جامعة برج العرب التكنولوجية
BORG AL ARAB TECHNOLOGICAL UNIVERSITY

BATU Arduino Training Bag

Arduino Educational Trainer Catalog

"Shape the future of electronics by sparking innovation today."

Under Supervision: Prof. Osama Elnahas
Co-Supervision: Prof. Eman Shawky

Information Technology
Department Labs
معامل قسم تكنولوجيا المعلومات
حلـــول ذكـــية مـن أجـل غدٍ أكثر إشراقًا
Smart Solutions for a Brighter Tomorrow

برعاية

أ.د /محمد مرسي الجوهري
رئيس جامعة برج العرب التكنولوجية

أ.د/علاء عرفة
عميد كلية تكنولوجيا الصناعة والطاقة

أ.م.د /أسامة النحاس
رئيس قسم تكنولوجيا المعلومات

Information Technology Department Labs

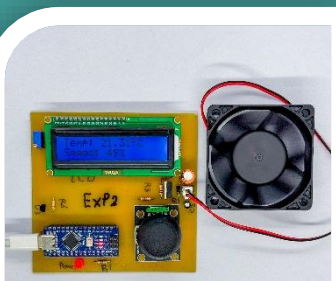
معامل قسم تكنولوجيا المعلومات

حلـــول ذكـــية مـن أجـل غدٍ أكثر إشراقًا

Smart Solutions for a Brighter Tomorrow

Welcome to our world of future of electronics

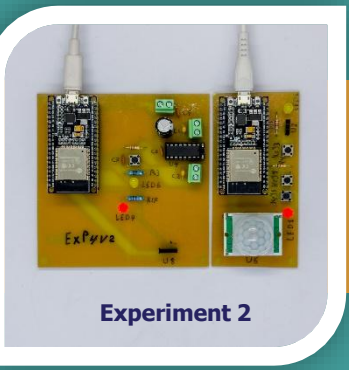
The educational experiment kit offers a comprehensive set of practical projects to learn modern technologies. The experiments include the LoRa WAN system for long-range communication, ESP32 MQTT for smart irrigation, door lock control using RFID, DC motor control via Wi-Fi, and fan control using a temperature sensor and joystick with speed displayed on an LCD. This kit is an ideal tool for students and enthusiasts to understand smart applications practically and effectively.



Experiment 1

Fan control with temperature sensor and joystick and show speed on LCD

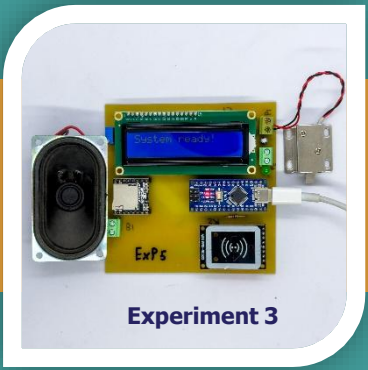
The project controls fan speed via a temperature sensor and joystick, with PWM adjustment. An LCD displays speed for easy monitoring and control.



Experiment 2

DC Motor With Wireless Control

A wireless motor control system using an nRF24L01 module, with real-time status updates on an OLED screen for enhanced usability and monitoring.



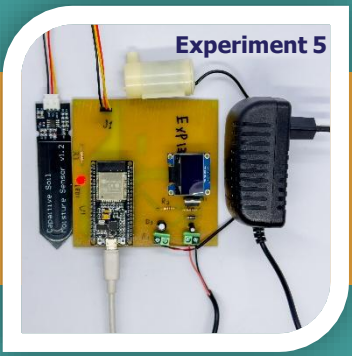
Experiment 3

Door Lock Control

An rfid-based locking system unlocks with authorized cards. A red LED shows power status. Suitable for homes, offices, and secure areas.

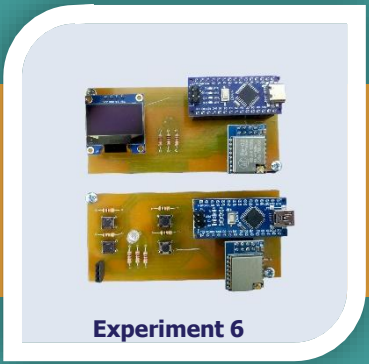
Water level measurement with ultrasonic

An ultrasonic system measures and displays water levels via a microcontroller and LCD. It's ideal for homes, agriculture, and industrial water management.



Smart irrigation system

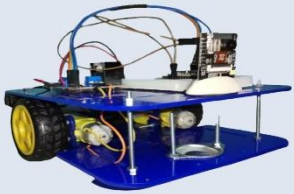
An ESP32 MQTT system automates irrigation using soil moisture sensors. It controls watering via a pump and provides smart, efficient plant care management.



Lora wan

Lora WAN is a low-power iot protocol enabling long-range, cost-effective communication using lora modulation for efficient wide-area data transmission.

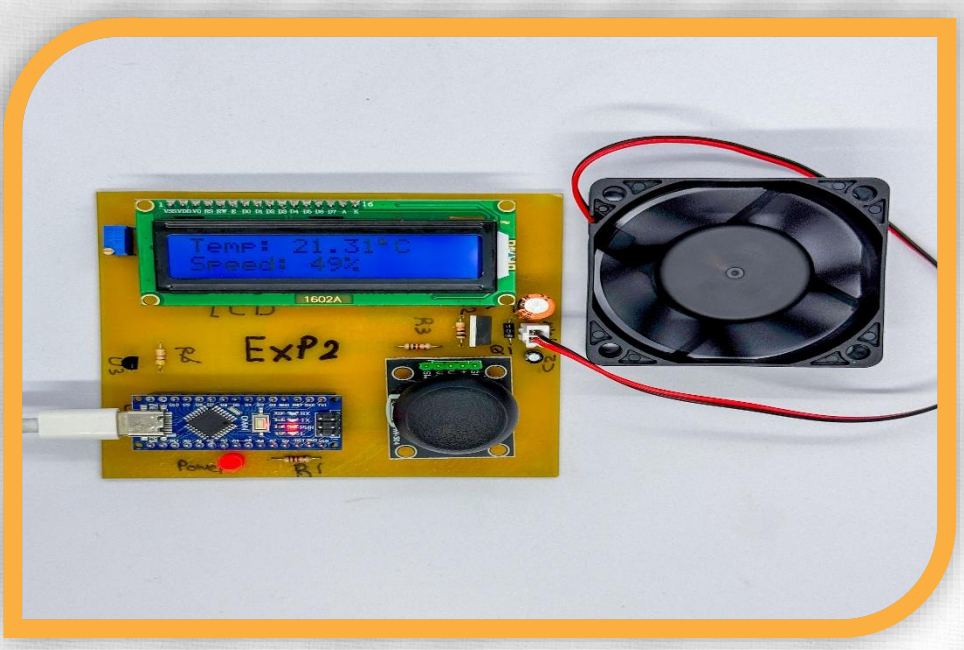
Experiment 7



ESP32-CAM Streaming and Remote Control

The car is designed to be controlled remotely via a controller, allowing for navigation and operation. The ESP32-CAM streams live video to a web application, enabling users to see what the camera captures in real-time. This integration of components facilitates an interactive and engaging experience, combining mobility with visual feedback.

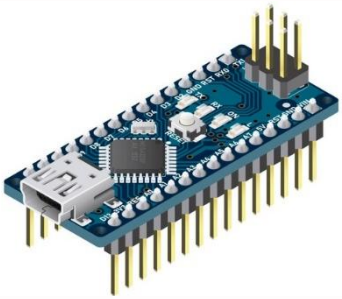
Experiment 1



Fan control with temperature sensor and joystick and show speed on LCD

The project aims to control the fan speed using a temperature sensor to measure the temperature and a joystick to manually adjust the speed.The microcontroller (suchasarduino)reads the temperature from the sensor, and when the temperature increases, the fan speed is automatically adjusted using PWM. The user can also manually change the speed via the joystick. The LCD screen displays the current fan speed, making it easy to monitor and control the system flexibly.

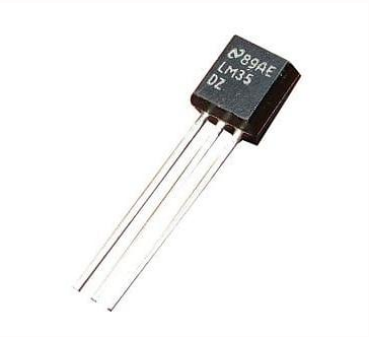
I. Components



1x arduino nano

It is a small microcontroller used as the "brain" of the project to control other components.

1



1x temperature sensor

The temperature sensor measures the surrounding temperature or the temperature of a specific object

2

1x fan

Its function is cooling or moving air to reduce temperature or improve ventilation

3

I. Components

1x joystick

It is an input device used for manual control of any system

4

1x LCD

Its function is to visually display information, such as measured values (temperature), speeds, or any textual messages.

5

1x driver

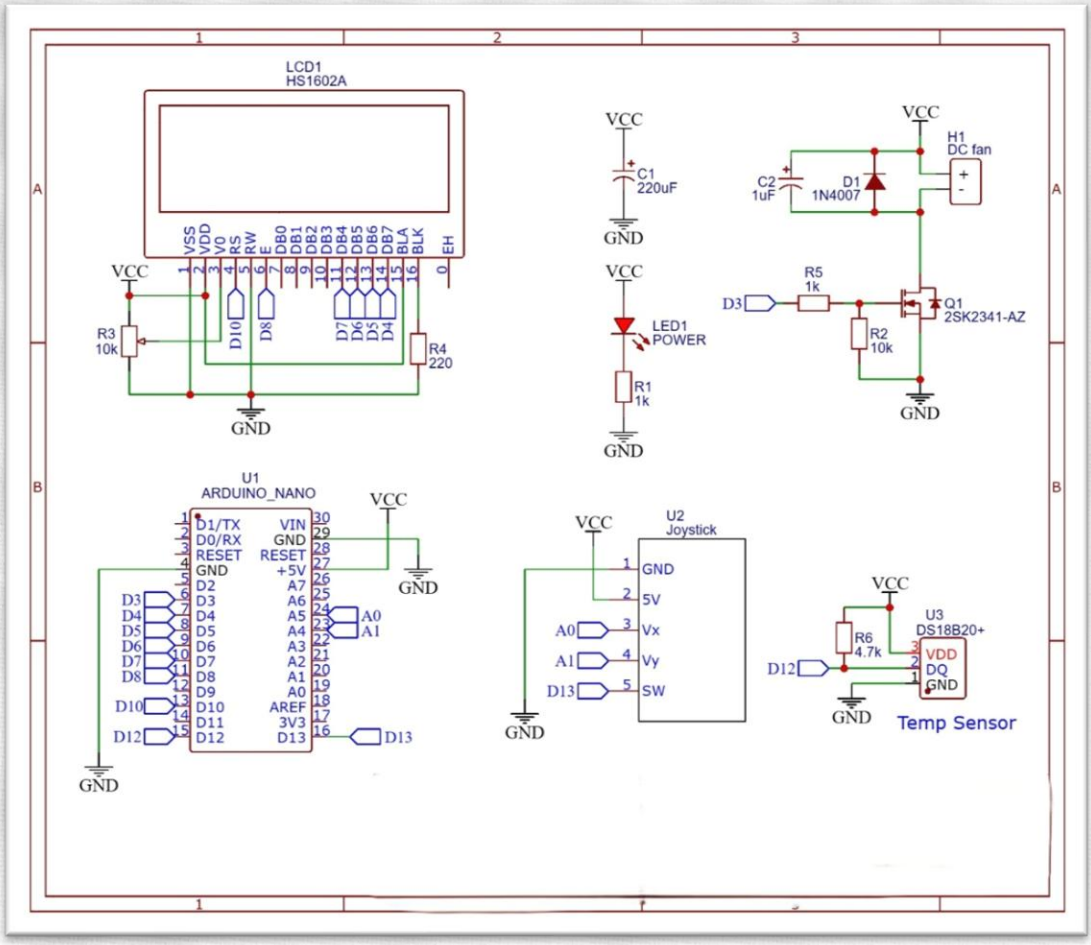
Controlling the operation of motors, stopping them, changing their speed, or altering their rotation direction.

6

Fan control with temperature sensor and joystick and show speed on LCD

Reference	Component	Value	Quantity
U1	Arduino Nano	-	1
LCD1	LCD Display	HS1602A	1
U2	Joystick Module	-	1
U3	Temperature Sensor	DS18B20+	1
H1	DC Fan	-	1
D3, D1	Diode/LED	LED-TH-5mm_R, 1N4007	2
Q1	MOSFET	2SK2341-AZ	1
R1, R5	Resistor	1kΩ	2
R2, R3	Resistor	10kΩ	2
R4	Resistor	220Ω	1
R6	Resistor	4.7kΩ	1
C1, C2	Capacitor	220µF, 1µF	2

II Schematic



Experiment 1

Step 1

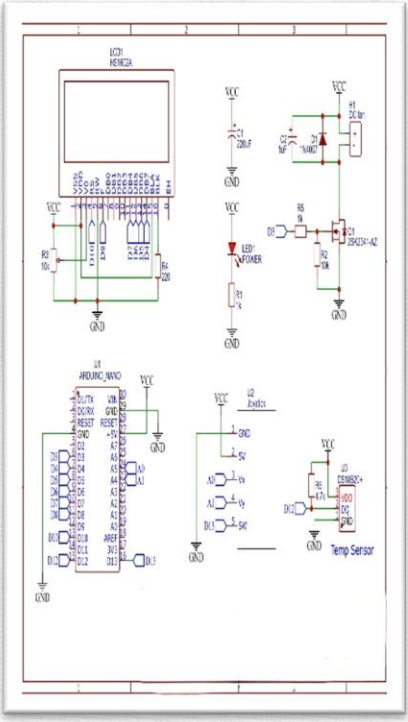
Connect the Arduino Nano

- Plug the Arduino Nano into your computer using a USB cable.
- Open the Arduino IDE on your computer to begin writing the code.

Step 2

Connect the Components

- Wire the LCD, fan, joystick, and temperature sensor as per the schematic.
- Ensure all connections are correct and secure.



Experiment 1

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>

// Define LCD pins
const int RS = 10, EN = 8, DB4 = 7, DB5 = 6, DB6 = 5, DB7 = 4;
LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);

// Define DS18B20 temperature sensor pins
#define ONE_WIRE_BUS 12
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Define joystick, fan, and LED pins
const int joystickY = A4;   // Y-axis of the joystick
const int fanPWM = 3;       // PWM pin for the fan (IRFZ44N)
const int ledPin = 2;      // Red LED pin

// System variables
int fanSpeed = 0;
float temperature = 0;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);

  // Initialize temperature sensor
  sensors.begin();

  // Set output pins
  pinMode(fanPWM, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // Startup message
  lcd.print("System Ready");
  delay(1000);
  lcd.clear();
}
```

Step 3

Write the Code

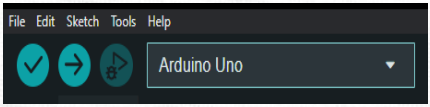
Use the Arduino IDE to write a program that:

- Reads temperature data from the DS18B20.
- Displays the temperature on the LCD.
- Controls the fan speed using the joystick input.

Step 4

Compile and Upload the Code

- Upload the program to the Arduino Nano.
- Verify that the data is displayed correctly on the LCD.



Experiment 1

Step 5

Assemble the PCB

- After testing the circuit, mount all components onto the PCB.
- Secure the LCD, joystick, and other components in their designated positions.

Step 6

Test Your Code

- Power the circuit and verify its functionality.
- Check the temperature readings on the LCD.
- Adjust the joystick to control the fan speed.

Step 7

Troubleshooting

- If the system doesn't work as expected:
  - Double-check all wiring connections.
  - Re-upload the code to the Arduino.
  - Ensure the power supply is sufficient for all components.

