NANDHA ENGINEERING COLLEGE

(Autonomous)

Erode - 52

****

**RECORD NOTE BOOK**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(Internet of Things)**

**22CIP09 - EMBEDDED SYSTEM LABORATORY**

**III YEAR/ V SEMESTER**

**NANDHA ENGINEERING COLLEGE, Erode - 638 052**

**(Autonomous)**

**BONAFIDE CERTIFICATE**

**Register No.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

Certified that this is the bonafide Record of work done by Mr./Ms……………………………………………………..of **III** year **V** semester **B.E. COMPUTER SCIENCE AND ENGINEERING (INTERNET OF THINGHS)** during the academic year **2022 - 2023** in the **22CIP09 - EMBEDDED SYSTEM LABORATORY**.

## Staff in Charge Head of the Department

### Submitted for the End Semester Practical Examination held on……………………

**Internal Examiner External Examiner**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **22CIP09 - EMBEDDED SYSTEM LABORATORY** | | | | |
|  | **L** | **T** | **P** | **C** |
|  | **0** | **0** | **4** | **2** |
| **OBJECTIVES:** | | | | |
| • Provide a comprehensive understanding of embedded systems and their application in IoT  • Equip students with the knowledge and skills to effectively implement and manage a range of IoT communication protocols. | | | | |
| **COURSE OUTCOMES:** | | | | |
| At the end of this course, the students will be able to  **CO1:** Utilize the knowledge of embedded systems and their range of uses in the Internet of Things environment.  **CO2**: Interfacing the various kinds of embedded system components with Internet of Things.  **CO3**: Apply embedded programming techniques to solve real-world problems.  **CO4**: Develop an integrated hardware and software solutions for embedded systems to ensuring functionality and efficiency.  **CO5**: Develop an integrated hardware and software solutions for embedded systems to ensuring functionality and efficiency. | | | | |
| **List of Experiments:**  **1.** Monitoring a machinery vibration using vibration sensors  **2.** Interfacing an MQ-2 Gas Sensor with an LED  **3.** Interfacing an ADXL345 Accelerometer with ARM  **4.** Interfacing soil moisture sensor with ARM  **5.** Implementing a program to heartbeat sensor and ARM  **6.** Interfacing UART for LED Control between IoT and PC  **7.** Application to transmit & receive a character through RS232 and Bluetooth low energy Communication  **8.** Interfacing GSM Module with IoT and Sending Sensor Data to Cloud  **9.** Interfacing ESP8266(WIFI Module) with IoT for HTTP Communication  **10.** Implement Zigbee interface for Data Transmission with IoT | | | | |

**TOTAL: P: 60 = 60 PERIODS**

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exercise** | **DATE** | **TITLE** | **PAGE NO.** | **MARKS** | **SIGN** |
| **No.** |
| **1** |  | Monitoring a machinery vibration using vibration sensors |  |  |  |
| **2** |  | Interfacing an MQ-2 Gas Sensor with an LED |  |  |  |
| **3** |  | Interfacing an ADXL345 Accelerometer with ARM |  |  |  |
| **4** |  | Interfacing soil moisture sensor with ARM |  |  |  |
| **5** |  | Implementing a program to heartbeat sensor and ARM |  |  |  |
| **6** |  | Interfacing UART for LED Control between IoT and PC |  |  |  |
| **7** |  | Application to transmit & receive a character through RS232 and Bluetooth low energy Communication |  |  |  |
| **8** |  | Interfacing GSM Module with IoT and Sending Sensor Data to Cloud |  |  |  |
| **9** |  | Interfacing ESP8266(WIFI Module) with IoT for HTTP Communication |  |  |  |
| **10** |  | Implement Zigbee interface for Data Transmission with IoT |  |  |  |
| **AVERAGE** | | |  |  |  |

| **EX .NO : 01** | **MONITORING A MACHINERY VIBRATION USING VIBRATION SENSORS** |
| --- | --- |
| **DATE :** |

**AIM:**

To monitor a machinery vibration using vibration sensors.

