Internet of Things Coursework

Offline-first Automatic Pet Feeder

up899244

May 13, 2022

Abstract – The Internet of Things is an emerging topic of technical, social, and economic significance representing the future of computing and communications. With the popularity of the IoT technology, technical issues are found every day as we use it on a daily basis. This paper aims to find a solution to the issue which was recently raised – pet feeders not delivering food to pets when going offline. Literature relevant to the current trends in this area will be researched, critically reviewed and conclusions will be drawn to design an IoT application. The outlined solution will be implemented, tested and the overall outcome will be discussed. A conclusion of the work performed will be deduced, and future work recommendations will be given.

Index Terms - Internet of Things, Pet Feeder, Web Server, Arduino

1 Introduction

1.1 What is the Internet of Things?

The term "Internet of Things" (IoT) was first used in 1999 by British technology pioneer Kevin Ashton to describe a system in which objects in the physical world could be connected to the Internet with radio-frequency identification technology (Gokhale et al., 2018). In the past few years, IoT has been developed rapidly and a large number of enabling technologies have been proposed which leads to the exact definition of IoT to be still in the forming process that is subject to the perspectives taken (Ma, 2014).

Nowadays, the term Internet of Things refers to scenarios where network connectivity and computing capability extends to objects, sensors, and everyday computing items (Eldridge et al., 2015). However, there is still no universal definition. In general terms, authors Xia et al. state that IoT refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence (2012).

1.2 Problem Overview

Smart Pet Feeders are popular IoT solutions nowadays for pet owners looking to schedule meals and set portions for their pets when they are unable to. While they ensure that their pet is fed on time and stays healthy, they are also prone to malfunctions such as going offline (BBC, 2020). As this has serious consequences for the pets, this paper will investigate how it can mitigate the outcome of this fault by implementing an Internet of Things solution.

1.3 Application Scenario

The proposed scenario of the IoT application is to have the user choose the time their pet should be fed on a web page, submit it to the IoT device and have the device store the time value. As the time value will be stored, the device will be able to provide feed even if the application goes offline. The user should be able to update the value and the application should deliver food to the new time specified.

1.4 Design Decomposition

An example of a Pet Feeder IoT system is Adriansyah et al.'s IoT application consisting of a Wi-Fi submodule which connects to the web server through Arduino Uno R3 (2016) and uses motors to deliver pet food. Authors Birha et al. and Own et al. (2013) also suggest that a timer module is necessary for the pet feeder to be automatic and support the view that a web server is needed. These works highlight that a suitable power supply is also required for the Arduino board's components to function well.

One of the proposed non-hardware components used to deliver the food are a base, a feeding bowl with pie-shaped divisions, a timer module, a bowl cover, and a locking mechanism to hold the entire unit in place (Own, 2013). Another suggestion is to have an upper part (a food storage), a bottom part (the pour-out-food container) which are both cut out a 90-degree radial opening and to have the center of them connected with the servo (Chen and El-shakankiri, 2020).

From the literature research conducted it was concluded that the proposed solution should contain a web server served by the Arduino board, a submodule which measures the time, a network submodule, a motor which will enable the release of food at the specified time and non-hardware components to implement the release of food. The system submodules will be chosen in the Design section.

2 Selecting hardware components

2.1 Main board

The main board chosen for the IoT application was the Elegoo UNO (Figure 1), which is a reproduced version of Arduino UNO R3. It was chosen for its relatively low cost, the easy implementation with the other selected components by using libraries and the ability to connect it with the developer's computer with a USB power supply.

Other boards which were considered are the Elegoo Mega R3 controller board and the Leonardo R3 board. The main difference between the Arduino Elegoo UNO board and the Elegoo Mega R3 board is the larger flash memory (256kB) of the second board mentioned (Pedamkar, 2022). As the designed application is planned to use a small amount of code and the cost of the Arduino Elegoo UNO was recorded to be less on Amazon, the latter was found to be more suitable.

