**DEVELOPMENT OF MOBILE APPLICATION:   
SMART FOOD SPOILAGE DETECTION USING ARDUINO SENSORS**

A Thesis

Presented to the Faculty of the

**Computer Studies Department**

College of Science

Technological University of the Philippines

Ayala Boulevard, Manila

**by**

**Alexandra Loreinne S. Guzman**

**Mhiguel Rovy G. De Nava**

**Norman V. Cabiling**

**Julie Ann C. Cosico**

**In Partial Fulfillment of the**

**Requirements for the Degree**

**Bachelor of Science in Information Technology**

**January 2022**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | | **Page** |
| Title Page  Table of Contents  List of Tables  List of Figures  List of Appendices | | 1  2  4  5  6 |
| **Chapter 1** | **THE PROBLEM AND ITS SETTING** |  |
|  | Background of the Study  Objectives of the Study  Scope and Limitations of the Study  Significance of the Study | 7  9  11  13 |
| **Chapter 2** | **CONCEPTUAL FRAMEWORK** |  |
|  | Review of Related Literature  Related Studies  Conceptual Model of the Study  Operational Definition of Terms | 15  57  62  64 |
| **Chapter 3** | **METHODOLOGY** |  |
|  | Project Design  Project Development  Operation and Testing Procedure  Evaluation Procedure | 65  69  75  80 |
| **REFERENCES** | | 82 |
| **APPENDIXES** | | 96 |
| **RESEARCHER’S PROFILE** | | 102 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table** |  | **Page** |
| 1 | Recommended Temperatures for Storage Areas | 17 |
| 2 | Recommended Food Storage Conditions and Shelf-life | 18 |
| 3  4  5  6  7 | Preheat Time for Arduino Gas Sensors  List of Components for Arduino Portable Device  Accuracy Testing Procedure of the Arduino Device  Accuracy Testing Procedure of the Arduino Device with the Mobile App  Likert’s Scale | 29  71  77  79  81 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure** |  | **Page** |
| 1 | Screenshot of Arduino Board Parts | 22 |
| 2 | Screenshot of Function that Handles IoT communication | 25 |
| 3  4  5  6  7  8  9  10 | Sensitivity Characteristics of the MQ2  Sensitivity Characteristics of the MQ3  Sensitivity Characteristics of the MQ135  Sensor Circuit of Arduino Gas Sensors  Simplified Sensor Circuit of Arduino Gas Sensors  Combined RS and RL of Simplified Sensor Circuit  Diagram of the Connection between ESP8266, Firebase, Mobile App and Web App over the Internet  Conceptual Model of the Study | 30  31  31  32  33  33  48  63 |
| 11 | Block Diagram of Smart Food Spoilage Detection System | 66 |
| 12 | Use Case Diagram of Smart Food Spoilage Detection System | 67 |
| 13 | Database Design of Smart Food Spoilage Detection System | 69 |
| 14  15 | Schematic Installation of Arduino Device  NoSQL Document Model of Smart Food Spoilage Detection System | 70  72 |
|  |  |  |

**LIST OF APPENDICES**

|  |  |  |
| --- | --- | --- |
| **Table** |  | **Page** |
| 1 | Sample Evaluation Sheet | 83 |

**Chapter 1**

**THE PROBLEM AND ITS SETTINGS**

**Background of the study**

Access to ample, safe, and nutritious food is a basic necessity and a fundamental human right, as it is a key to good health and well-being (Parekh, 2021). With the development of new technologies and improved food production and distribution systems, food has become accessible almost anywhere. Aside from the typical home-prepared foods available on the refrigerators, people can now order ready-to-eat foods online and go to fast-food chains and other eateries outside. The increase in food variety causes food production to account for around 26% of global greenhouse gas emissions. However, about one-quarter of the food the world produces is just thrown away, spilled or spoiled in supply chains, or are wasted by retailers, restaurants, and consumers (Poore & Nemecek, 2018).

In the Philippines, Filipino waste about 308,000 tons of rice every year, and approximately 2,175 tons of food scraps are thrown in the garbage every day in Metro Manila alone. People waste food when they cook or prepare too much (BusinessMirror Editorial, 2019). When there is too much food than needed, the excess food spoils most of the time and ends up in the trash. Furthermore, food spoilage can also be contagious, just like when the fruit ripens, it releases a hormone in a gaseous form called ethylene, a catalyst for ripening fruit. Fruits quickly ripen when exposed to ethylene, giving off more ethylene themselves—forming a domino effect that speeds up the ripening of every fruit and even some vegetables nearby. It is one of the problems of wholesalers or vendors; they cannot spot those fruits that are just starting to ripen, causing the spread of ethylene to every other fruit in the warehouse, also leading to more food waste (Finley, 2015).

The potential for food spoiling increases as more people handles it. Microorganisms can be found everywhere in the environment, from manufacturing to processing to transit to retailers and consumers, and there is always the possibility of food spoilage if foods are exposed to unfavorable conditions. Light, oxygen, heat, humidity, temperature and spoilage bacteria can all have an impact on the safety and quality of food. (United States Department of Agriculture, 2016).

According to World Health Organization (2020), around 600 million, almost 1 in 10 people in the world, fall ill due to food poisoning, where 420,000 die every year. Unfortunately, on the report created by the Partnership for Food Safety Education, 50% of the people who get food poisoning annually are children below 15 years of age (Duran, 2021). One of the culprits of this food poisoning is spoiled foods because of their unnoticeable significant signs of degrading, which induces people to consume them, even after too much time. Spoiled foods often produce an off odor, flavor, weird color, or texture due to naturally occurring spoilage bacteria, which is more recognizable by people on perishable foods such as eggs, meat, fruits, and vegetables. Nevertheless, not all people, especially children, can spot slight degradation in the food, particularly in the early stages. That is why, if food has acquired any spoilage characteristics, it is advised not to consume it anymore because some people who consume spoiled foods experience chills, stomach cramps, diarrhea, nausea, and vomiting (Younghans, 2020).

With all these food poisoning incidents due to food spoilage and the significant amount of food waste produced globally, the researchers decided to devise an Arduino device that could detect food spoilage.

**Objectives of the Study**

***General Objective***

The general objective of the study is to devise a food spoilage detection system.

***Specific Objectives***

The study has the following specific objectives:

1. Design a smart food spoilage detection system with the following components:
   1. Arduino device with the following features:

* Food Inspection
* Food Condition Reporting
* Alerts for detected spoiling and spoiled food
* Dedicated food container
  1. Mobile application as command interface to support the Arduino device with the following features:
* Registration of Arduino device
* Option to select specific food to inspect
* Button to inspect food
* Button to monitor food for signs of spoilage
* Data visualization of sensor readings
* Alert Notification for food condition and device battery life
* Food Storage and Shelf-life Tips

1. Develop a smart food spoilage detection system with the following components:
   1. Arduino device using the following tools:

* Microcontroller (Arduino Uno)
* Gas Sensors for Methane, Ammonia, Ethanol
* Humidity and Temperature Sensor
* ESP8266 Wi-Fi Module
* Other components such as Pushbutton, Buzzer, Red Green Blue Light-Emitting Diode (RGB LED), Liquid-Crystal Display (LCD) Screen, Battery, Breadboard, and Jumper Wires
* Acrylic Panels
* Arduino Integrated Development Environment (IDE)
  1. Mobile application as command interface to support the Arduino device using the following tools:
* Visual Studio Code
* JavaScript
* Node.js
* React Native
* EXPO CLI
* Firebase Realtime Database

1. Test and improve the accuracy of the Arduino device together with the mobile application on foods under three conditions, Fresh, Stale, and Spoiled.
2. Evaluate each system component's acceptability based on applicable ISO 25010 criteria such as functionality suitability, performance efficiency, compatibility, usability, reliability, maintainability, and portability.

**Scope and Limitations of the Study**

The smart food spoilage detection system is a system designed to prevent food from being wasted due to spoilage and help users distinguish spoiled food from safe and consumable foods.

The current system includes the design of the Arduino device that could detect indications of spoilage on food. The device contains a pushbutton for the users to initiate an inspection of food. The device provides a food condition report to indicate the condition levels specified as Safe, Warning, and Danger, informing the user that the food is consumable, perishing soon, and spoiled, respectively, based on the amount of gas emitted by the food. However, the device includes gas sensors MQ2, MQ3, and MQ135, detecting methane, ethanol, and ammonia gasses, respectively, making the spoilage detection limited only to foods emitting the specified gasses. Those foods include meats; fruits such as apple, banana, peach, tomato; fruit juices such as orange juice and apple juice; milk; prepared meals such as Kare-Kare, Bicol Express, Carbonara; and other foods that have milk as an ingredient. The device would only to detect gas emissions no farther than 5 inches away from the food. The alert function immediately notifies the user when the device detects significant changes in a food's condition. The device includes a dedicated food container constructed from acrylic panels with 12 inches in length and 7.5 inches in width and height as a specific place to attach or detach the Arduino device. However, the Arduino device is a standalone device that can be used with or without the food container or the mobile app.

The current system also includes designing the mobile app integrated to support the Arduino device. Using JavaScript as the language for mobile development, React Native as the framework while using Node.js as a runtime environment for handling real-time data, Firebase Realtime Database as the database to store the modules information, and Visual Studio Code along with the EXPO CLI for the development of the mobile app. The mobile app can run on any Android device with at least version 4.4 of the Android Operating System.

The integration of the mobile app as the optional controller allows the user to control the Arduino device. In the app, the user can search for the specific food to be inspected, for more accurate detection, because every food has its own amount of gas emissions when spoiling. The application allows the user to check the food’s condition immediately or monitor the food. During monitoring, data visualization of sensors reading would be available for the user on the mobile app. The application also includes alert notification, notifying the user of significant changes in food condition of the monitored one (safe, warning, danger) and the device’s battery life. The mobile app will also consist of food-related tips to avoid food spoilage or consuming spoiled food.

Each system component would be evaluated by 40 purposively selected respondents composed of ten (10) information technology professionals, ten (10) food specialists, ten (10) fruit and vegetable vendors, and ten (10) parents of children ages fifteen (15) years old and below.

The evaluation instrument that would be used to assess the acceptability of the Arduino device and mobile app consists of the ISO 25010 Software Quality Model (see Appendix A).

**Significance of the Study**

Food is essential to all living beings, particularly humans, because it is the third most important thing needed by the human body to produce energy and growth development. Therefore, it plays a vital role in the promotion of health and prevention of sickness.

The current system will help lessen food waste and mitigate health-related problems caused by food poisoning and food spoilage. The end-users that will benefit from the current system are society, market vendors, parents, and children. As for society, the current system could help in minimizing foods thrown away every single day. Thus, providing a solution to the massive food waste problem in the community. For the market vendors, the current system will be useful in preserving the quality of commodities such as fruits. It could help them immediately separate perished goods that could affect the entire product. Then for the parents, the current system could help ensure that the foods they prepare for their children are safe and consumable. Apart from that, they could also refrain their children from having illnesses and other health-related concerns. For the children, the current system can aid them in distinguishing spoiled food from safe and consumable foods. Also, this could serve as their guide for recognizing the difference between what should be and should not be eaten or fresh and stale foods. Lastly, to future researchers, this study could serve as a guide or be a reference in conducting further research about food spoilage detection systems and other related future studies.

**Chapter 2**

**CONCEPTUAL FRAMEWORK**

This chapter presents the review of related literature, related studies, the conceptual model of the study, and the operational definition of terms relevant to this study.

**Review of Related Literature**

**Introduction**

One of the most important things for living beings is food, as humans need to eat and require a reliable food supply to meet metabolic requirements for maintenance, growth, and reproduction (The Scientific World, 2018; Hammond et al., 2015). However, food has a life cycle wherein it eventually undergoes the process of spoiling. As the food is harvested, either in the form of vegetables, fruits, or meats from animals, it is detached from the source that gave it life, beginning to die immediately (Wonderopolis, n.d.). Being unable to handle and monitor the food throughout its life cycle could result in food spoilage, or technically food wastes.

The problem is not everyone is capable of determining if food is already spoiled. Furthermore, not everyone can monitor the life cycle of food based on its shelf-life or as it spoils. Thus, resulting in the consumption of spoiled food, which can lead to illness, and in the worst case, death (Amit et al., 2017). As the technologies continue to advance, various tools and approaches in monitoring and preventing food spoilage arise. One way is through the use of gas sensors for monitoring gas concentration that indicates spoilage.  The data that would be gathered from these sensors could be more valuable if analyzed and reflected in real-time on easy access mobile devices, which send alert messages about the status of or any significant changes in the food condition.

***Food Spoilage***

According to Cambridge University Press (n.d.), food spoilage is the process in which food stops being good enough, suitable, or edible for humans to use or eat. In scientific knowledge, food spoilage is a metabolic process causing foods to be unacceptable for human consumption due to sensory characteristics changes such as visual, tactile, olfactory or smell, and gustatory or taste (Sahu & Bala, 2017). Furthermore, aside from these changes, physical, microbiological, and chemical changes could also result in food spoilage (Petruzzi et al., 2017).

According to Amit et al. (2017), the three primary factors affecting food spoilage are physical, microbiological, and chemical changes and thus, identify spoilage in these three classifications. The key factors affecting physical spoilage include moisture content, temperature, and physical damage. Moisture content causes food degradation when food loses or gains water or moisture content. Temperature changes affect storage time and speed up the decay process, for there are specific optimal temperatures for every ingredient. Food deteriorates faster at higher temperatures facilitating the exponential growth of microorganisms (Henderson, 2020). Physical Damage such as bruises and cracks on raw produce, improperly packaged foods, dented cans, and broken packages introduces the risk of bacterial growth as they provide places for air, light, and microorganisms to enter and grow (Wright, 2019).

On the other hand, microbiological spoilage occurs due to the growth and activities of microorganisms such as bacteria, yeast, and molds that produce enzymes leading to the formation of undesirable by-products in the food (Petruzzi et al., 2017). Microorganisms utilize food as a source of nutrients for their growth by breaking down food materials, synthesizing new compounds, which results in the deterioration of food (Kumar et al., 2018). Chemical food spoilage alters food’s sensory characteristics, including changes in color, appearance, texture, aroma, flavor, nutrition, safety, and functional properties when food components react with each other or with some added component like oxygen and light that could cause lipid oxidation (rancidity) and enzymatic browning (Erkmen & Bozoglu, 2016).

All these factors are not necessarily mutually exclusive, as one factor can stimulate another. Hence, the current system primarily focuses on monitoring the physical changes such as temperature and moisture, as it can arouse other factors affecting food spoilage. According to Singh and Anderson (2004), the temperature is one of the most critical variables that influence most types of spoilages alongside pH, water activity or moisture, exposure to oxygen, and light for dairy products, frozen meats, fruits, and vegetables. Generally, according to Albrecht (2018), a food specialist, the recommended temperatures for storage areas are shown in table 1.

Table 1.

*Recommended Temperatures for Storage Areas.*

|  |  |  |
| --- | --- | --- |
| Storage Area | Temperature (°F) | Temperature (°C) |
| Cupboard or Pantry | 50 – 70 | 10.00 – 21.11 |
| Refrigerator | 34 – 40 | 1.11 – 4.44 |
| Freezer | 0 or below | -17.78 or below |

Aside from recommended temperature for storage areas, each food has optimal food storage conditions such as temperature and humidity used as a guide to maintain the quality and maximize the storage life of foods. Shown in table 2 are the optimal storage temperature, relative humidity, and approximate shelf-life of some foods with regards to optimal storage conditions (Cooperative Extension Publications, 2021; Engineering ToolBox, 2004). These tables would guide the researchers in setting threshold values and analyzing food conditions during food inspection and monitoring. Aside from these, the current system would monitor the end product, specifically the gas production produced by microorganisms and chemical reactions causing a foul odor.

Table 2.

*Recommended Food Storage Conditions and Shelf-life*

|  |  |  |  |
| --- | --- | --- | --- |
| Food | Temperature (°F) | Relative Humidity (%) | Approximate  Storage Life |
| Apple | 30 – 40 | 90 – 95 | 1 – 12 months |
| Avocado (ripe) | 38 – 45 | 85 – 95 | less than 6 weeks |
| Avocado (unripe) | 45 – 50 | 85 – 95 |  |
| Banana (ripe) | 56 – 60 | 85 – 95 | 2 – 3 weeks |
| Cantaloupe (3/4 slip) | 36 – 41 | 90 – 95 | 15 days |
| Cantaloupe (full slip) | 32 – 36 | 90 – 95 | 5 – 14 days |
| Peach | 31 – 32 | 90 – 95 | 2 – 4 weeks |
| Pineapple (Partially ripe) | 50 – 55 | 85 – 95 | 14 – 20 days |
| Tomato (mature green) | 50 – 70 | 90 - 95 | 1 – 3 weeks |
| Tomato (firm ripe) | 45 – 50 | 90 - 95 | 4 – 7 days |
| Milk | 32 – 40 |  | 1 week |
| Milk (opened, canned) | 32 – 40 |  | 4 – 5 days |
| Cooked Meat Dishes | 40 |  | 3 – 4 days |

***Detecting Food Spoilage***

The signs of food spoilage include changes in appearance, color, texture, odor, or taste (Magoulas, 2016). These indications of spoilage, such as slimy film, visible mold, discoloration of food, funky or rancid odor, abnormal texture, are perceivable by human senses. Hence, the most common and easiest way of examining food and detecting food spoilage is through taste, smell, touch, and sight (Cooper, 2019).