**Procedure:**

**Step-1:** Connect the vibration sensor to the microcontroller (VCC, GND, and signal pin to ADC/Digital pin).

**Step-2:** Assign the ADC or Digital pin in the microcontroller code.

**Step-3:** Initialize the ADC/Digital pin in input mode to read vibration data.

**Step-4:** Write the code to capture vibration data and trigger actions (e.g., LED blink or buzzer).

**Step-5:** Upload and execute the program.

**Step-6:** Observe the vibration level readings or indicator response.

**Program:**

#include <Wire.h>

#define Register\_ID 0

#define Register\_2D 0x2D

#define Register\_X0 0x32

#define Register\_X1 0x33

#define Register\_Y0 0x34

#define Register\_Y1 0x35

#define Register\_Z0 0x36

#define Register\_Z1 0x37

int ADXAddress = 0x53; //I2C address

int reading = 0;

int val = 0;

int X0,X1,X\_out;

int Y0,Y1,Y\_out;

int Z1,Z0,Z\_out;

double Xg,Yg,Zg;

void setup()

{

Serial.begin(9600);

delay(100);

Wire.begin();

delay(100);

Wire.beginTransmission(ADXAddress);

Wire.write(Register\_2D);

Wire.write(8);

Wire.endTransmission();

Serial.println("Accelerometer Test ");

}

void loop()

{

Wire.beginTransmission(ADXAddress);

Wire.write(Register\_X0);

Wire.write(Register\_X1);

Wire.endTransmission();

Wire.requestFrom(ADXAddress,2);

if(Wire.available()<=2);

{

X0 = Wire.read();

X1 = Wire.read();

X1 = X1<<8;

X\_out = X0+X1;

}

Wire.beginTransmission(ADXAddress);

Wire.write(Register\_Y0);

Wire.write(Register\_Y1);

Wire.endTransmission();

Wire.requestFrom(ADXAddress,2);

if(Wire.available()<=2);

{

Y0 = Wire.read();

Y1 = Wire.read();

Y1 = Y1<<8;

Y\_out = Y0+Y1;

}

Wire.beginTransmission(ADXAddress);

Wire.write(Register\_Z0);

Wire.write(Register\_Z1);

Wire.endTransmission();

Wire.requestFrom(ADXAddress,2);

if(Wire.available()<=2);

{

Z0 = Wire.read();

Z1 = Wire.read();

Z1 = Z1<<8;

Z\_out = Z0+Z1;

}

Xg = X\_out/256.00;

Yg = Y\_out/256.00;

Zg = Z\_out/256.00;

Serial.print("X=");

Serial.print(Xg);

Serial.print("\tY=");

Serial.print(Yg);

Serial.print("\tZ=");

Serial.println(Zg);

delay(300);

}

**CONNECTIONS:**

* VCC → 3.3V
* GND → GND
* SDA → A4
* SCL → A5

**Output:**

X=0.10 Y=-0.02 Z=0.99

X=0.11 Y=-0.03 Z=0.98

X=0.12 Y=-0.01 Z=1.00

**VIVA QUESTIONS**

* What are the types of vibration sensors available, and which one did you use in this experiment?
* How does the microcontroller process the signal from the vibration sensor?
* Why is it important to monitor machinery vibrations?
* What is the relationship between vibration frequency and machinery health?
* How would you calibrate the vibration sensor for different machinery?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

Vibration sensors detect machinery vibrations and display the value of the vibration in serial monitor.

| **EX .NO : 02** | **INTERFACING AN MQ-2 GAS SENSOR WITH AN LED** |
| --- | --- |
| **DATE :** |

**AIM:**

To Interfacing an MQ-2 Gas Sensor with an LED

**Procedure:**

**Step-1:** Connect the MQ-2 gas sensor to the microcontroller (VCC, GND, and signal pin to ADC).

**Step-2:** Assign the signal pin and an LED pin in the program.

**Step-3:** Initialize the signal pin as input and the LED pin as output.

**Step-4:** Write the code to read gas levels and control the LED based on a threshold.

**Step-5:** Upload and execute the program.

**Step-6:** Test with gas exposure and observe LED behavior.

**Program:**

int redLed = 12;

int greenLed = 11;

int buzzer = 10;

int smokeA0 = A5;