The Leonardo R3 board was considered due to its lower cost and better performance compared to the Arduino Elegoo UNO board. However, the definitive advantage that the Arduino Elegoo UNO board had over the Leonardo R3 board was seen in the larger community with firm support and lessons, which were more easily accessible (Pedamkar, 2022).

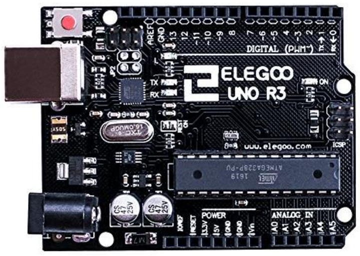


Figure 1: Arduino Elegoo UNO board

2.2 Wi-Fi submodule

ESP8266 ESP-01 is the Wi-Fi submodule which was selected to connect the main board and create the web server (Figure 2). Other hardware components which were considered are the ESP32 which had faster Wi-Fi and Bluetooth but was more expensive. As there is currently no plan to use Bluetooth for the application, the ESP8266 ESP-01 submodule was found suitable.

Figure 2: ESP8266 ESP-01

2.3 Clock submodule

The clock submodule chip was looked for based on accuracy, consistent timekeeping that doesn't reset when the Arduino battery dies/is reprogrammed and for the ability to connect to the board. A submodule in this category that was considered is the DS1302. However, it did not support I2C communication and synchronous 3-wire serial communication was needed to connect it to the Arduino Elegoo UNO board. Since this would have required more jumper wires, the Real Time Clock RTC DS3231 (Figure 3) was chosen as it fulfilled the requirements outlined above.

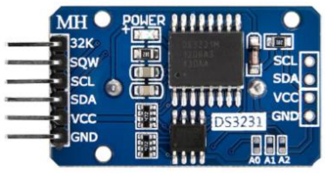


Figure 3: Real Time Clock RTC DS3231

2.4 Micro-servo Motor

The motor used for the IoT application is the SG90 Micro-servo motor (Figure 4). As the payload in the food container box was expected to be small for testing purposes and the food to be released fast, this motor provided the speed and torque this solution would need initially. In terms of cooling inbetween movement, the food was expected to be released once in a long period of time, providing the motor with a rest which will allow it to function longer.

Other motor which was considered is the MG995 micro servo motor compatible with Arduino. Due to its faster speed and better performance against the payload, this submodule was found to be more expensive on Amazon. Since currently it is not considered to be necessary for the application, it is recommended to be used in the future development.

Figure 4: SG90 Micro-servo motor

3 Selecting non-hardware components

The components which realized the food control were selected from the literature researched in the Introduction section. Since it was not possible to find a feeding bowl with pie shaped divisions compatible with a servo motor and implement it at a low cost, a mock food container was used. Duct tape and elastic bands were utilized to attach the servo motor to the food container and card paper was used to cover the food container's opening. As these components are primarily used for testing purposes, it is advised to replace them by sturdier materials in the future.

4 Design

4.1 Selecting Communication Protocols

After reviewing the application scenario and the chosen hardware components, it was decided that the most suitable main communication protocol for the application will be Wi-Fi. The ESP8266 submodule chosen in the second section allowed for the efficient use of this protocol by providing a full TCP/IP stack and HTTP protocol requests. Other protocols which were considered are the Bluetooth and the Zigbee protocols. As they required additional hardware to connect them to the board and were not seen as necessary to the implementation, they were not put in service.

Another protocol which is expected to be used is the I2C Communication Protocol, required by the Real Time Clock RTC DS3231 submodule to connect to the Arduino board. Libraries which will be included to communicate with the submodule are the Wire and the DS3231 library (Rodan, 2018). To control the Servo motor, the Servo library will be applied.

For the programming of the application, the Arduino IDE will be utilized for its ease of use and compatibility with the selected Arduino board. The library SoftwareSerial will be applied to provide Wi-Fi communication and debug the program during the development stage. The Arduino Elegoo UNO board will be set to run at 9600 baud, which means that the connection will be able to transfer a maximum of 9600 bits per second.

