



Enhancing Hand Gesture Recognition Utilizing the PAJ7620 Sensor and Arduino



by Maker_Superman

In this project, we'll enhance a Hand Gesture Recognition System using the PAJ7620 Sensor and an Arduino Board. The PAJ7620 sensor is capable of recognizing nine distinct gestures, including moving up, down, left, right, forward, backward, clockwise, anti-clockwise, and waving. This system can find extensive use in medical applications, or for controlling lights, robots, human-machine interfaces, and games by leveraging the power of an IR LED and optical CMOS array.

Gesture Recognition is a technology that capitalizes on sensors to interpret hand movements as distinct commands. While gestures can originate from any body motion or state, they are typically derived from the face or hands. A gesture, by definition, must be visible to another party and must convey specific information. Thus, Gesture Recognition can be viewed as a means for computers to understand and respond to human body language.

Previously, we delved into the APDS9960 Gesture Recognition and RGB Color Sensor.

Supplies:

Bill of Materials

[Arduino Nano Board](#)

[PAJ7620 Gesture Recognition Sensor](#)

[16x2 I2C LCD Display](#)

[Jumper Wires](#)

[Breadboard](#)



Step 1: PAJ7620 Gesture Recognition Sensor

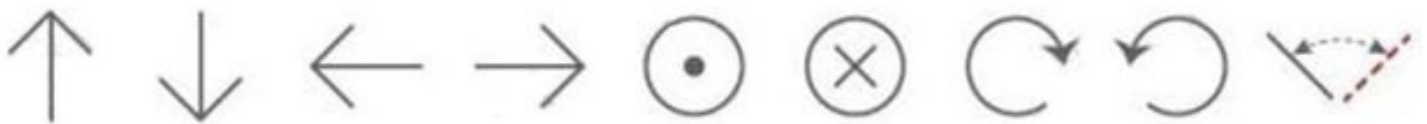
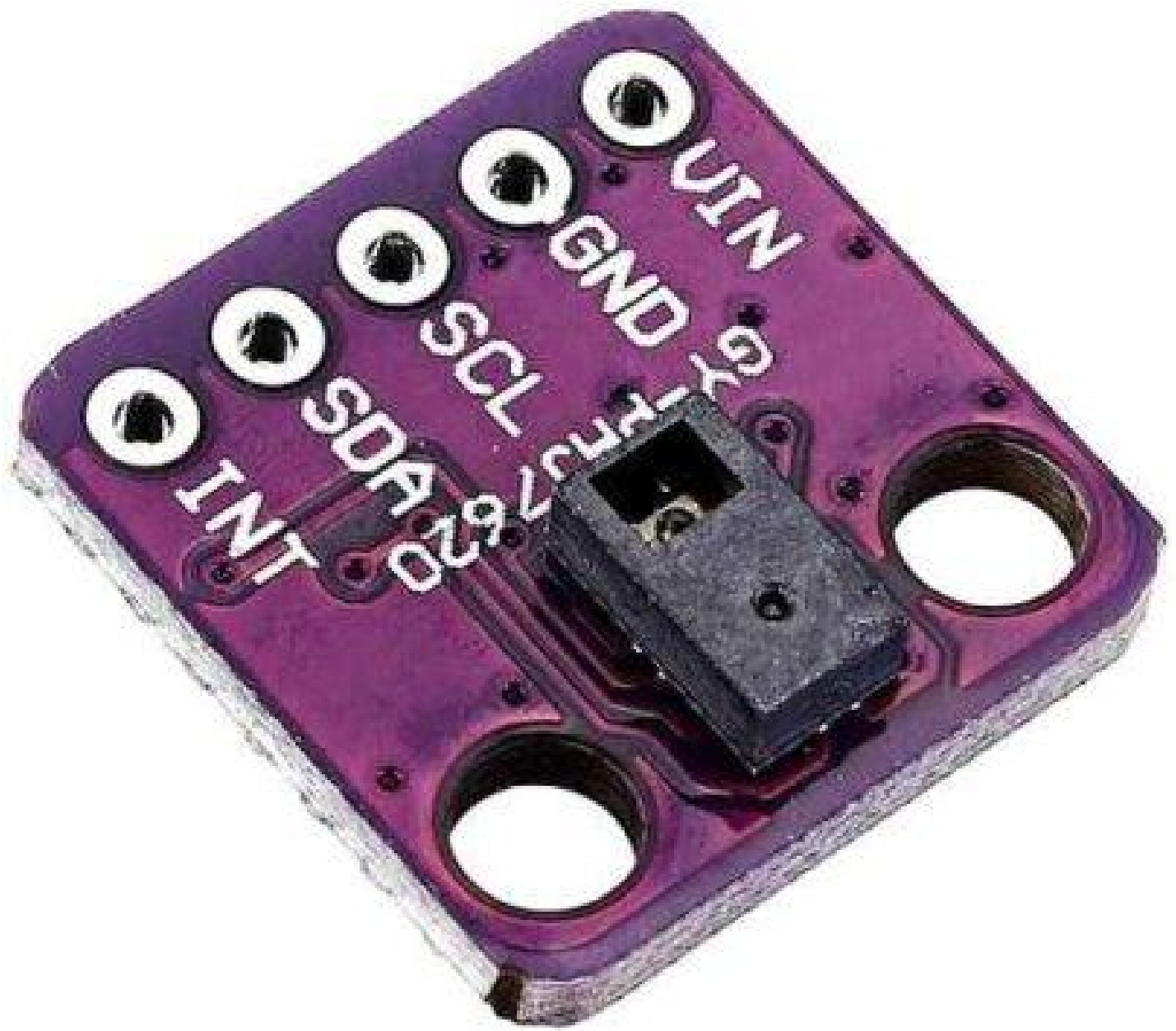
This product utilizes the sophisticated PAJ7620U2 gesture recognition sensor, which can detect nine distinct gestures in various directions, including up, down, left, right, front, back, clockwise, counterclockwise, and wave.