// Your threshold value

int sensorThres = 250;

void setup() {

  pinMode(redLed, OUTPUT);

  pinMode(greenLed, OUTPUT);

  pinMode(buzzer, OUTPUT);

  pinMode(smokeA0, INPUT);

  Serial.begin(9600);

}

void loop() {

  int analogSensor = analogRead(smokeA0);

  Serial.print("Pin A0: ");

  Serial.println(analogSensor);

  // Checks if it has reached the threshold value

  if (analogSensor > sensorThres)

  {

    digitalWrite(redLed, HIGH);

    digitalWrite(greenLed, LOW);

    tone(buzzer, 1000, 200);

  }

  else

  {

    digitalWrite(redLed, LOW);

    digitalWrite(greenLed, HIGH);

    noTone(buzzer);

  }

  delay(100);

}

**Connections:**

**MQ-2 Gas Sensor:**

VCC → 5V

GND → GND

AOUT → A5

**Red LED:**

Anode → Pin 12 (via 220Ω resistor)

Cathode → GND

**Green LED:**

Anode → Pin 11 (via 220Ω resistor)

Cathode → GND

**Buzzer:**

Positive Terminal → Pin 10

Negative Terminal → GND

**Output:**

Pin A0: 150

Pin A0: 180

Pin A0: 200

Pin A0: 280

Pin A0: 310

Pin A0: 220

**LED light blinks**

**VIVA QUESTIONS:**

* What types of gases can the MQ-2 sensor detect?
* How does the MQ-2 sensor communicate with the microcontroller?
* Why is the MQ-2 gas sensor used in safety applications?
* How would you calibrate the MQ-2 sensor for different gases?
* Can you explain how to convert the analog output of the MQ-2 to a readable value?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

MQ-2 detects gas levels, turning on an LED when the threshold is surpassed.

| **EX .NO : 03** | **INTERFACING AN ADXL345 ACCELEROMETER WITH ARM** |
| --- | --- |
| **DATE :** |

**AIM:**

To interfacing an ADXL345 Accelerometer with ARM.

**Procedure:**

**Step-1:** Connect the ADXL345 to the ARM board via I2C/SPI (VCC, GND, SCL/SDA).

**Step-2:** Assign the I2C/SPI pins in the ARM program.

**Step-3:** Configure the I2C/SPI interface to initialize communication with ADXL345.

**Step-4:** Write the code to read accelerometer data (X, Y, Z axes).

**Step-5:** Upload and execute the program.

**Step-6:** Observe real-time acceleration data on the serial monitor or display.

**Program:**

rom smbus2 import SMBus

import time

**# ADXL345 I2C address**

ADXL345\_ADDRESS = 0x53

**# Register Addresses**

DATA\_FORMAT\_REGISTER = 0x31

POWER\_CTL\_REGISTER = 0x2D

DATA\_REGISTER\_START = 0x32 # Starting register for XYZ data

**# Initialize I2C bus**

bus = SMBus(1)

def init\_adxl345():

try:

**# Set the power control register to turn the device on**

bus.write\_byte\_data(ADXL345\_ADDRESS, POWER\_CTL\_REGISTER, 0x08) # Measure mode

**# Set data format to full resolution, +/- 16g**

bus.write\_byte\_data(ADXL345\_ADDRESS, DATA\_FORMAT\_REGISTER, 0x0B) # Full resolution

print("ADXL345 initialized successfully.")

except OSError as e:

print(f"Error initializing ADXL345: {e}")

def read\_accelerometer\_data():

try:

**# Read 6 bytes of data (2 bytes for each axis: X, Y, Z)**

data = bus.read\_i2c\_block\_data(ADXL345\_ADDRESS, DATA\_REGISTER\_START, 6)

**# Convert the data to 16-bit values**

x = (data[1] << 8) | data[0] # X-axis data

y = (data[3] << 8) | data[2] # Y-axis data

z = (data[5] << 8) | data[4] # Z-axis data

return x, y, z

except OSError as e:

print(f"Error reading accelerometer data: {e}")

return None, None, None

**# Initialize the ADXL345**

init\_adxl345()