4.2 Overall Solution Design

This solution works automatically to provide feed. The user enters a time value to show when the pet should be fed and the food valve will open at this time. This will occur continuously with a predetermined schedule, without the need of the device to be online all the time. The following flowchart demonstrates the data flow of the hardware components in the finalized design (Figure 5).

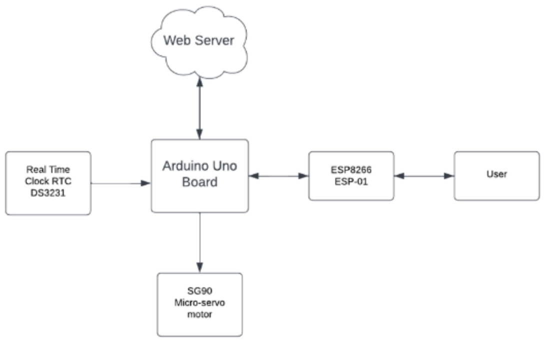


Figure 5: Flowchart

4.3 Algorithm Design

4.3.1 Pseudocode

The main algorithm which decides when the food valve should be opened at the time specified by the user can be seen in Figure 6.

```
if ((t.hour == chosenHour) && (t.min == chosenMin) && t.sec == 0) {
    Move servo motor
}
```

Figure 6: Pseudocode

4.3.2 Algorithm Logic

The algorithm starts with the main page menu (Start), assumes that the page is connected to the Arduino web server provided by the board and follows the algorithm logic outlined in Figure 7.

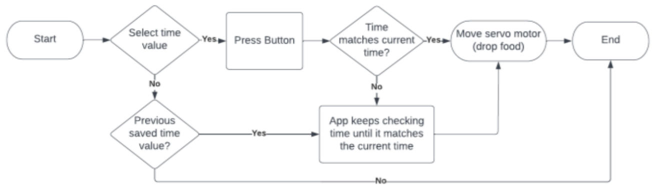

Start
Select time value
Yes
Press Button
Time matches current time?
Yes
Move servo motor (drop food)
End
No
No
Previous saved time value?
Yes
App keeps checking time until it matches the current time
No

Figure 7: Algorithm Logic

4.3.3 Application Circuit

Figure 8 represents the chosen hardware components with the circuit which was planned to be implemented. The Micro-servo motor and the RTC submodule were connected to the Arduino by using the breadboard to reduce the number of wires needed. The ESP8266 ESP-01 required the use of solderless flexible breadboard jumper wires as it was not possible to connect it to the breadboard and the Arduino board with jumper wires. The Arduino board was connected to the developer's computer by using the USB provided, with power input of 5V.

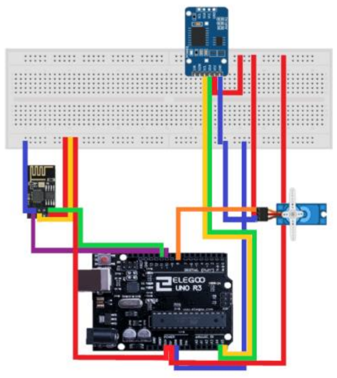


Figure 8: Application circuit

4.4 Limitations

This solution is not being able to accept user input onto the web page if the web page is not fully functional. As a result, it is mandatory to ensure that the time input and the data submission are operating to set the pet feeding time.

Another working condition is the successful connection of the server to the Wi-Fi. Currently the debugging tool is used on the server to secure the connection to the Wi-Fi and to generate the IP address needed by the web page.

4.5 Cost

The overall cost of the equipment can be seen in Table 1. The price of the materials can vary depending on where they were purchased from. For this paper, the materials were purchased from Amazon and the price they had at the time this paper was made was recorded.

