, ensuring that sensor data is acquired and transmitted seamlessly. The integration of UART communication facilitates the transmission of sensor data to external devices, enabling connectivity with platforms like NodeMCU. This, in turn, enables the relay of data to cloud services such as ThingSpeak, enhancing the system's capabilities for remote monitoring and analysis.

In summary, this project not only exemplifies the synergy of sensor technologies within a unified framework but also underscores the importance of real-time operating systems like FreeRTOS in orchestrating the complex interplay of tasks. The amalgamation of sensor networks in this project lays the foundation for diverse applications, ranging from industrial automation to environmental monitoring, thereby contributing to the paradigm shift towards smart and interconnected systems.

REFERENCES

[1] Cheng, L., Yu, H., Research on intelligent maintenance unit of rotary machine, Computer Integrated Manufacturing Systems, vol. 10, Issue: 10, page 1196-1198, 2004.

[2] Yu, C., Zhong, Ou., Zhen, D., Wei, F., Design and Implementation of Monitoring and Management Platform in Embedded Fault Diagnosis System, Computer Engineering, vol. 34, Issue: 8, page 264-266, 2008.

[3] Bi, D., Gui, T., Jun, S., Dynam . Behavior of a High-speed Hybrid Gas Bearing-rotor System for a Rotating ramjet, Journal of Vibration and Shock, vol. 28, Issue: 9, page 79-80, 2009.

[4] Hai, L., Jun, S., Research of Driver Based on Fault Diagnosis System Data Acquisition Module, Machine Tool& Hydraulics, vol. 38, Issue: 13, page 166-168, 2011.

[5] Hao, W., Qin, W., Xiao, S., Optimized. Design of Embedded Parallel Data Acquisition System, Computer Engineering and Design, vol. 32, Issue: 5, page 1622-1625, 2011.