'Tonito' - Pet Feeder

Member 1 - Jash Shah

Member 2 - Sebastian Tovar

Member 3 - Estefania Saucedo

Member 4 - Stephen Tinubu

Abstract:

The *TOÑITO* pet feeder is an innovative solution designed to enhance the convenience and reliability of feeding pets. This project introduces an automated pet feeding system that not only schedules daily meals for pets through a user-friendly interface but also ensures that pets are fed consistently, even in the absence of their owners. Our pet feeder automates your pet's feeding schedule with a user-friendly LCD screen for setting daily meal times. It dispenses food through a motorized door, ensuring your pet is fed consistently and notifying you when it's time to refill.

Overall Description Of Project Idea

The TOÑITO Automated Pet Feeder is a revolutionary pet care solution that redefines the feeding experience for pets and their owners. Engineered with precision and a touch of whimsy,this feeder stands out with its unique cat-shaped design, promising not only to feed your pet but also to become a cherished part of your home. Powered by four dedicated Arduino microcontrollers, TOÑITO offers unparalleled ease of use and reliability.

It allows pet owners to effortlessly program feeding schedules on a user-friendly LED screen, ensuring pets receive their meals on time, every time. The system intelligently manages portion sizes and monitors food levels, alerting owners when a refill is needed. Moreover, with interactive features like RFID-triggered sounds, it turns mealtime into a joyful event for your furry friend. TOÑITO is more than just a pet feeder; it's a smart companion that cares for your pet's nutritional needs while bringing joy and peace of mind to pet ownership

Final Project Design stating how Multiple Arduinos will be used

Arduino 1: The Command Interface

This Arduino is integral to user interaction, equipped with an LED display that provides real-time status updates on the pet feeder's operations. It is designed to receive specific scheduling commands from the user via increment and decrement buttons, which allow for setting and adjusting the feeding times in hours and minutes. After processing these inputs, Arduino 1 generates corresponding output signals to instruct Arduino 2 to actuate the feeding mechanism at the designated times.

Arduino 2: The Precision Dispenser

Acting upon instructions from Arduino 1, this microcontroller manages a servo motor that controls the opening of the food container's flap, thus regulating the release of pet food. It's a pivotal component in the physical dispensation of the feed, ensuring that meals are delivered reliably at the set schedule. Additionally, it interfaces with a force sensor that provides vital feedback on the quantity of dispensed food by weight, enabling precise portion control and preventing overfeeding.

Arduino 3: The Inventory Monitor

With an ultrasonic sensor connected, Arduino 3 performs continuous monitoring of the food level within the container. It measures the distance from the sensor to the surface of the food, allowing it to calculate how full or empty the container is. These measurements are sent as input to Arduino 1 for user information and also determine the output behavior of the LED lights to signal the need for refilling.

Arduino 4: The Interactive Beacon

This Arduino enhances the feeding experience by providing interactive feedback to the pet. It utilizes an RFID sensor to detect the presence of a tagged pet nearby and, in response, activates a speaker system to play selected sounds. These audio cues not only entertain but can also be used to train pets to recognize feeding times, supporting consistent feeding habits.

Final Plan for Use and Communication between the multiple Arduinos

I2C Communication Protocol: Given the nature of this project, we selected I2C as the communication protocol due to its capability to connect multiple devices (a master and multiple slaves) over a single bus. This is beneficial for a system like TOÑITO, where one Arduino needs to communicate with several others in a coordinated fashion.

Master-Slave Hierarchy: Arduino 1 will function as the master device. It's responsible for initiating communication with the slave Arduino (Arduino 2) and coordinating the overall operation.

Arduinos 3, and 4 will be configured as master-slave devices, each with a unique I2C address allowing the master to individually address and communicate with the slave.

Final Communication Flow:

From Master to Slaves: Arduino 1 will send commands to Arduino 2 to dispense food at scheduled times. Arduino 3 will also send instructions to Arduino 4 indicating sounds alert for availability

From Slaves to Master: Arduino 2 will acknowledge the commands and confirm the action of food dispensing. Arduino 3 will send the food level data, to Arduino 4 where respective sounds alert will be played for food availability

Data Handling:

1. **Data Integrity**: To ensure data integrity, each transmission will include a checksum or a form of error-checking to allow the detection of any corruption in the messages sent over the I2C bus.
2. **Error Handling:** In the event of communication errors or detected anomalies in data, the master will have a retry mechanism to request the information again, ensuring reliability in the system's performance.

Power Management: To maintain stable communication, all Arduinos will share a common or similar power source to prevent potential issues from voltage drops that could affect the I2C signals.

Synchronization: The master Arduino will manage a timing mechanism to ensure that all feeding-related activities are synchronized, avoiding conflicts or overlapping operations. Basically, we are double checking the inputs received from each arduino and also making sure that correct outputs are forwarded in the outgoing communication from each arduino.

Final List of Expected Inputs and Outputs

Arduino 1 : LED Screen

LED Screen - this will be the main connection between the user and the motor. The purpose of this LED screen is to display the status of the food container and as well receive input of time (Hour : Minutes) from the user. The LED Screen will also give output to the motor which will rotate the flap(Arduino 2).

Buttons - This is what the user will use to input Hour Minutes to the LED Screen. Everyday at this hour we will dispense the food.

Arduino 2 : Food Dispenser

3V/6V Motor - The main purpose of this motor is to move the flap. This flap will be the covering of the food container. As soon as this motor gets the command to dispense food (From Arduino 1) it will turn on and move the flap for certain seconds and then go back to the original state.

Ultrasonic Sensor or a Force Sensor - The purpose of this sensor is basically to double check how much food is left in the food bowl. If the food in the bowl is full then we don't need to dispense anything. Basically double checking before dispensing using the motor. The ultrasonic sensor can be used to measure the distance in the food bowl assuming we know its dimensions and then we can perform calculations and check using the updated distance in the bowl. This however makes it a little bit complex, so another option we are thinking of is having a pressure plate sensor in the bowl which can detect if there is food inside it based upon the changing pressure.

