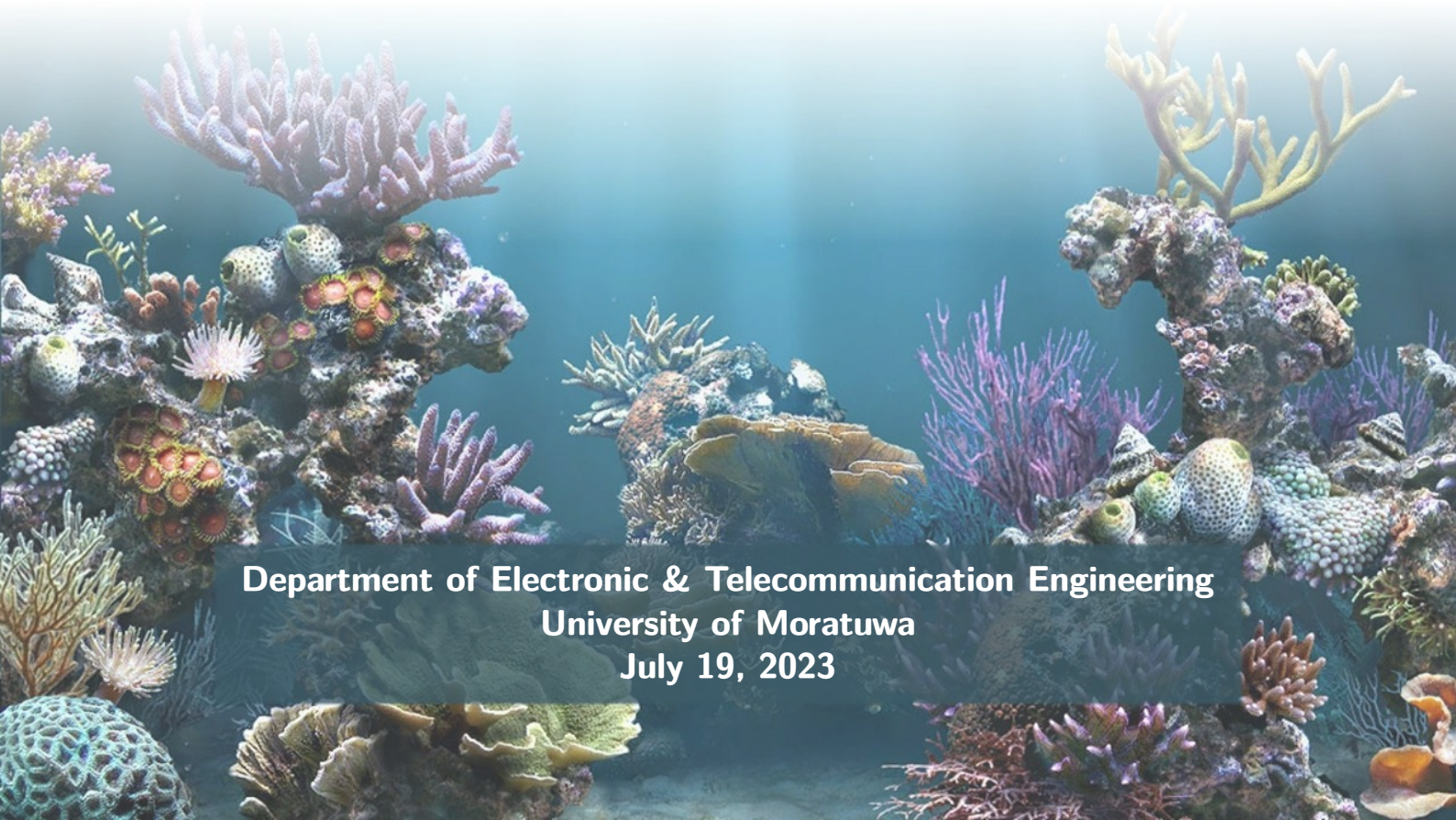
Automatic Fish Feeder

Project Report

EN2160: Electronic Design Realization

PERERA N.W.P.R.A. 200462U



Department of Electronic & Telecommunication Engineering
University of Moratuwa
July 19, 2023

Contents

1 | Introduction

In today's fast-paced world, automation has become an integral part of our daily lives, simplifying tasks and enhancing convenience. With the advancement of technology, even mundane activities like feeding our pets can be automated for efficiency and peace of mind. In line with this concept, the focus of this individual project is to design and implement an automatic fish feeder that will revolutionize the way fish enthusiasts care for their aquatic companions.

The goal of this project was to make improvements to an existing market product and make it more marketable. After careful consideration, an automatic fish feeder was chosen as the ideal candidate. This product aims to alleviate the concerns of manually feeding fish multiple times a day and the worry of missing a feeding time, ensuring the well-being of the fish while providing convenience to the user.

