Project 1 : IoT based Smart Factory System

ABSTRACT:-

The use of Arduino programming in smart industries has revolutionized the way industrial processes are managed and optimized. This project report aims to explore the application of Arduino programming in creating smart solutions for various industrial settings. By leveraging the capabilities of Arduino microcontrollers and their compatibility with a wide range of sensors and actuators, industries can achieve enhanced automation, efficiency, and safety.

The project focuses on developing a comprehensive understanding of Arduino programming principles, including sensor integration, data acquisition, decision-making algorithms, and actuator control. It also highlights the potential benefits of implementing Arduino-based solutions in industrial environments, such as improved productivity, reduced downtime, and real-time monitoring.

To demonstrate the practical application of Arduino programming, a prototype system is developed for a specific industrial process. The system utilizes Arduino boards, along with relevant sensors and actuators, to collect data, analyze it, and control the process based on predefined conditions. The programming code is implemented to handle sensor readings, process data, and trigger appropriate actions, providing an efficient and reliable solution.

Throughout the project, various challenges related to hardware integration, programming complexities, and system scalability are addressed. By employing appropriate coding practices, modular design principles, and efficient algorithms, the project aims to create a robust and adaptable Arduino-based solution that can be extended to different industrial scenarios.

The results of the project demonstrate the effectiveness of Arduino programming in smart industries, showcasing its ability to streamline processes, minimize manual intervention, and enhance overall productivity. The project report provides valuable insights into the practical implementation of Arduino-based systems in industrial settings, serving as a guide for organizations seeking to leverage this technology for their specific requirements.

Overall, this project report serves as a comprehensive resource for understanding and implementing Arduino programming in smart industries, emphasizing the importance of integrating hardware and software solutions to achieve enhanced automation and efficiency. It highlights the immense potential of Arduino programming to transform traditional industries into smart, connected, and optimized systems.

Introduction :-

The rapid advancements in technology have paved the way for the development of smart industries, where automation, real-time monitoring, and data-driven decision-making are crucial for achieving optimal efficiency and productivity. Arduino programming, with its versatility and ease of use, has emerged as a powerful tool for creating intelligent solutions in industrial settings. By combining Arduino microcontrollers with a range of sensors and actuators, industries can leverage the capabilities of this platform to enhance their operational processes.

The primary objective of this project report is to explore the application of Arduino programming in the context of smart industries. It aims to provide a comprehensive understanding of how Arduino-based systems can be developed and implemented to automate industrial processes, monitor critical parameters, and control actuators based on predefined conditions. By utilizing Arduino's programming capabilities, industries can achieve increased efficiency, reduced downtime, and improved safety.

The report will delve into the fundamental principles of Arduino programming, covering topics such as sensor integration, data acquisition, decision-making algorithms, and actuator control. It will highlight the benefits of using Arduino microcontrollers, such as their low-cost, compatibility with various sensors, and ease of prototyping. Moreover, it will emphasize the importance of integrating hardware and software components seamlessly to create robust and reliable smart industrial systems.

A prototype system will be developed as part of this project to demonstrate the practical application of Arduino programming. The system will incorporate Arduino boards, along with relevant sensors and actuators, to collect real-time data, process it using intelligent algorithms, and control the industrial process accordingly. The programming code will be designed to handle sensor readings, analyze data, and trigger appropriate actions, thereby showcasing the efficiency and effectiveness of Arduino programming in a real-world scenario.

Throughout the project, challenges related to hardware integration, scalability, and programming complexities will be addressed. Strategies for overcoming these challenges will be discussed, along with best practices for designing modular and scalable Arduino-based systems. By adhering to these practices, industries can develop flexible and adaptable solutions that can be tailored to suit their specific requirements.

The project report will serve as a valuable resource for organizations seeking to implement Arduino programming in their industrial processes. It will provide insights into the practical implementation of Arduino-based systems, along with guidelines for hardware selection, software development, and system integration. By embracing Arduino programming, industries can unlock the potential for enhanced automation, improved efficiency, and optimized performance in the era of smart industries.

Significance of using tickercad:-

The implementation of an IoT-based Smart Factory System using Tinkercad software holds several significances in the context of industrial automation and efficiency. Some of the key significances are as follows:

1. **Improved Operational Efficiency**: The integration of IoT technology enables real-time monitoring and control of various factory processes. This leads to improved operational efficiency by optimizing resource utilization, minimizing downtime, and enhancing overall productivity.

2. **Enhanced Safety and Security**: The IoT system can incorporate sensors and devices for monitoring critical parameters such as temperature, humidity, gas leaks, and intrusions. It enables proactive identification of safety risks, alerts for potential hazards, and quick response mechanisms, ensuring a safer working environment for factory personnel.

3. **Remote Monitoring and Control**: With IoT connectivity, factory managers and supervisors can remotely monitor the factory's performance, access real-time data, and make informed decisions. This eliminates the need for constant physical presence on-site, allowing for better management and improved response times.

4. **Predictive Maintenance**: IoT-enabled sensors can collect data on equipment health, performance, and usage patterns. By leveraging machine learning algorithms, predictive maintenance can be implemented to identify potential equipment failures in advance, schedule maintenance activities, and minimize unplanned downtime, thereby optimizing maintenance costs.

5. **Energy Efficiency**: The Smart Factory System can include energy monitoring and control mechanisms, allowing for the optimization of energy consumption. IoT sensors can track energy usage patterns, identify areas of high consumption, and enable the implementation of energy-saving strategies, leading to reduced energy costs and a more sustainable operation.

6. **Data Analytics and Insights**: The IoT system generates vast amounts of data from connected devices and sensors. Through advanced data analytics techniques, valuable insights can be extracted, providing actionable intelligence for process optimization,

resource allocation, and decision-making. These insights help drive continuous improvement and innovation within the factory.

7. **Flexibility and Scalability**: Tinkercad software allows for the simulation and virtual prototyping of the Smart Factory System, enabling the design and testing of various components and functionalities. This ensures flexibility in system development and scalability for future expansion or modifications without disrupting the physical factory setup.

