Project Name: **Touchless Dustbin**

- ❖ **Components Required**

1. Arduino UNO
2. Ultrasonic Sensor (HC-SR04)
3. Servo Motor (SG90)
4. Breadboard
5. Jumper Wires
6. USB Cable
7. Dustbin with a movable lid

- ❖ **Connections**

HC-SR04 Ultrasonic Sensor

Sensor Pin	**Arduino UNO**
VCC	5V
GND	GND
TRIG	D9
ECHO	D10

Servo Motor

Servo Wire	**Arduino UNO**
Red (VCC)	5V
Brown/Black (GND)	GND
Orange/Yellow (Signal)	D6

Working

- The ultrasonic sensor detects a person's hand near the dustbin.
- When the hand comes within **20 cm**, Arduino rotates the servo to **90°** and opens the lid.
- After **3 seconds**, the servo returns to **0°** and closes the lid.

- This allows the dustbin to be used **without touching the lid**.

Arduino Code

```
#include <Servo.h>

Servo lidServo;

#define TRIG_PIN 9
#define ECHO_PIN 10
#define SERVO_PIN 6

long duration;
int distance;

void setup() {
  Serial.begin(9600);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  lidServo.attach(SERVO_PIN);
  lidServo.write(0);   // Lid closed
}

void loop() {
  distance = getDistance();
```

```
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance <= 20 && distance > 0) {
    lidServo.write(90);   // Open lid
    delay(3000);

    lidServo.write(0);    // Close lid
    delay(1000);
  }

  delay(100);
}

int getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
```

```
  if (duration == 0) {
    return 400;
  }

  return duration * 0.034 / 2;
}
```

Note

- **20 cm** is the detection distance and can be changed in the code.
- 0° = lid closed and 90° = lid open. Adjust these angles according to your dustbin mechanism.
- If the servo is large or the lid is heavy, use a **separate 5V supply for the servo** and connect its GND to Arduino GND.
- Do not power a high-current servo from the Arduino 5V pin.

Project Name: **Automatic Door System**

- ❖ **Components Required**

1. Arduino UNO
2. IR Sensor Module
3. Servo Motor (SG90)
4. Buzzer
5. Breadboard
6. Jumper Wires
7. USB Cable
8. Small door/cardboard door model

- ❖ **Connections**

IR Sensor

IR Sensor Pin	Arduino UNO
VCC	5V
GND	GND
OUT	D2

Servo Motor

Servo Wire	Arduino UNO
Red (VCC)	5V
Brown/Black (GND)	GND
Orange/Yellow (Signal)	D6

Buzzer

Buzzer Pin	Arduino UNO
+	D8

Buzzer Pin Arduino UNO

– GND

Working

- The IR sensor detects a person/object approaching the door.
- When an object is detected, the **servo rotates to 90°** and opens the door.
- The **buzzer gives a short beep** to indicate that the door is opening.
- After **3 seconds**, the servo returns to **0°** and closes the door.
- The system works automatically without touching the door.

Arduino Code

```
#include <Servo.h>

Servo doorServo;

#define IR_PIN 2
#define SERVO_PIN 6
#define BUZZER_PIN 8

void setup() {
  pinMode(IR_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  doorServo.attach(SERVO_PIN);
  doorServo.write(0);   // Door closed
}

void loop() {
```

```
  int irState = digitalRead(IR_PIN);


  // Most IR modules give LOW when an object is detected
  if (irState == LOW) {


    doorServo.write(90);   // Open door


    digitalWrite(BUZZER_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZER_PIN, LOW);


    delay(3000);          // Keep door open


    doorServo.write(0);    // Close door
    delay(1000);
  }
}
```

Note

- Most IR sensor modules give **LOW when an object is detected**. If your sensor works opposite, change LOW to HIGH.
- Adjust the IR sensor's onboard potentiometer to set the detection distance.
- Change 90 in the code if your door requires a different servo angle.
- If the servo is under heavy load, use a separate 5V supply and connect its **GND to Arduino GND**.

Project Name: Smart Barrier Gate

- ❖ **Components Required**

1. Arduino UNO
2. Ultrasonic Sensor (HC-SR04)
3. Servo Motor (SG90)
4. Buzzer
5. Breadboard
6. Jumper Wires
7. USB Cable
8. Small barrier gate model

- ❖ **Connections**

HC-SR04 Ultrasonic Sensor

Sensor Pin	Arduino UNO
VCC	5V
GND	GND
TRIG	D9
ECHO	D10

Servo Motor

Servo Wire	Arduino UNO
Red (VCC)	5V
Brown/Black (GND)	GND
Orange/Yellow (Signal)	D6

Buzzer

Buzzer Pin Arduino UNO

+	D8
–	GND

Working

- The ultrasonic sensor detects an approaching vehicle/object.
- When the object comes within **20 cm**, the servo rotates to **90°** and lifts the barrier.
- The buzzer gives a short beep when the barrier opens.
- After **3 seconds**, the servo returns to **0°** and lowers the barrier.
- The process repeats automatically.

