

SPIDER ROBOT FOR THE HAZARDIOUS GAS DETECTION



A Thesis submitted to
Rajiv Gandhi proudyogiki Vishwavidyala , Bhopal
Towards the Partial fulfilment of the degree of
Bachelor of Technology
in
Electronics and Telecommunication Engineering

Guided By :

**Prof. Anjulata Yadav
Prof. Neeta Sharma**

Submitted By:

**Ankita Dehariya
(0801EC201014)
Babita Bisi
(0801EC201027)
Nidhi Jagtap
(0801EC201058)
Shivanshu Bhandari
(0801EC201085)**

**Department of Electronics and Telecommunication Engineering
Shri G. S. Institute of technology and Science Indore – 452003 (M.P.)
May 2024**

SPIDER ROBOT FOR THE HAZARDIOUS GAS DETECTION



A Thesis submitted to
Rajiv Gandhi proudyogiki Vishwavidyala , Bhopal
Towards the Partial fulfilment of the degree of
Bachelor of Technology
in
Electronics and Telecommunication Engineering

Guided By :

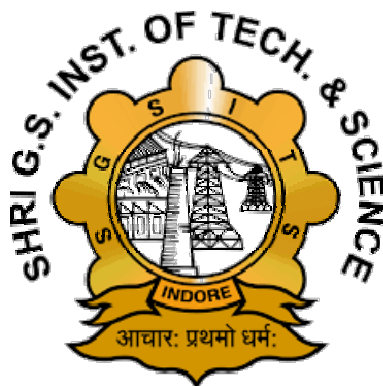
**Prof. Anjulata Yadav
Prof. Neeta Sharma**

Submitted By:

**Ankita Dehariya
(0801EC201014)
Babita Bisi
(0801EC201027)
Nidhi Jagtap
(0801EC201058)
Shivanshu Bhandari
(0801EC201085)**

**Department of Electronics and Telecommunication Engineering
Shri G. S. Institute of technology and Science Indore – 452003 (M.P.)
May 2024**

**SHRI G. S. INSTITUTE OF TECHNOLOGY AND SCIENCE
INDORE**



RECOMMENDATION

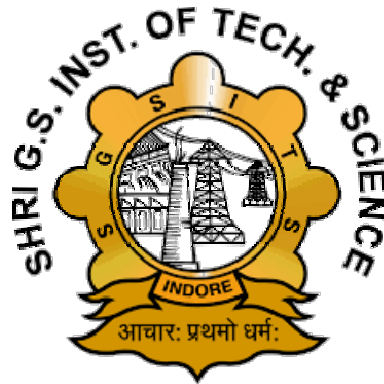
This thesis entitled, “ **Spider Robot for the Hazardious Gas Detection**”, submitted to the Rajiv Gandhi proudyogiki Vishwavidyalaya, Bhopal, by **Shivanshu Bhandari, Ankita dehariya, Nidhi Jagtap , Babita Bisi** during the academic year 2023-2024, as a partial fulfilment for the award of the degree of **bachelor of technology** in **Electronics and Telecommunication Engineering** is a record of student’s own work carried out by them under our direct supervision, in the **Department of Electronics and Telecommunication Engineering, S.G.S.I.T.S. Indore**. The work contained in the thesis is a satisfactory account of their project work and is recommended for the award of the degree.

Prof. Anjulata Yadav
(Guide)

Prof. Neeta Sharma
(Guide)

Prof. Anjana Jain
(Head of Department)
Electronics and Telecommunication Engineering

**SHRI G. S. INSTITUTE OF TECHNOLOGY AND SCIENCE
INDORE**



CERTIFICATE

This is to certify that the thesis entitled , “ **Spider Robot for the Hazardious Gas Detection**”, submitted to the Rajiv Gandhi proudyogiki Vishwavidyalaya, Bhopal, by **Shivanshu Bhandari, Ankita dehariya, Nidhi Jagtap , Babita Bisi** during the academic year 2023-2024. Engineering is a record of student’s own work carried out by them under our direct supervision, in the **Department of Electronics and Telecommunication Engineering.**

Internal Examiner

Date:

 **External examiner**

Date:

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **Prof. Anjulata Yadav** Department of Electronics and Telecommunication, S.G.S.I.T.S, Indore whose constant encouragement enabled us to work enthusiastically. His patience and technical expertise in discussion benefited us, in completing our thesis work. The trust he placed in our abilities was always a great source of motivation. We would also like to express our gratitude towards our co-guide **Prof. Neeta Sharma**, for his valuable time and suggestions given on our project work.

We are also grateful to our Head of Department **Prof. Anjana Jain** for his caring attitude and encouragement. We thank all other faculty members, staff, our seniors and our batch-mates whose consistent effort in maintaining academic excellence has resulted in NBA accreditation of our B.E. (Electronics and Telecommunication Engineering) course. We also appreciate the support of **Prof. R. K. Saxena**, Director, SGSITS Indore for providing the academic facilities in this Institute.

Finally We owe our special thanks to the Almighty God and our beloved Parents for supporting and encouraging us with their support, patience, blessings and understanding to pursue this work, without, which we would not have finished our degree.

Shivanshu Bhandari

Ankita Dehariya

Nidhi Jagtap



Babita Bisi

ABSTRACT

it should be more technical and precise

The purpose of our project is to create an robot of small size ,which can be used for the places where human being can't go .The Hazardous gas detector spider robot is amobile controlled robot that has the movement like a spider and this movement is useful to walk through difficult terrain.Basically we are using these spider robot for rat hole mining purpose where the spider can go,detect gases and environmental condition .We have focused on creating a device that is useable, rugged and safely contained.This spider robot can be used in the industries and hence prevent the workers to avoid the hazardous environmental conditions. and accidents like the Bhopal gas tragedy.

Table of Contents

List of Figures	9
List of Tabela	10
 List of Abbreviations	11
List of Symbols	13
CHAPTER 1	14
Introduction	14
Literature Review	16
chapter 2? Introduction	16
CHAPTER 3	19
Methodology.....	19
3.1 Design Overview	19
3.2 Components Used.....	19
3.3 Arduino Nano	20
3.4 ESP32	23
3.5 MQ2 Gas Sensor.....	26
3.6 DHT11 (Temperature and Humidity Sensor)	32
3.8 Bluetooth Module	40
3.9 Servo Motor.....	42
3.10 DC-DC Buck Converter.....	44
3.11 Lithium-ion battery	46
3.12 Power Management	48
3.13 Data Transmission to ThingSpeak.....	49
CHAPTER 4.....	50
Design and Implementation	50
4.1 Spider Robot Structure.....	50
4.2 Arduino Nano Programming.....	53
4.3 ESP32 Programming.....	57
4.4 Sensor Calibration and Testing:	61
4.5  Integration Testing.....	63

CHAPTER 5 66

Results and Analysis 66

5.1 Servo Motor Control Performance 66

5.2 Sensor Data Accuracy 68

5.3 Power Consumption Analysis 69

5.4 Data Visualization on ThingSpeak 71

CHAPTER 6 73

Conclusion 73

CHAPTER 7 73

References 73

List of Figures

Figures

Figure 3.3: Arduino Nano Pinout Overview

Figure 3.4: ESP32 Microcontroller Overview

Figure 3.5.1: Internal Structure of MQ2 Gas Sensor

Figure 3.5.2: MQ2 Gas Sensor with Anti-Explosion Network

Figure 3.5.3: MQ2 Gas Sensor with Mesh Removed

Figure 3.5.4: MQ2 Gas Sensor Structure with Leads

Figure 3.5.5: Tubular Sensing Element of MQ2 Gas Sensor

Figure 3.5.6: Working Principle of MQ2 Gas Sensor

Figure 3.5.7: MQ2 Gas Sensor Module Hardware Overview

Figure 3.5.8: Relationship between Gas Concentration and Output Voltage

Figure 3.5.9: Digital Output Pin of MQ2 Gas Sensor Module

Figure 3.5.10: Potentiometer for Adjusting Digital Output Sensitivity

Figure 3.6: working principle of humidity sensors.

Figure 3.7.1: Illustration explaining atmospheric pressure and its measurement.

Figure 3.7.2: Diagram illustrating different pressure units and their conversions.

Figure 3.8: Diagram depicting the HC-05 Bluetooth module and its pinout.

Figure 3.9: Illustration showing the structure and components of a servo motor.

Figure 3.10: Diagram explaining the operation and components of a DC-DC buck converter.

Figure 3.11: Schematic representation of the structure and components of a lithium-ion battery.

Figure 5.4: Enhanced User Understanding and Engagement

List of Tables

Tables

Table 1 : Features of Arduino Nano

Table 2: Arduino Nano Pinout

Table 3: Arduino Nano communications protocols

Table 4: Memory in Arduino Nano

Table 5: Technical Specifications of MQ 02 sensor

Table 6: Basic Characteristics of Lithium Ion battery

LIST OF ABBREVIATIONS

ATmega328p - Microcontroller used in Arduino Nano

DIP30 - Dual Inline Package with 30 pins

IDE - Integrated Development Environment

MHz - Megahertz

USB - Universal Serial Bus

DC - Direct Current

ICSP - In-Circuit Serial Programming

RX - Receiver

TX - Transmitter

SPI - Serial Peripheral Interface

MOSI - Master Output Slave Input

MISO - Master Input Slave Output

SCK - Serial Clock

SDA - Serial Data Line

SCL - Serial Clock Line

EEPROM - Electrically Erasable Programmable Read-Only Memory

BLE - Bluetooth Low Energy

GPIO - General Purpose Input/Output

ADC - Analog-to-Digital Converter

DAC - Digital-to-Analog Converter

PWM - Pulse Width Modulation

SD - Secure Digital

MMC - MultiMediaCard

RSA - Rivest–Shamir–Adleman (cryptographic algorithm)

SHA - Secure Hash Algorithm

ECC - Elliptic Curve Cryptography

IoT - Internet of Things

PID - Proportional-Integral-Derivative

hPa - Hectopascal

LPG - Liquefied Petroleum Gas

mA - Milliampere

Ah - Ampere-hour

API - Application Programming Interface

LIST OF SYMBOLS

C	Cipher Text
D	Dealer
D_K	Decryption Algorithm
E_K	Encryption Algorithm
g	Generator Polynomial Integer
$h(P)$	Hash Function
K_{AB}	Symmetric Key between A and B
$\overset{Pr}{K}_A$	Private Key Of Principle A
$\overset{Pu}{K}_A$	Public Key Of Principle A
$MAC_k(M)$	Message Authentication Code
N	Nonce a number to be used once
N_A	Nonce of a participant or principle
P	Plain Text
Pr	Probability
S	Secret
$SIG(M)$	Signature of message M
X_i	Input Plain text Stream
Y_i	Output Cipher text Stream
Z_i	Encryption key Stream
k	Boltzman Constant

CHAPTER 1

Introduction

Rewrite introduction, it seems different paragraphs are just arranged, it should be well organized and conveyed its purpose

The introduction looks like discussion of its application. it is one part, other than this technicality part missing.

font changed from this page why ?

In an age where industrial accidents and environmental hazards pose significant risks to human life and the environment, innovative solutions are needed to mitigate these dangers effectively. The Hazardous Gas Detector Spider Robot represents a groundbreaking advancement in robotics technology, offering a multifaceted approach to ensuring safety and enabling exploration in challenging environments. Inspired by the dexterity and adaptability of spiders, this mobile-controlled robot integrates advanced sensors and sophisticated locomotion mechanisms to navigate through diverse terrains while detecting hazardous gases and assessing environmental conditions in real-time.

introduction switch to application directly why?

Application in Industrial Settings

The Hazardous Gas Detector Spider Robot's primary application lies within industrial environments, where workers are frequently exposed to hazardous gases and unpredictable conditions. By deploying these robots in manufacturing plants, chemical facilities, and refineries, companies can significantly enhance safety protocols and prevent accidents akin to the tragic Bhopal gas disaster. Equipped with an array of sensors including gas detectors, temperature sensors, humidity sensors, and environmental monitors, the robot continuously monitors its surroundings, providing real-time data to alert workers and operators of potential risks.

Moreover, the robot's unique spider-like locomotion enables it to traverse challenging terrain with ease, accessing confined spaces, elevated platforms, and areas inaccessible to human workers. This capability not only enhances safety by reducing human exposure to hazards but also facilitates more comprehensive monitoring of industrial facilities. The data collected by the robot can be integrated into existing safety systems, allowing for proactive risk management and targeted interventions to mitigate potential accidents before they occur.

Exploration and Research Endeavors

Beyond industrial applications, the Hazardous Gas Detector Spider Robot holds immense promise for research and exploration endeavors in diverse fields. In the realm of space exploration, where the harsh conditions of extraterrestrial environments present formidable challenges, these robots offer a versatile platform for conducting scientific research and exploration missions. Equipped with advanced sensors, imaging systems, and autonomous navigation capabilities, they can navigate planetary surfaces, analyze soil samples, and investigate atmospheric conditions with unprecedented precision.

Similarly, in defense applications, the spider robot's agility and stealth capabilities make it an invaluable asset for reconnaissance missions in hostile environments. Whether conducting surveillance in urban landscapes, monitoring border regions, or assessing disaster zones, these robots provide critical intelligence and situational awareness to military personnel and security forces.

Moreover, in scientific research, the spider robot's adaptability and mobility enable researchers to study environments that are hazardous or inaccessible to humans. From exploring deep-sea ecosystems and volcanic regions to monitoring wildlife in remote habitats, these robots offer a versatile platform for conducting field research and collecting valuable data for scientific study.

Future Prospects and Conclusion

As technology continues to advance, the potential for further innovation and refinement of the Hazardous Gas Detector Spider Robot is virtually limitless. Future iterations of the robot may incorporate advanced AI algorithms for autonomous decision-making, enhanced sensor suites for detecting a wider range of hazardous substances, and improved durability and ruggedness for operating in extreme environments.

In conclusion, the Hazardous Gas Detector Spider Robot represents a paradigm shift in robotics technology, offering a versatile solution for ensuring safety in hazardous environments and facilitating exploration in challenging terrains. From industrial applications to scientific research and space exploration, its impact spans across diverse sectors, promising to revolutionize the way we approach safety and exploration in the 21st century. As we continue to harness the power of robotics and artificial intelligence, the possibilities for leveraging these technologies to overcome the challenges of hazardous environments and push the boundaries of exploration are limitless.