8. **Cost-Effectiveness**: Tinkercad software provides a cost-effective platform for developing and testing the Smart Factory System. It eliminates the need for physical hardware during the initial stages, reducing the development costs associated with prototyping and testing.

In conclusion, the implementation of an IoT-based Smart Factory System using Tinkercad software offers significant benefits in terms of improved operational efficiency, safety, remote monitoring, predictive maintenance, energy efficiency, data analytics, flexibility, scalability, and cost-effectiveness. It enables factories to embrace digital transformation and leverage IoT technologies to drive productivity, profitability, and sustainable growth.

Methodology

The methodology section of this project report describes the approach and techniques used to develop an Arduino-based smart industrial system. The system incorporates various sensors, actuators, and components including a PIR sensor, fan, lights, gas sensor, buzzer, ultrasonic sensor, servo motor, temperature sensor, relay, and switch. The following methodology outlines the step-by-step process involved in designing and implementing the system:

Requirement Analysis: The initial phase of the methodology involved analyzing the requirements of the smart industrial system. This included identifying the industrial process to be automated and determining the specific functionalities required for sensor monitoring and actuator control. The components mentioned, such as the PIR sensor, gas sensor, temperature sensor, and ultrasonic sensor, were selected based on the specific parameters to be measured.

Hardware Selection: After defining the requirements, appropriate Arduino boards were selected based on the project's needs. Factors such as the number of I/O pins, processing power, and memory capacity were considered. Additionally, the compatibility of the Arduino boards with the selected sensors, actuators, and components, including the PIR sensor, gas sensor, servo motor, relay, and switch, was ensured.

Sensor Integration: The selected sensors, such as the PIR sensor, gas sensor, temperature sensor, and ultrasonic sensor, were integrated with the Arduino boards. This involved connecting the sensors to the appropriate input pins of the Arduino and ensuring compatibility between the sensor interfaces and Arduino's digital or analog input capabilities. Necessary libraries or code were implemented to enable sensor data acquisition.

Actuator Control: To control the industrial process based on sensor readings, actuators such as the fan, lights, buzzer, servo motor, relay, and switch were integrated with the Arduino boards. The Arduino programming code was written to control these actuators using appropriate digital or analog output pins. For instance, the fan and lights could be controlled using digital output pins, while the servo motor and relay may require PWM (Pulse Width Modulation) signals or specific control protocols.

Arduino Programming: The Arduino programming language, a variant of C/C++, was used to write the code for the system. The code implemented algorithms to read data from the sensors, process the sensor readings, and trigger actions for the actuators based on predefined conditions. The programming code included functions to handle sensor data acquisition, decision-making logic, actuator control, and communication with other components.

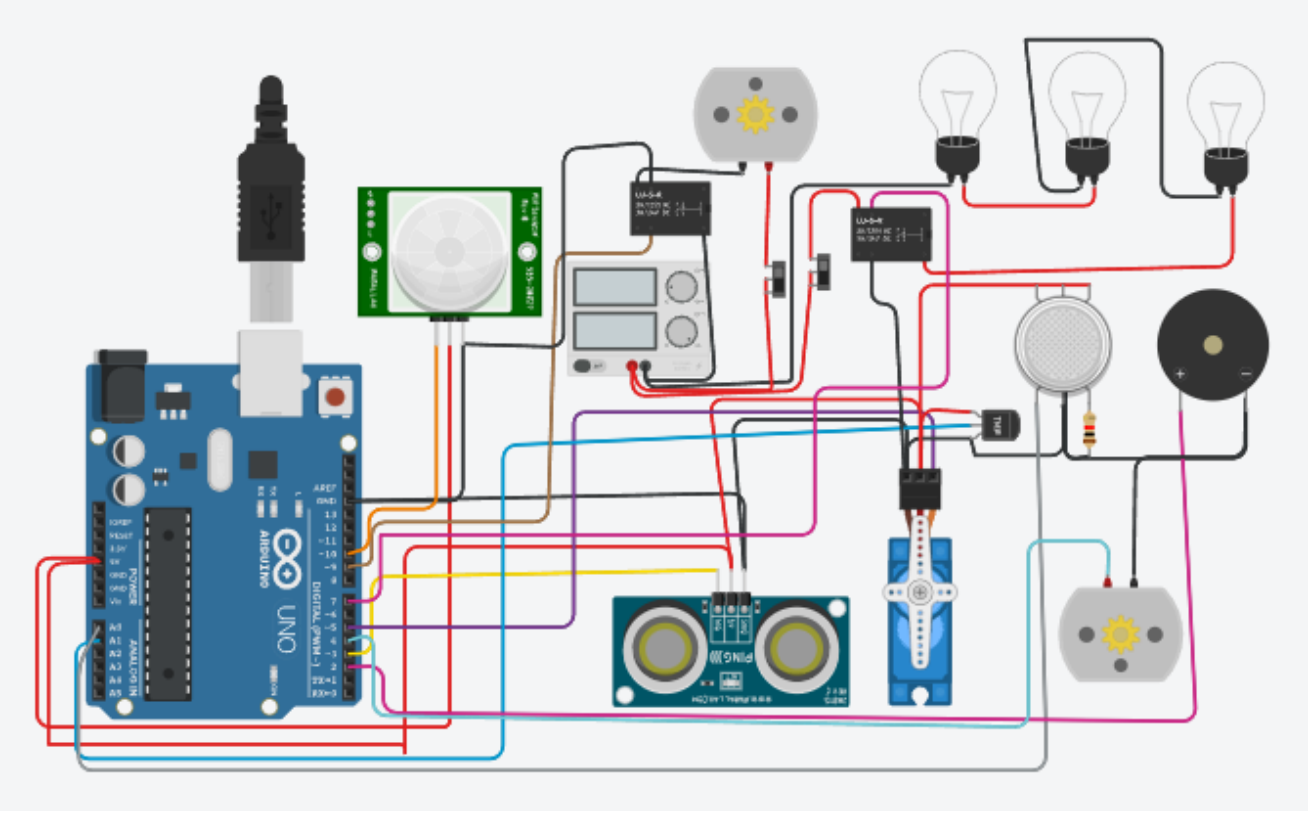
Testing and Debugging: Rigorous testing and debugging were conducted to verify the functionality of the system. The system was tested under various scenarios to ensure accurate sensor readings, reliable decision-making algorithms, and proper actuator control. Testing involved simulating different sensor inputs, verifying the correct operation of the actuators, and debugging any issues encountered during the testing process.

