Jacobs University

Browser Control Using Ultrasonic Sensors

by

Taiyr Begeyev
Youn Jae Jo
Ailin Liu

CH09-320113 Introduction to Intelligent Mobile Systems Lab I
Ph.D. Fangning Hu
30 November 2018

Table of Contents

1.0 Introduction **3**

2.0 Circuit **3**

2.1 Apparatus 3

2.2 Circuit 4

3.0 Source Codes **5**

3.1 Arduino Code 5

3.2 Python Code 8

4.0 Division of Tasks **10**

Works Cited **11**

1.0 Introduction

A Human-Machine Interface (HMI) refers to "a user interface or dashboard that connects a person to a machine, system, or device". HMI comes in various forms such as machine screens, computer monitors or tablets (Inductive Automation). This project uses another way of interaction between a user and a device—a hand gesture recognition method. Essentially, the device should function according to a user's hand gestures as it would with a mouse, touchpad or a keyboard.

The purpose of this report is to outline an Arduino project that deals with controlling a web browser using hand gestures. The project will use ultrasonic sensors to detect the motion of the hand and utilise Arduino and Python to connect the motion to the web browser of a laptop. It is expected that the project allows hand gestures to carry out certain functions on a web browser such as scrolling up and down or switching to different tabs open in a window.

2.0 Circuit

2.1 Apparatus

- 1 Arduino USB board
- 1 Arduino USB cable
- 2 Ultrasonic sensors
- 8 or more jumper cables
- 1 Laptop with Internet connection

2.2 Circuit

The following two Figures represent the circuit that was used for this project. Figure 1 is a diagram of the circuit used for this project, and Figure 2 is an image of what the circuit actually looked like. The method is implicitly described in Figure 1, and the Arduino USB board has to be connected to the laptop by the Arduino USB cable.

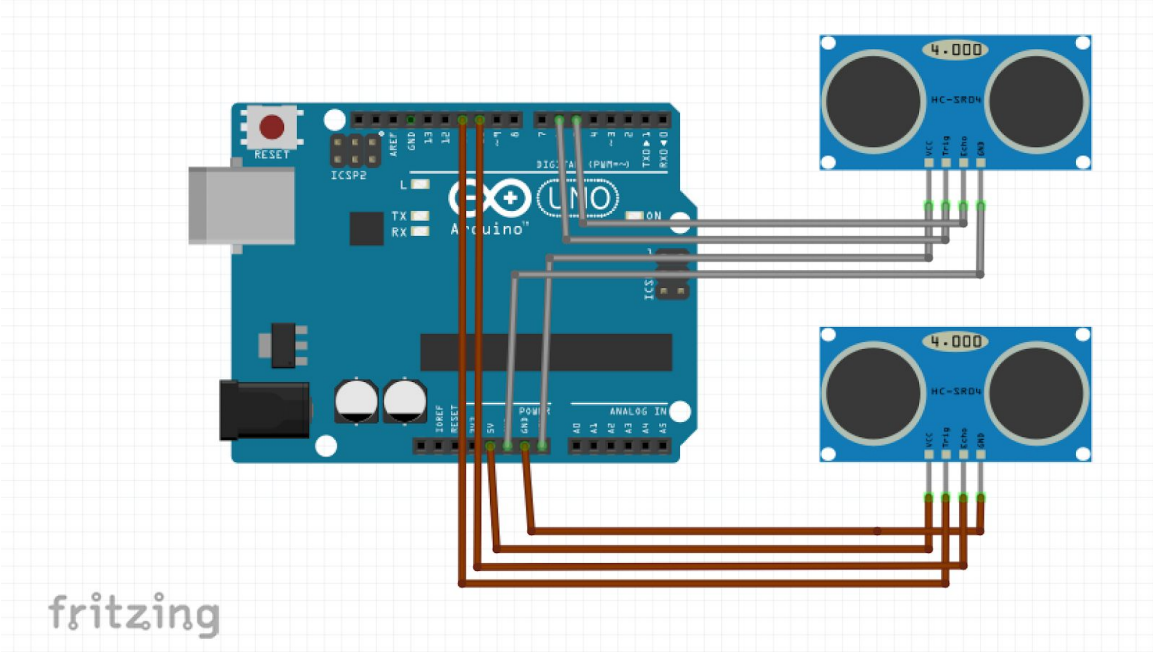

Figure 1. Circuit Diagram (Fritzing)