The automatic fish feeder will incorporate additional features that set it apart from existing market products. One notable enhancement is the use of an AC-DC power adapter instead of relying on battery power. This modification offers several advantages, including uninterrupted operation without the need for battery replacements, simplified circuitry resulting in cost reduction, space optimization within the enclosure, and reduced weight.

Moreover, the user interface of the automatic fish feeder has been designed with simplicity in mind. Rather than complex LCD displays and multiple buttons, the product utilizes intuitive controls using rocker switches. The switches are accompanied by clearly labeled indicators, allowing users to effortlessly navigate between timing and quantity options. This streamlined approach reduces circuit complexity, component costs, and eliminates the need for redundant visual indicators.

Furthermore, the automatic fish feeder incorporates extra features to enhance functionality and user experience. It includes a level measurement system that detects when the fish food storage compartment runs low. A color-changing RGB LED and an alarm system promptly notify the user, ensuring the fish are not deprived of their meals. The alarm is designed to ring at regular intervals, minimizing user irritation while effectively conveying the need for a refill.

To accommodate different tank setups, the automatic fish feeder is designed to be mountable on both sides of the tank. This versatility caters to covered as well as uncovered fish tanks, providing ease of access and visibility for users. Additionally, a touch sensor is employed as the manual feed key, offering a seamless and user-friendly interaction. It allows users to manually feed their fish in real time or reset the timer functionality with a simple touch, while also serving for testing and demonstration purposes.

Through the implementation of these innovative features and modifications, the automatic fish feeder aims to provide fish owners with an efficient, reliable, and user-friendly solution for automatic fish feeding.

2 | Functionality & Features

The automatic fish feeder offers a range of functionality and features to ensure efficient and customized fish feeding. This section will outline the various capabilities and characteristics of the product.

1. Automatic Dispensing: The primary function of the automatic fish feeder is to dispense fish food automatically according to the user's requirements. By selecting one of three quantity options (Low, Medium, High), the user can determine the amount of fish food to be dispensed at each feeding.
2. Flexible Timing Options: The fish feeder provides flexibility in feeding frequency with three timer options available for selection:
   - 24 Hours : Feeding occurs once a day.
   - 12 Hours : Feeding occurs twice a day.
   - 8 Hours : Feeding occurs three times a day.
3. Continuous Monitoring: The controller continuously checks the timing and quantity options selected by the user. Based on these settings, the device takes appropriate actions to dispense fish food at the programmed intervals.
4. Adjustable Countdown Timer: The fish feeder offers two methods to reset the countdown timer, allowing users to specify when the feeding cycle should start:
   [a] Device Power-On: When the device is powered on, the countdown timer resets and begins timing from the starting point.
   [b] Manual Key Press: Pressing the manual key immediately dispenses fish food based on the currently selected quantity option. This action also resets the countdown timer to start from that moment onwards.

   The decision to exclude a real-time clock module from the design was made with cost reduction and user simplicity in mind. Incorporating a real-time clock module would have necessitated the user to initially set the time, requiring additional controls and potentially complicating the user interface. By eliminating this component, the overall design emphasizes a more streamlined and intuitive user experience.
5. Level Notification System: To ensure the user is promptly informed when the fish food storage compartment needs to be refilled, the product incorporates two indications:
   - LED Indicator: A RGB LED changes from green to red when the food level falls below a certain threshold. Once the compartment is refilled, the LED returns to green.
   - Alarm Indicator: An intermittent alarm serves as a reminder for users to refill the compartment. The alarm is designed to ring once every 6 hours, minimizing user irritation while effectively conveying the need for a refill.
6. Redundant Indicators: The presence of both the LED indicator and alarm ensures redundancy in notifying the user. If the user fails to notice the LED indicator, they will still hear the alarm indicator, ensuring timely action.
7. Intuitive Controls: The fish feeder simplifies user interaction through the use of two rocker switches, each with clearly labeled positions. All six controls, including the three timing and three quantity options, can be easily adjusted by positioning the rocker switches accordingly.
8. Power Source: The automatic fish feeder is powered using an AC-DC power adapter, eliminating the need for battery replacements.
9. Versatile Mounting Options: To accommodate different tank setups, the automatic fish feeder is designed to be mountable on both sides of the tank. This versatility caters to covered as well as uncovered fish tanks, providing ease of access and visibility for users. This is achieved by having redundant indicators and controls on 2 sides of the device.

By incorporating these functionality and features, the automatic fish feeder offers users a convenient and customizable solution for feeding their fish. From adjustable feeding quantities and timings to intuitive controls and versatile mounting options, this product aims to enhance the fish feeding experience while ensuring the well-being of aquatic companions.

