1. In a petrochemical industry having boiler for generating steam, you need to develop a system to monitor the temperature of the boiler. The following conditions are crucial for monitoring the boiler.

Temperature >= 600°C-Display an alert message "Boiler is Overheating!" in serial monitor and glow red LED. Also turn On the cooling system.

Temperature < 450°C (idle state) - Display "Idle State" in serial monitor and glow yellow LED.

Temperature between 450°C and 600°C Display "Boiler Working" in serial monitor and glow green LED

Write an Arduino program to develop an optimal embedded system for the petrochemical plant.

#include <DHT.h>

#define DHTPIN 2 // Pin for DHT sensor data

#define RED\_LED\_PIN 3 // Pin for Red LED

#define YELLOW\_LED\_PIN 4 // Pin for Yellow LED

#define GREEN\_LED\_PIN 5 // Pin for Green LED

#define COOLING\_PIN 6 // Pin for cooling system

// Initialize DHT sensor

DHT dht(DHTPIN, DHT11);

void setup() {

// Start serial communication

Serial.begin(9600);

// Initialize DHT sensor

dht.begin();

// Set LED and cooling system pins as outputs

pinMode(RED\_LED\_PIN, OUTPUT);

pinMode(YELLOW\_LED\_PIN, OUTPUT);

pinMode(GREEN\_LED\_PIN, OUTPUT);

pinMode(COOLING\_PIN, OUTPUT);

// Start with all LEDs off and cooling system off

digitalWrite(RED\_LED\_PIN, LOW);

digitalWrite(YELLOW\_LED\_PIN, LOW);

digitalWrite(GREEN\_LED\_PIN, LOW);

digitalWrite(COOLING\_PIN, LOW);

}

void loop() {

// Read the temperature from the DHT sensor

float temperature = dht.readTemperature();

// Control LEDs and cooling system based on temperature

if (temperature >= 600) {

Serial.println("Boiler is Overheating!");

digitalWrite(RED\_LED\_PIN, HIGH);

digitalWrite(YELLOW\_LED\_PIN, LOW);

digitalWrite(GREEN\_LED\_PIN, LOW);

digitalWrite(COOLING\_PIN, HIGH);

} else if (temperature < 450) {

Serial.println("Idle State");

digitalWrite(RED\_LED\_PIN, LOW);

digitalWrite(YELLOW\_LED\_PIN, HIGH);

digitalWrite(GREEN\_LED\_PIN, LOW);

digitalWrite(COOLING\_PIN, LOW);

} else {

Serial.println("Boiler Working");

digitalWrite(RED\_LED\_PIN, LOW);

digitalWrite(YELLOW\_LED\_PIN, LOW);

digitalWrite(GREEN\_LED\_PIN, HIGH);

digitalWrite(COOLING\_PIN, LOW);

}

// Wait for a while before the next reading

delay(1000);

}

2. Write an Arduino program to design an automatic hand sanitizer dispenser and temperature measurement system for a super market whose requirements are given below.

The system should be contact less. The door opens only if the temperature is within the normal temperature limit. An alarm is sounded if a person is identified with temperature which is not within the limit. Normal temperature limit: 97F to 99F.

#include <Servo.h>

#include <DHT.h>

int trigPin = 7; // Trig pin of ultrasonic sensor

int echoPin = 6; // Echo pin of ultrasonic sensor

int buzzerPin = 8; // Buzzer pin

int servoPin = 9; // Servo motor pin

int dhtPin = 2; // DHT sensor pin

int distanceThreshold = 10; // Distance in cm to trigger sanitizer (hand detected)

Servo doorServo;

DHT dht(dhtPin, DHT22);

void setup() {

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(buzzerPin, OUTPUT);

doorServo.attach(servoPin);

