COMPUTER ORGANIZATION AND ARCHITECTURE PROJECT REPORT

Smart Pet Feeder

Jaypee Institute of Information Technology

Submitted by

Ananya Kapoor	**20103104**	**B4**
Dhairya Sachdeva	**20103098**	**B4**

Submitted To

Dr. Janardhan Verma

DECLARATION

We hereby declare that the project titled *“Smart Pet Feeder”* submitted for the degree of Bachelors in Technology is a bonafide record submitted to Jaypee Institute of Information Technology, under the guidance of Dr. Janardhan Verma, has been carried out by our own efforts and is a record of our original work.

1. Introduction

Pets need special treatment and special care. Due to nowadays busy life style, this task is not as simple as it used to be. Most people that have pets know the struggle that comes with it. The goal of this project is to introduce, design and implement a smart pet system. To address this issue, ‘Smart Pet Feeder’ is an IOT-enabled, button-controlled pet feeder that can feed your pet in your absence. Smart Pet Feeder is powered by an Arduino Uno at its core and is controlled by a button that triggers the ‘feed’ action.

2. Working

To create the circuit, we have use buttons to set the timer for when the pet has to be fed.

The lid of the food container which is connected to the servo motor will open according to the time set using the buttons. The servo motor will act as the lid to control the amount of food released from the container.

An LCD screen allows you to set the time. It also displays the remaining time and tells us when the pet has been fed.

There are ‘mode change buttons’ to help one to go through the various options.

When the most left button ("mode change button") is pressed the mode changes

mode 0 - home screen - "Pet Feeder V1"

mode 1 - set hours

mode 2 - set minutes

mode 3 - start timer

mode 4 - cancel current timer

A prototype of a real timer

Uses "delay();" to act as a real timer

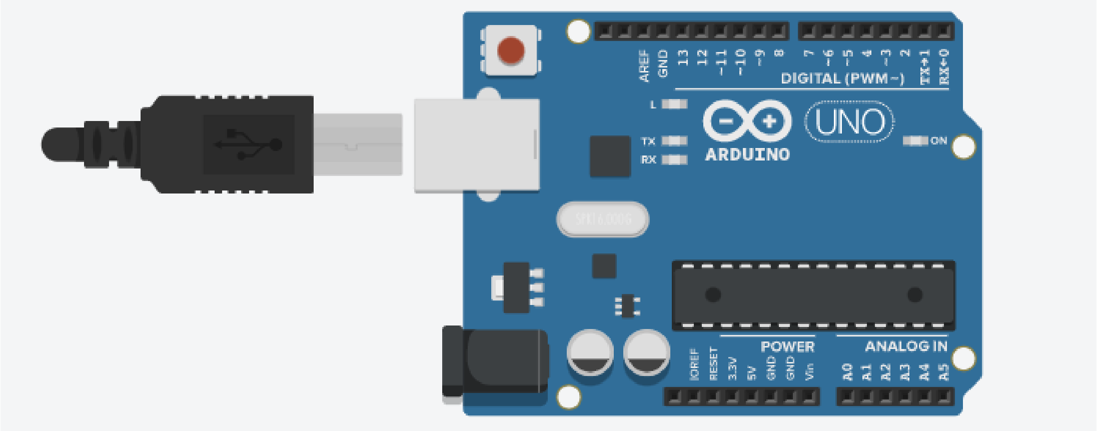
We have also added a buzzer option which would be enabled when the pet is being fed. This informs the user that their pet is being fed and the food is being released.

In this circuit we have used resistors and wires to complete the circuit wiring and control the amount of electricity through the circuit.

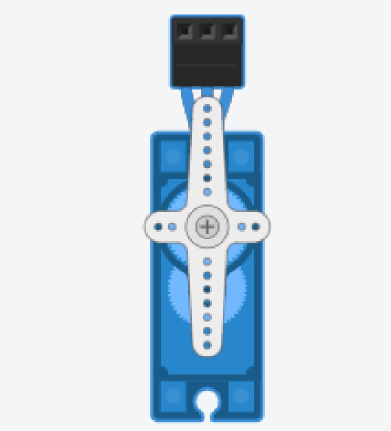
The circuit is coded using TinkerCad online platform which allows block-based programming language of Arduino Uno.

