Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Smart Blind Stick

CSE 3216

Microcontroller Based System Design Lab

Submitted By:

Tonmoy Mohajan 16.01.04.035

Date of Submission: **18 April, 2019**

Introduction:

Visually impaired people find difficulties detecting obstacles in front of them, during walking in the street, which makes it dangerous. The smart stick comes as a proposed solution to enable them to identify the world around. In this project we propose a solution, represented in a smart stick with pair of ultrasonic sensors to detect any other obstacles in front of the user, within a fix range of distance.

Equipment:

Hardware:

- Light Dependent Resistor (LDR)
- Sonar Sensor (HC-SR04)
- 1.5”x 5” proto board
- Soil Moisture Sensor
- Buzzer
- 74HC595 Shift Register
- LED
- Breadboard

Software:

- **Arduino IDE:** Arduino IDE is used for coding, debugging and uploading code to the board.

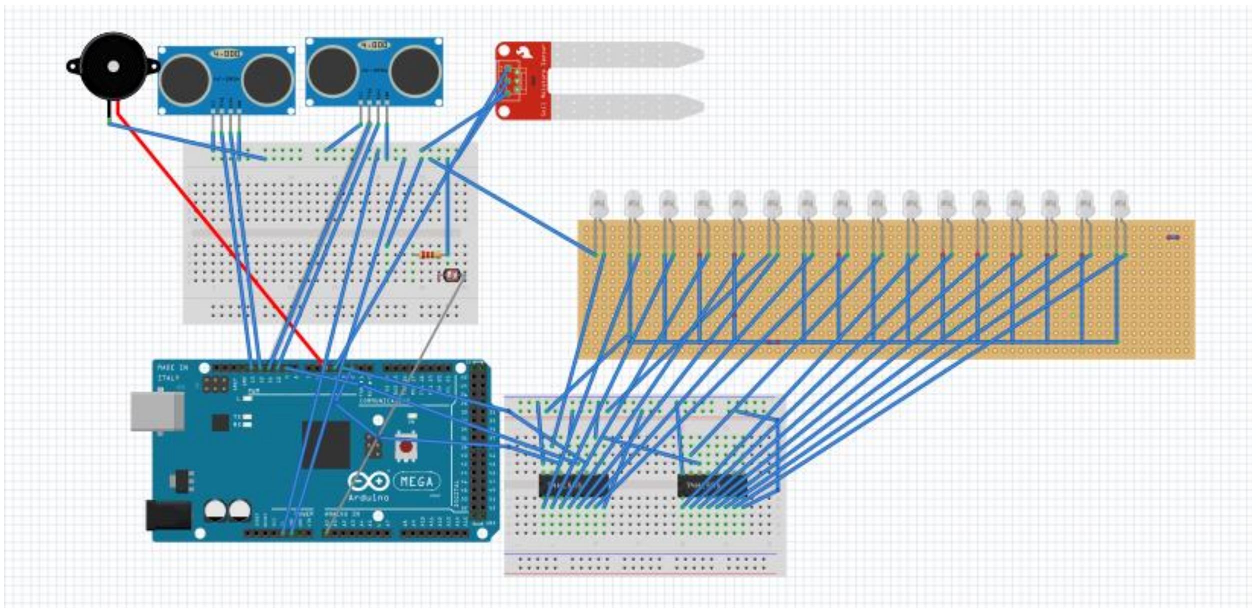
Features:

- ✓ When the obstacle stands between 70 cm the buzzer will make a noise.
- ✓ When the stick goes closer (about 40cm) the vibrator motor will produce a vibration.
- ✓ When the stick gets in touch of water it will produce a long beep by the buzzer.
- ✓ When the stick goes through a dark place the 16 led will lighten up.