Equipment	Cost
Hardware components	£14.99
Elegoo Arduino Uno Board with USB (1pk)	
SG90 Micro-servo motor (1pk)	£5,49
ESP8266 ESP-01 (1pk)	£4,49
Real Time Clock RTC DS3231	£6.48
Jumper wires (40 pcs)	£3.49
Breadboard (1pk)	£2.99
Solderless flexible breadboard jumper Wires (40pcs)	£3.99
Non-hardware components	
Mock food container	£3.00
Duct tape	£1.50
Elastic bands	£1.00
Card paper	£1.00
Total	£48.42

Table 1: Equipment Cost

5 Implementation

5.1 Communication

The Arduino Elegoo Uno was connected to the ESP8266 ESP-01, Real Time Clock RTC DS3231 and the SG90 Micro-servo motor as outlined from the 4.3.3 Application circuit (Figure 9).

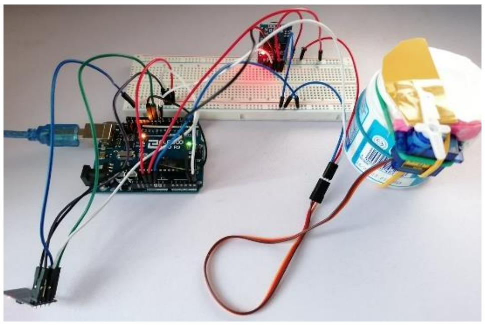


Figure 9: Wired modules

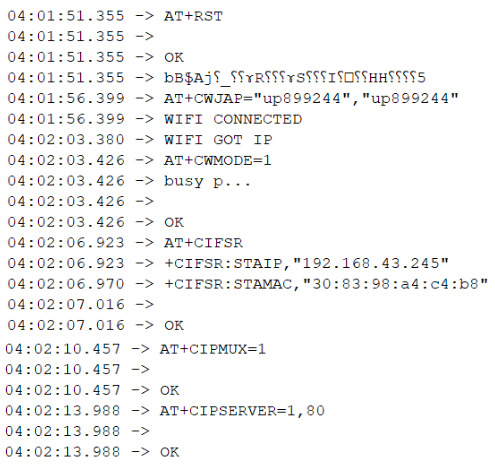
5.2 Programming

The application was programmed with C++, HTML, CSS and JavaScript, using the libraries mentioned in 4.1. A library which was added during the implementation process is the JQuery library, used to connect the application to the web server.

The code starts by initializing the variables needed for setting the time of the RTC submodule and for moving the servo motor. Next, the software and hardware serial communication speed (baud rate) were set to 9600.

5.2.1 Wi-Fi initialization and connection

For the web server part, a sendData function was created to regulate how the AT Commands will get sent to the ESP8266 submodule. This function was used inside the InitWifiModule which gave the commands to initialize a communication between the ESP8266 and the mobile hotspot. Delays were added to make sure that the Wi-Fi connection is made. The commands and their output were printed in the Serial Monitor as seen from Figure 10.


```
04:01:51.355 -> AT+RST
04:01:51.355 ->
04:01:51.355 -> OK
04:01:51.355 -> bB$Aj⸮_⸮⸮rR⸮⸮⸮rS⸮⸮⸮I⸮□⸮⸮HH⸮⸮⸮⸮5
04:01:56.399 -> AT+CWJAP="up899244","up899244"
04:01:56.399 -> WIFI CONNECTED
04:02:03.380 -> WIFI GOT IP
04:02:03.426 -> AT+CWMODE=1
04:02:03.426 -> busy p...
04:02:03.426 ->
04:02:03.426 -> OK
04:02:06.923 -> AT+CIFSR
04:02:06.923 -> +CIFSR:STAIP,"192.168.43.245"
04:02:06.970 -> +CIFSR:STAMAC,"30:83:98:a4:c4:b8"
04:02:07.016 ->
04:02:07.016 -> OK
04:02:10.457 -> AT+CIPMUX=1
04:02:10.457 ->
04:02:10.457 -> OK
04:02:13.988 -> AT+CIPSERVER=1,80
04:02:13.988 ->
04:02:13.988 -> OK
```
Figure 10: Serial monitor Wi-Fi initialization

5.2.2 Setting up the clock time

The Wire.h library was initialized for the I2C interface of the RTC, and the clock was synchronized to match the current time. After the synchronization was done, the code was commented out as the clock recorded the current time without the need of a reset.