Components List:

- LCD 16X4
- Push Button x3
- 10 k ohm Resistor x3
- 1k ohm Resistor
- 330-ohm Resistor
- A small Breadboard
- Piezo
- Arduino Uno R3 - A programmable board you can use to build interactive circuits.



- Micro Servo - A motor whose position can be controlled using a microcontroller like an Arduino.



3. Block Diagram

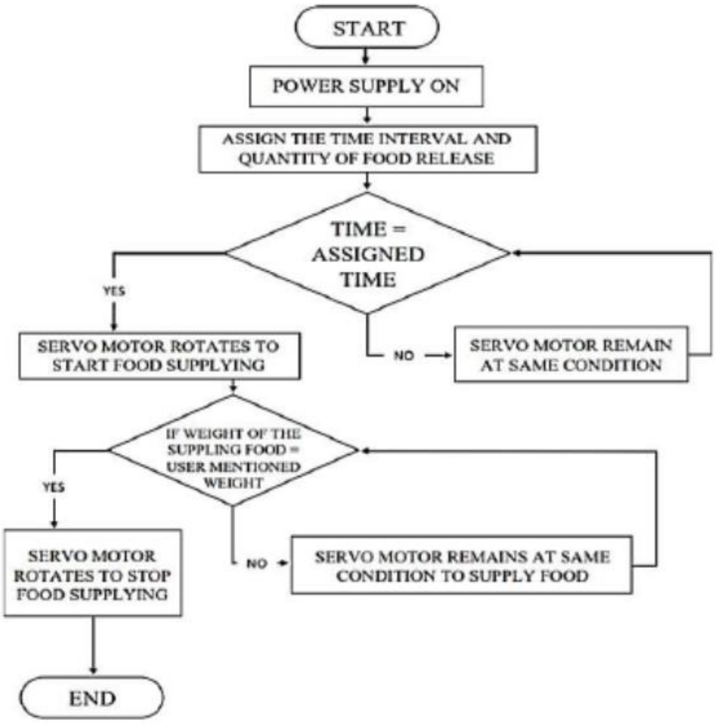

START
POWER SUPPLY ON
ASSIGN THE TIME INTERVAL AND QUANTITY OF FOOD RELEASE
TIME = ASSIGNED TIME
YES
NO
SERVO MOTOR ROTATES TO START FOOD SUPPLYING
SERVO MOTOR REMAIN AT SAME CONDITION
IF WEIGHT OF THE SUPPLING FOOD = USER MENTIONED WEIGHT
YES
NO
SERVO MOTOR ROTATES TO STOP FOOD SUPPLYING
SERVO MOTOR REMAINS AT SAME CONDITION TO SUPPLY FOOD
END