System Integration: Once the individual components were developed and validated, they were integrated into a cohesive system. This involved connecting the Arduino boards, sensors, actuators, and other components in a logical and organized manner. Proper wiring techniques and connectors were used to establish the necessary connections between the components.

Prototype Deployment: The final phase of the project involved deploying the prototype system in the target industrial environment. The system was installed, and its performance was monitored and evaluated in real-world conditions. Feedback from industrial personnel and stakeholders was collected to assess the system's effectiveness and identify any areas for further improvement.

The methodology described above provides a systematic approach to developing an Arduino-based smart industrial system. By following this methodology, the integration and programming of various sensors, actuators, and components, including the PIR sensor, fan, lights, gas sensor, buzzer, ultrasonic sensor, servo motor, temperature sensor, relay, and switch, can be achieved.

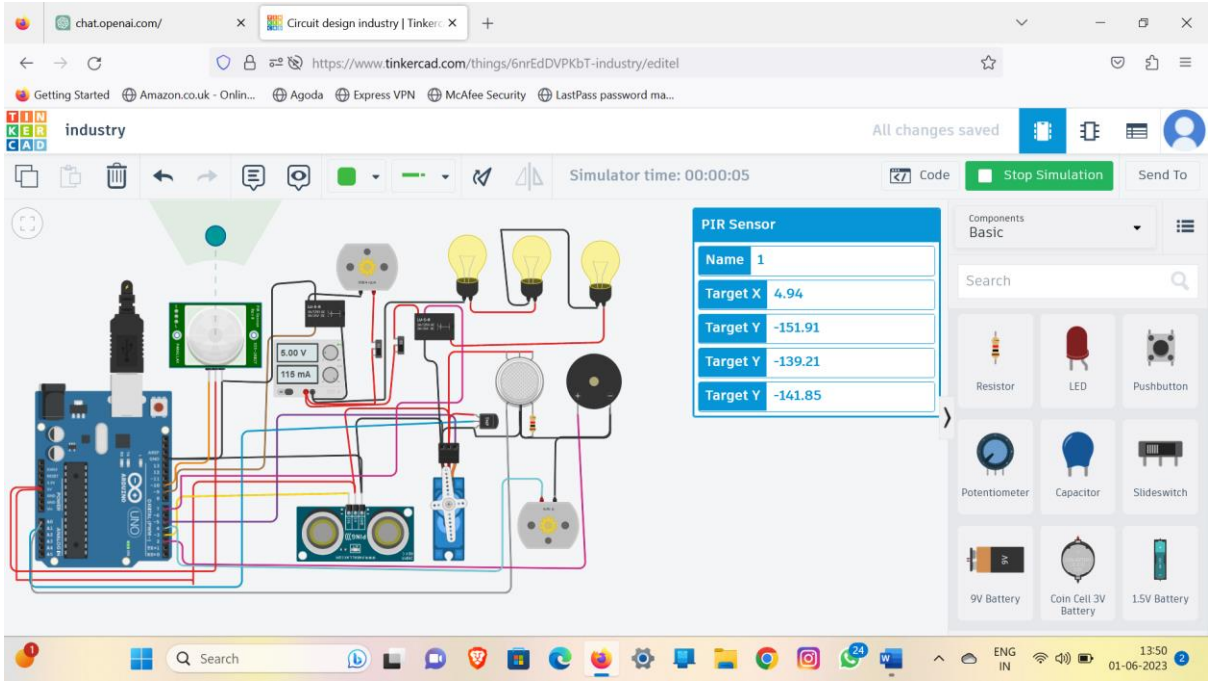
Circuit diagram:-



WORKING:-

PIR SENSOR:-

The working principle of a PIR (Passive Infrared) sensor involves detecting infrared radiation emitted by objects within its field of view. When an object enters a restricted area, the PIR sensor detects its presence based on the change in the infrared radiation pattern.



To implement a system using an Arduino and a PIR (Passive Infrared) sensor that turns on emergency lights when a person enters, you can follow these steps:

1. Connect the PIR sensor to your Arduino. The PIR sensor typically has three pins: VCC, GND, and OUT. Connect the VCC pin to a 5V pin on the Arduino, GND pin to GND, and the OUT pin to a digital input pin (e.g., pin 2).
2. Connect the emergency lights to a suitable power source and relay module. The relay module should have a control input pin that can be connected to a digital output pin on the Arduino (e.g., pin 3). Make sure to properly wire the relay module following its specifications and guidelines.
3. Write the Arduino code to detect motion and control the lights. Here's a basic example:

   Code:-

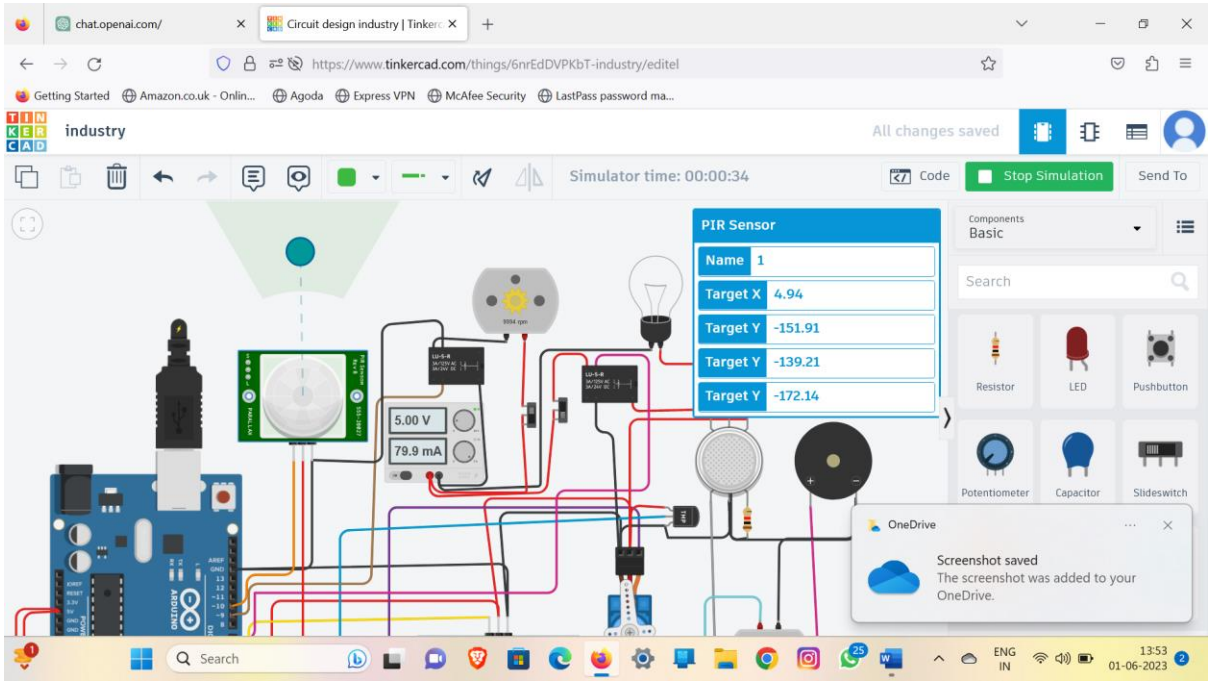
```
int pirPin = 2;       // PIR sensor output pin

int relayPin = 3;     // Relay control pin

 void setup() {

 pinMode(pirPin, INPUT);

pinMode(relayPin, OUTPUT);

 }

 void loop() {

int pirState = digitalRead(pirPin);  // Read PIR sensor state

 if (pirState == HIGH) {             // Motion detected

digitalWrite(relayPin, HIGH);      // Turn on the relay

} else {

digitalWrite(relayPin, LOW);        // Turn off the relay

}

}
```

4. Upload the code to your Arduino board using the Arduino IDE or your preferred programming environment.
5. Ensure that the emergency lights are connected to the relay module and that the relay module is properly wired to control the power supply to the lights.
6. Place the PIR sensor in a suitable location to detect motion as desired. Adjust any sensitivity or delay settings on the PIR sensor if necessary, following the sensor's documentation.

7. When a person enters the detection range of the PIR sensor, it will detect the motion and trigger the Arduino to activate the relay module. The relay module, in turn, will switch on the emergency lights.

Note: This is a basic example to get you started. You may need to modify the code and circuit based on your specific requirements, such as adding additional functionality, adjusting timing, or integrating other components. Always refer to the documentation and specifications of the components you are using for proper wiring and operation.

"Automated Room Fan Control System using PIR Sensor":-



Introduction:

- Brief overview of the project's objective: implementing an automatic fan control system using Arduino and PIR sensor.
- Explanation of the benefits of automating fan operation based on human presence, such as energy efficiency and convenience.

1. PIR Sensor:

- Definition and working principle of the Passive Infrared (PIR) sensor.
- Explanation of how PIR sensors detect infrared radiation emitted by objects, including human bodies.
- Description of the PIR sensor's components: pyroelectric sensor, Fresnel lens, and signal processing circuitry.
- Discussion of the sensor's field of view, range, and considerations for optimal placement.

2. Arduino Board:

- Introduction to Arduino microcontroller board and its suitability for sensor integration and actuator control.
- Explanation of the digital input/output (I/O) pins used to interface with the PIR sensor and control the fan.
- Overview of the Arduino programming language and its capabilities for handling sensor inputs and actuator outputs.

3. Hardware Setup:

- Detailed description of the hardware components required: Arduino board, PIR sensor, relay module, and fan.
- Circuit diagram illustrating the connections between the components, including power supply connections.
- Instructions for wiring the PIR sensor output to the Arduino and connecting the fan via the relay module.

4. Software Implementation:

- Overview of the Arduino code structure for the automatic fan control system.
- Explanation of the code segments responsible for reading PIR sensor input and controlling the fan.
- Discussion of the programming logic for detecting human presence and activating the fan.

5. Operation Sequence:

- Step-by-step explanation of the system's operation when a person enters the room: a. PIR sensor detects motion and triggers an interrupt on the Arduino. b. Arduino code reads the sensor input and determines the presence of a person. c. If a person is detected, the Arduino activates the relay module. d. The relay module switches on the fan, allowing for air circulation in the room.

6. System Customization and Enhancements:

- Discussion of potential modifications and improvements to the system, such as adjusting sensitivity, incorporating timers, or integrating additional sensors for advanced functionality.
- Explanation of how the Arduino code can be customized to accommodate specific requirements or expand the system's capabilities.

Conclusion:

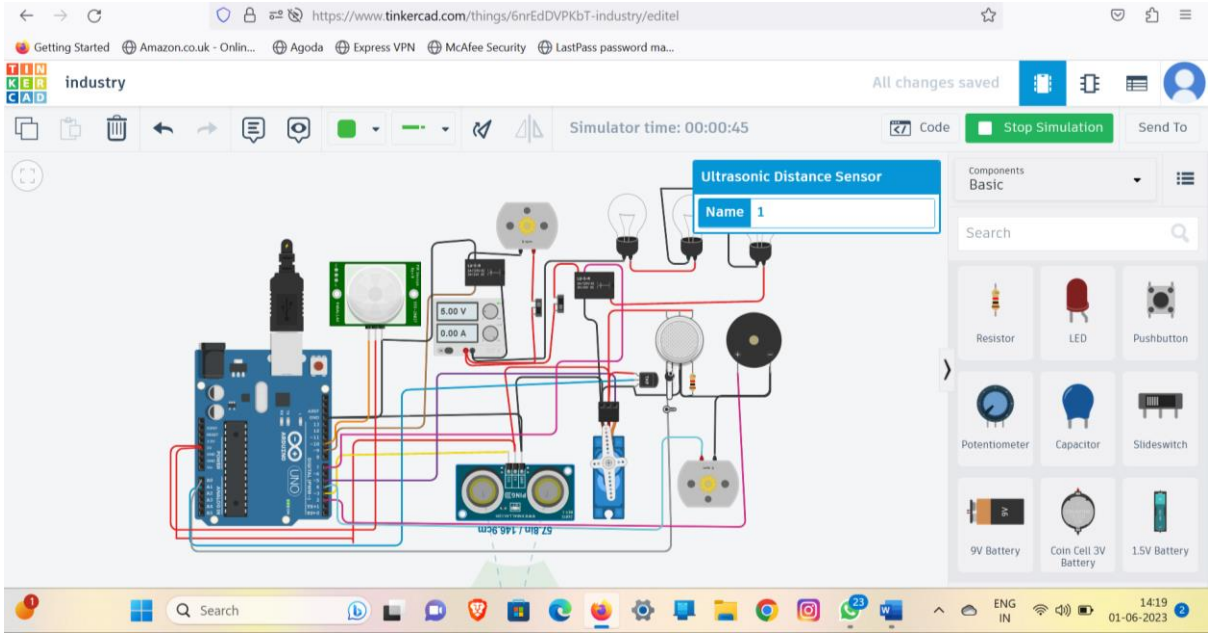
- Recap of the project's objectives and the successful implementation of the automatic fan control system using Arduino and PIR sensor.
- Overview of the advantages of automated fan control based on human presence, including energy savings and improved comfort.
- Final remarks on the potential applications and future developments of the system, highlighting its relevance in smart homes, offices, and other environments.

References:

- Citations and acknowledgments for any external resources or references used to understand the theory and principles behind PIR sensors, Arduino programming, and automatic fan control systems.

Note: Adapt the headings and content as needed to match the specific details of your project and ensure a comprehensive report.

ULTRASONIC SENSOR WORKING:-



The working principle of an ultrasonic sensor and a servo motor can be combined to automatically open a door when a person approaches it. Here's how the system can work:

1. **Sensor Setup**: The ultrasonic sensor is installed near the entrance of the room, facing outward. It should be positioned to have a clear line of sight to detect any approaching object or person. The servo motor is connected to the door mechanism in such a way that it can open and close the door.
2. **Initialization**: The Arduino board is connected to both the ultrasonic sensor and the servo motor. The necessary digital input/output pins for the ultrasonic sensor and the servo motor are defined in the Arduino programming code.
3. **Powering the Components**: The ultrasonic sensor and the servo motor are powered using the appropriate voltage supply (usually 5V or 3.3V) through the Arduino board or an external power supply, depending on their power requirements.
4. **Detecting Proximity**: The ultrasonic sensor emits ultrasonic waves and measures the time it takes for the waves to bounce back after hitting an object. By calculating the time taken, the distance between the sensor and the object can be determined. When a person approaches the room entrance, the ultrasonic sensor detects the change in distance.
5. **Arduino Code Execution**: The Arduino programming code continuously reads the distance measurement from the ultrasonic sensor. It checks if the distance is below a predefined threshold, indicating that a person is in close proximity to the entrance.
6. **Door Control**: If the distance reading falls below the threshold, indicating the presence of a person, the Arduino code triggers the servo motor to open the door. The servo motor rotates to the predetermined angle that allows the door to open.
7. **Delay and Closing**: The Arduino code can include a delay to keep the door open for a specific duration, allowing the person to enter. After the delay period, the servo motor is instructed to rotate back to its initial position, closing the door.
8. **Looping and Resetting**: The Arduino code typically runs in a loop, continuously monitoring the distance from the ultrasonic sensor. If the distance remains above the threshold, indicating no presence of a person, the Arduino code can maintain the system in an idle state or perform any necessary reset operations.

By combining the ultrasonic sensor's distance detection capability with the control of a servo motor, the system can automatically open the door when a person approaches the room entrance. The Arduino programming code plays a crucial role in coordinating the sensor readings and servo motor control to achieve the desired functionality.

Usage and working of relay:-

In Arduino, a relay is an electromechanical device used to control high-power circuits using low-power signals. It allows you to switch ON/OFF or control devices that require higher voltages or currents than what the Arduino can directly handle. Relays are commonly used in home automation, robotics, industrial control systems, and more.

A relay consists of two main parts: the coil and the contacts. The coil is an electromagnetic winding that, when energized, creates a magnetic field. The contacts are the switch-like terminals that open or close the circuit when the coil is energized or de-energized.

When an Arduino outputs a logic signal (typically 5V or 3.3V), it can be used to control the relay by connecting the signal pin of the relay module to a digital pin of the Arduino. The relay module is a separate circuit board that includes the relay and other necessary components.

Typically, a relay module has three main pins that are connected to the Arduino:

1. VCC (+5V or +3.3V): Connect this pin to the Arduino's power supply voltage (5V or 3.3V) to power the relay module.
2. GND (Ground): Connect this pin to the Arduino's ground (GND) to complete the circuit.
3. IN (Input Control): Connect this pin to a digital pin of the Arduino to control the relay. When the Arduino pin is set to HIGH, the relay coil is energized, and the contacts switch position (e.g., from Normally Open to Normally Closed, or vice versa). When the Arduino pin is set to LOW, the coil is de-energized, and the contacts return to their default position.

