Project

Pet Food Dispenser

Dragan Iulia-Andreea

1. Introduction

The purpose of this project was to design and implement a Pet Food Dispenser with the help of an Arduino development board. It is a perfect project for beginners in the field, who not only wish to develop their skills, but also to create something interesting and practical for everyday life.

Several versions of this project have already been made and they are available to study, but the main goal of this assignment was to learn how to use and link several components and, as a result, to create something from scratch.

Personally, I wanted to build something that can be both enjoyable and useful. While the pet food dispenser can function with the purpose that its name implies, it can also distribute various other types of foods, with the condition that they are of rather small dimensions.

2. Bibliographic research

The project was designed and implemented by researching various other similar projects, from the hardware components to software features, to the physical container / dispenser. As a result, I have decided to develop a rather well-known project idea in a way as original as possible.

While the microcontroller is an original Arduino UNO, by overlooking the cost of that component and that of the physical structure (the container and the support), the entire project added up to approximately 80RON. In terms of power consumption, the controller is connected to a wall socket through a 9V power source. The power generated is enough for the circuit to be able to run smoothly.

3. Design and implementation

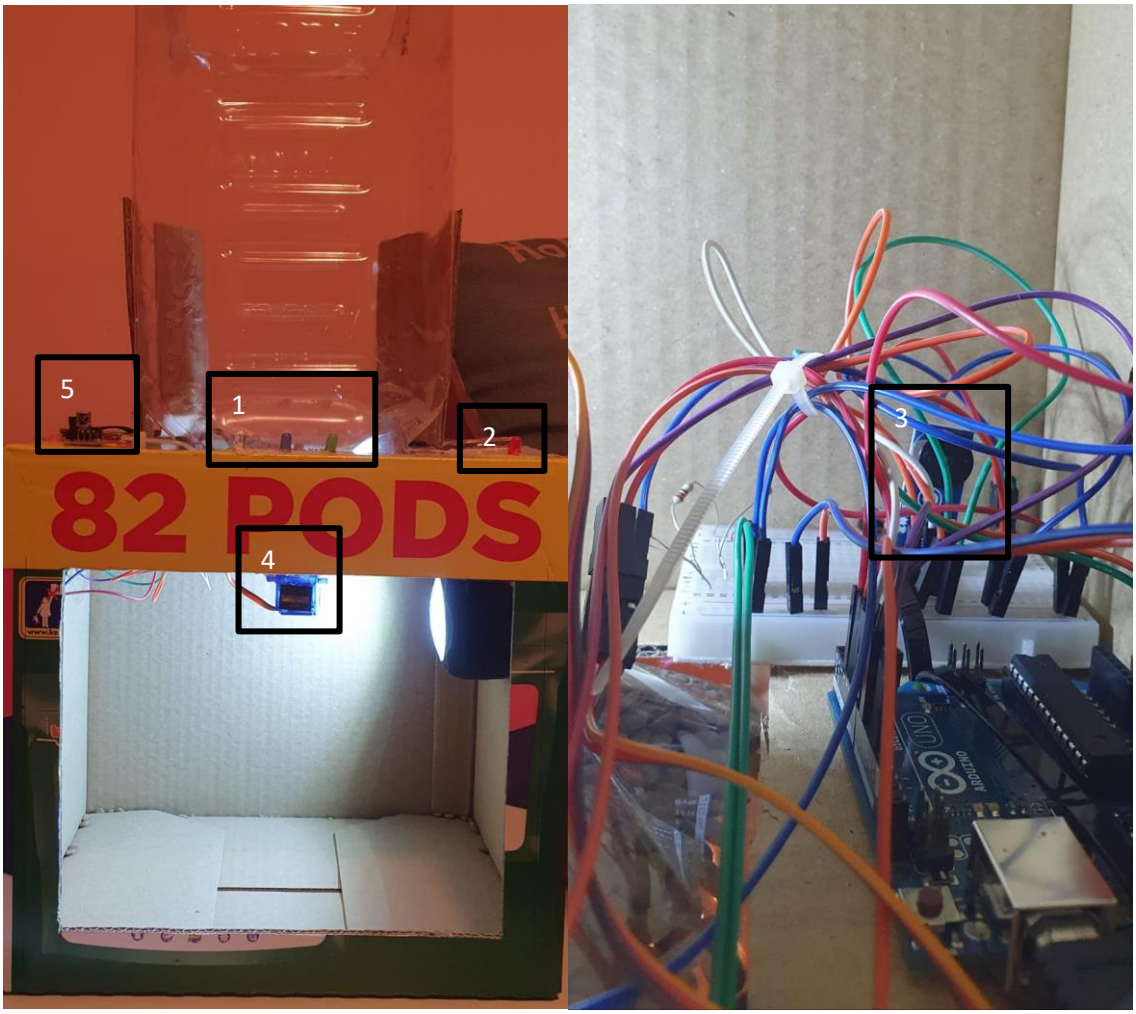
The circuit was designed to be as condensed as possible in order to have the possibility to place it inside the physical structure of the dispenser (i.e. the supporting box and the food container itself).