Serial.begin(9600);

dht.begin();

doorServo.write(0); // Door initially closed

}

void loop() {

// Measure distance to detect hand

long duration, distance

duration = pulseIn(echoPin, HIGH);

distance = (duration / 2) /29; // Convert to cm

// If hand is detected within threshold distance

if (distance < distanceThreshold) {

float tempF = dht.readTemperature(true); // Read temperature in Fahrenheit

if (tempF >= 97 && tempF <= 99) {

// Open the door if temperature is within the normal range

doorServo.write(90); // Open door

Serial.println("Temperature Normal. Door Opening...");

delay(5000); // Keep door open for 5 seconds

doorServo.write(0); // Close door

} else {

// Sound alarm if temperature is out of range

Serial.println("Temperature Abnormal. Alarm Triggered!");

digitalWrite(buzzerPin, HIGH);

delay(5000); // Buzzer durartion

digitalWrite(buzzerPin, LOW);

}

}

delay(1000); // Wait before next reading

}

3. Write an Arduino program to design a smart parking system using HC-SR04 ultrasonic sensor, servo motor, buzzer and Arduino Uno. With the following specification:

The ultrasonic sensor module placed near the entry gate continuously checks for the incoming vehicles.  
When a vehicle comes closer to the ultrasonic sensor detection area and parking slot is available then the system opens a gate barrier to 90° (close after 10 seconds) to allow the vehicle to the parking slot and decrements the available parking slot by 1.  
If no parking slot is available, switch on the buzzer for 5 seconds.  
Have a similar system on the exit and increment the free slot by 1 for every vehicle which leaves the parking slot.  
Note: Total capacity of the parking slot is 15.

#include <Servo.h>

// Pin definitions

int trigPinEntry = 7; // Trig pin of entry ultrasonic sensor

int echoPinEntry = 6; // Echo pin of entry ultrasonic sensor

int trigPinExit = 5; // Trig pin of exit ultrasonic sensor

int echoPinExit = 4; // Echo pin of exit ultrasonic sensor

int buzzerPin = 8; // Buzzer pin

int servoPinEntry = 9; // Servo motor pin for entry gate

int servoPinExit = 10; // Servo motor pin for exit gate

int distanceThreshold = 10; // Distance in cm to detect vehicle

int buzzerDuration = 5000; // Buzzer duration in ms

int parkingCapacity = 15; // Total parking slots

int availableSlots = parkingCapacity;

Servo entryGateServo(servoPinEntry,DHT11);

Servo exitGateServo(servoPinExit,DHT11);

void setup() {

pinMode(trigPinEntry, OUTPUT);

pinMode(echoPinEntry, INPUT);

pinMode(trigPinExit, OUTPUT);

pinMode(echoPinExit, INPUT);

pinMode(buzzerPin, OUTPUT);

entryGateServo.attach(servoPinEntry);

exitGateServo.attach(servoPinExit);

Serial.begin(9600);

entryGateServo.write(0); // Entry gate initially closed

exitGateServo.write(0); // Exit gate initially closed

}

void loop() {

// Check for incoming vehicle at entry

if (detectVehicle(trigPinEntry, echoPinEntry)) {

if (availableSlots > 0) {

availableSlots--;

Serial.print("Vehicle Entered. Available Slots: ");

Serial.println(availableSlots);

entryGateServo.write(90);

delay (5000);

entryGateServo.write(0);

} else {

Serial.println("Parking Full. Triggering Buzzer.");

digitalWrite(buzzerPin, HIGH);

delay(buzzerDuration); // Buzzer on for 5 seconds

digitalWrite(buzzerPin, LOW);

}

}

// Check for exiting vehicle at exit

if (detectVehicle(trigPinExit, echoPinExit)) {

if (availableSlots < parkingCapacity) {

availableSlots++;

Serial.print("Vehicle Exited. Available Slots: ");

Serial.println(availableSlots);

exitGateServo.write(90);

delay (5000);

exitGateServo.write(0);

}

}

delay(1000); // Small delay before the next check

}

bool detectVehicle(int trigPin, int echoPin) {

long duration, distance;

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) /29; // Convert to cm

return (distance < distanceThreshold);

}

5. Design an automatic lawn water sprinkler system which is scheduled to water the lawn once in every 8 hours with the following condition.