```
void loop() {
  // Read temperature
  sensors.requestTemperatures();
  temperature = sensors.getTempCByIndex(0);

  // Read joystick value and map it to fan speed (0-255)
  int joyValue = analogRead(joystickY);
  fanSpeed = map(joyValue, 0, 1023, 0, 255);

  // Apply minimum fan speed (50/255) to prevent it from stopping
  fanSpeed = constrain(fanSpeed, 50, 255);

  // Control fan and LED
  analogWrite(fanPWM, fanSpeed);
  digitalWrite(ledPin, fanSpeed > 0 ? HIGH : LOW);

  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print((char)223);  // Celsius degree symbol
  lcd.print("C  ");

  lcd.setCursor(0, 1);
  lcd.print("Speed: ");
  lcd.print(map(fanSpeed, 0, 255, 0, 100)); // Convert speed to percentage
  lcd.print("%   ");

  delay(100); // Short delay to improve responsiveness
}
```

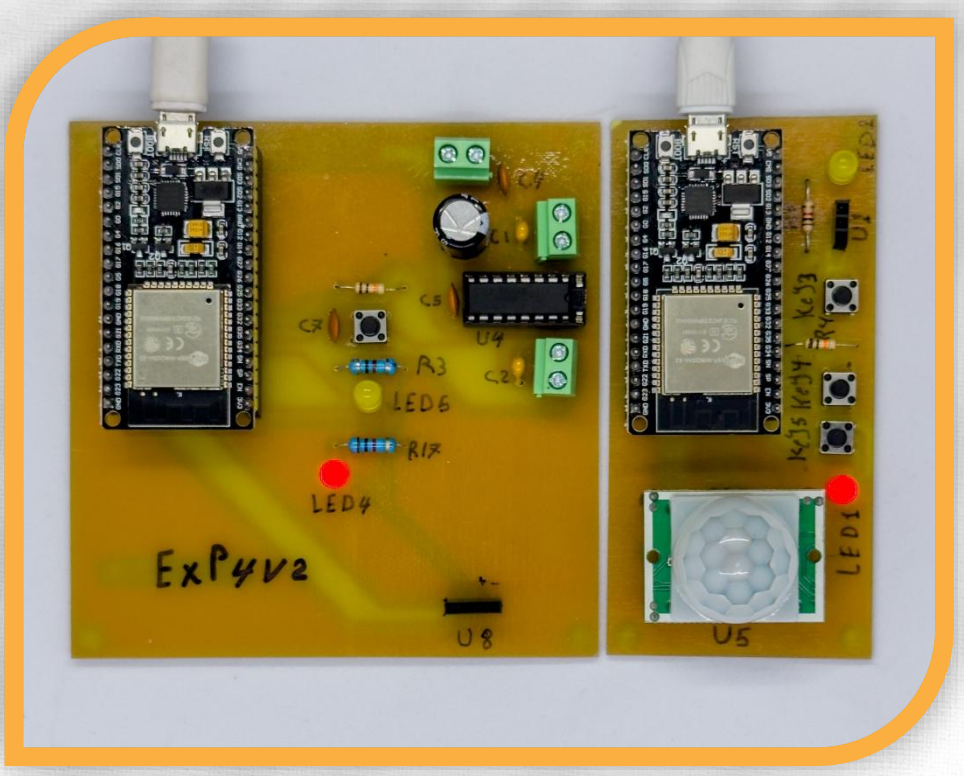
IV. Code

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>

// Define LCD pins
const int RS = 10, EN = 8, DB4 = 7, DB5 = 6, DB6 = 5, DB7 = 4;
LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);

// Define DS18B20 temperature sensor pins
#define ONE_WIRE_BUS 12
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Define joystick, fan, and LED pins
const int joystickY = A4;   // Y-axis of the joystick
const int fanPWM = 3;       // PWM pin for the fan (IRFZ44N)
const int ledPin = 2;      // Red LED pin

// System variables
int fanSpeed = 0;
float temperature = 0;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);

  // Initialize temperature sensor
  sensors.begin();

  // Set output pins
  pinMode(fanPWM, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // Startup message
  lcd.print("System Ready");
  delay(1000);
  lcd.clear();
}
```

IV. Code

```
void loop() {
  // Read temperature
  sensors.requestTemperatures();
  temperature = sensors.getTempCByIndex(0);

  // Read joystick value and map it to fan speed (0-255)
  int joyValue = analogRead(joystickY);
  fanSpeed = map(joyValue, 0, 1023, 0, 255);

  // Apply minimum fan speed (50/255) to prevent it from stopping
  fanSpeed = constrain(fanSpeed, 50, 255);

  // Control fan and LED
  analogWrite(fanPWM, fanSpeed);
  digitalWrite(ledPin, fanSpeed > 0 ? HIGH : LOW);

  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print((char)223);  // Celsius degree symbol
  lcd.print("C  ");

  lcd.setCursor(0, 1);
  lcd.print("Speed: ");
  lcd.print(map(fanSpeed, 0, 255, 0, 100)); // Convert speed to percentage
  lcd.print("%   ");

  delay(100); // Short delay to improve responsiveness
}
```

Experiment 2



DC Motor With Wireless Control

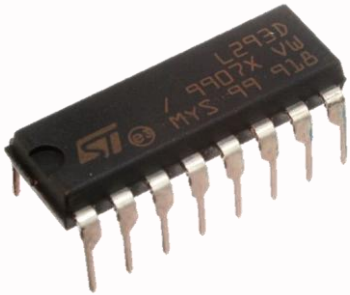
This Project Demonstrates A Wireless Control System Designed To Control Motor Directions Using An Nrf24l01 Wireless Module, With Feedback Displayed On An OLED Screen. The System Receives Commands Wirelessly And Interprets Them To Perform Specific Actions, Such As Moving Forward, Backward, Or Stopping. The OLED Display Provides Real-time Status Updates For Better Usability And Monitoring.

I. Components

1x OLED SSD1306 0.96in

A 0.96-inch OLED display module for text and graphics, supporting I2C and SPI interfaces, ideal for low-power applications.

1



1x L293D Quadruple Half-H Motor Driver IC

A dual-channel motor driver IC for controlling the direction and speed of two DC motors, ideal for robotics and motorized projects.

2

2x Ceramic Capacitor 220nF 50V

High-quality 50V capacitors for decoupling, noise suppression, and energy storage in electronic circuits.

3

I. Components

1x Capacitor 1000uF

A high-capacity electrolytic capacitor for power smoothing, filtering, or energy storage in power supply circuits. Rated at 16V.

4

3x MLCC – SMD/SMT 100nF

Surface-mount ceramic capacitors (Multilayer Ceramic Chip) used for decoupling and noise filtering in compact PCB designs.

5

3x Ceramic Capacitor 100nF 50V

Standard ceramic capacitors with 100nF capacitance, perfect for decoupling and general-purpose applications. Rated for 50V.

6

I. Components

3x Terminal Block 2 Pins KF126-5.0-2P

Screw terminal connectors for easily securing wires to a circuit board. Widely used in power supply or industrial applications.

4

4x Mini Push Button

Small tactile switches designed for user input in electronics. Commonly used for reset buttons or user interfaces.

5

4x Resistor 10k

Fixed-value resistors with a resistance of 10 kilo-ohms. Used for voltage division, current limiting, and pull-up or pull-down applications.

6

I. Components

4x Resistor 1k

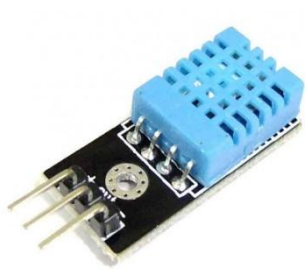
Fixed-value resistors with a resistance of 1 kilo-ohm. Versatile and widely used in circuits for current regulation and signal conditioning.

7

1x PIR HC-SR501

A Passive Infrared (PIR) motion sensor for detecting movement. Commonly used in security systems, lighting automation, and presence detection.

8



1x DHT11

A digital sensor for measuring temperature and humidity. Easy to use with microcontrollers like Arduino or Raspberry Pi.

9

I. Components

1x Base IC 18 Pins

A socket designed to hold 18-pin integrated circuits, allowing easy replacement or mounting on a circuit board.

10

3x LED-TH-5mm (2 red, 1 blue, 1 green)

Standard 5mm through-hole LEDs in red, blue, and green colors. Useful for indicating statuses or adding visual effects in projects.

11

2x DC Gearbox Motor

Compact DC motors with built-in gearboxes for increased torque and reduced speed. Ideal for robotic and mechanical applications.

12

I. Components

3x Pin Headers Female 2.54mm: 40-Pin

Female pin header connectors with a 2.54mm pitch. Often used for connecting modules, boards, or components in electronics.

13

2x nRF24L01+ Wireless Module

The nRF24L01+ is a low-power 2.4 GHz RF transceiver module, ideal for IoT, wireless sensor networks, remote controls, and embedded systems.

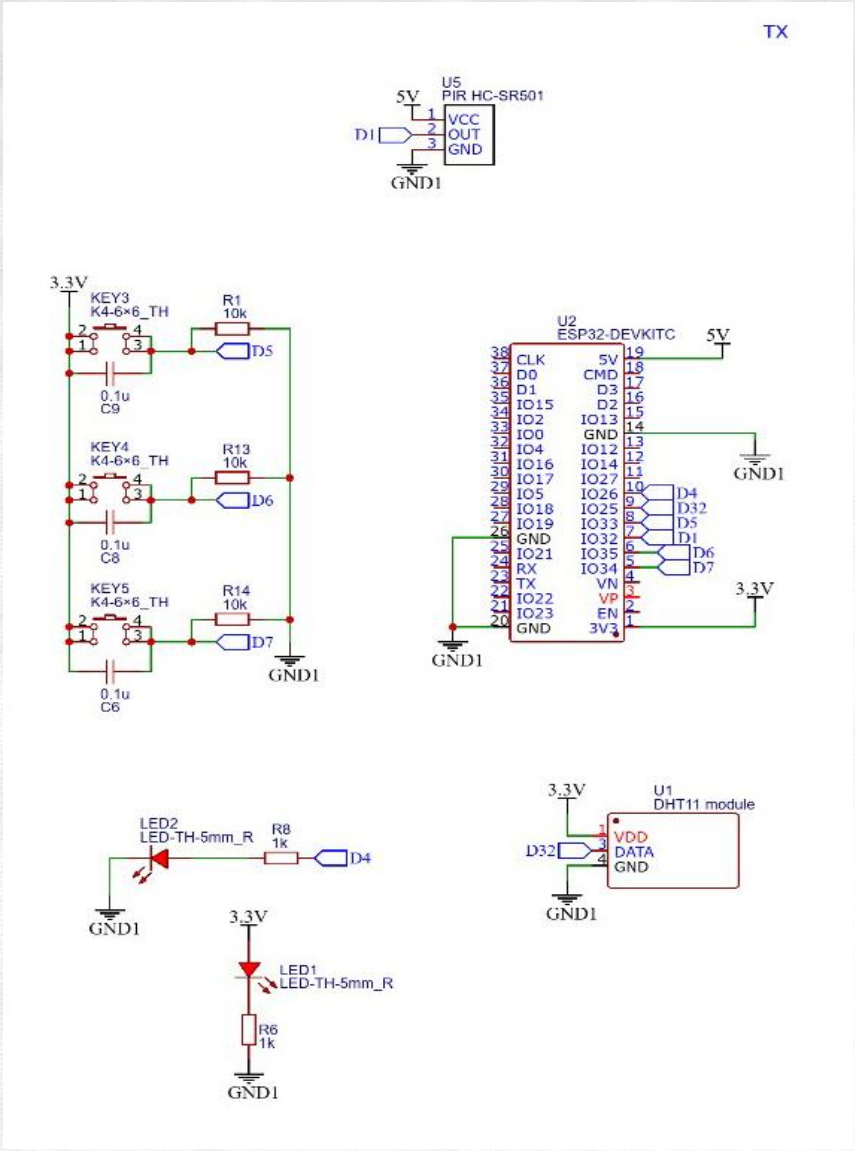
14

DC Motor with wireless control

Reference	Component	Value	Quantity
U2, U3	ESP32-DEVKITC	-	2
U5	HC-SR501 PIR Sensor	-	1
LED1, LED2, LED3, LED4	LED (Red)	-	4
U1	DHT11	-	1
U4	L293D Motor Driver	-	1
KEY2, KEY3, KEY4, KEY5	Push Button	-	4
R1, R13, R14, R3, R17, R5, R6, R8	Resistor	Mixed (1kΩ, 10kΩ, etc.)	8
C1, C2, C3, C4, C6, C7	Capacitor	Mixed (0.1µF, 100µF, etc.)	6
U8	OLED Display	-	1
P2, P3	Motor Power In	-	1
U2, U3	ESP32-DEVKITC	-	2
U5	HC-SR501 PIR Sensor	-	1