PAJ7620 Gesture Recognition Sensor

The gesture recognition module communicates via the I2C interface, enabling it to be programmed and controlled using the dedicated PAJ7620 Arduino library function. Signals emitted by this module can be harnessed as control signals for robots, facilitating seamless robot operation.

Our product incorporates an intelligent recognition algorithm that liberates users from conventional and limiting button-based controls. Gesture recognition sensors are well-suited for contactless control applications such as touch-free mice, smart homes, vehicle control device interactions, robot interfaces, and more.

Additionally, the PAJ7620U2 features an integrated proximity detection function, enabling it to sense objects as they approach or move away. The PAJ7620U2 boasts a flexible power-saving design, operating efficiently between 2.8V to 3.3V in a temperature range of -40°C to +85°C. The pull-up voltage for the I2C bus and the interrupt line ranges from 1.8V to 3.3V.



Step 2: Features of PAJ7620

- Nine gesture recognition (Up / Down / Left / Right / Push / Pull / CW / CCW / Wave)
- Gesture speed is 60°/s to 600°/s in Normal Mode and 60°/s to 1200°/s in Gaming Mode
- The supply voltage is 2.8V- 3.3V and I/O voltage is 1.8V~3.3V
- Working Current: 3mA-10mA
- Detection distance: 50-100mm
- Ambient light immunity: < 100k Lux
- Built-in proximity detection

- Flexible power-saving scheme
- I2C interface up to 400 kbit/s, Pull-up voltage from 1.8V to 3.3V
- Ambient light noise cancellation

PAJ7620 Applications

- Smart Home
- Offices and teaching
- Human-robot interaction
- Gesture toys
- Domatosensory game equipment
- PAD Phone
- Tablet Personal Computer
- Automobile Application

