International Journal of Advanced Smart Convergence Vol.7 No.2 112-118 (2018)
http://dx.doi.org/10.7236/IJASC.2018.7.2.112

IJASC 18-2-13

Design and Implementation of Vehicle Internal Alarm System using Raspberry-pie Multi-sensor

MyeongBok Choi*, SungKon Park**

**Department of Multimedia Engineering, Professor, Gangneung-Wonju National University, Wonju-si, Kangwon-do, Korea, cmb5859@gmail.com*
***Department of Multimedia Engineering, Professor, Gangneung-Wonju National University, Wonju-si, Kangwon-do, Korea, spark@gwnu.ac.kr*

Abstract

This paper describes the design and implementation of a vehicle internal alarm system using raspberry-pie and gas sensor. It provides a notification system for sleepiness during driving, a driving time notification system and a smoking detection system. We coded using 'Python'. And we use 'MySQL' and 'PHP' to build the necessary servers and web pages for gathering sensing data and monitoring. The developed system was tested by several methods. All experiments showed satisfactory response signals and detected with immediate responses.

Keywords: *Vehicle Internal Alarm System, Raspberry-pie, Multi-sensor, Smoking Detection System*

1. Introduction

Today, car accidents are a hot topic brought up when watching the news. These accidents commonly arise due to as many calls it, 'sleep-driving'. Many drivers will close their windows to either use the heater or air-conditioner, however at the same time, the carbon dioxide (CO_2) that is created as we breathe out will build up inside our cars. This will eventually cause the people in the car to become drowsy which may lead to an accident. To prevent this from happening we have built a microcomputer called Raspberry-pie[1] which warns the driver to ventilate the car when the possibility of sleep-driving seems too high. On the other hand, many people today use car sharing services and although smoking while using such benefits are banned, it's extremely hard for owners of these car sharing services to completely exterminate the use of cigarettes. Furthermore, for future customers the stench of cigarettes may be very uncomfortable. Hence to fix this issue, the 'Raspberry-pie' can be used to inform the car sharing service owners/companies if smoking is detected within the cars. Finally, long-distance drives can increase the chances of a car accident; therefore,

a system will be created to inform the driver to take breaks during certain intervals. There were many researches using sensor in the traffic field [2]. But this paper is a research for easy application to the real world. Section 2 of the paper briefly describes the major hardware components used in the system. Section 3 describes the design and implementation of the system in detail, section 4 explains the experimental results and section 5 will be the conclusion.

Manuscript Received: May. 17, 2018 / Revised: May. 24, 2018 / Accepted: May. 30, 2018
Corresponding Author: spark@gwnu.ac.kr
Tel: +82-33-760-8683, Fax: +82-33-760-8681
Department of Multimedia Engineering, Gangneung-Wonju National University, Korea

2. Major hardware and software environment

There are three main functions used to create the system in raspberry-pie. These include an alarm that notifies the need of ventilation by the measure of air quality, a smoke detection system and a Timer-based 'break-reminder' system. Inside the car, the raspberry-pie's air quality sensor [3, 4] will measure the carbon dioxide levels in real time. If the levels are too high the system will output a message to ventilate the car. Inside the car, the raspberry-pie's air quality sensor will measure the carbon dioxide levels in real time. If the levels are too high the system will output a message to ventilate the car. Furthermore, the timer function in raspberry-pie will immediately activate as the car starts up and will output a message to take a break during the chosen time intervals to avoid sleep-driving for long-distance drivers. For the car sharing services, raspberry-pie's smoke detection sensor [3, 4] is placed within the car and will directly send the customer's information through the owner/company's server's real time if smoking is detected.

Figure 1. Air quality

Figure 2. Smoke detection sensor

Figure 3. Raspberry-Pie

The most important component in the proposed system's software environment is the construction and reconfiguration of the server as the correct configuration as it will allow the server to manage the information from the raspberry-pie. To successfully achieve this system, an Apache web server [5] will be made and My-SQL DB [6, 7] will be installed and used as a database. Also we used three different types of languages including, PHP [7, 8], Python [9] and JavaScript to implement the system.

3. System Design and Implementation

3.1 System Design and Algorithm

Figure 4 shows the overall structure of the site's design and implementation. When the air quality sensor detects a certain amount of carbon dioxide in the vehicle, the system will inform the driver through either a display or sound system in the vehicle. Meanwhile a transmission system will be constructed for the car renting company. As soon as the smoke sensor inside the car encounters smoke, it'll quickly record the time of when the smoke was recognized and the car number which will be sent directly to the car company. Also, for the timer inside the car, a system was implemented that will notify the driver to take a break after a certain predefined period has elapsed since the vehicle was started.