Arduino Code

```
#include <Servo.h>

Servo gateServo;

#define TRIG_PIN 9
#define ECHO_PIN 10
#define SERVO_PIN 6
#define BUZZER_PIN 8

long duration;
int distance;

void setup() {
  Serial.begin(9600);
```

```
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  gateServo.attach(SERVO_PIN);
  gateServo.write(0);   // Barrier closed
}

void loop() {
  distance = getDistance();

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance <= 20 && distance > 0) {

    gateServo.write(90);   // Open barrier

    digitalWrite(BUZZER_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZER_PIN, LOW);

    delay(3000);          // Keep barrier open

    gateServo.write(0);    // Close barrier
```

```
    delay(1000);
  }

  delay(100);
}

int getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
  if (duration == 0) {
    return 400;
  }
  return duration * 0.034 / 2;
}
```

Note

- **20 cm** is the vehicle detection distance and can be changed.
- 0° = barrier closed and 90° = barrier open. Adjust the angles according to your model.
- For a heavy barrier, use a separate suitable power supply for the servo and connect its **GND to Arduino GND**.

Project Name: **Ultrasonic Radar Scanner** 📡

❖ **Components Required**

1. Arduino UNO
2. HC-SR04 Ultrasonic Sensor
3. Servo Motor (SG90)
4. TFT Display (1.8" ST7735)
5. Breadboard
6. Jumper Wires
7. USB Cable

❖ **Connections**

HC-SR04 Ultrasonic Sensor

Sensor Pin	Arduino UNO
VCC	5V
GND	GND
TRIG	D7
ECHO	D8

Servo Motor

Servo Wire	Arduino UNO
Red (VCC)	5V
Brown/Black (GND)	GND
Orange/Yellow (Signal)	D6

1.8" ST7735 TFT Display

TFT Pin	Arduino UNO
VCC	5V

TFT Pin	Arduino UNO
GND	GND
SCL/CLK	D13
SDA/MOSI	D11
CS	D10
DC/A0	D9
RST	D5

Important: The TFT pin names can vary between modules. Check the labels printed on your particular TFT before connecting it.

Working

- The servo rotates the ultrasonic sensor from **0° to 180°**.
- At each angle, the HC-SR04 measures the distance of an object.
- The Arduino sends the **angle and distance** information to the TFT display.
- The display shows a simple radar-style scanning screen.
- When an object is detected within the set range, its position can be observed based on the servo angle.

Arduino Code

Install these libraries in Arduino IDE:

- **Adafruit GFX Library**
- **Adafruit ST7735 and ST7789 Library**
- **Servo**

```
#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Adafruit_ST7735.h>

#include <Servo.h>
```

```
// TFT pins
#define TFT_CS   10
#define TFT_DC   9
#define TFT_RST  5

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

Servo radarServo;

// Ultrasonic pins
#define TRIG_PIN 7
#define ECHO_PIN 8

int angle;
int distance;

void setup() {
  Serial.begin(9600);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  radarServo.attach(6);

  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
```

```
  tft.fillScreen(ST77XX_BLACK);
  tft.setTextColor(ST77XX_GREEN);
  tft.setTextSize(2);
  tft.setCursor(25, 5);
  tft.println("RADAR");

  delay(1000);
}

void loop() {

  // Scan from 0 to 180 degrees
  for (angle = 0; angle <= 180; angle += 5) {

    radarServo.write(angle);
    delay(50);

    distance = getDistance();

    displayData(angle, distance);
  }

  // Scan back from 180 to 0 degrees
  for (angle = 180; angle >= 0; angle -= 5) {

    radarServo.write(angle);
```

```
    delay(50);

    distance = getDistance();

    displayData(angle, distance);
  }
}

int getDistance() {

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 30000);

  if (duration == 0) {
    return 400;
  }

  int d = duration * 0.034 / 2;
```

```
  if (d > 400) {
    d = 400;
  }

  return d;
}

void displayData(int angle, int distance) {

  tft.fillRect(0, 30, 160, 90, ST77XX_BLACK);

  tft.setTextSize(1);
  tft.setTextColor(ST77XX_WHITE);

  tft.setCursor(5, 35);
  tft.print("Angle: ");
  tft.print(angle);
  tft.println(" deg");

  tft.setCursor(5, 55);
  tft.print("Distance: ");
  tft.print(distance);
  tft.println(" cm");

  // Object detection
  tft.setCursor(5, 80);
```

```
  if (distance < 50) {
    tft.setTextColor(ST77XX_RED);
    tft.println("OBJECT DETECTED!");
  }
  else {
    tft.setTextColor(ST77XX_GREEN);
    tft.println("AREA CLEAR");
  }
  Serial.print("Angle: ");
  Serial.print(angle);
  Serial.print("  Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

Note

- The **5° step** can be changed to 1, 10, etc. Smaller steps give smoother scanning but take longer.
- The **50 cm** detection limit can be changed in the code.
- If the servo moves beyond the mechanical limit of your setup, reduce the scanning angle.
- If the TFT stays blank, first check the **CS, DC, RST, MOSI and SCK** connections and confirm the correct ST7735 initialization tab for your display.
- For a larger servo, use a separate suitable 5V supply and connect its **GND to Arduino GND**.

Project Name: **Visitor Counter**

- ❖ **Components Required**

1. Arduino UNO
2. IR Sensor Module
3. 1.8" TFT Display (ST7735)
4. Breadboard
5. Jumper Wires
6. USB Cable

- ❖ **Connections**

IR Sensor

IR Sensor Pin	**Arduino UNO**
VCC	5V
GND	GND
OUT	D2

1.8" ST7735 TFT Display

TFT Pin	**Arduino UNO**
VCC	5V*
GND	GND
SCL/CLK	D13
SDA/MOSI	D11
CS	D10
DC/A0	D9
RST	D8

*Check your TFT module's voltage requirement before connecting VCC.

Working

- The IR sensor detects when a person passes in front of it.
- Every time a person is detected, the Arduino increases the **visitor count by 1**.
- The current number of visitors is displayed on the **1.8" TFT screen**.
- A small delay is used so the same person is not counted multiple times while they remain in front of the sensor.

Arduino Code

Install:

- **Adafruit GFX Library**
- **Adafruit ST7735 and ST7789 Library**