5.2.3 Creating the web page and submitting data

The web page was created as seen in Figure 11, using the HTML, CSS and JavaScript languages. The input type for the time chosen by the user was a timestamp.

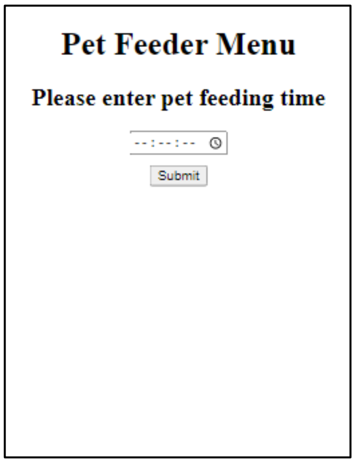
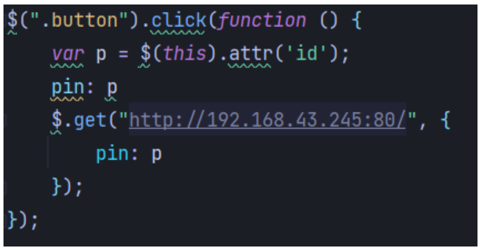


Figure 11: Application web page

If the button from the page was clicked the code in Figure 12 in the script of the HTML page would get executed. By having a button, it was ensured that the application did not submit any time data and no time value was initially set. Consequently, as no feeding time was specified, the application did not send any data and the feeder did not release any food with the motor. On button click, a get request to the web server was sent with the selected data from the web page in the form of the IP address with the pin id.


```
$(".button").click(function () {
    var p = $(this).attr('id');
    pin: p
    $.get("http://192.168.43.245:80/", {
        pin: p
    });
});
```
Figure 12: Sending a get request to the web server

5.2.4 Receiving data from the web page

To receive the data from the web page and set the motor to work at the chosen time, inside the loop it was firstly checked if there is any data received and stored in the serial receiver buffer. The "+IPD," string was searched for in the incoming data and a delay was added to fill the buffer with the data received from the web page. The connection was read as a decimal value. The pin number was read from the Arduino input buffer by adding each number byte by byte. The third byte, which was the chosen time from the web page was read and received by the web server.

An issue which was faced was that the time received was not correct. Due to the time restrictions, it was chosen to hardcode the values in the code which releases the food from the food storage to perform the testing (Figure 13).

```
if ((t.hour == 22) && (t.min == 54) && t.sec >= 0 && t.sec <= 10) {
  servo.attach(8);

  for(angle = 120; angle > 80; angle--) {
    servo.write(angle);
    delay(15);
  }
  delay(15);
  for(angle = 100; angle < 120; angle++) {
    servo.write(angle);
    delay(15);
  }
}
```

Figure 13: Main Algorithm

Finally, the time logged by the RTC was made printable in the Serial Monitor to check against the chosen time and test whether the food is released at the right time.