PAJ7620 Pinout

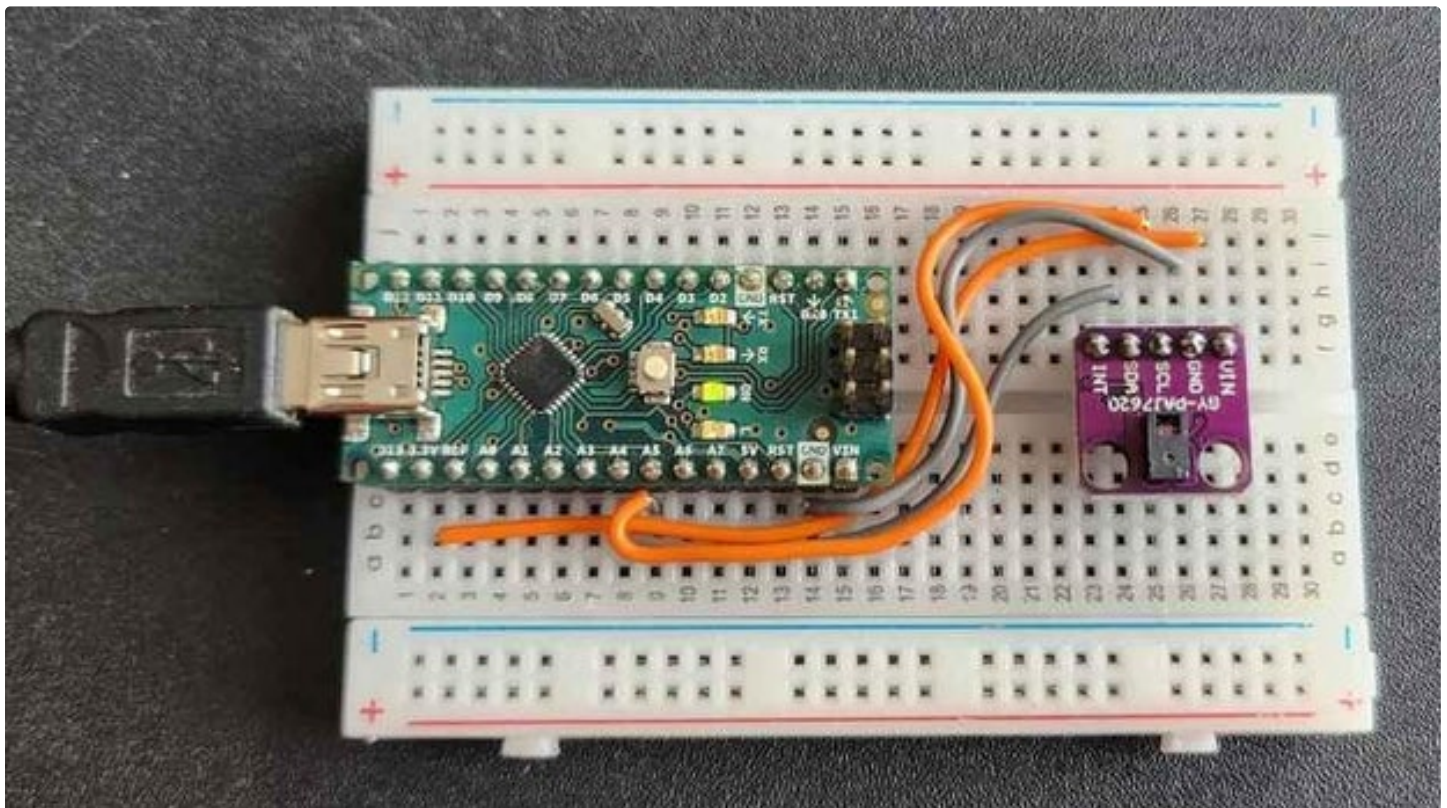
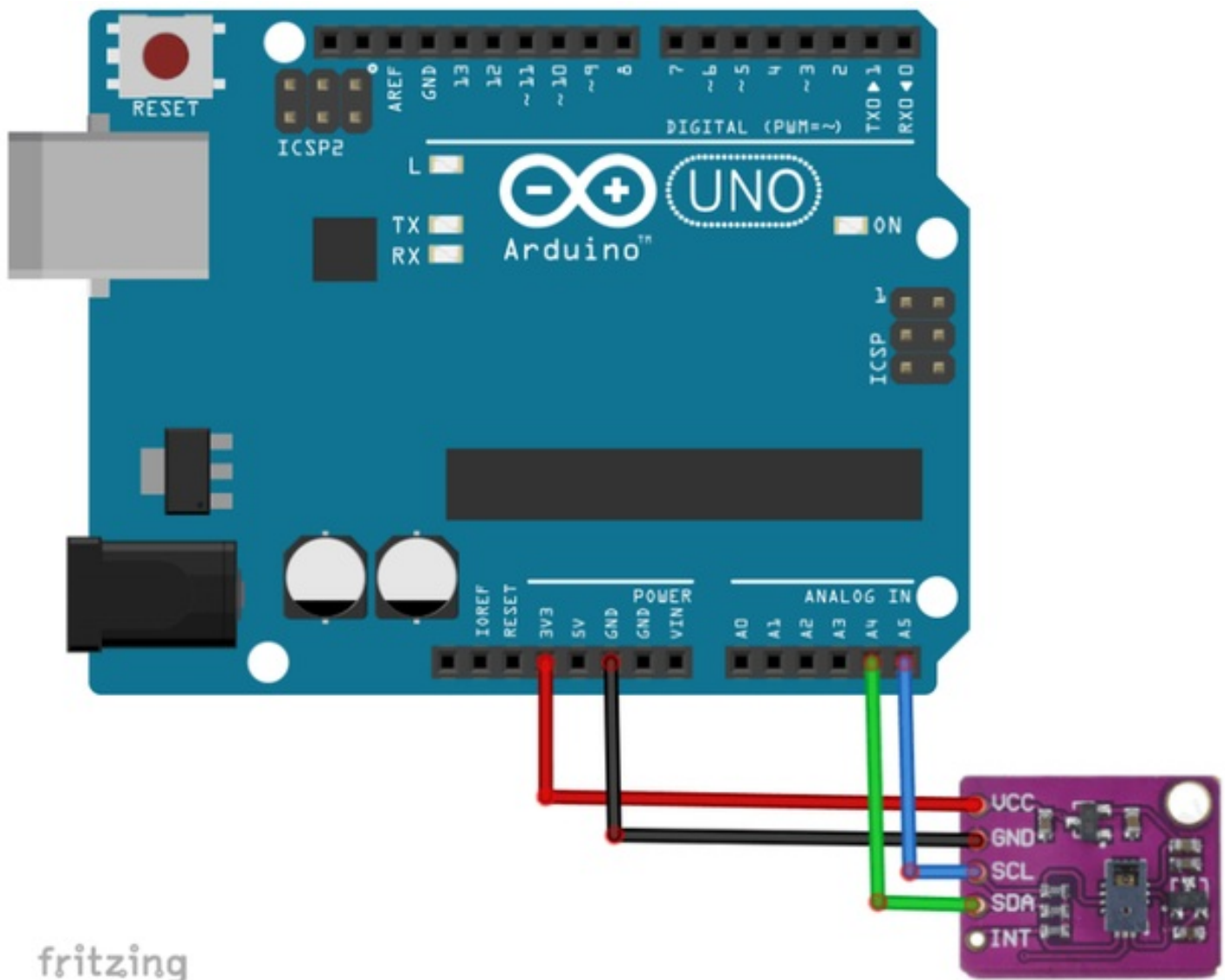
PIN	SYMBOL	Description
1	VCC	3.3V/5V
2	GND	Ground
3	SDA	I2C data pin
4	SCL	I2C clock pin
5	INT	External interrupt pin