```
#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Adafruit_ST7735.h>


// TFT pins

#define TFT_CS   10

#define TFT_DC   9

#define TFT_RST  8


Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);


// IR sensor

#define IR_PIN 2


int visitorCount = 0;

int lastIRState = HIGH;
```

```
void setup() {
  pinMode(IR_PIN, INPUT);

  Serial.begin(9600);

  // Start TFT
  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);

  displayCount();
}

void loop() {

  int irState = digitalRead(IR_PIN);

  // Most IR sensors give LOW when an object is detected
  if (irState == LOW && lastIRState == HIGH) {

    visitorCount++;

    Serial.print("Visitors: ");
    Serial.println(visitorCount);

    displayCount();
```

```
    delay(500);
  }

  lastIRState = irState;

  delay(50);
}

void displayCount() {

  tft.fillScreen(ST77XX_BLACK);

  tft.setTextColor(ST77XX_GREEN);
  tft.setTextSize(2);

  tft.setCursor(20, 25);
  tft.println("VISITOR");

  tft.setCursor(35, 50);
  tft.println("COUNTER");

  tft.setTextColor(ST77XX_WHITE);
  tft.setTextSize(3);

  tft.setCursor(55, 85);
```

```
  tft.print(visitorCount);
}
```

Note

- Most IR sensor modules give **LOW when an object is detected**. If your module gives HIGH, change the detection condition accordingly.
- The delay(500) helps prevent one person from being counted repeatedly.
- This project counts **each detection as one visitor**. For accurate **entry and exit counting**, two IR sensors should be used.
- If the TFT display is blank, check the **CS, DC, RST, MOSI and SCK** connections and the correct ST7735 initialization setting.

Project Name: Digital Weather Station

- ❖ **Components Required**

1. Arduino UNO
2. DHT11 Temperature & Humidity Sensor
3. Rain Sensor Module
4. 1.8" TFT Display (ST7735)
5. Breadboard
6. Jumper Wires
7. USB Cable

- ❖ **Connections**

DHT11 Sensor

DHT11 Pin	Arduino UNO
VCC	5V
GND	GND
DATA	D2

Rain Sensor Module

Rain Sensor Pin	Arduino UNO
VCC	5V
GND	GND
DO	D3

1.8" ST7735 TFT Display

TFT Pin	Arduino UNO
VCC	5V*
GND	GND

TFT Pin	Arduino UNO
SCL/CLK	D13
SDA/MOSI	D11
CS	D10
DC/A0	D9
RST	D8

*Check the voltage requirement printed on your TFT module.

Working

The Digital Weather Station measures and displays:

- **Temperature** in °C
- **Humidity** in %
- **Rain condition**
- **Simple weather prediction**

The DHT11 measures temperature and humidity, while the rain sensor detects whether rain is present.

The TFT displays all the information in real time.

Weather Prediction Logic

For a simple school-level prediction:

Condition	Prediction
Rain detected	**Rainy Weather**
No rain + Humidity ≥ 80%	**Cloudy / Rain Possible**
No rain + Humidity 60–79%	**Humid Weather**
No rain + Humidity < 60%	**Clear Weather**

Important: This is a **sensor-based indication**, not a true weather forecast. A rain sensor can detect current moisture/rain but cannot reliably predict future rainfall by itself.

Arduino Code

Install these libraries from **Arduino IDE → Library Manager**:

- **DHT sensor library**
- **Adafruit GFX Library**
- **Adafruit ST7735 and ST7789 Library**

```
#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Adafruit_ST7735.h>

#include <DHT.h>


// --------------------

// TFT Pins

// --------------------

#define TFT_CS   10

#define TFT_DC   9

#define TFT_RST  8


Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);


