Automated Pet Feeder

Deepak Gadhve (23110110), Faayza Vora (23110109), Gadiparthi Abhinav (23110111)

***Abstract*—The aim of this project is to design an automatic pet feeder that will help pet owners maintain proper feeding schedules for their pets. In order to achieve this, we have used the DS3231 RTC module to schedule the meal time and a servo motor, which is connected to the lid of the container. When the servo motor rotates, the lid opens for 2 sec, and the food falls into the bowl. The procedure involves programming the Arduino board to interface with the servo motor and the RTC module.**

I. Aim

Design an automated pet feeder that ensures precise portion control and scheduling, providing convenience for pet owners and promoting healthy eating habits for pets.

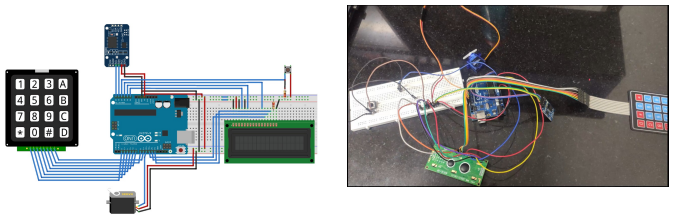
II. Materials used

1) Breadboard (To connect a single VCC pin of ardiuno to multiple devices
2) Jumper wires (For connections)
3) Ardiuno (For communication between devices)
4) DS3231 Real Time Clock(RTC) module (To keep a track of time)
5) 4×4 matrix keypad (To take the input for the feeding time from teb user)
6) 16×2 Liquid crystal display(LCD) (To display the time being set by the user)
7) Servo motor (To open and close the lid at the scheduled time)

III. Procedure

1) Start with setting up a common ground and 5V VCC connection on the breadboard, as there are multiple components.
2) Connect the 8 pins of the 4x4 numeric keypad to 8 digital pins on the Arduino Uno.
3) Connect the ground and VCC pins of the Real Time Clock(RTC) module to the common ground and 5V connection in the breadboard done in step 1.
4) Then, connect its SDA and SCL pins to 2 analog pins.
5) Connect the servo motor's ground and VCC pins to the common ground and 5V connection in the breadboard done in step 1, and connect the 3rd pin to a digital pin.
6) Connect the 16*2 LCD, which gives power through pin 1(ground) and pin 2(VCC). The backlight can be controlled through pins 15 and 16.
7) Connect the data pins DB4(pin 11) to the Analog pin and DB5(pin 12), DB6(pin 13), and DB7(pin 14) to digital pins through which data will be received and displayed in the LCD.
8) Leave the rest of the 4 data pins as they are.
9) Connect the contrast pin(pin 3) to the ground, RS pin(pin 4) to the analogue pin, R/W pin(pin 5) to the ground, and Enable pin(pin 6) to the analogue pin.
10) Then, connect the push button switch and its ground terminal to the Analog pin.

The circuit is as follows:



(a) Circuit Diagram (b) Circuit made in the lab

Fig. 1: Setting the feeding time

IV. Code Explanation

1) Adding Libraries to our code

```
#include <RTClib.h>
#include <Servo.h>
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <Wire.h>
```

Fig. 2: Code snippet

2) Mapping the 4x4 Numeric Keypad to the digital pins.

```
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = { 2,3, 4, 5};
byte colPins[COLS] = { 6,7, 8, 9 };

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

Fig. 3: Code snippet

3) Setting up the RTC module, servo motor, and the cursor on the LCD to (0,0)

```
RTC_DS3231 rtc; // Create an instance of the RTC_DS3231 class
Servo servoMotor;
LiquidCrystal lcd(A0, A1, A2, 11, 12, 13);

int feedingHour = 0;
int feedingMinute = 0;
bool feedingTimeSet = false;

void setup() {
  Wire.begin(); // Initialize I2C communication for A4 (SDA) and A5 (SCL)
  rtc.begin();
  lcd.begin(16, 2);
  servoMotor.attach(10);

  Serial.begin(9600);
  pinMode(A3, INPUT_PULLUP); // Push button pin

  lcd.setCursor(0, 0);
  lcd.print("Set feeding time");
}
```

Fig. 4: Code snippet

4) Setting Feeding Hour(24-hour clock format) using the 4x4 numeric keypad. Then, push the '#' button to set the hour and switch to setting the feeding minute.