Step 3: Interfacing PAJ7620 Gesture Recognition Sensor With Arduino

Let us interface the PAJ7620 Gesture Recognition Sensor with Arduino using the simple I2C interface.

The setup process is straightforward. Simply connect the VCC, GND, SDA, and SCL pins of the PAJ7620 to the corresponding 3.3V, GND, A4, and A5 pins on the Arduino, respectively.

In my instance, I've opted for the Arduino Nano board, but feel free to use any other Arduino-compatible microcontroller board according to your preference or project needs.



Step 4: Source Code/Program

Various libraries can facilitate the interfacing of the PAJ7620 Sensor with Arduino. Notably, Seeed Studio has authored one such library. You can download this library using the provided link.

https://github.com/Seeed-Studio/Gesture_PAJ7620

The subsequent code snippet is sourced from the library's example files. To implement it, copy and paste the code into your Arduino IDE, then proceed to upload it.

```
#include <Wire.h>
#include "paj7620.h"

/*
  Notice: When you want to recognize the Forward/Backward gestures, your gestures' reaction time must less than GES_ENTRY_TIME(0.8s).
  You also can adjust the reaction time according to the actual circumstance.
*/
#define GES_REACTION_TIME    500      // You can adjust the reaction time according to the actual circumstance.
#define GES_ENTRY_TIME      800      // When you want to recognize the Forward/Backward gestures, your gestures' reaction time must less than GES_ENTRY_TIME
(0.8s).
#define GES_QUIT_TIME       1000

void setup() {
  uint8_t error = 0;

  Serial.begin(9600);
  Serial.println("\nPAJ7620U2 TEST DEMO: Recognize 9 gestures.");

  error = paj7620Init();      // initialize Paj7620 registers
  if (error) {
    Serial.print("INIT ERROR.CODE:");
    Serial.println(error);
  } else {
    Serial.println("INIT OK");
  }
  Serial.println("Please input your gestures:\n");
}

void loop() {
  uint8_t data = 0, data1 = 0, error;

  error = paj7620ReadReg(0x43, 1, &data);      // Read Bank_0_Reg_0x43/0x44 for gesture result.
  if (!error) {
    switch (data) {      // When different gestures be detected, the variable 'data' will be set to different values by paj7620ReadReg(0x43, 1, &data).
      case GES_RIGHT_FLAG:
        delay(GES_ENTRY_TIME);
        paj7620ReadReg(0x43, 1, &data);
        if (data == GES_FORWARD_FLAG) {
          Serial.println("Forward");
          delay(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
          Serial.println("Backward");
          delay(GES_QUIT_TIME);
        } else {
          Serial.println("Right");
        }
        break;
      case GES_LEFT_FLAG:
        delay(GES_ENTRY_TIME);
        paj7620ReadReg(0x43, 1, &data);
        if (data == GES_FORWARD_FLAG) {
          Serial.println("Forward");
          delay(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
          Serial.println("Backward");
          delay(GES_QUIT_TIME);
        } else {
          Serial.println("Left");
        }
        break;
    }
  }
}
```

```
case GES_UP_FLAG:
    delay(GES_ENTRY_TIME);
    paj7620ReadReg(0x43, 1, &data);
    if (data == GES_FORWARD_FLAG) {
        Serial.println("Forward");
        delay(GES_QUIT_TIME);
    } else if (data == GES_BACKWARD_FLAG) {
        Serial.println("Backward");
        delay(GES_QUIT_TIME);
    } else {
        Serial.println("Up");
    }
    break;
case GES_DOWN_FLAG:
    delay(GES_ENTRY_TIME);
    paj7620ReadReg(0x43, 1, &data);
    if (data == GES_FORWARD_FLAG) {
        Serial.println("Forward");
        delay(GES_QUIT_TIME);
    } else if (data == GES_BACKWARD_FLAG) {
        Serial.println("Backward");
        delay(GES_QUIT_TIME);
    } else {
        Serial.println("Down");
    }
    break;
case GES_FORWARD_FLAG:
    Serial.println("Forward");
    delay(GES_QUIT_TIME);
    break;
case GES_BACKWARD_FLAG:
    Serial.println("Backward");
    delay(GES_QUIT_TIME);
    break;
case GES_CLOCKWISE_FLAG:
    Serial.println("Clockwise");
    break;
case GES_COUNT_CLOCKWISE_FLAG:
    Serial.println("anti-clockwise");
    break;
default:
    paj7620ReadReg(0x44, 1, &data1);
    if (data1 == GES_WAVE_FLAG) {
        Serial.println("wave");
    }
    break;
}
}
delay(100);
}
```