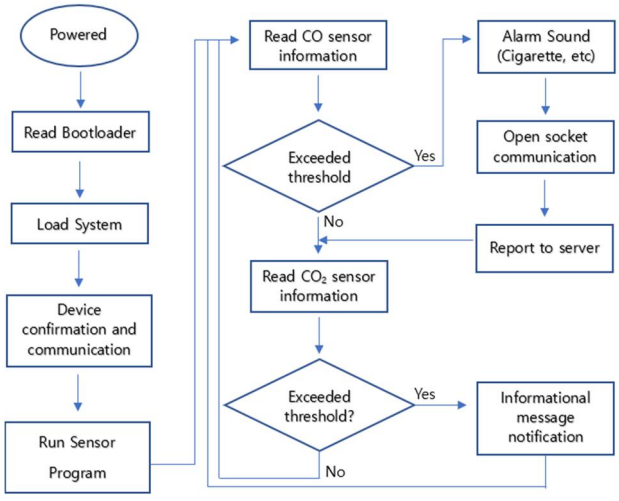


Figure 4. Configuration Diagram

3.2 System details and implementation

3.2.1 Air quality measurement

Air quality measurements will be made using a sensor that identifies carbon dioxide and air quality called ZP01-MP503. First, when looking at the data sheet in Figure 5, pin 1 is ground, pin 2 is VCC, pin 3 is signal A and pin 4 is signal B. When both signal A and B from the sensor show 0V, the air quality is very good. When signal A shows 0V while B shows 5V the air quality is good. Vice versa, when Signal A shows 5V and B shows 0V the air quality is normal and finally, when both Signal A and B show 5V the air quality is bad. By using this as a basis of air quality measurement we have decided that ventilation is required when the air quality reaches bad (both Signal A and B show 5V).

Pin definition

Pin	Name	Function
1	GND	Input power -
2	5V	Input power +
3	A	Output A
4	B	Output B

Output Signal

Grade	Output A	Output B	Pollution Grade
0	0V	0V	Clean
1	0V	+5V	Light pollution
2	+5V	0V	Moderate pollution
3	+5V	+5V	Severe pollution

Figure 5. CO_2 Sensor Data

The ADC (MCP3008) utilizes 7 pins. As channels 1~7 are left, these are connected to channels 1 and 2. It is important to note that if signal A and B is connected to the MCP3008 directly when emitting 5V it'll most likely damage the ADC; hence it should be adjusted within the range of 3~4V with the appropriate resistance. Now the components are connected as the following. Air quality sensor to channel 0, ZP01-MP503's signal A to channel 1 and ZP01-MP503's signal B to channel 2. This connection structure makes the development much easier. Figure 6 shows a code that reads can read multiple channels, furthermore this code detects and outputs once per second. Ultimately, while channel 0 gets its value from the smoke sensor, channels 1 and 2 gets its value from the air quality sensor. This value is then calculated and the code will output a voltage

value.

The coding in figure 6 is saved as spisensor.py to run. “sudo python spisensor.py” can then be inputted into the command line to run this code, furthermore the results are as shown in figure 7. In figure 7, channel 0 is reading the value from the carbon dioxide sensor while channel 1 and 2 are the values of ZP01-MP503’s A and B. It is also seen that channels 1 and 2 are below 0.2V, so this voltage is treated as 0V. Overall, as A and B are both assumed to be emitting no signals the air quality is good.

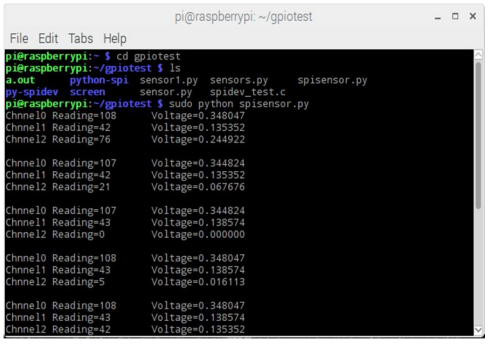
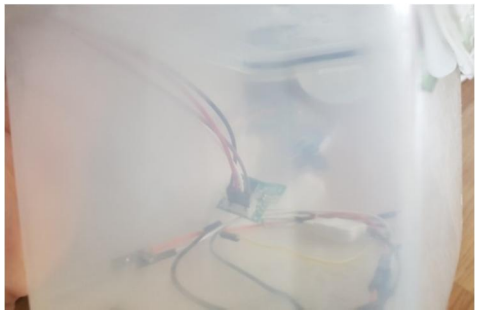


Figure 6. CO_2 Sensor additional coding

Figure 7. CO_2 Sensor

Figure 8 shows an experiment in which a sensor is placed inside a sealed container which is filled with carbon dioxide. This is the type of environment with a lack of oxygen with leads to drowsiness.