Need to vary the angle of the sprinkler by 1degree for every 2 seconds to reach 180° (4 marks).

During returning cycles from 180° to 0° the watering hose should stop for 5 seconds for every 2° step (4 marks).

#include <Servo.h>

int servoPin = 9; // Pin for the servo motor

int wateringInterval = 8 \* 60 \* 60 \* 1000; // 8 hours in milliseconds

const int stepDelay = 2000; // 2 seconds delay per 1 degree

const int stopDelay = 5000; // 5 seconds delay per 2 degrees during return cycle

unsigned long previousMillis = 0;

Servo sprinklerServo;

void setup() {

sprinklerServo.attach(servoPin);

sprinklerServo.write(0); // Start with the sprinkler at 0 degrees

Serial.begin(9600);

}

void loop() {

unsigned long currentMillis = millis();

// Check if it's time to start the watering cycle

if (currentMillis - previousMillis >= wateringInterval) {

previousMillis = currentMillis;

// Watering cycle: Rotate from 0° to 180°

for (int angle = 0; angle <= 180 angle++) {

sprinklerServo.write(angle);

delay(stepDelay);

}

// Return cycle: Rotate from 180° to 0° with stops

for (int angle = 180; angle >= 0; angle--) {

sprinklerServo.write(angle);

if (angle % 2 == 0) {

delay(stopDelay); // Stop for 5 seconds for every 2° step

} else {

delay(stepDelay); // Continue moving for other angles

}

}

}

}

6. Write an Arduino program for controlling the solar panel with the following requirements. The sun is moving from east to west at a rate of 15 degree/hour and the solar panel is also synchronized with the motion of sun to achieve maximum efficiency.

Design a panel control system which varies the solar panel angle automatically for every 1 hours synchronized with the sun movement (8 marks).

Check the temperature of the panel, if it is greater than 90°C turn ON the cooling system using digital interface (5 marks).

Also, Display the warning sign by glowing RED LED with the message indicating "solar panel overheating" using serial interface (2 marks).

#include <Servo.h>

#include <DHT.h>

int servoPin = 9; // Pin for the servo motor controlling the solar panel

int dhtPin = 2; // Pin for the DHT temperature sensor

int coolingSystemPin = 10; // Digital pin to control the cooling system

int redLEDPin = 11; // Digital pin for the red LED

int updateInterval = 3600000; // 1 hour in milliseconds

// Create Servo and DHT objects

Servo solarPanelServo;

DHT dht(dhtPin, DHT11); // Initialize DHT sensor (use DHT22 if you have that model)

int previousMillis = 0;

int currentAngle = 0;

void setup() {

solarPanelServo.attach(servoPin);

pinMode(coolingSystemPin, OUTPUT);

pinMode(redLEDPin, OUTPUT);

Serial.begin(9600);

dht.begin(); // Initialize the DHT sensor

solarPanelServo.write(currentAngle); // Initialize servo position

digitalWrite(coolingSystemPin, LOW); // Cooling system off initially

digitalWrite(redLEDPin, LOW); // Red LED off initially

}

void loop() {

int currentMillis = millis();

// Check if it's time to adjust the panel angle

if (currentMillis - previousMillis >= updateInterval) {

previousMillis = currentMillis;

// Update the solar panel angle

currentAngle += 15;

if (currentAngle > 180) {

currentAngle = 0; // Reset to 0 if it exceeds 180 degrees

}

solarPanelServo.write(currentAngle);

// Check the temperature of the panel using DHT sensor

float temperature = dht.readTemperature();

// Check if the temperature exceeds the limit

if (temperature > 90) {

digitalWrite(coolingSystemPin, HIGH); // Turn on the cooling system

digitalWrite(redLEDPin, HIGH); // Turn on the red LED

Serial.println("Warning: Solar panel overheating!");

} else {

digitalWrite(coolingSystemPin, LOW); // Turn off the cooling system

digitalWrite(redLEDPin, LOW); // Turn off the red LED

}

delay(1000); // Short delay before the next temperature check

}