CHAPTER 2

Literature Review

check spacing in this section. different spacing used make it uniform

Introduction

Hazardous gas detection in industrial settings, disaster areas, and confined spaces is a critical task that demands efficiency and safety. Traditional methods often involve human intervention, which can pose significant risks. Spider robots, with their ability to navigate complex environments and carry specialized sensors, offer a promising solution for hazardous gas detection. This literature review examines existing research on spider robots for hazardous gas detection, focusing on methodologies, common themes, disagreements, impact of specific studies, emerging trends, and commonly cited sources in this field.

Methodologies Used in Studies

Research on spider robots for hazardous gas detection has employed various methodologies:

- **Sensor Integration:** Integration of gas sensors, environmental sensors, and cameras onto spider robot platforms to enable comprehensive data collection and analysis.
- **Navigation and Localization:** Implementation of simultaneous localization and mapping (SLAM) algorithms, path planning strategies, and obstacle avoidance techniques to facilitate accurate navigation and gas source localization.
- **Communication Systems:** Development of robust communication protocols to enable real-time data transmission between spider robots and control centers, facilitating remote monitoring and control.
- **Data Processing and Analysis:** Utilization of advanced data processing algorithms for real-time gas detection, classification, and concentration estimation, enabling timely decision-making and response.

Common Themes in Literature

Several common themes emerge across the literature:

- **Miniaturization:** Emphasis on miniaturizing spider robots to enhance agility and maneuverability, enabling them to access confined spaces and navigate complex environments effectively.
- **Autonomy:** Focus on developing autonomous capabilities in spider robots to enable independent operation and adaptation to dynamic environments without human intervention.
- **Multi-Sensor Fusion:** Integration of multiple sensors and fusion of sensor data to improve the accuracy and reliability of gas detection in diverse environmental conditions.

Disagreements in Literature

While the literature generally supports the potential of spider robots for hazardous gas detection, disagreements may arise regarding optimal design choices, sensor configurations, and algorithmic approaches. Variations in performance and effectiveness can result from different priorities and perspectives among researchers.

Impact of Specific Studies

The impact of specific studies on the broader field depends on their novelty, effectiveness, and relevance. Pioneering research introducing innovative sensor technologies, navigation algorithms, or autonomous capabilities can significantly influence future research directions and technological advancements.

Emerging trends in the literature include:

- **Soft Robotics:** Exploration of soft robotics principles to enhance spider robot adaptability and safety in dynamic environments.
- **Machine Learning:** Utilization of machine learning algorithms for advanced data analysis and decision-making, enabling improved gas classification and anomaly detection.
- **Swarm Robotics:** Investigation of swarm robotics approaches for coordinating multiple spider robots to cover larger areas and enhance gas detection efficiency.

Commonly Cited Sources

Commonly cited sources include seminal papers on robotics, gas sensing technologies, and relevant conference proceedings and journals. Additionally, sources detailing advances in sensor integration, navigation algorithms, and robotics applications in hazardous environments are frequently referenced.

Conclusion

In conclusion, spider robots equipped with advanced sensing and mobility capabilities hold immense potential for enhancing hazardous gas detection in challenging environments. By integrating multi-sensor data fusion, autonomous navigation, and advanced data analysis techniques, these robots offer a promising solution to mitigate risks and improve response capabilities in hazardous situations. Continued innovation and collaboration are crucial for addressing remaining challenges and realizing the full potential of spider robots for hazardous gas detection.

CHAPTER 3

flow diagram included that tells the methodology

Methodology

Design or block diagram should be included

3.1 Design Overview

The Hazardous gas detector spider robot is a mobile controlled robot that has the movement like a spider and this movement is useful to walk through difficult terrain. It has various sensors on it and these sensors are used to sense the hazardous gases and the conditions of the environment that helps a person to know the environmental conditions beforehand, and hence prevent from accidents like the Bhopal gas tragedy. This spider robot can be used in the industries and hence prevent the workers to avoid the hazardous environmental conditions. It can also be used for research purposes in the field of space sector, defence sector and also in the sectors where it is difficult for humans to go. Here, in our project, we are just demonstrating the prototype and this can be Rebuilt with advance technology for various purpose. In these we are using many sensor for temperature and humidity DHT11 sesnsor And we are using MQ2 gas sensor BMP180 Biometric pressure sensor.

3.2 Components Used

- 1) Arduino nano
- 2) Node MCU (ESP8266)
- 3) Servo motors
- 4) BMP 180 Pressure sensor
- 5) MQ2 smoke sensor
- 6) 18650 Li-Po Battery (3.7 volts)
- 7) HC 05 Bluetooth module
- 8) DHT 11 sensor
- 9) DC DC Buck Converter
- 10) Antenna

3.3 Arduino Nano

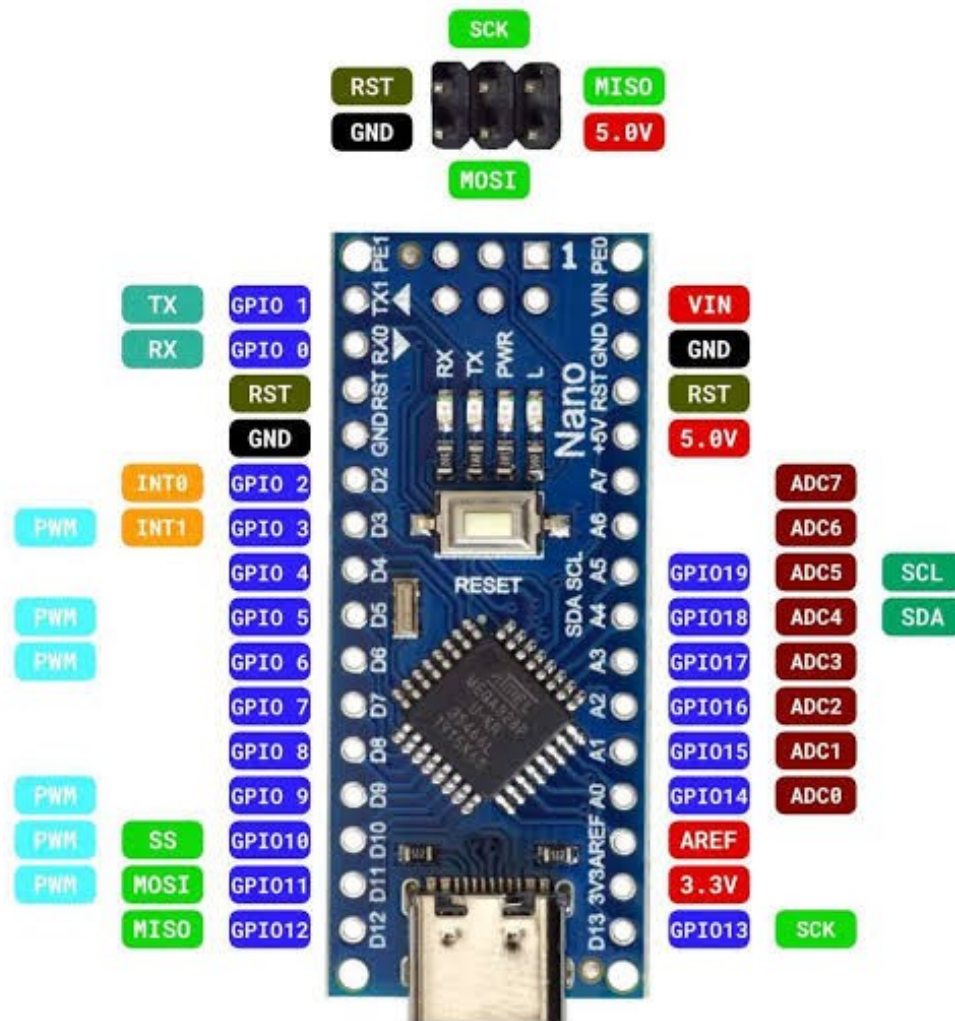


Figure 3.3

- **Arduino Nano** is a small, complete, flexible and breadboard-friendly Microcontroller board, based on **ATmega328p**, developed by Arduino.cc in Italy in 2008 and contains 30 male I/O headers, configured in a **DIP30 style**.
- **Arduino Nano Pinout** contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins.
- It is programmed using **Arduino IDE**, which can be downloaded from the Arduino Official site.

- Arduino Nano is simply a smaller version of Arduino UNO, thus both have almost the same functionalities.
- It comes with an **operating voltage of 5V**, however, the input voltage can vary from **7 to 12V**.
- Arduino Nano's **maximum current rating is 40mA**, so the load attached to its pins shouldn't draw a current more than that.
- Each of these Digital & Analog Pins is assigned with multiple functions but their main function is to be configured as **Input/Output**.
- Arduino Pins are acted as **Input Pins** when they are interfaced with sensors, but if you are driving some load then we need to use them as an Output Pin.
- Functions like **pinMode()** and **digitalWrite()** are used to control the operations of digital pins while **analogRead()** is used to control analog pins.
- The analog pins come with a total **resolution of 10-bits** which measures the value from 0 to 5V.
- Arduino Nano comes with a **crystal oscillator of frequency 16 MHz**. It is used to produce a clock of precise frequency using constant voltage.
- There is one limitation of using Arduino Nano i.e. it doesn't come with a **DC power jack**, which means you can not supply an external power source through a battery.
- This board doesn't use standard USB for connection with a computer, instead, it comes with **Type-B Micro USB**.
- The tiny size and breadboard-friendly nature make this device an ideal choice for most applications where the size of the electronic components is of great concern.
- **Flash memory is 16KB or 32KB** that all depends on the Atmega board i.e. Atmega168 comes with 16KB of flash memory while Atmega328 comes with a flash memory of 32KB. Flash memory is used for storing code. The 2KB of memory out of total flash memory is used for a bootloader.
- The **SRAM memory of 2KB** is present in Arduino Nano.

- Arduino Nano has an **EEPROM memory of 1KB**.
- Here's the table showing important features of Arduino Nano:

No.	Nano Features	Value
1	Microcontroller	Atmega328p
2	Crystal Oscillator	16MHz
3	Operating Voltage	5V
4	Input Voltage	6V-12V
5	Maximum Current Rating	40mA
6	USB	Type-B Micro USB
7	ICSP Header	Yes
8	DC Power Jack	No

- Here's the quick overview of Arduino Nano Pinout:

No.	Pin Number	Pin Description
1	D0 - D13	Digital Input / Output Pins.
2	A0 - A7	Analog Input / Output Pins.
3	Pin # 3, 5, 6, 9, 10, 11	Pulse Width Modulation (PWM) Pins.
4	Pin # 0 (RX) , Pin # 1 (TX)	Serial Communication Pins.
5	Pin # 10, 11, 12, 13	SPI Communication Pins.
6	Pin # A4, A5	I2C Communication Pins.
7	Pin # 13	Built-In LED for Testing.
8	D2 & D3	External Interrupt Pins.

- Arduino Nano offers three types of communications protocols, shown in the below table:

No.	Communication Protocols	Description
6	Serial Port	1 (Pin#0 is RX, Pin#1 is TX).
7	I2C Port	1 (Pin#A4 is SDA, Pin#A5 is SCL).
8	SPI Port	1 (Pin#10 is SS, Pin#11 is MOSI, Pin#12 is MISO, Pin#13 is SCK).

- Here's the memory details present in Arduino Nano:

No.	Memory Type	Value
7	Flash Memory	32KB
8	SRAM Memory	2KB
7	EEPROM	1KB

3.4 ESP32

The ESP32 is a powerful microcontroller developed by Espressif Systems. Here's a detailed description of its features:

- **Dual-Core Processor:** The ESP32 features two Tensilica LX6 microprocessor cores, allowing for parallel processing and improved performance.
- Wireless Connectivity:
- **Wi-Fi:** It supports 802.11 b/g/n Wi-Fi standards, making it capable of connecting to Wi-Fi networks. It also supports Wi-Fi Direct (P2P), allowing devices to connect to each other without the need for a central access point.

- Bluetooth: The ESP32 includes Bluetooth Low Energy (BLE) support, enabling communication with other BLE devices such as smartphones, sensors, and wearables.

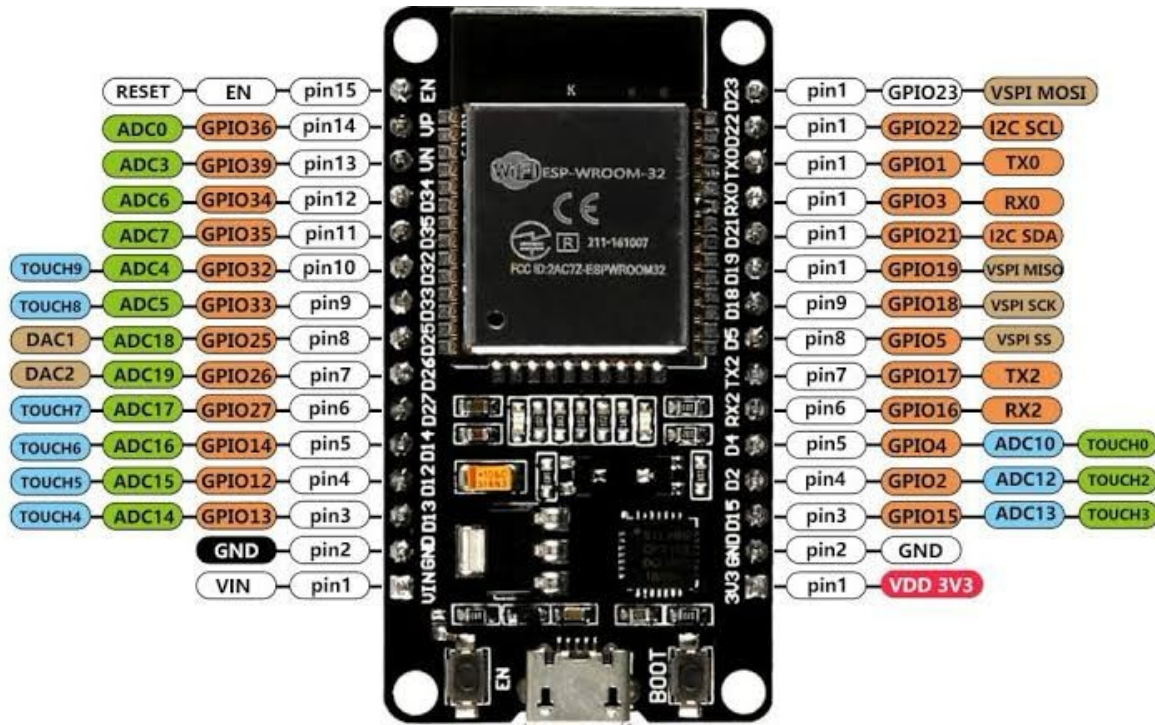


Figure 3.4

- Bluetooth Classic: In addition to BLE, it also supports Bluetooth Classic for legacy device compatibility.
- Peripheral Interfaces:
- GPIO: The ESP32 provides a large number of General Purpose Input/Output pins, allowing it to interface with various sensors, actuators, and other external devices.
- SPI, I2C, UART: It supports multiple communication protocols such as SPI, I2C, and UART, enabling communication with a wide range of external devices.
- ADC, DAC: The ESP32 features Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC) channels for analog sensor interfacing and audio applications.
- PWM: It offers Pulse Width Modulation (PWM) channels for controlling LEDs, motors, and other PWM-compatible devices.