Arduino 3 : LED Lights

Ultrasonic sensor - This sensor's job is to measure how much food is left in the container. We are assuming the food container's dimensions are known. For example, let's say it's 30 cm in height. The ultrasonic sensor sends out waves and we record the output. We calculate the distance from where the waves detected the food, let's say distance is at 13 cm from the sensor. This would indicate that 17cm (30-13) of the container has something inside it. Using this we have predetermined levels such as if the height is less than 15 food is low and if it is above the food is high. We display whatever the status of the food is in the container.

LED Lights - Fairly simple, the purpose of this is to light green if the food is high, or blind red if the food is low. It will take input from the sensor.

Arduino 4 : Speaker

RFID Sensor with a RFID Tag - this tag would be worn by the pet. The sensor senses this tag whenever it's in close proximity. Also takes a RX-TX connection by Arduino 3 whenever the food is low and plays audio sound based on it. Also plays audio

Speaker - Speaker depending on the input given by the sensor will play a sound (meow, never gonna give you up, etc) Also takes a RX-TX connection by Arduino 3 whenever the food is low and plays audio sound based on it.

Original work

The original work here is to use the I2C Connections and serial connections between the arduinos. Also the use of ultrasonic sensors in the food container to   measure the level of food present in the container. We also have another ultrasonic sensor or a pressure plate sensor in the food bowl of the pet which tells us if the food is present in the bowl or not, information on which our dispensing motor's action will be based upon.

All of this is combined with a RFID Sensor and a tag which is kind of a fun interactive thing for the pets. The arduino could be programmed to play any audio actually, it could be the sound of an alarm going off saying that food is low or it could be just a meow sound to lure the cat towards it, you could use this sensor as an awareness system as well. You will know the cat is near the food because of the music coming from the speaker.

Moreover, this is all serial connection. In future use outside this project timeline we can also add bluetooth connection and give simple string output to an android/windows device saying High/Low depending upon the food level decided by the ultrasonic sensor.

LAB REPORT

Arduino 1

Description :

When powered on, the Arduino displays a welcoming message, "Welcome to TONITO". The LCD then prompts the user to set the feeding schedule by displaying the message "Food Countdown:" followed by the default time, which is initialized to 15:00 minutes

Four buttons are used to adjust the feeding time:

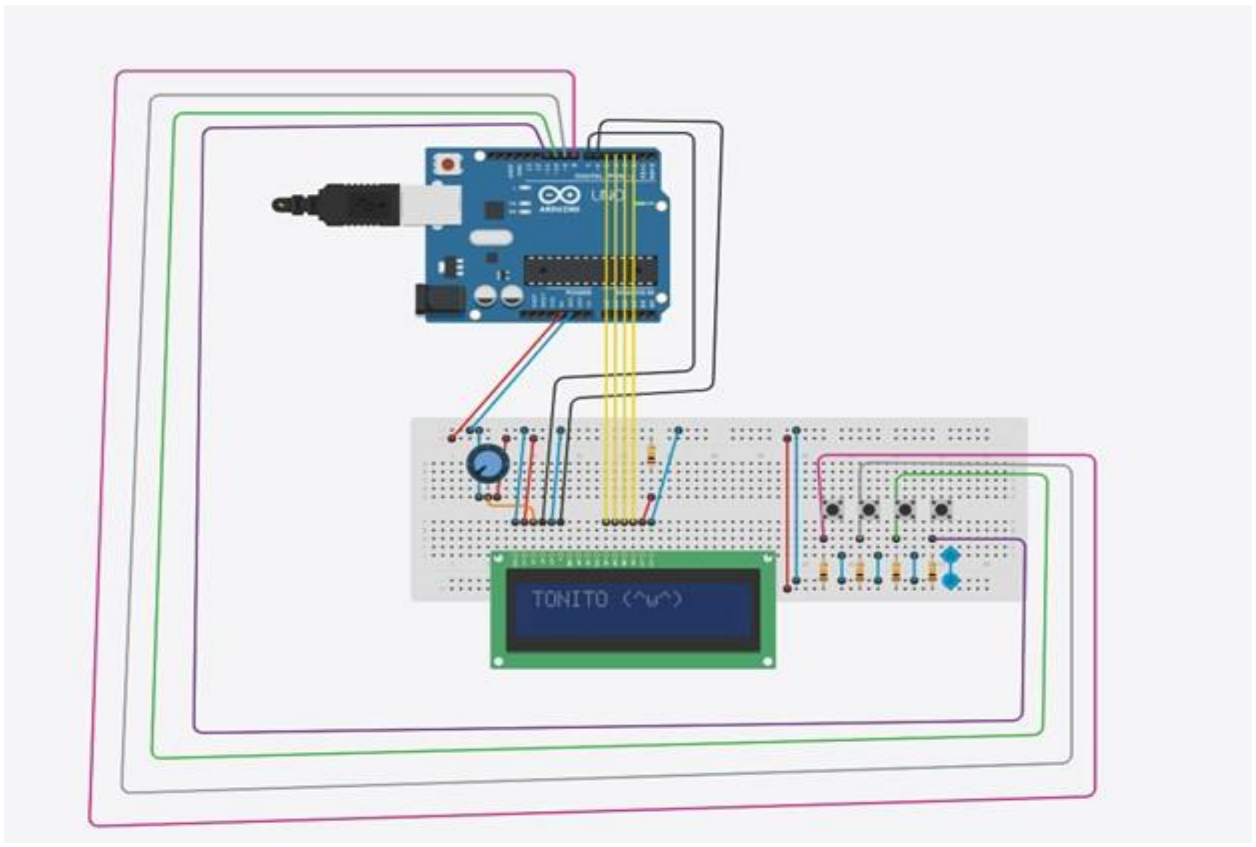
- ○ Two buttons are assigned to increment and decrement the timer countdown for food dispensing. Each increment/ decrement changes the timer countdown for 15 minutes
- ○ The other two buttons are used to start and reset the timer respectively

- Pressing the hour or minute buttons updates the time displayed on the LCD in real-time and sends the updated time through the serial connection in the format "FOOD COUNTDOWN HH:MM:SS". This could be received by another Arduino configured to listen on the serial port for such a message.

