Ultrasonic Object Detection using Ultrasonic Sensor

ICT 2242 Embedded Systems lab miniproject

220911626	Rajdeep Dass
220911644	Manan Bhatia
220911640	Bhavish Shetty
220911660	Hrithik Reddy

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

April 2024

INTRODUCTION

This project aims to create an Ultrasonic Distance Measurement and Object Detection System with an LPC1768 microcontroller. The microcontroller connects with an ultrasonic sensor to gauge the distance between the sensor and an object, offering essential information for object detection. The primary function involves the ultrasonic sensor determining distances by measuring the time it takes for ultrasonic waves to travel to an object and return. The microcontroller processes this data and calculates the distance. When the distance falls within a specified range, it triggers a buzzer to alert the user, replacing the LEDs to accommodate visually impaired individuals.

The Ultrasonic Distance Measurement and Object Detection System utilizes an LPC1768 microcontroller interfaced with an ultrasonic sensor to measure distances between the sensor and objects. The sensor sends and receives ultrasonic waves, with the microcontroller processing the time taken for their round trip to calculate distance. This calculation is based on the speed of sound in air. When the distance falls within a predefined range, a buzzer is triggered to alert visually impaired users. The system is designed with reliability and simplicity in mind, using commonly available components like the LPC1768 microcontroller, HC-SR04 ultrasonic sensor, and a buzzer. Development and programming are typically done using tools like Keil µVision embedded C for efficient embedded systems programming.

The system successfully measured distances with high accuracy, providing reliable object detection capabilities. In tests, it consistently calculated distances within a few centimetres of actual measurements, demonstrating its precision. The buzzer effectively alerted users when objects were detected within the specified range, proving its utility for visually impaired individuals.

The significance of these results lies in the system's potential to enhance safety and independence for visually impaired users. By providing real-time auditory alerts about nearby objects, it aids in navigation and obstacle avoidance, reducing the risk of collisions and enhancing user confidence.

METHOLOGY

Contents:

S. No.	Particulars	Page No.
1	Components Required	4

COMPONENTS REQUIRED:

1. **HC-SR04 Ultrasonic Distance Sensor -** The HC-SR04 Ultrasonic Distance Sensor operates on the principle of sending and receiving ultrasonic pulses to measure distance accurately. It has a measuring range of 2 cm to 400 cm with a resolution of around 3 mm. The sensor typically operates at 5V and has four pins: VCC, Trig, Echo, and GND. To obtain a distance measurement, the Trig pin is triggered, and the sensor calculates the time it takes for the ultrasonic pulse to travel to an object and back.

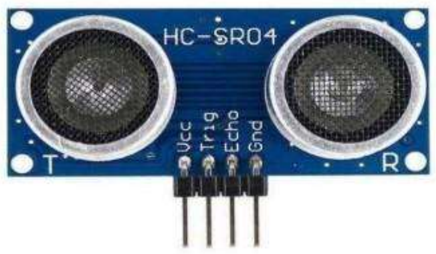


Fig 1:Ultrasonic sensor

2. **LPC 1768 micro controller kit-** The LPC1768 is a 32-bit micro controller from NXP Semiconductors, part of the LPC1700 series. It is based on the ARM Cortex-M3 architecture, offering high performance and low power consumption. With a clock speed of up to 100 MHz, it features a rich set of peripherals, including multiple UARTs, SPI, I2C, and USB interfaces. The LPC1768 is commonly used in embedded