// --------------------

// DHT11

// --------------------

#define DHT_PIN 2

#define DHT_TYPE DHT11


DHT dht(DHT_PIN, DHT_TYPE);
```

```
// --------------------
// Rain Sensor
// --------------------
#define RAIN_PIN 3

void setup() {
  Serial.begin(9600);

  dht.begin();

  pinMode(RAIN_PIN, INPUT);

  // Start TFT
  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);

  tft.fillScreen(ST77XX_BLACK);

  tft.setTextColor(ST77XX_GREEN);
  tft.setTextSize(2);
  tft.setCursor(15, 5);
  tft.println("WEATHER");

  tft.setCursor(25, 25);
  tft.println("STATION");
```

```
  delay(2000);
}

void loop() {

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  int rainState = digitalRead(RAIN_PIN);

  // Check DHT reading
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("DHT11 Error!");
    delay(2000);
    return;
  }

  // Most rain sensor modules give LOW when rain is detected
  bool rainDetected = (rainState == LOW);

  // --------------------
  // Weather Prediction
  // --------------------
  String prediction;
```

```
if (rainDetected) {
  prediction = "Rainy";
}
else if (humidity >= 80) {
  prediction = "Cloudy";
}
else if (humidity >= 60) {
  prediction = "Humid";
}
else {
  prediction = "Clear";
}

// --------------------
// Serial Monitor
// --------------------
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" C");

Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println(" %");

Serial.print("Rain: ");
```

```
if (rainDetected) {
  Serial.println("Detected");
}
else {
  Serial.println("Not Detected");
}

Serial.print("Prediction: ");
Serial.println(prediction);

Serial.println("--------------------");

// --------------------
// TFT Display
// --------------------
tft.fillScreen(ST77XX_BLACK);

tft.setTextSize(2);
tft.setTextColor(ST77XX_CYAN);
tft.setCursor(20, 5);
tft.println("WEATHER");

tft.setTextSize(1);
tft.setTextColor(ST77XX_WHITE);

// Temperature
```

```
tft.setCursor(10, 35);
tft.print("Temperature: ");

tft.setTextColor(ST77XX_YELLOW);
tft.print(temperature, 1);
tft.println(" C");

// Humidity
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(10, 55);
tft.print("Humidity: ");

tft.setTextColor(ST77XX_CYAN);
tft.print(humidity, 1);
tft.println(" %");

// Rain
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(10, 75);
tft.print("Rain: ");

if (rainDetected) {
  tft.setTextColor(ST77XX_RED);
  tft.println("DETECTED");
}
else {
```

```
  tft.setTextColor(ST77XX_GREEN);
  tft.println("NO RAIN");
 }

 // Prediction
 tft.setTextColor(ST77XX_WHITE);
 tft.setCursor(10, 100);
 tft.print("Prediction:");

 tft.setCursor(10, 115);

 if (prediction == "Rainy") {
  tft.setTextColor(ST77XX_RED);
 }
 else if (prediction == "Cloudy") {
  tft.setTextColor(ST77XX_YELLOW);
 }
 else {
  tft.setTextColor(ST77XX_GREEN);
 }

 tft.println(prediction);

 delay(2000);
}
```

Note

- Most rain sensor modules give **LOW when rain is detected**. If yours works opposite, change:

bool rainDetected = (rainState == LOW);

to:

bool rainDetected = (rainState == HIGH);

- Use the small potentiometer on the rain sensor module to adjust its sensitivity.
- Keep the DHT11 away from direct water.
- The prediction is designed for a **school STEM demonstration** and should be presented as a simple sensor-based weather indication, not professional forecasting.
- If your TFT has a different ST7735 pin arrangement or initialization color tab, the display setup may need adjustment.

Project Name: Smart Parking Assistant

- **Components Required**

1. Arduino UNO
2. HC-SR04 Ultrasonic Sensor
3. 1.8" TFT ST7735 Display
4. Active Buzzer
5. Breadboard
6. Jumper Wires
7. USB Cable

- **Connections**

HC-SR04

Sensor Pin	Arduino UNO
VCC	5V
GND	GND
TRIG	D7
ECHO	D8

1.8" TFT ST7735

TFT Pin	Arduino UNO
VCC	5V*
GND	GND
SCL/CLK	D13
SDA/MOSI	D11
CS	D10
DC/A0	D9

TFT Pin	Arduino UNO
RST	D6

Active Buzzer

Buzzer	Arduino UNO
+	D4
–	GND

*Check your particular TFT module's voltage requirement.

Working

The ultrasonic sensor measures the distance between the car and the parking obstacle.

Distance	Parking Status	Buzzer
> 50 cm	PARKING AVAILABLE	OFF
20–50 cm	MOVE SLOWLY	Slow beep
< 20 cm	STOP!	Fast beep

The TFT displays the **distance and parking status**.

Arduino Code

Install these libraries:

- **Adafruit GFX Library**
- **Adafruit ST7735 and ST7789 Library**

```
#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Adafruit_ST7735.h>


#define TFT_CS   10

#define TFT_DC   9

#define TFT_RST  6
```

```
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

#define TRIG_PIN 7
#define ECHO_PIN 8
#define BUZZER   4

long duration;
int distance;

void setup() {
  Serial.begin(9600);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER, OUTPUT);