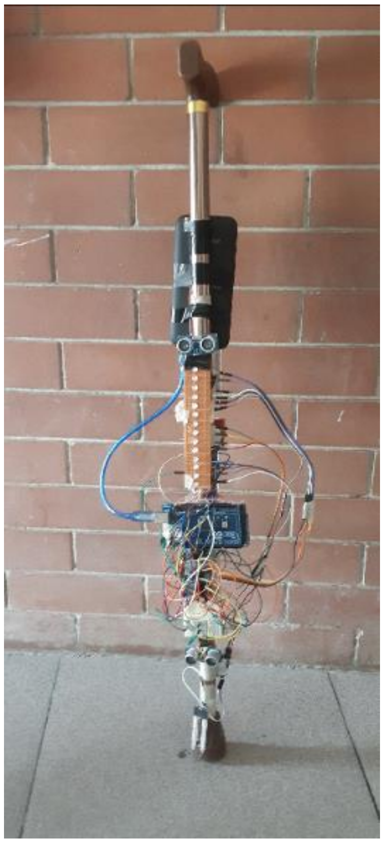
Working Principle:

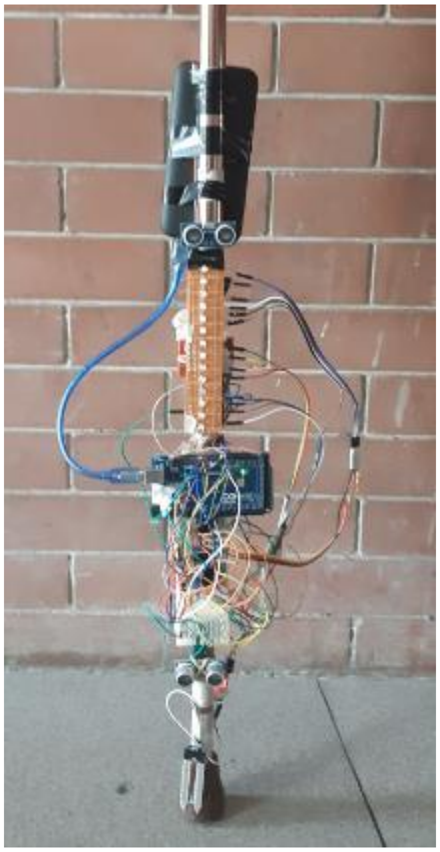
- ❖ This stick will work depend on ultrasonic sensor which will help to find any object in a specified range.
- ❖ If it finds any object between 70 cm it will make a beep by the buzzer.
- ❖ If the object will come closer (40 cm) then the vibrator motor will vibrate.
- ❖ If the stick gets in touch of water, then it will make a long beep by the buzzer.
- ❖ If the stick goes through a dark place, then the 16 led will lighten up.

Circuit Diagram:



Figures of The Project:





Constrains:

- The stick should move slowly
- The stick should hold vertically to get perfect result
- If the object is in between the two ultrasonic sensors, then it may fall into the blind spot in where it may not detect the object.

Dos and Don’ts:

1. The object can be detected in between 70 cm.
2. The water can be detected.
3. It can produce different types of tune using a single buzzer.
4. In the dark place the led will lighten up.
5. The stick may not work properly if the stick doesn’t hold vertically.
6. The stick should move slowly to detect the obstacle.

Conclusion:

The Smart Stick acts as a basic platform for the coming generation of more aiding devices to help the visually impaired to be safer. It is effective and afford. It leads to good results in detecting the obstacles lying ahead of the user in a range of fix distance and water pits.

Appendix:

Program Code:

```
const int trigPin = 10;
const int echoPin = 11;
const int ldrPin = A0;
const int trigPin2 = 12;
const int echoPin2 = 13;
int motor = 7;
int digitalInput = 4;
int speaker = 5;
float duration, distance;
float duration2, distance2;
int latchPin = 9;  //Storage register clock pin
int clockPin = 3;  //Shift register clock pin
int dataPin = 2;
int numOfRegisters = 2;
byte* registerState;

long effectId = 0;
long prevEffect = 0;
long effectRepeat = 0;
long effectSpeed = 30;


void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(motor, OUTPUT);
  pinMode(speaker, OUTPUT);
  pinMode(digitalnput, INPUT);

  //Initialize for led
  registerState = new byte[numOfRegisters];
  for (size_t i = 0; i < numOfRegisters; i++) {
    registerState[i] = 0;
  }

  //set pins to output so you can control the shift register
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);

  //Init LDR
  pinMode(ldrPin, INPUT);
  Serial.begin(9600);

}
```

```
void loop() {

  checkDarkness();
  firstSensor();
  secondSensor();
  soilSensor();
}


void checkDarkness() {
  int ldrStatus = analogRead(ldrPin);
  if (ldrStatus <= 300) {
    Serial.print("Its DARK value is: ");
    Serial.println(ldrStatus);
      ledOn();

  } else {
      effectF(10);
    Serial.print("Its BRIGHT, Turn off the LED : ");
    Serial.println(ldrStatus);
  }
}

void ledOn() {
  effectId = random(6);
  effectSpeed = 40;
  effectB(effectSpeed);
  effectC(effectSpeed);
  effectD(effectSpeed);
  effectE(effectSpeed);
  effectF(effectSpeed);

}
```

```
void soilSensor() {
  if (digitalRead(digitalnput) == HIGH)
  {
    Serial.println("Dry");
  }
  if (digitalRead(digitalnput) == LOW)
  {
    //digitalWrite(speaker, HIGH);
    tone(speaker, 1000);
    delay(500);
    //digitalWrite(speaker, LOW);
    noTone(speaker);

    Serial.println("Wet");
  }
}


void firstSensor() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;
  Serial.println("A is");
  Serial.println(distance);
  if (distance > 40 && distance < 70) {
    tone(speaker, 1000);
    delay(10);
    noTone(speaker);
    Serial.println("buzzer on");
  }
  if (distance < 40)// This is where checking the distanceyou can change the value
  {
    digitalWrite(motor, HIGH);

  } else
  {
    digitalWrite(motor, LOW);
  }
  delay(500);
}



void secondSensor() {
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  duration2 = pulseIn(echoPin2, HIGH);
  distance2 = (duration2 / 2) / 29.1;
  Serial.println("B is");
  Serial.println(distance2);
  if (distance2 > 40 && distance2 < 70) {
    tone(speaker, 1000);
    delay(10);
    noTone(speaker);
    Serial.println("buzzer on");
  }
  if (distance2 < 40)// This is where checking the distanceyou can change the value
  {
    digitalWrite(motor, HIGH);
  }
  else
  {
    digitalWrite(motor, LOW);
  }
  delay(500);

}
void effectB(int speed) {
  for (int i = 15; i >= 0; i--) {
    for (int k = 0; k < i; k++) {
      regWrite(k, HIGH);
      delay(speed);
      regWrite(k, LOW);
    }

    regWrite(i, HIGH);
  }
}

void effectC(int speed) {
  int prevI = 0;
  for (int i = 0; i < 16; i++) {
    regWrite(prevI, LOW);
    regWrite(i, HIGH);
    prevI = i;
   delay(speed);
  }
  for (int i = 15; i >= 0; i--) {
    regWrite(prevI, LOW);
    regWrite(i, HIGH);
    prevI = i;
    delay(speed);
  }
}
void effectD(int speed) {
  for (int i = 0; i < 8; i++) {
    for (int k = i; k < 8; k++)
    {
      regWrite(k, HIGH);
      regWrite(15 - k, HIGH);
      delay(speed);
      regWrite(k, LOW);
      regWrite(15 - k, LOW);
    }

    regWrite(i, HIGH);
    regWrite(15 - i, HIGH);
  }
}

void effectE(int speed) {
  for (int i = 7; i >= 0; i--) {
    for (int k = 0; k <= i; k++)
    {
      regWrite(k, HIGH);
      regWrite(15 - k, HIGH);
      delay(speed);
      regWrite(k, LOW);
      regWrite(15 - k, LOW);
    }
    regWrite(i, HIGH);
    regWrite(15 - i, HIGH);
  }
}
void effectF(int speed) {
  for (int i = 7; i >= 0; i--) {
    for (int k = 0; k <= i; k++)
    {
      regWrite(k, LOW);
      regWrite(15 - k, LOW);
      delay(speed);
      regWrite(k, LOW);
      regWrite(15 - k, LOW);
    }
    regWrite(i, LOW);
    regWrite(15 - i, LOW);
  }
```

```
}


void regWrite(int pin, bool state) {
  //Determines register
  int reg = pin / 8;
  //Determines pin for actual register
  int actualPin = pin - (8 * reg);
  //Begin session
  digitalWrite(latchPin, LOW);
  for (int i = 0; i < numOfRegisters; i++) {
    //Get actual states for register
    byte* states = &registerState[i];
    //Update state
    if (i == reg) {
      bitWrite(*states, actualPin, state);
    }
    //Write
    shiftOut(dataPin, clockPin, MSBFIRST, *states);
  }
  //End session
  digitalWrite(latchPin, HIGH);
}
```