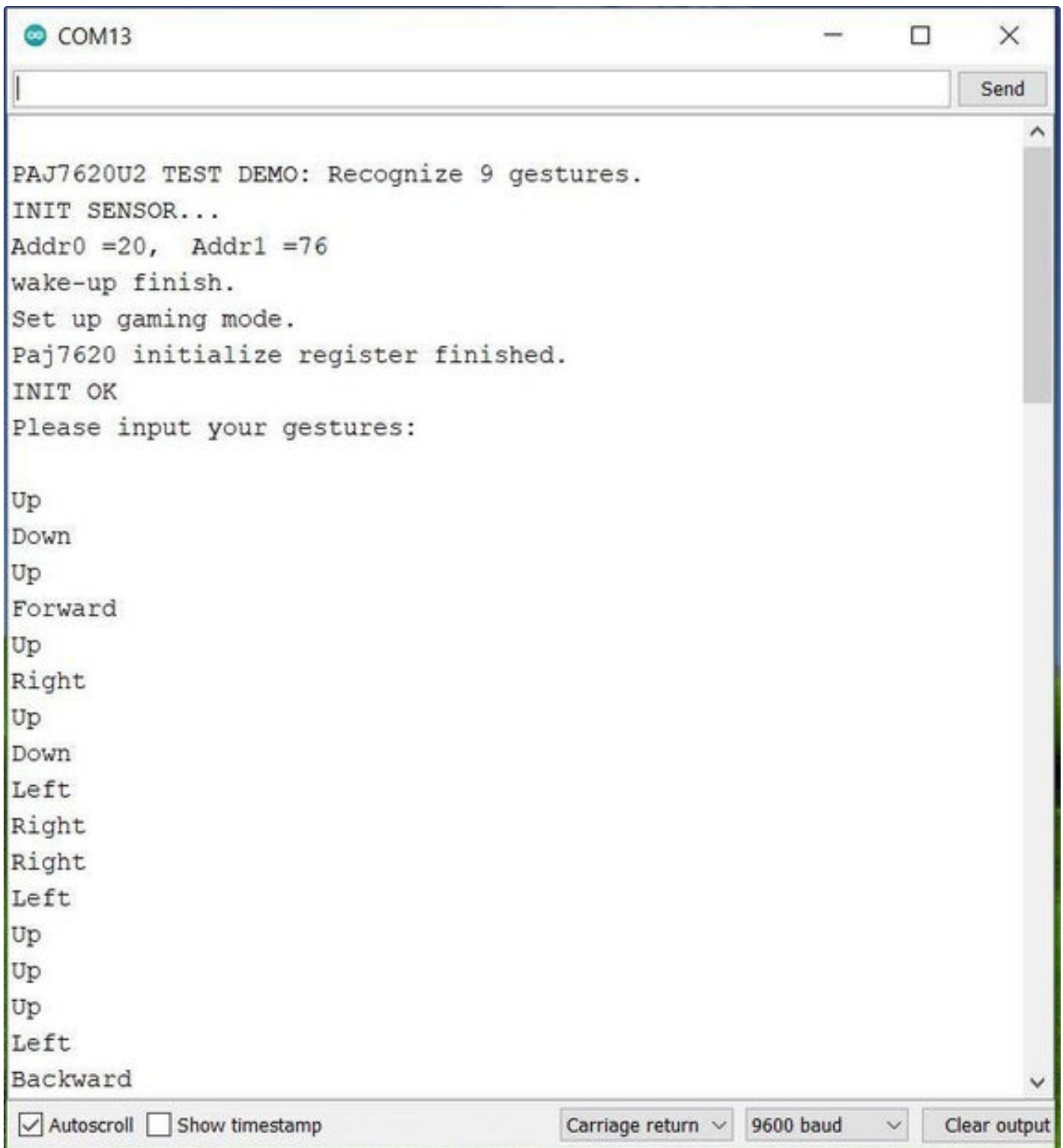
Once the code has been uploaded, launch the Serial Monitor. Experiment by moving your hand in varying directions and forming different gestures. The corresponding gesture indications will appear on the Serial Monitor. The testing process is intuitive – simply plug in the device and start moving your hands to see the results.

The system offers two modes of detection:

1. Near mode, which operates within a 5-15 cm distance at a 60° angle.
2. Far mode, effective within a 15-30 cm distance at a 30° angle.