  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
  tft.fillScreen(ST77XX_BLACK);

  tft.setTextColor(ST77XX_GREEN);
  tft.setTextSize(2);
  tft.setCursor(15, 10);
  tft.println("SMART");
```

```
  tft.setCursor(15, 35);
  tft.println("PARKING");

  delay(2000);
}

void loop() {

  // Ultrasonic trigger
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH, 30000);

  if (duration == 0) {
    distance = 400;
  } else {
    distance = duration * 0.034 / 2;
  }

  Serial.print("Distance: ");
```

```
Serial.print(distance);
Serial.println(" cm");

// Clear display
tft.fillScreen(ST77XX_BLACK);

tft.setTextSize(2);
tft.setTextColor(ST77XX_CYAN);
tft.setCursor(10, 5);
tft.println("PARKING");

tft.setTextSize(1);
tft.setTextColor(ST77XX_WHITE);

tft.setCursor(10, 35);
tft.print("Distance: ");

tft.setTextColor(ST77XX_YELLOW);
tft.print(distance);
tft.println(" cm");

// Parking conditions
if (distance > 50) {

  digitalWrite(BUZZER, LOW);
```

```
  tft.setTextColor(ST77XX_GREEN);
  tft.setCursor(10, 70);
  tft.println("PARKING");

  tft.setCursor(10, 85);
  tft.println("AVAILABLE");

}

else if (distance >= 20 && distance <= 50) {

  digitalWrite(BUZZER, HIGH);
  delay(100);
  digitalWrite(BUZZER, LOW);

  tft.setTextColor(ST77XX_YELLOW);
  tft.setCursor(10, 70);
  tft.println("MOVE");

  tft.setCursor(10, 85);
  tft.println("SLOWLY");

}

else {
```

```
   digitalWrite(BUZZER, HIGH);


   tft.setTextColor(ST77XX_RED);
   tft.setCursor(10, 70);
   tft.println("STOP!");


   tft.setCursor(10, 85);
   tft.println("TOO CLOSE");
  }


  delay(200);
}
```

Note

- Mount the **HC-SR04 facing the parking obstacle**.
- The distance limits can be changed according to your model.
- An **active buzzer** can be controlled directly with HIGH/LOW.
- This is a **school-level parking assistance model**, not a vehicle safety system.

Project Name: **TFT Information Dashboard**

- ❖ **Components Required**

1. Arduino UNO
2. 1.8" TFT ST7735 Display
3. Soil Moisture Sensor
4. Water Level Sensor
5. DHT11 Temperature & Humidity Sensor
6. Breadboard
7. Jumper Wires
8. USB Cable

- ❖ **Connections**

Soil Moisture Sensor

Pin	**Arduino UNO**
VCC	5V
GND	GND
AO	A0

Water Level Sensor

Pin	**Arduino UNO**
VCC	5V
GND	GND
SIG/AO	A1

DHT11

Pin	**Arduino UNO**
VCC	5V

Pin	Arduino UNO
GND	GND
DATA	D2

1.8" TFT ST7735

TFT Pin	Arduino UNO
VCC	5V*
GND	GND
SCL/CLK	D13
SDA/MOSI	D11
CS	D10
DC/A0	D9
RST	D8

*Check the voltage requirement of your particular TFT module.

Working

The project collects information from three different sensors and displays it on the TFT screen:

- **Temperature** – DHT11
- **Humidity** – DHT11
- **Soil Moisture** – Soil sensor
- **Water Level** – Water level sensor

The TFT continuously updates the readings and shows the condition of the soil and water.

Arduino Code

Install these libraries in Arduino IDE:

- **DHT sensor library**
- **Adafruit GFX Library**

- **Adafruit ST7735 and ST7789 Library**

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <DHT.h>

// TFT pins
#define TFT_CS   10
#define TFT_DC   9
#define TFT_RST  8

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

// DHT11
#define DHT_PIN 2
#define DHT_TYPE DHT11

DHT dht(DHT_PIN, DHT_TYPE);

// Sensors
#define SOIL_PIN A0
#define WATER_PIN A1

void setup() {
  Serial.begin(9600);
```

```
  dht.begin();

  pinMode(SOIL_PIN, INPUT);
  pinMode(WATER_PIN, INPUT);

  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
  tft.fillScreen(ST77XX_BLACK);

  tft.setTextColor(ST77XX_GREEN);
  tft.setTextSize(2);
  tft.setCursor(10, 5);
  tft.println("DASHBOARD");

  delay(2000);
}

void loop() {

  // Read sensors
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  int soilValue = analogRead(SOIL_PIN);
  int waterValue = analogRead(WATER_PIN);
```

```
// Check DHT11
if (isnan(temperature) || isnan(humidity)) {
  Serial.println("DHT11 Error!");
  delay(2000);
  return;
}

// Soil condition
String soilStatus;

if (soilValue > 700) {
  soilStatus = "DRY";
}
else if (soilValue > 400) {
  soilStatus = "MOIST";
}
else {
  soilStatus = "WET";
}

// Water condition
String waterStatus;

if (waterValue < 300) {
  waterStatus = "LOW";
}
```

```
else if (waterValue < 600) {
  waterStatus = "MEDIUM";
}
else {
  waterStatus = "HIGH";
}

// Serial Monitor
Serial.println("----------------------");

Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" C");

Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println(" %");

Serial.print("Soil Value: ");
Serial.println(soilValue);

Serial.print("Soil Status: ");
Serial.println(soilStatus);

Serial.print("Water Value: ");
Serial.println(waterValue);
```

```
Serial.print("Water Status: ");
Serial.println(waterStatus);

// Clear TFT
tft.fillScreen(ST77XX_BLACK);

// Title
tft.setTextColor(ST77XX_CYAN);
tft.setTextSize(2);
tft.setCursor(15, 5);
tft.println("DASHBOARD");

// Temperature
tft.setTextSize(1);
tft.setTextColor(ST77XX_WHITE);

tft.setCursor(5, 30);
tft.print("Temp: ");

tft.setTextColor(ST77XX_YELLOW);
tft.print(temperature, 1);
tft.println(" C");

// Humidity
tft.setTextColor(ST77XX_WHITE);
```

```
tft.setCursor(85, 30);
tft.print("Hum: ");

tft.setTextColor(ST77XX_CYAN);
tft.print(humidity, 1);
tft.println(" %");

// Soil
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(5, 55);
tft.print("Soil: ");

tft.setTextColor(ST77XX_GREEN);
tft.println(soilStatus);

tft.setTextColor(ST77XX_WHITE);
tft.setCursor(85, 55);
tft.print("Value: ");
tft.println(soilValue);

// Water
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(5, 80);
tft.print("Water: ");

tft.setTextColor(ST77XX_BLUE);
```

```
  tft.println(waterStatus);


  tft.setTextColor(ST77XX_WHITE);
  tft.setCursor(85, 80);
  tft.print("Value: ");
  tft.println(waterValue);


  // Overall status
  tft.setCursor(5, 110);
  tft.setTextColor(ST77XX_GREEN);
  tft.println("SYSTEM MONITORING");


  delay(2000);
}
```