Debounce logic is implemented to avoid registering multiple button presses when only a single press is intended. After a button press, the code pauses for 100 milliseconds to allow for the physical button's contact bounce to settle.

Components needed:

1. 1x Arduino Uno R3
2. 1x LCD 16x2
3. 5x 10 kOhm Resistor
4. 1x 10 kOhm Potentiometer
5. 4x PushButton **Diagram:**



Connection on Arduino Uno:

LCD: rs = 13, en = 12 , d4 = 11 , d5 = 10 , d6 = 9 , d7 = 8

Push buttons: 4,5,6,7

btnIncrementHourPin = 4

btnDecrementHourPin = 5

btnIncrementMinutePin = 6

btnDecrementMinutePin = 7

Code:

```
Abstract:
```

```
The TOÑITO pet feeder is an innovative solution designed to enhance the
convenience and reliability of feeding pets.

This project introduces an automated pet feeding system that not only schedules
daily meals for pets through a user-friendly interface but also ensures that pets
are fed consistently, even in the absence of their owners. Our pet feeder
automates your pet's feeding schedule with a user-friendly

LCD screen for setting daily meal times. It dispenses food through a motorized
door, ensuring your pet is fed consistently and notifying you when it's time to
refill.

*/



#include <LiquidCrystal.h>

#include <stdlib.h> // Include for random functionality



// Initialize the LCD library with the numbers of the interface pins

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);



// Pin setup for the buttons
```

```
const int buttonPins[] = {4, 5, 6, 7}; // Pin assignments: Increment, Decrease,
Start/Pause, Reset (Left to right)
const int numButtons = 4;


int buttonState[numButtons];

int lastButtonState[numButtons] = {HIGH, HIGH, HIGH, HIGH};

String clearScr = "                "; // 16 spaces



// Timing and state management
```

```
unsigned long lastDebounceTime[numButtons] = {0, 0, 0, 0};
unsigned long debounceDelay = 100; // milliseconds


unsigned long countdownTime = 900; // 15 minutes in seconds

unsigned long lastInteractionTime; // Mainly used for transitioning between
active and idle states

unsigned long lastScrollTime = 0;

int scrollDelay = 700; // milliseconds for idle scroll

boolean isCounting = false;

String currentState = "active"; // Start with initial state as "active"



// Idle state animation const
String idleMessages[] = {


    "Cats can make over 100 vocal sounds.",

    "Cats sleep 70% of their lives.",

    "A group of cats is called a clowder.",

    "Cats have five toes on front paws."

};




int idleMessageIndex = 0;
int idlePosition = 0;
```

```




void setup() {
  lcd.begin(16, 2); // set up the LCD's number of columns and rows

  for (int i = 0; i < numButtons; i++) {
    pinMode(buttonPins[i], INPUT_PULLUP);

  }

  lastInteractionTime = millis();

  displayWelcomeMessage();

}



void loop() { int idleTime = 10000; // 10 seconds; The screen will go to idle
  state after 10

seconds of inactivity (No button press)

  boolean anyButtonPressed = false;



  for (int i = 0; i < numButtons; i++) {
    int reading = digitalRead(buttonPins[i]);



    if (reading != lastButtonState[i]) {
      lastDebounceTime[i] = millis();

    }



    if ((millis() - lastDebounceTime[i]) > debounceDelay) {
      if (reading != buttonState[i]) {

        buttonState[i] = reading;


        if (buttonState[i] == LOW) {

          anyButtonPressed = true;
          lastInteractionTime = millis();
          if (currentState.equals("idle")) {

            currentState = "active"; // Exit idle state on any button press

            displayTime();

          } else {

            handleButtonPress(i);

          }

        }

      }

    } lastButtonState[i] =
    reading;


  }



  // Check to enter idle state regardless of counting state if
  (!anyButtonPressed && millis() - lastInteractionTime > idleTime &&

currentState.equals("active")) {

    enterIdleState();

  }



  // Scroll idle messages in idle state and manage scrolling timing

  if (currentState.equals("idle") && millis() - lastScrollTime > scrollDelay) {
    scrollIdleMessage();


    lastScrollTime = millis();

  }
```

```
  // Update countdown independently of the current state, but only update display
```

```
if in active state
  if (isCounting) {


    updateCountdown();

    if (currentState.equals("active")) {

      displayTime(); // Refresh time display only if not in idle state

    }

  }




  delay(100);
}




void updateCountdown() {
  static unsigned long lastUpdateTime = 0;

  if (millis() - lastUpdateTime >= 1000) {

    lastUpdateTime = millis();
    if (countdownTime > 0) {


      countdownTime--;

    } else { isCounting = false; // Stop countdown when time
      reaches zero


      lcd.clear();

      lcd.setCursor(0, 0);

      lcd.print("Time's Up!");

    }

  }

}
```

```
void displayWelcomeMessage() {

  String message = "Welcome 2 TONITO";

  String displayText = "";

  int length = message.length();



  // Fill the displayText with random alphanumeric characters
  for (int i = 0; i < length; i++) {

    displayText += (message[i] == ' ') ? ' ' : char(random(33, 126)); // Random char between '0' and 'z'

  }



  lcd.clear();
  lcd.setCursor(0, 0);




  //Deciphering animations; Aesthetic purposes

  // Gradually change random characters to the correct ones
  for (int i = 0; i < length; i++) {

    delay(150); // Delay between transitions

    displayText[i] = message[i];
    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(displayText);

  }


  delay(2000); // Hold the final message for 2 seconds
  currentState = "active"; displayTime(); // Ensure display is
  refreshed from welcome message
```

```
}



void handleButtonPress(int buttonIndex) {

  if (currentState.equals("idle")) {


    displayTime(); // Ensure display is refreshed from idle state


  }




  switch (buttonIndex) {


    case 0: // Increase timer if
      (countdownTime < 43200) {



        countdownTime += 900;


      }
      break;



    case 1: // Decrease timer


      if (countdownTime > 900) {


        countdownTime -= 900;

      }
      break;


    case 2: // Start/Pause isCounting = !isCounting; //
      Toggle counting state



      break;


    case 3: // Reset timer


      countdownTime = 900; isCounting = false; //
      Stop counting when reset



      break;


  }
```

```
  displayTime(); // Refresh the display every time a button is pressed and
handled
}




void displayTime() {

  lcd.clear(); lcd.setCursor(1, 0); // Center
  the top text


  lcd.print("Food Countdown");

  lcd.setCursor(4, 1); // Position timer at the bottom center int
  seconds = countdownTime % 60;


  int minutes = (countdownTime / 60) % 60;
  int hours = (countdownTime / 3600);


  lcd.print((hours < 10 ? "0" : "") + String(hours) + ":");

  lcd.print((minutes < 10 ? "0" : "") + String(minutes) + ":");

  lcd.print(seconds < 10 ? "0" : "");
  lcd.print(seconds);


}



void enterIdleState() {
  lcd.clear();

  currentState = "idle";

  scrollIdleMessage(); // Start with the first message immediately

}



void scrollIdleMessage() {
  lcd.clear();
```

```
  lcd.setCursor(0, 0);

  String displayText = idleMessages[idleMessageIndex].substring(idlePosition) +
clearScr;

  if (displayText.length() < 16) {

    displayText += idleMessages[idleMessageIndex].substring(0, 16 -
displayText.length());

  }

  lcd.print(displayText);
  idlePosition++;
  if (idlePosition >= idleMessages[idleMessageIndex].length() + 1) {

    idlePosition = 0; idleMessageIndex = (idleMessageIndex + 1) %
    (sizeof(idleMessages) /

sizeof(idleMessages[0]));

  }

}
```