You have the flexibility to adjust the detection speed universally or exclusively for forward and backward movements. To ensure optimal functionality, place the module on a flat surface. Failure to do so may result in clockwise and anti-clockwise gestures not being accurately detected.

```
#define GES_REACTION_TIME    500
#define GES_ENTRY_TIME      800
#define GES_QUIT_TIME       1000
```



```
COM13

PAJ7620U2 TEST DEMO: Recognize 9 gestures.
INIT SENSOR...
Addr0 =20,  Addr1 =76
wake-up finish.
Set up gaming mode.
Paj7620 initialize register finished.
INIT OK
Please input your gestures:

Up
Down
Up
Forward
Up
Right
Up
Down
Left
Right
Right
Left
Up
Up
Up
Left
Backward

☒ Autoscroll ☐ Show timestamp
Carriage return 9600 baud Clear output
```

Step 5: Displaying Gestures on 16×2 LCD Display

In the following segment, I'll be incorporating an LCD i²c display to visually present the detected gesture. This approach can be easily adapted to suit any project you're currently working on.

Connect the I2C LCD Display to the I2C Pin on the Arduino and power the display using the 5V Pin provided by the Arduino board.

Source Code/Program

Copy the following code and upload it to the Arduino Board.

```
#include <Wire.h>
#include "paj7620.h"
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

#define GES_REACTION_TIME 300 // You can adjust the reaction time according to the actual circumstance.
#define GES_ENTRY_TIME 500 // When you want to recognize the Forward/Backward gestures, your gestures' reaction time must less than GES_ENTRY_TIME(0.8s)
.
#define GES_QUIT_TIME 1000

void setup()
{
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);

  Serial.println("\nPAJ7620U2 TEST DEMO: Recognize 9 gestures.");

  lcd.setCursor(0, 0);
  lcd.print(" Gesture ");
  lcd.setCursor(0, 1);
  lcd.print(" Recognition ");
  delay(4000);

  uint8_t error = 0;

  error = paj7620Init(); // initialize Paj7620 registers
  if (error)
  {
    Serial.print("INIT ERROR, CODE:");
    Serial.println(error);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("INIT ERROR, CODE:");
    lcd.setCursor(0, 1);
    lcd.print(error);
    delay(3000);
  }
  else
  {
    Serial.println("INIT OK");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("INIT OK");
    delay(3000);
  }
  Serial.println("Please input your gestures:\n");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Input Your");
  lcd.setCursor(0, 1);
  lcd.print("Gestures");
  delay(3000);
}

void loop()
{
  uint8_t data = 0, data1 = 0, error;

  error = paj7620ReadReg(0x43, 1, &data); // Read Bank_0_Reg_0x43/0x44 for gesture result.
  if (!error)
  {
    switch (data)
    {
      // When different gestures be detected, the variable 'data' will be set to different values by paj7620ReadReg(0x43, 1, &data).
      case GES_RIGHT_FLAG:
        delay(GES_ENTRY_TIME);
        paj7620ReadReg(0x43, 1, &data);
        if (data == GES_FORWARD_FLAG)
        {
          Serial.println("Forward");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Forward");
          delay(4000);
        }
        else if (data == GES_BACKWARD_FLAG)
        {
          Serial.println("Backward");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Backward");
          delay(4000);
        }
        else if (data == GES_LEFT_FLAG)
        {
          Serial.println("Left");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Left");
          delay(4000);
        }
        else if (data == GES_RIGHT_FLAG)
        {
          Serial.println("Right");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Right");
          delay(4000);
        }
        else if (data == GES_STOP_FLAG)
        {
          Serial.println("Stop");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Stop");
          delay(4000);
        }
        else if (data == GES_EXIT_FLAG)
        {
          Serial.println("Exit");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Exit");
          delay(4000);
        }
        else if (data == GES_UNKNOWN_FLAG)
        {
          Serial.println("Unknown");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Unknown");
          delay(4000);
        }
        else
        {
          Serial.println("Error");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Error");
          delay(4000);
        }
        break;
    }
  }
  else
  {
    Serial.println("Error");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Error");
    delay(4000);
  }
}
```

```

    lcd.setCursor(0, 0);
    lcd.print("Forward");
    delay(GES_QUIT_TIME);
}
else if (data == GES_BACKWARD_FLAG)
{
    Serial.println("Backward");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Backward");
    delay(GES_QUIT_TIME);
}
else
{
    Serial.println("Right");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Right");
}
break;

case GES_LEFT_FLAG:
    delay(GES_ENTRY_TIME);
    paj7620ReadReg(0x43, 1, &data);
    if (data == GES_FORWARD_FLAG)
    {
        Serial.println("Forward");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Forward");
        delay(GES_QUIT_TIME);
    }
    else if (data == GES_BACKWARD_FLAG)
    {
        Serial.println("Backward");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Backward");
        delay(GES_QUIT_TIME);
    }
    else
    {
        Serial.println("Left");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Left");
    }
    break;

case GES_UP_FLAG:
    delay(GES_ENTRY_TIME);
    paj7620ReadReg(0x43, 1, &data);
    if (data == GES_FORWARD_FLAG)
    {
        Serial.println("Forward");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Forward");
        delay(GES_QUIT_TIME);
    }
    else if (data == GES_BACKWARD_FLAG)
    {
        Serial.println("Backward");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Backward");
        delay(GES_QUIT_TIME);
    }
    else
    {
        Serial.println("Up");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Up");
    }
    break;

