4.1 Experiment No. 1

Aim: Write a program for sending alert messages to the user for controlling and interacting with your environment.

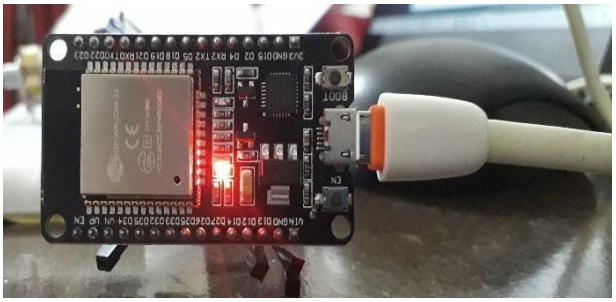
Software Requirement: Arduino IDE

Hardware Requirement: ESP-WROOM 32 board, Micro USB Data Cable, bread board, Potentiometer 10K, Male to female wires, Laptop/PC.

Theory: The program is written in the Arduino programming language and is designed to read the value of a potentiometer connected to GPIO 34 (Analog ADC1_CH6) on an Arduino board. It continuously reads the potentiometer value and prints it to the Serial Monitor. If the potentiometer value exceeds 2000, it alsoprints "warning" to the Serial Monitor.

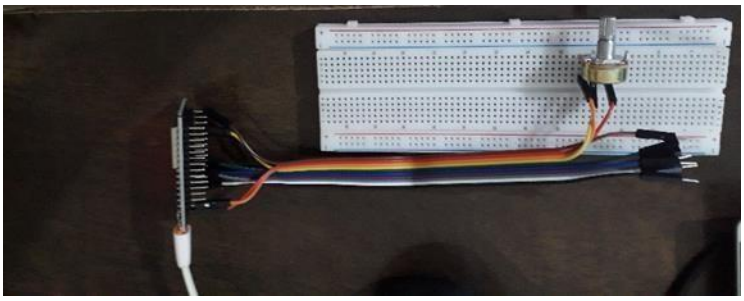
Connections:

1. First connect the data cable to the ESP 32 board, check out the notch and insertthe cable in straight manner, without any tilt.

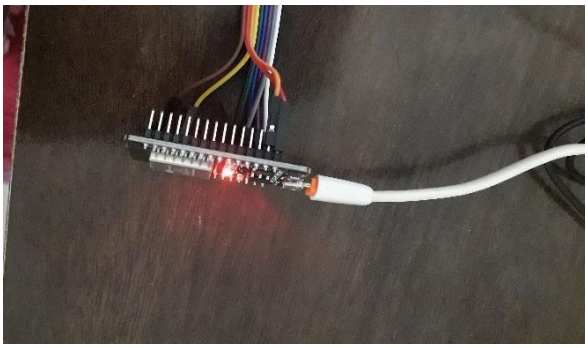


2. Connect the other end of the cable to the USB port of Laptop /PC and you shouldsee a blue light glows.

3. Connect the potentiometer to the bread board as shown. Use the male female cables to connect the potentiometer to ESP 32.



4. The middle pin (yellow) of potentiometer is connected to GPIO 34, the red pin is connected to 3.3 V and the orange pin is connected to ground.



3. Open Arduino IDE, then go to File□Examples□ESP32□Analog Read

4. Analog reading is useful to read values from variable resistors like potentiometers, or analog sensors.

4. Upload the code provided to your ESP32.

5. Before uploading the code check if the right port is selected.

6. If the port is not selected then will get an error message.

4. Then go to Tools□Ports

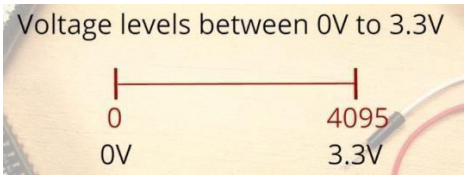
5. Install CH340g Driver from Google (exe file).