The pet food dispenser consists of three main components: the food dispenser, the supporting box, and the circuit. As functionality, there are three food distribution modes: small amount of food, medium amount of food, and large amount of food.

The user uses a remote controller (buttons 1, 2, 3) to specify the amount of food they wish to be poured from the container. Once the data is received and analyzed, the following three actions will take place:

- The led corresponding to the chosen operation will light up: white for small, blue for medium, green for large.
- The buzzer will be activated for various amounts of time, depending on the operation: once for small, twice for medium, thrice for large.
- The servomotor will make a number of full revolutions equal to the number of buzzer activations.

For example, when a large amount of food has been instructed to be poured, the green led will light up, the buzzer will activate three times, and the servomotor will execute three full revolutions, in order to allow a larger quantity of food to be distributed.

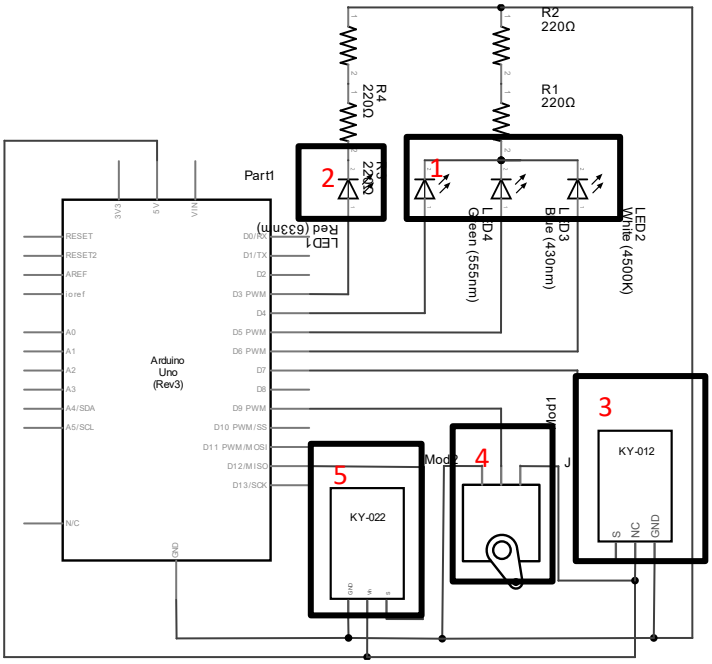


3.1. Hardware

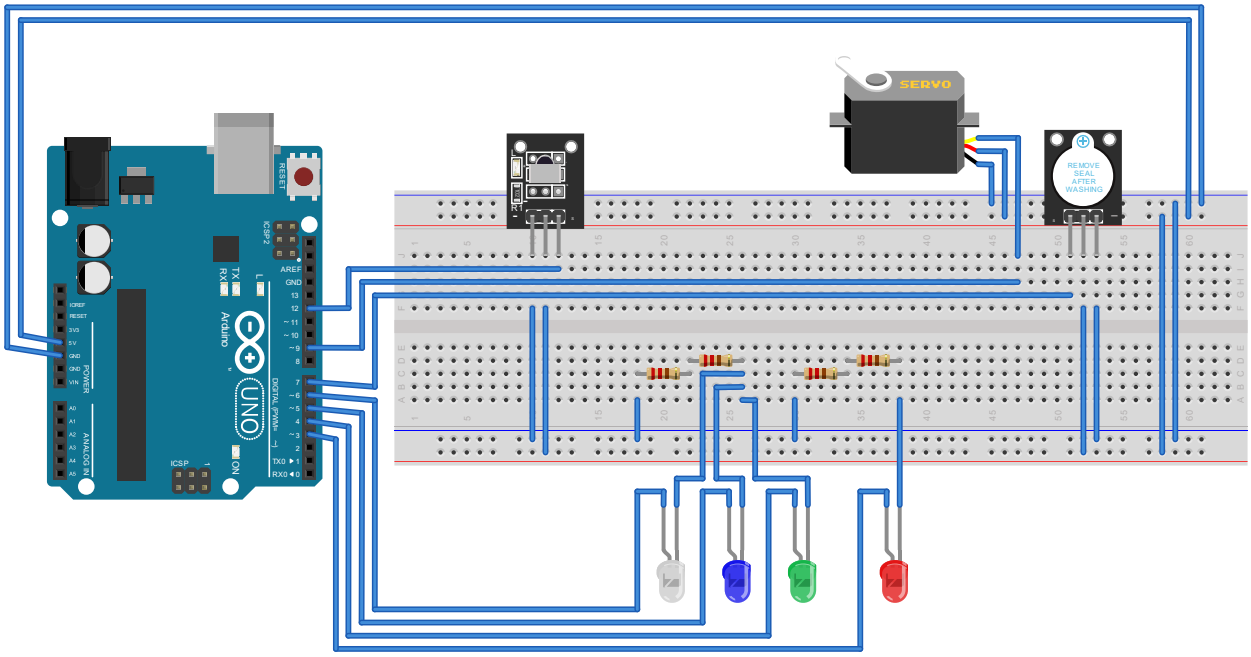
In terms of hardware components, the project is quite simple. For the microcontroller part, I used an Arduino UNO development board, as it was sufficient for the amount of components which were connected to it. Below is a list of the sensors and actuators used in the circuit:

- Sensors: Infrared sensor (KY-022) for receiving the input data from the paired remote controller
- Actuators:
  - Four leds, connected to 220Ω resistors in order to reduce the light intensity, used to signal the current state of the execution
  - One active buzzer to signal the execution of the program
  - One servomotor

Electrical schema:



Circuit diagram:



As it can be noticed in the circuit schema, each of the separate components are connected to one of the pins of the microcontroller.