According to Miller (2017), sensory evaluation is a scientific discipline used to evoke, measure, analyze, and interpret human reactions to the characteristics of foods as perceived by sight, smell, taste, touch, and hearing senses. Furthermore, the Canadian Institute of Food Safety (2017) stated that sensory monitoring is only one of several techniques to monitor food safety and detect food spoilage. Other methods include chemical monitoring such as checking acidity levels or conducting a nutritional analysis, physical monitoring such as temperature or weight checking, polymerase chain reactions, spectroscopy, microbiological techniques like cell culture, and electrochemical or esoteric assays. However, the detection techniques mentioned are mostly time-consuming and require professional experts and specialized tools. As the food industry looks for alternatives, more innovative and accurate methods to monitor the food safety and quality control of food products, the use of detection technologies such as gas sensors devices in electronic nose systems is recognized to identify food spoilage (Mohammadi & Jafari, 2020).

The gas sensor-based detection systems have been made possible because of the nature that the spoilage process incurs. In the study conducted by Benabdellah et al. (2017), an electronic nose composed of array sensors comprised of humidity and temperature sensor and TGS822 gas sensor detected and defined the rotten food before the human nose in spoiled meat. Another study by Parmar et al. (2020) utilized Arduino-based sensors for detecting freshness and quality of food by identifying the gas levels as food decay. These methods open a way for alternative spoilage detection that is easier to use, cost-efficient, and based on open-source electronic platforms. The current system would utilize an array of Arduino sensors to monitor and detect the smell of food, particularly the gas produced by decaying food that causes an offensive odor without human intervention.

***Gas Formation on Spoiled Food***

The most common reason for food releasing an awful odor is the growth of spoilage microorganisms such as bacteria, yeasts, and mold. This odor is either produced directly by microbes or released from food as microbes decompose it (Wolfe, 2014). In the spoilage process of meat, microbes break down food molecules for their own metabolic needs. Microbes first used sugars and carbohydrates, degrading plant pectin, attacking proteins producing volatile compounds with characteristic smells such as ammonia, amines, and sulfides (Abdel-Aziz et al., 2016). Moreover, the research from Eom et al. (2014) states that during the spoilage process of meats, bacteria and enzymes decompose meat composition such as water, protein, fat, and some carbohydrates that resulted in the production of gases such as ammonia, hydrogen sulfide, and ethanethiol.

The formation of ammonia gas is one of the indications of food spoilage as organic matter decomposes. As the food spoils, minimal amounts of ammonia for about less than 30 parts per million (ppm) are released (Mohammadi & Jafari, 2020). The current system would use a gas sensor that conforms with the detecting concentration requirement stated earlier to detect ammonia gas in determining the food's condition.

On the other hand, ethanol, also known as alcohol, is another gas formed due to food spoilage, especially in fresh-cut fruit (Technical Research Centre of Finland (VTT), 2015). According to Shahzad et al. (2018), the ripening of fruits can be associated with changes in color, taste, sugar, and ethanol content. As fruit like banana ripens, its chemical properties change and produce different gases like ethylene and ethanol in small out that increase over time. Moreover, aside from bananas, fruits such as tomato, avocado, apple, melon, peach, and kiwi produce ethylene gas as they ripen and rise even more as they spoilt (Moirangthem & Tucker, 2018).

The dominant microorganisms involved in the spoilage of fruits and vegetables are fungi and bacteria. Bacterial spoilage softens the tissues characterized as rot, making vegetables degenerate into a slimy mass. After starches and sugars are metabolized, unpleasant odors, along with lactic acid and ethanol, are produced (Bhat & Rao, 2019). In some studies, spoiled fruits such as orange, grape, apple, pineapple, banana, and papaya are used as an alternative for ethanol production as biofuel through fermentation and microorganisms (Balasubramanian et al., 2017; Chitranshi & Kapoor, 2021; Kumar & Senan, 2020; Moneruzzaman Khandaker et al., 2020). In this current study, the production and concentration of ethanol in fruits would be detected and monitored through a gas sensor to avoid spoilage or consumption of spoiled fruits.

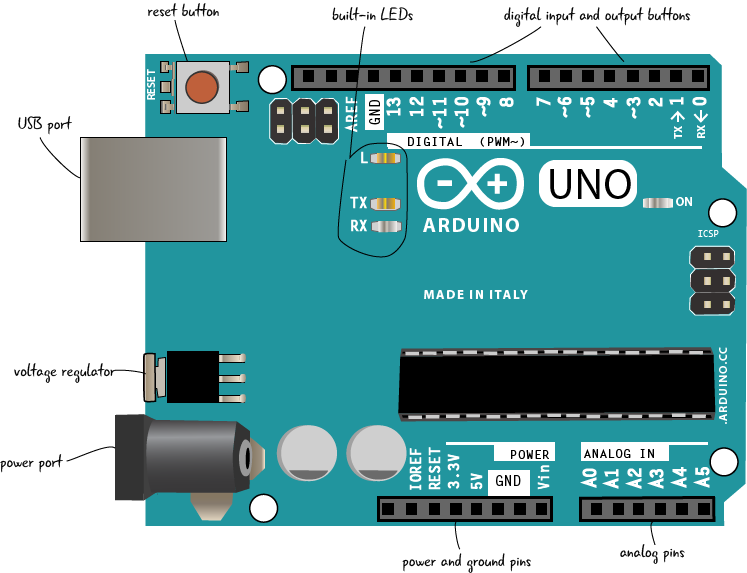
Another type of gas released when food spoils is methane (Helmke, 2021). According to Matindoust et al. (2016), methane, with a chemical formula CH4, is a volatile hydrocarbon perceived as a bad smell given off by spoiled fish, dairy, eggs, fruits, and meats. The presence of methane in all types of food would be detected and monitored to indicate spoilage using an Arduino-based gas sensor.

***Arduino***

Arduino is a piece of technology, a tool, a community, and a way of thinking that affects how people use and understand technology. Arduino is a tiny open-source electronics platform designed to use a microcontroller and other inputs and outputs (Nussey, 2018). Similarly, Geddes (2016) said that Arduino is a small and inexpensive computer that can be programmed with an easy-to-use Arduino IDE to connect and control various electronic parts. It has several pins that can be set as input, meaning it can receive data from switches, buttons, and sensors, or output, which means it can send data to control items such as motors, lights, and buzzers. With all these capabilities, Culkin and Hagan (2017) stated that Arduino could take different information from the environment using various sensors. Furthermore, Louis (2016) explained that Arduino could also receive and send data over the internet, making it a popular platform for Internet of Things (IoT) product development and other STEM projects. With all these features and capabilities, Arduino is a better choice for the researchers to utilize Arduino as a microcontroller device for the current system.

***Arduino Basic Hardware Components***

Based on Arduino2Go (2016), the basic hardware components of an Arduino board consist of a reset button, Universal Serial Bus (USB) port, voltage regulator, power port, built-in Light-Emitting Diodes (LEDs), digital Input/Output (I/O) pins, TX/RX pins, ATmega328P**,**power and ground pins, and analog pins. The reset button is for restarting the code uploaded in the Arduino. USB port serves as a cable connection on a computer for programming the Arduino board and may also serve as a power source. Voltage regulator converts power plugged into the power port into five (5) volts and 1-ampere standard used by the Arduino. The power port is a barrel-style connector that can be connected to a battery or a wall source power. The built-in LEDs acts as an indicator for power and sending or receiving data. Digital I/O pins are the holes on the side of the board that can sense the outside world as input and control peripherals or the output. ATmega328P is the black chip in the middle of the board that acts as the brain, interpreting inputs/outputs and programming code uploaded in the Arduino board. Power and ground pins are power-related pins, while analog pins take sensor readings in the analog range. TX/RX pins are special pins used for serial communication (Ghosh, 2018).



***Figure 1.***Screenshot of Arduino Board Parts

Likewise, according to Hussaini (2020), the Arduino Uno board is versatile because all the tiny components mounted on it have a purpose. It consists of an ATmega328P microcontroller, 16MHz crystal Oscillator, serial communication, voltage regulator, and more to support the microcontroller, 14 digital input/output pins, and six (6) analog input pins, an ICSP header, a power barrel jack, a USB connection, and a reset button. Besides, according to Makerspaces.com (2017), Arduino is an excellent platform for prototyping projects and inventions but can be hard when choosing the appropriate board. There are many variations of Arduino board, such as the LilyPad board for wearable projects, an Arduino Pro Mini for small form ones, and the most widely used and documented is the Arduino Uno board. For this reason, the study focuses on utilizing the Arduino Uno board for the current system, and this Arduino board is programmed using the software Arduino IDE.

***Arduino IDE Software***

     According to Louis (2016), Arduino is the software used to develop sketches in Arduino and mainly comprises four parts: the text editor, message area, text, and console toolbar. The text editor is where the code is written using a simplified version of the C++ programming language. The message area displays the error and gives feedback when saving exporting the sketch. The next part is the text where the console displays text output by the Arduino IDE; it includes the complete message of errors and other information. The last part is the console toolbar, where various buttons such as Verify, Upload, New, Open, Save, and Serial Monitor are located.

     Based on the team of Circuito (2018), Arduino serial monitor is used mainly for the interaction of the Arduino board using a computer and is used as a tool for real-time monitoring and debugging. Similarly, Arduino serial plotter is also a component of Arduino IDE and generates real-time monitoring of the serial via a graph. The serial plotter gives a visual display to analyze the data better. In Arduino, the code structure has built-in libraries that provide basic functions. Additionally, it’s probable to import other libraries and extend the Arduino board capabilities and features further. Arduino libraries are divided to interact with specific components and modules. These libraries can be implemented inside the sketches in Arduino IDE.

     Based on Badami (2016), In the Arduino IDE is where all the Arduino programs are made. The Arduino IDE allows writing sketches. All sketches in Arduino follow the main structure, including the function blockssetup and loop*.*The setup is the preparation block. This block is only called once and executed first when the Arduino program is uploaded. This block is also used to initialize pin modes, start the serial communication, and is always included even if no statements are written to be executed.

     On the other hand, the loop function is the execution block where the set of statements in this function is executed repeatedly. The current system uses Arduino IDE software to write and debug program code and upload sketches to the Arduino device. This program code is written using the designated programming language of Arduino IDE.

***Arduino Programming Language***

According to Copes (2020), Arduino Programming is built from the top of the C++ programming language. A program written in Arduino programming is called a sketch. Arduino programming is generally used in the Arduino IDE. Arduino IDE is the official software introduced by Arduino.cc. The Arduino IDE is used primarily for editing, compiling, and uploading sketches on the Arduino board. Arduino IDE is open-source software available for MAC, Windows, Linux, and compatible with the Java Platform that has a pre-built function that acts as an essential tool for debugging, editing, and compiling the code in the Arduino IDE (Fezari & Al Dahoud, 2018). The current system uses the Arduino Programming Language for writing sketches using Arduino IDE to be uploaded on the Arduino device.

Prasaath (2018) demonstrated on a project that the setup function is called to establish serial communications for communicating with the Wi-Fi module while the loop function is called to fetch and store the data in the sensors to the initialized variables. The transmission of data to the cloud using a Wi-Fi module is depicted by the ESP8266 function shown in Figure 2. With the use of ESP8266, the Arduino can communicate with other devices, software, and even mobile applications.



***Figure 2.***Screenshot of Function that Handles IoT communication

***Arduino Peripheral Modules and Gas Sensors***

According to Mitchell (2018), when it comes to Arduinos and other microcontrollers, the word peripherals refer to a component on the board dedicated to a duty independent of the CPU. While Dalmaris (2021) stated that Arduino could not do much alone; fortunately, countless peripherals can support Arduino in sensing the world, such as sensors, or changing something in it, such as actuators. These peripheral modules are utilized by connecting them to the Arduino’s digital input and output pins. Moreover, they are also available on almost all functionalities, including measuring light, motion, temperature, humidity and moisture, vibration, pressure, sound, gas, and more. With these sensors of all kinds, the current system will only utilize temperature and humidity sensors and gas sensors to detect food spoilage (Alexaki and Dalmaris, 2020).

The commonly used temperature and humidity sensor is the DHT11 (DHT11–Temperature and Humidity Sensor, 2021). The DHT11 sensor measures temperature and humidity complex with a calibrated digital signal output. It has an exclusive digital-signal-acquisition method and temperature and humidity sensing technology that guarantees high reliability and long-term stability. Furthermore, each DHT11 element is calibrated with high humidity accuracy in a laboratory. Its calibration coefficients are stored as programs in its OTP memory used by the internal signal detecting process (Mouser Electronics, n.d.). Zuo (2020) also claims that the sensor DHT11 provides a pre-calibrated digital output and is also considered a unique capacitive sensor element that measures relative humidity and temperature by a negative temperature coefficient (NTC) Thermistor. With that said, DHT11 no longer needs to be calibrated to have accurate temperature and humidity measurements.

However, it is different for gas sensors. Gas sensors are devices that can measure different types of gasses. They vary widely in size, range, and gasses that can be detected (Chaware, 2021). In the current study, only three gas sensors will be used, such as MQ-2, MQ-3, and MQ135 sensors.

The MQ2 Gas sensor can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane, and Carbon Monoxide concentrations anywhere from 200 to 10000ppm (Last Minute Engineers, 2020a). While the MQ3 sensor is most suitable for detecting Alcohol, Ethanol, and Smoke concentrations anywhere from 25 to 500ppm (Last Minute Engineers, 2020b). Then the MQ135 sensor is used to detect Ammonia, Sulphide, and Benzene steam presence in the range of 100ppm to 1000ppm (Macfos, 2021). Measuring the PPM of these sensors is the best way to have some accuracy on sensor readings and be able to distinguish its gasses from another. However, in order to measure PPM, the sensors must be calibrated first (MQ2 Gas Sensor, 2018).

For instance, Heyasa and Galarpe (2017), in their study Initial Development and Testing of Microcontroller-MQ2 Gas Sensor for University Air Quality Monitoring, the core hardware of the produced gas sensor includes a microcontroller Arduino and an MQ2 gas sensor. The microcontroller device used was an Arduino Uno R3, and for the MQ2 gas sensor device, the P/N 80-031414-1 MQ-x gas sensor kit was utilized. LPG, butane, propane, methane, alcohol, hydrogen, smoke, and carbon monoxide are all detected with the MQ2 gas sensor. However, as stated in their conclusion, the design and sensor need improvement and calibration for better and accurate sampling.

***Sensor Calibration***

Sensor calibration refers to the adjustments performed to make a sensor or instrument function and operate accurately or error-free. Calibrating sensors is vital to yield accurate measurements and ensure that the process runs efficiently and safely (Sommer, 2019). According to Dorcea et al. (2018), sensor calibration is the process of obtaining the most accurate sensor data or measurement. Since MQ sensors measure different types of gas simultaneously, knowing the substance to measure is necessary to extract the formula from the datasheet provided by the manufacturer. However, these formulas for each gas type of MQ sensor are not explicitly given in the datasheet but can be extracted using the datasheet graphic.

But before proceeding with calibration, the MQ sensors must undergo a one-time preheating process. Every MQ gas sensor has a specific preheat time given on the datasheet (Carrillo-Amado et al., 2020). Table 3 shows the different preheating times for each Arduino Gas Sensors. In addition, Hareendran (2020) believed that the MQ sensors also need to be powered up and preheated for about thirty (30) seconds to three (3) minutes to warm up before starting to work. Similarly, for ElectronFun.com (2014), sensor readings take about two (2) to three (3) minutes after powering up to be reliable.

Table 3.

*Preheat Time for Arduino Gas Sensors.*

|  |  |
| --- | --- |
| Gas Sensor | Time |
| MQ-2 | Over 24 hours |
| MQ-3 | Over 24 hours |
| MQ-135 | Over 24 hours |

According to Christophe (2021), the sensors must be placed in clean air, preferably outdoors, during the calibration. The R0, or the product of the sensor calibration process, is also ideally determined each time the sensor is turned on because the resistance of MQ sensors can change over time due to the presence of dust affecting the sensors' active surface. Since the current study utilizes MQ sensors as gas sensors, preheating and calibrating these Arduino Gas Sensors must be considered or performed to yield accurate measurement of gas in ppm value.

***Sensor Sensitivity Characteristics Curve***

Essential pieces of information from the datasheet are the graphs represented in figures 3, 4, and 5 that show the sensitivity characteristics curve of each sensor (Hanwei Electronics Co., Ltd., 2010a, 2010b, 2010c). These curves show the concentration of a gas in ppm according to the resistance ratio of the sensor (RS/R0). The RS refers to the resistance of the sensor that changes depending on the concentration of gas. On the other hand, R0 is the resistance of the sensor at a known concentration without the presence of other gases or in the fresh air. These figures serve as the reference for determining the R0 value of a specific substance measured for each sensor necessary for measuring gas concentration.