When looking at the results of the above experiment in figure 9, it can be seen that both Channel 1 and 2 show the value of 1023. This outputs the fact that the air at this point is bad according to figure 5. When actually measuring the voltage value it comes to about 3.5V, meanwhile the value in figure 7 says 3.296777 which is about 3.3V. When looking at the datasheet of MCP3008, there is a voltage limit of 5.5V, hence to the appropriate resistance was used to attain a stable result. Overall, when signal A and B shows the value of 1023, there will be a message for the driver to ventilate the vehicle.



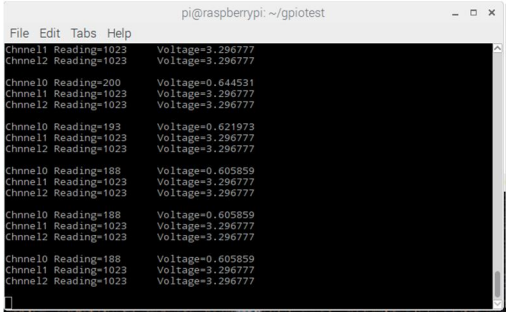


Figure 8. CO_2 Sensor measurement

Figure 9. CO_2 Sensor measurement results

3.2.2 Smoke sensor

Figure 10 shows the process of reading the analog sensor values. The MCP3008 has a total of 8 channels ranging from 0~7. One sensor was installed at channel 0 and a code was made as shown in figure 10 which reads the output value of channel 0 once per second. Moreover, the principle of analog sensors was used to create the coding in figure 10. The ADC MCP3008 receives the voltage of 3.3V and reads the voltage value of channels 0~7 and returns the value of the digital signals. To immediately run the python code that was created above, the command 'sudo' can be used as shown in figure 11. To use raspberry pie's hardware functions, the use of "sudo" command is necessary as you need super user privileges.

Figure 10. Analog sensor value load

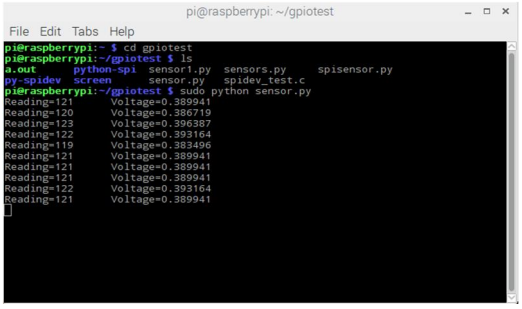


Figure 11. The command 'sudo'

In figure 12, hazardous materials were created to check if the sensors are functioning properly. In this figure the CO sensor (MQ-7) is designed to adjust the sensitivity as it is in analog format. Figure 13 shows that the gas leaks readings ranged from 120 to 555. From here the voltage value from 0V~3.3V was calculated which the results show that it outputs from the range of 0~1023. So, when 3.3/1024 is done, there will be an output of voltage per 1 value. Therefore, if the reading value is 120 is equivalent voltage value is about 0.386719V and this will output a digital value of 120. The CO sensor used in this case detects a certain type of chemical reaction when it is preheated and when it is exposed to gas, the resistance value decreased while the voltage value will increase. Furthermore, the CO sensor usually has a voltage value of less than 0.5 when it's undetectable, meanwhile it can be detected when raised to more than 1.5V.