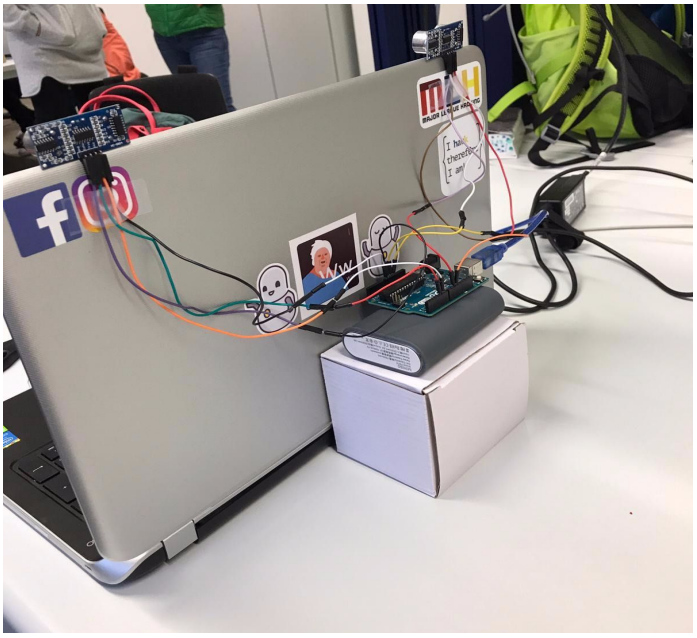

Figure 2. Actual Circuit Setup

3.0 Source Codes

This project is implemented by Arduino and Python and therefore requires a source code for each of them. We use two programming languages in our project: C and Python. We use C in order to communicate with our Arduino and Python for communicating with the operating system. The following Arduino and Python codes were taken from Electronic Hub.

3.1 Arduino Code

```
/*
 * gesture control program for controlling certain functions in windows pc
 * Code by BalaAppu
 * Website: www.electronicshub.org
 */

const int trigPin1 = 11;
const int echoPin1 = 10;
const int trigPin2 = 6;
const int echoPin2 = 5;

long duration;
int distance1, distance2;
float r;
unsigned long temp = 0;
int temp1 = 0;
int l = 0;

void find_distance (void);

void find_distance (void)
{
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);

  duration = pulseIn(echoPin1, HIGH, 5000);

  r = 3.4 * duration / 2;
  distance1 = r / 100.00;

  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);

  duration = pulseIn(echoPin2, HIGH, 5000);
```

```
  r = 3.4 * duration / 2;
  distance2 = r / 100.00;
  delay(100);
}

void setup()
{
  Serial.begin(9600);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  delay(1000);
}

void loop()
{
  find_distance();

  if(distance2 <= 35 && distance2 >= 15)
  {
    temp = millis();

    while(millis() <= (temp + 300))
    find_distance();

    if(distance2 <= 35 && distance2 >= 15)
    {
     temp = distance2;

     while(distance2 <= 50 || distance2 == 0)
     {
       find_distance();

       if((temp + 6) < distance2)
       {
        Serial.println("down");
       }
       else if((temp - 6) > distance2)
       {
        Serial.println("up");
       }
     }
    }

    else
    {
      Serial.println("next");
    }
  }
```

```
  else if(distance1 <= 35 && distance1 >= 15)
  {
    temp = millis();

    while(millis() <= (temp + 300))
    {
       find_distance();
       if(distance2 <= 35 && distance2 >= 15)
       {
         Serial.println("change");
         l = 1;
         break;
       }
    }

    if(l == 0)
    {
      Serial.println("previous");
      while(distance1 <= 35 && distance1 >= 15)
      find_distance();
    }

    l = 0;
  }

}
```

First, the `find_distance` function is implemented. The value of the `pulseIn` function to the `duration` variable is assigned. It does not wait for more than 5000 us for the ultrasonic sound to came back. This means that it does not measure more than 60 cm. A distance is defined for detecting movements. The function returns zero if the distance is greater than 60 cm. After the value of the `duration` variable is set, the distance is calculated using math formulas. In the `setup` function, the trigger is initialised and the echo pins of both sensors are established as input and output.