Figure 1: Flow Chart of smart pet feeder

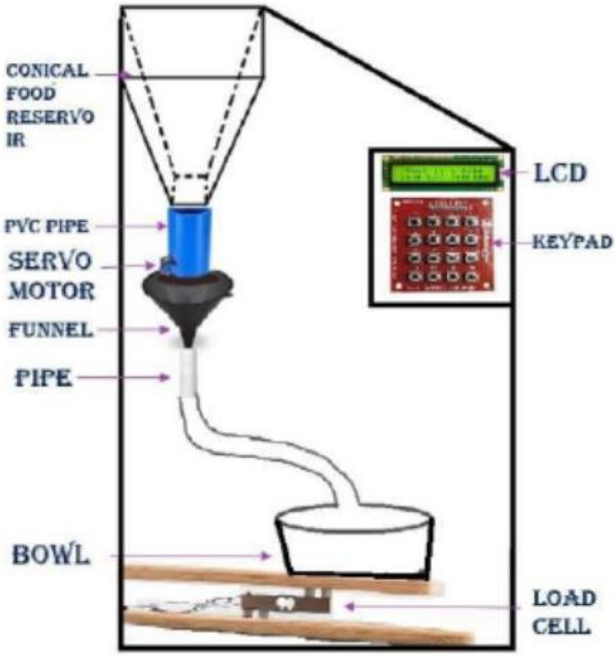
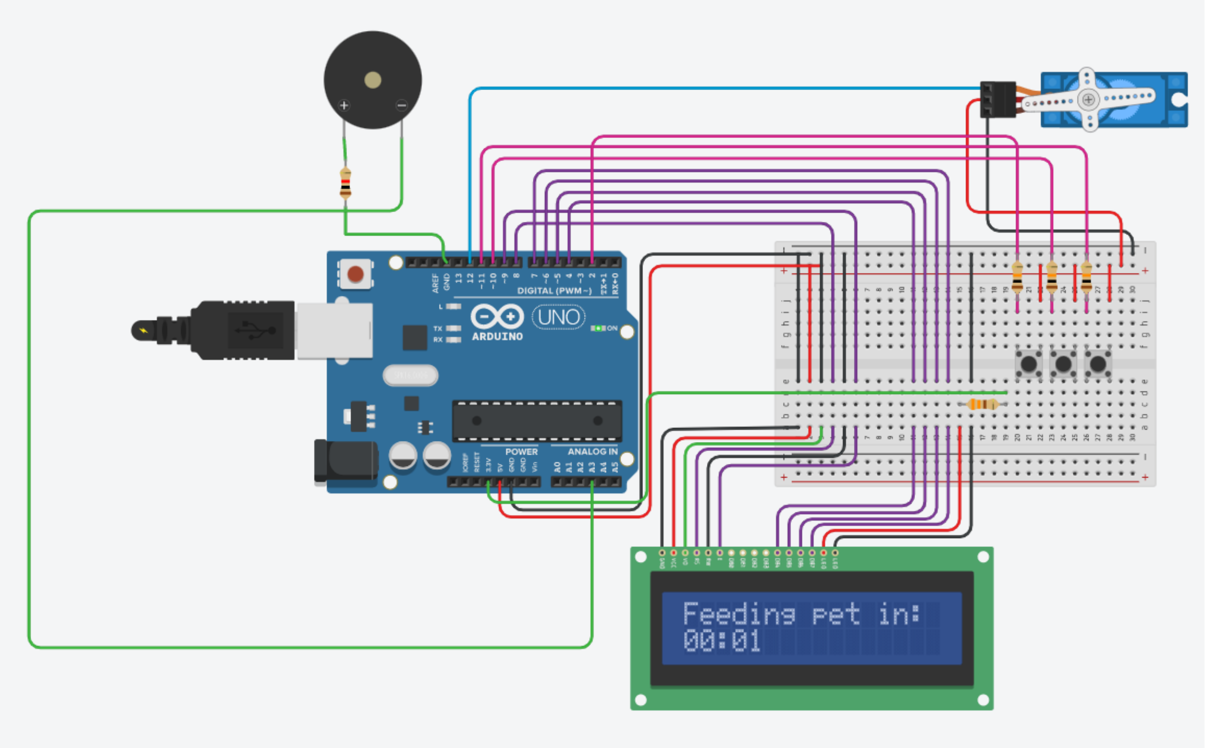


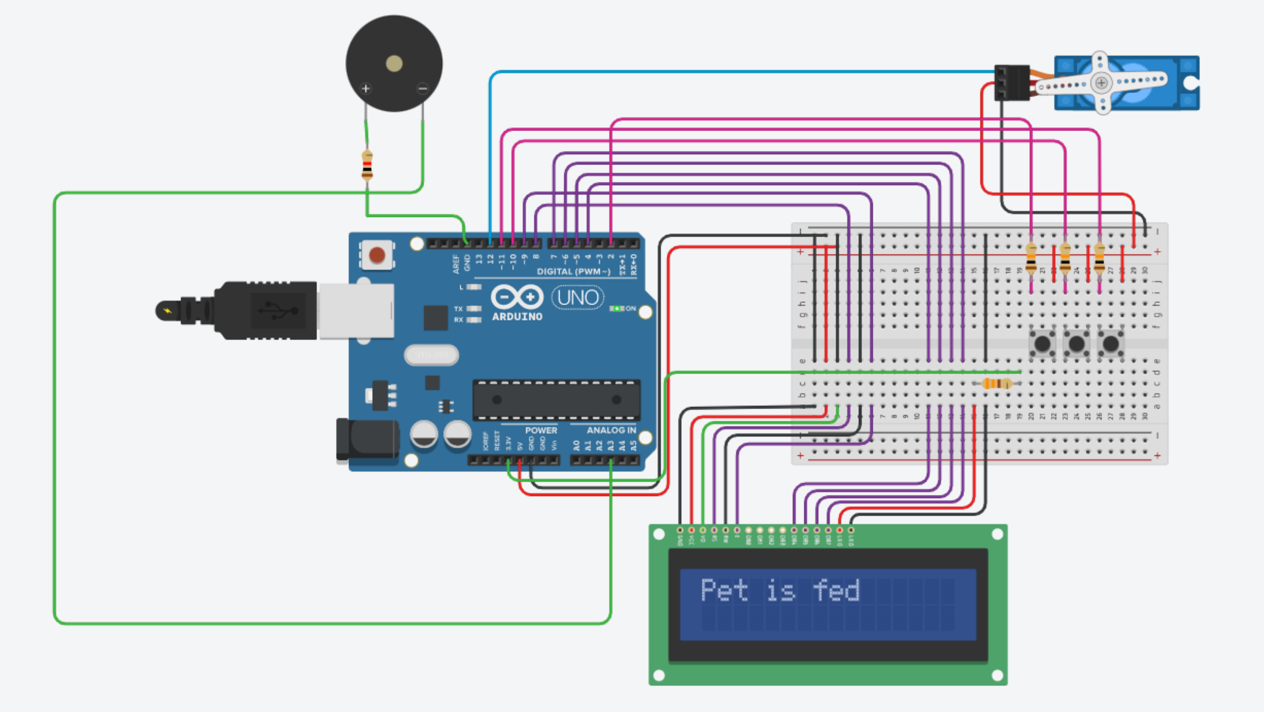
Figure 2: Schematic diagram of auto pet feeder

