PARALYZED PEOPLE NEEDS DETECTOR

(MOTION SENSOR)

**TEAM MEMBERS AND MENTORS**

**TEAM MEMBERS**

* U.VISHNU-22BEC126
* K.SANAY KUMAR-22BEC063
* K.THARUN-22BEC064
* M.MANIDEEP-22BEC047
* R.KEERTHI-22BEC092

**PROJECT MENTOR**

**DR.DIP PRAKASH SAMAJDAR SIR**

**REPORT: Paralysed People Needs Sensor**

**1. Introduction**

**1.1 Background and Objective**

The objective of this project is to create a communication system between two Arduino boards using RF transmission. One Arduino is connected to a gyro sensor (GY-61 ADXL335) and an RF transmitter (R433S). The second Arduino, equipped with an RF receiver, an LCD display, a buzzer, and a push button, will receive messages based on the tilt direction of the gyro sensor. The system will transmit different messages based on the tilt direction, and the receiving Arduino will display the message and sound a buzzer. The buzzer will only stop when the push button is pressed.

**1.2 Scope**

The report covers:

* **System Overview**: Description of components and their functions.
* **Wiring Diagram**: How to connect the components.
* **Libraries**: Required libraries for Arduino code.
* **Code for Transmitting Arduino**: Detailed code for sending messages.
* **Code for Receiving Arduino**: Detailed code for receiving messages, controlling the buzzer, and displaying on the LCD.
* **Testing and Troubleshooting**: Steps to verify and troubleshoot the system.

**2. System Overview**

**2.1 Components**

1. **Arduino Uno (2 units)**
2. **RF Transmitter (R433S)**
3. **RF Receiver (R433S)**
4. **Gyro Sensor (GY-61 ADXL335)**
5. **LCD Display (16x2)**
6. **Buzzer**
7. **Push Button**
8. **Resistors (for button debouncing)**
9. **Breadboard and Jumper Wires**

**2.2 Functionality**

* **Transmitting Arduino**: Reads the tilt direction from the gyro sensor and transmits a corresponding message using the RF transmitter.
* **Receiving Arduino**: Receives the message via RF receiver, displays it on the LCD, and activates the buzzer. The buzzer stops only when the push button is pressed.

**3. Wiring Diagram**

**3.1 Transmitting Arduino Circuit**

* **GY-61 ADXL335 to Arduino:**
  + VCC to 3.3V (or 5V if sensor supports it)
  + GND to GND
  + X to A0
  + Y to A1
  + Z to A2
* **RF Transmitter (R433S) to Arduino:**
  + VCC to 5V
  + GND to GND
  + DATA to Digital Pin 12

**3.2 Receiving Arduino Circuit**

* **RF Receiver (R433S) to Arduino:**
  + VCC to 5V
  + GND to GND
  + DATA to Digital Pin 11
* **LCD Display (16x2) to Arduino:**
  + VCC to 5V
  + GND to GND
  + RS to Digital Pin 7
  + E to Digital Pin 8
  + D4 to Digital Pin 9
  + D5 to Digital Pin 10
  + D6 to Digital Pin 11
  + D7 to Digital Pin 12
  + (Contrast Pin to 5V through a potentiometer if required)
* **Buzzer to Arduino:**
  + Positive to Digital Pin 6
  + Negative to GND
* **Push Button to Arduino:**
  + One terminal to Digital Pin 5
  + Other terminal to GND
  + (Use a pull-up resistor or enable the internal pull-up resistor in code)

**3.3 Wiring Diagram**

Below is a simplified description of how to connect the components. For detailed wiring, refer to the schematic diagram.

**4. Libraries**

**4.1 Required Libraries**

* **RF24**: For RF communication.
* **Wire**: For I2C communication with the LCD.
* **LiquidCrystal**: For controlling the LCD display.

To install these libraries, go to the Arduino IDE and use the Library Manager (Sketch -> Include Library -> Manage Libraries) or download them directly from the Arduino website.

**5. Transmitting Arduino Code**

#include <RH\_ASK.h> // Include the RadioHead ASK library

#include <SPI.h> // Include SPI library (needed for RadioHead)

// Define the analog pins connected to ADXL335 axes

const int xPin = A0;

const int yPin = A1;

const int zPin = A2;

// Define the reference voltage for the analog-to-digital conversion

const float referenceVoltage = 5.0; // Change to 3.3 if using 3.3V power supply

// Define the sensitivity of the ADXL335

const float sensitivity = 0.2; // mV per mg (milli-g)

// Define the RF transmitter pin

const int txPin = 12;

// Create an instance of the RH\_ASK driver

RH\_ASK rf\_driver(2000, txPin, -1, -1); // 2000 baud rate, txPin, no rxPin, no pttPin

void setup() {

Serial.begin(9600); // Start serial communication at 9600 baud

if (!rf\_driver.init()) {

Serial.println("RF init failed");

} else {

Serial.println("RF init successful");

}

}

void loop() {

// Read the analog values from each axis

int xRaw = analogRead(xPin);

int yRaw = analogRead(yPin);

int zRaw = analogRead(zPin);