Chart

Description automatically generated

***Figure 3.***Sensitivity Characteristics of the MQ2

Chart

Description automatically generated

***Figure 4.***Sensitivity Characteristics of the MQ3

Chart

Description automatically generated

***Figure 5.***Sensitivity Characteristics of the MQ135

***Sensor Circuit***

The sensor circuit of the Arduino gas sensors is shown in figure 6 based on the datasheet from Hanwei Electronics Co., Ltd. (2010a, 2010b, 2010c). According to Jaycon Systems (2021), this figure depicts that the sensor has four pins to have connections and also indicates that the sensor has a built-in resistor between pins A and B, which value varies depending on the gas concentration. This resistance is called RS or variable sensor resistance. Moreover, according to Last Minute Engineers (2020a), this resistance is inversely proportional to the gas concentration. As gas concentration increases, resistance decreases. On the other hand, the resistance increases if the gas concentration is low.

Diagram, engineering drawing

Description automatically generated

***Figure 6.***Sensor Circuit of Arduino Gas Sensors

The simplified version of the sensor circuit is shown in figure 7. This figure helps understand how the formula for RS, which would be used to calculate R0, is derived. Based on figure 7, RS is in series with RL or load resistance, and when combining these two resistors, the sensor circuit is simplified shown in figure 8 (Jaycon Systems, 2021).

Graphical user interface, application, Teams

Description automatically generated

***Figure 7.***Simplified Sensor Circuit of Arduino Gas Sensors

A white circle with a black background

Description automatically generated with low confidence

***Figure 8.***Combined RS and RL of Simplified Sensor Circuit

***Formulas and Calculations***

Based on Ohm's law shown in Equation 1, the formula for current is derived*,*shown in Equation 2 (Fluke, 2021). This formula can be applied in finding the current in figure 8, as shown in Equation 3. Using Ohm's law again from Equation 1, the formula of current in Equation 2 is substituted to obtain the output voltage at the load resistor shown in Equation 4, which can be further simplified into Equation 5 by multiplying the load resistance to the current. From Equation 5, the formula for RS is solved by deriving it on Equations 6 to 11. The RS formula in Equation 11 is used to find the sensor resistance values for different gases (Jaycon Systems, 2021).

(1)

(2)

(3)

(4)

(5)

(6)

(7)

(8)

(9)

(10)

(11)

I = current; V = voltage; R = resistance; VC = input voltage; VRL = output voltage;   
RS = sensor resistance; RL = load resistance

According to Agilo Research (2019), in calculating R0, the average of multiple analog readings from the sensor must be taken and converted into voltage using the 10-bit DAC Conversion formula shown in Equation 12. The output voltage resulting from this equation will be used in finding the value of the RS in fresh air using the formula in Equation 11.

(12)

Vout = output voltage; Vref = voltage reference; n = number of bits;   
Din = digital input number or sensor value

Based on the given graphs in figures 3, 4, and 5, the resistance ratio (RS/R0) in the fresh air of all the Arduino gas sensors is constant. These RS/R0 values, together with the computed RS, would be used to calculate the R0 for each Arduino gas sensor with the given formula in Equation 13 (Jaycon Systems, 2016). The calibration process finishes after computing R0, which would be used in calculating the presence of gas in ppm.

(13)

RS = sensor resistance in fresh air; RSR0 = resistance ratio of gas sensor in fresh air

***Calculating Gas Concentration in PPM***

Once the product of the sensor calibration process or R0 is computed, it can now be used in calculating gas concentration in ppm. In order to calculate the PPM for a particular gas, the graph presented in figures 3, 4, and 5 must be analyzed in terms of its scale, the range of data for gas concentration, and the relation between resistance ratio and gas concentration that may seem linear but in reality, it is not. However, the lines will be treated as linear to make use of the formula that relates the ratio and the concentration. In conducting that, the gas concentration at any ratio value even outside of the graph's boundaries. The formula for a line to be used is on Equation 14, and the log-log scale formula is on Equation 15, wherein the log is base 10.

(14)

(15)

y = X value; x = X value; m = Slope of the line; b = Y-intercept

For calculating the slope, choosing 2 points from the graph is required. For instance, Agilo Research (2019) chooses the points (200, 1.6) and (10000, 0.27) from the LPG line on the MQ2 graph. The formula in Equation 16 will be used to calculate slope, but if the logarithmic quotient rule is applied, its formula can be simplified into Equation 17. The chosen points in the graph will substitute on x, x0, y, and y0 to compute the slope or the value of m.

(16)

(17)

After computing the slope, its value will be used in calculating the y-intercept. In performing the equation, choosing one point from the same graph and line is also required. So Agilo Research (2019) chose point (5000, 0.46) on the LPG line from the MQ2 graph. The formula for the y-intercept in Equation 18 is derived from Equation 15, wherein the x and y of the point must be substituted.

(18)

Suppose the slope or value of m and y-intercept or b are computed. In that case, the gas concentration for any ratio can now be calculated using the formula in Equation 19. However, to get the actual value of the gas concentration based on the log-log plot, it is necessary to find the inverse log of x using the formula in Equation 20.

(19)

(20)

***Android***

Based on The Editors of Encyclopaedia Britannica (2020), Android is a mobile operating system for smartphones and tablets. In 2003, Android originated as a project of Android Inc., an American technology firm, to build an operating system for digital cameras. In 2004, the concept evolved into a smartphone operating system. Similarly, according to Chen (2021), the Android operating system was created by Google to be used mainly on touchscreen devices such as cell phones and tablets. The design of Android allows users to interact with mobile devices intuitively, using finger gestures that mimic typical actions like pinching, swiping, and tapping. Google's Android software can also be found in televisions, vehicles, and wristwatches, each with its unique user interface. One of the components of the current system is a mobile app as the command interface of the Arduino device. This mobile app would be developed specifically for Android smartphones.

According to Christensson (2016), Android phones often come with several pre-installed apps and support third-party apps. The Android software development kit is a free tool that allows developers to create apps for Android (SDK). Android applications are written in Java that runs on a Java Virtual Machine (JVM) designed specifically for mobile devices. The "Dalvik" JVM was used until Android 4.4 when it was superseded by the Android Runtime, or "ART." Android apps may be downloaded and installed via Google Play and other sources. The software in an Android is called an android application.

     According to Mansoor (2020), Apps for Android may be written in various languages, including Java, Kotlin, C#, JavaScript, and DART, to mention a few. However, because the Android OS is developed in Java, most Android apps are created in this language. Android Studio is the most often utilized software/program for Android development. Android Studio is an Integrated Development Environment (IDE) for developing Android applications that include all the tools needed to create and distribute apps. As a result, Android Studio is large, heavy, and difficult to launch and browse since it consumes the computer's RAM, ROM, and Processor. Lightweight text editors like Notepad/TextEdit have replaced IDEs in modern development because these editors can open files quickly as less than a second and are particularly useful for making quick changes. Among developers, the most popular is Microsoft's Visual Studio Code derived from the open-source text editor Atom. One of the popular features of VS Code is the ability to install "Extensions" that were produced for free by other developers. Once the correct Android SDK loaded, Dave Holoway, a passionate Senior Developer in London, designed the extension "Android" to build and debug Android apps using VS Code.

***Visual Studio Code***

Visual Studio or VS Code is a free source code editor that helps programmers write codes, debug, and correct codes using IntelliSense. It aids the users in writing codes effortlessly; that is why many people say that VS Code is half an IDE and an editor (Pedamkar, 2021). As stated by Thakur (2018), it is a super fast and lightweight code editor developed by Microsoft for Windows, Linux, and macOS. Moreover, it comes with built-in support such as Javascript, Typescript, Node.js, and a rich ecosystem of extensions that help users utilize their languages (Visual Studio Code, n.d.). In addition, it also allows users to change the design theme, keyboard shortcuts, preference and install additional extensions to upgrade overall project functionality (Uzayr, 2021).

VS Code is one of the most popular and best IDE that lets users collaborate with their teams and live cooperative development (McKinnon, 2020). According to Softermii (2021), over 50% of respondents in StackOverflow Developer Survey 2019 composed of Web, Mobile, and SRE/DevOps developers chose Visual Studio Code as their best IDE in C/C++. Likewise, Visual Studio Code is the second most popular IDE among students, Data Engineers, and Software engineers (Hayes, 2020). That is because of its unique features: support for multiple programming languages, IntelliSense, Cross-platform, Repository, Web Support, Terminal Support, Git Support, and other Support and Extensions (Pedamkar, 2021).

Therefore, the current system will utilize Visual Studio Code as the open-source editor for developing the mobile app as VS Code supports different programming languages, primarily JavaScript, the programming language for developing the mobile app. It also offers many extensions such as Arduino, JavaScript IntelliSense, debugging, formatting, code navigation, and many other advanced language features, easing the problem in the compatibility of software and languages used in the current system.

***JavaScript***

JavaScript, also known as JS, is a dynamic programming language often used in Web development, Web applications, Game development, Mobile applications, and others. It allows the programmer to implement dynamic features on web pages that cannot be done only with HTML and CSS (Megida, 2021). It helps create web pages alive in which it adds interactivity to websites making web pages more interactive (Hamza, 2020). Also, according to Mckenzie (2018), JS adds logic and interactivity to static pages. In which it became one of the competing languages and technology to become the standard for browser-based programming.

Since Mobile app development started to direct the world, JavaScript allows several ways for developing a mobile application even though it is not known as a primary language for app development (Philip, 2019). JavaScript framework, a collection of JavaScript code libraries, provides developers with pre-written code for routine programming tasks. It is a structure with a particular context that helps create applications within that context (Anderson, 2021). Frameworks are reusable abstractions of usable code through a well-defined Application Programming Interface or API. Many JS frameworks have different purposes, such as creating user interfaces, building mobile apps, and many more (Kidder, 2020). In addition, JS Frameworks are also a tool that makes working easier and smoother, making it more manageable for the programmer to code the applications more responsive (Arora, 2021).

Moreover, Javascript offers mobile app developers the benefits of creating an interactive and dynamic UI/UX expected of the clients. Building a mobile app with JavaScript is also faster and more responsive than other languages (Philip, 2019). According to Dionne (2021), JS is an insanely popular web development programming language that brings faster user experiences, such as interface interactivity and good, responsive design, making learning easy. Furthermore, installing Node.js is necessary to use JavaScript code outside browsers (Clark, 2020). Node.js is a popular environment for developing server-side and networking applications. It is used to cater to the backend Application Programming Interface services through the development environment (Bhargava, 2020), allowing the creation of real-time applications with fast synchronization, which many cloud platforms support Node.js because of its real potential in WebSockets (Nagar, 2019).

Therefore, the current system will be using JavaScript programming language and Node.js in developing the mobile app that will serve as the controller of the Arduino device. The reason is that JavaScript offers many libraries and frameworks, both mobile and Arduino platforms, and Node.js as its backend for connecting the Arduino device and mobile app.

***React Native***

React Native is a JavaScript framework based on React, Facebook's JavaScript library for building user interfaces in browsers but instead, React Native targets mobile platforms (O'Reilly Online Learning, n.d). It is a framework for creating cross-platform mobile applications being one of the most popular and widely recognized or used platforms for mobile application development (K, 2020). According to Thinkwik (2018), React Native helps create a natural and exciting mobile application using JavaScript only, supporting Android and iOS. It is a framework that builds a hierarchy of UI components to make JavaScript codes.

Moreover, React Native features JavaScript as its programming language, which is an essential part that influences developers. It is also UI-focused on having extraordinary responsive and rendering abilities. In addition, React Native has been one of the most trusted frameworks when it comes to cross-platform mobile application development since Facebook built it. It also shows supports for third-party libraries giving freedom of choice to developers (K, 2020). plus, it is primarily known for its ability to generate applications that can be both Android and iOS operating systems or, more referred to as cross-platform.

Furthermore, React Native is easier to learn since its programming language is the most used in the tech community (SpeedySense Editorial, 2021). Also, it uses CLI or Command Line Interface, which is also one of the tools needed in developing React Native applications. Such as the React Native CLI and the Expo CLI. The React Native CLI is the default CLI of React, while Expo is a set of tools built around React Native (Setting up the Development Environment, 2021). Expo CLI is considered an improved version of the standard because the project can run directly without any external dependency but only the development environment (Singh, 2020). Also, it is a free third-party service with many convenient utilities that simplify the development of applications (Weerasinghe, 2020). Expo also gives the convenience of having a dedicated mobile application, which can be used to preview the built application without using Android Studio or Xcode emulators (Piyadigama, 2020). Therefore, the current system will be utilizing React Native as its framework since it will be using JavaScript as its primary programming language for the mobile app. Additionally, it features cross-platform and supports third parties in making the Arduino device’s connection to the mobile app possible. Plus, using Expo CLI will help the developers to view the current system while working on it.

***Google Firebase***

Google Firebase is a popular Backend-as-a-Service (BaaS) software development platform for mobile and web-based applications. It offers services that focus on building, improving, and managing apps and businesses (Kumar, 2018). As stated by Cucullo (2017), the Firebase platform is a server, hosting platform, database, and authentication tool in a single solution. Since it launched, developing cross-platform apps has become much more accessible for developers as it enables them to store and synchronize data among multiple users. On top of that, Nguyen (2018) claims that Firebase is an excellent option in developing IoT applications.

In building applications, the available Firebase services are the Realtime Database, Cloud Firestore, Cloud Storage and Functions, Authentication, Hosting, and Machine Learning Kit (Editor, 2020). Apart from that, Firebase Analytics, Notifications, Cloud Messaging, and Test Lab for Androids are also supported to use by developers (Khawas & Shah, 2018).

Firebase Realtime Database was the first feature launched by Google Firebase, making it the most stable service on the platform (Puffelen, 2016). It provides a free and paid plan, labeled as Spark and Blaze, respectively. The free plan is limited to only one database, 1 GB of storage, 10 GB of download, and 100 simultaneous connections. In comparison, the paid plan allows multiple databases, $5 for every GB of storage, $1 for every GB of download, and 200k simultaneous connections for each database (Bohbot, 2021).

According to Shaikh (2021), the Firebase Realtime Database is a JSON formatted NoSQL database hosted on the cloud that enables storing data in single-rooted JSON format. Most databases require HTTP calls to be able to get and sync the data; however, with Firebase Realtime Database, the application is not using the standard HTTP connection but through a WebSocket, which allows event-based synchronization of data throughout the session, and is much faster to use (Ashwini, 2017). That is why it is most beneficial for data-intensive IoT systems (Sullivan & Sullivan, 2016). As specified by Rashmin (2018), it can easily be used as an intermediate communication medium between IoT devices and applications with the help of APIs. For instance, the study conducted by Al-Kababji et al. (2019) used Firebase Realtime Database to establish a communication between IoT-Based Fall and ECG Monitoring device and Vitals Monitoring mobile application because it has lower latency which enhances real-time data visualization implementation.

From all the services offered by Google Firebase, the current system will only utilize the Firebase Realtime Database on the free plan. The Firebase Realtime Database will enable real-time communication between the Arduino device and the mobile app. It will serve as a storage of the data gathered from the sensors of the Arduino device for the mobile app to retrieve. Simultaneously, this is also where the mobile app command or request will be sent for the Arduino device to do the action.

***Establishing Connection Between Arduino Device and Android Application***

Arduino microcontrollers are top-rated electronic devices that can be managed by an Android phone thru an application (Long, 2020). Similarly, Sandhu (2020) stated that Android devices are excellent candidates for Arduino integration due to their relatively open platform. In fact, there are various ways to establish communication between Arduino and Android, developing an IoT.

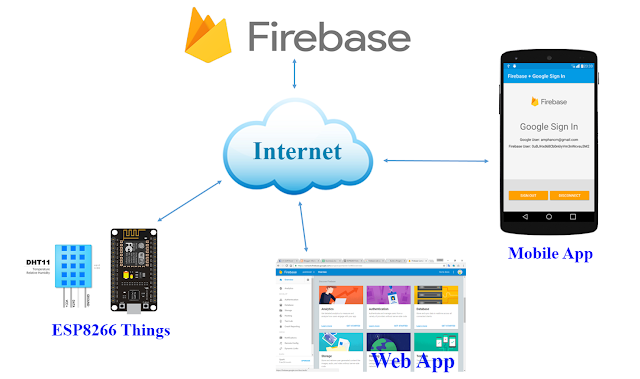
Patni (2020) attempted to list some of the ways to connect Arduino to Android:

* **HTTP requests.** HTTP POST and GET requests can be used to set up a communication between Arduino and Android with the ESP8266 module. It is best for applications where data does not always need to be retrieved because a new connection must be made every time an event occurs.
* **Bluetooth.** Bluetooth connection can be set with the use of a Bluetooth module for Arduino. It is best for short-range wireless communication; however, it is claimed not to function sometimes, even in the short-range.
* **Online services.** Online services such as Blynk have an Android mobile app and a library for Arduino to allow communication. It usually requires less coding and effort and enables device control wherever in the world, but it is often not free and has restrictions that might not suit the system requirements.
* **WebSocket.** WebSocket can also use the ESP8266 module to set the connection. It allows open connection unless one disconnects, unlike HTTP. It is best for real-time applications where a continuous data flow is necessary, but it is considered computationally expensive than HTTP.