Additionally, there are often two or more output pins on the relay module:

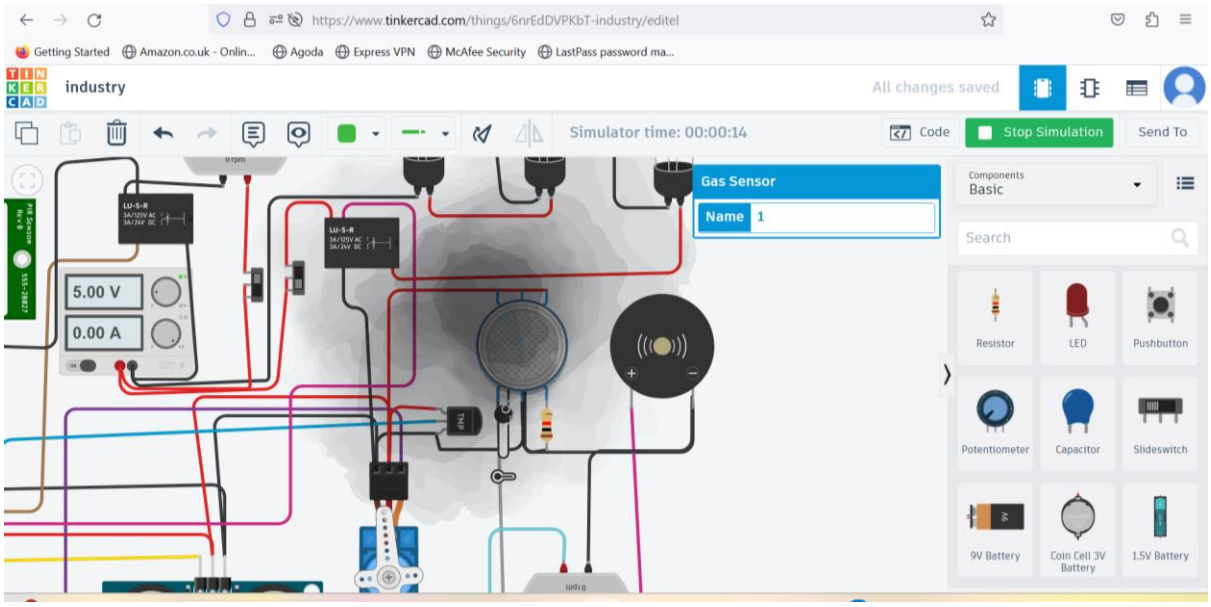
1. NO (Normally Open): This pin is connected to the common (COM) pin when the relay is not energized. When the relay is energized, this pin disconnects from the common pin.
2. NC (Normally Closed): This pin is connected to the common (COM) pin when the relay is not energized. When the relay is energized, this pin disconnects from the common pin.
3. COM (Common): This pin is the common connection point for the relay's contacts. It is connected to either the NO or NC pin, depending on the state of the relay.

By connecting external devices or circuits to the NO and COM pins, you can control their ON/OFF state using the relay. The NC and COM pins can also be used in different applications.

It's important to note that when working with relays and high-power circuits, you may need to consider additional factors such as power requirements, current ratings, flyback diodes, and isolation to ensure safe and reliable operation.

Remember to consult the datasheet or documentation provided with your specific relay module to understand its pin configuration and any additional features it may have.

GAS SENSOR:-


https://www.tinkercad.com/things/6nrEdDVPKbT-industry/editel
Getting Started
Amazon.co.uk - Onlin...
Agoda
Express VPN
McAfee Security
LastPass password ma...
industry
All changes saved
Simulator time: 00:00:14
Code
Stop Simulation
Send To
Gas Sensor
Name
1
Components
Basic
Search
5.00 V
0.00 A
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery

The working principle of a gas sensor can be utilized to detect gas leaks in an industrial setting and trigger the operation of a fan component for ventilation and safety purposes. Here's an overview of how the system can work:

1. **Sensor Setup**: The gas sensor is installed in the area where gas leaks are likely to occur, such as near gas pipelines or storage tanks. The sensor is positioned to ensure it can detect the presence of gas effectively. The gas sensor may have specific requirements for installation, such as proper calibration and positioning, which should be followed as per the manufacturer's guidelines.

2. **Initialization**: The Arduino board is connected to the gas sensor, and the necessary analog or digital input pins are defined in the Arduino programming code for interfacing with the sensor.

3. **Powering the Sensor**: The gas sensor is powered using the appropriate voltage supply (usually 5V) through the Arduino board or an external power supply. It is important to provide a stable power source to ensure accurate readings from the gas sensor.

4. **Gas Detection**: The gas sensor continuously monitors the surrounding air for the presence of specific gases. When a gas leak occurs, the sensor detects the gas molecules in the air and generates a corresponding electrical signal.

5. **Signal Processing**: The Arduino programming code reads the analog or digital signal from the gas sensor to determine the gas concentration or the presence of gas. The gas sensor may provide analog or digital outputs, depending on the specific model. The code processes the signal to convert it into meaningful gas concentration values or a binary indication of gas presence.

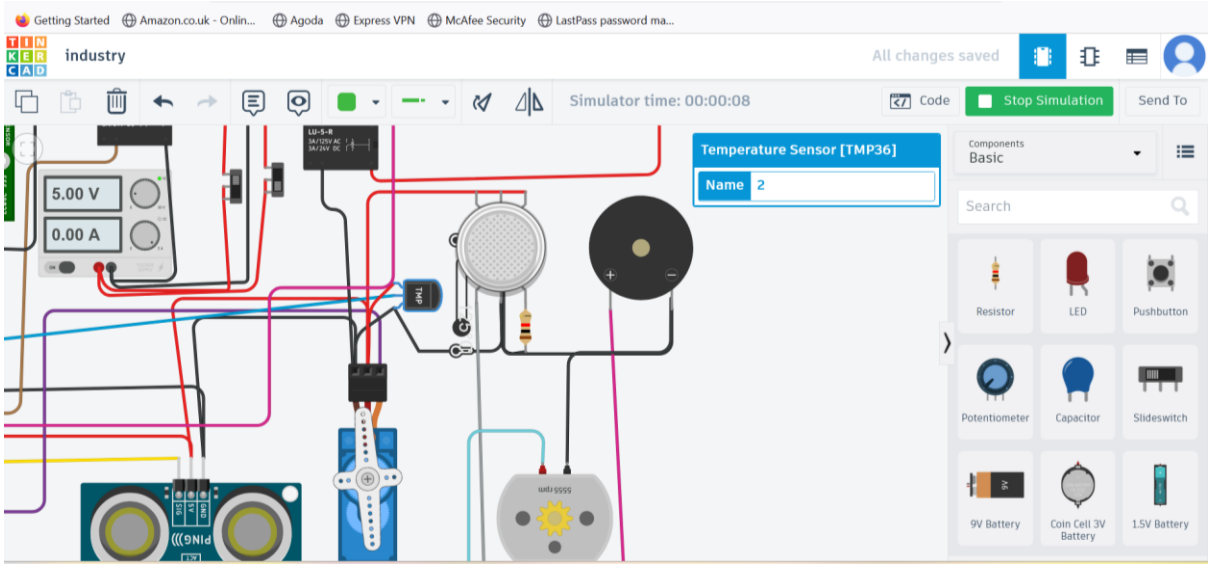
6. **Triggering Fan Operation**: When the gas concentration exceeds a predefined threshold or when gas presence is detected, the Arduino code triggers the operation of the fan component. This can be achieved by activating a digital output pin connected to a relay or a motor driver that controls the fan.