6. Now check on the port. It should show com3 or com4.Select the port.

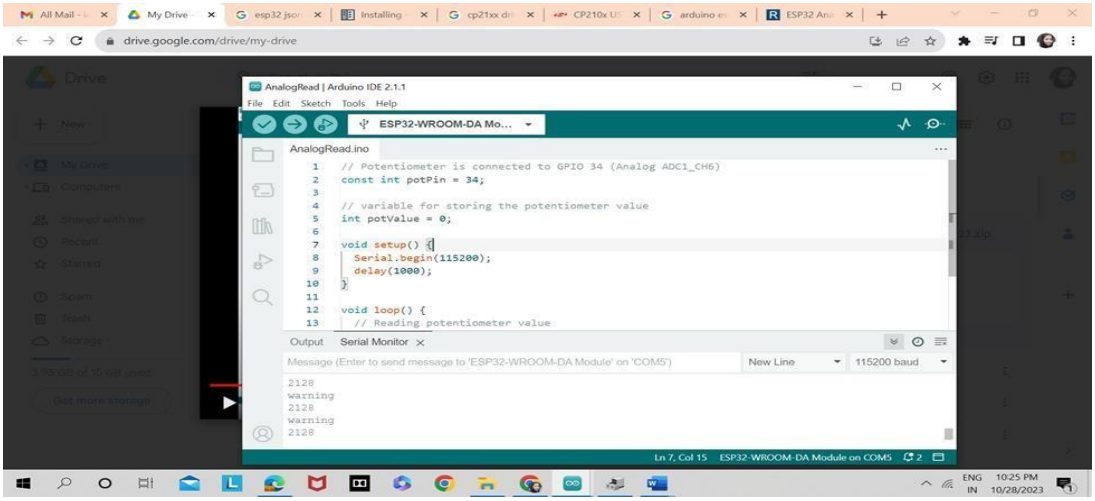
7. After uploading the code and pressing the ESP32 reset button, open the Serial Monitor at a baud rate of 115200. Rotate the potentiometer and see the values changing.

8. Read potentiometer ESP32 analogRead.

9. The maximum value you'll get is 4095 and the minimum value is 0.



10. Now rotate the potentiometer and if the potValue exceeds 2000, it issues a"warning" message and prints the value to the Serial Monitor.



Important functions and variables

1. A constant integer variable potPin is defined and assigns it the value 34. Itrepresents the GPIO pin to which the potentiometer is connected.

2. int potValue = 0 declares an integer variable potValue and initializes it to 0. Thisvariable will be used to store the value read from the potentiometer.

3. void setup ()- This is the setup function. It runs once when the Arduino board ispowered up or reset.

4. Serial.begin(115200)- Initializes the serial communication with a baud rate of115,200. This is used for printing messages to the Serial Monitor.

5. delay(1000)- A delay of 1000 milliseconds (1 second) is added to ensure that theArduino has enough time to initialize before the loop() function starts executing.

6. void loop()- This is the main loop function. It runs continuously after the setup()function is executed.

7.potValue = analogRead(potPin)-Reads the analog voltage on the potPin (GPIO 34)and stores the result in the potValue variable. The analogRead function converts the voltage on the pin to a digital value between 0 and 4095 (assuming a 12-bit ADCresolution).

8. if (potValue > 2000) {Serial.println("warning");}-Checks if the value of potValue isgreater than 2000. If it is, a "warning" message is printed to the Serial Monitor. This condition is used to detect when the potentiometer's value exceeds a certain threshold.

9. Serial.println(potValue)-Prints the current value of the potentiometer to the Serial Monitor. This line is executed regardless of the value of the potentiometer.

10. delay(500);: Adds a delay of 500 milliseconds (0.5 seconds) before the next iteration of the loop() function. This helps slow down the rate at which potentiometer values are printed to the Serial Monitor.