- (1) - White / blue / green leds (pins 4-6) - In order to diminish the light intensity and to avoid burning the leds, they were all connected to two 220Ω resistors. These three leds help the user visualize the current action of the program, i.e. the three possible food distribution state. Since only one of the three leds can be lit up at the same time, they are sharing the same set of resistors, which helps to reduce the number of components used.
- (2) - Red led (pin 3) – This led is lit up throughout the entire process of execution. This means that for as long as the circuit is active, so is the led. For this reason, it is connected to its own set of resistors, independent of the other ones.
- (3) - Active buzzer (pin 7) – Similarly to the three leds, it is used to illustrate the current state, depending on the number of times it becomes active.
- (4) - Servomotor (pin 9) – Since the change of the position of the motor must be constant and fluid, it is connected to a PWM pin. The servomotor is actioned when the flap of the container must be opened so that the food can pour.
- (5) - Infrared sensor (pin 12) – The purpose of the sensor is to receive the data transmitted from the controller in order for the microcontroller to execute the desired operation.

3.2. Software

In case of the software, it is of importance to note that two external libraries were used: <Servo.h> for functions which help control the servomotor and <IRremote.h> in order to acquire and interpret data received from the infrared sensor.

3.2.1. Overview

As a general overview of the code, all pin and component set-ups were done in the setup() function. The main functionalities of the project are integrated within the loop() function. During each loop iteration, the program verifies if a signal was sent from the infrared controller through the decode() function.

If it has, the value received is stored in a flag variable, which will help decide which of the food distribution modes will be executed. If the value received is equal to one of the three accepted ones, the food2() function is executed with the corresponding parameter (1 for small, 2 for medium, 3 for large).

The food2() functions performs two main operations: activating the buzzer by calling the sing() function and rotating the servomotor. The amount parameter determines the number of times the buzzer will sing, as well as the number of full rotations (180° back and forth) the motor will execute. The sing() function performs in a similar way.

4. Testing and validation

4.1. Debugging process

The initial idea of the project involved using a Bluetooth module instead of an infrared sensor to select the desired operation. Still, the Bluetooth module introduced several connectivity issues and I have

not yet discovered their cause. In case of the infrared sensor, it requires no additional connections, as it is simply operated through a remote control, therefore I opted for this alternative.

Nevertheless, using the infrared sensor was also challenging at the beginning. Initially, both the servomotor and the sensor libraries were using the default timer, but that generated conflicts and an odd behavior of the program, in which the information was either lost, or not interpreted correctly. For this reason, the code had to be modified such that the sensor library was connected to TIMER1, hence the definition (#define IR_USE_TIMER1).

There was also the intention of using three different tones, rather than a different number of buzzer activations to signal the current state of the process. However, since the buzzer that was used is active, it only has the options of being turned on and off, i.e. it only accepts HIGH and LOW signals.

4.2. Final results

In the case of the servomotor, there was the possibility of only performing one rotation for each of the possible command, with the condition of varying the time intervals (delays) between the moment it has reached the final position and the one in which it begins to rotate back to its original state. However, the issue with that was that the amount of food that was dispensed was the same regardless of the value of the delay. The reason for that was the fact that the food was not distributed during the entire process, but only during the rotation process, as it was pushed towards the opening. For this reason, in order to distribute a varying amount of food, the most logical solution was to introduce multiple rotations with the same delay.

The project now performs as expected and the error percentage is low. The only type of errors that can appear are generated by the environment. This means that external infrared signals (that are not transmitted by the default input controller) can create slight disturbances and cause the sensor to intercept incorrect input values. However, if that is not the case, the sensor will read the correct value and execute the desired operation.

5. Conclusion

In conclusion, the project currently is in a satisfactory state. While both in terms of hardware and software it is quite simple, there were certain issues which prevented it from operating properly. However, the development and implementation processes mentioned above ensure that it would perform as intended with a low margin of error in its current state.

The pet food dispenser can be used as its name suggest, but it can distribute food of any kind, as long as its dimensions are small enough for it to fit through the opening. That includes cereals or, for example, rice.

As further enhancements, the circuit could be designed to distribute food at regular time intervals, in addition to the remote controller ability. As a result, the user would not be required to manually press a button each time they wish for food to be distributed. Values corresponding to other buttons on the controller could correspond to these additional features (for example, by pressing button 6, one could set the time interval between regular feedings to be 2 hours).