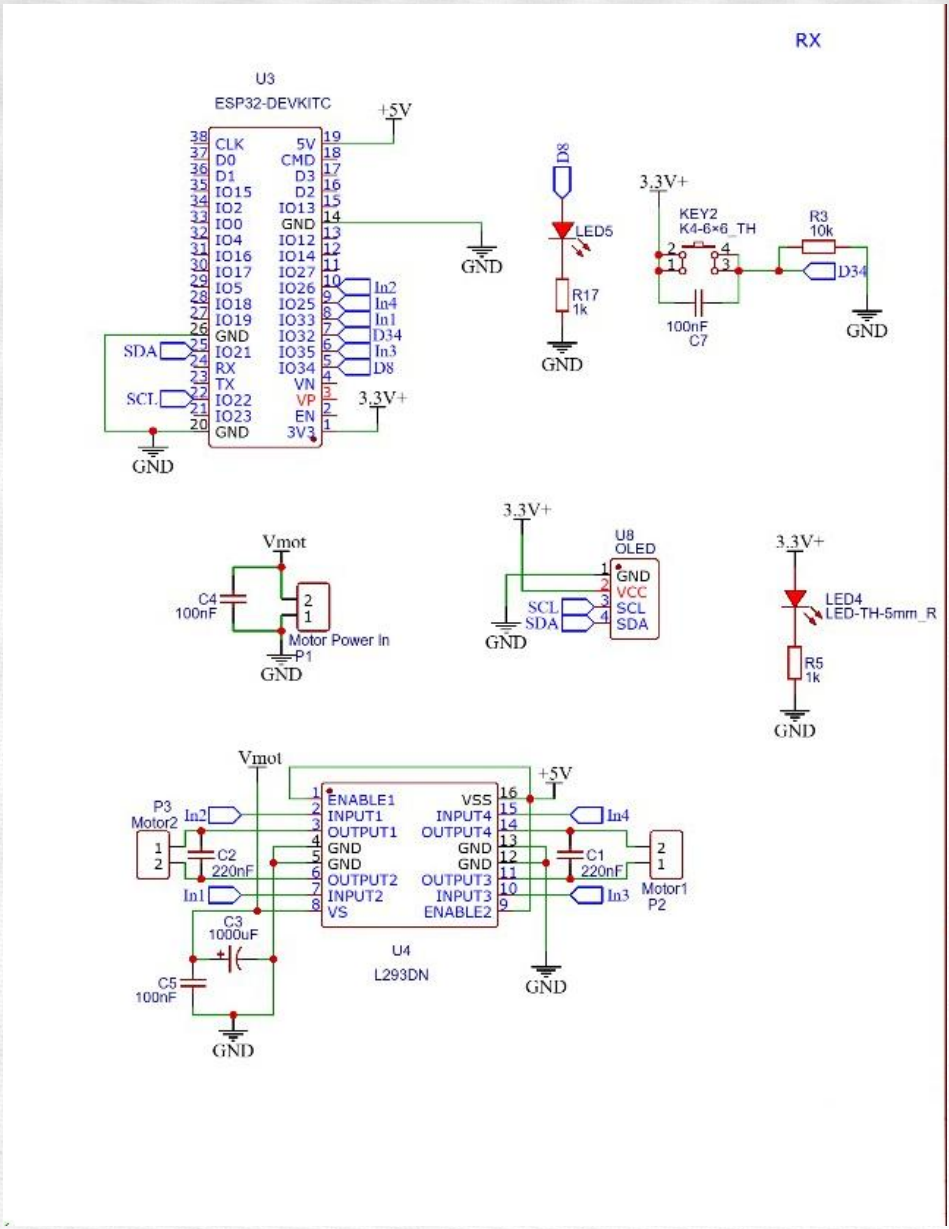
II. Schematic

DC Motor with wireless control



II. Schematic

DC Motor with wireless control



III. Experiment

Step 1

Connect the ESP32 Development Board

1. Connect the ESP32-DEVKITC board to your computer using a USB cable.
2. Open Arduino IDE on your computer to write the code.
3. Make sure the ESP32 board drivers are installed on Arduino IDE.

Step 2

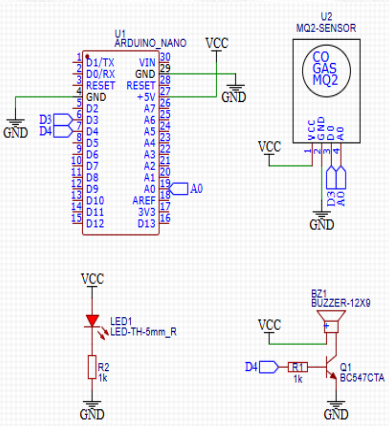
Connect the Sensors

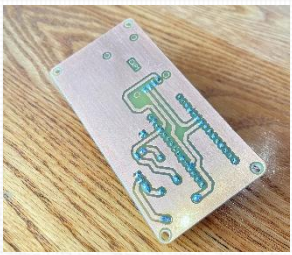
- PIR Sensor (U5):

1. Connect (**VCC**) to 5V.
2. Connect (**GND**) to GND.
3. Connect (**OUT**)to D1.

- DHT11 Sensor (U1):

1. Connect (**DATA)** to D32.
2. Connect (**VCC)** to 3.3V.
3. Connect (**GND**)to GND.





III. Experiment

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN pins for nRF24L01
const byte address[6] = "00001"; // Communication address

// Define button and LED pins
#define KEY1 2
#define KEY2 3
#define KEY3 4
#define LED_PIN 5

void setup() {
  // Set button pins as input with pull-up resistors
  pinMode(KEY1, INPUT_PULLUP);
  pinMode(KEY2, INPUT_PULLUP);
  pinMode(KEY3, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);

  // Initialize the radio
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN); // Set power level to minimum
  radio.stopListening(); // Set as transmitter
}
```

Step 3

Install Required Libraries

Install the following libraries in the Arduino IDE via Library Manager:

RF24 (for nRF24L01 communication)

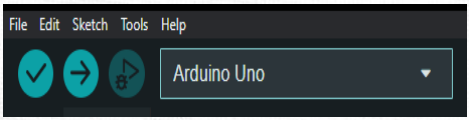
Adafruit SSD1306 (for the OLED display)

Adafruit GFX (required by the SSD1306 library).

Step 4

.Upload the Code

- Copy the provided code into the Arduino IDE.
- Connect the Arduino to your computer via USB.
- Select the appropriate Board and Port from the Tools menu.
- Upload the sketch to the Arduino.



III. Experiment

Step 5

Prepare the Transmitter

- Ensure the transmitter is programmed to send commands ('F', 'B', 'S') to the same nRF24L01 address used in the receiver code.
- Use a second Arduino or microcontroller for the transmitter setup.

Step 6

Power Up and Test the System

- Power the receiver system (Arduino, nRF24L01, OLED, and motor driver).

Power the transmitter system.

Use the transmitter to send commands:

'F' for forward motion.

'B' for backward motion.

'S' to stop the motors.

Observe the OLED display for feedback on the received command and the motor's current action.

IV. Code Receiver

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>

RF24 radio(7, 8); // CE, CSN pins for nRF24L01
const byte address[6] = "00001"; // Same address as transmitter

// Define motor driver pins
#define IN1 10
#define IN2 9
#define IN3 6
#define IN4 5

// Define OLED display
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  // Initialize motor driver pins
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  stopMotor(); // Stop motor initially

  // Initialize OLED display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("System Ready!");
  display.display();

  // Initialize the radio
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN); // Set power level to minimum
  radio.startListening(); // Set as receiver
}
```

IV. Code Receiver

```
void loop() {
  if (radio.available()) {
    char command;
    radio.read(&command, sizeof(command));

    display.clearDisplay();
    display.setCursor(0, 0);

    // Execute commands based on received data
    switch (command) {
      case 'F':
        forward();
        display.println("Moving Forward");
        break;
      case 'B':
        backward();
        display.println("Moving Backward");
        break;
      case 'S':
        stopMotor();
        display.println("Stopped");
        break;
    }
    display.display();
  }
}

// Motor control functions
void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

void stopMotor() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

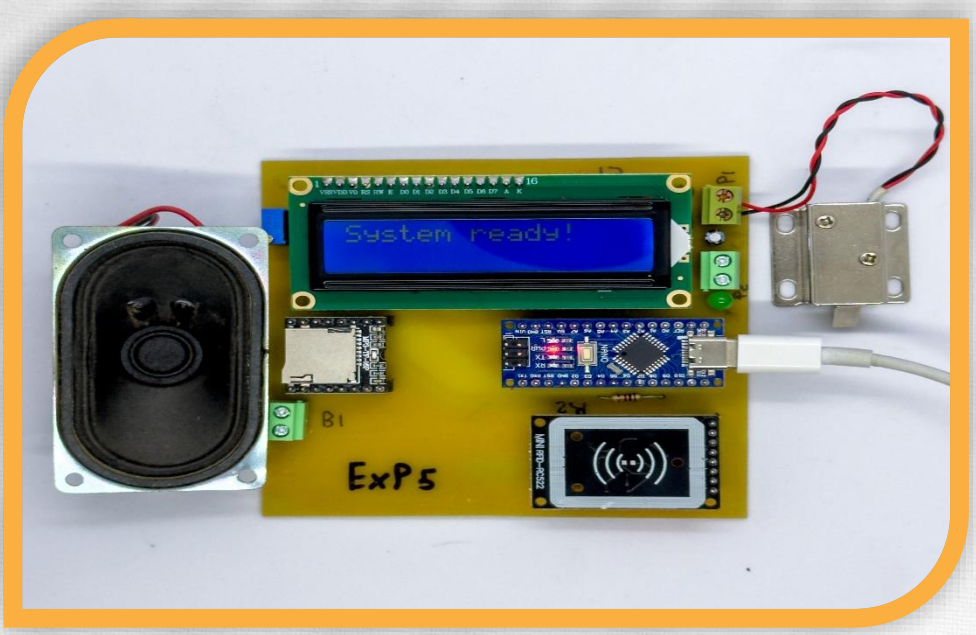
IV. Code transmits

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN pins for nRF24L01
const byte address[6] = "00001"; // Communication address

// Define button and LED pins
#define KEY1 2
#define KEY2 3
#define KEY3 4
#define LED_PIN 5

void setup() {
  // Set button pins as input with pull-up resistors
  pinMode(KEY1, INPUT_PULLUP);
  pinMode(KEY2, INPUT_PULLUP);
  pinMode(KEY3, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);

  // Initialize the radio
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN); // Set power level to minimum
  radio.stopListening(); // Set as transmitter
}
```

IV. Code transmits

```
void loop() {
  // Read button states
  bool key1State = !digitalRead(KEY1);
  bool key2State = !digitalRead(KEY2);
  bool key3State = !digitalRead(KEY3);

  // Send data based on button pressed
  if (key1State) {
    radio.write("F", 1); // Send 'F' for forward
    digitalWrite(LED_PIN, HIGH); // Turn on LED
  } else if (key2State) {
    radio.write("B", 1); // Send 'B' for backward
    digitalWrite(LED_PIN, HIGH); // Turn on LED
  } else if (key3State) {
    radio.write("S", 1); // Send 'S' for stop
    digitalWrite(LED_PIN, LOW); // Turn off LED
  } else {
    radio.write("S", 1); // Default to stop
    digitalWrite(LED_PIN, LOW); // Turn off LED
  }
  delay(100); // Delay to avoid rapid sending
}
```

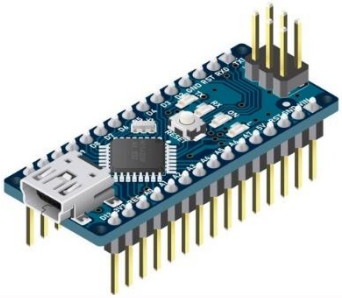
Experiment 3



Door lock control

An Electronic Locking System Using RFID Technology And A Microcontroller For Secure And Convenient Access. It Unlocks The Door Upon Detecting An Authorized RFID Card, With A Red LED Indicating System Power Only. Ideal For Homes, Offices, And Restricted Areas.

I. Components



1x Arduino Nano

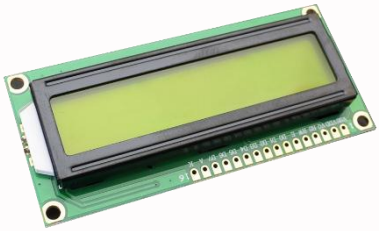
A Compact Microcontroller Board That Acts As The Brain Of The PCB, Processing Sensor Data And Controlling Outputs Like Alarms And Leds.

1

1x MINI REID-RC522

A Module That Scans RFID Cards Or Tags And Sends Unique ID Data To The Arduino For Secure Access Control.

2

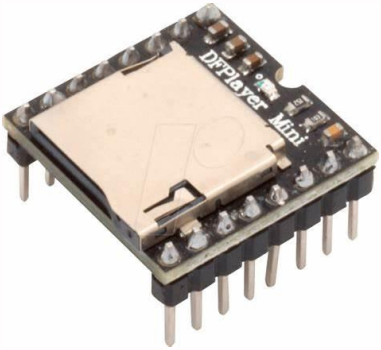


1x LCD HS 1602A

An LCD Display Device (HS1602A) That Shows A Warning Message When Triggered By The Circuit.

3

I. Components



1x DFPLAYER MINI MP3

An MP3 Playback Module (Dfplayer Mini) That Plays Audio Files When Triggered By The Circuit.

4

1x SPEAKER CONNECTOR

A Speaker Connector That Links An Audio Output Device To The Circuit For Sound Playback.

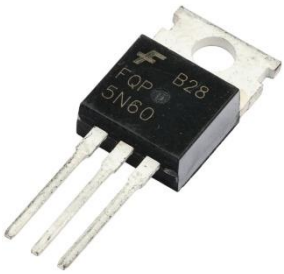
5

1x Solenoid Lock

A Solenoid Lock That Activates Or Unlocks When Triggered By The Circuit.

6

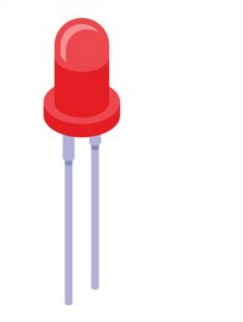
I. Components



1x 5N60 Transistor

A small signal NPN transistor that amplifies or switches electronic signals in the circuit.

7



2x LED

An LED indicator that lights up when triggered by the circuit, serving as a visual signal for specific events.

8

5x 1k Ohm Resistor

Components used to limit current flow, protecting sensitive parts of the circuit from damage.

9

I. Components

1x Diode 1N4007

That Protects The Circuit By Blocking Reverse Current And Ensuring Safe Operation Of Connected Components.

10

2x Capacitor

A Capacitor That Stores And Releases Electrical Energy In The Circuit, Used For Filtering, Smoothing, Or Stabilizing Voltage.

11

2x 10k Ohm Resistor

Components Used To Limit Current Flow, Protecting Sensitive Parts Of The Circuit From Damage.