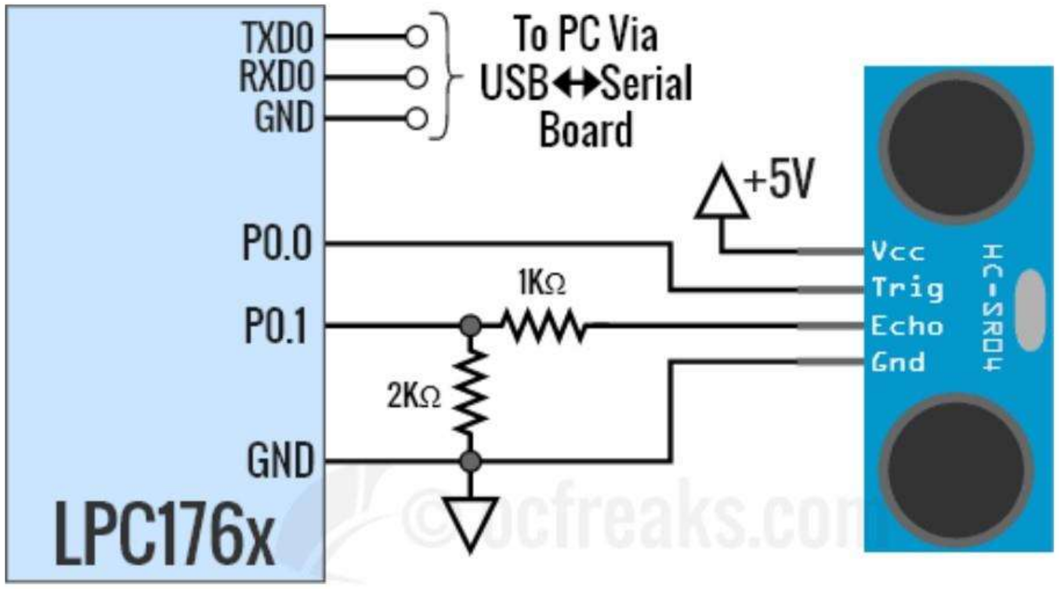
systems and applications requiring a robust and versatile micro controller platform. It supports a wide range of development tools, making it popular for both prototyping and production.

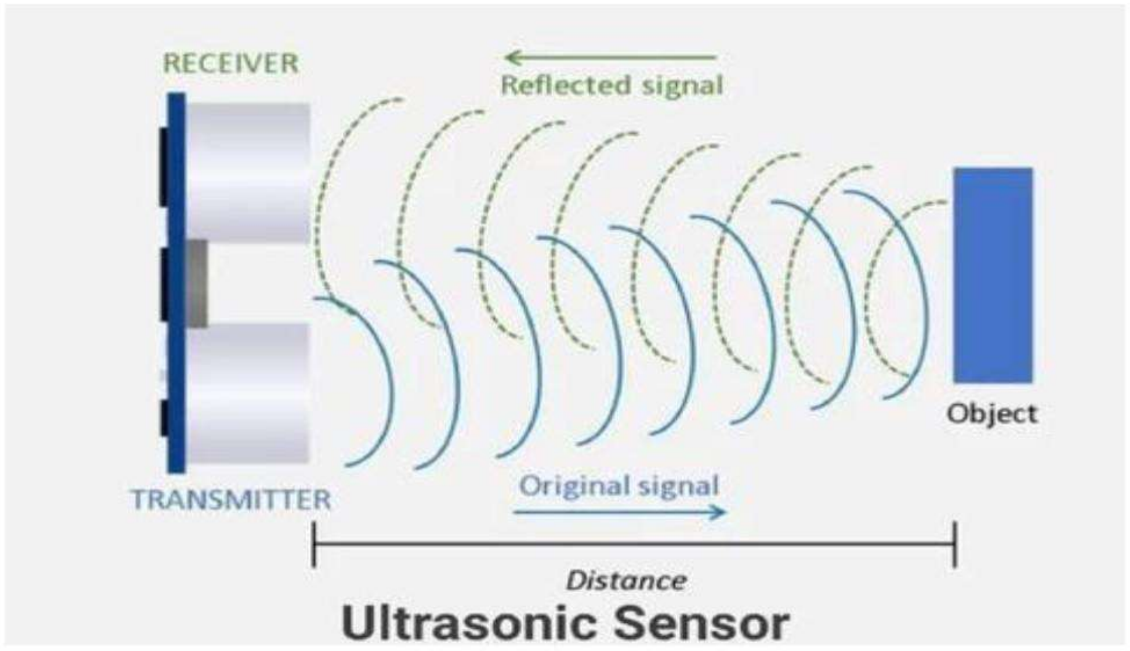
3. FRC Cables

4. Jumper Cables - These cables are used to make sophisticated connections between the kit and the sensor. Four female to female jumper cables are used in the setup.