- **SD/MMC Interface:** Some variants of the ESP32 include built-in support for Secure Digital (SD) and MultiMediaCard (MMC) memory cards.
- **Security Features:**
 - **Secure Boot:** The ESP32 supports secure boot, ensuring that only authorized firmware is executed during the boot process, enhancing system security.
 - **Cryptographic Acceleration:** It includes hardware accelerators for cryptographic operations such as AES, RSA, SHA, and ECC, enabling secure communication and data encryption.
 - **Secure Storage:** The ESP32 provides secure storage options for sensitive data, protecting it from unauthorized access.
- **Low Power Consumption:**
 - The ESP32 offers various low-power modes, allowing it to minimize power consumption in battery-operated and energy-efficient applications.
 - It includes features such as deep sleep mode, which drastically reduces power consumption during idle periods.
- **Development Environment:**
 - The ESP32 can be programmed using the Arduino IDE, ESP-IDF (IoT Development Framework), or other development platforms such as PlatformIO.
 - It has a large and active community, providing extensive documentation, tutorials, and libraries to support development efforts.
- Overall, the ESP32 is a versatile and feature-rich microcontroller suitable for a wide range of IoT (Internet of Things), embedded, and wireless communication applications. Its combination of dual-core processing, wireless connectivity options, peripheral interfaces, security features, and low power consumption make it a popular choice among developers for building connected devices and IoT solutions.

3.5 MQ2 Gas Sensor

The MQ2 sensor is one of the most widely used in the MQ sensor series. It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as Chemiresistors because sensing is based on the change in resistance of the sensing material when exposed to gasses.



Figure 3.5.1

The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW. It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations ranging from 200 to 10000 ppm.

What does the concentration of 1 ppm mean?

Parts-per-million, or ppm for short, is the most commonly used unit for measuring gas concentration. ppm is simply the ratio of one gas to another. For example, 500ppm of carbon monoxide means that if you could count a million gas molecules, 500 would be carbon monoxide and the remaining 999500 would be other gasses.

Note that the MQ2 gas sensor detects multiple gases, but cannot identify them! That is normal; most gas sensors operate in this manner. Therefore, it is best suited for measuring changes in a known gas density rather than detecting which one is changing.

Internal structure of MQ2 Gas Sensor

The MQ2 is a heater-driven sensor. It is therefore covered with two layers of fine stainless steel mesh known as an “anti-explosion network”. It ensures that the heater element inside the sensor does not cause an explosion because we are sensing flammable gasses.



Figure 3.5.2

It also protects the sensor and filters out suspended particles, allowing only gaseous elements to pass through the chamber. A copper-plated clamping ring secures the mesh to the rest of the body.

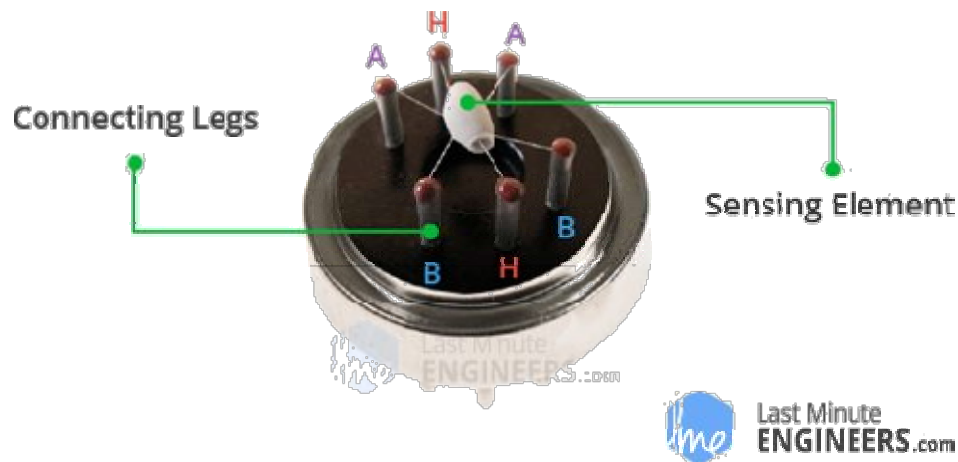


Figure 3.5.3

When the outer mesh is removed, the sensor looks like this. The sensing element and six connecting legs that extend beyond the Bakelite base form the star-shaped structure. Two (H) of the six leads are in charge of heating the sensing element and are linked together by a Nickel-Chromium coil (a well-known conductive alloy).

The remaining four signal-carrying leads (A and B) are connected with platinum wires. These wires are connected to the body of the sensing element and convey slight variations in the current flowing through the sensing element.

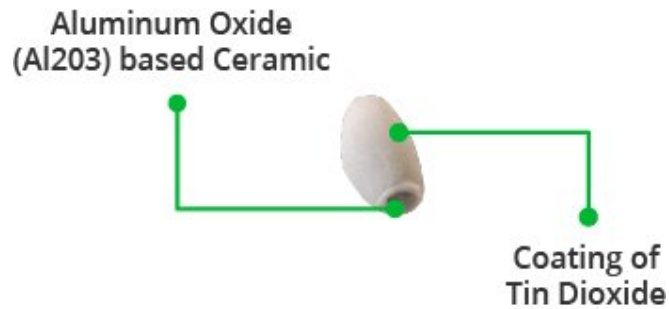


Figure 3.5.4

The tubular sensing element is made of Aluminum Oxide (Al₂O₃) based ceramic with a Tin Dioxide coating (SnO₂). Tin Dioxide is the most important material because it is sensitive to combustible gasses. The ceramic substrate, on the other hand, improves heating efficiency and ensures that the sensor area is continuously heated to the working temperature.

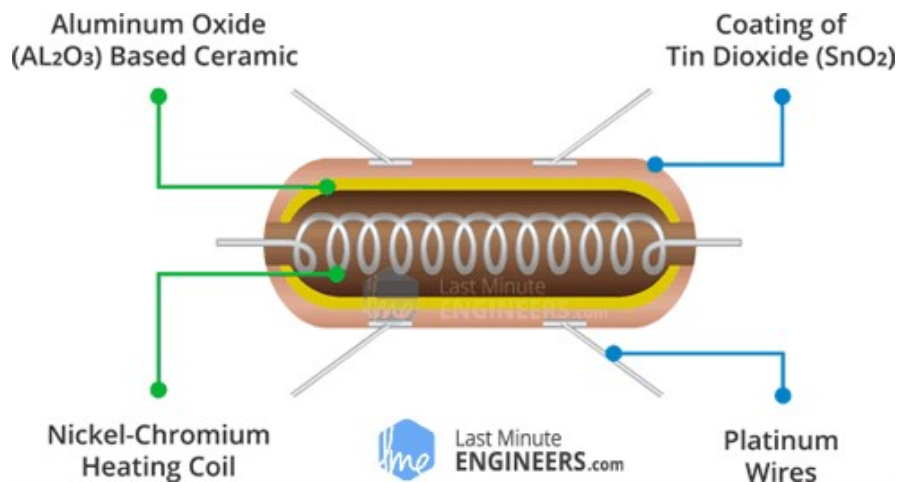


Figure 3.5.5

To summarize, the Heating System is composed of a Nickel-Chromium coil and an Aluminum Oxide-based ceramic, while the Sensing System is composed of Platinum wires and a Tin Dioxide coating.

Working working of.... specify

When a SnO₂ semiconductor layer is heated to a high temperature, oxygen is adsorbed on the surface. When the air is clean, electrons from the conduction band of tin dioxide are attracted to oxygen molecules. This creates an electron

depletion layer just beneath the surface of the SnO₂ particles, forming a potential barrier. As a result, the SnO₂ film becomes highly resistive and prevents electric current flow.

In the presence of reducing gasses, however, the surface density of adsorbed oxygen decreases as it reacts with the reducing gasses, lowering the potential barrier. As a result, electrons are released into the tin dioxide, allowing current to freely flow through the sensor.

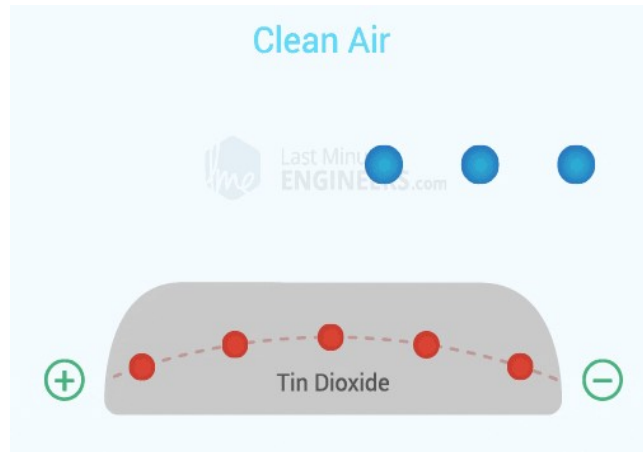


Figure 3.5.6

MQ2 Gas Sensor Module Hardware Overview

The MQ2 gas sensor is simple to use and has two different outputs. It not only provides a binary indication of the presence of combustible gasses, but also an analog representation of their concentration in air.



Figure 3.5.7

The sensor's analog output voltage (at the A0 pin) varies in proportion to the concentration of smoke/gas. The higher the concentration, the higher the output voltage; the lower the concentration, the lower the output voltage. The

animation below shows the relationship between gas concentration and output voltage.

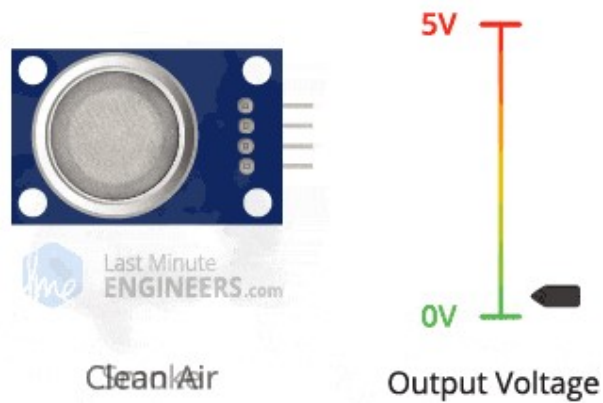


Figure 3.5.8

This analog signal is digitized by an LM393 High Precision Comparator and made available at the Digital Output (D0) pin.

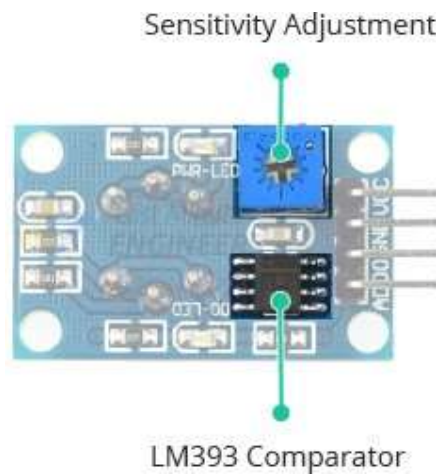


Figure 3.5.9

The module includes a potentiometer for adjusting the sensitivity of the digital output (D0). You can use it to set a threshold so that when the gas concentration exceeds the threshold value, the module outputs LOW otherwise HIGH.

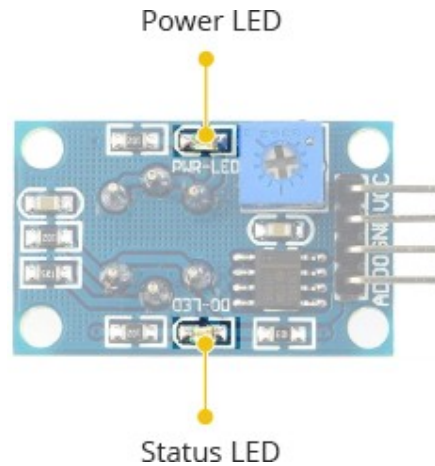


Figure 3.5.10

In addition, the module has two LEDs. The Power LED illuminates when the module is turned on, and the Status LED illuminates when the gas concentration exceeds the threshold value.

Technical Specifications

Here are the specifications:

Operating voltage	5V
Load resistance	20 KΩ
Heater resistance	33$\Omega \pm 5\%$
Heating consumption	<800mw
Sensing Resistance	10 KΩ – 60 KΩ
Concentration Range	200 – 10000ppm
Preheat Time	Over 24 hour

MQ2 Gas Sensor Module Pinout

supplies power to the module. Connect it to the 5V output of your Arduino. is the ground pin. indicates the presence of combustible gasses. D0 becomes LOW when the gas concentration exceeds the threshold value (as set by the potentiometer), and HIGH otherwise. produces an analog output voltage proportional to gas concentration, so a higher concentration results in a higher voltage and a lower concentration results in a lower voltage.

Calibrating the MQ2 Gas Sensor

Because the MQ2 is a heater-driven sensor, the calibration of the sensor may drift if it is left in storage for an extended period of time.

When first used after a long period of storage (a month or more), the sensor must be fully warmed up for 24-48 hours to ensure maximum accuracy.

If the sensor has recently been used, it will only take 5-10 minutes to fully warm up. During the warm-up period, the sensor typically reads high and gradually decreases until it stabilizes.

3.6 DHT11 (Temperature and Humidity Sensor)

The percentage of water present in the air is termed as humidity. Water as gaseous state called vapor. As the temperature of the air increases more water vapor can be generate.

Humidity measurement in industries is critical because it may affect the business cost of the product and the health and safety of the personnel. So, its huge importance of humidity sensor is very important, especially in the control systems for industrial processes like chemical gas purification, dryers, ovens, film desiccation, paper and textile production, and food processing. In agriculture, measurement of humidity is important for plantation protection (green house), soil moisture monitoring, etc.