ARDUINO 2

Project Objective:

To construct and oversee a comprehensive pet feeding system that not only schedules and dispenses food accurately but also measures the quantity dispensed. This system employs Arduino 2 to control a servo motor for opening the food container and to process data from a force sensor for precise portion management. The ultimate goal is efficient, reliable pet feeding with controlled portions to avoid overfeeding.

Components Required:

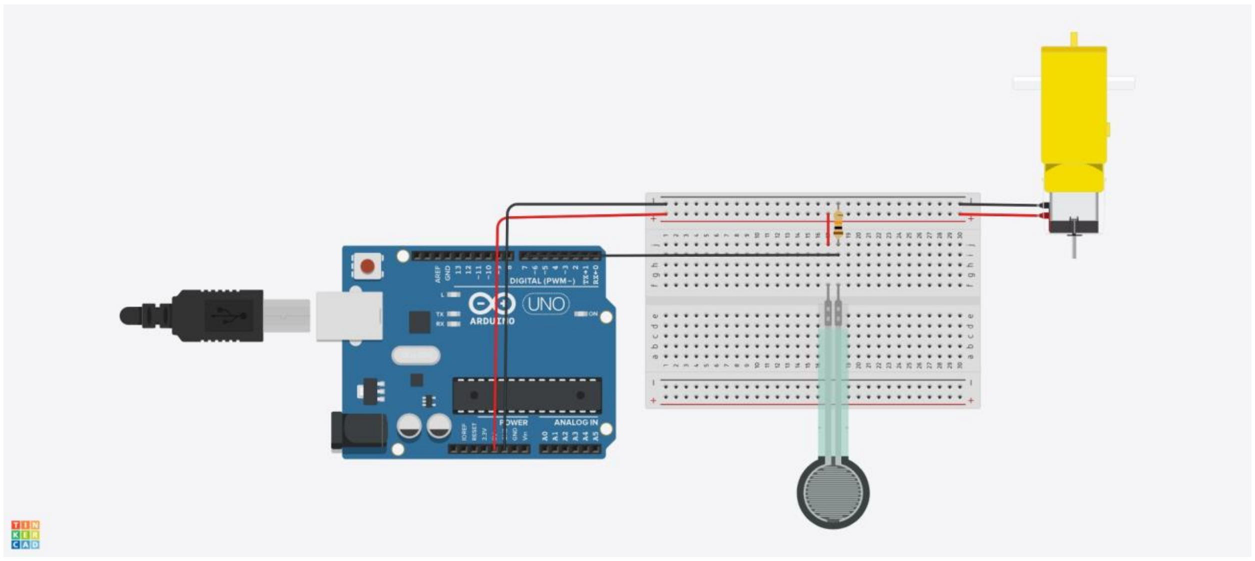
1. Arduino Uno (Arduino 2) - Acts as the central control unit for managing the feeding schedule and servo motor operation.
2. Servo Motor - To control the opening and closing of the food container's flap, allowing the food to be dispensed.
3. Force Sensor - For measuring the weight of the dispensed food, ensuring accurate portions.
4. Connecting Wires - For establishing electrical connections between components.
5. Breadboard - For assembling a prototype of the circuit.

6. Power Supply - To provide power to Arduino, the servo motor, and the force sensor.

System Design:

- The servo motor is connected to Arduino 2 and is responsible for opening the flap of the food container according to the feeding schedule.
- The force sensor, also connected to Arduino 2, is placed under the food bowl to measure the weight of the food dispensed. This feedback allows for precise control over food portions, ensuring pets receive the correct amount of food.
- Arduino 2 is programmed to control the servo motor's operation based on the feeding schedule, while simultaneously reading the force sensor's data to monitor and adjust the amount of food dispensed.

Diagram:



Implementation Steps:

Connect the servo motor to a designated digital pin on Arduino 2 for control signals.

Place the force sensor under the food bowl and connect it to Arduino 2 for weight measurement. Use connecting wires to establish proper connections between the servo motor, force sensor, and Arduino on the breadboard. Ensure the power supply is correctly connected to power Arduino, the servo motor, and the force sensor.

