**SMART CANE FOR VISUALLY IMPAIRED**

#include <Stepper.h>

#include <SoftwareSerial.h>

// Stepper motor settings

#define STEPS\_PER\_REV 2048

#define MOTOR\_SPEED 10

// Define ultrasonic sensor pins

#define TRIG\_PIN 7

#define ECHO\_PIN 6

// Define buzzer pin

#define BUZZER\_PIN 12

// Define stepper motor control pins (ULN2003 driver)

//Stepper motor1(STEPS\_PER\_REV, 0, 1, 8, 11);

Stepper motor2(STEPS\_PER\_REV, 5, 4, 3, 2);

// GSM Module

SoftwareSerial mySerial(9, 10); // TX, RX for GSM module

const int sosSwitch = 13; // Define the pin for the SOS switch

bool systemStarted = false; // Flag to check if startup message is sent

void setup() {

Serial.begin(9600);

mySerial.begin(9600);

// Set up ultrasonic sensor

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

// Set up buzzer

pinMode(BUZZER\_PIN, OUTPUT);

// Set up SOS switch

pinMode(sosSwitch, INPUT\_PULLUP);

// Set stepper motor speed

//motor1.setSpeed(MOTOR\_SPEED);

motor2.setSpeed(MOTOR\_SPEED);

Serial.println("System Initialized");

// Send startup message

SendMessage();

systemStarted = true; // Set flag to true after sending message

}

void loop() {

long duration;

int distance;

// Trigger ultrasonic sensor

digitalWrite(TRIG\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

// Read echo pulse

duration = pulseIn(ECHO\_PIN, HIGH);

distance = duration \* 0.034 / 2; // Convert to cm

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance > 0 && distance < 30) {

digitalWrite(BUZZER\_PIN, HIGH); // Turn on buzzer

Serial.println("Obstacle detected! Stopping motors...");

// Stop motors

//motor1.step(0);

motor2.step(0);

} else {

digitalWrite(BUZZER\_PIN, LOW); // Turn off buzzer

Serial.println("Path clear. Resuming movement...");

// Move motors continuously forward

//motor1.step(10);

motor2.step(10);

}

// If SOS switch is pressed, make a call

if (digitalRead(sosSwitch) == LOW) {

Serial.println("Manual SOS Triggered!");

MakeCall();

delay(10000); // Avoid multiple triggers

}

}

// Function to send SOS message when the system starts

void SendMessage() {

Serial.println("Sending Startup SOS Message...");

mySerial.println("AT+CMGF=1");

delay(1000);

mySerial.println("AT+CMGS=\"+916379361758\"\r"); // Replace with your number

delay(1000);

mySerial.println("System Activated. Location: https://www.google.com/maps?q=13.15025,80.19192"); // Replace with actual GPS coordinates

delay(100);

mySerial.println((char)26); // CTRL+Z to send message

delay(1000);

Serial.println("Startup SOS Message Sent!");

}

// Function to make an SOS call when the button is pressed

void MakeCall() {

Serial.println("Calling SOS number...");

mySerial.println("ATD+916379361758;"); // Replace with your number

delay(100); // Wait for 30 seconds before hanging up

HangupCall();

}

// Function to hang up a call

void HangupCall() {

mySerial.println("ATH"); // End the call

Serial.println("Call Ended");

}