Note: Soil and water sensor values vary depending on the actual sensors and environment. These threshold values are starting values; check the readings in the Serial Monitor and adjust them for your model.

Project Name: Interactive Sensor Monitor

Components Required

1. Arduino UNO
2. Flame Sensor Module
3. Soil Moisture Sensor
4. LDR Module
5. 1.8" TFT ST7735 Display
6. Breadboard
7. Jumper Wires
8. USB Cable

Connections

Flame Sensor

Pin	**Arduino UNO**
VCC	5V
GND	GND
DO	D2

Soil Moisture Sensor

Pin	**Arduino UNO**
VCC	5V
GND	GND
AO	A0

LDR Module

Pin	**Arduino UNO**
VCC	5V

Pin	Arduino UNO
GND	GND
AO	A1

1.8" TFT ST7735

TFT Pin	Arduino UNO
VCC	5V*
GND	GND
SCL/CLK	D13
SDA/MOSI	D11
CS	D10
DC/A0	D9
RST	D8

*Check the voltage requirement of your TFT module.

Working

The Arduino continuously monitors **three environmental conditions**:

- **Flame Sensor** → detects fire/flame
- **Soil Sensor** → checks soil moisture
- **LDR** → measures surrounding light

The TFT displays all sensor readings and their current status.

Status Logic

Sensor	Condition	Status
Flame	Flame detected	**FIRE ALERT**
Flame	No flame	**SAFE**
Soil	Value > 700	**DRY**

Sensor	Condition	Status
Soil	401–700	**MOIST**
Soil	≤400	**WET**
LDR	Value < 400	**DARK**
LDR	400–700	**NORMAL**
LDR	>700	**BRIGHT**

Arduino Code

Install:

- **Adafruit GFX Library**
- **Adafruit ST7735 and ST7789 Library**

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>

// TFT pins
#define TFT_CS   10
#define TFT_DC   9
#define TFT_RST  8

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

// Sensor pins
#define FLAME_PIN 2
#define SOIL_PIN  A0
#define LDR_PIN   A1
```

```
void setup() {
  Serial.begin(9600);

  pinMode(FLAME_PIN, INPUT);
  pinMode(SOIL_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);

  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
  tft.fillScreen(ST77XX_BLACK);

  tft.setTextColor(ST77XX_GREEN);
  tft.setTextSize(2);
  tft.setCursor(10, 10);
  tft.println("SENSOR");
  tft.setCursor(10, 35);
  tft.println("MONITOR");

  delay(2000);
}

void loop() {

  // Read sensors
  int flameState = digitalRead(FLAME_PIN);
```

```
int soilValue = analogRead(SOIL_PIN);
int ldrValue = analogRead(LDR_PIN);