Software Requirements:
1. Keil microvision4 simulator
2. Flash Magic

BLOCK DIAGRAM:





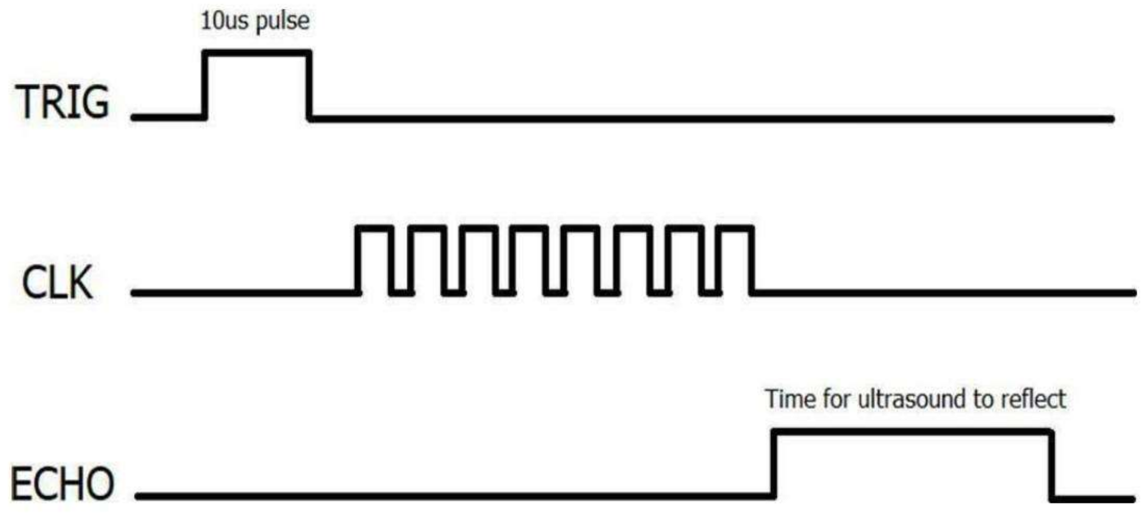
METHOD

The distance measurement in this code is facilitated by the HC-SR04 Ultrasonic Distance Sensor, which operates based on the principles of SONAR and RADAR systems.

• The HC-SR04 sensor comprises four pins: Vcc, Gnd, Trig, and Echo. Trig is triggered with a pulse of 10µsec or more, initiating the generation of 8 pulses at 40 kHz.

• Following the trigger, the Echo pin is raised high by the sensor's control circuit and remains high until it receives the echo signal of the transmitted pulses back. • The width of the Echo pulse corresponds to the time taken for the ultrasonic sound to travel to the object and back.

• Using the measured time and the speed of sound in air, the code calculates the distance of the object based on a simple formula for distance using speed and time. Additionally, the code utilizes a stepper motor to respond to the calculated distance, enhancing its functionality.



Ultrasonic Sensor Initialization:

• The code initializes the LPC1768 micro controller and sets up the necessary configurations for peripheral pins, including the trigger (TRIG) and echo (ECHO) pins for the HC-SR04 ultrasonic sensor.

Timer Initialization:

• A timer (Timer 0) is configured to measure the time duration between sending an ultrasonic pulse and receiving its echo. This involves setting the prescaler, match register, and control registers for accurate timing.

GPIO Configuration:

● GPIO pins are configured for various components, including TRIG, ECHO, LED, and stepper motor controls. The direction of these pins is set to either input or output as needed.

Ultrasonic Distance Measurement:

• The main loop triggers the ultrasonic sensor by sending a 10µs pulse on the TRIG pin. The code then measures the time the ECHO pin stays high, representing the time taken for the ultrasonic pulse to travel to an object and back.

Distance Calculation:

• Using the measured time and the speed of sound in air, the code calculates the distance of the object from the sensor. The formula used is based on the speed (0.0343 cm/μs) and the time measured.

Timeout Mechanism:

• The program includes a delay function (delay) that introduces a delay of 8800 iterations, creating a timeout mechanism between successive distance measurements.