**# Main loop to read and print data**

try:

while True:

x, y, z = read\_accelerometer\_data()

if x is not None and y is not None and z is not None:

print(f"X: {x}, Y: {y}, Z: {z}")

else:

print("Failed to read data")

time.sleep(1)

except KeyboardInterrupt:

print("Program stopped by user")

finally:

bus.close()

**Connections:**

VCC (ADXL345) → 3.3V

GND (ADXL345) → GND

SDA (ADXL345) → GPIO 2 (SDA)

SCL (ADXL345) → GPIO 3 (SCL)

**Output:**

X: 1024, Y: 2048, Z: 512

X: 1050, Y: 2050, Z: 540

X: 1070, Y: 2080, Z: 550

**VIVA QUESTIONS**:

* What is the range of acceleration measurable by the ADXL345 accelerometer?
* How does the ADXL345 communicate with the ARM microcontroller?
* What is the importance of the accelerometer in motion detection systems?
* How would you use the accelerometer data in practical applications like mobile phones or drones?
* Can you explain how to convert the raw accelerometer data into meaningful units like m/s²?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

ADXL345 measures Vibrations in three axes and ARM processor processes the data and displays the result in monitor.

| **EX .NO : 04** | **INTERFACING SOIL MOISTURE SENSOR WITH ARM** |
| --- | --- |
| **DATE :** |

**AIM:**

To interfacing soil moisture sensor with ARM

**Procedure:**

**Step-1:** Connect the soil moisture sensor to the ARM board (VCC, GND, and signal pin to ADC).

**Step-2:** Assign the ADC pin in the ARM program.

**Step-3:** Configure the ADC pin to read analog values from the sensor.

**Step-4:** Write the code to convert sensor readings into soil moisture levels.

**Step-5:** Upload and execute the program.

**Step-6:** Insert the sensor into the soil and observe the moisture levels.

**Program:**

import RPi.GPIO as GPIO

import time

**#GPIO SETUP**

channel = 4

GPIO.setmode(GPIO.BCM)

GPIO.setup(channel, GPIO.IN)

def callback(channel):

        if GPIO.input(channel):

                print ("Water Detected!")

        else:

                print ("Water Detected!")

GPIO.add\_event\_detect(channel, GPIO.BOTH, bouncetime=300)

GPIO.add\_event\_callback(channel, callback)

while True:

        time.sleep(0)

**Connections:**

* VCC -> 5V
* GND -> GND
* DATA-> GPIO4

**Output:**

GPIO Value: 1

**Water Detected!**

GPIO Value: 0

**No Water Detected!**

GPIO Value: 1

**Water Detected!**

GPIO Value: 0

**No Water Detected!**

**VIVA QUESTIONS:**

* What is the principle of operation of the soil moisture sensor?
* How can the data from the soil moisture sensor be used in an automated irrigation system?
* What factors can affect the accuracy of the soil moisture sensor readings?
* How do you calibrate the soil moisture sensor?
* How would you interface a soil moisture sensor with a cloud-based system for remote monitoring?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

Soil moisture levels are detected and processed by ARM to automate irrigation systems.

| **EX .NO : 05** | **IMPLEMENTING A PROGRAM TO HEARTBEAT SENSOR AND ARM** |
| --- | --- |
| **DATE :** |

**AIM:**

To implementing a program to heartbeat sensor and ARM**.**

**Procedure:**

**Step-1:** Connect the heartbeat sensor to the ARM board (VCC, GND, and signal pin to ADC).

**Step-2:** Assign the ADC pin for the heartbeat sensor in the program.

**Step-3:** Configure the ADC pin to read analog signals and filter the data.

**Step-4:** Write the code to calculate and display the heart rate.

**Step-5:** Upload and execute the program.

**Step-6:** Test with a finger on the sensor and observe the heart rate on the display/serial monitor.

**Program:**

from smbus2 import SMBus

import time

MAX30100\_ADDRESS = 0x57

FIFO\_DATA\_REGISTER = 0x07 # Adjust this register based on the datasheet

**# Initialize I2C bus**

bus = SMBus(1)

def init\_max30100():

try:

**# Reset the MAX30100**

bus.write\_byte\_data(MAX30100\_ADDRESS, 0x00, 0x40) # Reset register

**# Mode config register**

bus.write\_byte\_data(MAX30100\_ADDRESS, 0x06, 0x03) # Heart rate + SpO2 mode

**# SPO2 config register**

bus.write\_byte\_data(MAX30100\_ADDRESS, 0x07, 0x27) # 100 samples/second, 400ms pulse width

**# LED config register**

bus.write\_byte\_data(MAX30100\_ADDRESS, 0x09, 0x24) # Set Red and IR LED currents

print("MAX30100 initialized successfully.")

time.sleep(2) # Allow the sensor to stabilize

except OSError as e:

print(f"Error initializing MAX30100: {e}")

def read\_sensor\_data():

try:

**# Read 6 bytes from the FIFO**

data = bus.read\_i2c\_block\_data(MAX30100\_ADDRESS, FIFO\_DATA\_REGISTER, 6)

if len(data) != 6:

print("Unexpected data length: ", len(data))

return None, None

if all(x == 0 for x in data[:4]): # Check if the first four data bytes are all zero

print("No data available in FIFO.")

return None, None

**# Extract the IR and Red values**

ir\_data = (data[0] << 8) | data[1] # IR data from two bytes

red\_data = (data[2] << 8) | data[3] # Red data from two bytes

return ir\_data, red\_data

except OSError as e:

print(f"Error reading sensor data: {e}")

return None, None

**# Initialize the sensor**

init\_max30100()

**# Main loop to read and print data**

try:

while True:

ir, red = read\_sensor\_data()

if ir is not None and red is not None:

print(f"IR Data: {ir}, Red Data: {red}")

else:

print("Failed to read data")

time.sleep(1)

except KeyboardInterrupt:

print("Program stopped by user")

finally:

bus.close()

**Connections:**

**VCC** (MAX30100) → **3.3V**

**GND** (MAX30100) → **GND**

**SDA** (MAX30100) → **GPIO 2** (SDA)

**SCL** (MAX30100) → **GPIO 3** (SCL)

**Output:**

IR Data: 12345, Red Data: 67890

IR Data: 12350, Red Data: 67895

IR Data: 12360, Red Data: 67910

**VIVA QUESTIONS:**

* What is the working principle behind the heartbeat sensor?
* How does the microcontroller process the analog signal from the heartbeat sensor?
* What are the potential applications of heartbeat monitoring in healthcare?
* How can you improve the accuracy of the heartbeat sensor?
* What challenges do you face in interpreting heart rate data from the sensor?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

Heartbeat sensor detects pulse rate, which is processed and displayed by ARM.

| **EX .NO : 06** | **INTERFACING UART FOR LED CONTROL BETWEEN IOT AND PC** |
| --- | --- |
| **DATE :** |

**AIM:**

To Interfacing UART for LED Control between IoT and PC.

**Procedure:**

**Step-1:** Connect the microcontroller (IoT device) to the PC using UART pins (TX, RX, GND).

**Step-2:** Assign UART communication pins and the LED pin in the program.

**Step-3:** Initialize UART communication with the correct baud rate (e.g., 9600 bps).

**Step-4:** Write the code to receive data from the PC and control the LED (e.g., '1' to turn ON, '0' to turn OFF).

**Step-5:** Upload and execute the program.

**Step-6:** Send commands from the PC via a serial terminal and observe the LED's behavior.

**Program:**

int led = 13;

int value = 0;

void setup() {

Serial.begin(9600);

pinMode(led, OUTPUT);

}

void loop() {

if (Serial.available() > 0) {

value = Serial.read();

delay(5);

if (value == '1') {

digitalWrite(led, HIGH);

Serial.println("LED is ON");

}

if (value == '0') {

digitalWrite(led, LOW);

Serial.println("LED is OFF");