// Flame status
String flameStatus;

if (flameState == LOW) {
  flameStatus = "FIRE ALERT";
} else {
  flameStatus = "SAFE";
}

// Soil status
String soilStatus;

if (soilValue > 700) {
  soilStatus = "DRY";
}
else if (soilValue > 400) {
  soilStatus = "MOIST";
}
else {
  soilStatus = "WET";
}

// Light status
```

```
String lightStatus;

if (ldrValue < 400) {
  lightStatus = "DARK";
}
else if (ldrValue <= 700) {
  lightStatus = "NORMAL";
}
else {
  lightStatus = "BRIGHT";
}

// Serial Monitor
Serial.println("--------------------");

Serial.print("Flame: ");
Serial.println(flameStatus);

Serial.print("Soil: ");
Serial.print(soilValue);
Serial.print(" - ");
Serial.println(soilStatus);

Serial.print("LDR: ");
Serial.print(ldrValue);
Serial.print(" - ");
```

```
Serial.println(lightStatus);

// TFT
tft.fillScreen(ST77XX_BLACK);

// Title
tft.setTextSize(2);
tft.setTextColor(ST77XX_CYAN);
tft.setCursor(15, 5);
tft.println("SENSOR MONITOR");

tft.setTextSize(1);

// Flame
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(5, 35);
tft.print("Flame: ");

if (flameState == LOW) {
  tft.setTextColor(ST77XX_RED);
} else {
  tft.setTextColor(ST77XX_GREEN);
}

tft.println(flameStatus);
```

```
// Soil
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(5, 60);
tft.print("Soil: ");

tft.setTextColor(ST77XX_YELLOW);
tft.print(soilValue);

tft.setTextColor(ST77XX_WHITE);
tft.print("  ");
tft.println(soilStatus);

// LDR
tft.setTextColor(ST77XX_WHITE);
tft.setCursor(5, 85);
tft.print("Light: ");

tft.setTextColor(ST77XX_CYAN);
tft.print(ldrValue);

tft.setTextColor(ST77XX_WHITE);
tft.print("  ");
tft.println(lightStatus);

// Overall status
tft.setCursor(5, 115);
```

```
 if (flameState == LOW) {
   tft.setTextColor(ST77XX_RED);
   tft.println("WARNING: FIRE!");
 }
 else {
   tft.setTextColor(ST77XX_GREEN);
   tft.println("SYSTEM NORMAL");
 }


 delay(1000);
}
```

Note

- Most flame sensor modules give **LOW when flame is detected**. If yours works opposite, change flameState == LOW to flameState == HIGH.
- Soil and LDR values vary according to the sensor and environment, so the thresholds should be calibrated using the **Serial Monitor**.
- The TFT makes the project interactive by giving students a live visual dashboard of all three sensors.

Project Name: RFID Access Gate

- ❖ **Components Required**

1. Arduino UNO
2. RFID Sensor – RC522
3. RFID Card/Key Tag
4. SG90 Servo Motor
5. 1.8" TFT ST7735 Display
6. Breadboard
7. Jumper Wires
8. USB Cable

- ❖ **Connections**

RFID RC522

RC522 Pin	**Arduino UNO**
3.3V	3.3V
GND	GND
SDA/SS	D10
SCK	D13
MOSI	D11
MISO	D12
RST	D7

Important: RC522 must be powered with **3.3V**, not 5V.

Servo Motor

Servo Wire	**Arduino UNO**
Signal (Orange/Yellow)	D6

Servo Wire	Arduino UNO
VCC (Red)	5V
GND (Brown/Black)	GND

1.8" TFT ST7735

TFT Pin	Arduino UNO
VCC	5V*
GND	GND
SCL/CLK	D13
SDA/MOSI	D11
CS	D9
DC/A0	D8
RST	D5

*Check your TFT module's voltage requirement.

Working

The RFID reader scans the RFID card/tag.

- **Authorized card** → TFT shows **ACCESS GRANTED** → servo opens the gate.
- **Unauthorized card** → TFT shows **ACCESS DENIED** → gate remains closed.

The servo can be used as a small parking/security gate.

Libraries Required

Install these from **Arduino IDE → Library Manager**:

- MFRC522
- Adafruit GFX Library
- Adafruit ST7735 and ST7789 Library
- Servo

Arduino Code

```
#include <SPI.h>

#include <MFRC522.h>

#include <Servo.h>

#include <Adafruit_GFX.h>

#include <Adafruit_ST7735.h>


// ---------------- RFID ----------------

#define SS_PIN 10

#define RST_PIN 7


MFRC522 rfid(SS_PIN, RST_PIN);


// ---------------- SERVO ----------------

#define SERVO_PIN 6


Servo gateServo;


// ---------------- TFT ----------------

#define TFT_CS   9

#define TFT_DC   8

#define TFT_RST  5


Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);


// ------------------------------------------------
```

```
// Replace this UID with your RFID card UID
// Check Serial Monitor to find your UID.
// Example UID: DE AD BE EF
// ------------------------------------------------

byte authorizedUID[] = {0xDE, 0xAD, 0xBE, 0xEF};

void setup() {

  Serial.begin(9600);