Types of humidity:---

- 1) Relative Humidity = (density of water vapor / density of water vapor at saturation) x 100%
- 2) Absolute=Mass(vapour) / volume. Unit-grams/m³
- 3) Specific:Mass(vapour) / total mass.
- 4) Dew Point:Temperature(above 0°C) at which the water vapor in a gas condenses to liquid water)
- 5) Frost POINT: Temperature(below 0°C) at which the water vapor in a gas condenses to ice

Most Common:

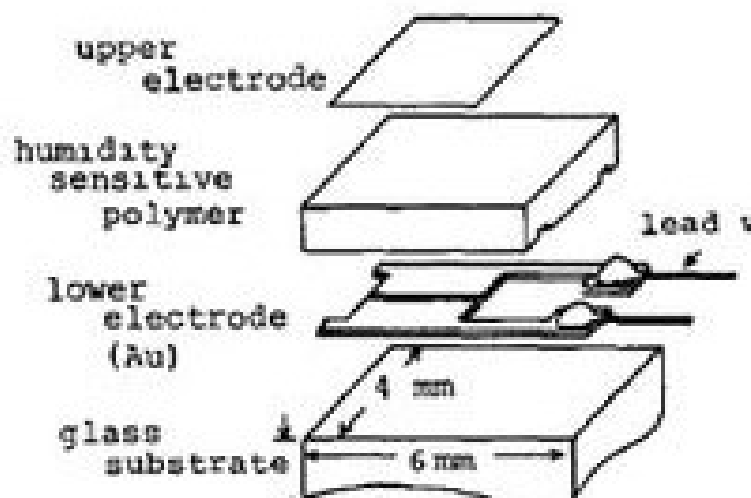
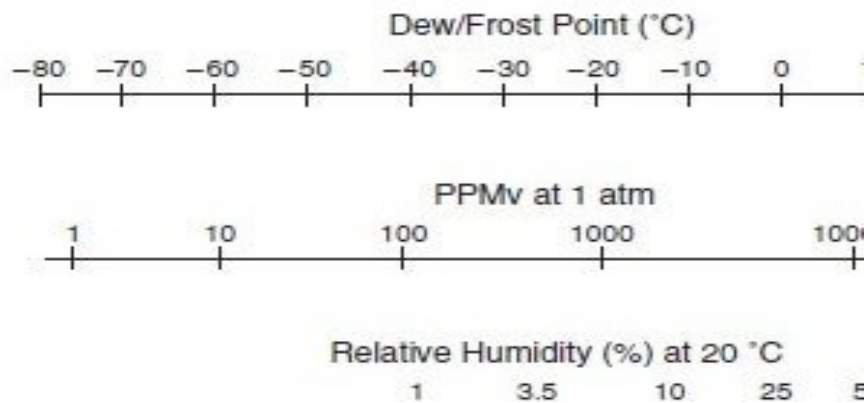
- Relative Humidity (RH),
- Dew/Frost point (D/F PT) and
- Parts Per Million (PPM) are used.

RH is a function of temperature, and thus it is a relative measurement.

Dew/Frost point is a function of the pressure of the gas but is independent of temperature and is defined as absolute humidity measurement.

PPM is also an absolute measurement.

Step 1: Working of Humidity Sensor



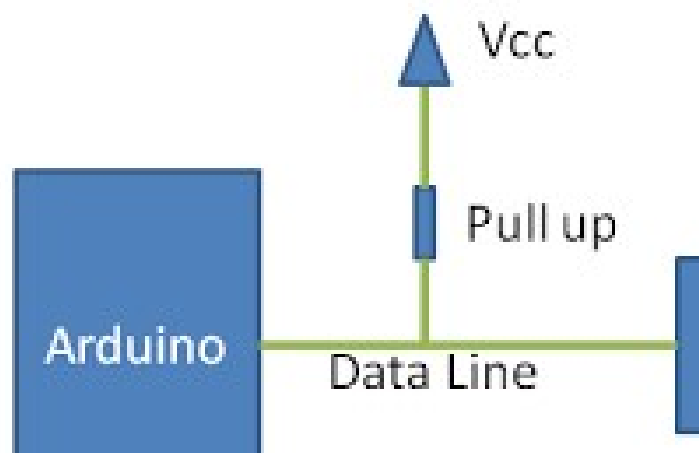
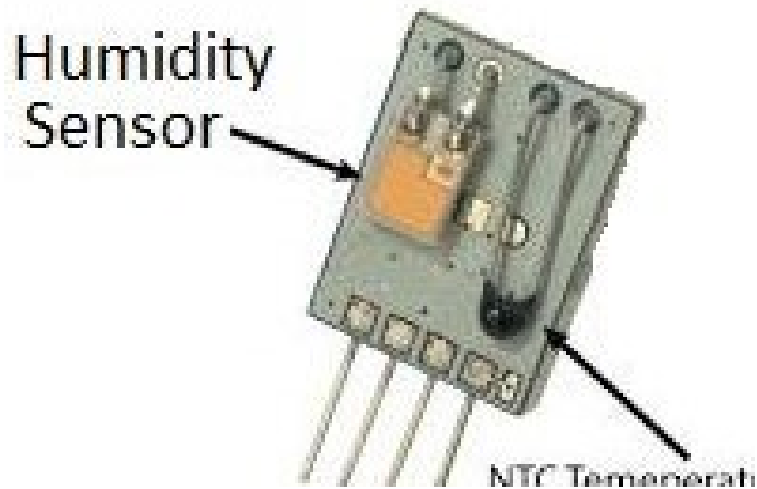


Figure 3.6

Most humidity sensors use capacitive measurement to determine the amount of moisture in the air.

This type of measurement relies on two electrical conductors with a non-conductive polymer film laying between them to create an electrical field between them. Moisture from the air collects on the film and causes changes in the voltage levels between the two plates. This change is then converted into a digital measurement of the air's relative humidity after taking the air temperature into account. Above figure can clear your doubt.

DHT11 Technical Specifications:

Humidity Range: 20-90% RH



Humidity Accuracy: $\pm 5\%$ RH

spacing and why copied you can type this specification

Temperature Range: 0-50 °C

Temperature Accuracy: $\pm 2\%$ °C

Operating Voltage: 3V to 5.5V

The DHT11 calculates relative humidity by measuring the electrical resistance between two electrodes.

The humidity sensing component of the DHT11 is a moisture holding substrate with the electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes while lower relative humidity increases the resistance between the electrodes.

The DHT11 converts the resistance measurement to relative humidity on a chip mounted to the back of the unit and transmits the humidity and temperature readings directly to the Arduino Nano.

3.7 BMP180 Barometric Pressure Sensor



The BMP180 Breakout is a barometric pressure sensor with an I²C ("Wire") interface. Barometric pressure sensors measure the absolute pressure of the air around them. This 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.

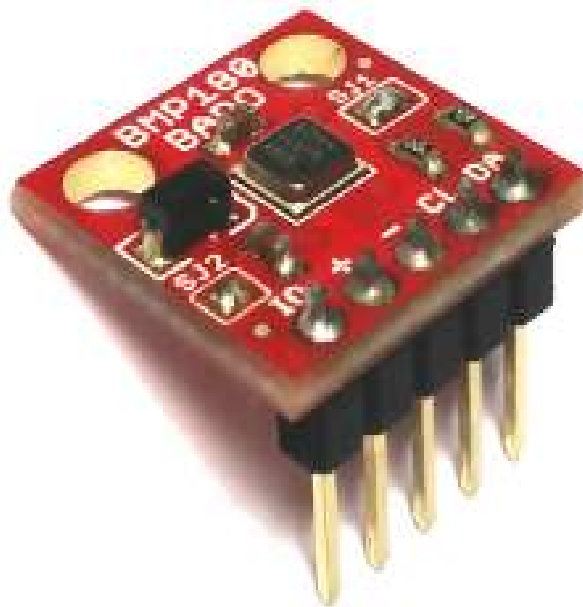


Figure 3.7.1

What is Atmospheric Pressure?

The definition of pressure is a force "pressing" on an area. A common unit of pressure is pounds per square inch (psi). One pound, pressing on one square inch, equals one psi. The SI unit is newtons per square meter, which are called pascals (Pa).

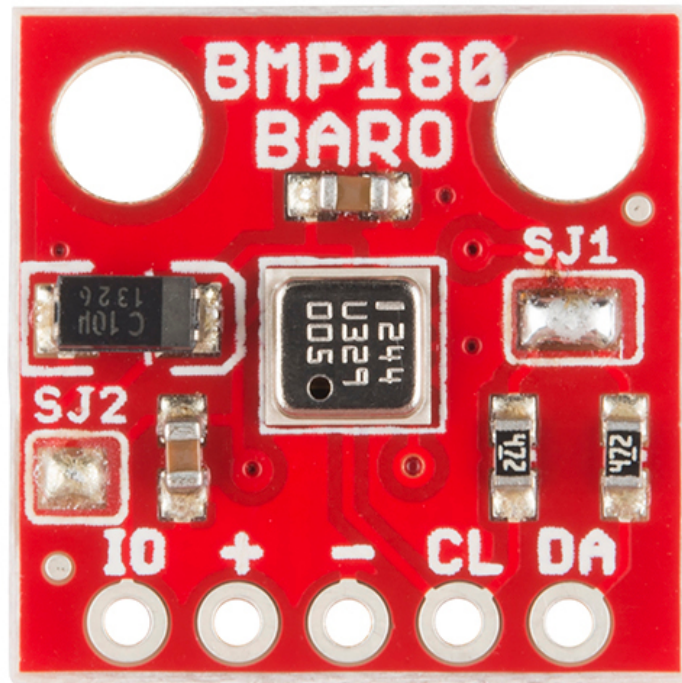


Figure 3.7.2

The BMP180 outputs absolute pressure in pascals (Pa). One pascal is a very small amount of pressure, approximately the amount that a sheet of paper will exert resting on a table. You will more often see measurements in hectopascals (1 hPa = 100 Pa) or kilopascals (1 kPa = 1000 Pa). The Arduino library we've provided outputs floating-point values in hPa, which also happens to equal one millibar (mbar).

Here are some conversions to other pressure units:

$$1 \text{ hPa} = 100 \text{ Pa} = 1 \text{ mbar} = 0.001 \text{ bar}$$

$$1 \text{ hPa} = 0.75006168 \text{ Torr}$$

$$1 \text{ hPa} = 0.01450377 \text{ psi (pounds per square inch)}$$

$$1 \text{ hPa} = 0.02953337 \text{ inHg (inches of mercury)}$$

$$1 \text{ hPa} = 0.00098692 \text{ atm (standard atmospheres)}$$

Temperature Effects

Because temperature affects the density of a gas, and density affects the mass of a gas, and mass affects the pressure (whew), atmospheric pressure will change dramatically with temperature. Pilots know this as "density altitude", which makes it easier to take off on a cold day than a hot one because the air is more dense and has a greater aerodynamic effect.

To compensate for temperature, the BMP180 includes a rather good temperature sensor as well as a pressure sensor. To perform a pressure reading, you first take a temperature reading, then combine that with a raw pressure reading to come up with a final temperature-compensated pressure measurement. (Don't worry, the Arduino library makes all of this very easy.)

Measuring Absolute Pressure

As we just mentioned, if your application requires measuring absolute pressure, all you have to do is get a temperature reading, then perform a pressure reading (see the [example sketch](#) for details). The final pressure reading will be in hPa = mbar. If you wish, you can convert this to a different unit using the above conversion factors.

Note that the absolute pressure of the atmosphere will vary with both your altitude and the current weather patterns, both of which are useful things to measure.

Determining Altitude

Since pressure varies with altitude, you can use a pressure sensor to measure altitude (with a few caveats).

The average pressure of the atmosphere at sea level is 1013.25 hPa (or mbar). This drops off to zero as you climb towards the vacuum of space. Because the curve of this drop-off is well understood, you can compute the altitude difference between two pressure measurements (p and p_0) by using this equation:

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0} \right)^{\frac{1}{5.255}} \right)$$

There are two ways you can take advantage of this.

1. If you use sea level pressure (1013.25 hPa) as the baseline pressure (p_0), the output of the equation will be your current altitude above sea level.
2. Or, if you take a single pressure reading at your current location, and use that as your baseline (p_0), all subsequent pressure readings will result in *relative* altitude changes from the baseline. Climb the stairs and you should see the altitude go from zero to 3 or 4 meters. Go down to the basement, and you'll see -3 or -4 meters. There's an example sketch included with the library called BMP180_altitude_example.ino that shows how to do this.

There's a function in the library called `altitude(P,P0)` that lets you accomplish both of these things. If you give it the sea level pressure (1013.25 hPa) for p_0 , and your local pressure for p , it will give you your altitude above sea level. If you use a local pressure measurement for p_0 , subsequent p pressure readings will give you your change in altitude from the baseline.

Now for the caveats:

Accuracy: How accurate is this? The theoretical noise level at the BMP180s highest resolution is 0.25m (about 10 inches), though in practice we see noise on the order of 1m (40 inches). You can improve the accuracy by taking a large number of readings and averaging them, although this will slow down your sample rate and response time.

Weather: You should also remember that pressure changes due to weather will affect your altitude readings. The best accuracy will be obtained if you take a "fresh" p_0 when you need it and don't rely on it to be accurate for extended periods due to changes in the weather.

Maximum altitude: The BMP180 can't measure all the way down to vacuum (or up to space). It's advertised lower limit is about 300 hPa (or mbar), which corresponds to an altitude of about 3000m or 30,000 feet. People have flown these to higher altitudes and gotten useful results, but this isn't guaranteed or likely to be accurate. (You might consider using GPS for high-altitude measurements).

Minimum altitude: Similarly, this sensor isn't suited for large pressures either. The advertised upper limit is 1100 hPa=mbar (or 16 psi), which is about 500 feet below sea level (that's in air - the BMP180 isn't submersible in water). This sensor isn't a good choice for submersible or compressed-gas measurements.

3.8 Bluetooth Module

HC-05 Bluetooth Module Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth. It has 6 pins,

1.Key/EN: It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

- . Data mode: Exchange of data between devices.
- Command mode: It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.

2. VCC: Connect 5 V or 3.3 V to this Pin.

3. GND: Ground Pin of module.

4. TXD: Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

5. RXD: Receive data serially (received data will be transmitted wirelessly by Bluetooth module).

6. State: It tells whether module is connected or not.

HC-05 module Information

- HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.
- This module works on 3.3V. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.
- As HC-05 Bluetooth module has 3.3V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.
- The data transfer rate of HC-05 module can vary up to 1Mbps is in the range of 10 meters. Specification of HC-05 Bluetooth Module

- Standby current: less than 2.5mA
- Sleep current: less than 1mA
- Interface: UART (Universal Asynchronous Receiver/Transmitter)
- Baud rates: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400,
- Overall, the HC-05 Bluetooth module is a versatile and cost-effective solution for adding Bluetooth connectivity to embedded systems and DIY projects, offering reliable wireless communication capabilities in a compact form factor.