```
void loop() {
  char key = keypad.getKey();

  if (key != NO_KEY) {
    if (key >= '0' && key <= '9') {
      lcd.print(key);
      if (!feedingTimeSet) {
        feedingHour = (feedingHour * 10 + (key - '0')) % 24;
        lcd.setCursor(0, 1);
        lcd.print("Hour:   ");
        lcd.setCursor(6, 1);
        lcd.print(feedingHour);
```

Fig. 5: Code snippet

5) Setting the feeding minute and pressing '#' completes the time setup.

```
void loop() {
  char key = keypad.getKey();

  if (key != NO_KEY) {
    if (key >= '0' && key <= '9') {
      lcd.print(key);
      if (!feedingTimeSet) {
        feedingHour = (feedingHour * 10 + (key - '0')) % 24;
        lcd.setCursor(0, 1);
        lcd.print("Hour:   ");
        lcd.setCursor(6, 1);
        lcd.print(feedingHour);
```

Fig. 6: Code snippet

6) The Servo motor rotates at an angle of 90 degrees at the set time and opens the flap under the container to dispense food.

```
      } else {
        feedingMinute = (feedingMinute * 10 + (key - '0')) % 60;
        lcd.setCursor(0, 1);
        lcd.print("Minute: ");
        lcd.setCursor(8, 1);
        lcd.print(feedingMinute);
      }
    } else if (key == '#') {
      if (!feedingTimeSet) {
        feedingTimeSet = true;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Set minute:");
      } else {
        feedingTimeSet = false;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Feeding time set");
      }
    }
```

Fig. 7: Code snippet

V. Discussion

We successfully designed an automatic pet feeder that serves a scheduled meal to the pet. The future prospects of our project are to configure it in such a way that multiple meals can be scheduled. Moreover, our original idea also included the stands to support the container and the servo motor with an attached lid for a clean finish.

VI. Images of the working model

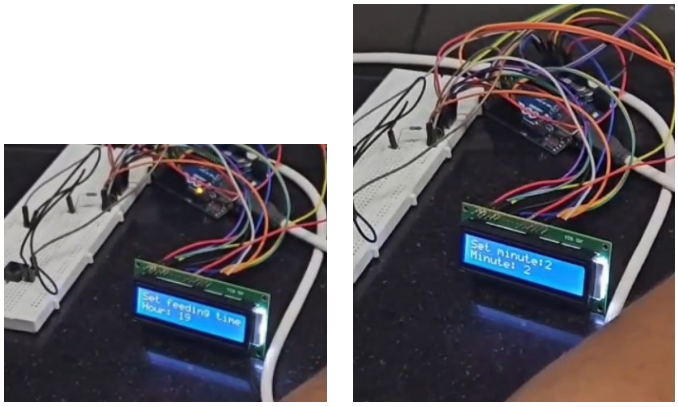


Fig. 8: Setting the feeding time

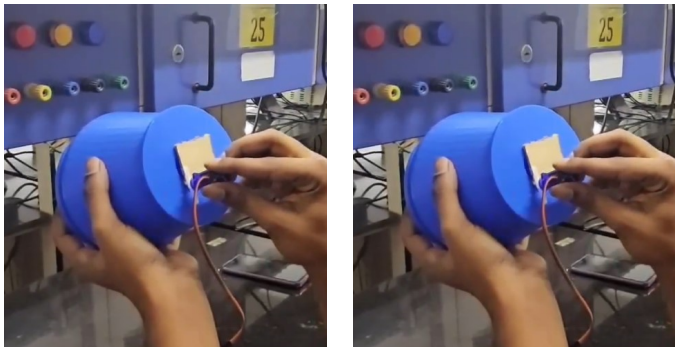


Fig. 9: Working model