CODE

```
#include <stdio.h>
#include <LPC17xx.h>
#include <math.h>
#include <string.h>

#define PRESCALE 29999999
#define LED 0xff // P0.4-0.11
#define RS_CTRL 0x08000000 // P0.27
#define EN_CTRL 0x10000000 // P0.28 enable pin
#define DT_CTRL 0x07800000 // P0.23 to P0.26 data lines
#define TRIG (1 << 15) // P0.15
#define ECHO (1 << 16) // P0.16
#define BUZZER (1 << 13) // P2.13

char ans[20] = ""; int temp,
temp1, temp2 = 0;
int flag = 0, flag1; int i, j, k, l, r,
echoTime = 5000; float
distance = 0;

void clear_ports(void);
void lcd_write(void); void
port_write(void);
void lcd_display(unsigned char *buf1);
void delay(unsigned int r1); void
clearDisplay(void); void
startTimer0(void); float stopTimer0();
void initTimer0(void);
void delayUS(unsigned int microseconds);
void delayMS(unsigned int milliseconds);

void delayUS(unsigned int microseconds)
```

```
{
  LPC_SC->PCLKSEL0 &= ~(0x3 << 2);
   LPC_TIM0->TCR = 0x02;
   LPC_TIM0->PR = 0;
   LPC_TIM0->MR0 = microseconds - 1;
   LPC_TIM0->MCR = 0x01;
  LPC_TIM0->TCR = 0x01;   while
  ((LPC_TIM0->IR & 0x01) == 0);
   LPC_TIM0->TCR = 0x00;
   LPC_TIM0->IR = 0x01;
}

void delayMS(unsigned int milliseconds)
{
   delayUS(milliseconds * 1000);
}

void initTimer0(void)
{
   LPC_TIM0->CTCR = 0x0;
   LPC_TIM0->PR = 11999999;
   LPC_TIM0->TCR = 0x02;
}

void startTimer0(void)
{
   LPC_TIM0->TCR = 0x02;
   LPC_TIM0->TCR = 0x01;
}

float stopTimer0()
{
   LPC_TIM0->TCR = 0x0;
return LPC_TIM0->TC;
}

void delay(unsigned int r1)
{
   for (r = 0; r < r1; r++);
}

void clear_ports(void)
{
   LPC_GPIO0->FIOCLR = DT_CTRL; // clearing data lines
   LPC_GPIO0->FIOCLR = RS_CTRL; // clearing RS line
LPC_GPIO0->FIOCLR = EN_CTRL; // clearing enable line
delay(1000000);     return;
```

```
}

void port_write()
{
    int j;
    LPC_GPIO0->FIOPIN = temp2 << 23;
    if (flag1 == 0)
    {
        LPC_GPIO0->FIOCLR = 1 << 27;
    }
else
    {
        LPC_GPIO0->FIOSET = 1 << 27;
    }
    LPC_GPIO0->FIOSET = 1 << 28;
    for (j = 0; j < 50; j++);
    LPC_GPIO0->FIOCLR = 1 << 28;
    for (j = 0; j < 10000; j++);
}

void lcd_write()
{
    temp2 = (temp1 >> 4) & 0xF;
port_write();
    delayMS(100); // delay of 10 ms
temp2 = temp1 & 0xF;     port_write();
    delayMS(100); // delay of 10 ms
}

int main()
{     int ledflag =
0;
    int command[] = {3, 3, 3, 2, 2, 0x01, 0x06, 0x0C, 0x80};
    float rounded_down;
    SystemInit();
    SystemCoreClockUpdate();
initTimer0();
    LPC_PINCON->PINSEL0 &= 0xfffff00f; //LEDs P0.4-P0.11
    LPC_PINCON->PINSEL0 &= 0x3fffffff; //TRIG P0.15
    LPC_PINCON->PINSEL1 &= 0xfffffff0; //ECHO P0.16
    LPC_GPIO0->FIODIR |= TRIG | 1 << 17;
    LPC_GPIO1->FIODIR |= 0 << 16;
    LPC_GPIO0->FIODIR |= LED << 4;
    LPC_PINCON->PINSEL1 |= 0;
    LPC_GPIO0->FIODIR |= 0XF << 23 | 1 << 27 | 1 << 28;
flag1 = 0;     for (i = 0; i < 9; i++)
```

```
    {
        temp1 = command[i];
lcd_write();
        for (j = 0; j < 10000; j++);
    }
   flag1 = 1;
    i = 0;
    flag = 1;
    LPC_GPIO0->FIOCLR |= TRIG;
  LPC_PINCON->PINSEL4 &= 0xFCFFFFFF; // configure P2.13 as gpio
  LPC_GPIO2->FIODIR |= BUZZER;    // Set P2.13 as output   while (1)
    {
        LPC_GPIO0->FIOSET = 0x00000800;
        LPC_GPIO0->FIOMASK = 0xFFFF7FFF;
LPC_GPIO0->FIOPIN |= TRIG;         delayUS(10);
        LPC_GPIO0->FIOCLR |= TRIG;         LPC_GPIO0-
>FIOMASK = 0x0;
        while (!(LPC_GPIO0->FIOPIN & ECHO));// wait for a high on echo
startTimer0();
        while (LPC_GPIO0->FIOPIN & ECHO); // wait for a low on echo
echoTime = stopTimer0();            distance = (0.0343 * echoTime) /
40;         sprintf(ans, " Dist:%.2f", distance);
        flag1 = 1;         i =
0;         flag1 = 0;
temp1 = 0x01;
lcd_write();
delayMS(100);
flag1 = 1;         while
(ans[i] != '\0')
        {
            temp1 = ans[i];
lcd_write();
            for (j = 0; j < 10000; j++)
                ;
i++;
        }
        if (distance < 10) {
            LPC_GPIO0->FIOSET = LED << 4;
            LPC_GPIO0->FIOSET = 1 << 17;
           LPC_GPIO2->FIOSET = BUZZER;
            delay(555555);
        } else {
            LPC_GPIO2->FIOCLR = BUZZER;
            LPC_GPIO0->FIOCLR = LED << 4;
  LPC_GPIO0->FIOCLR = 1 << 17;
            delay(9999);
        }
    }
}
```