3.9 Servo Motor

A servo motor is a type of rotary actuator that enables precise control of angular position. It comprises a motor, feedback mechanism (such as a potentiometer or encoder), and a controller. Servo motors are widely used in applications requiring controlled motion, such as robotics, RC vehicles, automation, and industrial machinery. They offer high precision, speed, and torque, making them suitable for tasks that demand accuracy and responsiveness.

•we are using plastic and metal servo motors,there are total 12 servo motor used in these project .

•About plastic servo motor-

A plastic servo motor refers to a servo motor with a plastic housing or casing. These servo motors are often more lightweight and cost-effective compared to their metal counterparts. They are commonly used in hobbyist projects, RC vehicles, and other applications where weight and cost are important factors. While plastic servo motors may not be as durable or robust as metal ones, they still offer good performance and reliability for many applications.

•About metal servo motor-

A metal servo motor refers to a servo motor with a metal housing or casing, typically made of aluminum or other alloys. These servo motors are known for their durability, strength, and resistance to wear and tear. They are commonly used in industrial applications, robotics, aerospace, and other environments where ruggedness and reliability are crucial. Metal servo motors offer higher torque and are better suited for heavy-duty tasks compared to their plastic counterparts. They are often preferred in applications where precision and longevity are essential.



Figure 3.9

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

It consists of three parts:

1. Controlled device
2. Output sensor
3. Feedback system

It is a closed-loop system where it uses a positive feedback system to control motion and the final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

Here reference input signal is compared to the reference output signal and the third signal is produced by the feedback system. And this third signal acts as an input signal to the control the device. This signal is present as long as the feedback signal is generated or there is a difference between the reference input signal and reference output signal. So the main task of servomechanism is to maintain the output of a system at the desired value at presence of noises.

3.10 DC-DC Buck Converter

A DC-DC buck converter is a type of power converter used to step down (reduce) a DC voltage level to a lower voltage level while maintaining a stable output voltage. Here's a detailed description of its operation and features:



- **Input Voltage:** The buck converter takes a higher DC input voltage and converts it to a lower DC output voltage. The input voltage can vary, depending on the specific design and application requirements of the converter.
- **Switching Element:** The heart of the buck converter is a switching element, typically a MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) or a BJT (Bipolar Junction Transistor). This switching element is controlled by a pulse-width modulation (PWM) signal generated by the converter's control circuitry.
- **Inductor:** The buck converter includes an inductor connected in series with the load and the switching element. During the switching process, energy is stored in the inductor when the switching element is on, and then released to the load when the switching element is off.
- **Diode:** A diode, often called a freewheeling diode or a catch diode, is connected in parallel with the load to provide a path for the inductor current when the switching element is off. This diode prevents voltage spikes and ensures smooth operation of the converter.
- **Control Circuitry:** The buck converter contains control circuitry that regulates the duty cycle of the PWM signal applied to the switching element. By adjusting the duty cycle, the converter can regulate the output voltage to the desired level, compensating for variations in input voltage and load conditions.
- **Output Voltage Regulation:** The buck converter provides stable and regulated output voltage, typically through feedback control mechanisms. Feedback components such as resistors and capacitors are used to monitor the output voltage and adjust the duty cycle of the PWM signal accordingly to maintain the desired output voltage level.
- **Efficiency:** Buck converters are known for their high efficiency, especially when stepping down voltage levels. The efficiency of a buck converter is typically greater than 80% and can exceed 90% in many cases, depending on the specific design and operating conditions.
- **Applications:** Buck converters are widely used in various electronic devices and systems where efficient voltage conversion is required. Common applications include power supplies for portable devices, battery chargers, LED drivers, motor control systems, and renewable energy systems.
- **Overall,** the DC-DC buck converter offers a compact, efficient, and reliable solution for stepping down DC voltages, making it an essential component in modern electronics and power systems.

3.11 Lithium-ion battery

A lithium-ion (Li-ion) battery is a type of rechargeable battery that uses lithium ions as the main component of its electrochemical system. Here's a detailed description of its structure, operation, and features:

Composition: A lithium-ion battery consists of several key components:

Positive Electrode (Cathode): Typically made of lithium cobalt oxide (LiCoO_2), lithium iron phosphate (LiFePO_4), or other lithium-based compounds. The positive electrode receives lithium ions during charging.

Negative Electrode (Anode): Usually composed of carbon/graphite materials, which intercalate lithium ions during charging.

Separator: A porous membrane that physically separates the positive and negative electrodes while allowing the flow of lithium ions.

Electrolyte: A conductive solution containing lithium salts, such as lithium hexafluorophosphate (LiPF_6), dissolved in a solvent. The electrolyte facilitates the movement of lithium ions between the electrodes.

Ion Movement: During charging, lithium ions are extracted from the positive electrode and migrate through the electrolyte to the negative electrode, where they are stored. During discharging, the lithium ions move back to the positive electrode, releasing electrical energy in the process.

Voltage: The nominal voltage of a lithium-ion cell is typically around 3.7 volts. However, the actual voltage can vary depending on the specific chemistry and design of the battery.

Rechargeability: Lithium-ion batteries are rechargeable, meaning they can be charged and discharged multiple times without significant degradation in performance. They have a high energy density, allowing them to store a large amount of energy relative to their size and weight.

Protection Circuit: Many lithium-ion batteries include a built-in protection circuit that monitors voltage, current, and temperature to prevent overcharging, over-discharging, and short circuits.

Thermal Management: Some batteries incorporate thermal management systems to dissipate heat and prevent overheating during charging and discharging.

Venting Mechanisms: In rare cases of extreme abuse or failure, lithium-ion batteries may include venting mechanisms to release built-up pressure and prevent rupture or explosion.

Applications: Lithium-ion batteries are widely used in portable electronic devices such as smartphones, laptops, tablets, and digital cameras, as well as in electric vehicles (EVs), hybrid electric vehicles (HEVs), and energy storage systems for renewable energy sources.

Charging Characteristics: Lithium-ion batteries require specific charging protocols to optimize performance and lifespan. Overcharging or charging at high temperatures can degrade the battery and reduce its lifespan.

Basic Characteristics:

5.1 Capacity (25±5°C)	Nominal Capacity: 2600mAh (0.52A Discharge, 2.75V) Typical Capacity: 2550mAh (0.52A Discharge, 2.75V) Minimum Capacity: 2500mAh (0.52A Discharge, 2.75V)
5.2 Nominal Voltage	3.7V
5.3 Internal Impedance	≤70mΩ
5.4 Discharge Cut-off Voltage	3.0V
5.5 Max Charge Voltage	4.20±0.05V
5.6 Standard Charge Current	0.52A
5.7 Rapid Charge Current	1.3A
5.8 Standard Discharge Current	0.52A
5.9 Rapid Discharge Current	1.3A
5.10 Max Pulse Discharge Current	2.6A
5.11 Weight	46.5±1g
5.12 Max. Dimension	Diameter(Ø): 18.4mm Height(H): 65.2mm
5.13 Operating Temperature	Charge: 0 ~ 45°C Discharge: -20~60°C
5.14 Storage Temperature	During 1 month: -5~35°C During 6 months: 0~35°C

3.12 Power Management

Power management of lithium-ion batteries involves various techniques to optimize their performance, extend their lifespan, and ensure safe operation. Here are some key aspects:

State of Charge (SOC) Monitoring: Monitoring the SOC helps prevent overcharging (which can degrade the battery) and deep discharging (which can damage it). It involves accurately measuring the amount of charge remaining in the battery.

Battery Management Systems (BMS): BMS monitors and manages various parameters of the battery, including voltage, current, temperature, and SOC. It ensures the battery operates within safe limits and provides protection against overcharging, over-discharging, and overheating.

Temperature Management: Keeping the battery within its optimal temperature range (usually between 15-25°C or 59-77°F) helps maintain its performance and prolong its lifespan. Cooling or heating systems may be employed to regulate temperature.

Charge/Discharge Control: Controlling the rate and depth of charge/discharge can optimize battery performance. Techniques such as fast charging, slow charging, and limiting the maximum discharge current can be used based on specific requirements.

Cell Balancing: Balancing ensures that each cell within a battery pack has a similar SOC and voltage. This helps prevent overcharging of individual cells and maximizes the overall capacity of the battery pack.

Load Management: Managing the power demands from connected devices or systems can help optimize the battery's usage and prolong its runtime. This involves intelligently prioritizing power consumption and potentially reducing non-essential loads.

Predictive Maintenance: Utilizing algorithms and sensors to predict battery health and performance degradation allows for proactive maintenance, reducing the risk of unexpected failures and optimizing the battery's lifespan.

Energy Harvesting: In some applications, energy harvesting techniques can be employed to capture and store energy from ambient sources such as solar or kinetic energy, supplementing the battery's power and extending its runtime.

3.13 Data Transmission to ThingSpeak

ThingSpeak is an IoT (Internet of Things) platform that allows users to collect, analyze, and visualize data from sensors or devices in real-time. Here's a detailed explanation of how data is transferred through ThingSpeak:

Data Acquisition: Devices equipped with sensors collect data from their environment. This data could be temperature, humidity, pressure, motion, or any other parameter depending on the sensors installed. The devices typically have a microcontroller or microprocessor that processes the sensor data and prepares it for transmission.

Data Transmission to ThingSpeak: ThingSpeak supports various communication protocols for data transmission, including HTTP, HTTPS, MQTT, and others. The device sends the data to the ThingSpeak platform using one of these protocols. For example, it might make an HTTP POST request to send the data to a specific ThingSpeak channel.

ThingSpeak Channel: ThingSpeak channels act as containers for the collected data. Each channel has fields where specific data values are stored. When setting up a ThingSpeak channel, users define the structure of the data by specifying the number and names of fields.

API Key: To ensure secure communication between devices and the ThingSpeak platform, each channel is associated with an API key. Devices use this API key to authenticate themselves when sending data to the channel.

Data Storage: Upon receiving the data, ThingSpeak stores it in the corresponding fields of the channel. Each entry in the channel represents a data point with a timestamp.

Data Processing and Visualization: ThingSpeak offers built-in tools for data analysis and visualization. Users can create MATLAB® scripts or use built-in MATLAB functions to perform custom analyses on the collected data. Visualizations such as charts, gauges, and maps can be created to display the data in real-time or over a specified time period.

Alerts and Reactivity: ThingSpeak allows users to set up alerts based on predefined conditions. When a condition is met, ThingSpeak can trigger notifications via email, SMS, or other channels..

Data is transferred through ThingSpeak by devices sending data to ThingSpeak channels using communication protocols like HTTP or MQTT.

CHAPTER 4

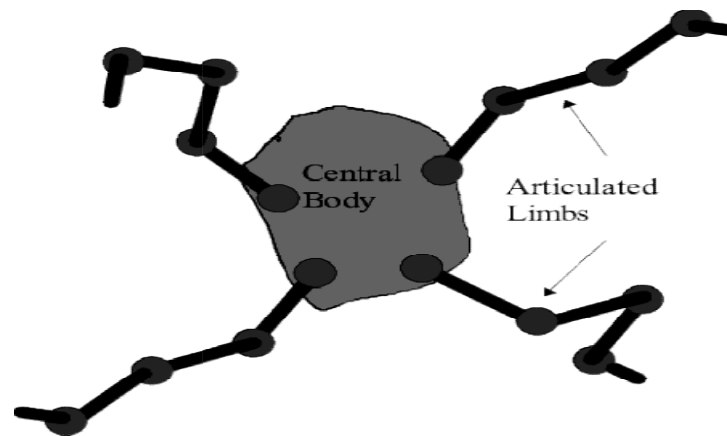
Design and Implementation

4.1 Spider Robot Structure

Introduction:

no need of intro. again . here expalin how the design done

Spider robots represent a fascinating branch of robotics, drawing inspiration from the agile and adaptable locomotion of arachnids. This section provides an in-depth exploration of the design and components comprising the structure of a spider robot, elucidating the intricate engineering principles and functional considerations involved in its development.



Leg Configuration:

At the core of the spider robot's structure lies its quadrupedal leg configuration, meticulously crafted to emulate the biomechanical prowess of spiders. Each leg is composed of multiple segments, typically encompassing a coxa, femur, tibia, and tarsus, interconnected by articulating joints. This modular leg architecture affords the robot unparalleled versatility in navigating diverse terrains, enabling fluid movements ranging from precise steps to agile leaps.

Materials and Construction:

The choice of materials for constructing the spider robot is a pivotal determinant of its performance and durability. Engineers often opt for advanced composite materials such as carbon fiber-reinforced polymers or lightweight metals like aluminum alloys to strike a delicate balance between structural integrity and weight optimization. Employing cutting-edge manufacturing techniques such as additive manufacturing or CNC machining ensures meticulous fabrication of

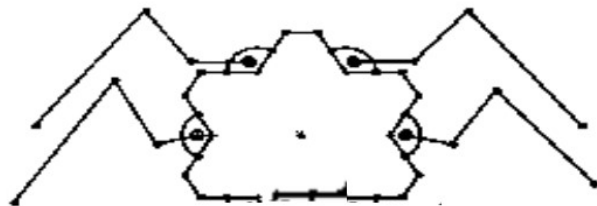
intricate components, fostering an optimal blend of strength, flexibility, and dimensional accuracy.

Actuation Mechanism:

Central to the locomotion of the spider robot is its sophisticated actuation mechanism, which imparts controlled motion to its articulated legs. Each joint within the leg assembly is driven by high-torque servo motors or electric actuators, meticulously calibrated to deliver precise torque and angular displacement. The actuation system is engineered to synchronize the movement of multiple joints, enabling the robot to execute intricate gaits and traverse challenging terrain with remarkable dexterity and efficiency.

Control and Coordination:

The control architecture of the spider robot serves as the proverbial brain, orchestrating the harmonious coordination of its leg movements. Housed within the robot's central processing unit, sophisticated motion control algorithms govern the activation of individual actuators based on predefined gait patterns or sensory inputs. Real-time feedback from an array of proprioceptive and exteroceptive sensors, including encoders, accelerometers, and force sensors, informs the control algorithms, enabling dynamic adaptation to changing environmental conditions and terrain topology.



Sensory Integration:

A hallmark of the spider robot's structural design is its adept integration of sensory systems, enabling perceptive interaction with its surroundings. Proprioceptive sensors embedded within each leg provide feedback on joint angles and limb position, facilitating precise proprioception and kinematic awareness. Exteroceptive sensors, such as proximity sensors, infrared rangefinders, and depth cameras, augment the robot's perception capabilities,

enabling robust obstacle detection, terrain mapping, and environmental monitoring.