5.3 Issues

The main issue met with the implementation was to receive the time value from the web page to the web server as mentioned in 5.2.4. For example, the time which was picked was 15:00:00, however, by debugging the time value received it was noted to be -16 (Figure 14).

```
03:52:59.365 -> Date : 13/5/2022  Hour : 3:52.20
03:53:10.378 -> 0
03:53:10.378 -> 12      -16
03:53:13.410 -> HTTP/1.1
03:53:13.410 -> Host: 192.168.43AT+CIPCLOSE=0
03:53:13.458 -> 0,CLOSED
03:53:13.458 ->
03:53:13.458 -> OK
03:53:13.458 -> Date : 13/5/2022  Hour : 3:52.34
```

Figure 14: Issue receiving time value

5.4 Testing

The type of testing performed for the application was user testing based on the false positive rate of the application mistakenly releasing feed at wrong time. This involved the user testing the application and giving feedback. As users were not able to fully test the web page, they were provided with instructions on how to test the application from the web server. Overall, feedback described that the components performed as expected, with the only difference being that food was always released as the web page was not fully functional (see algorithm in 4.3.2).

6. Conclusion and Evaluation

6.1 Design Summary

Overall, this paper was partially successful in designing an Offline-first Automatic Pet Feeder IoT application which works offline. As noted from the Testing section, the pet feeder successfully delivered food at the example specified times. However, the time value was not successfully received by the Arduino web server, which resulted in the application to work with hardcoded time values.

As a result, the application was successfully tested by following the logic set in the Design section and the results obtained were analyzed. In conclusion, it is recommended for the developer to familiarize themselves earlier on with the tools required for the development of this solution as this would be an advantage to the implementation of the application.

6.2 Future Work Recommendations

Since more functionalities will be likely added to the application in the future, the code will become larger, and more memory will be needed. Due to the current memory limitations of the Arduino Elegoo Uno (32 kB flash memory), it is recommended to change the main board to the Arduino Mega R3 (256kB flash memory).

In time to come, it is likely that the food container will be replaced to match the feeding needs of the pets it is designed for. Consequently, the SG90 Micro-servo motor can be replaced by the MG995 model which will provide faster speed and performance against the payload.

As for the non-hardware components, they can be replaced by materials which will make the pet feeder more sturdy and secure. Consequently, the pet food will be harder to access by the pets and it will be ensured that the pet's diet is kept.

Bibliography

[1] Pets 'go hungry' after smart feeder goes offline. (2020). BBC. https://www.bbc.co.uk/news/technology-51628795

[2] P. Pedamkar. (2022). Arduino Leonardo vs Uno. https://www.educba.com/arduino-leonardo-vs-uno/

[3] P. Pedamkar. (2022). Arduino Mega vs Uno. https://www.educba.com/arduino-mega-vs-uno/

[4] P. Rodan. (2018). Arduino library for DS3231 RTC. https://github.com/rodan/ds3231

[5] Rose, K., Eldridge, S., & Chapin, L. (2015). The internet of things: An overview. The internet society (ISOC), 80, 1-50.

[6] Ma, J. (2014, June). Internet-of-Things: Technology evolution and challenges. In 2014 IEEE MTT-S International Microwave Symposium (IMS2014) p. 1-4. IEEE. https://doi.org/10.1109/MWSYM.2014.6848429

[7] Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. International journal of communication systems, 25(9), https://doi.org/10.1002/dac.2417

[8] Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. International Advanced Research Journal in Science, Engineering and Technology, 5(1), 41-44. https://doi.org/10.17148/IARJSET.2018.517

[9] Adriansyah, A., Wibowo, M. A., & Ihsanto, E. (2016). Design of Pet Feeder Using Web Server as Internet of Things Application. In International Conference on Electrical Engineering and Informatics, p. 151-156.

[10] C. Own, H. Shin and C. Teng. (2013). The Study and Application of the IoT in Pet Systems, Advances in Internet of Things, Vol. 3 No. 1, p. 1-8. https://doi.org/10.4236/ait.2013.31001.

[11] Birha, P., Ingle, R., Tajne, S., Mule, P., Pandey, A., Kukekar, S., & Kadu, A. (2022). Design and Development of IoT Based Pet Feeder (No. 7877). EasyChair.

[12] Chen, Y., & Elshakankiri, M. (2020, April). Implementation of an IoT based pet care system. In 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC) (pp. 256-262). IEEE. https://doi.org/10.1109/FMEC49853.2020.9144910