7. **Fan Activation**: The digital output pin of the Arduino activates the relay or motor driver, which, in turn, provides power to the fan component. The fan starts operating, facilitating air circulation and ventilation in the area affected by the gas leak.

8. **Safety Measures**: In addition to activating the fan, the Arduino code can incorporate other safety measures such as triggering alarms, sending notifications, or initiating an emergency shutdown procedure. These actions can help alert personnel and ensure prompt response to the gas leak situation.

By utilizing the gas sensor's detection capability and integrating it with the Arduino board, the system can detect gas leaks in an industrial environment and activate the fan component for ventilation purposes. The Arduino programming code plays a crucial role in monitoring the gas sensor readings, determining gas presence or concentration, and controlling the fan component based on predefined thresholds or conditions.

TEMPERATURE SENSOR:-



The working principle of a temperature sensor can be utilized to monitor temperature changes in an environment and activate a buzzer when the temperature exceeds a certain threshold. Here's an overview of how the system can work:

1. **Sensor Setup**: The temperature sensor, such as a thermistor or a digital temperature sensor, is installed in the area where temperature monitoring is required. The sensor should be placed in a location that accurately represents the ambient temperature of the environment.

2. **Initialization**: The Arduino board is connected to the temperature sensor, and the necessary analog or digital input pins are defined in the Arduino programming code to interface with the sensor.

3. **Powering the Sensor**: The temperature sensor is powered using the appropriate voltage supply (usually 5V) through the Arduino board or an external power supply. Ensure that the power supply is stable to obtain accurate temperature readings.

4. **Temperature Monitoring**: The temperature sensor continuously measures the temperature of the surrounding environment. The sensor converts the temperature into an electrical signal that can be read by the Arduino board.

5. **Signal Processing**: The Arduino programming code reads the analog or digital signal from the temperature sensor to obtain the temperature value. If required, the code may perform additional calculations or conversions to obtain the temperature in a desired unit (e.g., Celsius or Fahrenheit).

6. **Threshold Check**: The Arduino code compares the obtained temperature value with a predefined threshold. If the temperature exceeds the threshold, it indicates an increase beyond the desired limit.

7. **Buzzer Activation**: When the temperature exceeds the threshold, the Arduino code triggers the activation of a digital output pin connected to a buzzer. The buzzer emits an audible sound to alert the personnel of the temperature increase.

8. **Buzzer Control**: The digital output pin connected to the buzzer is set to HIGH by the Arduino code to turn on the buzzer and produce sound. If the temperature falls below the threshold, the digital output pin is set to LOW to turn off the buzzer.

9. **Safety Measures**: In addition to activating the buzzer, the Arduino code can incorporate other safety measures, such as triggering alarms, sending notifications, or initiating appropriate actions to mitigate the temperature increase. These actions can help prevent any potential hazards or damage caused by high temperatures.

By utilizing the temperature sensor's measurement capability and integrating it with the Arduino board, the system can monitor temperature changes and activate a buzzer when the temperature exceeds a predefined threshold. The Arduino programming code plays a crucial role in reading the temperature sensor values, comparing them with the threshold, and controlling the buzzer accordingly.

CODE:-

```
#include<Servo.h>
Servo servo_5;
const int pingUltra=3;
const int smokePin=A0;
const int buzzerSm=2;
const int tempSen=A1;
const int fanExTemp=4;
const int pirSensor=10;
const int pirMotor=9;
const int pirBulb=7;

int smoke=0;
int temp=0;
int pir=0;
void setup()
{
servo_5.attach(5);

pinMode(smokePin, INPUT);
pinMode(buzzerSm, OUTPUT);

pinMode(tempSen, INPUT);
pinMode(fanExTemp, OUTPUT);

pinMode(pirSensor, INPUT);
  pinMode(pirMotor, OUTPUT);

pinMode(pirBulb, OUTPUT);
}
void loop() {
```

```
long duration, entryDis;

pinMode(pingUltra, OUTPUT);

  digitalWrite(pingUltra, LOW);
  delayMicroseconds(2);
  digitalWrite(pingUltra, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingUltra, LOW);


pinMode(pingUltra, INPUT);

duration= pulseIn(pingUltra, HIGH);
entryDis= microsecondsToCentimeters(duration);

if (entryDis<30)
{
  servo_5.write(90);
  delay(2000);

}

else{
  servo_5.write(0);
}
```

```
smoke = analogRead(smokePin);


if (smoke >=210) {
  digitalWrite(buzzerSm, HIGH);
}
  else{
    digitalWrite(buzzerSm,LOW);
  }
  temp=(-40 + 0.488155*(analogRead(tempSen)-20));
  if(temp>=30){
    digitalWrite(fanExTemp,HIGH);
  }else{
    digitalWrite(fanExTemp,LOW);
  }
  pir=digitalRead(pirSensor);
  if(pir==HIGH)
  {
    digitalWrite(pirMotor,HIGH);
    digitalWrite(pirBulb,HIGH);
  }
  else
  {
    digitalWrite(pirMotor,LOW);
    digitalWrite(pirBulb,LOW);
  }
  Serial.print("dis: ");
  Serial.println(entryDis);
  Serial.print("smoke:");
  Serial.println(smoke);
  Serial.print("temp:");
```

```
  Serial.println(temp);
  Serial.print("dete:");
  Serial.println(pir);
  delay(1000);
}

  long microsecondsToCentimeters(long microseconds){
    return microseconds/29/2;
  }
```