Conclusion: This program reads the analog value from a potentiometer, prints the value to the Serial Monitor, and issues a "warning" message if the potentiometer value is greater than 2000. It does this repeatedly in a loop with a 0.5-second delaybetween readings.

Outcomes

4.2 Experiment No. 2

Aim: Write an Arduino/ Raspberry pi program for interfacing with PIR sensor Experiment

Software Required: Tinkercad.

Theory:

PIR sensors allow you to sense motion. They are used to detect whether a human has moved in or out of the sensor range. They are commonly found in appliances and gadgets used at home or for businesses. **They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.**

Following are the advantages of PIR Sensors −

- Small in size
- Wide lens range
- Easy to interface
- Inexpensive
- Low-power
- Easy to use
- Do not wear out

Procedure:

1. Assemble all the parts.

2. Connections:

1. Connect the ground pin of PIR sensor to the ground (GND) pin of Arduino.

2. Connect the power pin to PIR sensor to the 5V pin of Arduino.

3. Connect the signal pin of Arduino to pin 2 of Arduino as shown in Fig.1.

3. Copy the code by clicking on the code button on RHS and then in Blocks button there is a drop-down menu from which select the text option, click on continue and paste the code there. Then click on start simulation button.

Important functions and variables

1. First declares two integer variables: `pirPin` and `ledPin`. These variables will be used to store the pin numbers to which the PIR sensor output and LED are connected, respectively.

2. Initialize serial communication for debugging.

3. The `setup()` function is a special function in Arduino that is executed only once when the Arduino board is powered up or reset.

4. In this setup function: `pinMode(pirPin, INPUT)` sets the `pirPin` (connected to the PIR sensor) as an input pin, indicating that it will be used to read data from the sensor and `pinMode(ledPin, OUTPUT)` sets the `ledPin` (connected to an LED) as an output pin, indicating that it will be used to control the LED.

5. `Serial.begin(9600)` initializes serial communication at a baud rate of 9600 bits per second. This is used for debugging purposes to send messages to your computer for monitoring.

6. The `loop()` function is where the main code execution takes place. It runs in a continuous loop after the `setup()` function is executed.

In this loop:

7. In this loop: `int motionState = digitalRead(pirPin)` reads the state of the PIR sensor. If the sensor detects motion, it will read `HIGH`, and if there's no motion, it will read `LOW`.

8. The code then uses an `if` statement to check whether `motionState` is `HIGH`, indicating that motion has been detected.

9. If motion is detected, `digitalWrite(ledPin, HIGH)` turns on the LED by setting the `ledPin` to `HIGH`.

10. `Serial.println("Motion detected!")` sends a message to the serial monitor indicating that motion has been detected.

11. `delay(1000)` causes the program to pause for one second before continuing. This provides a delay between LED on and off states.

12. If no motion is detected (`motionState` is `LOW`), the LED is turned off using `digitalWrite(ledPin, LOW)` and a message is printed to the serial monitor indicating that no motion was detected.

Conclusion: This code snippet demonstrates the basic operation of interfacing a PIR sensor with an Arduino. When the PIR sensor detects motion, it turns on an LED and sends a message to the serial monitor, and when no motion is detected, the LED is turned off .

4.3 Experiment No. 3

Aim: Write a Program to design and develop a user interface for monitoring and controlling CPS system

Software Requirement: Arduino IDE

Hardware Requirement: ESP-WROOM 32 board, Micro USB Data Cable, bread board, Resistor (220 Ω), Male to female wires, Laptop/PC, LED Lights, Jumper wires.

Theory:

Cyber-Physical Systems: CPS refers to systems that integrate physical processes with computer-based control and monitoring. These systems can be found in various applications, such as industrial automation, autonomous vehicles, smart cities, and more. CPS combines real-world, physical components with computational elements to make decisions and control actions.

In this practical an ESP32 LED web server is built from scratch. The following steps are performed-first setting up the ESP32, connecting the LEDs, and creating a web interface to control them.