// Convert the raw values to voltage

float xVolts = (xRaw / 1023.0) \* referenceVoltage;

float yVolts = (yRaw / 1023.0) \* referenceVoltage;

float zVolts = (zRaw / 1023.0) \* referenceVoltage;

// Convert the voltage to acceleration (in g's)

float xAccel = (xVolts - (referenceVoltage / 2.0)) / sensitivity;

float yAccel = (yVolts - (referenceVoltage / 2.0)) / sensitivity;

float zAccel = (zVolts - (referenceVoltage / 2.0)) / sensitivity;

// Debugging: Print the raw and calculated values to the Serial Monitor

Serial.print("xRaw: "); Serial.print(xRaw);

Serial.print(" yRaw: "); Serial.print(yRaw);

Serial.print(" zRaw: "); Serial.print(zRaw);

Serial.print(" | xVolts: "); Serial.print(xVolts, 2);

Serial.print(" yVolts: "); Serial.print(yVolts, 2);

Serial.print(" zVolts: "); Serial.print(zVolts, 2);

Serial.print(" | xAccel: "); Serial.print(xAccel, 2);

Serial.print(" yAccel: "); Serial.print(yAccel, 2);

Serial.print(" zAccel: "); Serial.println(zAccel, 2);

// Check conditions and send the corresponding message

const char\* message = nullptr;

if (yAccel < -5.3) {

message = "Water";

} else if (yAccel > -3.5) {

message = "Washroom";

} else if (xAccel < -5.4) {

message = "Emergency";

} else if (xAccel > -3.5) {

message = "Hungry";

}

if (message != nullptr) {

rf\_driver.send((uint8\_t \*)message, strlen(message));

rf\_driver.waitPacketSent(); // Wait for the message to be sent

// Print the message to the Serial Monitor for debugging

Serial.print("Sent: ");

Serial.println(message);

}

// Delay for a bit before the next reading

delay(1000);

}

**6. Receiving Arduino Code**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <RH\_ASK.h>

#include <SPI.h>

// Define pins

const int buzzerPin = 8;

const int buttonPin = 7;

const int rxPin = 11;

// Initialize the I2C LCD display (address 0x27, 16 columns, 2 rows)

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Create an instance of the RH\_ASK driver

RH\_ASK rf\_driver(2000, -1, rxPin, -1); // 2000 baud rate, no txPin, rxPin, no pttPin

void setup() {

// Initialize serial communication

Serial.begin(9600);

// Initialize the RF driver

if (!rf\_driver.init()) {

Serial.println("RF init failed");

} else {

Serial.println("RF init successful");

}

// Initialize the LCD

lcd.begin(16,2);

lcd.backlight();

// Initialize the buzzer pin as an output

pinMode(buzzerPin, OUTPUT);

digitalWrite(buzzerPin, LOW);

// Initialize the button pin as an input with internal pull-up resistor

pinMode(buttonPin, INPUT\_PULLUP);

}

void loop() {

uint8\_t buf[50];

uint8\_t buflen = sizeof(buf);

// Check if a message has been received

if (rf\_driver.recv(buf, &buflen)) {

buf[buflen] = 0; // Null-terminate the string

char\* message = (char\*)buf;

// Print the received message to the serial monitor

Serial.print("Received: ");

Serial.println(message);

// Display the message on the LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Message:");

lcd.setCursor(0, 1);

lcd.print(message);

// Buzz the buzzer

digitalWrite(buzzerPin, HIGH);

// Wait until the button is pressed to stop the buzzer

while (digitalRead(buttonPin) == HIGH) {

// Do nothing, just wait for the button press

}

// Stop the buzzer

digitalWrite(buzzerPin, LOW);

}

}

**7. Testing and Troubleshooting**

**7.1 Testing Procedure**

1. **Transmitting Arduino Setup**:
   * Connect the GY-61 ADXL335 sensor and RF transmitter as described.
   * Upload the transmitting code to the Arduino.
   * Monitor the Serial Monitor for debugging messages.
2. **Receiving Arduino Setup**:
   * Connect the RF receiver, LCD display, buzzer, and push button as described.
   * Upload the receiving code to the Arduino.
   * Ensure the LCD displays the received message and the buzzer sounds.
3. **Test Tilt Directions**:
   * Tilt the gyro sensor in different directions.
   * Verify that the receiving Arduino displays the correct message and activates the buzzer.

**7.2 Troubleshooting**

* **RF Communication Issues**: Ensure that the antennas are properly connected and that the receiver and transmitter are on the same frequency.
* **LCD Display Issues**: Verify wiring connections and adjust the contrast potentiometer if necessary.
* **Buzzer Not Stopping**: Check the push button wiring and debounce logic.

**8. Conclusion**

This project successfully demonstrates a wireless communication system between two Arduino boards using RF transmission and a gyro sensor. The transmitting Arduino reads the tilt direction of the sensor and sends corresponding messages, while the receiving Arduino displays these messages on an LCD, sounds a buzzer, and stops the buzzer when a push button is pressed.

Future improvements could include enhancing the sensitivity and accuracy of the tilt detection, adding more complex messages, or integrating additional sensors for richer functionality.