}

}

}

**Output:**

1

**LED is ON**

0

**LED is OFF**

**VIVA QUESTIONS:**

* What is UART, and how does it facilitate communication between IoT and PC?
* What are the key differences between UART and other serial communication protocols like SPI or I2C?
* What role does baud rate play in UART communication?
* How would you troubleshoot UART communication errors?
* What is the significance of flow control in UART communication?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

UART enables communication between IoT and PC to control an LED remotely.

| **EX .NO : 07** | **APPLICATION TO TRANSMIT & RECEIVE A CHARACTER THROUGH RS232 AND BLUETOOTH LOW ENERGY COMMUNICATION** |
| --- | --- |
| **DATE :** |

**AIM:**

To Application to transmit & receive a character through RS232 and Bluetooth low energy Communication

**Procedure:**

**Step-1:** Connect the RS232 interface and BLE module to the microcontroller (VCC, GND, TX, RX).

**Step-2:** Assign UART pins for RS232 and Bluetooth communication in the program.

**Step-3:** Initialize UART communication for both interfaces with appropriate baud rates.

**Step-4:** Write the code to receive a character from RS232 and forward it via Bluetooth, and vice versa.

**Step-5:** Upload and execute the program.

**Step-6:** Test by sending and receiving characters through both interfaces.

**Program:**

int led\_pin = 2;

void setup() {

pinMode(led\_pin, OUTPUT);

Serial.begin(9600);

}

void loop() {

if (Serial.available())

{

char data\_received;

data\_received = Serial.read();

if (data\_received == 'O')

{

digitalWrite(led\_pin, HIGH);

Serial.write("LED is now ON!\n");

}

else if (data\_received == 'X')

{

digitalWrite(led\_pin, LOW);

Serial.write("LED is now OFF!\n");

}

else

{

Serial.write("Specify correct option\n");

}

}

}

**Output:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Action** | **Output in Serial Monitor** |
| 'O' | LED ON | LED is now ON! |
| 'X' | LED OFF | LED is now OFF! |
| Any other | Invalid input | Specify correct option |

**VIVA QUESTIONS**:

* What is the difference between RS232 and Bluetooth Low Energy in terms of data transmission?
* How do you establish a connection between the microcontroller and a BLE device?
* Why would you choose BLE over RS232 in IoT applications?
* How do you handle data loss or transmission errors in BLE communication?
* What are the advantages and limitations of using RS232 and BLE for communication?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

RS232 and BLE transmit/receive characters between IoT and devices for wireless communication.

| **EX .NO : 08** | **INTERFACING GSM MODULE WITH IOT AND SENDING SENSOR DATA TO CLOUD** |
| --- | --- |
| **DATE :** |

**AIM:**

To Interfacing GSM Module with IoT and Sending Sensor Data to Cloud

**Procedure:**

**Step-1:** Connect the GSM module to the IoT device (VCC, GND, TX, RX).

**Step-2:** Assign UART pins for GSM communication and connect a sensor (e.g., temperature sensor).

**Step-3:** Initialize UART communication and configure the GSM module (AT commands) for GPRS/HTTP.

**Step-4:** Write the code to read sensor data and send it to a cloud server (e.g., Thingspeak) via HTTP POST.

**Step-5:** Upload and execute the program.

**Step-6:** Observe the sensor data on the cloud dashboard.

**Program:**

#include <SoftwareSerial.h>  
SoftwareSerial gsmSerial(2, 3); // **RX (Arduino) -> TX (GSM), TX (Arduino) -> RX (GSM)**  
void setup() {  
  Serial.begin(9600**);    // Serial Monitor**  
  gsmSerial.begin(9600); **// GSM Module Baud Rate**  
   