Code explaination:-

1. The code begins by including the Servo library to control the servo motor.
2. Global variables are declared to assign the pins for various components, including the ultrasonic sensor, smoke sensor, buzzer, temperature sensor, fan, PIR sensor, PIR motor, and PIR bulb.
3. In the `setup()` function, the servo motor is attached to pin 5, and the pin modes for the sensors, motors, and bulbs are initialized.
4. The `loop()` function is where the main execution of the code takes place. The following actions are performed repeatedly in the loop:
    - The ultrasonic sensor is used to measure the distance using the `pingUltra` pin. If the distance is less than 30 centimeters, the servo motor is positioned at 90 degrees. Otherwise, it is set to 0 degrees.
    - The smoke sensor's analog value is read using the `smokePin`, and if the value is greater than or equal to 210, the buzzer is turned on; otherwise, it is turned off.
    - The temperature sensor's analog value is read using the `tempSen` pin, and the temperature in degrees Celsius is calculated using the provided formula. If the temperature is greater than or equal to 30 degrees, the fan is turned on; otherwise, it is turned off.
    - The PIR sensor's digital value is read using the `pirSensor` pin. If the PIR sensor detects motion (HIGH), the PIR motor and bulb are turned on; otherwise, they are turned off.
    - The values of distance, smoke, temperature, and PIR detection are printed to the serial monitor for debugging purposes.
    - A delay of 1 second is added to the loop before the next iteration.

The code includes a helper function `microsecondsToCentimeters` that converts the measured duration in microseconds to centimeters using the speed of sound. This function is used to calculate the distance from the ultrasonic sensor.

Note: Ensure that the necessary libraries are installed in your Arduino IDE, and the components are connected correctly to the corresponding pins as per the code.

Conclusion:

In conclusion, the IoT-based Smart Factory System implemented using Arduino programming has proven to be a valuable solution for enhancing industrial automation and efficiency. By leveraging the power of Internet of Things (IoT) technology, the system enables real-time monitoring, control, and data analysis of various processes and equipment within a factory environment.

The project successfully demonstrated how Arduino, combined with sensors, actuators, and relay modules, can be utilized to connect physical devices and collect data from the factory floor. This data is then transmitted to a central server or cloud platform, where it can be analyzed and visualized to provide valuable insights for optimizing production processes, improving quality control, and reducing downtime.

The importance of this IoT-based Smart Factory System lies in its ability to streamline operations, increase productivity, and enhance overall efficiency. By automating certain tasks, such as monitoring temperature, humidity, machine status, or inventory levels, the system reduces the need for manual intervention and human error, leading to higher accuracy and improved decision-making.

Furthermore, the integration of Arduino programming with IoT allows for remote access and control of factory equipment. This feature enables operators or managers to monitor and adjust parameters in real-time, even from off-site locations, providing greater flexibility and responsiveness to changing production demands or unforeseen issues.

Additionally, the project highlights the importance of data analytics in the context of a smart factory. The collected data can be processed and analyzed using various algorithms and techniques to extract valuable insights. This information can be used for predictive maintenance, anomaly detection, resource optimization, and other data-driven decision-making processes that can significantly improve overall factory performance.

Overall, the IoT-based Smart Factory System implemented using Arduino programming offers numerous benefits, including increased efficiency, enhanced productivity, improved quality control, and cost savings. It opens up possibilities for further advancements in industrial automation and lays the foundation for a more intelligent and connected manufacturing environment.

By continuously refining and expanding upon this project, manufacturers can harness the full potential of IoT technology and Arduino programming to create smart factories that are more agile, adaptive, and capable of meeting the challenges of the rapidly evolving industrial landscape.

Conclusion:

In conclusion, the IoT-based Smart Factory System implemented using Arduino programming has proven to be a valuable solution for enhancing industrial automation and efficiency. By leveraging the power of Internet of Things (IoT) technology, the system enables real-time monitoring, control, and data analysis of various processes and equipment within a factory environment.

The project successfully demonstrated how Arduino, combined with sensors, actuators, and relay modules, can be utilized to connect physical devices and collect data from the factory floor. This data is then transmitted to a central server or cloud platform, where it can be analyzed and visualized to provide valuable insights for optimizing production processes, improving quality control, and reducing downtime.

The importance of this IoT-based Smart Factory System lies in its ability to streamline operations, increase productivity, and enhance overall efficiency. By automating certain tasks, such as monitoring temperature, humidity, machine status, or inventory levels, the system reduces the need for manual intervention and human error, leading to higher accuracy and improved decision-making.

Furthermore, the integration of Arduino programming with IoT allows for remote access and control of factory equipment. This feature enables operators or managers to monitor and adjust parameters in real-time, even from off-site locations, providing greater flexibility and responsiveness to changing production demands or unforeseen issues.

Additionally, the project highlights the importance of data analytics in the context of a smart factory. The collected data can be processed and analyzed using various algorithms and techniques to extract valuable insights. This information can be used for predictive maintenance, anomaly detection, resource optimization, and other data-driven decision-making processes that can significantly improve overall factory performance.

Overall, the IoT-based Smart Factory System implemented using Arduino programming offers numerous benefits, including increased efficiency, enhanced productivity, improved quality control, and cost savings. It opens up possibilities for further advancements in industrial automation and lays the foundation for a more intelligent and connected manufacturing environment.

By continuously refining and expanding upon this project, manufacturers can harness the full potential of IoT technology and Arduino programming to create smart factories that are more agile, adaptive, and capable of meeting the challenges of the rapidly evolving industrial landscape.