12

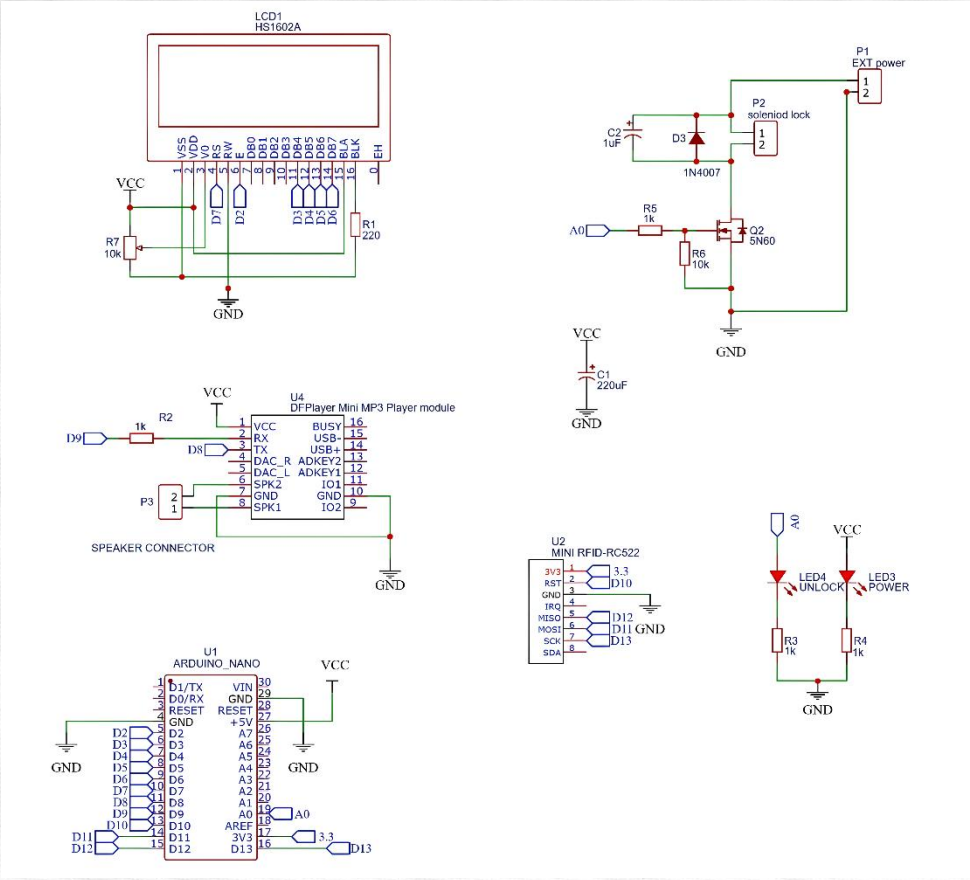
II. Schematic

Door lock control

Reference	Component	Value	Quantity
LCD1	LCD Display	--	1
U1	Arduino Nano	--	1
U2	RFID Module	--	1
U4	MP3 Player Module	--	1
P2	Solenoid Lock	--	1
P1	Ext Power Connector	--	1
P3	Speaker Connector	--	1
R1, R2, R3, R4, R5	Resistors	1kΩ	5
R6, R7	Resistors	10kΩ	2
C1	Capacitor	220µF	1
C2	Capacitor	1µF	1
D3	Diode	--	1
Q2	Transistor	NPR	1
LED3, LED4	LED	--	2

II. Schematic

Door lock control



III. Experiment

Step 1

Connect the Arduino Nano

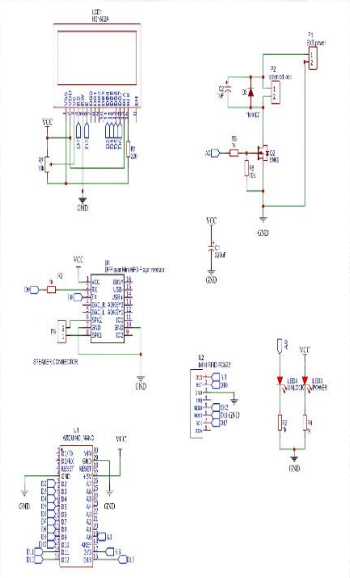
- Plug the Arduino Nano into your computer using a USB cable.
- Open the Arduino IDE on your computer to begin writing the code.

Step 2

Assemble the Circuit

Connect the components based on the provided schematic:

1. Attach the RFID module to the Arduino Nano via SPI communication pins.
2. Connect the solenoid lock using a transistor for control, with a flyback diode for protection.
3. Attach the LCD display to display system status (e.g., "Scan Card" or "Access Granted").
4. Connect the MP3 player module for audio feedback and attach the speaker.
5. Add power and unlock indicator LEDs, with resistors to limit current.
6. Use capacitors to stabilize the power supply.



III. Experiment

```
void loop() {
  // Check for a new card
  if
 (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
    delay(50);
    return;
  }

  // Read UID
  String uid = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    uid += String(mfrc522.uid.uidByte[i], HEX);
  }

  // Check if the card is allowed
  bool isAllowed = true;
  for (byte i = 0; i < 4; i++) {
    if (mfrc522.uid.uidByte[i] != allowedUID[i]) {
      isAllowed = false;
      break;
    }
  }

  // Execute actions based on card validity
  if (isAllowed) {
    lcd.clear();
    lcd.print("Welcome!");
    digitalWrite(ledBlue, HIGH);
    digitalWrite(ledRed, LOW);
    digitalWrite(solenoidPin, HIGH); // Unlock the solenoid
    myDFPlayer.play(1); // Play audio file 1
    delay(3000); // Wait for 3 seconds
    digitalWrite(solenoidPin, LOW); // Lock the solenoid
    digitalWrite(ledBlue, LOW);
    digitalWrite(ledRed, HIGH);
  } else {
    lcd.clear();
    lcd.print("Access Denied!");
    digitalWrite(ledRed, HIGH);
    myDFPlayer.play(2); // Play audio file 2
    delay(2000);
  }

  lcd.clear();
  lcd.print("System Ready");
  mfrc522.PICC_HaltA();
}
```

Step 3

Write the Code

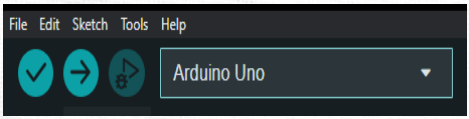
In the Arduino IDE, write a program to:

- Read RFID card data using the RFID module.
- Compare the card data against a list of authorized Ids stored in the code.
- Trigger the solenoid lock and play an "access granted" sound if the card is authorized.
- Blink the unlock LED for visual feedback.
- Display appropriate messages on the LCD (e.g., Unauthorized" or "Access Granted").

Step 4

Compile and Upload the Code

- Compile the code in the Arduino IDE to ensure there are no errors.
- Upload the code to the Arduino Nano using the "Upload" button.



III. Experiment

Step 5

Assemble the PCB

- Disconnect the Arduino Nano from the USB cable.
- Connect it to the PCB according to the schematic.
- Secure all components in their designated positions on the PCB.

Step 6

Test Your Code

- Power the circuit using the external power supply.
- Scan an RFID card:
    - If authorized, the solenoid lock should disengage, allowing the door to open.
    - The unlock LED should blink, and the MP3 module should play the "access granted" sound.
    - If unauthorized, the lock remains engaged, and the LCD displays "Access Denied."

Step 7

Troubleshooting

If the solenoid lock doesn't operate or the system behaves unexpectedly:

1. Recheck all connections against the schematic.
2. Verify the code and re-upload it if necessary.
3. Ensure the power supply provides sufficient voltage and current.

```
void loop() {
  // Check for a new card
  if
 (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
    delay(50);
    return;
  }

  // Read UID
  String uid = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    uid += String(mfrc522.uid.uidByte[i], HEX);
  }

  // Check if the card is allowed
  bool isAllowed = true;
  for (byte i = 0; i < 4; i++) {
    if (mfrc522.uid.uidByte[i] != allowedUID[i]) {
      isAllowed = false;
      break;
    }
  }

  // Execute actions based on card validity
  if (isAllowed) {
    lcd.clear();
    lcd.print("Welcome!");
    digitalWrite(ledBlue, HIGH);
    digitalWrite(ledRed, LOW);
    digitalWrite(solenoidPin, HIGH); // Unlock the solenoid
    myDFPlayer.play(1); // Play audio file 1
    delay(3000); // Wait for 3 seconds
    digitalWrite(solenoidPin, LOW); // Lock the solenoid
    digitalWrite(ledBlue, LOW);
    digitalWrite(ledRed, HIGH);
  } else {
    lcd.clear();
    lcd.print("Access Denied!");
    digitalWrite(ledRed, HIGH);
    myDFPlayer.play(2); // Play audio file 2
    delay(2000);
  }

  lcd.clear();
  lcd.print("System Ready");
  mfrc522.PICC_HaltA();
}
```

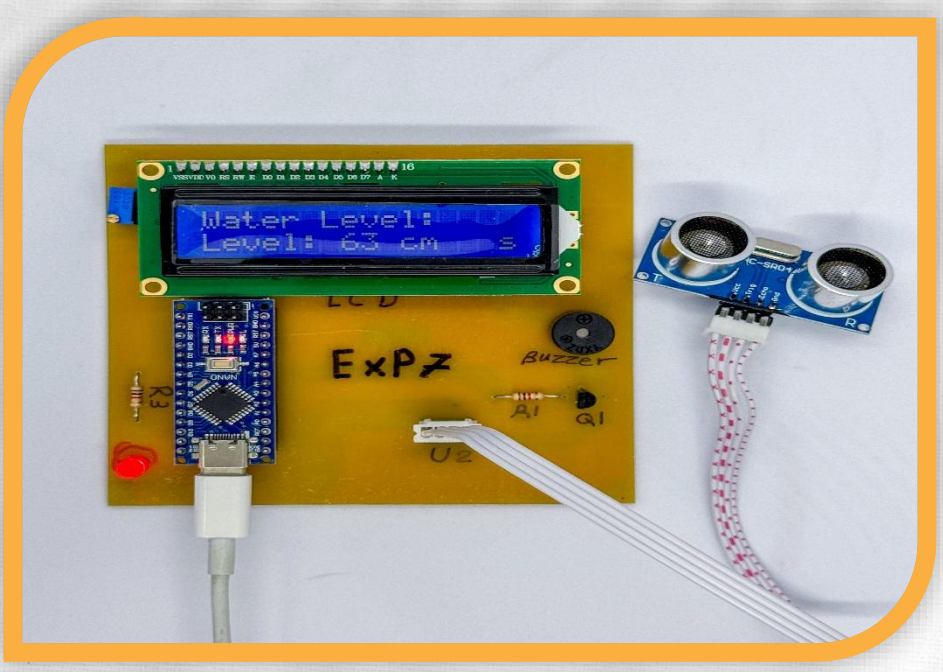
IV. Code

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>

// Define LCD pins
const int RS = 10, EN = 8, DB4 = 7, DB5 = 6, DB6 = 5, DB7 = 4;
LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);

// Define DS18B20 temperature sensor pins
#define ONE_WIRE_BUS 12
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Define joystick, fan, and LED pins
const int joystickY = A4;   // Y-axis of the joystick
const int fanPWM = 3;       // PWM pin for the fan (IRFZ44N)
const int ledPin = 2;      // Red LED pin

// System variables
int fanSpeed = 0;
float temperature = 0;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);

  // Initialize temperature sensor
  sensors.begin();

  // Set output pins
  pinMode(fanPWM, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // Startup message
  lcd.print("System Ready");
  delay(1000);
  lcd.clear();
}
```

IV. Code

```
void loop() {
  // Read temperature
  sensors.requestTemperatures();
  temperature = sensors.getTempCByIndex(0);

  // Read joystick value and map it to fan speed (0-255)
  int joyValue = analogRead(joystickY);
  fanSpeed = map(joyValue, 0, 1023, 0, 255);

  // Apply minimum fan speed (50/255) to prevent it from stopping
  fanSpeed = constrain(fanSpeed, 50, 255);

  // Control fan and LED
  analogWrite(fanPWM, fanSpeed);
  digitalWrite(ledPin, fanSpeed > 0 ? HIGH : LOW);

  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print((char)223);  // Celsius degree symbol
  lcd.print("C  ");

  lcd.setCursor(0, 1);
  lcd.print("Speed: ");
  lcd.print(map(fanSpeed, 0, 255, 0, 100)); // Convert speed to percentage
  lcd.print("%   ");