4. Circuit Diagram

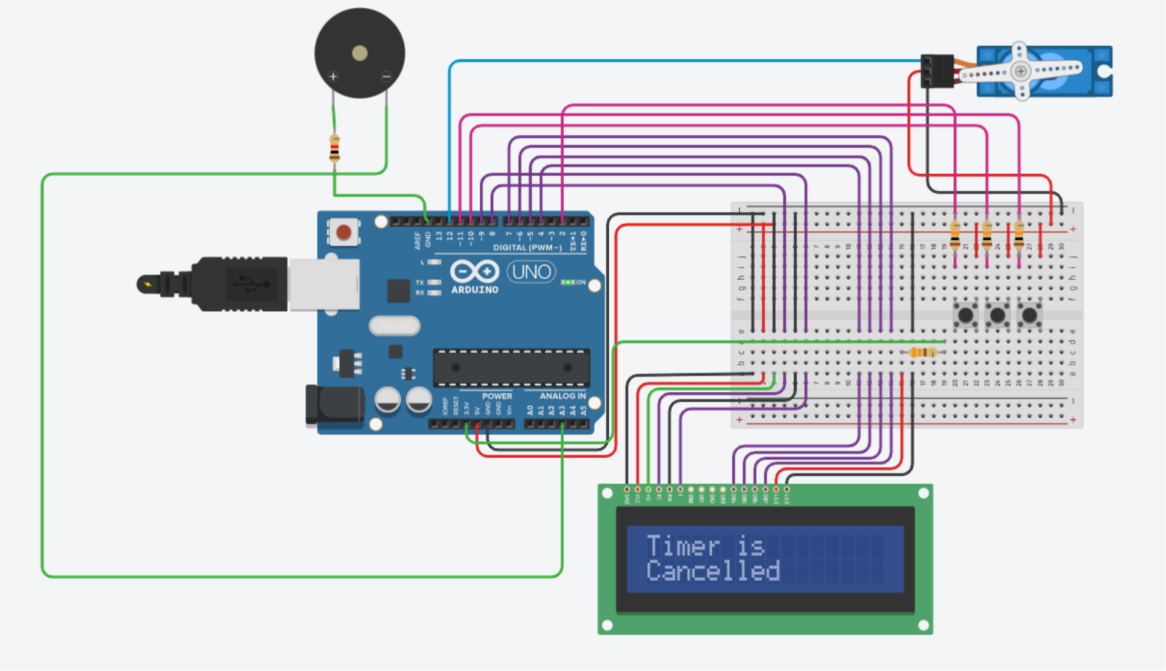
1st step is to set the timer:



Here, the timer has gone off and the pet is being fed (food is released).



After the food is released, servo motor goes back to its original position and timer is reset.