With the stated various ways of connecting Arduino to Android, the current system will want to make use of WebSocket to establish the connection between the Arduino device and the mobile app. That is because the current system requires real-time data processing to function well with the mobile app and form data visualizations.

However, sometimes, WebSocket can be difficult to establish (Brown, 2020). Tucker (2017) also claimed that WebSocket is annoying and complex to scale, but with the use of Firebase Realtime Database, WebSocket was simulated easier. Based on Ali et al. (2021) study, the Firebase Real-time Database service uses a WebSocket communication protocol compared to a standard HTTP communication protocol. That is why Firebase was able to provide high-speed communication between the server and the client.

In order to make use of the Firebase Realtime Database in establishing a connection, there has to be a Wi-Fi module, such as the ESP8266 (Levy, 2019). The ESP8266 is a low-cost Wi-Fi module used to control devices on the Internet. It can work with a micro-controller like Arduino, but some can be programmed to work independently. There is only one ESP8266 processor that can be found on various boards but may differ on which pins are exposed and the size of the flash memory (TonesB, 2017). For instance, the study conducted by Varshney et al. (2019) used ESP8266 NodeMCU as a Wi-Fi module to send and store the real-time information of patient health from the IoT-based eHealth Management System to the Firebase Realtime Database. However, the current system will utilize the ESP8266-01 as it can work with an Arduino UNO and can also be programmed using Arduino IDE. Moreover, the module can function both as an Access Point to create a hotspot and as a station to connect to Wi-Fi. Therefore, it can easily fetch data and upload it to the Internet, making the Internet of Things easier (Components101, 2018).



***Figure 9.***Diagram of the Connection between ESP8266, Firebase, Mobile App and Web App over the Internet

The connection between an Arduino device's ESP8266 to the Firebase Realtime Database and a mobile app is shown in Figure 9 (Microcontrollerkits, 2016). In preparation for the connection or storing and retrieving data on the Firebase Realtime Database, some boards and libraries must be installed first in the Arduino IDE, such as the ESP8266 board, Firebase library, and ArduinoJson (Gharssellaoui, 2018). While for the development of the Android application, it is also necessary to install Firebase on the project to be able to use Firebase Realtime Database on the application (Rupareliya, 2019).

Establishing a connection between the Arduino device and the mobile app is necessary for the current system as the connection will allow the Arduino device to be managed by a user through the mobile app. At the same time, receive alerts and notifications and view data visualizations on the mobile app regarding the food condition. The current system will be utilizing Firebase Realtime Database to have a real-time or WebSocket connection that will allow event-based synchronization and continuous data flow with a minor delay that is effective to use on data visualizations.

***Data Visualization***

Data Visualization refers to drawing graphic displays to present data (Unwin, 2020). It enables users to identify patterns quickly in data and formulate insights that would otherwise go unrecognized. Similarly, Malysheva (2021) claimed that data visuals attract user attention, and at the same time, make complex data comprehensible since most people are visual learners.

On top of all of that, data visualization makes application management and control less complicated. Whether it is stock price performance or the sensors on an athlete, data visualization helps monitor operations, optimizes processes, and makes quick and correct judgments (Brych, 2021). However, choosing the correct data presentation type is vital to clearly send the message to users (MSIS 2629, 2019).

Dijk (2018) listed the basic presentation types that can be used depending on their purpose are as follows:

* **Comparison charts** are for comparing one set of values with another. It tries to set variables apart from one another and display how those variables interact. It can be presented through Bar, Line, and Two-axis charts.
* **Composition charts** show how parts of data comprise the whole and are displayed together in one chart. It can be presented through Pie/Donut, Stacked Colum, and Waterfall charts.
* **Distribution charts** show the distribution of a set of values. It lays out a collection of information in order to see its correlation and interaction. It can be presented through Columns, Scatter plots, and Line charts.
* **Relationship charts** show the relationship between a series of variables. It is like comparison charts but focuses on showing a direct correlation between compared variables. It can be presented through Scatter plots, bubble, and Two-axis charts.
* **Trend charts** show the trend over time of variables to improve the understanding of the performance of a process. It can be presented through Line, Column, Scatter plot, and Area charts.

With all the stated presentation types, the current system will require a trend chart, as it will help display how data from sensors may change over time in monitoring a food condition on the mobile app. With the varieties of trend charts, the mobile app will be implementing line charts for the data from the sensors, because as stated by Gulbis (2016), lines are used best on continuous data sets.

In order to display line charts on an Android application, there are various libraries to choose from, particularly for React Native framework. Mittal (2021) attempted to list the following React Native chart libraries for the year 2021. Some of them can be used on streaming or real time data, such as react-native-svg, react-native-chart-kit, and react-native-responsive-linechart.

* **react-native-svg** library provides Scalable Vector Graphics (SVG) chart on cross-platform using the react-native-svg package and d3 library. It supports charts including area, line, bar, pie, progress circle, and more. Moreover, it provides a standard set of props for each component to customize the chart to represent data accurately, making it easier to use in real-world scenarios.
* **react-native-chart-kit** is built on top of popular projects like react-native-svg and paths-js. It supports line, pie, stacked bar, contribution graphs, and more. Furthermore, it allows the rendering of responsive charts and chart style customization.
* **react-native-responsive-linechart** is focused exclusively on representing data in the form of lines. It supports a huge number of data and depends on react-native-svg library and react-native-gesture-handler library. On top of that, it only has a 62KB package size.

With the said libraries for React Native chart, the current system will be using the react-native-svg library as it works for expo applications, and at the same time, it is said to be easier to use for real-life scenarios. Moreover, Swizec (2017) used react-native-svg library for real-time visualization of Bitcoin transactions, making it appropriate for the current system. The current system also requires real-time data visualizations for the user to understand the changes in the data from the gas sensors regarding the condition of the food.

***Software Quality Evaluation***

The ISO 25010 "Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models" is a software quality standard that describes models and characteristics (Britton, 2021). It is the enhanced version of the ISO 9126 that comprises two quality models focused on software engineering, the Quality in Use Model and the Product Quality Model (Hammar, 2017).

The Product Quality Model deals with software quality evaluation from different perspectives (Tambotoh et al., 2016). This model ensures that the software meets the software quality standards by evaluating or measuring the quality internally or the software itself, and externally or the quality of the software concerning hardware, data, and computer systems (Gustriansyah et al., 2021). In addition, it is the cornerstone of a product quality assessment system as it determines which quality characteristics are to be considered when evaluating the properties of a software product (ISO/IEC 25010, n.d.).

According to Hammar (2017), the Product Quality Model defines a set of eight quality characteristics composed of two to six sub-characteristics. The eight defined characteristics are functional stability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability, are expounded below:

**Functional Suitability** represents the degree to which a product or system provides functions that meet defined and implied needs when used under specified conditions. It is composed of the following sub-characteristics:

* **Functional completeness.** The degree to which the set of functions covers all the specified tasks and user objectives.
* **Functional correctness.** The degree to which a product or system provides the correct results with the required degree of precision.
* **Functional appropriateness.** The degree to which the functions facilitate the accomplishment of specified tasks and objectives.

**Performance Efficiency** represents the performance relative to the number of resources used under stated conditions. It is composed of the following sub-characteristics:

* **Time behavior.** The degree to which a product or system's response and processing times, and throughput rates meet requirements when performing its functions.
* **Resource utilization.** The degree to which the amounts and types of resources used by a product or system meet requirements when performing its functions.
* **Capacity.** The degree to which the maximum limits of a product or system parameter meet requirements.

**Compatibility** refers to the extent to which a product, system, or component can exchange information with other products, systems or components, and perform its required functions while sharing the same hardware or software environment. It is composed of the following sub-characteristics:

* **Co-existence.** The degree to which a product can perform its required functions efficiently while sharing a typical environment and resources with other products, without detrimental impact on any other product.
* **Interoperability.** The degree to which two or more systems, products, or components can exchange information and use the information exchanged.

**Usability** refers to the extent to which specified users can use a product or system to achieve specified goals with effectiveness, efficiency, and satisfaction in a particularized context of use. It is composed of the following sub-characteristics:

* **Appropriateness recognizability.** The degree to which users can recognize whether a product or system is appropriate for their needs.
* **Learnability.** The degree to which specific users can use a product or system to achieve specific goals with effectiveness, efficiency, and satisfaction is a particular user context.
* **Operability.** The degree to which a product or system has attributes that make it easy to operate and control.
* **User error protection.** The degree to which a system protects users against making mistakes.
* **User interface aesthetics.** The degree to which a user interface allows pleasing and satisfying interaction for the user.
* **Accessibility.** The degree to which a product or system can be used by people with the broadest range of characteristics and capabilities to achieve a specific goal in a defined user context.

**Reliability** refers to which a system, product, or component performs specific functions under specified conditions for a specified period of time. It is composed of the following sub-characteristics:

* **Maturity.** The degree to which a system, product, or component meets the reliability specification under operating conditions.
* **Availability.** The degree to which a system, product, or component is operational and accessible when required for use.
* **Fault tolerance.** The degree to which a system, product, or component operates as intended despite the presence of hardware or software faults.
* **Recoverability.** The degree to which in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the system's desired state.

**Security** refers to the extent to which a product or system protects information and data so that users or other products or systems have access to data depending on their types and levels of authorization. It is composed of the following sub-characteristics:

* **Confidentiality.** The degree to which a product or system ensures that data is accessible only to those authorized to have access.
* **Integrity.** The degree to which a system, product, or component prevents unauthorized access to computer programs or data modification.
* **Non-repudiation.** The degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later.
* **Accountability.** The degree to which the actions of an entity can be detected uniquely to the entity.
* **Authenticity.** The degree to which the identity of a subject or resource can be proved to be the one claimed.

**Maintainability** represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it, or adapt it to changes in the environment and requirements. It is composed of the following sub-characteristics:

* **Modularity.** The degree to which a system or computer program is comprised of discrete components so that the change in one component has minimal impact on other components.
* **Reusability.** The degree to which an asset can be used in more than one system or in building other assets.
* **Analyzability.** The degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its components, diagnose a product for deficiencies or causes of failures, or identify parts to be modified.
* **Modifiability.** The degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
* **Testability.** The degree of effectiveness and efficiency with which test criteria can be established for a system, product, or component and tests can be performed to determine whether those criteria have been met.

**Portability** refers to the degree of effectiveness and efficiency with which a system, product, or component can be transferred from one hardware, software, or other operational or usage environment to another. It is composed of the following sub-characteristics:

* **Adaptability.** The degree to which a product or system can be adapted effectively and efficiently for different or evolving hardware, software, or other operational or usage environments.
* **Installability.** The degree of effectiveness and efficiency with which a product or system can be successfully installed and uninstalled in a specified environment.
* **Replaceability.** The degree to which a product can replace another specified software product for the same purpose in a similar environment.

*Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010*

The acceptability of each component in the current system will be evaluated based on all the listed ISO 25010 criteria, such as functionality suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability because all of the said characteristics are appropriate for the current system.

**Synthesis of the Review of Related Literature**

Based on the information gathered by the researchers, food spoilage is a metabolic process causing foods to be unacceptable for human consumption due to sensory, physical, microbiological, and chemical changes. Most foods usually give signs when being spoiled. However, some get spoiled without any noticeable changes, which leads the researcher to develop a system that automatically detects food spoilage. This approach is through the use of low-cost Arduino sensors that could detect gas concentration from decaying foods. These gases are produced during the spoilage process. For meats, there is a production of gases such as ammonia and ethanethiol. For fruits such as tomato, avocado, apple, melon, peach, kiwi, and banana produce ethanol and ethylene gases as they ripen and rise even more as they spoilt. Hence, detecting gases like ammonia, methane, and ethanol, which are emitted naturally in decaying food, can be used to detect food spoilage.

The detection of food spoilage will be developed using Arduino, an open-source electronics platform that can be programmed using Arduino IDE to take input from the data coming from the sensors and outputs that can be used to control other modules from the Arduino. Arduino also receives and sends information over the internet, allowing the device to be integrated with IoTs. As the Arduino board for the Arduino device, Arduino Uno is flexible in terms of the tiny components embedded in itself, such as an ATmega328p microcontroller and serial communication voltage regulator, and more. Other hardware components essential to the Arduino device are the gas sensors MQ135 sensor to detect Ammonia, MQ2 sensor for the detection of methane, and MQ3 sensor for the detection of alcohol or ethanol. This sensor provides analog output for testing the gas concentration, which can be converted into output voltage to measure the presence of the gas in ppm. These sensors need to be preheated once over 24 hours and calibrated each time the Arduino device is turned on to measure gas concentrations accurately.

In line with the information gathered. The Arduino programming for the system component Arduino device will follow the structure of sketches, starting with the initialization of pins and variables. Next is the setup function for establishing the serial communications to allow the communications between Arduino Uno board, ESP8266 Wi-Fi Module, and sensors. The loop function is where the data from sensors are fetched and stored in the initialized variables at the start of the sketch or setup function. Last is the ESP8266 function to transmit data gathered from sensors to the cloud and deliver those data to the mobile applications, establishing communication between the mobile app and Arduino device.

The mobile app integrated to support the Arduino device that could detect food spoilage will be developed exclusively for the Android platform. It will be a native type of application for it to be able to utilize the device features with the help of JavaScript and React Native framework. In order to build the React Native application, there are a few requirements needed, including the Visual Studio Code as the development environment, Node.js, the React Native itself, Expo CLI, and knowledge in JavaScript. With the VS Code features that allow the addition of libraries, extensions, and packages, it will be easy to set up the database development for Firebase Realtime Database. It also allows real-time synchronization and works excellent with Android APIs and Arduino kits, such as the sensors.

The connection between the Arduino device and the mobile app will be via Wi-Fi through WebSockets to allow long-distance communication and continuous flow of real-time data. To be able to provide the said connection, a NodeMCU ESP8266 Wi-Fi module must be connected to the Arduino device first, as it will enable the device to store collected data on the Firebase database so that the integrated mobile app can retrieve it. Apart from that, some libraries and modules must be installed on the Arduino IDE and the project. On the Arduino IDE, there must be ESP826 Board, Firebase library, and ArduinoJson. Then Firebase on the mobile app project for it to be able to do CRUD operations on the data from the Arduino device that could detect food spoilage.

**Related Studies**

Various works and researches have already been conducted about detecting food spoilage utilizing different methods such as Radio-Frequency Identification (RFID) Tag, Electronic Nose (E-Nose), and low-cost gas sensors with Arduino. A paper worked by Voorugonda (2021) entitled Food Spoilage Detection focuses on detecting if the food is fresh or spoiled using an MQ3 sensor, Arduino Uno board, 16x2 LCD, and power supply. The author's work similarly focuses on detecting spoilage, especially in fruits. However, unlike the current study, the author did not include other parameters like temperature and humidity to monitor or detect spoilage since these can affect food conditions. Moreover, this work concentrated only on detecting the gas concentration around the food without specifying which specific gas is present, causing the food spoilage. The author calibrated the sensor using its built-in potentiometer to set the threshold value for determining food conditions such as fresh or spoiled. This method might be the easiest way to detect spoilage, but accuracy might also be compromised. For this reason, the researchers chose to perform calibration and calculate a specific gas concentration on each Arduino gas sensor.

Another related study is the work done by Sharada et al. (2019), entitled Smart stale food detector using IoT. In this work, aside from using similar components such as Arduino Uno as a microcontroller, DHT-11 sensor for temperature, and humidity, MQ3 sensor for ethanol gas, NodeMCU ESP8266 Wi-Fi Modem is also included to provide an open-source IoT platform. Similar to the current system, this work used the ESP8266 Wi-Fi Modem to transfer the information gathered by the Arduino interfaced sensors to an IoT platform. The difference is that the researchers used Firebase Realtime Database to have more flexibility with data and used it on a dedicated mobile app while the authors utilized ThingSpeak Server. The information transferred on the ThingSpeak server is visualized and shown online through the use of an advanced dashboard called I.B.M. Watson, which requires a registered account for sending alert messages on the registered mobile phone. On the other hand, through the dedicated mobile app of the current system, the user could visualize data, receive alert messages, and also send commands to the Arduino device.