In the `loop` function, the `find_distance` function is called. It stores the current distance measured by the sensors in two variables: `distance1` and `distance2`. The program calls function `find_distance` all the time.

Next, there are several conditions in the code. Each of them will be discussed in order.

The first condition stands for the right sensor. If the user places their hand in front of the right sensor within the range of 15 to 35 cm, this condition becomes true. Inside this condition, a few actions are performed. First, the number of milliseconds since the Arduino board began running the current program using the `millis` function in `temp` variable is stored. Secondly, there is a `while` loop, which measures the distance for another 300 ms. It helps to find the difference between the swipe and stay of the user's hand in front of the right sensor. If the user places their hand in front of the right sensor for more than 300 ms, it tries to detect either scroll up or down. As long as the user does not remove their hand from the right sensor, the `find_distance` function is called over and over again. If the user moves their hand away from the right sensor, then it is detected as an "`up`". Otherwise, it is detected as a "`down`". If the user does not place their hand in front of the sensor for more than 300 ms, then it is just "`swipe`".

The second condition stands for the left sensor. If the user places their hand in front of the left sensor within the range of 15 and 35 cm, this condition becomes true. The same thing happens again with the `while` loop; it measures the distance for another 300 ms. If the user swipes their hand from the left sensor to the right sensor, the hand motion is recognised as "`change`". Otherwise, if the user swipes their hand in front of left sensor, then the action "`previous`" is carried out.

3.2 Python Code

```
# gesture control python program for controlling certain functions in
# windows pc
# Code by BalaAppu
# Website: www.electronicshub.org

import serial
import pyautogui

Arduino_Serial = serial.Serial('com12', 9600)

while 1:
	incoming_data = str (Arduino_Serial.readline())
	print incoming_data

	if 'next' in incoming_data:
		pyautogui.hotkey('ctrl', 'pgdn')

	if 'previous' in incoming_data:
		pyautogui.hotkey('ctrl', 'pgup')

	if 'down' in incoming_data:
		#pyautogui.press('down')
		pyautogui.scroll(-100)

	if 'up' in incoming_data:
		#pyautogui.press('up')
		pyautogui.scroll(100)

	if 'change' in incoming_data:
		pyautogui.keyDown('alt')
		pyautogui.press('tab')
		pyautogui.keyUp('alt')

	incoming_data = "";
```

The 5 commands from Arduino, "`down`", "`up`", "`next`", "`change`", "`previous`", are sent through the serial port to Python for the purpose of detecting the hand gestures. In Python, the serial and pyautogui libraries are added using `import`. Then a serial port object

named `Arduino_Serial` is created before `print incoming_data` prints the incoming serial data from Arduino.

Next, there are 5 `if` statements for the 5 commands from Arduino and the operations they will perform, respectively. Table 1 demonstrates the incoming command and the corresponding operation and the corresponding action in the web browser.

Incoming command	**Operation**	**Action in web browser**
`next`	`ctrl+pgdn`	Move to the next tab
`previous`	`ctrl+pgup`	Move to the previous tab
`down`	↓ (Downward arrow)	Scroll down
`up`	↑ (Upwards arrow)	Scroll up
`change`	`alt+tab`	Switches the tab

Table 1. Resulting Incoming Command, Operation, and Action in Web Browser from the 5 `if` Statements

4.0 Division of Tasks

Group members: Taiyr Begeyev, Youn Jae Jo, Ailin Liu

Begeyev, Taiyr

- Building circuit
- Report: Explaining Arduino code (3.1)

Jo, Youn Jae

- Modifying source codes
- Report: Introduction (1.0), explaining Python code (3.2)

Liu, Ailin

- Building circuit
- Report: Circuit diagram using Fritzig (2.2 Figure 1)

Works Cited

Ravi. "Arduino Based Hand Gesture Control of Your Computer." *Electronics Hub*, Electronicshub.org, 15 Nov. 2017, www.electronicshub.org/arduino-based-hand-gesture-control-computer/.

"What Is HMI?" *Inductive Automation*, Inductive Automation, inductiveautomation.com/what-is-hmi.