  delay(100); // Short delay to improve responsiveness
}
```

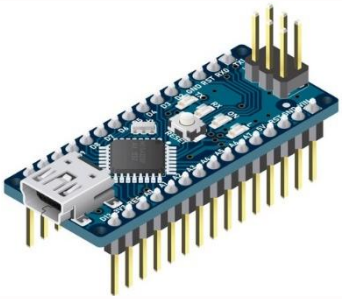
Experiment 4



Water Level Measurement With Ultrasonic

A Water Level Measurement System Using Ultrasonic Technology And A Microcontroller To Monitor And Display The Water Level In A Tank. The Ultrasonic Sensor Measures The Distance Between The Sensor And The Water Surface, And The Data Is Processed By The Microcontroller. The Results Are Displayed On An LCD, Providing A Clear And Real-time Indication Of Water Levels. Ideal For Use In Homes, Agriculture, And Industrial Water Management Systems.

I. Components



1x Arduino nano

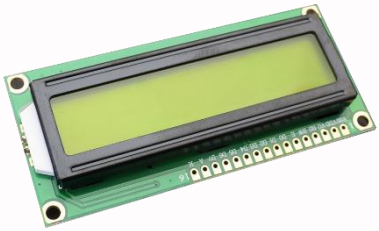
A compact microcontroller board that acts as the brain of the PCB, processing sensor data and controlling outputs like alarms and LEDs.

1

1x Ultrasonic Sensor (HC-SR04)

Measures distance by emitting sound waves and detecting their echo. Used in water level monitoring and obstacle detection.

2



1x LCD HS 1602A

An LCD display device (HS1602A) that shows a warning message when triggered by the circuit.

3

I. Components

1x Buzzer

An audio output device that produces a sound alarm when triggered by the circuit. Ideal for notifications and alerts.

4

1x 10k Ohm Resistor

Components used to limit current flow, protecting sensitive parts of the circuit from damage.

5

1x 220k Ohm Resistor

Components used to limit current flow, protecting sensitive parts of the circuit from damage.

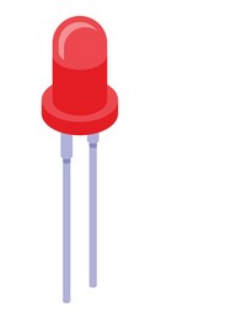
6

I. Components

1x BC547 Transistor

A small NPN transistor used to amplify or switch electronic signals in the circuit.

7



1x LED

An LED indicator that lights up when triggered by the circuit, serving as a visual signal for specific events.

8

2x 1k Ohm Resistor

Components used to limit current flow, protecting sensitive parts of the circuit from damage.

9

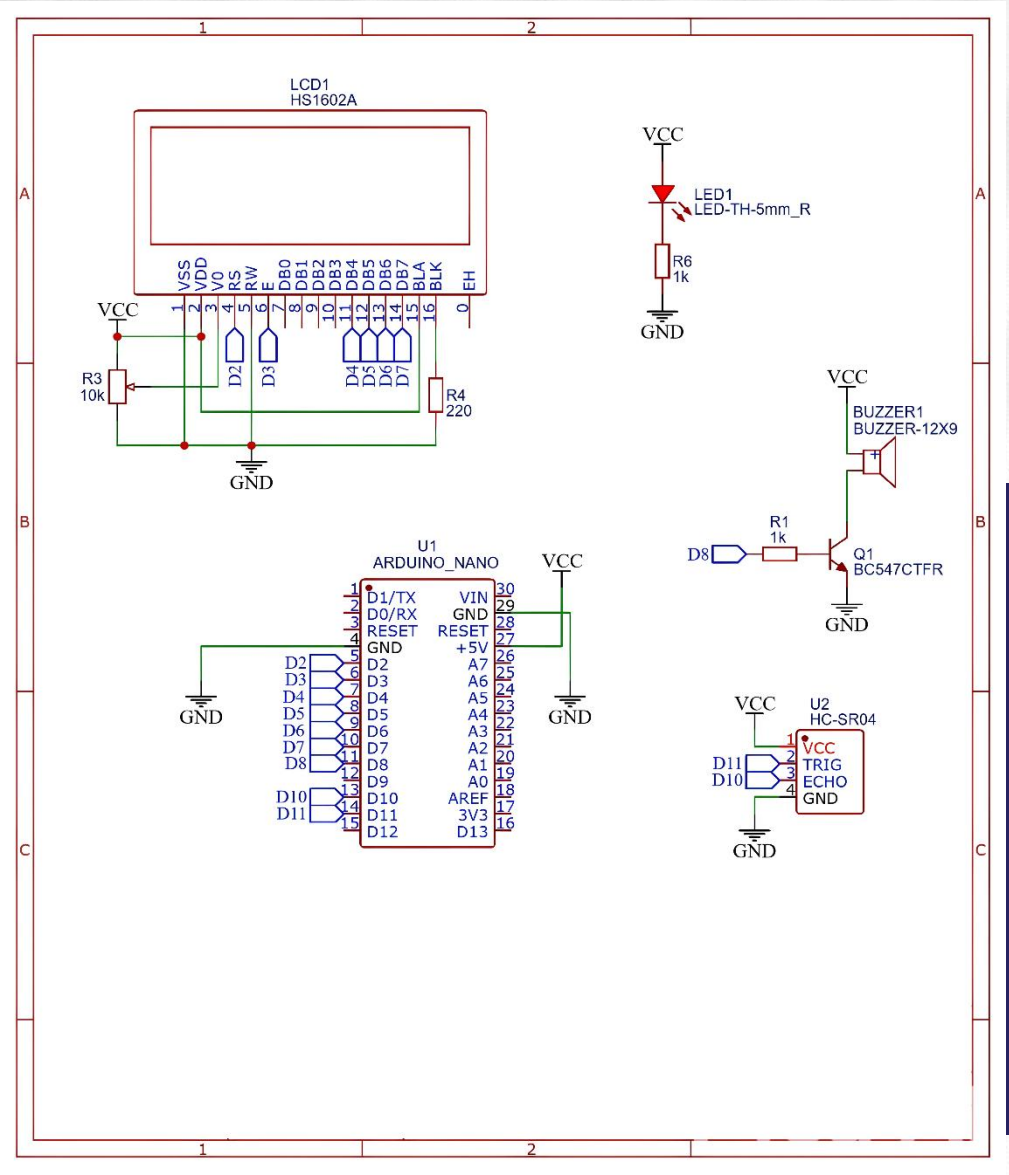
II. Schematic

water level measurement with ultrasonic

Reference	Component	Value	Quantity
LCD1	LCD Display	--	1
U1	Arduino Nano	--	1
U2	Ultrasonic Sensor	--	1
BUZZER1	Buzzer	--	1
Q1	Transistor	--	1
R1, R6	Resistors	1kΩ	2
R3	Resistor	10kΩ	1
R4	Resistor	220Ω	1
LED1	LED	--	1

II. Schematic

Water Level Measurement With Ultrasonic



III. Experiment

Step 1

Connect the Arduino Nano

- Plug the Arduino Nano into your computer using a USB cable.
- Open the Arduino IDE on your computer to begin writing the code.

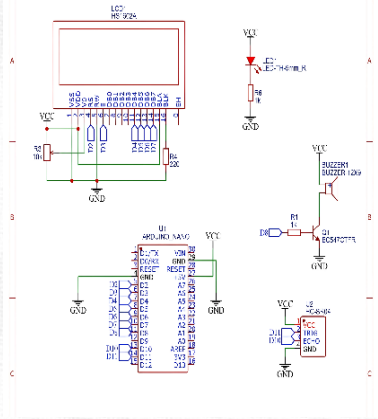
Step 2

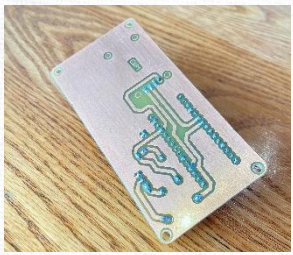
Gather and Connect Components

1. Attach the ultrasonic sensor (HC-SR04) to the Arduino Nano according to the schematic.
    - TRIG pin to a digital pin on the Arduino.
    - ECHO pin to another digital pin on the Arduino.
    - VCC and GND to the respective power pins.

2. Connect the LCD (HS1602A) to the Arduino using appropriate pins. Use a potentiometer to adjust the contrast.

3. Power the system using an external power source or USB cable.





III. Experiment

```
#include <LiquidCrystal.h>

// Define LCD pins (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

// Define ultrasonic sensor pins
const int trigPin = 11;
const int echoPin = 10;

// Define buzzer and LED pins
const int buzzerPin = 8;
const int ledPin = 11;

// Constants
const int TANK_HEIGHT_CM = 100; // Replace with actual tank height
const int ALERT_LEVEL_CM = 50;  // Water level threshold for alerts

// Variables
long duration;
int distance;
int waterLevel;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);
  lcd.print("Water Level:");

  // Initialize ultrasonic sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Initialize buzzer and LED
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  Serial.begin(9600); // For debugging
}
```

Step 3

Write the Code

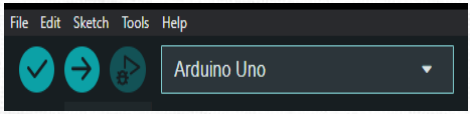
Using the Arduino IDE, write a program to:

- Send trigger pulses to the ultrasonic sensor.
- Calculate the distance by measuring the time taken for the echo pulse.
- Convert the distance into water level height.
- Display the water level percentage or height on the LCD.

Step 4

Test the Sensor

- Test the ultrasonic sensor by measuring distances in different scenarios. Ensure it accurately reads and displays the water level.



III. Experiment

Step 5

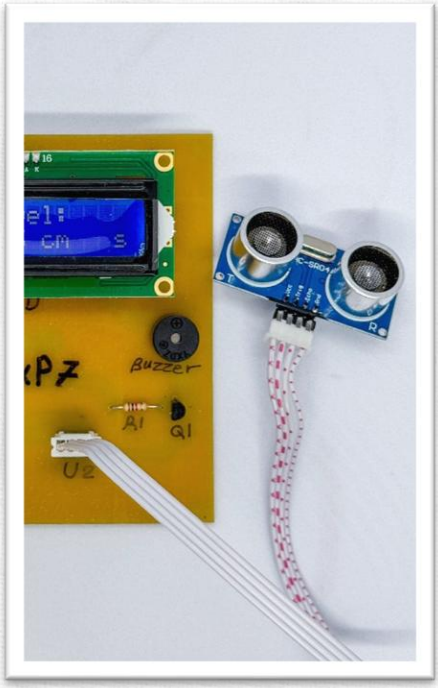
Assemble the PCB

- After verifying the functionality, disconnect the Arduino from the computer and assemble the components on the PCB. Secure the ultrasonic sensor, LCD, and other components in their designated positions.

Step 6

Calibrate the System

- Calibrate the ultrasonic sensor based on the tank's dimensions. Modify the code to reflect the tank's full and empty water levels.



Step 7

Troubleshooting

If the system behaves unexpectedly:

1. Recheck all wiring connections.
2. Verify and re-upload the code.
3. Ensure the power supply is sufficient for the components.

```
void loop() {
  // Measure distance using ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // Convert to cm

  // Validate distance (avoid negative values or out-of-range readings)
  if (distance <= 0 || distance > TANK_HEIGHT_CM) {
    lcd.setCursor(0, 1);
    lcd.print("Error:Check Sensor");
    tone(buzzerPin,800,1200);
    tone(buzzerPin,800,400);
    digitalWrite(ledPin, HIGH);
    delay(1000);
    return; // Skip invalid readings
  }

  // Calculate water level (sensor at top of tank)
  waterLevel = TANK_HEIGHT_CM - distance;

  // Display water level on LCD
  lcd.setCursor(0, 1);
  lcd.print("Level: ");
  lcd.print(waterLevel);
  lcd.print(" cm   "); // Extra spaces to clear old characters

  // Control buzzer and LED
  if (waterLevel < ALERT_LEVEL_CM) {
    digitalWrite(buzzerPin, HIGH); // For active buzzer
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(buzzerPin, LOW);
    digitalWrite(ledPin, LOW);
  }

  delay(500); // Adjust delay as needed
}
```

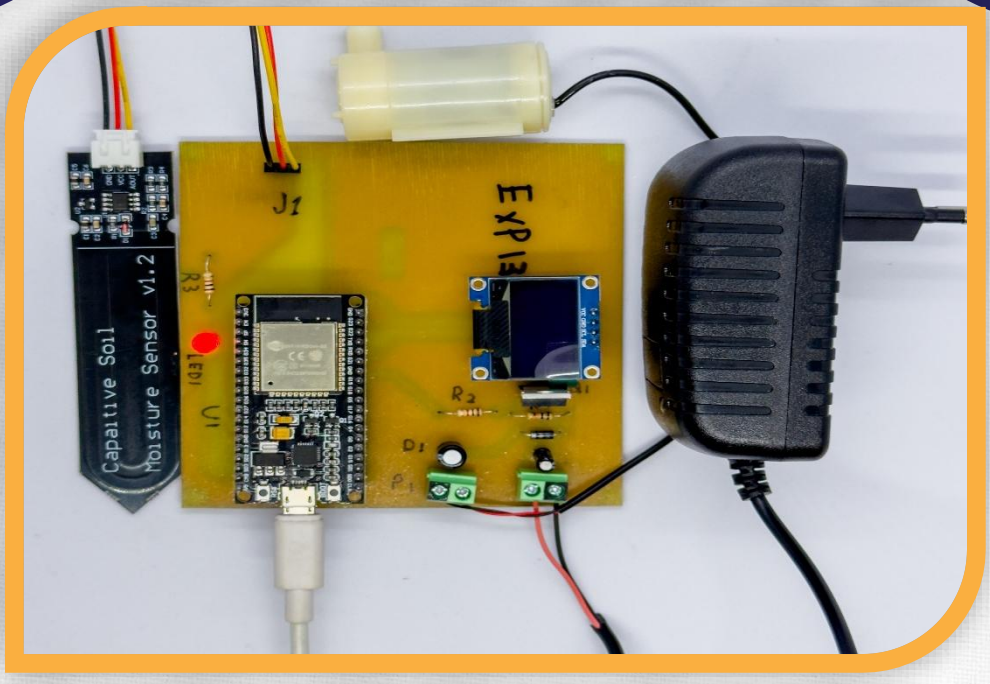
IV. Code

```
#include <LiquidCrystal.h>

// Define LCD pins (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

// Define ultrasonic sensor pins
const int trigPin = 11;
const int echoPin = 10;

// Define buzzer and LED pins
const int buzzerPin = 8;
const int ledPin = 11;

// Constants
const int TANK_HEIGHT_CM = 100; // Replace with actual tank height
const int ALERT_LEVEL_CM = 50;  // Water level threshold for alerts

// Variables
long duration;
int distance;
int waterLevel;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);
  lcd.print("Water Level:");

  // Initialize ultrasonic sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Initialize buzzer and LED
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  Serial.begin(9600); // For debugging
}
```

IV. Code

```
void loop() {
  // Measure distance using ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // Convert to cm

  // Validate distance (avoid negative values or out-of-range readings)
  if (distance <= 0 || distance > TANK_HEIGHT_CM) {
    lcd.setCursor(0, 1);
    lcd.print("Error:Check Sensor");
    tone(buzzerPin,800,1200);
    tone(buzzerPin,800,400);
    digitalWrite(ledPin, HIGH);
    delay(1000);
    return; // Skip invalid readings
  }

  // Calculate water level (sensor at top of tank)
  waterLevel = TANK_HEIGHT_CM - distance;

  // Display water level on LCD
  lcd.setCursor(0, 1);
  lcd.print("Level: ");
  lcd.print(waterLevel);
  lcd.print(" cm   "); // Extra spaces to clear old characters

  // Control buzzer and LED
  if (waterLevel < ALERT_LEVEL_CM) {
    digitalWrite(buzzerPin, HIGH); // For active buzzer
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(buzzerPin, LOW);
    digitalWrite(ledPin, LOW);
  }