The two mentioned studies above support the concept that the MQ3 gas sensor is capable and a key component in detecting the presence of ethanol types of gas, one of the indicators of food spoilage. However, MQ3 is limited only in detecting the presence of alcohol and ethanol gases (Chaware, 2020). Wijaya et al. (2018), the authors of the work entitled Electronic nose dataset for beef quality monitoring in uncontrolled ambient conditions, claims that utilizing more than one gas sensor can effectively detect food spoilage besides using a temperature and humidity sensor. The authors utilized sensors like MQ135 for carbon dioxide and ammonia, MQ136 for hydrogen sulfide, MQ2 for i-butane, propane, methane, alcohol, and hydrogen, MQ3 for methane and carbon dioxide, and others such as MQ4, MQ5, MQ6, MQ7, MQ8, MQ9 for the development of mobile electronic nose for beef quality monitoring. Moreover, the Arduino Mega SDK microcontroller is used as the mainboard to support all the components. Although using multiple sensors is a great option, this could cost the optimization of the system. Thus, the current system opts to limit sensors with DHT11, MQ2, MQ3, and MQ135 sensors as per the scope of the study. However, this limitation ensures that all known gasses produced by decaying food are covered. Moreover, the authors focused only on monitoring beef, while the current study focuses also includes other foods such as fruits and milk giving more variety of food selection for inspection and monitoring.

**Conceptual Model of the Study**

The conceptual model of the study is depicted in figure 10 using the Input-Process-Output diagram. This diagram gives an overview of the flow and whole concept of the study.

**OUTPUT**

**PROCESS**

**INPUT**

**Knowledge Requirement**

* Food Spoilage
* Detecting Food Spoilage
* Gas Formation on Spoiled Food
* Arduino, Peripheral Modules, and Gas Sensors
* Knowledge in Arduino Programming
* Sensor Calibration
* Visual Studio Code
* JavaScript
* React Native
* Google Firebase
* Development of Android Application
* Connecting Arduino Device and Android Application
* Data Visualization
* Software Quality Evaluation

**Software Requirement**

* Visual Studio Code
* Arduino IDE
* React Native Framework
* NodeJS
* Expo CLI
* Firebase Realtime Database
* Windows 10
* Web Browser
* Android OS

**Hardware Requirement**

* Computer or Laptop
* Android Mobile Phone
* Arduino Uno
* Arduino Gas Sensors (MQ135, MQ2, MQ3)
* DHT11 Sensor
* ESP8266 Wi-Fi Module
* 16x2 LCD Screen
* Push Button
* RGB LED
* Piezoelectric Buzzer
* Battery
* Resistors & Wires

Analyze

Design

Develop & Debug

Test

**EVALUATION**

**Arduino Device and Mobile App**

**For Smart Food Spoilage Detection System**

Using ISO 25010 Software Quality Model

***Figure 10.***Conceptual Model of the Study

**Input**

The input block contains the knowledge, software, and hardware requirements needed to develop the Arduino device and mobile app for smart food spoilage detection. The knowledge requirements are the concepts and information the researchers must understand and have a good grasp of to establish the foundation of the study. These requirements include the understanding of concepts regarding food spoilage; different methods to detect food spoilage; gas formation of spoiled food; the Arduino, peripheral modules, and gas sensors; knowledge in Arduino programming; developing Android application and the tools to be used; establishing the connection or communication between Arduino device and Android application; data visualization; and software quality evaluation or ISO 25010 software quality metrics for assessing the acceptability of mobile app. The software requirements are the application programs, programming languages, frameworks, databases, and operating systems needed to develop the smart food spoilage detection system. The hardware requirements are the tools needed in creating the Arduino device, such as Arduino Uno, Arduino Gas Sensors like MQ135, MQ2, and MQ3, DHT11 Sensor, ESP8266 Wi-Fi Module, 16×2 LCD Screen, Push Button, Battery, and Connecting Wires; and computer machine for developing the current system.

**Process**

The process block includes analyzing, designing, constructing/debugging, and testing activities to develop the Arduino device and the mobile app for smart food spoilage detection.

***Analyze***. During this phase, the researchers should analyze the background of the problem. There must be a series of researches, investigations, and data collections on how the current process of detecting food spoilage can be further improved or enhanced. At the end of this phase, the set of requirements should be identified and presented.

***Design.*** In this phase, the researchers should create the block diagram and the implementation scheme of the Arduino device as a detector device for smart food spoilage detection. The block diagram serves as the model of the system that shows the connection of each component and flow of interaction between the Arduino device and mobile app, describing the data transmission from the Arduino device to the mobile app and vice versa. On the other hand, the implementation scheme depicts how each hardware component pins and wirings are connected and assembled. For the mobile application, the researchers would prepare a use case diagram that specifies the expected functionalities of the system where the external users can interact. An entity-relationship diagram would be utilized to represent the database design and the relationship of each identified entity on the system.

***Develop & Debug.*** During this phase is the construction or assembling of Arduino hardware components. Also, this phase includes the actual coding or programming for the Arduino device using Arduino IDE and mobile apps using Visual Studio Code. The researchers use React Native framework in developing the mobile app, along with JavaScript and Node.js. Aside from actual coding is establishing the connection between Arduino device and mobile app with Firebase Realtime Database. This phase also includes testing or debugging, where the researchers ensure that the system works appropriately, addressing all errors or bugs. The output in this phase is the assembled and programmed Arduino device and developed mobile app.

***Test.*** After the development, the assembled and programmed Arduino device and developed mobile app should undergo tests to identify the system's accuracy in monitoring or detecting food spoilage under three food conditions of each food specified in the scope. A proper adjustment must be performed for every failed test case to ensure the functionality and accuracy of the system before subjecting them to evaluation.

**Output**

The output block shows the developed Arduino device and mobile app for smart food spoilage detection. The output, specifically mobile app, should be subjected to evaluation to determine its acceptability.

**Operational Definition of Terms**

     The following terminologies are defined for better understanding the study:

**Arduino Device**refers to the part of the output of the study which would be programmed and composed of Arduino sensors and peripherals used for monitoring the food.

**Arduino Gas Sensors** refer to the gas sensors, specifically MQ135, MQ2, and MQ3 sensors, that would be used in this study for checking the presence of ammonia, methane, and ethylene gas concentrations, respectively.

**Current System** refers to the food spoilage detection system of the researchers in this study.

**Mobile App** refers to the other part of the output of this study which would be developed for the Android platform that complements the Arduino Device by allowing the users to control and give commands to the Arduino device and alert them for any significant changes in the quality of food.

**Peripheral Modules** refers to the Arduino sensor modules that would be used in this study, specifically the DHT11 Sensor, ESP8266 Wi-Fi Module, RGB LED, LCD Screen, Push Button, and Buzzer.

**System Components** refer to both the Arduino device and mobile app that would be developed as the outputs of this study.

**Chapter 3**

**METHODOLOGY**

This chapter entails the research methodology of the study with the following sections: project design, project development, operation and testing procedure, and evaluation procedure.

**Project Design**

The study would develop an Arduino device integrated with biosensors to prevent and detect food spoilage. This device is designed to interact with the mobile app that allows users to give commands on the Arduino device, monitor the food, and receive alerts. The study would use several modeling tools to analyze the system's scope and requirements as a map or guide for project development.

***System Design***

The model of the system is represented through the use of a Block Diagram, as depicted in Figure 11. The diagram shows the input and output components needed to build the Arduino device and the flow of interaction between the Arduino device and mobile app.

The Arduino device consists of an Arduino Uno as a microcontroller, battery for power supply, pushbutton as command interface, and biosensors like DHT11 sensor, MQ135 gas sensor, MQ2 gas sensor, and MQ3 gas sensor that measure different parameters such as temperature and humidity, and ammonia, methane, and ethanol level. The device is also composed of output components such as an LCD screen for displaying the output, RGB LED and buzzer for signifying alert signals, and a Wi-Fi module as a gateway to establish communication from and to the mobile app through the internet. The data accumulated by Arduino Uno from the sensors are transmitted to the Firebase Realtime Database, wherein data are stored and synced automatically among the mobile app users in real-time. Likewise, sending commands to the Arduino device using the mobile app is retrieved through Firebase Realtime Database by the Wi-Fi module of the Arduino device.

Diagram

Description automatically generated

***Figure 11.***Block Diagram of Smart Food Spoilage Detection System

**Software Design**

The scope of the system’s software is represented through Use Case Diagram, as depicted in Figure 12. The diagram shows the external entities such as mobile application users, Arduino device users, and firebase database that all interact with the system. This diagram specifies the expected behavior, functionalities, and requirements of the software.

Diagram

Description automatically generated

***Figure 12.*** Use Case Diagram of Smart Food Spoilage Detection System

Based on system software requirements, the following are the features of the Smart Food Spoilage Detection System designed for mobile application:

* Register Arduino Device ID number
* Select specific food to inspect
* Inspect food to determine the condition level of food
* Monitor food for signs of spoilage
* Send notifications about significant changes in food conditions and Arduino device battery life
* View food storage and shelf-life tips
* View significant changes in food condition via Data Visualization

On the other hand, the following are the features of the Smart Food Spoilage Detection System designed for Arduino device:

* Inspect food to determine the condition level of food
* Display food conditions on LCD Screen
* Turn-on LED with specific color depending on the food condition
* Alarm the buzzer for warning and danger condition level to indicate the food is perishing soon or spoiled

**Database Design**

The scope of information maintained in the system is illustrated using an Entity-Relationship Diagram (ERD), as presented in Figure 13. The ERD shows five (5) primary entities consisting of the Arduino device, mobile app, sensor, inspected food, and monitored food. Every Arduino device has a unique identification number that can be registered on multiple mobile apps. It also has a device state that can tell if the device is on or off, the current mode, and its identification number to know if the device is monitoring or inspecting. The mobile app can process many food inspections and monitoring. For every food inspection, there must be one sensor data, while in food monitoring, there must be one or many sensor data since it is a continuous process of food inspection.

Diagram

Description automatically generated

***Figure 13.*** Database Design of Smart Food Spoilage Detection System

**Project Development**

This section discusses the procedures followed on how the smart food spoilage detection system is developed based on the design specifications.

***Schematic Installation of Arduino Device***

The schematic of the Arduino device is depicted below. This figure represents the circuit design and wiring scheme for the installation of all circuit components. Each component should be connected to the Arduino microcontroller with proper pining. The list of components used in this circuit is shown in Table 4.

Diagram, schematic

Description automatically generated

***Figure 14.***Schematic Installation of Arduino Device

Table 4.

*List of Components for Arduino Portable Device.*

|  |  |
| --- | --- |
| Components | Quantity |
| Arduino Uno | 1 |
| 4xAA Battery Holder | 1 |
| AA Battery | 4 |
| Mini Pushbutton Switch | 1 |
| DHT11 Humidity and Temperature Sensor | 1 |
| MQ135 Ammonia Gas Sensor | 1 |
| MQ2 Methane Gas Sensor | 1 |
| MQ3 Alcohol Gas Sensor | 1 |
| ESP8266 Wi-Fi Module | 1 |
| 16x2 LCD Screen | 1 |
| Piezo Buzzer | 1 |
| RGB LED Common Anode | 1 |
| Bi-Directional Logic Level Converter | 1 |
| Breadboard | 2 |
| 10K Ohm Resistor | 1 |
| M/M Jumper Wires Pack | 2 |
| M/F Jumper Wires Pack | 1 |

***Database Setup***

Based on the database design of the system in figure 13, the equivalent NoSQL Document Model for storing and retrieving data in Firebase Realtime Database is shown in figure 15.



***Figure 15.*** NoSQL Document Model of Smart Food Spoilage Detection System

***Program Coding***

The algorithm for the Arduino device includes:

1. Preheat MQ-2, MQ-3, and MQ-135 sensor by leaving it connected to the circuit for 24 hours.
2. Calibrate the gas sensors by uploading specific calibration codes on the Arduino and exposing them to clean air.
3. Connect each component to Arduino microcontroller with the appropriate pinning as shown in Figure 13.
4. Import all the necessary libraries in the Arduino IDE, which includes ESP8266 Board, Firebase library, and Arduino Json.
5. Initialized variables for input and output components.
6. Once the Arduino device is on, the user can then push the pushbutton to call preheat() function and enable the sensors to preheat for 30 seconds and then call calibrate() function to begin the sensor calibration for 2 minutes until the buzzer alarms.
7. If food inspection is initiated by pushing the pushbutton again, call getData() function to enable the Arduino sensors to collect data from the food.
8. Analyze the data collected based on the set threshold value on the Arduino sensors, using equivalentVal() function to know the food condition.
9. Display the analyzed food condition on the LCD screen in the Arduino device and set the color of the RGB L.E.D to alert the user using arduinoDisplay() function.
10. Send the analyzed condition to the Firebase Realtime database, using the sendData() function.
11. If the Arduino device is registered on the mobile application, whenever there is a new data on the Firebase collected food condition, it will send an alert message on the user’s mobile app regarding the food condition, using appDisplay() function.

The mobile application will be created using JavaScript with React Native framework. The following steps will be used to develop the mobile app:

1. Add and initialize Firebase JavaScript SDK on the mobile application project.
2. Design and code the interface and functions for the mobile application.
3. If a food is selected and the inspect button is pressed in the mobile app, it will send data on the Firebase Realtime database to initiate inspection on the Arduino device using foodInspect() function.
4. The foodInspect() function will call the getData() function in the Arduino.
5. The Arduino device will send the food condition thru alert message, using the appDisplay() function.
6. If a food is selected and the monitor button is pressed in the mobile app, it will send data on the Firebase Realtime Database to initiate monitoring on the Arduino device using foodMonitor() function.
7. The foodMonitor() function will call the getData() function in the Arduino.
8. Collection of data using the Arduino sensor will continue until the user terminates the monitoring process.
9. Initially, The Arduino device will send the food condition thru an alert message, using the appDisplay() function.
10. The succeeding alert message will be sent, if there are significant changes in the food’s condition.

**Operation and Testing Procedure**

The following procedure will be used to operate the built Arduino device:

1. Push the battery switch to turn on the Arduino device.
2. Push the pushbutton switch to begin the preheating and calibration of the sensors.
3. Wait for a buzzer sound to indicate that the preheating and calibration mode is over.
4. Push the pushbutton again to initiate food inspection.
5. View the food condition report displayed on the LCD screen and wait for the buzzer sound and the color of the LED emit.
6. Push the battery switch to turn off the Arduino device.

The following procedure will be used to operate the built Arduino device supported with the developed mobile application.

1. Connect the smartphone to the ESP8266 soft access point of the Arduino device.
2. Input the given default IP address in the address bar of the browser in the smartphone.
3. Input the Wi-Fi network credentials on the web page to connect the Arduino device to the Wi-Fi network.
4. Disconnect the smartphone from the soft access point and connect the smartphone to a similar Wi-Fi network.
5. Install the Android application package (APK) file of the mobile app on the smartphone.
6. Register the identification number of the Arduino device in the mobile app.
7. Select a specific food from the available options in the mobile app.
8. Place the selected food no farther than 5 inches away from the Arduino device, and then press the inspect button in the app to begin the food inspection.
9. Wait for a buzzer sound to indicate that the preheating and calibration mode is over.
10. Attach the Arduino device to the designated food container.
11. Select a specific food from the available options in the mobile app and place the selected food inside the designated food container.
12. Press the monitor button in the mobile application to initiate food monitoring.
13. Leave the food inside the food container to monitor the significant changes in the food condition through the data visualizations in the app.
14. View the food-related tips and the battery life of the Arduino device.

The following tables show the testing procedures to be conducted on the Arduino device and the mobile application in terms of accuracy

Table 5.

*Accuracy Testing Procedure of the Arduino Device.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Food Status | Food Type | | Steps to be Taken | Expected Output |
| Fresh | * 1. Apple   2. Avocado   3. Banana   4. Kiwi   5. Melon   6. Peach   7. Tomato   8. Orange juice   9. Apple Juice   10. Milk   11. Kare-Kare   12. Bicol Express   13. Carbonara   14. Meat | | 1. Push the mini pushbutton switch to initiate inspection on food. 2. Place the fresh food no farther than 5 inches away from the Arduino device. 3. Observe the Arduino device for the output. | The LCD screen will display “Food is safe for consumption.” There will be no buzzer alarms, and the RGB LED will emit the color green. |
| Stale | * 1. Apple   2. Avocado   3. Banana   4. Kiwi   5. Melon   6. Peach   7. Tomato   8. Orange juice   9. Apple Juice   10. Milk   11. Kare-Kare   12. Bicol Express   13. Carbonara   14. Meat | | 1. Push the mini pushbutton switch to initiate inspection on food. 2. Place the stale food no farther than 5 inches away from the Arduino device. 3. Observe the Arduino device for the output. | The LCD screen will display “Warning! The food is perishing soon.” There will be a short buzzer alarm, and the RGB LED will emit the color orange. |
| Spoiled | * 1. Apple   2. Avocado   3. Banana   4. Kiwi   5. Melon   6. Peach   7. Tomato   8. Orange juice   9. Apple Juice   10. Milk   11. Kare-Kare   12. Bicol Express   13. Carbonara   14. Meat | 1. Push the mini pushbutton switch to initiate inspection on food. 2. Place the spoiled food no farther than 5 inches away from the Arduino device. 3. Observe the Arduino device for the output. | | The LCD screen will display “Danger! The food is now spoiled & must not be consumed.” There will be a buzzer alarm, and the RGB LED will emit the color red. |

Table 6.