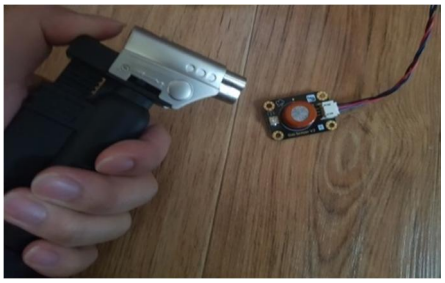


Figure 12. Hazardous material detection check

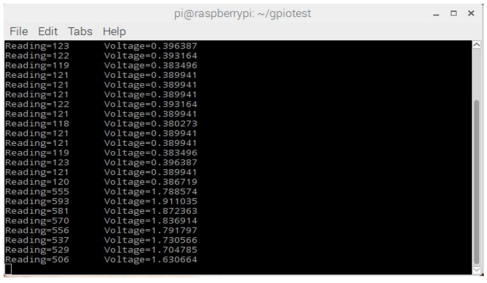


Figure 13. CO_2 Sensor detection screen

3.2.3 Smoking detection page

Figure 14 shows the page that displays the driver's information when smoking is detected through the smoking censor that is connected to the raspberry pie installed on the interior of the vehicle. As soon as the carbon dioxide figure reaches to 500 or more, the driver's personal information including name, contact number, car number, and time are collected via application program using php in the server. This is

Figure 14. Smoking detection information output page (log.php)

constructed so that the page is automatically update every two seconds.

Figure 15 shows the source code for the page shown in figure 14. The function mysqli_connect which is connected to the server's database is being saved to the variable connect. The parameter values indicate the server, ID, password, and database to be connected in order. If the database connections fail, there will be an output of error. The query variable stores a query that reads the name, phone number, car_no, time, and co_measure attributes when the co_measure is greater than 500 in the 'alert' table. The result variable stores a function that sends a query to the connected database. Also the while loop is the code that displays the driver's name, contact number, vehicle number, time and sensor's detected values. The result is a function that stores and returns a single place in an array via the mysqli_fetch_array variable.

```
<?php

 $connect = mysqli_connect("sql.ruger.co.kr", "capstone", "CTxyBWTS9q2tYB2v", "capstone");

if(mysqli_connect_errno($connect))
{
    echo "Error!" ;
    exit;
}

//////////////////////////DB접속//////////////////////////////////////////

echo '<center> <br><br> <h1>흡연자 적발 log </h1></center>' ;

$query = "SELECT name, phonenumber, car_no, time, co_measure FROM alert WHERE co_measure>=500";
$result=mysqli_query($connect, $query);


while($row=mysqli_fetch_array($result)){
    echo '<br><br><center>ㅍ    <b>이름</b> : <font color="red">'.$row['name'].'</font>
        <b>연락처</b> : <font color="red">'.$row['phonenumber'].'</font>     <b>차량번호</b> :
    <font color="red">'.$row['car_no'].'</font>     <b>시간</b> : <font color="red">'.$row['time'].'</font>
        <b>수치</b> : <font color="red">'.$row['co_measure'].'</font><br></center>' ;
    }


mysqli_close($connect);
echo '';

?>
```

Figure 15. Source code of the smoke detected information page

4. Experimentation and Results

The system which was implemented using raspberry pie and its sensor transmits various data using a Wi-Fi source inside the vehicle. Air quality sensor is used to measure the air quality in the vehicle to suggest

ventilation when carbon dioxide levels are too high for the prevention drowsiness while driving and the detection of smoke is performed by using a smoke sensor. Additionally, a there is a timer function operating to remind the driver to take a break. This system was tested by several methods. First, the system was installed on the interior of the vehicle and smoke levels were measured by intentionally smoking cigarettes inside the car. Second, the system was placed on a laboratory desk to detect hazardous substances. Third, the timer function inside the car was tested to observe if the vehicle was notified during certain intervals of driving time. All experiments showed satisfactory response signals and detected with immediate responses.

References

[1] Simon Monk, “Programming the Raspberry Pi, Second Edition: Getting Started with Python”, Oct. 5, 2015.
[2] Myeong-Bok Choi, You-Sik Hong, “Arduino Sensor based on Traffic Safety System using Intelligence”, IJIBC, Vol. 9. No.1 pp. 18-23, February 2017.
[3] Jacob Fraden, “Handbook of Modern Sensors: Physics, Designs, and Applications”, 5th ed., 2016.
[4] John R. Vacca, “Handbook of Sensor Networking: Advanced Technologies and Applications”, 2015.
[5] https://www.apache.org/
[6] https://www.mysql.com/
[7] Lynn Beighley, Michael Morrison, “Head First PHP & MySQL”, 2010.
[8] http://php.net/docs.php/
[9] Mark Myes, “A Smarter Way to Learn Python: Learn it faster. Remember it longer”, Aug. 9, 2017.