C**onnections:**

1. First connect the data cable to the ESP 32 board, check out the notch and insert the cable in straight manner, without any tilt.

2. Connect the other end of the cable to the USB port of Laptop /PC and you should see a blue light glows.

3. In order to make our web server work:

 Connect LEDs on GPIO12 and GPIO14 on ESP32 so that you can control them

 After the coding is done. Try to access the control webpage by inputting the IP address of esp32 in the browser

 Clicking the buttons on the webpage to turn the LED “ON” or “OFF”.

Working of ESP32 LED WebServer

The program works by creating a web server on the ESP32 microcontroller that serves a web page to a client device, such as a computer or mobile phone, over Wi- Fi. The web page contains controls that allow you to turn the LED lights on or off.

The step-by-step breakdown of how the ESP32 LED webserver program works:

1. The ESP32 microcontroller is connected to the LED lights and programmed to serve as a web server.
2. The web server code is uploaded to the ESP32 using the Arduino IDE or other programming tools.
3. The ESP32 is connected to a Wi-Fi network, allowing it to communicate with client devices over the network.
4. When a client device connects to the ESP32's IP address, the web server code serves a web page to the client device's web browser.
5. The web page contains controls for the LED lights, allowing the user to turn them on or off.
6. When the user interacts with the controls on the web page, the ESP32 microcontroller receives the command and adjusts the LED lights accordingly.
7. The ESP32 sends a response back to the client device, updating the web page with the current state of the LED lights.

Circuit Diagram of ESP32 LED WebServer

Below is the circuit diagram to control an **LED using ESP32 based webserver**: The ESP32 microcontroller has many GPIO pins that can be used for interfacing with external devices, such as LED lights. In this example, we will interface the ESP32 with two LED lights connected to GPIO pins 12 and 14.

To interface the ESP32 with two LED lights:

 Connect one end of a 220-ohm resistor to GPIO pin 12 on the ESP32, and connect the other end of the resistor to the positive (anode) leg of one LED light.

 Connect the negative (cathode) leg of the LED light to the ground (GND) on the ESP32.

 Repeat steps 1 and 2 for GPIO pin 14 and the other LED light.

Uploading the code on the ESP32 board:

To upload code to an ESP32 board, you'll need to follow these steps:

1. Connect your ESP32 board to your computer using a USB cable.

2. Open the Arduino IDE and go to "Tools" > "Board" and select your ESP32 board from the list.

3. Make sure to also select the correct port under "Tools" > "Port".

4. Write or copy your code into the Arduino IDE.

5. Verify that your code compiles by clicking the "Verify" button (checkmark icon) in the top-left corner of the IDE. If there are any errors, fix them before proceeding.

6. Upload your code to the ESP32 board by clicking the "Upload" button (right arrow icon) in the top-left corner of the IDE. The IDE will compile your code and then upload it to the ESP32 board. The progress of the upload will be shown in the bottom of the IDE.

7. Once the upload is complete, the ESP32 board will automatically reset and start running your code.

Finding the IP address of the ESP32 board

1. Open the Serial Monitor in the Arduino IDE by clicking on the magnifying glass icon in the top right corner of the IDE window.

2. Select a baud rate of 115200 from the dropdown list in the bottom right corner of the Serial Monitor window.

3. Reset the ESP32 board by pressing the EN(RST) button on the board.

4. Wait for the ESP32 board to connect to the WiFi network. The Serial Monitor will display messages that indicate the progress of the connection.

5. Once the ESP32 board is connected to the WiFi network, it will obtain an IP address from the network. The Serial Monitor will display the IP address of the ESP32 board.

Connecting to the ESP32 Webserver and testing it

1. Open the same network through a device or use the same device to whose hotspot your ESP32 is connected

2. Open a web browser on your computer or mobile device and enter the IP address of the ESP32 board into the address bar.