*Accuracy Testing Procedure of the Arduino Device with the Mobile App.*

|  |  |  |  |
| --- | --- | --- | --- |
| Food Status | Food Type | Steps to be Taken | Expected Output |
| Fresh | * 1. Apple   2. Avocado   3. Banana   4. Kiwi   5. Melon   6. Peach   7. Tomato   8. Orange juice   9. Apple Juice   10. Milk   11. Kare-Kare   12. Bicol Express   13. Carbonara   14. Meat | 1. Select the food from the options in the mobile app. 2. Place the selected fresh food no farther than 5 inches away from the Arduino device. 3. Press the inspect button in the app to begin the food inspection. 4. Observe the Arduino device and mobile app for the output. | The LCD screen and the mobile app will display “Food is safe for consumption.” There will be no buzzer alarms, and the RGB LED will emit the color green. |
| Stale | * 1. Apple   2. Avocado   3. Banana   4. Kiwi   5. Melon   6. Peach   7. Tomato   8. Orange juice   9. Apple Juice   10. Milk   11. Kare-Kare   12. Bicol Express   13. Carbonara   14. Meat | 1. Select the food from the options in the mobile app. 2. Place the selected stale food no farther than 5 inches away from the Arduino device. 3. Press the inspect button in the app to begin the food inspection. 4. Observe the Arduino device and mobile app for the output. | The LCD screen and the mobile app will display “Warning! The food is perishing soon.” There will be a short buzzer alarm, and the RGB LED will emit the color orange. |
| Spoiled | * 1. Apple   2. Avocado   3. Banana   4. Kiwi   5. Melon   6. Peach   7. Tomato   8. Orange juice   9. Apple Juice   10. Milk   11. Kare-Kare   12. Bicol Express   13. Carbonara   14. Meat | 1. Select the food from the options in the mobile app. 2. Place the selected stale food no farther than 5 inches away from the Arduino device. 3. Press the inspect button in the app to begin the food inspection. 4. Observe the Arduino device and mobile app for the output. | The LCD screen and the mobile app will display “Danger! The food is now spoiled & must not be consumed.” There will be a buzzer alarm, and the RGB LED will emit the color red. |

**Evaluation Procedure**

The evaluation instrument that will be used to assess the acceptability of the system was adapted from the ISO 25010 titled “Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.”

The following procedure will be conducted to evaluate the acceptability of the developed mobile app together with the Arduino device:

1. Invite 40 purposively selected respondents composed of ten (10) information technology professionals, ten (10) food specialists, ten (10) fruit and vegetable vendors, and ten (10) parents of children ages fifteen (15) years old and below.
2. Demonstrate and explain how to operate the mobile app together with the Arduino device.
3. The evaluators will be asked to try using the mobile app and the Arduino device on food that emits methane, ammonia, or ethylene gas, i.e., meat, apple, banana, tomato, milk, etc.
4. The evaluators will be requested to evaluate the mobile app and the Arduino device individually based on the given evaluation sheets using a 4-point Likert Scale shown in Table 7.
5. The accomplished evaluation sheets will be processed, and the data gathered will be tabulated to determine the mean ratings.
6. The adjectival ratings for the mean ratings will also be interpreted using the Likert Scale shown in Table 7.

Table 7.

*Likert’s Scale.*

|  |  |  |
| --- | --- | --- |
| Scale | Adjectival Rating | Range |
| 4 | Highly Acceptable | 3.4 – 4.0 |
| 3 | Very Acceptable | 2.6 – 3.3 |
| 2 | Acceptable | 1.8 – 2.5 |
| 1 | Not Acceptable | 1.0 – 1.7 |

**REFERENCES**

25000, I. S. O. (n.d.). AENOR present their Trust Platform: “Ensuring Software and Data Quality.” ISO 25000. Retrieved November 18, 2021, from https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Abdel-Aziz, S. M., Asker, M. M., Keera, A. A., & Mahmoud, M. G. (2016). Microbial food spoilage: Control strategies for shelf life extension. In N. Garg, S. M. Abdel-Aziz, & A. Aeron (Eds.), *Microbes in Food and Health* (1st ed., pp. 239–264). Spring. https://doi.org/10.1007/978-3-319-25277-3\_13

Agilo Research. (2019, July 20). *Interfacing MQ-2 gas sensor with evive*. STEMpedia. Retrieved December 29, 2021, from https://thestempedia.com/tutorials/interfacing-mq-2-gas-sensor-with-evive/#PictoBlox\_Scratch\_Program

Al-Kababji, A., Shidqi, L., Boukhennoufa, I., Amira, A., Bensaali, F., Gastli, M. S., Jarouf, A., Aboueata, W., & Abdalla, A. (2019). IoT-based fall and ECG monitoring system: Wireless communication system-based firebase realtime database. 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 1481–1483. https://doi.org/10.1109/smartworld-uic-atc-scalcom-iop-sci.2019.00267

Alexaki, L., & Dalmaris, P. (2020, April 25). Types of hardware that you can connect to an Arduino board. Arduino Project Hub. Retrieved January 3, 2022, from https://create.arduino.cc/projecthub/lina-tech-explorations/types-of-hardware-that-you-can-connect-to-an-arduino-board-50555c

Ali, H., Naing, H. H., & Yaqub, R. (2021). An IoT assisted real-time high CMRR wireless ambulatory ECG monitoring system with Arrhythmia detection. *Electronics*, *10*(16), 1871. https://doi.org/10.3390/electronics10161871

Amit, S. K., Uddin, M. M., Rahman, R., Islam, S. M. R., & Khan, M. S. (2017). A review on mechanisms and commercial aspects of food preservation and processing. *Agriculture & Food Security*, *6*(1). https://doi.org/10.1186/s40066-017-0130-8

Anderson, J. (2021, October 5). What is JavaScript framework | General assembly. General Assembly Blog. Retrieved October 24, 2021, from <https://generalassemb.ly/blog/what-is-a-javascript-framework/>

Arduino2go. (2016, August 20). *Chapter 1: Parts of an Arduino*. Arduino to Go. Retrieved December 29, 2021, from https://arduinotogo.com/2016/08/20/chapter-1-parts-of-an-arduino/

Arora, S. K. (2021, September 9). 10 best JavaScript frameworks to use in 2021. Hackr.Io. Retrieved October 24, 2021, from https://hackr.io/blog/best-javascript-frameworks

Ashwini, A. (2017, March 20). *Using Firebase to provide real-time notifications - the startup*. Medium. Retrieved November 18, 2021, from https://medium.com/swlh/using-firebase-to-provide-real-time-notifications-2f86b2f7d499

Badami, V. (2016, October 21). A tour of the Arduino UNO board. HackerEarth Blog. Retrieved October 24, 2021, from https://www.hackerearth.com/blog/developers/a-tour-of-the-arduino-uno-board/

Balasubramanian, K., Ambikapathy, & Panneerselvam, A. (2017). Studies on ethanol production from spoiled fruits by batch fermentations. *Journal of Microbiology and Biotechnology Research*, *1*, 158–163. https://www.semanticscholar.org/paper/Studies-on-ethanol-production-from-spoiled-fruits-Balasubramanian-Ambikapathy./34e563e9d3f75c9e1c8449be99abc046a0651f05

Benabdellah, N., Bourhaleb, M., Benazzi, N., Nasri, M., & Dahbi, S. (2017). The detection of smell in spoiled meat by TGS822 gas sensor for an electronic nose used in rotten food. In Á. Rocha, M. Serrhini, & C. Felgueiras (Eds.), *Europe and MENA cooperation advances in information and communication technologies (Advances in intelligent systems and computing book 520)* (1st ed. 2017 ed., Vol. 520, pp. 279–286). Springer, Cham. https://doi.org/10.1007/978-3-319-46568-5\_29

Bhargava, A. (2020, November 19). 8 famous mobile apps based on Node J.S. Mobile App Development. Apptunix Blog. Retrieved October 24, 2021, from <https://www.apptunix.com/blog/8-famous-mobile-apps-based-on-node-js-mobile-app-development/>

Bhat, R., & Rao, V. S. (2019). *Unit-4 food spoilage*. Indira Gandhi National Open University, New Delhi. Retrieved October 20, 2021, from http://egyankosh.ac.in//handle/123456789/73118

Bohbot, K. (2021, September 8). *Google Firebase pricing for dummies*. Back4App Blog. Retrieved November 18, 2021, from https://blog.back4app.com/firebase-pricing/#Real-Time\_Databases

Britton, J. (2021, May 6). *What is ISO 25010?* Perforce Software. Retrieved November 18, 2021, from https://www.perforce.com/blog/qac/what-is-iso-25010#are

Brown, J. (2020, October 20). *What I learned about WebSockets by building a real-time chat application using Socket.IO.* Medium. Retrieved November 19, 2021, from https://medium.com/@jamesbrown5292/what-i-learned-about-websockets-by-building-a-real-time-chat-application-using-socket-io-3d9e163e504

Brych, M. (2021, June 17). *Data visualization best practices in web and mobile apps*. Espeo Software. Retrieved November 19, 2021, from https://espeo.eu/blog/making-data-visualization-a-major-feature-of-your-next-app/

BusinessMirror Editorial. (2019, November 13). *We can help solve the food waste problem | BusinessMirror editorial*. BusinessMirror. Retrieved October 16, 2021, from https://businessmirror.com.ph/2019/11/14/we-can-help-solve-the-food-waste-problem/.

Cambridge University Press. (n.d.). Spoilage. In *Cambridge Dictionary*. Retrieved November 18, 2021, from https://dictionary.cambridge.org/us/dictionary/english/spoilage

Canadian Institute of Food Safety. (2017, July 29). *Perishable foods and food spoilage*. Retrieved November 18, 2021, from https://www.foodsafety.ca/blog/perishable-foods-and-food-spoilage

Carrillo-Amado, Y. R., Califa-Urquiza, M. A., & Ramón-Valencia, J. A. (2020). Calibration and standardization of air quality measurements using MQ sensors. *Respuestas*, *25*(1), 70–77. https://doi.org/10.22463/0122820x.2408

Chaware, S. (2020, August 15). *MQ series gas sensor*. Robu.In. Retrieved December 30, 2021, from https://robu.in/mq-series-gas-sensor/

Chen, J. (2021, February 3). *Android operating system: What you need to know*. Investopedia. Retrieved October 24, 2021, from https://www.investopedia.com/terms/a/android-operating-system.asp

Chitranshi, R., & Kapoor, R. (2021). Utilization of over-ripened fruit (waste fruit) for the eco-friendly production of ethanol. *Vegetos*, *34*(1), 270–276. https://doi.org/10.1007/s42535-020-00185-8

Christensson, P. (2016, May 16). *Android definition*. TechTerms. Retrieved October 24, 2021, from https://techterms.com/definition/android

Christophe. (2021, February 9). *MQ2. Get started with gas and smoke detector and Arduino. Calibration, value reading*. DIY Projects. Retrieved October 24, 2021, from https://diyprojects.io/mq2-gas-smoke-detector-arduino/#.YcmT6mBBw2x

Circuito Team. (2018, March 11). *Everything you need to know about Arduino code*. Circuito.io Blog. Retrieved October 24, 2021, from https://www.circuito.io/blog/arduino-code/

Clark, M. (2020, October 25). NodeJS backend for your React Native app. Back4App Blog.

Retrieved October 24, 2021, from <https://blog.back4app.com/react-native-app-with-node->backend/

Components101. (2018, March 20). ESP8266 - WiFi module. Retrieved January 23, 2022, from https://components101.com/wireless/esp8266-pinout-configuration-features-datasheet

Cooper, R. (2019, March 20). *6 ways to identify spoiled food*. Taste of Home. Retrieved October 20, 2021, from https://www.tasteofhome.com/collection/how-to-tell-if-food-is-spoiled/

Cooperative Extension Publications. (2021, August 11). *Bulletin #4135, storage conditions: Fruits and vegetables*. Retrieved October 24, 2021, from https://extension.umaine.edu/publications/4135e/

Copes, F. (2020, January 24). *Introduction to the Arduino programming language*. Flavio Copes. Retrieved October 24, 2021, from https://flaviocopes.com/arduino-programming-language/

Cucullo, N. (2021, May 4). *What is Google Firebase: Getting started with mobile application analytics tool*. DELVE. Retrieved November 18, 2021, from https://delvedeeper.com/articles/what-is-google-firebase/

Culkin, J., & Hagan, E. (2017). *Learn electronics with Arduino: An illustrated beginner’s guide to physical computing*. Maker Media. https://books.google.com.ph/books?hl=en&lr=&id=vMw2DwAAQBAJ

Dalmaris, P. (2021, December 14). Arduino tutorials for beginner and intermediate makers. Tech Explorations. Retrieved January 4, 2022, from https://techexplorations.com/arduino/

DHT11–Temperature and Humidity Sensor. (2021, July 16). Components101. Retrieved January 4, 2022, from https://components101.com/sensors/dht11-temperature-sensor

Dijk, E. v. (2018, June 10). Which chart type works best for your data? - Edo van Dijk. Medium.

Retrieved November 20, 2021, from https://medium.com/@edoch/which-chart-type-works-best-for-your-data-1fc309e953ed

Dionne, M. (2021, September 14). Reasons why JavaScript is omnipresent in modern development. Snipcart. Retrieved October 24, 2021, from https://snipcart.com/blog/why-javascript-benefits

Documentation for Visual Studio Code. (n.d.). Visual Studio Code. Retrieved October 31, 2021, from https://code.visualstudio.com/DOCS

Dorcea, D., Hnatiuc, M., & Lazar, I. (2018). Acquisition and calibration interface for gas sensors. *2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging​ (SIITME)*. https://doi.org/10.1109/siitme.2018.8599253

Duran, J. (2021, October 12). Parenting connection: Keeping kids away from foodborne illnesses. WLNS. Retrieved November 18, 2021, from https://www.wlns.com/news/parenting-connection/parenting-connection-keeping-kids-away-from-foodborne-illnesses/

Editor. (2020, February 18). *The good and the bad of Firebase backend services*. AltexSoft. Retrieved November 19, 2021, from https://www.altexsoft.com/blog/firebase-review-pros-cons-alternatives/

ElectronFun.com. (2014, July 21). *How to use gas sensors with Arduino* [Video]. YouTube. Retrieved November 19, 2021, from https://www.youtube.com/watch?v=BIf\_mpnsZvY

Engineering ToolBox. (2004). *Fruits and vegetables - Optimal storage conditions*. Retrieved October 24, 2021, from https://www.engineeringtoolbox.com/fruits-vegetables-storage-conditions-d\_710.html

Eom, K. H., Hyun, K. H., Lin, S., & Kim, J. W. (2014). The meat freshness monitoring system using the smart RFID tag. *International Journal of Distributed Sensor Networks*, *10*(7), 591812. https://doi.org/10.1155/2014/591812

Erkmen, O., & Bozoglu, F. T. (Eds.). (2016). Principles of food spoilage. In *Food Microbiology: Principles into Practice* (1st ed., Vol. 1, pp. 269–278). Wiley. https://doi.org/10.1002/9781119237860.ch15

Fezari, M., & Al Dahoud, A. (2018, October). *Integrated development environment “IDE” for Arduino.* Research Gate. Retrieved October 24, 2021, from https://www.researchgate.net/publication/328615543\_Integrated\_Development\_Environment\_IDE\_For\_Arduino

Finley, K. (2015, November 5). *Artificial nose "smells" when food is about to go bad.* Retrieved November 18, 2021, from from https://www.wired.com/2015/11/c2sense/.