Final code:

```
Abstract:
The TOÑITO pet feeder is an innovative solution designed to enhance the
convenience and reliability of feeding pets.
This project introduces an automated pet feeding system that not only schedules
daily meals for pets through a user-friendly interface but also ensures that pets
are fed consistently, even in the absence of their owners. Our pet feeder
automates your pet's feeding schedule with a user-friendly
LCD screen for setting daily meal times. It dispenses food through a motorized
door, ensuring your pet is fed consistently and notifying you when it's time to
refill.
*/
// force Sensor - A0 and resistor connected to GND and other end to power
// servo load - 10 (orange cable)
// servo power- power (red cable)
// servo ground - GND (brown cable)
//rx -tx
//tx -rx


#include <Servo.h>



////~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
// Force sensor logic stuff
bool lowFood = false;
int forceSensorPin= A0;
int forseSensorReading;


//motor stuff
Servo myServo;
const int servoPin = 10;


//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


void setup(void) {


  Serial.begin(9600);
  myServo.attach(servoPin);
  myServo.write(55); //<--- so it starts closed


}


//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void loop(void) {


  readForceSensor(); //<--- reads first so we can use it in loop


  if (Serial.available() > 0){
    char recv = Serial.read(); //<--- has to get a T from arduino 1


    if (lowFood && recv == 'T'){ //<-- when arduino 1 says so and theres less
```

```
than 10 g of food

        releaseFood();

    }
   }

}
```

```

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~
void releaseFood() {

  myServo.write(130); //<--opens lid

  delay(500); //<--- waits a little bit so food comes out

  myServo.write(55); //<---closes lid

  delay(3000);
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~`
// this function weights the food and when it reaches 10, changes the flag to
false
bool readForceSensor() {
```

```
   forseSensorReading = analogRead(forceSensorPin); //<---- just reading from
force sensor


  //~~~~~~~~~~~~~ printing how much food we have on bowl to serial monitor
  Serial.println("Analog reading = ");


  Serial.println(forseSensorReading);
  //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


  if(forseSensorReading <10 ){ //<-- lid should close if reaches 10
    lowFood = true;
  }
  else{
    lowFood = false; ///<---- stays open if not
  }


  return lowFood;


}
```

Expected Outcome:

Upon successful implementation, this automated pet feeding system will enable pets to be fed precise portions at scheduled times without manual intervention. The system's ability to control the opening of the food container and to measure the dispensed food's weight ensures a consistent, accurate feeding process. This automation aids in maintaining the pet's health by preventing overfeeding and ensures that pets are fed regularly, even in the absence of the owner.

ARDUINO 3:

Project Objective:

To design and implement a system that continuously monitors the food level within a storage container using an ultrasonic sensor and Arduino microcontrollers to inform users of the current food level and signal when a refill is needed.

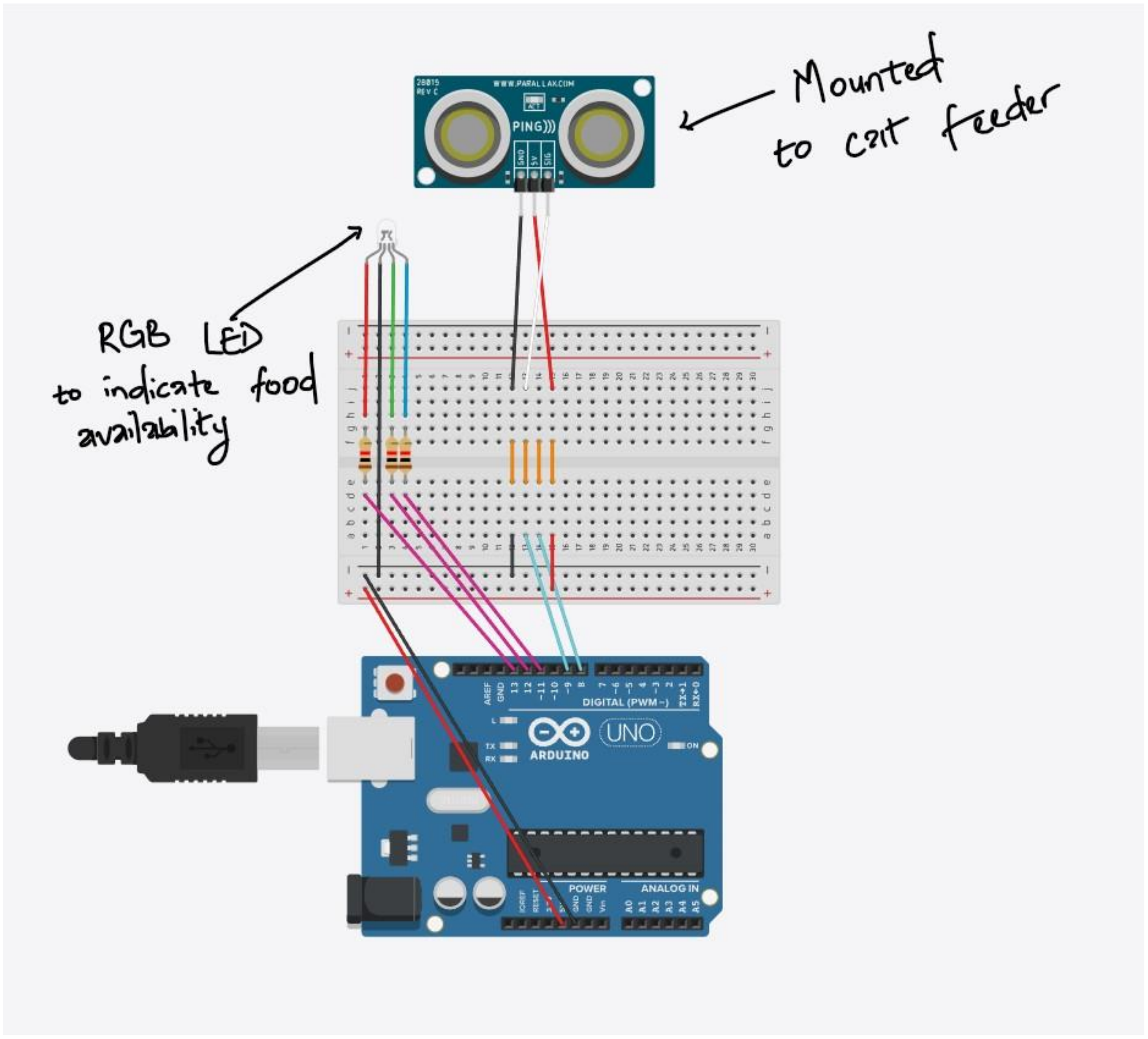
Components Required:

1. Arduino Uno (Arduino 3) - for monitoring inventory levels.
2. HC-SR04 Ultrasonic Sensor - to measure the distance to the food surface.
3. RGB LED Light - for visual indicators of food levels.
4. Connecting Wires - for establishing connections between components.
5. Breadboard - for prototyping the circuit.
6. Power Supply - to power the Arduinos and the sensor.
7. Resistor (220 ohms) - to protect the LEDs.

Description:

- The ultrasonic sensor will be mounted at the top inside of the container, facing downwards towards the food. It will be connected to Arduino 3. - Arduino 3 will be programmed to trigger the ultrasonic sensor at regular intervals to send out ultrasonic waves.
- When the waves hit the surface of the food, they will bounce back to the sensor, which will then calculate the time taken for the round trip. - Using the speed of sound and the time measurement, Arduino 3 will calculate the distance from the sensor to the food surface.
- Arduino 3 will then determine the level of food in the container based on pre-set thresholds (e.g., full, half, low, empty).
- This data will be sent from Arduino 3 to the RGB LED light to process and control the LED light:

- Green LED: Container is full.
- Purple LED: Container is at half capacity.
- Red LED: Container is low and needs refilling.



CONNECTION:

Mount the ultrasonic sensor at the top inside of the container, with the sensor facing downwards towards the food. Connect the VCC and GND pins of the HC-SR04 ultrasonic sensor to the 5V and GND on the Arduino, respectively. Connect the TRIG pin of the ultrasonic sensor to digital pin 8 on the Arduino. Connect the ECHO pin of the ultrasonic sensor to digital pin 9 on the Arduino. Connect the anodes of the RGB LED to digital pins 11 (Red), 12 (Green), and 13 (Blue) on the Arduino through 220-ohm resistors to protect them. Connect the cathode of the RGB LED to the GND on the Arduino.

Final code:

```
Abstract:
The TOÑITO pet feeder is an innovative solution designed to enhance the
convenience and reliability of feeding pets.
This project introduces an automated pet feeding system that not only schedules
daily meals for pets through a user-friendly interface but also ensures that pets
are fed consistently, even in the absence of their owners. Our pet feeder
automates your pet's feeding schedule with a user-friendly
LCD screen for setting daily meal times. It dispenses food through a motorized
door, ensuring your pet is fed consistently and notifying you when it's time to
refill.
*/


// ARDUINO 3: ULTRA SONIC SENSOR


// Define constants for ultrasonic sensor pins
const int TRIGGER_PIN = 8;
const int ECHO_PIN = 9;


// Define constants for RGB LED pins
const int RED_LED_PIN = 13;
const int GREEN_LED_PIN = 12;
const int BLUE_LED_PIN = 11;


// Define the speed of sound in cm/us
const float SPEED_OF_SOUND = 0.0343;
```

```
// Define thresholds for the food levels in cm
const int FULL_THRESHOLD = 8; // arbitrary number
const int HALF_THRESHOLD = 12; // arbitrary number
const int LOW_THRESHOLD = 20; // arbitrary number


void setup() {
  // Initialize ultrasonic sensor pins
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);


  // Initialize RGB LED pins
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  Serial.begin(9600);
}


void loop() {
  // Send ultrasonic pulse
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);


  // Read the echo time (round trip)
  long duration = pulseIn(ECHO_PIN, HIGH);


  // Calculate the distance based on the speed of sound
  float distance = duration * SPEED_OF_SOUND / 2;
  Serial.println(distance);
```

```
// Determine the food level and set the LED color
if (distance <= FULL_THRESHOLD) {
  // Green LED: Container is full
  setColor(0, 255, 0); // Green
  Serial.println("GREEN INDICATED");


} else if (distance > FULL_THRESHOLD && distance <= HALF_THRESHOLD) {
    // Yellow LED: Container is at half capacity
    setColor(255, 0, 255); // Purple
    Serial.println("PURPLE INDICATED");

  } else if (distance > HALF_THRESHOLD && distance <= LOW_THRESHOLD) {
    // Red LED: Container is low
    setColor(255, 0, 0); // Red
    Serial.println("RED INDICATED");
    sendMsg();

  } else {
    // No Light: Container is empty or sensor out of range
    setColor(0, 0, 0); // Off
    sendMsg();
  }

  // Delay for a short period before measuring again
  delay(500);
}

void setColor(int R, int G, int B) {
  analogWrite(RED_LED_PIN,   R);
  analogWrite(GREEN_LED_PIN, G);
  analogWrite(BLUE_LED_PIN, B);
}

void sendMsg(){
  Serial.write('T');
}
```

Expected Outcome:

On completion, the system will automatically monitor the level of food within the container and inform the user of the current level via LED indicators. This setup will facilitate efficient inventory management and timely refilling, reducing the risk of running out of essential supplies.