Redundancy and Robustness:

In recognition of the exigencies of real-world deployment, the spider robot structure incorporates redundant mechanisms to bolster resilience and fault tolerance. Redundancy features, such as duplicate actuators or sensor arrays, are strategically integrated to mitigate the impact of component failures and ensure uninterrupted operation in adverse conditions. Furthermore, modular design principles facilitate rapid component replacement and system reconfiguration, underscoring the robot's resilience and adaptability in the face of unforeseen challenges.

Conclusion:

In summation, the structure of the spider robot epitomizes the convergence of meticulous engineering design and biomimetic inspiration, culminating in a versatile and resilient robotic platform. Through the seamless integration of advanced materials, precision actuation mechanisms, sophisticated control algorithms, sensory perception systems, and redundant architectures, the spider robot stands poised to revolutionize myriad domains, ranging from exploration and reconnaissance to disaster response and industrial automation. As research endeavors continue to push the boundaries of innovation, the spider robot remains an emblem of the inexorable march towards the realization of agile and adaptive robotic systems.

4.2 Arduino Nano Programming

Introduction:

Arduino Nano programming entails harnessing the capabilities of the Arduino Nano microcontroller board, a compact yet versatile platform designed for prototyping and developing embedded projects. In this detailed exploration, we delve into the intricacies of Arduino Nano programming, covering software and hardware aspects, programming techniques, and practical applications.

Arduino Nano Board Overview:

The Arduino Nano board features an Atmel ATmega328P microcontroller, clocked at 16MHz, and offers an extensive array of digital and analog input/output pins. With its diminutive form factor and USB connectivity, the Arduino Nano is particularly well-suited for projects with space constraints. Onboard peripherals include UART, SPI, and I2C interfaces, PWM pins for analog output, and analog-to-digital converters (ADC) for analog input, providing ample options for interfacing with external sensors, actuators, and devices.



Arduino Integrated Development Environment (IDE):

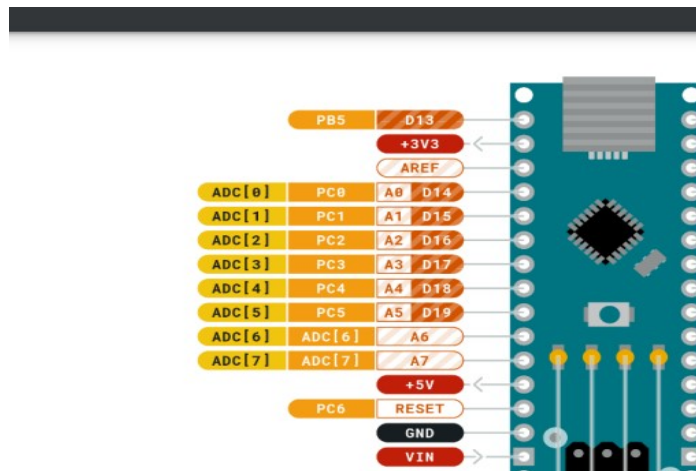
The Arduino IDE serves as the primary software tool for Arduino Nano programming, providing an intuitive and user-friendly interface for code development. Equipped with a built-in compiler and uploader, the IDE streamlines the process of writing, compiling, and uploading code to the Arduino Nano board. Furthermore, the IDE offers a vast repository of libraries and example code, simplifying the implementation of complex functionalities and reducing development time.

Basic Programming Concepts:

Arduino Nano programming hinges on fundamental programming concepts such as variables, data types, control structures, functions, and libraries. The Arduino programming language, a variant of C/C++, serves as the foundation for writing code to define the behavior and functionality of Arduino projects. A solid understanding of these programming concepts is essential for crafting efficient, readable, and maintainable code tailored to the Arduino Nano platform.

Input/Output (IO) Handling:

Arduino Nano facilitates seamless interaction with external devices through its digital and analog input/output pins. Digital pins can be configured as inputs to read binary data or outputs to control digital devices. Analog pins enable the acquisition of analog signals from sensors or the generation of analog output voltages using pulse-width modulation (PWM) techniques. Effective IO handling is crucial for interfacing with a diverse range of sensors, actuators, and peripherals, enabling data acquisition, control, and communication.



Sensor and Actuator Interfacing:

The versatility of Arduino Nano extends to interfacing with a myriad of sensors and actuators, facilitating the integration of various environmental monitoring, control, and automation functionalities into projects. Sensors such as temperature, humidity, motion, proximity, and light sensors can be seamlessly interfaced with Arduino Nano to gather environmental data. Actuators including

motors, servos, LEDs, relays, and displays can be controlled to effectuate physical actions or visual feedback based on sensor inputs or user commands.

Advanced Programming Techniques:

As proficiency in Arduino Nano programming grows, users can explore advanced programming techniques to enhance project functionality, performance, and scalability. These techniques may encompass multitasking using interrupts or timers to execute multiple tasks concurrently, implementing communication protocols such as MQTT, Bluetooth, or Wi-Fi for wireless connectivity, optimizing code for memory and performance efficiency, and leveraging external libraries or APIs to extend functionality and capabilities beyond the built-in features of the Arduino Nano platform.

Practical Applications:

Arduino Nano finds extensive applications across a spectrum of domains, including home automation, robotics, Internet of Things (IoT), wearable technology, educational projects, and prototyping. Examples range from building smart home devices such as automated lighting and environmental monitoring systems to developing robotic platforms for education or entertainment. Additionally, Arduino Nano serves as an invaluable tool for educators, students, hobbyists, and professionals alike to explore electronics, programming, and technology in a hands-on and engaging manner.

Conclusion:

In conclusion, Arduino Nano programming empowers individuals to explore the realms of embedded systems development, fostering creativity, innovation, and learning in the process. With its accessible IDE, robust hardware capabilities, extensive library support, and vibrant community, Arduino Nano serves as a catalyst for transforming ideas into tangible projects with real-world impact. As Arduino Nano programming continues to evolve and evolve, its potential for driving technological innovation, education, and societal change remains boundless, inspiring a new generation of creators, inventors, and innovators to push the boundaries of what's possible with embedded electronics.

```
sketch_apr8b | Arduino IDE 2.0.3
File Edit Sketch Tools Help
Arduino Nano

sketch_apr8b.ino sketch_apr8b.ino
1 #include <Servo.h> //to define and control servos
2 #include <FlexiTimer2.h> //to set a timer to manage all servos
3 /* Servos -----*/
4 //define 12 servos for 4 legs
5 Servo servo[4][3];
6 //define servos' ports
7 const int servo_pin[4][3] = { {2, 3, 4}, {5, 6, 7}, {8, 9, 10}, {11, 12, 13} };
8 /* Size of the robot -----*/
9 const float length_a = 55;
10 const float length_b = 77.5;
11 const float length_c = 27.5;
12 const float length_side = 71;
13 const float z_absolute = -28;
14 /* Constants for movement -----*/
15 const float z_default = -50, z_up = -30, z_boot = z_absolute;
16 const float x_default = 62, x_offset = 0;
17 const float y_start = 0, y_step = 40;
18 const float y_default = x_default;
19 /* variables for movement -----*/
20 volatile float site_now[4][3]; //real-time coordinates of the end of each leg
21 volatile float site_expect[4][3]; //expected coordinates of the end of each leg
22 float temp_speed[4][3]; //each axis' speed, needs to be recalculated before each movement
23 float move_speed; //movement speed
24 float speed_multiple = 1; //movement speed multiple
25 const float spot_turn_speed = 4;
26 const float leg_move_speed = 8;
27 const float body_move_speed = 3;
28 const float stand_seat_speed = 1;

sketch_apr8b | Arduino IDE 2.0.3
File Edit Sketch Tools Help
Arduino Nano

sketch_apr8b.ino sketch_apr8b.ino
29 volatile int rest_counter; //+1/0.02s, for automatic rest
30 //functions' parameter
31 const float KEEP = 255;
32 //define PI for calculation
33 const float pi = 3.1415926;
34 /* Constants for turn -----*/
35 //temp length
36 const float temp_a = sqrt(pow(2 * x_default + length_side, 2) + pow(y_step, 2));
37 const float temp_b = 2 * (y_start + y_step) + length_side;
38 const float temp_c = sqrt(pow(2 * x_default + length_side, 2) + pow(2 * y_start + y_step + length_side, 2));
39 const float temp_alpha = acos((pow(temp_a, 2) + pow(temp_b, 2) - pow(temp_c, 2)) / 2 / temp_a / temp_b);
40 //site for turn
41 const float turn_x1 = (temp_a - length_side) / 2;
42 const float turn_y1 = y_start + y_step / 2;
43 const float turn_x0 = turn_x1 - temp_b * cos(temp_alpha);
44 const float turn_y0 = temp_b * sin(temp_alpha) - turn_y1 - length_side;
45 /* -----*/
46
47 /*
48  * setup function
49  * -----*/
50
51 #include <SoftwareSerial.h>
52 SoftwareSerial BTSerial(A4, A5); // RX | TX
53 String readdata;
54
55 void setup()
56 {
57 //start serial for debug
58 Serial.begin(9600);
59 Serial.println("Robot starts initialization");
60 BTSerial.begin(9600); // HC-05 default speed in AT
61
62 //initialize default parameter
63 set_site(0, x_default - x_offset, y_start + y_step, z_boot);
64 set_site(1, x_default - x_offset, y_start + y_step, z_boot);
65 set_site(2, x_default + x_offset, y_start + y_step, z_boot);
66 set_site(3, x_default + x_offset, y_start + y_step, z_boot);
67 for (int i = 0; i < 4; i++)
68 {
69 for (int j = 0; j < 3; j++)
70 {
71 site_now[i][j] = site_expect[i][j];
72 }
73 }
74 //start servo service
75 FlexiTimer2::set(20, servo_service);
76 FlexiTimer2::start();
77 Serial.println("Servo service started");
78 //initialize servos
79 servo_attach();
80 Serial.println("Servos initialized");
81 Serial.println("Robot initialization Complete");
82 Serial.println("Stand");
83 stand();
84 delay(2000);
85 }

sketch_apr8b | Arduino IDE 2.0.3
File Edit Sketch Tools Help
Arduino Nano

sketch_apr8b.ino sketch_apr8b.ino
57 //start serial for debug
58 Serial.begin(9600);
59 Serial.println("Robot starts initialization");
60 BTSerial.begin(9600); // HC-05 default speed in AT
61
62 //initialize default parameter
63 set_site(0, x_default - x_offset, y_start + y_step, z_boot);
64 set_site(1, x_default - x_offset, y_start + y_step, z_boot);
65 set_site(2, x_default + x_offset, y_start + y_step, z_boot);
66 set_site(3, x_default + x_offset, y_start + y_step, z_boot);
67 for (int i = 0; i < 4; i++)
68 {
69 for (int j = 0; j < 3; j++)
70 {
71 site_now[i][j] = site_expect[i][j];
72 }
73 }
74 //start servo service
75 FlexiTimer2::set(20, servo_service);
76 FlexiTimer2::start();
77 Serial.println("Servo service started");
78 //initialize servos
79 servo_attach();
80 Serial.println("Servos initialized");
81 Serial.println("Robot initialization Complete");
82 Serial.println("Stand");
83 stand();
84 delay(2000);
```


4.3 ESP32 Programming

Introduction:

ESP32 programming represents a pivotal frontier in embedded systems development, offering a multifaceted platform for realizing IoT solutions, robotics applications, and smart devices. This thesis undertakes an exhaustive examination of ESP32 programming, delving deep into its architecture, programming paradigms, Wi-Fi integration, real-world applications, challenges, and future directions to elucidate its significance and transformative potential in the domain of embedded systems.



ESP32 Microcontroller Architecture:

The ESP32 microcontroller, meticulously engineered by Espressif Systems, embodies a sophisticated architecture tailored to address the demands of contemporary embedded applications. Anchored by a dual-core Xtensa LX6 CPU clocked at frequencies of up to 240MHz, the ESP32 boasts formidable processing capabilities essential for multitasking, real-time operations, and computational-intensive tasks. Augmenting its computational prowess are integrated Wi-Fi and Bluetooth connectivity modules, facilitating seamless

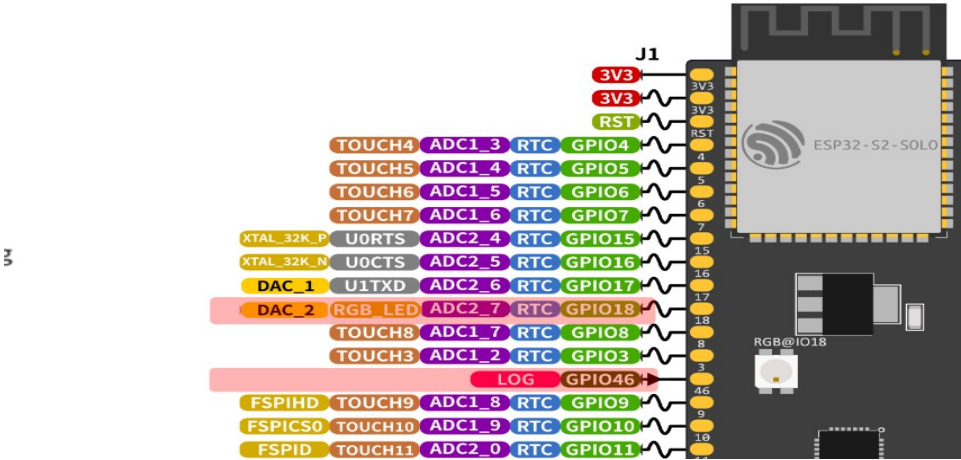
wireless communication, while an expansive array of GPIO pins, SPI, I2C, and UART interfaces provide versatile avenues for interfacing with sensors, actuators, and external devices.

Programming Methodologies:

ESP32 programming is characterized by its diverse array of methodologies, each tailored to cater to specific project requirements, development preferences, and proficiency levels. The Espressif IoT Development Framework (ESP-IDF) stands as a comprehensive ecosystem for low-level programming, affording developers direct access to hardware features and fine-grained control over system resources. Conversely, the Arduino IDE offers an intuitive and beginner-friendly environment equipped with a rich repository of pre-built functions and simplified syntax, ideal for rapid prototyping and iterative development cycles.

Core Concepts and Techniques:

A nuanced grasp of core concepts and techniques forms the bedrock of proficiency in ESP32 programming. Topics such as task scheduling, memory management, peripheral configuration, and communication protocols constitute the fundamental building blocks of embedded systems development on the ESP32 platform. Advanced techniques, including interrupt handling, power management, and wireless networking optimization, serve to elevate the capabilities and efficiency of ESP32-based projects, unlocking new dimensions for innovation and performance optimization.