Fluke. (2021, October 5). *What is Ohm’s Law?* Retrieved December 28, 2021, from https://www.fluke.com/en-ph/learn/blog/electrical/what-is-ohms-law

Geddes, M. (2016). *Arduino project handbook: 25 practical projects to get you started* (1st ed.). No Starch Press. https://books.google.com.ph/books?hl=en&lr=&id=Zni7DQAAQBAJ

Gharssellaoui, B. (2018, December 9). *Internet of Things prototyping with Firebase: How to do more with less*. FreeCodeCamp.Org. Retrieved November 18, 2021, from https://www.freecodecamp.org/news/iot-prototyping-with-firebase-doing-more-with-less-2f5c746dac8b/

Ghosh, A. (2018, May 11). *What are RX/TX LEDs and pins on Arduino boards?* The Customize Windows. Retrieved December 29, 2021, from https://thecustomizewindows.com/2018/05/what-are-rx-tx-leds-and-pins-on-arduino-boards/

Gulbis, J. (2016, March 1). Data visualization – how to pick the right chart type? EazyBI. Retrieved November 18, 2021, from https://eazybi.com/blog/data-visualization-and-chart-types

Gustriansyah, R., Suhandi, N., Alie, J., Antony, F., & Heryati, A. (2021). Optimization of laboratory application by utilizing the ISO/IEC 25010 model. *IOP Conference Series: Materials Science and Engineering*, *1088*(1), 012067. https://doi.org/10.1088/1757-899x/1088/1/012067

Hammar, K. (2017). *Content ontology design patterns: Qualities, methods, and tools*. Amsterdam University Press. https://doi.org/10.3384/diss.diva-139584

Hammond, S. T., Brown, J. H., Burger, J. R., Flanagan, T. P., Fristoe, T. S., Mercado-Silva, N., Nekola, J. C., & Okie, J. G. (2015). Food spoilage, storage, and transport: Implications for a sustainable future. *BioScience*, *65*(8), 758–768. https://doi.org/10.1093/biosci/biv081

Hamza, M. (2020, October 24). Introduction to JavaScript. DevTutorialsPioint. Retrieved October 23, 2021, from https://devtutorialspoint.com/introduction-of-javascript/

Hanwei Electronics Co., Ltd. (2010a). *Technical data MQ-2 gas sensor* [Datasheet]. Hanwei Electronics Co., Ltd. http://www.datasheet-pdf.com/PDF/MQ-2-Datasheet-Hanwei-622943

Hanwei Electronics Co., Ltd. (2010b). *Technical data MQ-3 gas sensor* [Datasheet]. Hanwei Electronics Co., Ltd. https://datasheet-pdf.com/PDF/MQ-3-Datasheet-HANWEIELETRONICS-904634

Hanwei Electronics Co., Ltd. (2010c). *Technical data MQ-135 gas sensor* [Datasheet]. Hanwei Electronics Co., Ltd. https://datasheet-pdf.com/PDF/MQ-135-Datasheet-Hanwei-605076

Hareendran, T. K. (2020, July 5). *How to use MQ-135 gas sensor*. Codrey Electronics. Retrieved December 29, 2021, from https://www.codrey.com/electronic-circuits/how-to-use-mq-135-gas-sensor/

Hayes, B. (2020, July 16). Most popular Integrated Development Environments (IDEs) used by data scientists. CustomerThink. Retrieved October 24, 2021, from <https://customerthink.com/most-popular-integrated-development-environments-ides-used-by-data-scientists/>

Helmke, R. (2021, March 23). *Plastics play a vital role in reducing food waste*. Plastic Ingenuity. Retrieved November 21, 2021, from https://www.plasticingenuity.com/blog/packaging-reduces-food-waste/

Henderson, C. (2020, December 18). *5 common causes and prevention of food spoilage for restaurants*. Hubworks. Retrieved November 18, 2021, from https://zipinventory.com/en/restaurant-service/causes-and-prevention-of-food-spoilage.html

Heyasa, B. B. L., & Galarpe, V. R. K. R. (2017). Preliminary development and testing of Microcontroller-MQ2 gas sensorfor university air quality monitoring. *IOSR Journal of Electrical and Electronics Engineering*, *12*(03), 47–53. https://doi.org/10.9790/1676-1203024753

Hussaini, U. (2020, January 13). *Arduino Uno hardware – ultimate guide to Arduino parts and components*. Technobyte. Retrieved October 24, 2021 from https://technobyte.org/arduino-hardware-parts-explained-components/

*ISO/IEC 25010*. (n.d.). ISO 25000. Retrieved November 16, 2021, from

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Jaycon Systems. (2021, December 1). *How do gas sensors work?* Retrieved December 28, 2021, from https://jayconsystems.com/blog/understanding-a-gas-sensor

Louis, L. (2016). Working principle of Arduino and using it as a tool for study and research. *International Journal of Control, Automation, Communication and Systems*, *1*(2), 21–29. https://doi.org/10.5121/ijcacs.2016.1203

K, J. (2020, December 2). React Native: Features, limitations, pros and cons. Web Solutions Blog. Retrieved October 22, 2021, from https://acodez.in/react-native-features/#:%7E:text=Features%20of%20React%20Native%201%20Write%20Once%20and,they%20call%20magic%20in%20mobile%20application%20development.%20

Khawas, C., & Shah, P. (2018). Application of Firebase in Android app development-a study. *International Journal of Computer Applications*, *179*(46), 49–53. https://doi.org/10.5120/ijca2018917200

Kidder, T. J. (2020, October 18). What is a JavaScript framework? LEARN academy - San Diego’s original coding bootcamp. Retrieved October 23, 2021, from <https://www.learnacademy.org/blog/what-is-a-javascript-framework/>

Kumar, A. (2018). *Mastering Firebase for Android development: Build real-time, scalable, and cloud-enabled Android apps with Firebase*. Packt Publishing. https://books.google.com.ph/books?id=RMNiDwAAQBAJ&lpg=PP1&pg=PP1#v=onepage&q&f=false

Kumar, G. K. K., & Senan, P. V. (2020). Bioethanol production from local fruit waste and its optimization. *Indian Journal of Experimental Biology*, *58*(12), 879–882. http://nopr.niscair.res.in/bitstream/123456789/55801/1/IJEB%2058%2812%29%20879-882.pdf

Kumar, K., Mathur, A., Rawal, M. K., Malhotra, A., Sandal, A., & Jain, S. (2018). *Food and dairy microbiology*. Vikas Publishing House Pvt. Ltd. https://alagappauniversity.ac.in/siteAdmin/dde-admin/uploads/2/PG\_M.Sc.\_Microbiology\_364%2023\_Food%20and%20Dairy%20Microbiology.pdf

Last Minute Engineers. (2020a, December 18). *How MQ2 gas/smoke sensor works? & interface it with Arduino*. Retrieved December 28, 2021, from https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

Last Minute Engineers. (2020b, December 18). How MQ3 alcohol sensor works? & Interface it with Arduino. Retrieved January 5, 2022, from https://lastminuteengineers.com/mq3-alcohol-sensor-arduino-tutorial/

Levy, P. (2019, September 6). Real-time IoT app with React + Firebase + esp8266 - the startup. Medium. Retrieved November 18, 2021, from <https://medium.com/swlh/real-time-iot-app-with-react-firebase-esp8266-e2e673efd1bf>

Long, M. (2020, May 4). *Best Arduino IDEs for Android*. Electromaker. Retrieved November 18, 2021, from https://www.electromaker.io/blog/article/best-android-arduino-ides-for-programming-arduino-with-android

Macfos, M. (2021, December 28). MQ135 air quality detector sensor module for Arduino. Robu.in | Indian Online Store | RC Hobby | Robotics. Retrieved January 5, 2022, from https://robu.in/product/mq-135-air-quality-gas-detector-sensor-module-for-arduino/

Magoulas, A. (2016, March 25). *Protecting your family from food spoilage*. USDA. Retrieved November 18, 2021, from https://www.usda.gov/media/blog/2016/03/25/protecting-your-family-food-spoilage

Makerspaces.com. (2017, February 7). *Arduino for beginners*. Retrieved November 20, 2021, from https://www.makerspaces.com/arduino-uno-tutorial-beginners/

Malysheva, V. A. I. (2021, July 29). *What is data visualization: Definition, examples, principles, tools*. Owox. Retrieved November 18, 2021, from https://www.owox.com/blog/articles/data-visualization/

Mansoor, S. (2021, June 17). *Building Android Java apps using MS VS code instead of Android Studio* [LinkedIn post]. LinkedIn. https://www.linkedin.com/pulse/building-android-apps-using-microsofts-vs-code-instead-saamer-mansoor/

Matindoust, S., Baghaei-Nejad, M., Shahrokh Abadi, M. H., Zou, Z., & Zheng, L. R. (2016). Food quality and safety monitoring using gas sensor array in intelligent packaging. *Sensor Review*, *36*(2), 169–183. https://doi.org/10.1108/sr-07-2015-0115

McKenzie, C. (2018, April 17). JavaScript. TheServerSide.Com. Retrieved October 23, 2021, from https://www.theserverside.com/definition/JavaScript

McKinnon, J. (2020, December 16). 10 best IDE software. WebsiteSetup. Retrieved October 24, 2021, from <https://websitesetup.org/best-ide->software/#:%7E:text=1%20Visual%20Studio.%20The%20Visual%20Studio%20IDE%20is,. . .%209%20RubyMine.%20. . .%2010%20Komodo%20IDE.%20

Megida, D. (2021, April 29). What is JavaScript? a definition of the JS programming language. FreeCodeCamp.Org. Retrieved October 23, 2021, from https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/

Microcontrollerkits. (2016, December 28). ESP8266 sent sensor data to Firebase Realtime Database. http://microcontrollerkits.blogspot.com/2016/12/esp8266-firebase.html

meat. In F. Toldra (Ed.), *Lawrie’s Meat Science (Woodhead Publishing Series in Food Science, Technology and Nutrition)* (8th ed., pp. 461–499). Woodhead Publishing. https://doi.org/10.1016/B978-0-08-100694-8.00015-7

Mitchell, R. (2018, August 30). *Common communication peripherals on the Arduino: UART, I2C, and SPI*. Maker Pro. Retrieved October 24, 2021, from https://maker.pro/arduino/tutorial/common-communication-peripherals-on-the-arduino-uart-i2c-and-spi

Mittal, A. (2021, July 15). The top 8 React Native chart libraries for 2021. LogRocket Blog. Retrieved October 22, 2021, from https://blog.logrocket.com/the-top-8-react-native-chart-libraries-for-2021/

Mohammadi, Z., & Jafari, S. M. (2020). Detection of food spoilage and adulteration by novel nanomaterial-based sensors. *Advances in Colloid and Interface Science*, *286*, 102297. https://doi.org/10.1016/j.cis.2020.102297

Moirangthem, K., & Tucker, G. (2018). How do fruits ripen? *Frontiers for Young Minds*, *6*. https://doi.org/10.3389/frym.2018.00016

Moneruzzaman Khandaker, M., Aliyu Abdullahi, U., Dogara Abdulrahman, M., Afiza Badaluddin, N., & Suryati Mohd, K. (2020). Bio-Ethanol production from fruit and vegetable waste by using Saccharomyces cerevisiae. *Bioethanol Technologies*. https://doi.org/10.5772/intechopen.94358

Mouser Electronics. (n.d.). *DHT11 Humidity & Temperature Sensor* [Datasheet]. Mouser Electronics. https://www.mouser.ph/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

MQ2 gas sensor. (2018, April 1). Components101. Retrieved January 5, 2022, from https://components101.com/sensors/mq2-gas-sensor

MSIS 2629. (2019, June 10). *Chapter 4 patterns | A reader on data visualization*. Retrieved November 19, 2021, from https://mschermann.github.io/data\_viz\_reader/patterns.html

Nagar, T. (2019, September 5). Why Node JS considered as best mobile app development platform. YourStory.Com. Retrieved October 24, 2021, from https://yourstory.com/mystory/why-node-js-considered-as-best-mobile-app-developm-93kihfrwnr/amp

Nguyen, D. (2018, June 4). *Firebase with realtime database for IoT applications*. Medium. Retrieved November 18, 2021, from https://medium.com/konel/firebase-with-realtime-database-for-iot-applications-e615a7057a48

Nussey, J. (2018). *Arduino for dummies* (2nd ed.). John Wiley & Sons. https://books.google.com.ph/books?id=qKVoDwAAQBAJ

O'Reilly Online Learning. (n.d.). Learning React Native. Retrieved October 22, 2021, from https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

Parekh, L. (2021, March 23). The basics of food security (and how it’s tied to everything). World Vision. Retrieved October 22, 2021, from https://www.worldvision.ca/stories/food/the-basics-of-food-security

Parmar, S., Manke, T., Badhan, N., Borase, P., & Ujgare, N. S. (2020). An efficient system to detect freshness and quality of food. *International Research Journal of Engineering and Technology*, *07*(04), 1333–1336. https://www.irjet.net/archives/V7/i4/IRJET-V7I4250.pdf

Patni, S. (2020, August 19). *How to get an Arduino and your smartphone to talk to each other*. Medium. Retrieved November 18, 2021, from <https://betterprogramming.pub/how-to-get-arduino-and-smartphone-talk-to-each-other-9abaf39d2ff>6

Pedamkar, P. (2021, October 7). What is Visual Studio Code? EDUCBA. Retrieved October 31, 2021, from https://www.educba.com/what-is-visual-studio-code/

Petruzzi, L., Corbo, M. R., Sinigaglia, M., & Bevilacqua, A. (2017). Microbial spoilage of foods. *The Microbiological Quality of Food*, 1–21. https://doi.org/10.1016/B978-0-08-100502-6.00002-9

Philip, R. E. (2019, December 30). Benefits of using JavaScript for mobile app development. Medium. Retrieved November 15, 2021, from https://medium.com/@rohithaelsa/benefits-of-using-javascript-for-mobile-app-development-e1e71aa94e21

Piyadigama, D. (2020, May 27). Expo vs React-Native-CLI - Dinuka Piyadigama. Medium. Retrieved November 22, 2021, from https://dinukapiyadigama.medium.com/expo-vs-react-native-cli-7a3019b2760d

Poore, J., & Nemecek, T. (2018)*.* Reducing food’s environmental impacts through producers and consumers. *Science,* 360(6392), 987–992.  <https://doi.org/10.1126/science.aaq0216>.

Prasaath, H. (2018, March 21). *Arduino based smart IoT food quality monitoring system*. Engineers Garage. Retrieved October 24, 2021, from https://www.engineersgarage.com/arduino-based-smart-iot-food-quality-monitoring-system/

Puffelen, F. V. (2016, July 8). *Have you met the realtime database?* The Firebase Blog. Retrieved November 18, 2021, from https://firebase.googleblog.com/2016/07/have-you-met-realtime-database.html

Rashmin, R. (2018, March 26). Arduino to Android, real time communication for IoT with Firebase. Medium. Retrieved November 18, 2021, from https://rashm1n.medium.com/arduino-to-android-real-time-communication-for-iot-with-firebase-60df579f962

Rupareliya, R. (2019, November 17). *Firebase Realtime Database with React Native - MindOrks*. Medium. Retrieved November 18, 2021, from https://medium.com/mindorks/firebase-realtime-database-with-react-native-5f357c6ee13b

Sahu, M., & Bala, S. (2017). Food processing, food spoilage and their prevention: An overview. *International Journal of Life-Sciences Scientific Research*, *3*(1). https://doi.org/10.21276/ijlssr.2017.3.1.1

Sandhu, R. (2020, December 2). *Using a mobile device to interface with Arduino*. Lifewire. Retrieved November 18, 2021, from https://www.lifewire.com/arduino-and-mobile-phone-projects-2495266

Setting up the development environment · React Native. (2021, March 12). React Native. Retrieved November 22, 2021, from <https://reactnative.dev/docs/environment-setup>

Shahzad, N., Iqbal, A., Khalid, U., & Rahman, M. U. (2018). eFresh – a device to detect food freshness. *International Journal of Soft Computing and Engineering*, *8*(3), 1–4. https://www.ijsce.org/wp-content/uploads/papers/v8i3/C3146078318.pdf

Shaikh, M. (2021, May 16). *Firebase Realtime Database*. Mindbowser. Retrieved November 17, 2021, from https://www.mindbowser.com/firebase-real-time-database/

Sharada, G., Sindhoora, S., Devi, D. S. L., Anisha, P. R., Murthy, B. V. R., & Reddy, C. K. K. (2019). Smart stale food detector using IoT (internet of things). *Global Journal of Engineering Science and Researches*, 315–327. http://www.gjesr.com/Issues%20PDF/ICITAIC-2019/37.pdf

Singh, P. (2020, August 7). Getting started with React Native on Android. Pluralsight. Retrieved November 22, 2021, from <https://www.pluralsight.com/guides/getting-started-with-reactnative-on-android>

Singh, R., & Anderson, B. (2004). The major types of food spoilage: An overview. *Understanding and Measuring the Shelf-Life of Food*, 3–23. https://doi.org/10.1533/9781855739024.1.3

Softermii. (2021, September 15). Most popular IDEs for software development: Overview for 2021. Retrieved November 1, 2021, from https://www.softermii.com/blog/top-ides-for-software-development

Sommer, S. (2019, January 21). *What is sensor calibration and why is it important?* PLC Programming Courses for Beginners | RealPars. Retrieved November 21, 2021, from https://realpars.com/sensor-calibration/

SpeedySense Editorial. (2021, May 13). What are the top React Native features, limitations, pros and cons. SpeedySense. Retrieved November 18, 2021, from https://speedysense.com/react-native-features-limitations/

Sullivan, J., & Sullivan, D. (2016, March 2). *How to use Google Firebase to build real-time IoT apps*. IoT Agenda. Retrieved November 18, 2021, from https://internetofthingsagenda.techtarget.com/feature/How-to-use-Google-Firebase-to-build-real-time-IoT-apps

Swizec, S. (2017, June 28). *Tutorial: Build a real-time visualization app of Bitcoin transactions*. Shoutem React Native School. Retrieved November 22, 2021, from https://school.shoutem.com/lectures/tutorial-bitcoin-transactions-react-native/

Tambotoh, J. J., Isa, S. M., Gaol, F. L., Soewito, B., & Warnars, H. L. H. S. (2016). Software quality model for Internet of Things governance. *2016 International Conference on Data and Software Engineering (ICoDSE)*, 2. https://doi.org/10.1109/icodse.2016.7936138

Technical Research Centre of Finland (VTT). (2015, May 6). *Sensor detects spoilage of food*. ScienceDaily. Retrieved November 19, 2021, from https://www.sciencedaily.com/releases/2015/05/150506084555.htm