  delay(500); // Adjust delay as needed
}
```

Experiment 5



Smart Irrigation System

An ESP32 MQTT Irrigation System Is A Smart Irrigation Setup That Uses The ESP32 Microcontroller And The MQTT Protocol To Automate Watering Plants Based On Soil Moisture Levels Automatic System Watering Arduino Using Circuit Plants Diagram Water Plant Irrigation Pump Sensor Build Soil Moisturewatering Arduino Circuit Moisture Soil Ozeki Kindpng Automatic Plant Watering Project Using Arduino |Arduino Smart Irrigationsystem Plant Watering Automatic Diagram Schematic Arduino Water Moisture Module Working Gadgetronicx Explanation.

I. Components

1x Oled ssd1306 0.96in

OLED (Organic Light Emitting Diode) displays offer excellent color accuracy, deep blacks, and energy efficiency, widely used in smartphones, TVs, and wearable devices

1

1x Water Pump

is a versatile and efficient tool used in various applications, from automotive and marine to gardening and home use.

2

1x Capacitive Soil Moisture Sensor V1.2

A capacitive soil moisture sensor is a durable, corrosion-resistant tool for monitoring soil moisture levels.

3

I. Components

1x IRFZ44N

Is an N-channel power MOSFET used in various electronic circuits due to its high current capacity and low on-resistance.

4

1x Risistor 10Ω

in various electronic circuits to limit current, divide voltage, or provide a specific resistance value.

5

1x ESP32 board

A popular IoT board with a micro-USB port for power and programming, featuring capacitors, resistors, and a metal shield covering the ESP32 chip.

6

I. Components

Terminal Block

Terminal Block used for connecting wires to a PCB.

7

1N4007

The 1N4007 is a popular general-purpose rectifier diode.

8

A 100µF 10V A capacitor

used for filtering, decoupling, and energy storage in various applications.

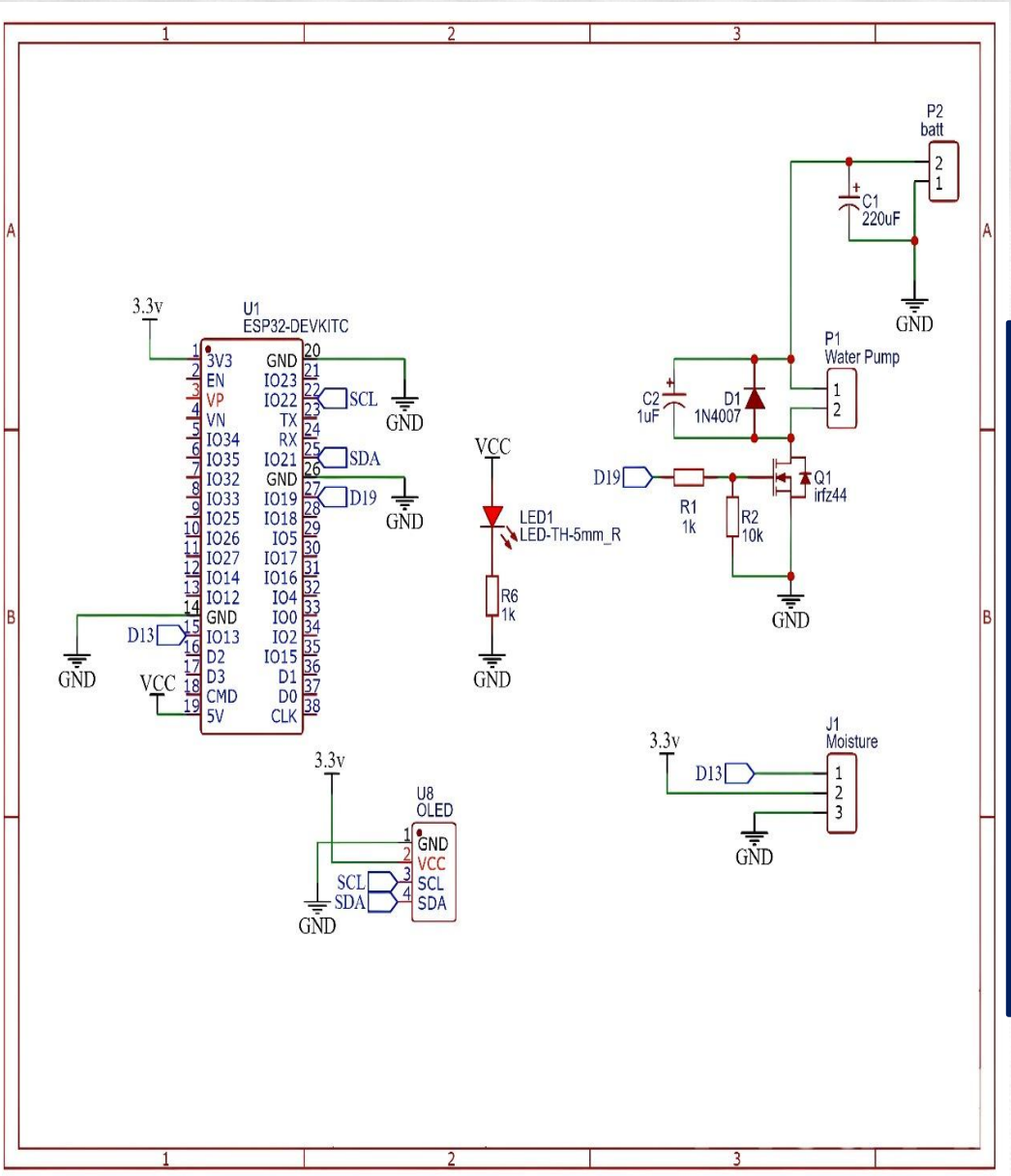
9

II. Schematic

ESP32 MQTT irrigation

Reference	Component	Value	Quantity
U1	ESP32	Wi-Fi & Bluetooth MCU	1
U2	Capacitive Soil Moisture Sensor V1.2	V1.2	1
D3	OLED Display	SSD1306 0.96in	1
P1	water pump	--	1
R1	Resistor	10kΩ	1
J1	Terminal Block	2 Pins KF126-5.0-2P	1
D2	Diode	1N4007	1
D1	Driver (MOSFET or Relay)	N-channel MOSFET or Relay	1
LED1	Red LED	--	1
LED2	Blue LED	--	1
C1	Capacitor	100uF 10V	1

II. Schematic



III. Experiment

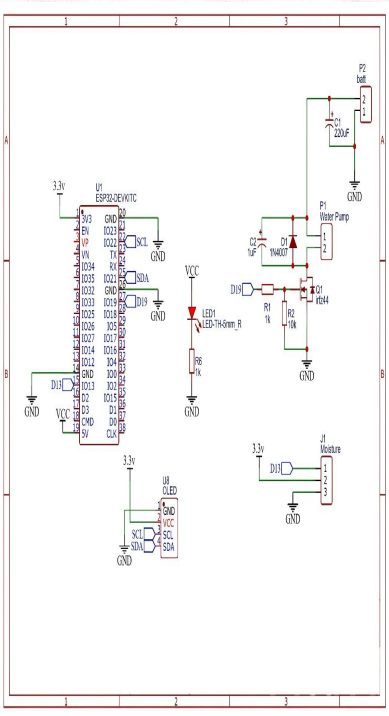
Step 1

- Connect the 3.3V power supply and battery to the circuit at P2.
- Use capacitor C1 (220uF) to filter the power supply and ensure stable voltage.

Step 2

Understand the Connections from the Schematic

- Place the ESP32-DEVKITC microcontroller (U1) at the center of the circuit.
- Connect pin 13 (D13) of the ESP32 to the moisture sensor (J1).
- Connect pins 22 (SCL) and 21 (SDA) of the ESP32 to the OLED display (U8).
- Connect pin 19 (D19) of the ESP32 to the gate of the MOSFET transistor (Q1).



III. Experiment

```
#include <Onewire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>

// Define LCD pins
const int RS = 10, EN = 8, DB4 = 7, DB5 = 6, DB6 = 5, DB7 = 4;
LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);

// Define DS18B20 temperature sensor pins
#define ONE_WIRE_BUS 12
Onewire onewire(ONE_WIRE_BUS);
DallasTemperature sensors(&onewire);

// Define joystick, fan, and LED pins
const int joystickY = A4;   // Y axis of the joystick
const int fanPWM = 3;       // PWM pin for the fan (IRFZ44N)
const int ledPin = 2;      // Red LED pin

// System variables
int fanSpeed = 0;
float temperature = 0;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);

  // Initialize temperature sensor
  sensors.begin();

  // Set output pins
  pinMode(fanPWM, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // Startup message
  lcd.print("System Ready");
  delay(1000);
  lcd.clear();
}
```

Step 3

Write the Code

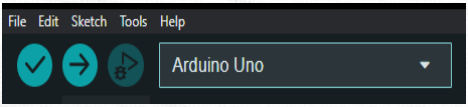
- Connect the moisture sensor (J1) to pin 13 (D13) of the ESP32.

The sensor will measure the soil moisture level and send the data to the ESP32.Connect the OLED display (U8) to the ESP32 using I2C communication via pins 22 (SCL) and 21 (SDA). The display will show the moisture level and other relevant information.

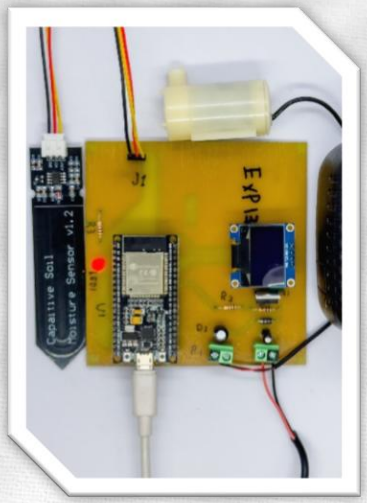
Step 4

Test the Sensor

- Test the ultrasonic sensor by measuring distances in different scenarios. Ensure it accurately reads and displays the water level.



III. Experiment



Step 5

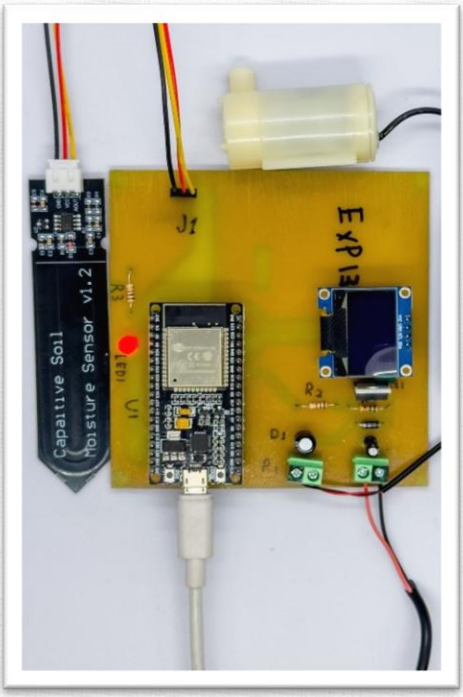
LED Indicator:

- Connect an LED (LED1) to the VCC and ground through a 1k resistor (R6).
- The LED serves as a power indicator, lighting up when the circuit is powered.

Step 6

Calibrate the System

Use R1 (1k) and R2 (10k) for current limiting and voltage division, C2 (1uF) for filtering, and D1 (1N4007) for reverse voltage protection. Program the ESP32 to connect to WiFi, read moisture sensor data, control the water pump, and publish data to an MQTT topic while subscribing to control commands.



Step 7

Troubleshooting

If the system behaves unexpectedly:

1. Recheck all wiring connections.
2. Verify and re-upload the code.
3. Ensure the power supply is sufficient for the components.

```
void loop() {
  // Read temperature
  sensors.requestTemperatures();
  temperature = sensors.getTempCByIndex(0);

  // Read joystick value and map it to fan speed (0-255)
  int joyValue = analogRead(joystickY);
  fanSpeed = map(joyValue, 0, 1023, 0, 255);

  // Apply minimum fan speed (50/255) to prevent it from stopping
  fanSpeed = constrain(fanSpeed, 50, 255);

  // Control fan and LED
  analogWrite(fanPWM, fanSpeed);
  digitalWrite(ledPin, fanSpeed > 0 ? HIGH : LOW);

  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print((char)223);  // Celsius degree symbol
  lcd.print("C  ");

  lcd.setCursor(0, 1);
  lcd.print("Speed: ");
  lcd.print(map(fanSpeed, 0, 255, 0, 100)); // Convert speed to percentage
  lcd.print("%   ");

  delay(100); // Short delay to improve responsiveness
}
```

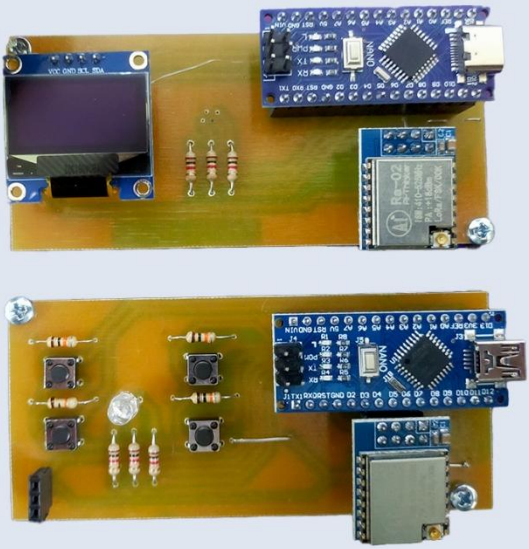
IV. Code

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>

// Define LCD pins
const int RS = 10, EN = 8, DB4 = 7, DB5 = 6, DB6 = 5, DB7 = 4;
LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);

// Define DS18B20 temperature sensor pins
#define ONE_WIRE_BUS 12
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Define joystick, fan, and LED pins
const int joystickY = A4;   // Y-axis of the joystick
const int fanPWM = 3;       // PWM pin for the fan (IRFZ44N)
const int ledPin = 2;      // Red LED pin

// System variables
int fanSpeed = 0;
float temperature = 0;

void setup() {
  // Initialize LCD
  lcd.begin(16, 2);

  // Initialize temperature sensor
  sensors.begin();

  // Set output pins
  pinMode(fanPWM, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // Startup message
  lcd.print("System Ready");
  delay(1000);
  lcd.clear();
}
```

IV. Code

```
void loop() {
  // Read temperature
  sensors.requestTemperatures();
  temperature = sensors.getTempCByIndex(0);

  // Read joystick value and map it to fan speed (0-255)
  int joyValue = analogRead(joystickY);
  fanSpeed = map(joyValue, 0, 1023, 0, 255);

  // Apply minimum fan speed (50/255) to prevent it from stopping
  fanSpeed = constrain(fanSpeed, 50, 255);

  // Control fan and LED
  analogWrite(fanPWM, fanSpeed);
  digitalWrite(ledPin, fanSpeed > 0 ? HIGH : LOW);

  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print((char)223);  // Celsius degree symbol
  lcd.print("C  ");

  lcd.setCursor(0, 1);
  lcd.print("Speed: ");
  lcd.print(map(fanSpeed, 0, 255, 0, 100)); // Convert speed to percentage
  lcd.print("%   ");

  delay(100); // Short delay to improve responsiveness
}
```

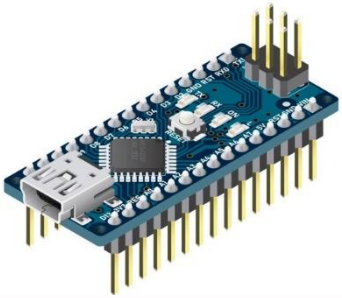
Experiment 6



Lora WAN

Lora WAN Is A Revolutionary Wireless Communication Protocol Specifically Designed For The Internet Of Things (Iot). It's Like A Specialized Language That Allows Devices To "Talk" To Each Other Over Long Distances While Using Very Little Power . It Leverages The Lora Radio Modulation Technique, Enabling Long-range, Low-power, And Cost-effective Data Transmission Over Wide Areas.

I. Components



2x Arduino nano

A compact microcontroller board that acts as the brain of the PCB, processing sensor data and controlling outputs like alarms and LEDs.

1

2x LoRa Modules

They are compact, integrated circuits that contain all the necessary hardware and firmware to enable long-range, low-power wireless communication

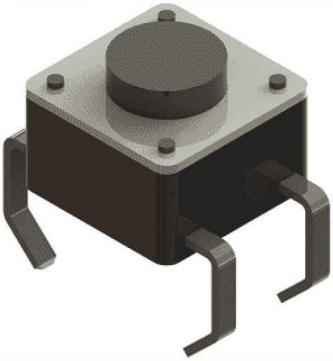
2

1x LCD

An LCD is a flat-panel display that uses liquid crystals to modulate light and create images. It doesn't emit light directly but relies on a backlight or reflector to produce the image.

3

I. Components



4x push btn

is a small, interactive component that users press to initiate a specific function or command. Typically found on devices like calculators, remote controls, and keypads, each button is labeled with a digit, symbol, or letter, enabling straightforward input and control.

4

2x Voltage Regulators

These components are used to regulate the voltage supply to different parts of the circuit.