3. After you've accessed the control page on your browser you can test it by clicking "ON" and "OFF" and checking the LED state and serial monitor simultaneously

4. Now if you click on the GPIO 12 button, you will see that the serial monitor gets a request on 12/on URL and then your led lights up.

5. Now if you click on the GPIO 12 button, you will see that the serial monitor gets a request on 12/on URL and then your led lights up.

6. After this is done, the Esp32 updates its state on the webpage as “ON” or “OFF”.

7. This will work the same for GPIO 14 too.

Conclusion: Thus the experiment is performed by writing a program to design and develop a user interface for monitoring and controlling CPS system.

Outcomes

4.4. Experiment No. 4

Aim: Write a program for sending sensor data to the cloud and displaying on web page.

Software Requirement: Arduino IDE

Hardware Requirement: ESP-WROOM 32 board, Micro USB Data Cable, bread board, Potentiometer 10K, Male to female wires, Laptop/PC.

Theory: The program is for an ESP32-based microcontroller and is designed to create a simple web server. It reads the value from a potentiometer and displays that value on a webpage hosted by the ESP32.

Connections:

1. First connect the data cable to the ESP 32 board, check out the notch and insert the cable in straight manner, without any tilt.
2. Connect the other end of the cable to the USB port of Laptop /PC and you should see a blue light glows.

3. Connect the potentiometer to the bread board as shown. Use the male female cables to connect the potentiometer to ESP 32.

4. The middle pin (yellow) of potentiometer is connected to GPIO 34, the red pin is connected to 3.3 V and the orange pin is connected to ground.

3.Open Arduino IDE, then go to File□New Sketch 4.And copy the given code in new sketch and upload it.

5. Make sure in Tools□Manage libraries□Library manager□type WebServer, and search for WebServer_ESP32_ENC library and install it.

6. When you run this code on your ESP32, it sets up a web server that you can access from a web browser.

7. Visiting the ESP32's IP address will display an HTML page showing the current potentiometer value.

8. For example, Potentiometer value:2233.

Procedure:

Explanation of code:

1. Include Libraries:

□ **#include <WiFi.h>:** This library allows you to use the ESP32's Wi-Fi functionality.

□ **#include <WebServer.h>:** This library enables you to create a web server on the ESP32.

2. Global Variables:

 `ssid` and `password`: These variables store the SSID (network name) and password of your Wi-Fi network.

 `potPin`: This variable stores the GPIO pin number to which the potentiometer is connected (pin 34 in this case).

 `potValue`: This variable is used to store the potentiometer reading.

3. WebServer Object:

 `WebServer server(80)`: An instance of the `WebServer` class is created on port 80. This server will handle incoming HTTP requests.

4. `handleRoot()` Function:

 This function is a callback that gets executed when someone accesses the root ("/") URL of the web server.

 It generates an HTML page with a header and displays the current potentiometer value on the webpage.

5. `setup()` Function:

 `Serial.begin(115200)`: Initializes serial communication for debugging purposes.

 `pinMode(potPin, INPUT)`: Sets the `potPin` as an input to read the potentiometer value.

Connecting to Wi-Fi:

 `WiFi.begin(ssid, password) `: Initiates a connection to the Wi-Fi network using the SSID and password.

 A `while` loop waits until the ESP32 successfully connects to the Wi-Fi network (`WL_CONNECTED` status).

 After successful connection, it prints the ESP32's local IP address to the serial monitor.

Setting up the root URL handler:

 `server.on("/", HTTP_GET, handleRoot)`: Configures the web server to call the `handleRoot()` function when a client requests the root URL ("/") using the HTTP GET method.

Starting the web server:

 `server.begin()`: Begins listening for incoming HTTP requests.

6. `loop()` Function:

- **`analogRead(potPin)`:** Reads the analog voltage at the `potPin` and stores it in the `potValue` variable.
- **`server.handleClient()`:** Handles incoming HTTP requests, including the root ("/") URL request, by calling the `handleRoot()` function.
- **`delay(100)`:** Introduces a short delay in the loop to avoid excessive server processing. Adjust the delay as needed.