  Serial.println("Initializing GSM Module...");  
  delay(3000); **// Delay to give the GSM module time to initialize**  **// Check GSM module connection**  gsmSerial.println("AT");  
  delay(1000);  
  if (gsmSerial.available()) {  
    String response = gsmSerial.readString();  
    Serial.println("GSM Module Response: " + response);  
    if (response.indexOf("OK") != -1) {  
      Serial.println("GSM Module initialized successfully!");  
    } else {  
      Serial.println("GSM Module not responding. Check connections and power.");  
    }  
  } else {  
    Serial.println("No response from GSM Module.");  
  }  
}  
void loop() {  
  // Command options through Serial Monitor  
  if (Serial.available() > 0) {  
    char command = Serial.read();  
  
    if (command == 's') { // Send SMS command  
      sendSMS("0123456789", "xxx yyy zzz"); **// Update with your number and message**  
    }  
    else if (command == 'c') { // Make a call command  
      makeCall("0123456789"); **// Update with your number**  
    }  
    else if (command == 'h') **// Hang up call command**  
 {

hangUpCall();  
    }  
  }  
  **// Check for responses from GSM module and print to Serial Monitor**  
  if (gsmSerial.available()) {  
    Serial.write(gsmSerial.read());  
  }  
}  
void sendSMS(String phoneNumber, String message) {  
  Serial.println("Sending SMS...");  
  gsmSerial.println("AT+CMGF=1"); **// Set SMS to text mode**  
  delay(1000);  
   gsmSerial.print("AT+CMGS=\"");  
  gsmSerial.print(phoneNumber);  
  gsmSerial.println("\"");  
  delay(1000);  
  gsmSerial.print(message);  
  delay(500);  
  gsmSerial.write(26); **// ASCII code for Ctrl+Z to send the SMS**  
  delay(5000); **// Wait for SMS to be sent**  Serial.println("SMS sent to " + phoneNumber);  
}  
void makeCall(String phoneNumber) {  
  Serial.println("Dialing " + phoneNumber);  
  gsmSerial.print("ATD");  
  gsmSerial.print(phoneNumber);  
  gsmSerial.println(";"); **// ';' indicates a voice call**  
  delay(1000);  
  Serial.println("Calling " + phoneNumber);  
}  
void hangUpCall() {  
  Serial.println("Ending call...");  
  gsmSerial.println("ATH"); **// Command to hang up**  
  delay(1000);  
  Serial.println("Call ended.");  
}

**Connection:**

|  |  |
| --- | --- |
| **GSM Module Pin** | **Arduino Nano Pin** |
| **VCC** | **5V** |
| **GND** | **GND** |
| **TXD** | **D2** (RX on Arduino) |
| **RXD** | **D3** (TX on Arduino) |

**Output:**

Initializing 4G LTE Module...

Module Response: OK

Module initialized successfully!

Sending SMS...

SMS sent to 0123456789

Dialing 0123456789

Calling 0123456789

Ending call...

Call ended.

**VIVA QUESTIONS**:

* How does the GSM module communicate with the IoT device?
* What is the role of the cloud in IoT applications?
* What challenges could arise when sending data over GSM compared to Wi-Fi or Ethernet?
* How would you implement security in data transmission via GSM?
* Can you explain the process of sending SMS or sensor data using the GSM module?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

GSM sends sensor data to the cloud for remote monitoring via cellular networks.

| **EX .NO : 09** | **INTERFACING ESP8266(WIFI MODULE) WITH IOT FOR HTTP COMMUNICATION** |
| --- | --- |
| **DATE :** |

**AIM:**

To Interfacing ESP8266 (WIFI Module) with IoT for HTTP Communication

**Procedure:**

**Step-1:** Connect the ESP8266 to the IoT device (VCC, GND, TX, RX).

**Step-2:** Assign UART pins for Wi-Fi communication and configure the ESP8266 (AT commands).

**Step-3:** Initialize UART communication and connect to a Wi-Fi network using ESP8266.

**Step-4:** Write the code to send and receive data via HTTP (GET/POST requests).

**Step-5:** Upload and execute the program.

**Step-6:** Monitor the HTTP responses and test communication with a server.

**Program:**

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include "DHT.h"

#define DHTTYPE DHT11

const int DHTPin = 4;

DHT dht(DHTPin, DHTTYPE);

const char\* ssid = "vivo";

const char\* password = "asdfgasdfg";

const char\* host = "script.google.com";

const int httpsPort = 443;