Thakur, N. (2018, June 20). What is Visual Studio Code and its advantages. Webner Blogs - ELearning, Salesforce, Web Development & More. Retrieved October 22, 2021, from https://blog.webnersolutions.com/visual-studio-code/

The Editors of Encyclopaedia Britannica. (2020, August 22). *Android*. Encyclopedia Britannica. Retrieved October 24, 2021, from https://www.britannica.com/technology/Android-operating-system

The Scientific World. (2018, July 7). *Importance of food in our daily life-how eating healthy affects our health?* the scientific world - let’s have a moment of science. Retrieved November 22, 2021, from https://www.scientificworldinfo.com/2018/07/importance-of-food-in-our-daily-life.html

Thinkwik. (2018, January 31). React Native: What is it? and, why is it used? - Thinkwik. Medium. Retrieved November 18, 2021, from https://medium.com/@thinkwik/react-native-what-is-it-and-why-is-it-used-b132c3581df

TonesB. (2017, September 25). ESP8266 WiFi module for dummies. Instructables. Retrieved January 23, 2022, from https://www.instructables.com/ESP8266-WiFi-Module-for-Dummies/

Tucker, C. (2017, April 11). *Replacing WebSockets with the auto-scaling Firebase database.* Medium. Retrieved November 19, 2021, from <https://medium.com/@christian.tucker_68732/replacing-websockets-with-the-auto-scaling-firebase-database-b451531acea>9

United States Department of Agriculture. (2016, March 25). *Protecting your family from food spoilage*. USDA. Retrieved October 20, 2021, from https://www.usda.gov/media/blog/2016/03/25/protecting-your-family-food-spoilage

Unwin, A. (2020, January 31). *Why is data visualization important? what is important in data visualization? · Issue 2.1, Winter 2020*. Harvard Data Science Review. <https://hdsr.mitpress.mit.edu/pub/zok97i7p/release/3>

Uzayr, B. S. (2021). Introduction to Visual Studio Code. SpringerLink. Retrieved November 1, 2021, from <https://link.springer.com/chapter/10.1007/978-1-4842-7344->9\_1?error=cookies\_not\_supported&code=59b0c1a4-99bf-4a37-b8b3-f34ba6acce3e

Varshney, H., Allahloh, A. S., & Sarfraz, M. (2019). IoT based eHealth management system using Arduino and Google Cloud Firestore. *2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. Published. https://doi.org/10.1109/upcon47278.2019.8980238

Voorugonda, M. (2021). Food spoilage detection. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3918278

Weerasinghe, R. (2020, June 15). Expo CLI/Tool vs React Native CLI - React Native for beginners. Medium. Retrieved November 22, 2021, from https://medium.com/react-native-for-beginners/expo-cli-tool-vs-react-native-cli-23bcd59f4a99

Wijaya, D. R., Sarno, R., & Zulaika, E. (2018). Electronic nose dataset for beef quality monitoring in uncontrolled ambient conditions. *Data in Brief*, *21*, 2414–2420. https://doi.org/10.1016/j.dib.2018.11.091

Wolfe, B. E. (2014, December 15). *Why does rotting food smell bad?* Tufts now. Retrieved November 19, 2021, from <https://now.tufts.edu/articles/why-does-rotting-food-smell-bad>

Wonderopolis. (n.d.). *Why does food rot?* Retrieved November 22, 2021, from https://wonderopolis.org/wonder/why-does-food-rot#:%7E:text=When%20food%20is%20exposed%20to%20light%2C%20its%20outer%20layers%20can,flavor%2C%20vitamins%2C%20and%20proteins

World Health Organization. (2020, April 30). *Food safety*. Retrieved October 20, 2021, from https://www.who.int/news-room/fact-sheets/detail/food-safety#:%7E:text=An%20estimated%20600%20million%20%E2%80%93%20almost,healthy%20life%20years%20(DALYs).

Wright, K. (2019, October 2). *What’s spoiling your food? find out the top 3 factors*. MadgeTech. Retrieved November 17, 2021, from https://www.madgetech.com/posts/blogs/whats-spoiling-your-food-find-out-the-top-3-factors/

Younghans, S.F. (2020, May 31). *5 side effects of eating expired foods*. Retrieved October 20, 2021, fromhttps://www.eatthis.com/eating-expired-foods/

Zuo, B. (2020). Grove - temperature&humidity sensor(DHT11) - seeed wiki. Seeed Wiki. Retrieved January 4, 2022, from https://wiki.seeedstudio.com/Grove-TemperatureAndHumidity\_Sensor/

**Appendix A**

**SAMPLE EVALUATION SHEET**

**Technological University of the Philippines**

**College of Science**

**Computer Studies Department**

Name(Optional):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Occupation:\_\_\_\_\_\_\_\_\_\_Date: \_\_\_\_\_\_\_\_\_\_

Direction: Please encircle the appropriate number of your rating to evaluate the project entitled “Development of Smart Food Spoilage Detection System using Arduino-Based Sensors Integrated in Mobile app” using the scale below:

4 – Highly Acceptable 3 – Very Acceptable 2 – Acceptable 1 – Not Acceptable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arduino Device** | | | | |
| **Criteria** | **Rating** | | | |
| 1. **Functionality Suitability** | | | | |
| 1. The device is suitable or fitted to the target functionality. (suitability) | **4** | **3** | **2** | **1** |
| 1. The functions included in the device are fully functional. (appropriateness) | **4** | **3** | **2** | **1** |
| 1. The functions included in the device cover all the specified tasks and user objectives. (completeness) | **4** | **3** | **2** | **1** |
| 1. The device provides the correct results with the needed degree of precision. (correctness) | **4** | **3** | **2** | **1** |
| 1. **Performance Efficiency** | | | | |
| * + - 1. The device can process and respond in a timely manner. (time behavior) | **4** | **3** | **2** | **1** |
| * + - 1. The amount and types of resources used in the device are efficient and appropriate. (resource utilization) | **4** | **3** | **2** | **1** |
| * + - 1. The device’s maximum limits of processes are reasonable. (capacity) | **4** | **3** | **2** | **1** |
| 1. **Compatibility** | | | | |
| * + 1. The device can perform the required functions efficiently while sharing resources with the mobile application. (co-existence) | **4** | **3** | **2** | **1** |
| 1. The device can exchange information with the mobile application and use the information that has been exchanged. (interoperability) | **4** | **3** | **2** | **1** |
| 1. **Usability** | | | | |
| * + 1. The users can recognize that the device is appropriate for their needs. (appropriateness recognizability) | **4** | **3** | **2** | **1** |
| * + 1. Users can use the device to achieve specified goals of learning to use the system with effectiveness, efficiency, free from risk, and satisfaction in a specified context of use. (learnability) | **4** | **3** | **2** | **1** |
| * + 1. The device has attributes that make it easy to operate and control. (operability) | **4** | **3** | **2** | **1** |
| * + 1. The device protects users against making errors. (user error protection) | **4** | **3** | **2** | **1** |
| * + 1. The device enables pleasing and satisfying interaction for the user. (user interface aesthetics) | **4** | **3** | **2** | **1** |
| * + 1. The device can be used by people with the widest range of capabilities to achieve the specified goal in a specified context of use. (accessibility) | **4** | **3** | **2** | **1** |
| 1. **Reliability** | | | | |
| 1. The device is stable for frequent use. (maturity) | **4** | **3** | **2** | **1** |
| 1. The device works even there are some hardware faults. (fault tolerance) | **4** | **3** | **2** | **1** |
| 1. The device can be accessed easily when needed. (availability) | **4** | **3** | **2** | **1** |
| 1. **Maintainability** | | | | |
| 1. Can identify the source of failure when an error is encountered. (analyzability) | **4** | **3** | **2** | **1** |
| 1. The device can easily be modified for expansion and correction. (modifiability) | **4** | **3** | **2** | **1** |
| 1. **Portability** |  |  |  |  |
| 1. The device can effectively adapt to changes in its environment or in parts of the system itself. | **4** | **3** | **2** | **1** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mobile Application** | | | | |
| **Criteria** | **Rating** | | | |
| 1. **Functionality Suitability** | | | | |
| 1. The mobile application is suitable or fitted to the target functionality. (suitability) | **4** | **3** | **2** | **1** |
| 1. The functions included in the mobile application are fully functional. (appropriateness) | **4** | **3** | **2** | **1** |
| 1. The functions included in the mobile application cover all the specified tasks and user objectives. (completeness) | **4** | **3** | **2** | **1** |
| 1. The mobile application provides the correct results with the needed degree of precision. (correctness) | **4** | **3** | **2** | **1** |
| 1. **Performance Efficiency** | | | | |
| 1. The mobile application can process and respond in a timely manner. (time behavior) | **4** | **3** | **2** | **1** |
| 1. The amount and types of resources used in the mobile application is efficient and appropriate. (resource utilization) | **4** | **3** | **2** | **1** |
| 1. The mobile application’s maximum limits of processes are reasonable. (capacity) | **4** | **3** | **2** | **1** |
| 1. **Compatibility** | | | | |
| 1. The mobile application can perform the required functions efficiently while sharing resources with the Arduino device. (co-existence) | **4** | **3** | **2** | **1** |
| 1. The mobile application can exchange information with the Arduino device and use the information that has been exchanged. (interoperability) | **4** | **3** | **2** | **1** |
| 1. **Usability** | | | | |
| 1. The users can recognize that the mobile application is appropriate for their needs. (appropriateness recognizability) | **4** | **3** | **2** | **1** |
| 1. Users can use the mobile application to achieve specified goals of learning with effectiveness, efficiency, free from risk, and satisfaction in a specified context of use. (learnability) | **4** | **3** | **2** | **1** |
| 1. The mobile application has attributes that make it easy to operate and control. (operability) | **4** | **3** | **2** | **1** |
| 1. The mobile application protects users against making errors. (user error protection) | **4** | **3** | **2** | **1** |
| 1. The mobile application enables pleasing and satisfying interaction for the user. (user interface aesthetics) | **4** | **3** | **2** | **1** |
| 1. The mobile application can be used by people with the widest range of capabilities to achieve the specified goal in a specified context of use. (accessibility) | **4** | **3** | **2** | **1** |
| 1. **Reliability** | | | | |
| 1. The mobile application is stable for frequent use. (maturity) | **4** | **3** | **2** | **1** |
| 1. The mobile application works even there are some software faults. (fault tolerance) | **4** | **3** | **2** | **1** |
| 1. The mobile application can be accessed easily when needed. (availability) | **4** | **3** | **2** | **1** |
| 1. **Security** | | | | |
| * + - 1. The mobile application prevents unauthorized access to the app and Arduino device. (integrity) | **4** | **3** | **2** | **1** |
| 1. **Maintainability** | | | | |
| * 1. Can identify the source of failure when an error is encountered. (analyzability) | **4** | **3** | **2** | **1** |
| * 1. The mobile application can easily be modified for expansion and correction. (modifiability) | **4** | **3** | **2** | **1** |
| 1. **Portability** | | | | |
| 1. The mobile application can run on different android smartphone devices with at least 4.4 version of Android Operating System. (adaptability) | **4** | **3** | **2** | **1** |
| 1. The mobile application can be easily installed (installability) | **4** | **3** | **2** | **1** |

Comments/Suggestions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**RESEARCHER’S PROFILE**

**NORMAN V. CABILING**

311-D Lakandula St., Tondo, Manila

0946-947-1381

normancabiling123@gmail.com

|  |  |
| --- | --- |
| **PERSONAL INFORMATION** | |
| Age: | 21 |
| Sex: | Male |
| Birthday: | January 1, 2000 |
| Civil Status: | Single |
| Citizenship: | Filipino |

|  |
| --- |
| **KNOWLEDGE AND SKILLS** |
| * Knowledgeable in C, C++, Java, MySQL, MongoDB, HTML, CSS, Bootstrap, Node.js, JavaScript, and Arduino. * Knowledgeable in multi-media editing using Microsoft Windows Programs. |

|  |
| --- |
| **ACCOMPLISHED PROJECTS** |
| * **Fingerprint Attendance System with Cloud Data (Arduino Project)**   + Developer   + Mar 2020 – Aug 2020 |
| * **EggBook (Web Project)**   + Backend Developer   + November 2020 – February 2021 |

|  |  |
| --- | --- |
| **EDUCATIONAL ATTAINMENT** | |
| Tertiary: | Technological University of The Philippines  Ayala Blvd, Ermita, Manila  Bachelor of Science in Information Technology |
| Secondary: | Raja Soliman Science and Technology High School  Urbiztondo Binondo, Manila  2012 - 2018 |
| Primary: | Isabelo Delos Reyes Elementary School  Zamora St., Tondo, Manila  2006 - 2012 |

**JULIE ANN C. COSICO**

3095 Col. Santos St., South Cembo, Makati City

0905-047-0778

cosicojulieann@gmail.com

|  |  |
| --- | --- |
| **PERSONAL INFORMATION** | |
| Age: | 20 |
| Sex: | Female |
| Birthday: | July 11, 2000 |
| Civil Status: | Single |
| Citizenship: | Filipino |

|  |
| --- |
| **KNOWLEDGE AND SKILLS** |
| * Knowledgeable in C, C++, Java, Visual Basic, MySQL, Oracle, PHP, HTML, CSS, Bootstrap, Node.js, JavaScript, and React Native. * Knowledgeable in mobile application development, web development, and scrum-agile methodology. * Skilled at project management and system analysis. |

|  |
| --- |
| **ACCOMPLISHED PROJECTS** |
| * **Bit OS (Java Application)**   Project Manager  Mar 2020 – Aug 2020 |
| * **NoteBook: A mobile application developed using React Native**   Backend Developer  Jun 2020 – Jul 2020 |

|  |  |
| --- | --- |
| **EDUCATIONAL ATTAINMENT** | |
| Tertiary: | Technological University of The Philippines  Ayala Blvd, Ermita, Manila  Bachelor of Science in Information Technology |
| Secondary: | Pitogo High School  Negros Street, Pitogo, Makati City  2012 - 2018 |
| Primary: | Pitogo Elementary School  Cebu Street, Pitogo, Makati City  2006 - 2012 |

**A picture containing person, clothing, indoor, young

Description automatically generatedMHIGUEL ROVY G. DE NAVA**

2092 V. Serrano St., Tondo, Manila

0915-234-6648

mhiguelrovy.denava@gmail.com

|  |  |
| --- | --- |
| **PERSONAL INFORMATION** | |
| Age: | 21 |
| Sex: | Male |
| Birthday: | August 25, 2000 |
| Civil Status: | Single |
| Citizenship: | Filipino |

|  |
| --- |
| **KNOWLEDGE AND SKILLS** |
| * Knowledgeable in C, C++, Visual Basic, Java, MySQL, HTML, CSS, Bootstrap, Node.js, PHP, JavaScript, and React Native. * Knowledgeable in web development and mobile application development. |

|  |
| --- |
| **ACCOMPLISHED PROJECTS** |
| * **Bit OS (Java Application)**   UX Developer  March 2020 – Aug 2020 |
| * **Inventory System (Visual Basic Application)**   Database Manager  April 2021 – June 2021 |

|  |  |
| --- | --- |
| **EDUCATIONAL ATTAINMENT** | |
| Tertiary: | Technological University of The Philippines  Ayala Blvd, Ermita, Manila  Bachelor of Science in Information Technology |
| Secondary: | Manila Cathedral School  287 Tayuman St., Tondo, Manila  2012 - 2018 |
| Primary: | Manila Cathedral School  287 Tayuman St., Tondo, Manila  2006 - 2012 |

**ALEXANDRA LOREINNE S. GUZMAN**



Blk. 17, Lot 7, Soldiers Hills Village, Putatan, Muntinlupa City

0905-392-2460

alexandraloreinne.guzman@gmail.com

|  |  |
| --- | --- |
| **PERSONAL INFORMATION** | |
| Age: | 21 |
| Sex: | Female |
| Birthday: | November 7, 1999 |
| Civil Status: | Single |
| Citizenship: | Filipino |

|  |
| --- |
| **KNOWLEDGE AND SKILLS** |
| * Knowledgeable in C, C++, Visual Basic, Java, MySQL, HTML, CSS, Bootstrap, Node.js, PHP, JavaScript, and React Native. * Knowledgeable in web development and mobile application development. |

|  |
| --- |
| **ACCOMPLISHED PROJECTS** |
| * **Auto Supply POS (Visual Basic Application)**   Developer  Aug 202 – Sept 2021 |
| * **i-Book Website (Web Project)**   Backend Developer  November 2020 – February 2021 |

|  |  |
| --- | --- |
| **EDUCATIONAL ATTAINMENT** | |
| Tertiary: | Technological University of The Philippines  Ayala Blvd, Ermita, Manila  Bachelor of Science in Information Technology |
| Secondary: | Muntinlupa Science High School  999 Buendia Street, Tunasan, Muntinlupa City  2012 - 2018 |
| Primary: | Bayanan Elementary School MAIN  Bayanan, Muntinlupa City  2006 - 2012 |