5. Code

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

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
```

```
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
```

```
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
```

```
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

#define melodyPin A3
//Mario main theme melody
int melody[] = {
  NOTE_E7, NOTE_E7, 0, NOTE_E7,
  0, NOTE_C7, NOTE_E7, 0,
  NOTE_G7, 0, 0,  0,
  NOTE_G6, 0, 0, 0,

  NOTE_C7, 0, 0, NOTE_G6,
  0, 0, NOTE_E6, 0,
  0, NOTE_A6, 0, NOTE_B6,
  0, NOTE_AS6, NOTE_A6, 0,

  NOTE_G6, NOTE_E7, NOTE_G7,
  NOTE_A7, 0, NOTE_F7, NOTE_G7,
  0, NOTE_E7, 0, NOTE_C7,
  NOTE_D7, NOTE_B6, 0, 0,

  NOTE_C7, 0, 0, NOTE_G6,
  0, 0, NOTE_E6, 0,
  0, NOTE_A6, 0, NOTE_B6,
  0, NOTE_AS6, NOTE_A6, 0,

  NOTE_G6, NOTE_E7, NOTE_G7,
  NOTE_A7, 0, NOTE_F7, NOTE_G7,
  0, NOTE_E7, 0, NOTE_C7,
  NOTE_D7, NOTE_B6, 0, 0
};
//Mario main them tempo
int tempo[] = {
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  9, 9, 9,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  9, 9, 9,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
};
Servo servomotor;
LiquidCrystal lcd(8,9,4,5,6,7);

int mode = 0;
const int servoPin = 12;

int h = 0; // hours
int m = 0; // minutes

void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.print("Welcome!  ");
  lcd.setCursor(0,1);
  lcd.print("Pet Feeder V1  ");
  pinMode(10,INPUT);
  pinMode(11,INPUT);
  pinMode(2, INPUT);
  servomotor.attach(servoPin);
  servomotor.write(0);
  delay(500);
}

/*
LEFT BUTTON - navigates through different modes
MIDDLE BUTTON - used for adding 1 hour/minute
RIGHT BUTTON - used for removing 1 hour/minute
*/
void loop()
{

  ChangeMode();

  if(mode == 0)
  {
      lcd.setCursor(0,0);
      lcd.print("Welcome!");
      lcd.setCursor(0,1);
      lcd.print("Pet Feeder V1");
  }

  if(mode == 1)
  {
    SetHours();
    }

  if(mode == 2)
  {
     SetMinutes();
     }
  if(mode == 3)
  {
    Timer();
  }

  if(mode == 4)
  {
     CancelTimer();
  }
}

/*
When the most left button ("mode change button") is pressed the mode changes /*
mode 0 - home screen - "Pet Feeder V1"
mode 1 - set hours
mode 2 - set minutes
mode 3 - start timer
mode 4 - cancel current timer
*/
void ChangeMode()
{
    if(digitalRead(2) == HIGH)
    {
        mode++;
        lcd.clear();
        Serial.println("Mode changed to ");
        Serial.println(mode);

        if(mode >= 5)
        {
            mode = 0;
```

```
      h = 0;
      m = 0;
    }
    delay(1000);
  }
}

/*
Sets the hour for the timer
*/
void SetHours()
{
 if(digitalRead(10) == HIGH)
 {
  if(h <= 24)
  {
   h++;
   if(h == 24)
   {
    h = 0;
   }
   Serial.print("Hour Added!\n");
   Serial.print(h);
   Serial.print("\n");
  }
 }

 if(digitalRead(11) == HIGH)
 {
  if(h > 0)
  {
      h--;
      Serial.print("Hour removed!\n");
      Serial.print(h);
      Serial.print("\n");
    }
  }
  Print(1);
  delay(150);
}


/*
Set the minutes for the timer
*/
void SetMinutes()
{
  if(digitalRead(10) == HIGH)
  {
    if(m <= 59)
    {
      m++;
      if(m==59)
      {
        m = 0;
      }
      Serial.println("Minute added!");
      Serial.println(m);
    }
  }

  if(digitalRead(11) == HIGH)
  {
    if(m > 0)
    {
      m--;
      Serial.println("Minute removed!");
      Serial.println(m);
    }
  }

  Print(2);
  delay(100);
}

/*
A prototype of a real timer
Uses "delay();" to act as a real timer
Checks
*/
void Timer()
{
  Print(0);

  /*
  1 minute = 60 000 ms = 600 * 100 -> (iterations * delay ms)

  Using a loop of 600 iterations each of which is delayed by 100 ms
  allows the user to cancel the timer in intervals of 100 ms.