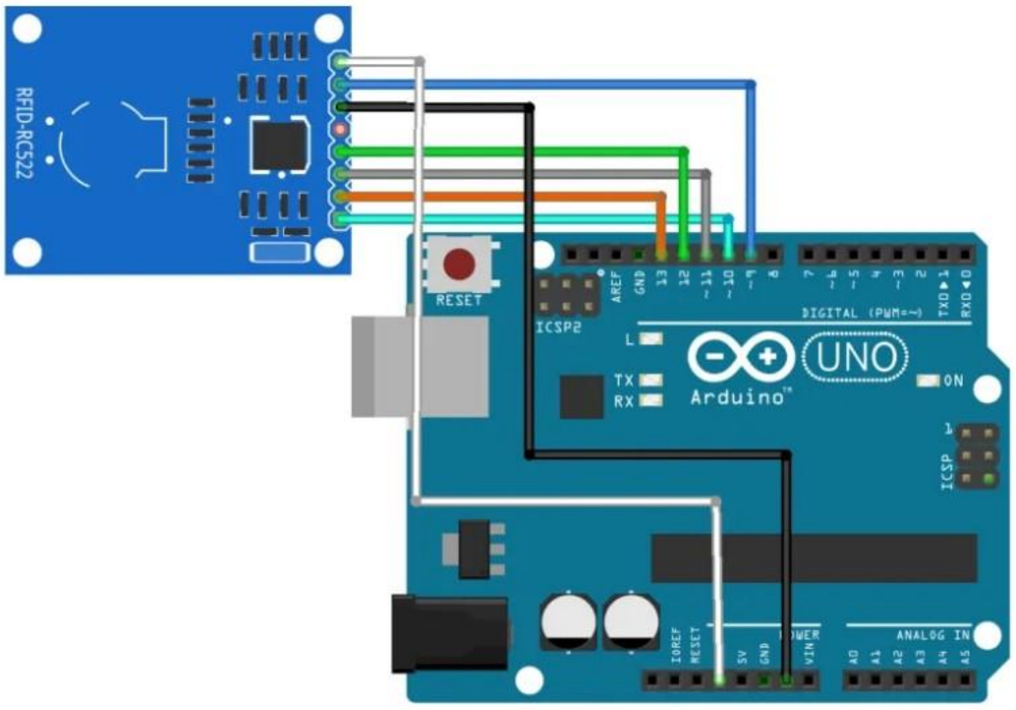
ARDUINO 4 :

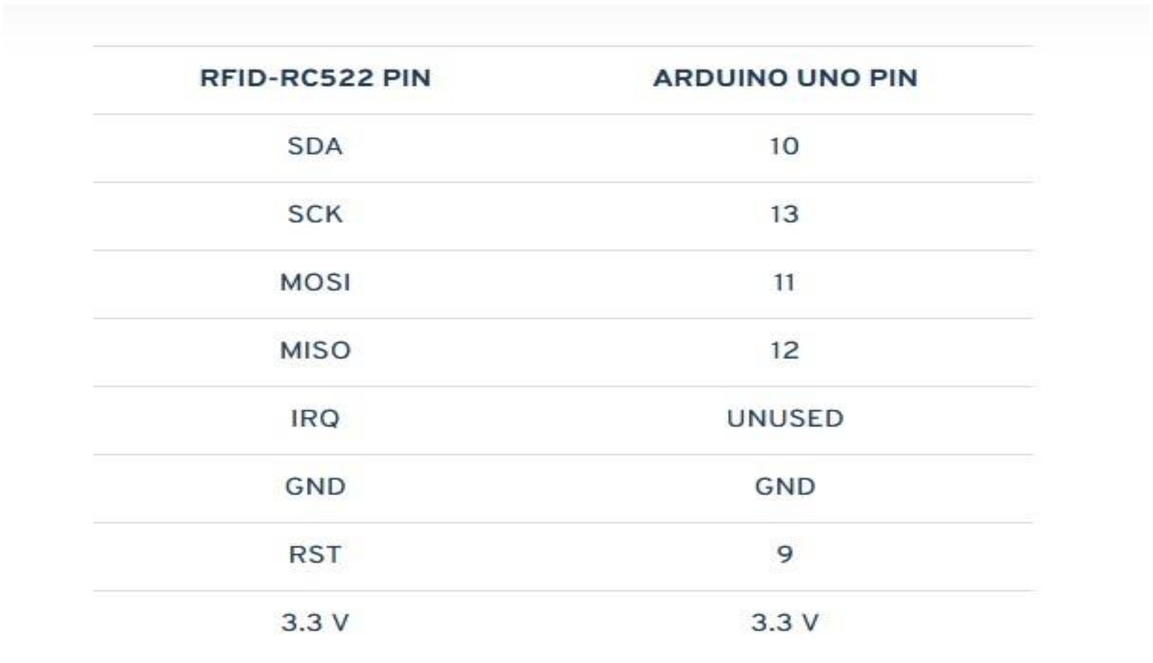
Description :

This Arduino code plays a melody using a buzzer and reads RFID tags with an MFRC522 module. It sends the RFID tag information to the Serial monitor when a tag is detected. The buzzer is connected to pin 7, and the RFID reader uses pins 9 and 10 for reset and communication, respectively. The sketch alternates between playing music and scanning for RFID tags in its main loop. This is still in progress, next step would be to ensure tests and debounce between speaker sound playing and the rfid scanning and also to configure the duration of sounds and amount of sounds. Currently it plays Star Wars music. Reference Arduino4 Code File