```

```

case GES_DOWN_FLAG:
  delay(GES_ENTRY_TIME);
  paj7620ReadReg(0x43, 1, &data);
  if (data == GES_FORWARD_FLAG)
  {
    Serial.println("Forward");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Forward");
    delay(GES_QUIT_TIME);
  }
  else if (data == GES_BACKWARD_FLAG)
  {
    Serial.println("Backward");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Backward");
    delay(GES_QUIT_TIME);
  }
  else
  {
    Serial.println("Down");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Down");
  }
  break;

case GES_FORWARD_FLAG:
  Serial.println("Forward");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Forward");
  delay(GES_QUIT_TIME);
  break;

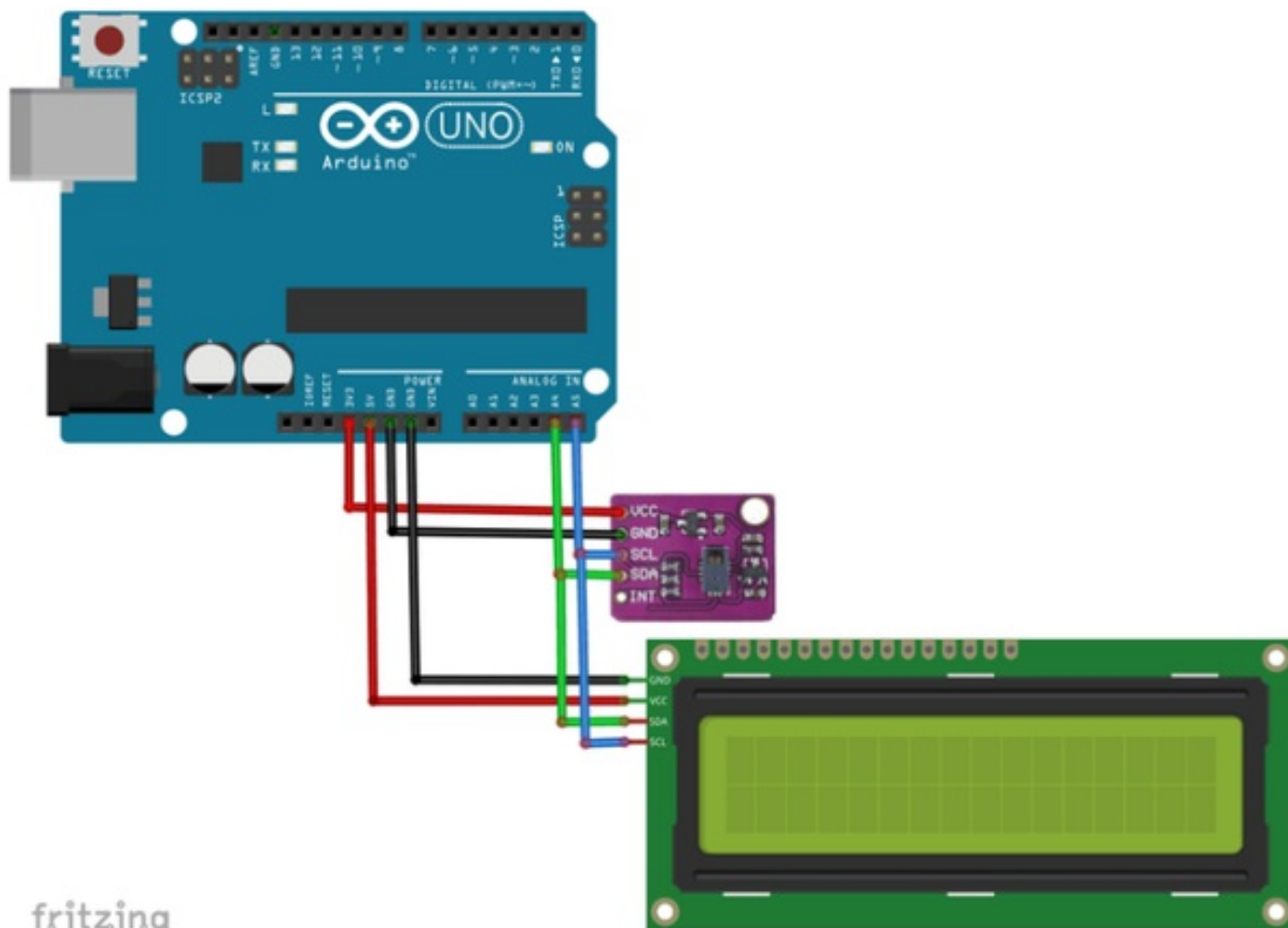
case GES_BACKWARD_FLAG:
  Serial.println("Backward");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Backward");
  delay(GES_QUIT_TIME);
  break;

case GES_CLOCKWISE_FLAG:
  Serial.println("Clockwise");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Clockwise");
  break;

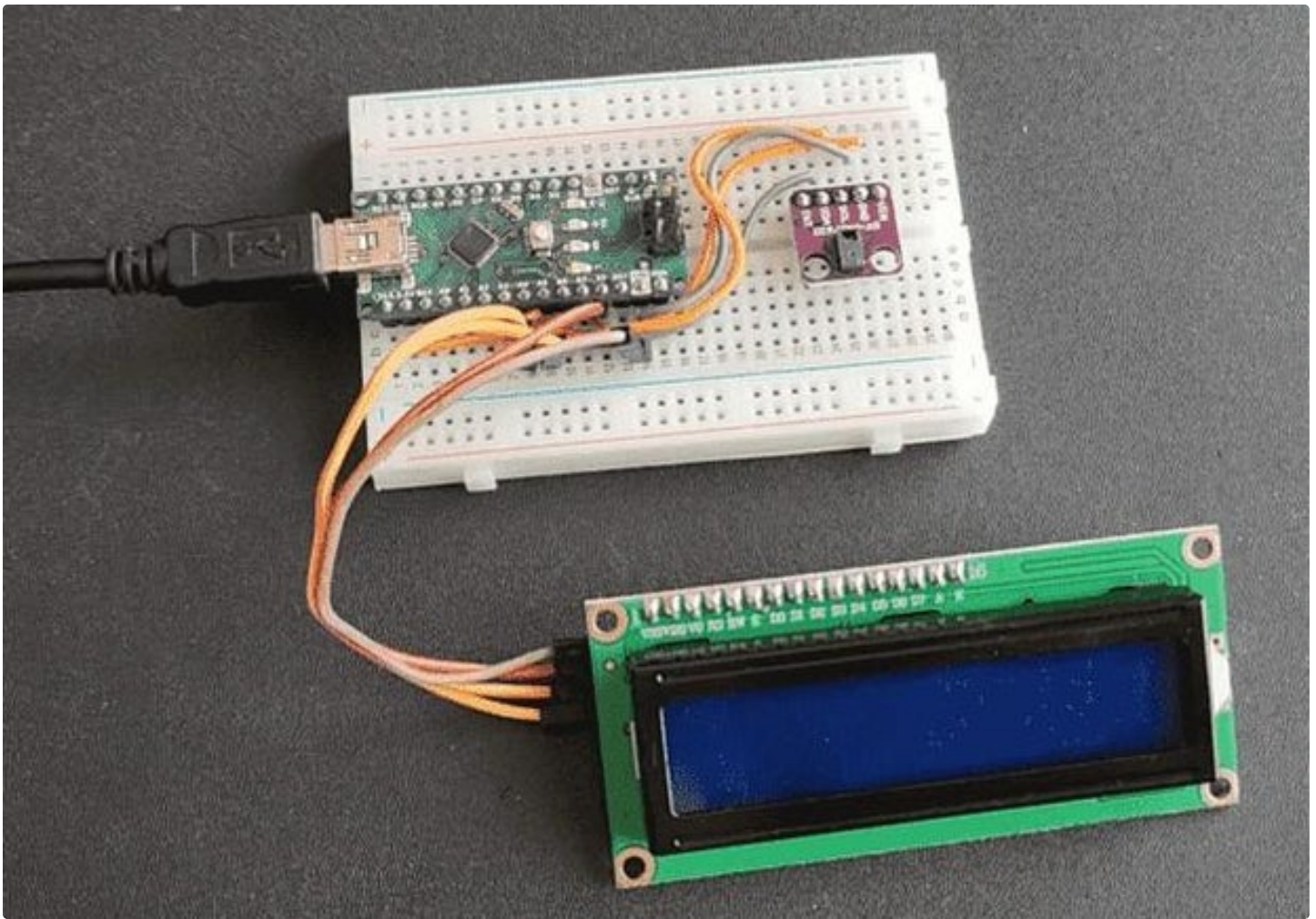
case GES_COUNT_CLOCKWISE_FLAG:
  Serial.println("anti-clockwise");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Anti-Clockwise");
  break;

default:
  paj7620ReadReg(0x44, 1, &data1);
  if (data1 == GES_WAVE_FLAG)
  {
    Serial.println("wave");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wave");
  }
  break;
}
}
delay(100);
}

```



fritzing



Step 6: Testing Gesture Movements

Upon successfully uploading the code, you'll see a message displayed on the LCD screen.

Subsequently, the LCD will prompt you to input a gesture.

At this point, you can move your hand in various directions to showcase the corresponding gesture movements on the LCD screen.

In a similar manner, moving your hand forwards and backwards will demonstrate the corresponding forward-backward gesture on the screen.

Likewise, moving your hand left and right will result in a corresponding left-right gesture being displayed on the screen.

Lastly, by moving your hand up and down, you'll be able to display the corresponding up-down gesture on the screen.

You can also perform a rotational gesture by moving your hand clockwise or counterclockwise to trigger rotational movement detection.

Beyond the fundamental nine gesture movements, you can introduce additional gestures such as right-left, left-right, up-down, down-up, forward-backward, and backward-forward to further expand your gesture detection capabilities.