  If 'delay(60000);' was used the user wouldn't be able to abort the timer
  before a minute passes
  */
```

```
 for(int i = 0; i < 600; i ++)
 {
   delay(100);
   ChangeMode();
   if(mode == 4)
   {
     CancelTimer();
     Serial.println("The timer has been cancelled!");
     return;
   }
 }

 if(m > 0)
 {
  m--;
  Serial.println("A minute has passed! Remaining minutes: ");
  Serial.println(m);
 }

 if(m == 0 && h != 0)
 {
  h--;
  m = 59;
  Serial.println("An hour has passed! Remaining hours: ");
  Serial.println(h);
 }




}
```

```
/*
Prints message and time in accordance to the given 'print type' code

CODES:
0 - none, print only timer
1 - set hours
2 - set minutes
*/
void Print(int type)
{
  if(type == 1)
  {
   lcd.setCursor(0,0);
   lcd.print("Setting hours:");
    PrintTime();
  }

  if(type == 2)
  {
  lcd.setCursor(0,0);
  lcd.print("Setting minutes:");
  PrintTime();
  }

  if(type == 0)
  {
    if(h == 0 && m == 0)
    {
      lcd.clear();
      Feed();
      lcd.print("Pet is fed");
      h = -1;
      m = -1;
      mode++;
    }
    else
    {
        lcd.setCursor(0,0);
                        lcd.print("Feeding pet in:");
            PrintTime();
        delay(1000);

    }
  }
}

/*
Prints only the time on the second line of the lcd matrix
*/
void PrintTime()
{
  lcd.setCursor(0,1);
  if(h<10)
  {
    lcd.print("0");
    lcd.print(h);
  }
  else
  {
    lcd.print(h);
  }
  lcd.print(":");

  if(m < 10)
  {
    lcd.print("0");
    lcd.print(m);
  }
  else
  {
    lcd.print(m);
  }
}

/*
Rotates servomotors wing allowing the food to drop
*/
void Feed()
{
  servomotor.write(90);
  sing();
  delay(1000);
  servomotor.write(0);
  Serial.println("Pet has just been fed!");
}

/*
Cancels the timer
Set hours and minutes to 0
*/
void CancelTimer()
```

```
{
    h = 0;
    m = 0;
  delay(2000);
    lcd.setCursor(0,0);
    lcd.print("Timer is          ");
    lcd.setCursor(0,1);
    lcd.print("Cancelled         ");


}
void buzz(int targetPin, long frequency, long length) {
  digitalWrite(13, HIGH);
  long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions
  //// 1 second's worth of microseconds, divided by the frequency, then split in half since
  //// there are two phases to each cycle
  long numCycles = frequency * length / 1000; // calculate the number of cycles for proper
timing
  //// multiply frequency, which is really cycles per second, by the number of seconds to
  //// get the total number of cycles to produce
  for (long i = 0; i < numCycles; i++) { // for the calculated length of time...
    digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphram
    delayMicroseconds(delayValue); // wait for the calculated delay value
    digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphram
    delayMicroseconds(delayValue); // wait again or the calculated delay value
  }
  digitalWrite(13, LOW);


}
void sing() {
  int size = sizeof(melody) / sizeof(int);
    for (int thisNote = 0; thisNote < size; thisNote++) {
```

```
  // to calculate the note duration, take one second
  // divided by the note type.
  //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
  int noteDuration = 1000 / tempo[thisNote];

  buzz(melodyPin, melody[thisNote], noteDuration);

  // to distinguish the notes, set a minimum time between them.
  // the note's duration + 30% seems to work well:
  int pauseBetweenNotes = noteDuration * 1.30;
  delay(pauseBetweenNotes);

  // stop the tone playing:
  buzz(melodyPin, 0, noteDuration);
 }}
```

6. Conclusion

Auto pet feeder is the solution for those who love to keep pets in their houses but because of busy schedules they may not be able to provide food to their beloved pets at fixed time. This prototype can be used to release food by inputting the chosen time.

The owner is also informed as music or some tone is played while the food is being released.

7. References

- https://www.researchgate.net/publication/349798792_Smart_Pet_Feeder
- https://www.tinkercad.com/dashboard
- https://circuitdigest.com/microcontroller-projects/automatic-pet-feeder-using-arduino