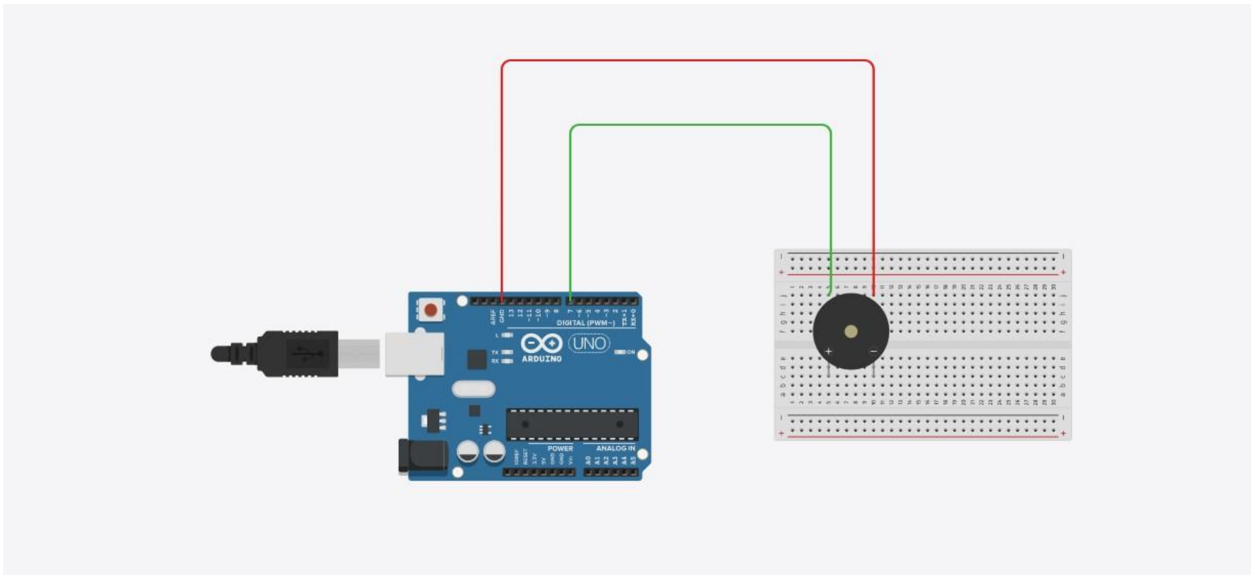
Components Required : RFID RC522 with Fob, Piezo buzzer, Hookup Wires

Circuit Diagram





RFID-RC522 PIN	ARDUINO UNO PIN
SDA	10
SCK	13
MOSI	11
MISO	12
IRQ	UNUSED
GND	GND
RST	9
3.3 V	3.3 V



Buzzer pin 1	GND
Buzzer pin 2	Arduino Uno Pin 8

TOÑITO Automated Pet Feeder User Guide

Congratulations on your new TOÑITO Automated Pet Feeder! This comprehensive guide will assist you in utilizing your TOÑITO to ensure your pet is fed timely and accurately.

Quick Start

Your TOÑITO is designed for ease of use, right out of the box. The Arduinos are pre-installed, making setup a breeze.

What's Included

Your TOÑITO comes fully equipped with:

- Pre-installed Arduino controllers for operation and interaction ● An LED Screen for easy programming and status updates.
- Increment and decrement buttons for time setting
- A motorized food dispenser with a flap
- An Ultrasonic Sensor for food dispensing accuracy
- An HC-SR04 Ultrasonic Sensor for monitoring food levels
- RGB LED lights indicating food container status
- A speaker for interactive feedback
- An RFID sensor for pet detection
- All necessary connecting wires and power supply components

Powering Up

Connect to Power: Plug your TOÑITO into a power source using the provided power supply.
Power On: Switch on the main power. The LED screen on the device will light up, indicating that TOÑITO is ready to be set up.

<u>Setting Up Your TOÑITO</u>

Programming Feeding Times

- Accessing the Menu: Use the buttons next to the LED screen on the feeder to navigate through the menu.
- Setting the Time: Select the time setting option and use the buttons to set the current time.
- Scheduling Feeds: Navigate to the feeding schedule menu. Set the desired feeding times for each meal.

<u>Operating Your Feeder</u>

Daily Use

- Automatic Feeding: TOÑITO will dispense food at the scheduled times. The LED screen displays upcoming feed times
- Manual Feeding: If an extra feeding is needed, you can manually activate the dispenser through the menu options on the LED screen.

Monitoring and Refilling

- Food Level Alerts: The RGB LED lights indicate the food level in the container: Green for full, Purple for half, and Red for low.
- Refilling: When the LED turns red, it's time to refill the food container. Simply lift the lid and add pet food until the container is full.

Interactive Features

- Pet Recognition: Place the RFID tag on your pet's collar. When your pet approaches, TOÑITO will detect it and can play sounds through the speaker, making feeding time fun.

Maintenance and Care

- Cleaning: Regularly clean the food container, flap, and feeding area to maintain hygiene.
- Check Connections: Periodically check the connections and components for wear and ensure everything is securely attached.
- Software Updates: Keep an eye out for firmware updates that may enhance the functionality of your TOÑITO. Check out the official github for more information

Troubleshooting & Support

Should you encounter any issues or have questions:

- Feeder Not Dispensing: Ensure there are no blockages and the scheduled feeding times are correctly set.
- Food Level Misreads: Verify the ultrasonic sensor is not obstructed and properly calibrated.
- Silent Speaker: Check the speaker and RFID connections.

Thank you for choosing TOÑITO, the ultimate companion in pet care automation. Enjoy the convenience and peace of mind knowing your pet is well-fed and happy.

List of Materials needed :

1. 4x Arduino R3 Microcontroller
2. 4x Breadboard
3. 1x Rectangle Box (Food Holder) with a Hole in bottom

Arduino 1 - LED screen

4. 4x Buttons
5. 1x LCD Module

Arduino 2 - Food Dispenser

6. 1x 3-6V Motor
7. 1x Ultrasonic Sensor OR Force Sensor (Pressure Plate)
8. 1x Square flap (Should be connectable to the Motor)

Arduino 3 - LED Lights

9. 1x HC-SR04 Ultrasonic Sensor
10. 1x RGV LED Lights

Arduino 4 - Speaker

11. 1x Speaker / Passive Buzzer
12. 1x RFID Sensor module (RC5222 Module)
13. 1x Passive RFID Tag (FOB)

Resources

1. https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/
2. https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1
3. https://arduinogetstarted.com/tutorials/arduino-force-sensor
4. https://www.elementzonline.com/blog/interfacing-rfid-rc522-with-arduino-uno
5. miguelbalboa/rfid: Arduino RFID Library for MFRC522
6. Play a Melody using the tone() function | Arduino Documentation
7. Playing popular songs with Arduino and a buzzer | Arduino Project Hub
8. https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1\

9. https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/
10. https://www.instructables.com/Arduino-Servo-Motors/
11. https://docs.arduino.cc/learn/electronics/servo-motors/
12. https://docs.arduino.cc/learn/electronics/lcd-displays/

Tools

1. GitHub
2. TinkerCad
3. ClickUp