Wi-Fi Integration:

Wi-Fi integration stands as a cornerstone feature of ESP32 programming, empowering seamless connectivity to local and global networks. Leveraging the ESP32's robust Wi-Fi capabilities, developers can configure Wi-Fi parameters, establish connections to access points, and implement advanced features such as Wi-Fi scanning, network configuration, and secure data transmission. Wi-Fi integration facilitates communication with cloud services, inter-device data exchange, and access to online resources, thereby catalyzing the realization of a myriad of IoT applications and services.

Real-World Applications:

The real-world impact of ESP32 programming spans a diverse spectrum of domains, spanning from smart home automation and industrial IoT to environmental monitoring and wearable technology. Case studies and practical examples offer compelling insights into the deployment of ESP32-based solutions in contexts such as sensor networks, remote monitoring systems, autonomous vehicles, and smart appliances. These applications underscore the versatility, scalability, and adaptability of ESP32 programming in addressing contemporary challenges and driving technological innovation.

Challenges and Future Directions:

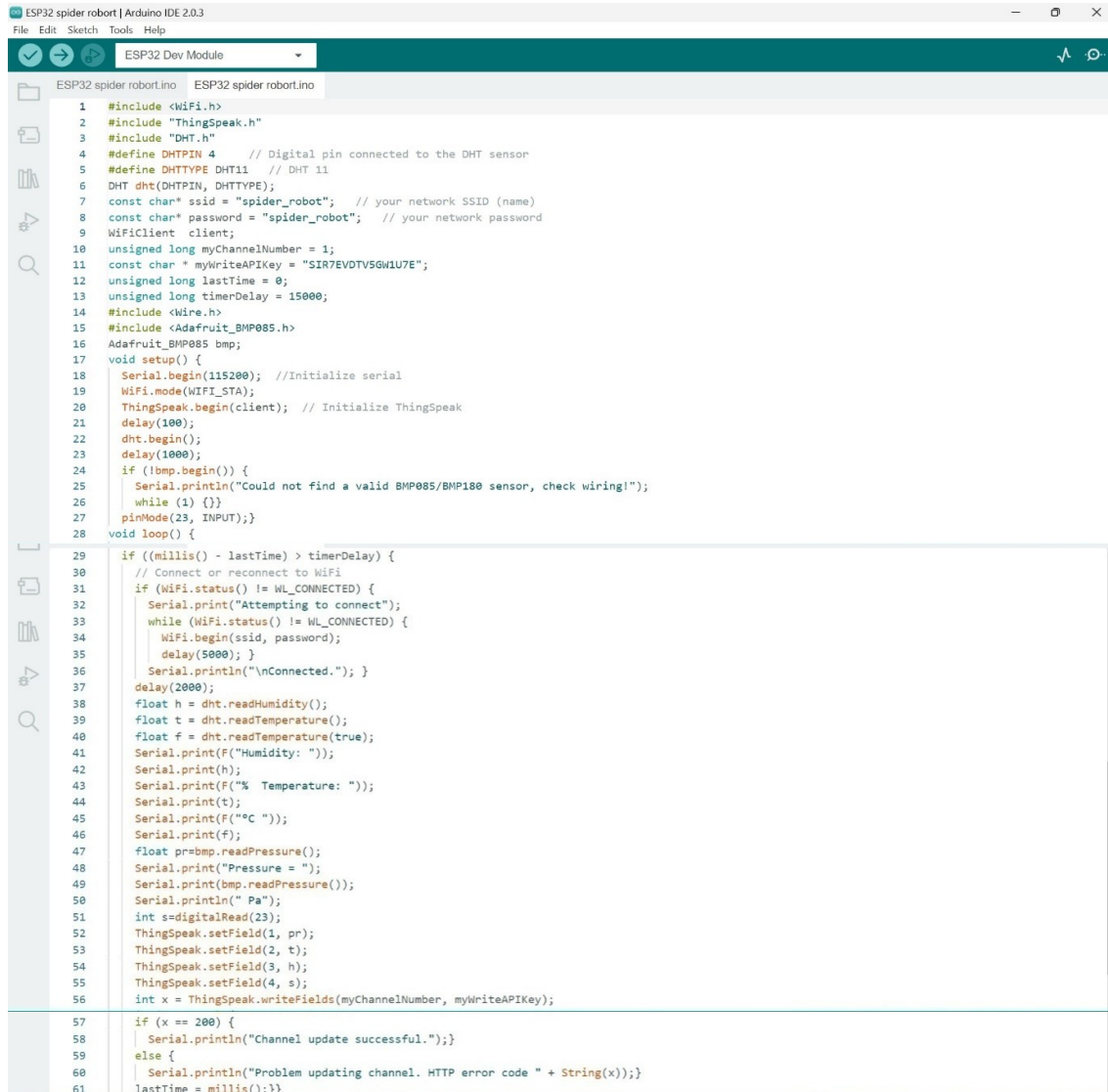
Despite its myriad advantages, ESP32 programming presents an array of challenges, ranging from power consumption optimization and security vulnerabilities to compatibility with legacy systems. Tackling these challenges necessitates sustained research and development efforts aimed at refining programming methodologies, enhancing hardware capabilities, and fortifying security measures. Furthermore, emerging trends such as edge computing, artificial intelligence, and 5G integration offer tantalizing prospects for future exploration and innovation in ESP32 programming, heralding a new era of embedded systems excellence.

Conclusion:

In conclusion, ESP32 programming emerges as a linchpin of embedded systems development, embodying a potent fusion of power, versatility, and accessibility. Through meticulous exploration of its architecture, programming

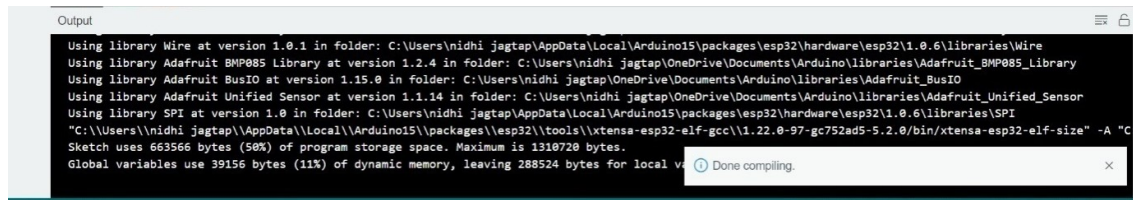
methodologies, core concepts, Wi-Fi integration, real-world applications, challenges, and future directions, this thesis underscores the profound impact of ESP32 programming in shaping the landscape of connected devices and IoT ecosystems. As researchers and practitioners continue to push the boundaries of innovation, ESP32 programming remains poised to drive transformative change and usher in a new epoch of embedded systems excellence.

ESP32 code:



```
1 #include <WiFi.h>
2 #include "ThingSpeak.h"
3 #include "DHT.h"
4 #define DHTPIN 4 // Digital pin connected to the DHT sensor
5 #define DHTTYPE DHT11 // DHT 11
6 DHT dht(DHTPIN, DHTTYPE);
7 const char* ssid = "spider_robot"; // your network SSID (name)
8 const char* password = "spider_robot"; // your network password
9 WiFiClient client;
10 unsigned long myChannelNumber = 1;
11 const char * myWriteAPIKey = "SIR7EVDTV5GW1U7E";
12 unsigned long lastTime = 0;
13 unsigned long timerDelay = 15000;
14 #include <Wire.h>
15 #include <Adafruit_BMP085.h>
16 Adafruit_BMP085 bmp;
17 void setup() {
18   Serial.begin(115200); //Initialize serial
19   WiFi.mode(WIFI_STA);
20   ThingSpeak.begin(client); // Initialize ThingSpeak
21   delay(100);
22   dht.begin();
23   delay(1000);
24   if (!bmp.begin()) {
25     Serial.println("Could not find a valid BMP085/BMP180 sensor, check wiring!");
26     while (1) {}
27   }
28   pinMode(23, INPUT);
29   void loop() {
30     if ((millis() - lastTime) > timerDelay) {
31       // Connect or reconnect to WiFi
32       if (WiFi.status() != WL_CONNECTED) {
33         Serial.print("Attempting to connect");
34         while (WiFi.status() != WL_CONNECTED) {
35           WiFi.begin(ssid, password);
36           delay(5000);
37         }
38         Serial.println("\nConnected.");
39         delay(2000);
40         float h = dht.readHumidity();
41         float t = dht.readTemperature();
42         float f = dht.readTemperature(true);
43         Serial.print(F("Humidity: "));
44         Serial.print(h);
45         Serial.print(F("% Temperature: "));
46         Serial.print(t);
47         Serial.print(F("°C "));
48         Serial.print(f);
49         float pr=bmp.readPressure();
50         Serial.print("Pressure = ");
51         Serial.print(bmp.readPressure());
52         Serial.println(" Pa");
53         int s=digitalRead(23);
54         ThingSpeak.setField(1, pr);
55         ThingSpeak.setField(2, t);
56         ThingSpeak.setField(3, h);
57         ThingSpeak.setField(4, s);
58         int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
59         if (x == 200) {
60           Serial.println("Channel update successful.");
61         }
62         else {
63           Serial.println("Problem updating channel. HTTP error code " + String(x));
64         }
65         lastTime = millis();
66       }
67     }
68   }
69 }
```

RESULTS/OUTPUT:



```
Output
Using library Wire at version 1.0.1 in folder: C:\Users\nidhi jagtap\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.6\libraries\Wire
Using library Adafruit BMP085 Library at version 1.2.4 in folder: C:\Users\nidhi jagtap\OneDrive\Documents\Arduino\libraries\Adafruit_BMP085_Library
Using library Adafruit BusIO at version 1.15.0 in folder: C:\Users\nidhi jagtap\OneDrive\Documents\Arduino\libraries\Adafruit_BusIO
Using library Adafruit Unified Sensor at version 1.1.14 in folder: C:\Users\nidhi jagtap\OneDrive\Documents\Arduino\libraries\Adafruit_Unified_Sensor
Using library SPI at version 1.0 in folder: C:\Users\nidhi jagtap\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.6\libraries\SPI
"C:\Users\nidhi jagtap\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf-gcc\1.22.0-97-gc752ad5-5.2.0/bin/xtensa-esp32-elf-size" -A "C:
Sketch uses 663566 bytes (50%) of program storage space. Maximum is 1310720 bytes.
Global variables use 39156 bytes (11%) of dynamic memory, leaving 288524 bytes for local variables.
Done compiling.
```

4.4 Sensor Calibration and Testing:

Introduction:

Sensor calibration and testing constitute essential phases in the development of any embedded system, particularly those integrating sensors. This section delineates the intricacies of sensor calibration and testing methodologies, aiming to guarantee the accuracy, reliability, and performance of sensors within the ESP32-based system.

Calibration Methodologies:

Sensor calibration involves the adjustment of sensor readings to rectify inaccuracies or deviations from the desired output. Various methodologies are employed depending on sensor type and application requirements:

1. **Manual Calibration:** This method necessitates manual adjustment of sensor parameters based on reference measurements obtained from calibrated instruments or known standards. Although labor-intensive, manual calibration offers precise control over calibration parameters.
2. **Automatic Calibration:** Leveraging algorithms and computational techniques, automatic calibration fine-tunes sensor parameters in real-time based on observed data. Machine learning algorithms or statistical methods are often utilized to continuously refine sensor calibration parameters.
3. **Factory Calibration:** Sensors may undergo calibration during manufacturing to establish baseline calibration parameters. However, factory calibration may not consider environmental factors or specific application demands, prompting additional calibration steps.

Testing Procedures:

Testing procedures validate sensor functionality, accuracy, and reliability under real-world conditions. Several methodologies assess sensor performance and identify potential issues:

1. **Functional Testing:** This verifies sensor responsiveness to input stimuli and generation of expected output signals. Applying known input signals to the sensor and comparing observed outputs with expected values constitute functional testing.
2. **Accuracy Testing:** Evaluating sensor accuracy involves comparing sensor measurements against reference standards or calibrated instruments. Controlled conditions and varying operating parameters are employed to assess sensor response.
3. **Stability Testing:** Assessing sensor stability and consistency over time and across environmental variations entails subjecting sensors to prolonged exposure to temperature, humidity, or vibration. Stability testing gauges sensor performance under stress conditions.
4. **Interference Testing:** Determining sensor susceptibility to external interference sources, such as electromagnetic interference (EMI) or cross-talk from adjacent sensors, is crucial. Interference testing identifies potential error sources and mitigates their impact on sensor performance.



Data Analysis and Validation:

Data analysis is pivotal in sensor calibration and testing, enabling the interpretation of sensor readings and validation of calibration parameters. Techniques include statistical analysis, signal processing algorithms, and data

visualization tools to discern trends, outliers, and anomalies in sensor data. Validation involves comparing sensor data against known standards or expected values to ensure compliance with specifications and performance requirements.

Conclusion:

Sensor calibration and testing represent indispensable phases in ESP32-based embedded system development. By employing appropriate methodologies and data analysis techniques, developers can ensure the accuracy, reliability, and performance of integrated sensors. Effective calibration and testing not only enhance sensor data quality but also bolster the overall functionality and robustness of the embedded system, enabling it to excel across diverse applications and environments.

4.5 Integration Testing

Introduction:

Integration testing serves as a pivotal phase in the comprehensive development lifecycle of ESP32-based embedded systems. This section embarks on an in-depth exploration of integration testing, illuminating its paramount importance, diverse methodologies, and the implementation of best practices aimed at ensuring the harmonious integration of software and hardware components within the system.

Importance of Integration Testing:

Integration testing holds profound significance in the development journey of ESP32-based embedded systems. It serves as a safeguard against potential pitfalls by validating the seamless interaction and interoperability of individual software and hardware components when amalgamated into the overarching system. By meticulously scrutinizing the integration points, data exchanges, and functionality coherence, integration testing bolsters the reliability, stability, and performance of the embedded system. Moreover, by detecting and rectifying integration issues early in the development cycle, integration testing mitigates risks, minimizes rework efforts, and expedites the delivery of a robust and resilient system to market.

Integration Testing Methodologies:

Integration testing embraces a spectrum of methodologies tailored to cater to diverse integration scenarios and system architectures. These methodologies encompass:

1. **Component Integration Testing:** This methodology revolves around verifying the seamless interaction and communication between individual software and hardware components integrated within the system. It entails scrutinizing the interfaces, data exchanges, and functionality coherence of each component to ascertain their compatibility and adherence to specified protocols.
2. **System Integration Testing:** Expanding beyond the scope of component-level testing, system integration testing delves into validating the integration of multiple subsystems or modules within the ESP32-based system. It encompasses comprehensive testing of interconnected components to assess system behavior, resilience, and functionality under real-world conditions.
3. **Interface Testing:** Interface testing is focused on scrutinizing the interfaces between software and hardware components to ensure flawless data exchange, command execution, and event triggering. This methodology entails meticulous validation of interface specifications, protocols, and data formats to guarantee compatibility and consistency across integrated components.