  // RFID
  SPI.begin();
  rfid.PCD_Init();

  // Servo
  gateServo.attach(SERVO_PIN);
  gateServo.write(0);

  // TFT
  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
  tft.fillScreen(ST77XX_BLACK);

  showMessage("RFID GATE", "SCAN CARD");
```

```
  Serial.println("RFID Access Gate Ready");
  Serial.println("Scan your RFID card...");
}

void loop() {

  // Check for new RFID card
  if (!rfid.PICC_IsNewCardPresent()) {
    return;
  }

  if (!rfid.PICC_ReadCardSerial()) {
    return;
  }

  // Print UID
  Serial.print("Card UID: ");

  for (byte i = 0; i < rfid.uid.size; i++) {

    Serial.print(rfid.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(rfid.uid.uidByte[i], HEX);
  }

  Serial.println();
```

```
// Check authorization
if (checkUID()) {

  Serial.println("ACCESS GRANTED");

  showMessage("ACCESS", "GRANTED");

  delay(500);

  // Open gate
  gateServo.write(90);

  showMessage("GATE", "OPEN");

  delay(3000);

  // Close gate
  gateServo.write(0);

  showMessage("RFID GATE", "SCAN CARD");

}
else {

  Serial.println("ACCESS DENIED");
```

```
    showMessage("ACCESS", "DENIED");

    delay(2000);

    showMessage("RFID GATE", "SCAN CARD");
  }

  // Stop RFID communication
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();

  delay(500);
}

// ------------------------------------------------
// Check RFID UID
// ------------------------------------------------

bool checkUID() {

  if (rfid.uid.size != sizeof(authorizedUID)) {
    return false;
  }

  for (byte i = 0; i < rfid.uid.size; i++) {
```

```
    if (rfid.uid.uidByte[i] != authorizedUID[i]) {
      return false;
    }
  }

  return true;
}

// ------------------------------------------------
// TFT Display Function
// ------------------------------------------------

void showMessage(String line1, String line2) {

  tft.fillScreen(ST77XX_BLACK);

  tft.setTextSize(2);

  tft.setTextColor(ST77XX_CYAN);
  tft.setCursor(15, 15);
  tft.println(line1);

  tft.setTextColor(ST77XX_WHITE);
  tft.setCursor(15, 55);
  tft.println(line2);
}
```

Note

- RC522 uses **3.3V**.
- If the servo causes the Arduino to reset, use a separate suitable 5V supply for the servo and connect the grounds together.
- This project demonstrates RFID-based access control; it is a **school STEM model**, not a security-grade access system.

Project Name **Human Following Car**

❖ **Components Required**

- Arduino UNO
- Motor Driver (L298N)
- HC-SR04 Ultrasonic Sensor
- 2 × BO Motors
- Robot chassis with wheels
- Battery
- Jumper wires

❖ **Connections**

HC-SR04 → Arduino UNO

HC-SR04	Arduino UNO
VCC	5V
GND	GND
TRIG	D9
ECHO	D10

L298N → Arduino UNO

L298N	Arduino UNO
IN1	D4
IN2	D5
IN3	D6
IN4	D7
GND	GND

Motors

- Left BO Motor → OUT1, OUT2
- Right BO Motor → OUT3, OUT4

Important: Connect the Arduino GND and motor-driver GND together. Power the motors using the battery through the L298N, not directly from the Arduino 5V pin.

Working

The ultrasonic sensor continuously measures the distance between the person and the car.

- **More than 80 cm:** Car moves forward to follow the person.
- **40–80 cm:** Car moves slowly/maintains position.
- **Less than 40 cm:** Car stops to avoid getting too close.

With only **one ultrasonic sensor**, this project can follow a person mainly in a straight line. It cannot reliably determine whether the person is to the left or right.

Arduino Code

```
#define TRIG_PIN 9

#define ECHO_PIN 10


// Motor Driver Pins

#define IN1 4

#define IN2 5

#define IN3 6

#define IN4 7


long duration;

int distance;


void setup() {

  Serial.begin(9600);
```

```
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  stopCar();
}

void loop() {

  distance = getDistance();

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance > 80 && distance < 300) {
    // Person is far → move forward
    forward();
  }
  else if (distance >= 40 && distance <= 80) {
    // Person is at a suitable distance
    forwardSlow();
```

```
  }
  else if (distance < 40) {
    // Person is too close → stop
    stopCar();
  }
  else {
    // No person/object detected
    stopCar();
  }

  delay(100);
}

// Measure distance
int getDistance() {

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH, 30000);

  if (duration == 0) {
```

```
    return 400;
  }

  return duration * 0.034 / 2;
}

// Move forward
void forward() {

  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

// Slow/controlled forward
void forwardSlow() {

  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
```

```
// Stop
void stopCar() {


  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);


  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

Note

If one motor rotates in the wrong direction, **swap the two wires of that motor** at the L298N output. The distance values (40 cm and 80 cm) can also be adjusted according to the required following distance.