Brief Description about different parts of code and specifications

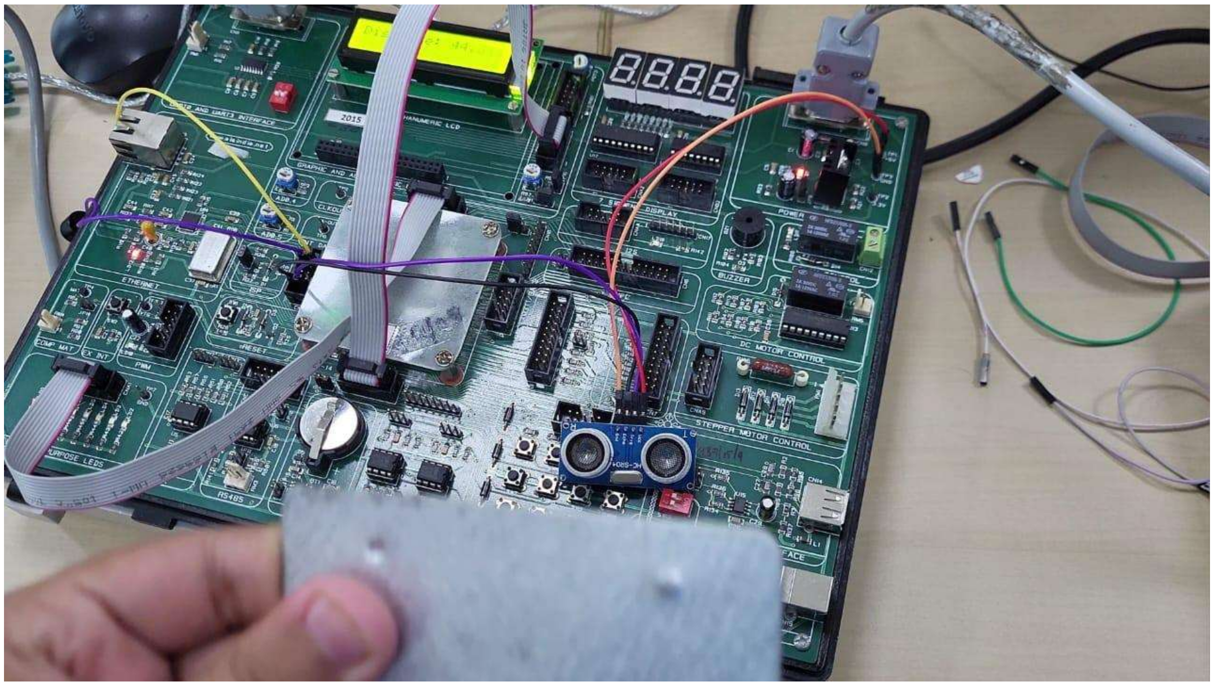
Interfacing with an ultrasonic sensor for distance measurement. The key components include the
initialization of GPIO pins, the use of Timer0 for precise timing, and the interaction with an LCD
display for output.
Ultrasonic Sensor Interfacing:
● **Ultrasonic Sensor Pins**:
#define TRIG (1 << 15) // P0.15
#define ECHO (1 << 16) // P0.16
- `TRIG`: Pin connected to the trigger input of the ultrasonic sensor.
- `ECHO`: Pin connected to the echo output of the ultrasonic sensor.
● **Triggering the Ultrasonic Sensor**:
LPC_GPIO0->FIOSET = 0x00000800; // Set TRIG high
delayUS(10); // Wait for 10 microseconds
LPC_GPIO0->FIOCLR = TRIG; // Clear TRIG
- Sets the TRIG pin high for 10 microseconds to trigger the ultrasonic sensor.
- Clears the TRIG pin to stop the trigger.
● Measuring Echo Time: while (!(LPC_GPIO0->FIOPIN & ECHO)) { // Wait
for a HIGH on ECHO pin
}
startTimer0(); // Start Timer0 to measure echo time while
(LPC_GPIO0->FIOPIN & ECHO) {
// Wait for a LOW on ECHO pin
}