5

1x 1k Ohm Resistor

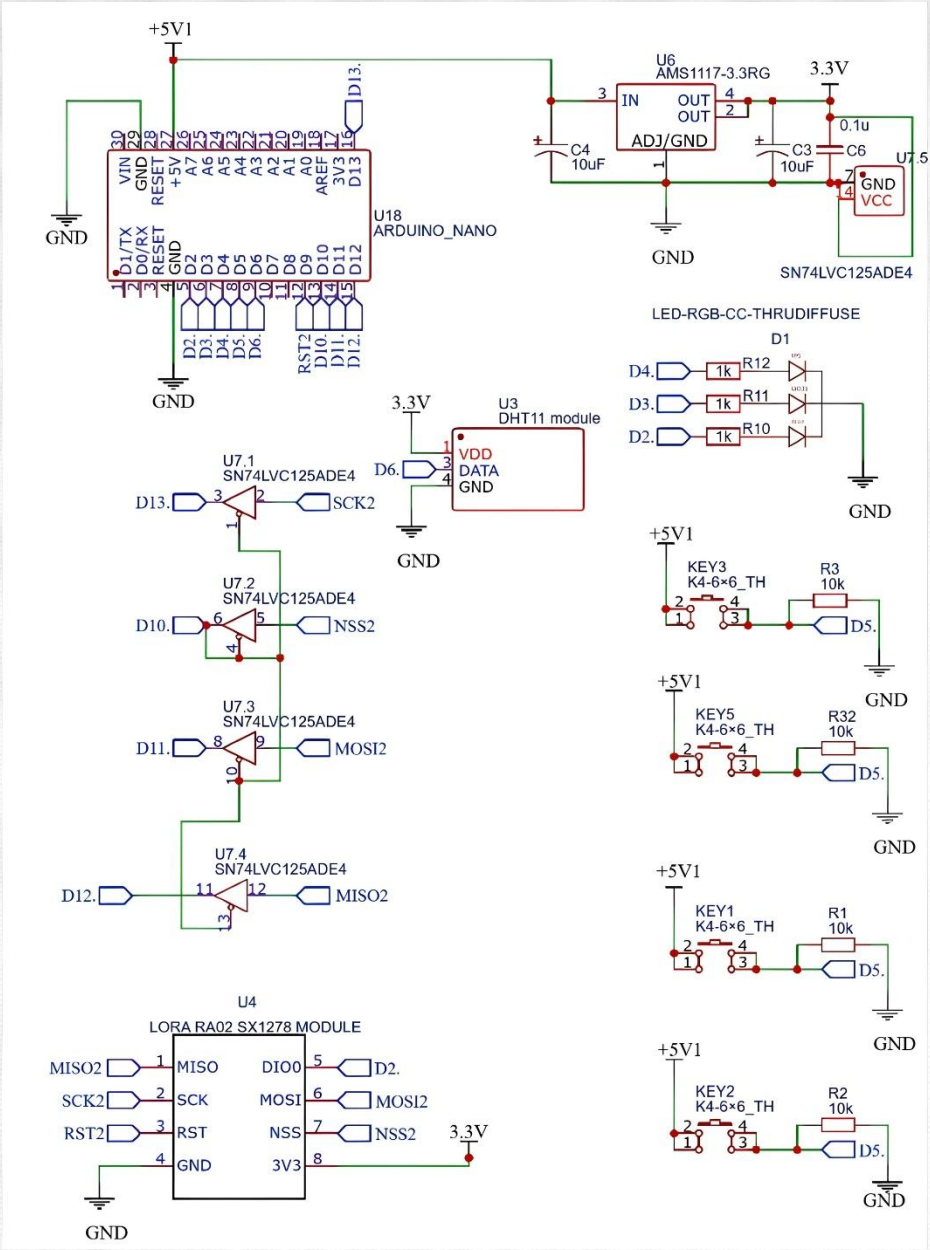
Components used to limit current flow, protecting sensitive parts of the circuit from damage.

6

II. Schematic

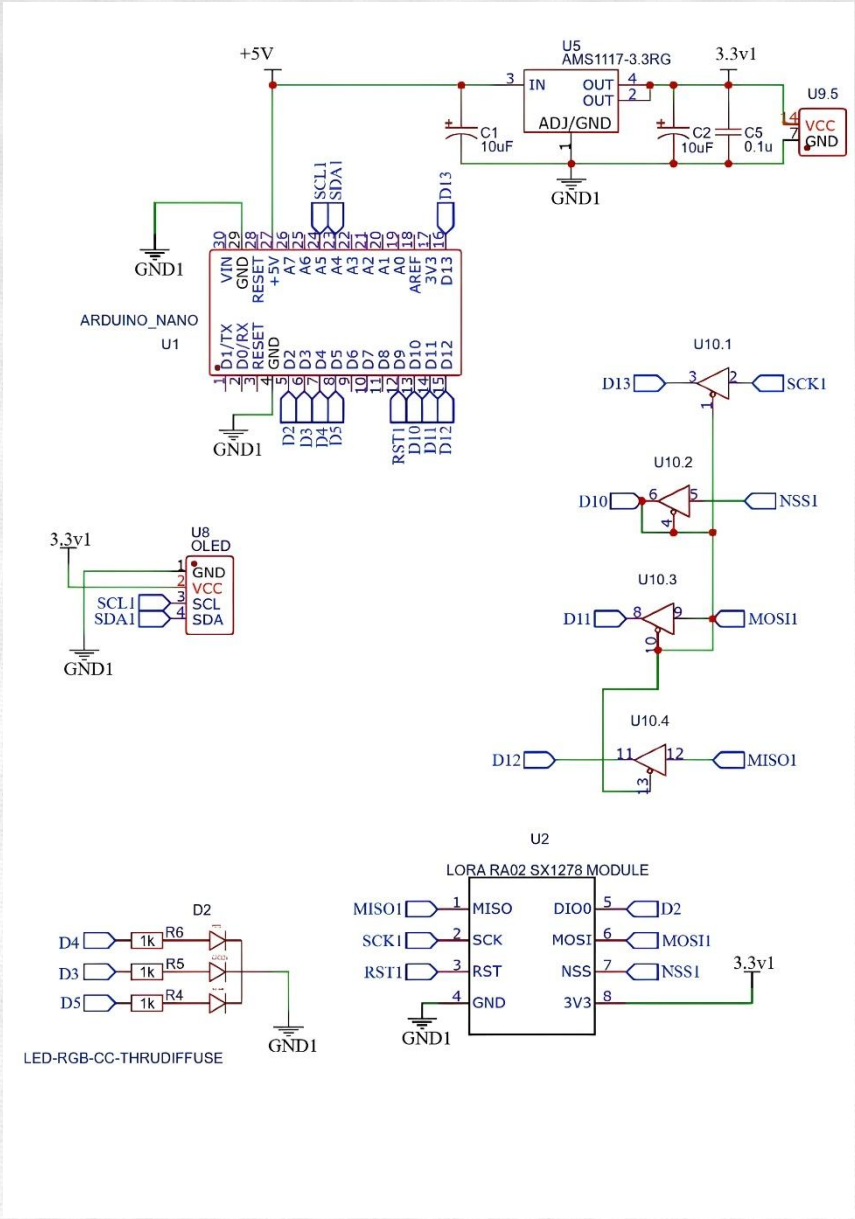
Reference	Component	Value	Quantity
U3	Sensor Module	DHT11	1
U8	Display	OLED (I2C)	1
U2, U9.5	Module	LoRa RA02 SX1278	2
J1, J2, J3	Connector	Header 2x3	3
KEY1, KEY2, KEY3, KEY5	Push Button	K4-6+6_TH	4
U3	Sensor Module	DHT11	1
R1, R2, R3, R4, R5, R6, R10, R11, R12, R32	Resistor	10kΩ	10
C1, C2, C3, C4, C5, C6	Capacitor	10µF	6
D1, D2, D3, D4, D5	LED (Diode)	RGB LEDs	5
U1, U18	Microcontroller	Arduino Nano	2
U5, U6	Voltage Regulator	AMS1117-3.3RG	2
U7.1, U7.2, U7.3, U7.4, U10.1, U10.2, U10.3, U10.4	Buffer IC	SN74LVC125A DE4	8

II. Schematic



II. Schematic

LoRa WAN



III. Experiment

Step 1

Connect the Arduino Nano

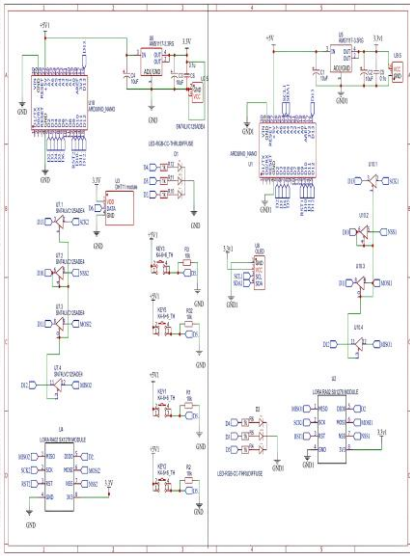
- Plug the Arduino Nano into your computer using a USB cable.
- Open the Arduino IDE on your computer to begin writing the code.

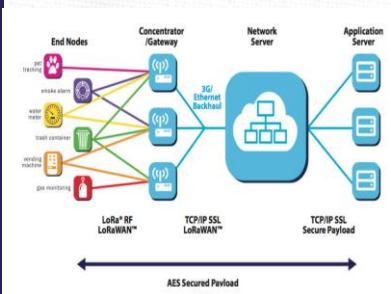
Step 2 :

Assemble the Circuit

Follow these steps to build the circuit:

1. Connect the LoRa module (RA-02) to the Arduino Nano:
   - MISO → D12
   - MOSI → D11
   - SCK → D13
   - NSS → D10
   - GND → GND
   - VCC → 3.3V (regulated by AMS1117).
2. Attach the DHT11 sensor:
   - VCC → 3.3V
   - DATA → D6
   - GND → GND.
3. Connect the OLED display:
   - SDA → A4
   - SCL → A5.
4. Add LEDs to D2, D3, D4, and D5 with 1kΩ resistors.
5. Wire push buttons to D7 and D8 with 10kΩ pull-down resistors.





III. Experiment

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD definition
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address and LCD dimensions

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for the Serial Monitor

  Serial.println("LoRa Receiver");
  if (!LoRa.begin(433E6)) { // LoRa frequency (433 MHz)
    Serial.println("LoRa init failed. Check your connections.");
    while (true);
  }

  // Initialize the LCD
  lcd.begin();
  lcd.backlight();
  lcd.print("LoRa Receiver");
}

void loop() {
  int packetSize = LoRa.parsePacket(); // Check for incoming LoRa packet
  if (packetSize) {
    char receivedChar = LoRa.read(); // Read the received character
    Serial.print("Received: ");
    Serial.println(receivedChar);
 // Display received data on Serial Monitor

    // Display the received data on the LCD
    lcd.clear();
    lcd.print("Received:");
    lcd.setCursor(0, 1);
    lcd.print(receivedChar);
  }
}
```

Step 3

Write the Code

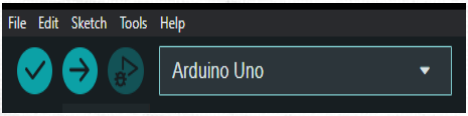
Using the Arduino IDE, write the program to:

1. Read data from the DHT11 sensor.
2. Display the data on the OLED screen.
3. Transmit the data via LoRa.
4. Blink LEDs or update based on received signals.
5. Respond to push-button inputs.

Step 4

Upload the Code

1. Compile the program in the Arduino IDE.
2. Upload the program to the Arduino Nano via USB.
3. Open the Serial Monitor to confirm the code is running properly.



Step 5

Test the System

1. Power the circuit through USB or an external power supply.
2. Verify that the OLED displays real-time sensor data.
3. Ensure the LoRa modules can transmit and receive data correctly.
4. Check that LEDs respond to events and buttons trigger desired actions.

Step 6

Troubleshooting

If the system does not work as expected:

1. Recheck all circuit connections.
2. Ensure the LoRa module frequency settings match.
3. Verify the code logic and re-upload if necessary.
4. Test the power supply to confirm sufficient voltage and current.

Step 7

Finalize the Project

1. Once testing is successful, solder the components onto a PCB for a permanent setup.
2. Secure the system in an enclosure to protect the hardware.
3. Deploy the project in your desired location, ensuring the LoRa modules are within communication range for effective data transmission.

```
#include <SPI.h>
#include <LoRa.h>
#include <Keypad.h>

// Keypad definition
const byte ROWS = 4; // Number of rows
const byte COLS = 3; // Number of columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {2, 3, 4, 5}; // Pins connected to the rows
byte colPins[COLS] = {6, 7, 8};    // Pins connected to the columns
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for the Serial Monitor

  Serial.println("LoRa Sender");
  if (!LoRa.begin(433E6)) { // LoRa frequency (433 MHz)
    Serial.println("LoRa init failed. Check your connections.");
    while (true);
  }
}

void loop() {
  char key = keypad.getKey(); // Read key from keypad
  if (key) { // If a key is pressed
    Serial.print("Sending: ");
    Serial.println(key); // Display the key on Serial Monitor
    LoRa.beginPacket(); // Start LoRa packet
    LoRa.print(key);    // Add the key to the packet
    LoRa.endPacket();   // Send the packet
  }
}
```

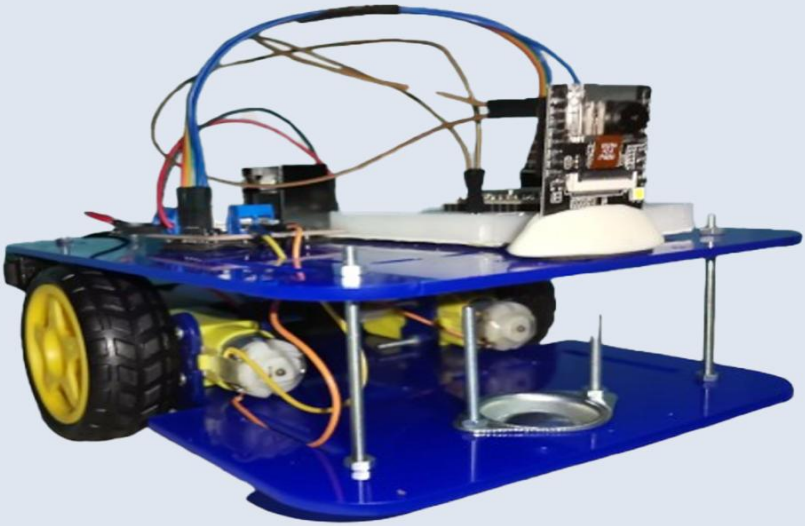
IV. Code Receiver

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD definition
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address and LCD dimensions

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for the Serial Monitor

  Serial.println("LoRa Receiver");
  if (!LoRa.begin(433E6)) { // LoRa frequency (433 MHz)
    Serial.println("LoRa init failed. Check your connections.");
    while (true);
  }

  // Initialize the LCD
  lcd.begin();
  lcd.backlight();
  lcd.print("LoRa Receiver");
}

void loop() {
  int packetSize = LoRa.parsePacket(); // Check for incoming LoRa packet
  if (packetSize) {
    char receivedChar = LoRa.read(); // Read the received character
    Serial.print("Received: ");
    Serial.println(receivedChar);
 // Display received data on Serial Monitor

    // Display the received data on the LCD
    lcd.clear();
    lcd.print("Received:");
    lcd.setCursor(0, 1);
    lcd.print(receivedChar);
  }
}
```

IV. Code Transmitter

```
#include <SPI.h>
#include <LoRa.h>
#include <Keypad.h>

// Keypad definition
const byte ROWS = 4; // Number of rows
const byte COLS = 3; // Number of columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {2, 3, 4, 5}; // Pins connected to the rows
byte colPins[COLS] = {6, 7, 8};    // Pins connected to the columns
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for the Serial Monitor

  Serial.println("LoRa Sender");
  if (!LoRa.begin(433E6)) { // LoRa frequency (433 MHz)
    Serial.println("LoRa init failed. Check your connections.");
    while (true);
  }
}

void loop() {
  char key = keypad.getKey(); // Read key from keypad
  if (key) { // If a key is pressed
    Serial.print("Sending: ");
    Serial.println(key); // Display the key on Serial Monitor
    LoRa.beginPacket(); // Start LoRa packet
    LoRa.print(key);    // Add the key to the packet
    LoRa.endPacket();   // Send the packet
  }
}
```

Experiment 7



ESP32-CAM Streaming and Remote Control

The car is designed to be controlled remotely via a controller, allowing for navigation and operation. The ESP32-CAM streams live video to a web application, enabling users to see what the camera captures in real-time. This integration of components facilitates an interactive and engaging experience, combining mobility with visual feedback.