Conclusion: Thus we have performed this experiment by using the ESP32 as a web server and reading an analog sensor's value to display on a webpage.

4.5. Experiment No. 5

Aim: Write a program for performing industrial data analysis using relevant tools and techniques.

Software Required: Excel

Theory:

The Internet of Things (IoT) is a network of interconnected devices & gadgets that can collect & share data by themself. IoT data analytics refers to the procedure of gathering, examining, and deciphering data produced by these devices to gain knowledge and make wise decisions. Data analytics uses bunches of hardware, software, and data science techniques to collect accurate information from massive data created by IoT devices.

Components of IoT Data Analytics

IoT data analytics involves four main components −

 Data Collection − IoT devices are embedded with various sensors that collect data on different parameters such as temperature, humidity, pressure, and motion. This data is transmitted to a central server or cloud platform for further processing.

 Data Storage − The data generated by IoT devices is massive and needs to be stored efficiently.

 Data Processing − IoT data analytics involves processing data to extract valuable insights. To make sure the data is correct, consistent, and prepared for analysis, data processing procedures including data cleansing, data transformation & data normalization are utilized.

 Data analysis − To find patterns and trends in the data, statistical & machine learning algorithms are employed.

 Data Visualization − IoT data analytics involves the use of data visualization tools to present insights and findings in a user-friendly and understandable format. Visualization tools like dashboards, charts & graphs help to understand the data quickly and then make decisions in a very logical and practical way. So, they can give an informed decision based on the insights derived from IoT data analysis.

Procedure:

1. Download the IOT data from Kaggle.

2. For our practical we have downloaded the IOT data “Farm Temperature and Humidity” from Kaggle which is an 86 KB file in excel sheet.

3. The dataset consists of Temperature and Humidity reading which was captured using Moisture and humidity sensors for 3 months in 4 farms which were 1 km apart and the readings were taken after every 1 hr.

4. So first we will select the 1st column which shows temperature readings and we will consider 1-day readings from 12 am to 12 pm i.e., 24 hrs.

5. Then click on Insert and click on Recommended Graphs and select any graph which will show the graph.

6. For longer cycle we can take 2 days data so select 48 hrs reading.

7. Then again click on Insert and click on Recommended Graphs and select any graph which will show the graph.

8. In the same way follow the same procedure for Humidity readings, which is the 2nd column and like wise you can try for next columns and so on.

Results

1. From the 24 hrs reading (1 day), from the graph we can infer that it shows a peak in temperature at 12 o'clock, which indicates the temperature is maximum. And then it comes down and at night the temperature falls.

2. For the 48 hrs reading (2 day), from the graph we can infer that it shows a peak in temperature at 12 o'clock, which indicates the temperature is maximum. And then the temperature falls, it shows a minimum temperature at 24 i.e., at 12 pm which indicates that at night the temperature falls.

4. On the 2nd day the peak is at 36 (24+12), i.e., at 12 again the temperature shows maximum value.

5. Since we stay near the equator this is the pattern we get, but the temperature might vary from location to location.

6. The Humidity data shows how much humid the weather is.

7. We can select the scatter plots to get a clear understanding of the graph. 8.The scatter graph for 1 day reading shows that at night the humidity increases.

9. For 2 days, humidity graph follows a cyclic pattern.

10. We can see this interesting fluctuation around 13 i.e., at 1 pm Humidity falls and for the next day at 37 (24+13) i.e., at 1 pm it shows similar fall in humidity.

11. So in this way we can infer a lot of things from graph.

12. So students can visualize the graph and check out what they can infer from the graphs and write them as observations.

Conclusion: In this way we have performed IOT data analysis of Farm Temperature and Humidity using Excel tools and techniques.