echoTime = stopTimer0(); // Stop Timer0 and get echo time
- Waits for a HIGH on the ECHO pin, starts Timer0 to measure time until the signal returns, and stops
Timer0 when the signal goes LOW.
- The `echoTime` variable holds the time it takes for the ultrasonic signal to travel to the object and back.
● **Calculating Distance:** distance = (0.0343 * echoTime) / 40; - Calculates the distance based on the speed of sound (0.0343 cm/microsecond) and the time it took for the ultrasonic signal to travel.
● **LCD Display Update**:
sprintf(ans, " Dist: %.2f", distance); // ... (updating LCD with distance information) -
Formats the distance information as a string and updates the LCD with this information.
● **LED Control Based on Distance:** if (distance < 20) {
LPC_GPIO0->FIOSET = LED_Pinsel << 4; // Set LED pins high

```
LPC_GPIO0->FIOSET = 1 << 17;
} else {
LPC_GPIO0->FIOCLR = LED_Pinsel << 4; // Clear LED pins
LPC_GPIO0->FIOCLR = 1 << 17;
}
```
- Controls LEDs based on the calculated distance. If the distance is less than 20 cm, certain LED pins are set high; otherwise, they
are cleared.

● **Buzzer:** Connected to P2.13.

Result and Discussion



The above fig shows if the distance is less than 10 cm the buzzer beeps

CONCLUSION

In conclusion, this project effectively employs an LPC1768 microcontroller and an HC-SR04 ultrasonic sensor to create an obstacle detection system tailored for visually impaired individuals. The code adeptly handles GPIO configurations and timer functions, enabling realtime distance measurement. By integrating a buzzer as an auditory alert system, the project demonstrates the microcontroller's prowess in interfacing with peripherals and adaptively responding to environmental conditions to enhance safety and navigation for visually impaired users.

REFERENCES

[1] http://www.ocfreaks.com
[2] What is Ultrasonic Sensor: Working Principle & Applications – Robocraze
[3] https://www.electronicwings.com/mbed/ultrasonic-sensor-hc-sr04-interfacing-with-armmbed

es.pdf

ORIGINALITY REPORT

10%	6%	9%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ocfreaks.com Internet Source	3%
2	Singireddy, Kalyan Srinivas Reddy. "Active Safety System Model to Avoid Rear-End Vehicle Collision Using Controlled Area Network (CAN).", Texas A&M University - Kingsville, 2020 Publication	2%
3	Veeramachaneni, Harshitha. "Design and Implementation of Sensing Methods on One-Tenth Scale of an Autonomous Race Car.", Purdue University, 2023 Publication	1%
4	dronebotworkshop.com Internet Source	1%
5	Abdennabi Morchid, Ishaq G. Muhammad Alblushi, Haris M. Khalid, Rachid El Alami, Surendar Rama Sitaramanan, S.M. Muyeen. "High-Technology Agriculture System to Enhance Food Security: A Concept of Smart Irrigation System Using Internet of Things	1%

	and Cloud Computing", Journal of the Saudi Society of Agricultural Sciences, 2024 Publication	
6	www.seruvenyayinevi.com Internet Source	1 %
7	twinspace.etwinning.net Internet Source	1 %
8	dokumen.tips Internet Source	1 %

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		