I. Components

2x dc motor

A DC motor is an electrical device that converts direct current (DC) electrical energy into mechanical energy through magnetic forces. They are commonly used in various applications due to their simplicity and ease of control.

1

1x Chassis

A gas-detecting component that identifies smoke or specific gases and sends corresponding data to the Arduino.

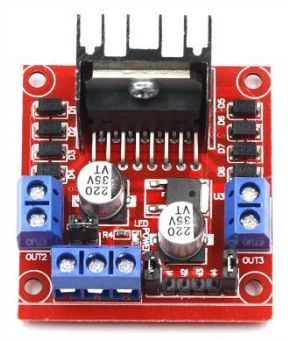
2

1x Esp32 cam

The ESP32-CAM is a compact and cost-effective development board that combines the capabilities of the ESP32 microcontroller with a camera module, making it ideal for various IoT and embedded applications.

3

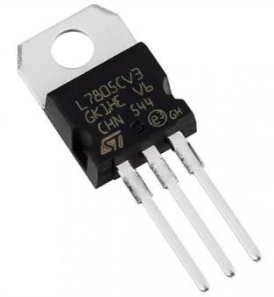
I. Components



1x Driver l298n

The L298N is a popular dual H-bridge motor driver IC that allows you to control the direction and speed of DC motors and stepper motors.

4



1x Regulator 5 volt

A 5V voltage regulator is an electronic component that provides a stable output voltage of 5 volts, regardless of variations in the input voltage or load conditions..

5

Battery holder

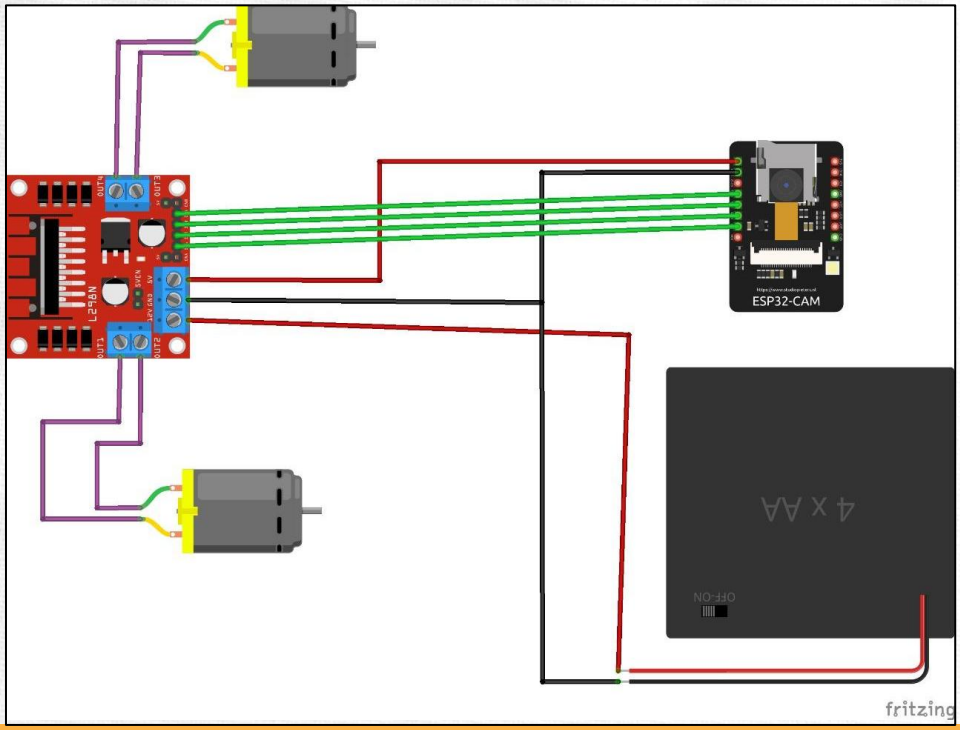
A battery holder is a device designed to securely hold one or more batteries in place while providing electrical connections to the battery terminals.

6

II. Diagram

ESP32-CAM Streaming and Remote Control

Component	Value	Quantity
dc motor	640 rpm	2
Chassis	--	1
Esp32 cam	--	1
Driver l298n	--	1
Regulator	5V	1
Battery holder	4 slots	1



III. Experiment

Step 1

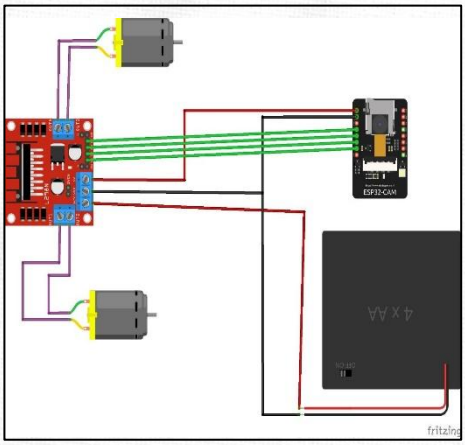
Connect the Esp8266 & esp_cam

- Plug the Esp8266 & esp-cam into your computer using a USB cable.
- Open the Arduino IDE on your computer to begin writing the code.

Step 2

Understand the Connections from the diagram

- Refer to the schematic diagram to identify how the components are connected:
    - MQ Gas Sensor:
        - A0 (sensor) → A0 (Arduino Nano)
        - D0 (sensor) → D3 (Arduino Nano)
    - Buzzer: D4 (Arduino Nano) → Positive terminal of buzzer
    - R2 (Power Indicator): Connected to indicate power to the PCB.



III. Experiment

Step 3

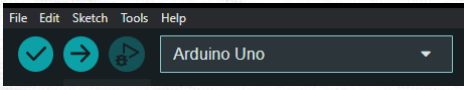
Write the Code for esp cam

- In the Arduino IDE, write the program to:
  - Write the wifi network details.
  - Make the ip static.
  - Stream the video from the esp cam.
  - You can either use esp cam example or scan this qr to get the full code
- Use the diagram as a reference to ensure the correct pin assignments in your code.

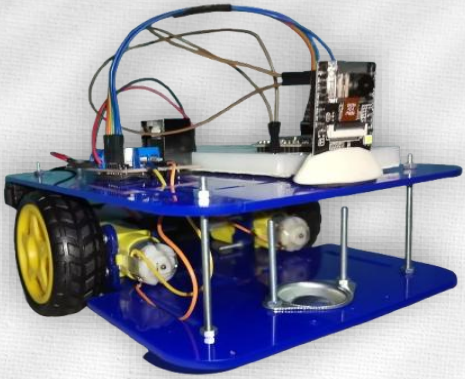
Step 4

Compile and Upload the Code

- Compile the code in the Arduino IDE to ensure there are no errors.
- Upload the code to the esp cam(AI thinker) using the "Upload" button.



III. Experiment



Step 5

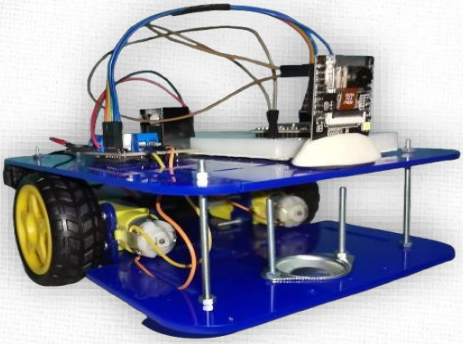
Assemble the car and check the power

- Carefully ensure the voltage for each component knowing that both esp cam, esp8266 receive 5V while the motor driver 12V

Step 6

Test Your Code

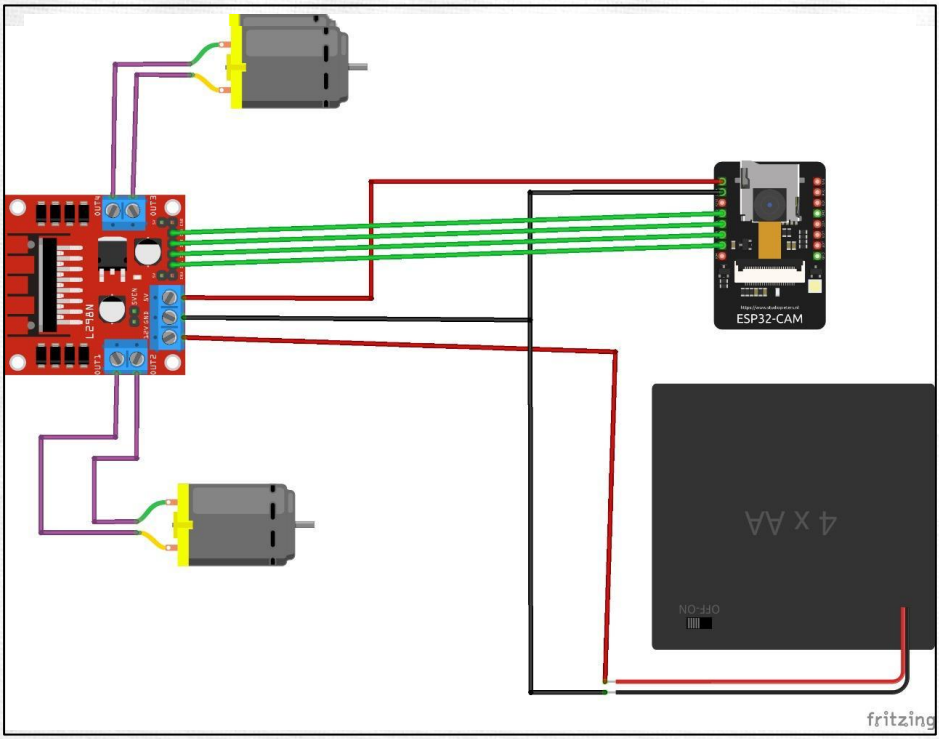
- Power on the Car:
    - Connect to your wifi network.
    - Open your browser and visit the car's ip
    - Start moving the car around using the web buttons.



Step 7

Troubleshooting

- If the stream doesn't start recheck the esp cam voltage ensuring it is getting its right volt and ampere.
- If the car movement isn't right try swapping the motor pins.



TEAM
Smart City

ID	الاسم بالإنجليزية	الاسم بالعربية
2220016	Ahmed Hassan El-Husseini Khtan	احمد حسن الحسيني
2220025	Ahmed Samir Ibrahim Darwish	احمد سمير ابراهيم درويش
2220037	Ahmed Mohamed Ahmed El-Sayed Saleh	احمد محمد احمد السيد
2220046	Ahmed Mohamed Fathy Mohamed	أحمد محمد فتحي محمد
2220047	Ahmed Mohamed Gad	احمد محمد جاد
3330031	Ahmed Abdelhamid Saqr	احمد عبد الحميد صقر
2220103	Ibrahim Saleh Ibrahim	ابراهيم صالح ابراهيم
2220117	Basma Mahmoud Mohamed Mahmoud	بسملة محمود محمد محمود
2220130	Jumana Mohamed Hassan Abouzeid	جمانه محمد حسن ابوزيد
2220169	Rawan Rabie El-Said Mahmoud	روان ربيع السعيد محمود
2220170	Rawan Magdy Ezzat Anbar	روان مجدي عزت عنبر
2220172	Rawan Yasser Hassan Ahmed	روان ياسر حسن احمد
2220180	Ziad Hassan Mohamed Mohamed	زياد حسن محمد محمد
2220200	Salma Atef Mohamed	سلمي عاطف محمد
2220205	Saif Eldin Saber Mohamed	سيف الدين صابر محمد
2220214	Sherif Hamdy Mohamed El-Sayed	شريف حمدي محمد السيد
2220219	Shahd Mohamed Abdelsamad Abdelhady	شهد محمد عبدالصمد

Team leader

1-Ziad Hassan Mohamed Mohamed 2220180
2- Abdelhamid Ahmed 2220556
3- Youssef Mohamed Ali 2220536
4- Omar Mohamed Ali El-Gamal 2220315

TEAM
Smart City

ID	الاسم بالإنجليزية	الاسم بالعربية
2220271	Abdullah Mohamed Gharib	عبدالله محمد غريب شحاته
2220281	Abdelrahman Samir Saad	عبدالرحمن سمير سعد مندور
2220285	Abdelrahman Mohamed Said	عبدالرحمن محمد سعيد محمد
2220315	Omar Mohamed Ali El- Gamal	عمر محمد محمد علي الجمل
2220415	Mohamed Medhat Hassan Ali	محمد مدحت حسن علي
2220466	Mostafa Ismail Fahmy Mostafa	مصطفى اسماعيل فهمي مصطفى
2220469	Mostafa Ayman Khalaf	مصطفى ايمن خلف زغلول
2220499	Nada Hany Mohamed	ندي هاني محمد محمود
2220505	Nour Mohamed Hafzy	نور محمد حفظي
2220510	Hager Suleiman	هاجر سليمان
2220527	Youssef Bahgat Ezzat Ahmed	يوسف بهجت عزت احمد
2220536	Youssef Mohamed Ali	يوسف محمد علي محمود علي
2220556	Abdelhamid Ahmed	عبدالحميد احمد عبدالحميد
2220221	Shaimaa Mohamed Abdel-Maqsoud	شيماء محمد عبد المقصود
2220255	Abdelrahman Mohamed	عبد الرحمن محمد السيد حسين
2220256	Abdelrahman Mohamed	عبدالرحمن محمد احمد

Team leader

1-Ziad Hassan Mohamed Mohamed 2220180
2- Abdelhamid Ahmed 2220556
3- Youssef Mohamed Ali 2220536
4- Omar Mohamed Ali El-Gamal 2220315

Borg Al Arab Technological University

Your Electronics Journey Begins Here

Thank you for exploring our electronics collection. Each PCB is designed with innovation and precision.

جامعة برج العرب التكنولوجية
BORG AL ARAB TECHNOLOGICAL UNIVERSITY