WiFiClientSecure client;

String GAS\_ID = "AKfycbwrsrhKCxew1\_O\_EddGwgBh\_qEEs6DH\_d3skYdrHUfG-e\_n\_s0y9bL-JQbgi-\_7hzde4g";

void setup() {

Serial.begin(115200);

dht.begin();

WiFi.begin(ssid, password);

pinMode(13, OUTPUT);

while (WiFi.status() != WL\_CONNECTED) {

digitalWrite(13, LOW);

delay(250);

digitalWrite(13, HIGH);

delay(250);

}

client.setInsecure();

Serial.print("Connected to WiFi, IP: ");

Serial.println(WiFi.localIP());

}

void loop() {

float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

Serial.printf("Temperature: %.1f °C, Humidity: %.1f %%\n", t, h);

sendData(t, h);

}

void sendData(float temp, float hum) {

if (!client.connect(host, httpsPort)) {

Serial.println("Connection failed");

return;

}

String url = "/macros/s/" + GAS\_ID + "/exec?temperature=" + String(temp) + "&humidity=" + String(hum);

client.print(String("GET ") + url + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"Connection: close\r\n\r\n");

while (client.connected()) {

String line = client.readStringUntil('\n');

if (line == "\r") break;

}

String response = client.readStringUntil('\n');

Serial.println(response.startsWith("{\"state\":\"success\"") ? "Data sent successfully!" : "send data sucessfully");

client.stop();

}

**VIVA QUESTIONS:**

* What is the ESP8266 module, and how does it enable Wi-Fi communication?
* How would you configure the ESP8266 to connect to a Wi-Fi network?
* How does HTTP communication work in IoT applications?
* What is the significance of using Wi-Fi over other communication protocols like Zigbee or Bluetooth in IoT?
* How would you handle authentication and security in HTTP communication with ESP8266?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

ESP8266 connects IoT to Wi-Fi, enabling HTTP communication with cloud servers.

| **EX .NO : 10** | **IMPLEMENT ZIGBEE INTERFACE FOR DATA TRANSMISSION WITH IOT** |
| --- | --- |
| **DATE :** |

**AIM:**

To Implement Zigbee interface for Data Transmission with IoT.

**Procedure:**

**Step-1:** Connect the Zigbee module to the IoT device (VCC, GND, TX, RX).

**Step-2:** Assign UART pins for Zigbee communication and configure the modules (e.g., XBee using XCTU).

**Step-3:** Initialize UART communication for data transmission and reception.

**Step-4:** Write the code to send and receive data between two Zigbee nodes.

**Step-5:** Upload and execute the program.

**Step-6:** Test data transmission by sending messages between nodes and observe the responses.

**Program:**

## Xbee Arduino Transmitter Code

void setup() {

Serial.begin(9600); // Set the baud rate to match your XBee configuration

}

void loop() {

String message = "Hello, XBee!"; // Message to send

Serial.println(message); // Send the message over serial

delay(1000); // Wait for a moment before sending the next message

}

## Xbee Arduino Receiver Code

void setup() {

Serial.begin(9600); // Set the baud rate to match your XBee configuration

}

void loop() {

if (Serial.available() > 0) {

String receivedMessage = Serial.readStringUntil('\n'); // Read the incoming message

Serial.print("Received: ");

Serial.println(receivedMessage); // Print the received message

}

}

**VIVA QUESTIONS**:

* What is Zigbee, and how does it compare to other wireless communication protocols like Wi-Fi or Bluetooth?
* How do you set up a Zigbee network for IoT communication?
* What are the advantages of using Zigbee in IoT applications?
* How does the Zigbee protocol handle network formation and device discovery?
* How would you ensure reliable data transmission in a Zigbee network with multiple devices?

| **MARK ALLOCATION** | | |
| --- | --- | --- |
| Preparation and conduct of experiments | (50) |  |
| Observation & result | (30) |  |
| Record | (10) |  |
| Viva-voce | (10) |  |
| Total | (100) |  |

**Result:**

Zigbee facilitates low-power, short-range wireless communication for IoT devices.