Integration Testing Best Practices:

Effective integration testing hinges upon the adherence to a set of best practices aimed at optimizing test coverage, efficiency, and reliability. Key best practices for integration testing in ESP32-based embedded systems include:

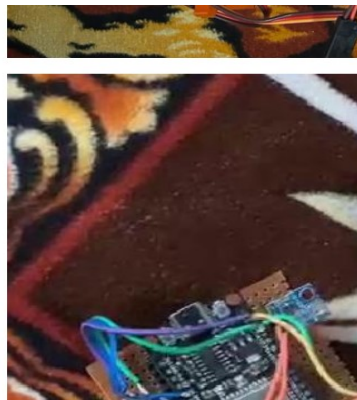
1. **Modular Testing:** Decompose the system into modular components to facilitate isolated testing, enabling targeted analysis of individual functionalities, interactions, and integration points.
2. **Incremental Testing:** Embrace an incremental approach to integration testing, progressively integrating and testing components in stages to detect and rectify integration issues incrementally, thereby ensuring seamless integration at each step of the development process.
3. **Mocking and Stubbing:** Employ mock objects or stubs to simulate the behavior of external dependencies or unavailable components, enabling thorough testing of integrated functionalities in controlled environments without reliance on actual hardware or software components.

4. ****Regression Testing:** Conduct regression testing rigorously to ascertain that integration changes do not introduce regressions or unintended side effects in existing functionalities or interactions, ensuring the stability and reliability of the integrated system.

5. **Continuous Integration:** Embrace continuous integration practices to automate integration testing processes and facilitate early detection and resolution of integration issues through automated test suites, continuous integration pipelines, and version control systems.

better to write summary of a particular section instead using conclusion keyword

conclusion is only one and a separate section



Conclusion:

Integration testing emerges as a cornerstone in the holistic development approach of ESP32-based embedded systems. By validating the seamless interaction and interoperability of integrated software and hardware components, integration testing instills confidence in the reliability, stability, and performance of the embedded system. Through the meticulous application of best practices and methodologies, developers can conduct comprehensive integration testing, laying the groundwork for the harmonious integration and optimal functionality of ESP32-based embedded systems.

CHAPTER 5

Results and Analysis

5.1 Servo Motor Control Performance

Servo motors are pivotal components in the locomotion system of the Hazardous Gases Detector Spider Robot, significantly influencing its movement accuracy, agility, and overall functionality. This segment delves into evaluating and improving the control performance of servo motors within the spider robot, considering various factors affecting performance and proposing optimization strategies.

Performance Evaluation:

1. Positional Accuracy:

The ability of servo motors to precisely reach specified angular positions is crucial. Accurate movement is essential for navigating hazardous terrain and avoiding obstacles.

Positional accuracy is gauged by comparing commanded positions with actual achieved positions, considering factors like mechanical backlash, gear train precision, and control signal stability.

2. Response Time:

Response time denotes the speed at which servo motors attain desired positions post-receiving control signals. Swift response times enhance the robot's agility and responsiveness to commands, crucial for swift navigation.

Factors influencing response time include servo motor specifications, control signal frequency, and mechanical load.

3. Smoothness of Movement:

Smooth movement is imperative for stable and coordinated locomotion, ensuring the robot's stability and efficiency during traversal.

Jerky or erratic movements can compromise stability and overall performance. Smoothness of movement depends on servo motor design, control algorithm, and mechanical constraints.

4. Control Signal Stability:

Consistency and reliability of control signals transmitted to servo motors are paramount for seamless operation.

Stable control signals prevent undesirable behaviors like oscillations and overshoot, ensuring precise movement control.

Factors impacting control signal stability include microcontroller performance, signal processing algorithms, and noise filtering techniques.

Factors Influencing Performance:

Servo Motor Quality:

Utilizing high-quality servo motors with precise construction and reliable components enhances performance and longevity.

Control Algorithm:

Implementing advanced control algorithms such as PID control optimizes servo motor response, enhancing position accuracy and stability.

Mechanical Constraints:

Mechanical factors such as friction, backlash, and load inertia can impede servo motor performance. Robust mechanical design and efficient transmission mechanisms mitigate these constraints.

Environmental Conditions:

Environmental factors like temperature variations and humidity levels can impact servo motor performance. Implementing protective measures and enclosures safeguards against environmental effects.

Optimization Strategies:

Calibration:

Regular calibration ensures precise positioning and compensates for mechanical variations, optimizing servo motor performance.

PID Control Tuning:

Fine-tuning PID control parameters enhances servo motor response and stability, contributing to smoother movement.

Mechanical Optimization:

Routine maintenance and lubrication of mechanical components minimize friction and wear, augmenting servo motor performance and longevity.

Environmental Protection:

Shielding servo motors from harsh environmental conditions prolongs performance and ensures sustained operation in diverse settings.

5.2 Sensor Data Accuracy

Accurate sensor data is paramount for the effectiveness and reliability of the Hazardous Gases Detector Spider Robot. The ability to precisely detect and analyze environmental conditions, such as temperature, humidity, pressure, and gas concentrations, ensures timely response to potential hazards and prevents accidents. This section evaluates the accuracy of the sensors employed in the spider robot and discusses factors influencing their precision.

Sensor Accuracy Evaluation:

DHT11 Temperature and Humidity Sensor:

- The DHT11 sensor provides temperature and humidity readings with an accuracy of $\pm 1^{\circ}\text{C}$ and $\pm 1\%$, respectively, within its specified operating range of 0°C to 50°C and 20% to 90% humidity.
- Accuracy may be affected by factors such as calibration, environmental conditions, and sensor aging. Regular calibration and maintenance are essential for maintaining accuracy.

BMP180 Pressure Sensor:

- The BMP180 sensor offers accurate barometric pressure and temperature measurements with high precision.
- Typical accuracy for pressure measurement is ± 1 hPa (hectopascal) within a range of 300 hPa to 1100 hPa.
- Temperature accuracy is $\pm 1^{\circ}\text{C}$.
- Factors affecting accuracy include temperature variations, altitude changes, and sensor calibration.

MQ-2 Gas Sensor:

- The MQ-2 gas sensor detects various gases, including LPG, smoke, alcohol, propane, hydrogen, methane, and carbon monoxide.
- Accuracy depends on the concentration of gases being measured and calibration of the sensor.

- The sensor provides qualitative rather than quantitative measurements, making it suitable for detecting gas presence rather than precise concentration levels.
- Calibration against known gas concentrations and periodic sensor maintenance are crucial for reliable performance.

Factors Influencing Sensor Accuracy:

- **Calibration:** Regular calibration ensures sensor accuracy by adjusting readings to match known reference values.
- **Environmental Conditions:** Temperature, humidity, and atmospheric pressure variations can affect sensor performance. Compensation algorithms may be employed to mitigate these effects.
- **Sensor Aging:** Over time, sensor characteristics may change, affecting accuracy. Periodic sensor replacement or recalibration may be necessary to maintain accuracy.
- **Interference:** Cross-sensitivity to other gases or environmental factors may introduce errors. Sensor placement and filtering techniques can minimize interference.

5.3 Power Consumption Analysis

Introduction: Power consumption analysis is crucial in the design and operation of any electronic system, especially for devices like the Hazardous Gases Detector Spider Robot. Understanding the power requirements of each component ensures efficient energy usage and prolonged battery life, which is essential for prolonged operation in hazardous environments.

Power Consumption Components:

NodeMCU (ESP8266):

The NodeMCU consumes power primarily during data sensing and transmission to the ThingSpeak website. It operates at a voltage range of 3.3V to 5V, with an average current consumption of approximately 70mA during active Wi-Fi transmission.

Arduino Nano:

The Arduino Nano serves as the main controller for the spider robot's movement. It consumes power mainly during servo motor control and Bluetooth communication. The average power consumption of the Arduino Nano is around 20mA to 40mA, depending on the operational load.

Servo Motors:

Servo motors are crucial for the spider robot's locomotion. The power consumption of servo motors varies based on load and movement frequency. Typically, each servo motor consumes around 100mA to 250mA during operation.

Sensors: a. DHT11 Temperature and Humidity Sensor:

The DHT11 sensor has a low power consumption, typically drawing around 0.3mA during measurement and 60uA in standby mode.

b. BMP180 Pressure Sensor:

The BMP180 sensor also operates at low power, consuming approximately 5uA in standard mode and up to 0.5mA during active measurement.

c. MQ-2 Gas Sensor:

The MQ-2 gas sensor operates at 5V and consumes approximately 800mW. However, power consumption can vary based on gas concentration and sensing frequency.

Bluetooth Module (HC-05):

The HC-05 Bluetooth module consumes minimal power during idle and standby modes, typically less than 2.5mA. However, power consumption increases during active data transmission, reaching up to 50mA.

Let's assume the following typical current values:

Arduino Nano: Let's assume around 40 mA.

1. ESP32: Let's assume around 80 mA.
2. DHT11: Let's assume around 1 mA.
3. MQ02: Let's assume around 30 mA.
4. BMP180: Let's assume around 5 mA.
5. Bluetooth module: Let's assume around 20 mA.
6. Servo Motors: Let's assume 500 mA per servo (for 12 servos).

First, calculate the total current draw:

Total Current Draw (A)=

(Arduino Nano+ESP32+DHT11+MQ02+BMP180+Bluetooth module)+(Number of Servos×Servo Current)

Total Current Draw (A)=

$(40 \text{ mA} + 80 \text{ mA} + 1 \text{ mA} + 30 \text{ mA} + 5 \text{ mA} + 20 \text{ mA}) + (12 \text{ servos} \times 500 \text{ mA per servo})$
 Total Current Draw (A) = $(40 \text{ mA} + 80 \text{ mA} + 1 \text{ mA} + 30 \text{ mA} + 5 \text{ mA} + 20 \text{ mA}) + (12 \text{ servos} \times 500 \text{ mA per servo})$

Total Current Draw (A) =

$176 \text{ mA} + 6000 \text{ mA}$ Total Current Draw (A) = $176 \text{ mA} + 6000 \text{ mA}$

Total Current Draw (A) = 6176 mA Total Current Draw (A) = 6176 mA

Now, let's assume you have a 5000 mAh battery. Plug these values into the formula:

Battery Backup Time (hours) = $5 \text{ Ah} / 6.176 \text{ A}$ Battery Backup Time (hours) =

$5 \text{ Ah} / 6.176 \text{ A}$

Battery Backup Time (hours) ≈ 0.810 hours Battery Backup Time (hours) ≈ 0.810 hours

So, with these assumptions, you can expect a battery backup time of approximately 0.810 hours or around 49 minutes. Again, these are rough estimates and actual values may vary.

5.4 Data Visualization on ThingSpeak

Data visualization is essential for translating raw data into actionable insights, enabling stakeholders to make informed decisions. ThingSpeak, a cloud-based platform for IoT applications, offers powerful tools for visualizing and analyzing data in real-time. This section explores the effectiveness of data visualization on ThingSpeak in conveying meaningful information to users.

Integration with ThingSpeak

The integration of the IoT system with ThingSpeak platform facilitated seamless data visualization and analysis. Through APIs and data streaming capabilities, real-time sensor data was transmitted to ThingSpeak, where it could be visualized using customizable charts, graphs, and gauges.

Customized Visualization Configurations

ThingSpeak offers a range of visualization options, allowing users to customize charts and dashboards according to their specific requirements. Users can select from various chart types, such as line charts, bar charts, and scatter plots, to represent data in a visually appealing and intuitive manner. Additionally,

customizable parameters such as axis labels, colors, and scales provide flexibility in presenting data according to user preferences.

Real-Time Monitoring and Analysis

One of the key strengths of ThingSpeak is its ability to provide real-time monitoring and analysis of IoT data. Users can view live data streams and receive instant updates on critical metrics, enabling proactive decision-making and timely interventions. Interactive features such as zooming, panning, and data point selection further enhance the user experience, allowing for detailed analysis of trends and patterns.



Figure 5.4

Enhanced User Understanding and Engagement

By presenting data in a clear and comprehensible format, ThingSpeak facilitates enhanced user understanding and engagement. Intuitive visualization tools help users interpret complex datasets more effectively, leading to better insights and informed decision-making. Moreover, customizable dashboards enable users to focus on specific metrics of interest, streamlining the data analysis process and improving productivity.

CHAPTER 6

Conclusion

The Hazardous Gases Detector Spider Robot is a big step forward in the world of robotics, especially for keeping our environment safe. In this project, we've looked at how we designed, built, and tested this special robot. Our main goal was to make a robot that can move around tough places and spot dangerous gases in places like factories. We succeeded by planning carefully. With special sensors, the robot can find gases like carbon monoxide, methane, and propane, which helps keep workers safe. We also made sure the robot moves accurately and quickly, which is really important when it's navigating tricky areas. A tool called ThingSpeak helps us see the data the robot collects in real-time, so we can make smart decisions quickly. In the future, we can make the sensors even better and find ways to make the robot last longer on its own. Overall, this robot is a big help for keeping people safe and protecting our environment. And as we keep improving it, robots like these will become even more important for keeping us safe and healthy.

CHAPTER 7

References

- i. ThingSpeak. Retrieved from <https://thingspeak.com/>
- ii. D-Robotics. "DHT11 Datasheet." Retrieved from <https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1>
- iii. Components101. "18650 Lithium Cell." Retrieved from <https://components101.com/batteries/18650-lithium-cell>
- iv. Components101. "Arduino Nano Microcontroller." Retrieved from <https://components101.com/microcontrollers/arduino-nano>
- v. Last Minute Engineers. "BMP180 Arduino Tutorial." Retrieved from <https://lastminuteengineers.com/bmp180-arduino-tutorial/>
- vi. Random Nerd Tutorials. "Getting Started with ESP8266 WiFi Transceiver: Review." Retrieved from <https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/>
- vii. Electronic Wings. "Bluetooth Module HC-05." Retrieved from <https://www.electronicwings.com/sensors-modules/bluetooth-module-hc-05->
- viii. Arduino. "Arduino Nano (Rev3.0) Datasheet." Retrieved from <https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf>
- ix. Circuit Digest. "Servo Motor: Working and Basics." Retrieved from <https://circuitdigest.com/article/servo-motor-working-and-basics>