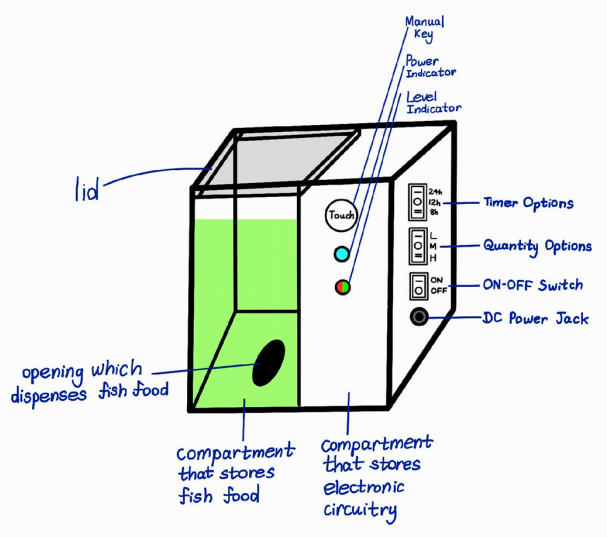


Figure 2.1: Initial Sketch of the Product

3 | Proposed User Interface

A primary objective of the product is to provide a user interface that is straightforward and intuitive. Recognizing that a device like an automatic fish feeder should not overwhelm users with complex controls and options, this product aims to deliver a seamless user experience. This section will outline the structure and components of the proposed user interface.

1. Inputs/User Controls:
   - 3 position rocker switch - Quantity Control (3 Options).
   - 3 position rocker switch - Timer Control (3 Options).
   - Touch Sensor - Manual Feeding & Timer Reset.
   - ON-OFF Switch - To turn the device ON or OFF.
2. Outputs/Indications to the User:
   - RGB Led - Turns from green to red when fish food is low.
   - Blue LED - To indicate that the device is receiving power.
   - Buzzer - Alarm functionality for when fish food is low.

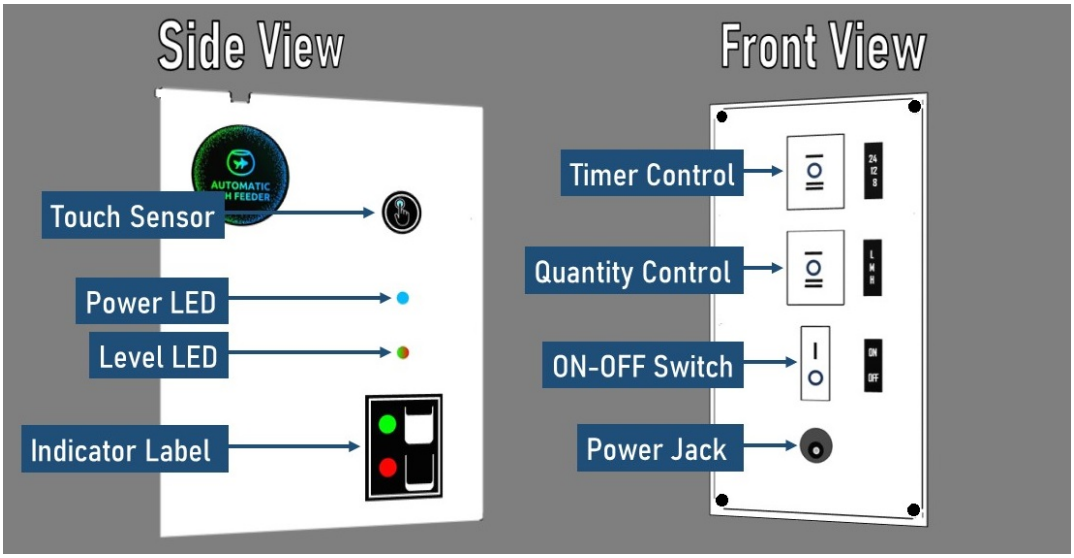


Figure 3.1: Proposed User Interface

A primary objective of this automatic fish feeder is to provide users with an interface so intuitive that they can effortlessly operate the device without the need for extensive user manuals or documentation. The focus is on simplicity, allowing users to understand and utilize the feeder's functions instinctively. In comparison to existing market products, this interface offers a significant reduction in complexity by removing features that are deemed unnecessary for an automatic fish feeder. For instance:

- Clock Feature: Since the device's primary function is automatic fish feeding, incorporating a clock feature is deemed unnecessary as users are unlikely to refer to the device for timekeeping purposes.
- Wi-Fi and Bluetooth: Integrating Wi-Fi and Bluetooth capabilities would complicate the device and introduce unnecessary reliance on smartphone apps. The goal is to automate the fish feeding process without requiring additional technological complexity.
- LCD Displays and Buttons: Instead of utilizing LCD displays and multiple buttons, the proposed interface utilizes only two rocker switches to achieve all six user controls. Labels adjacent to each rocker switch position clearly indicate the function associated with that position, simplifying operation and eliminating clutter.

To provide a visual comparison, two example interfaces from existing market products are given below.

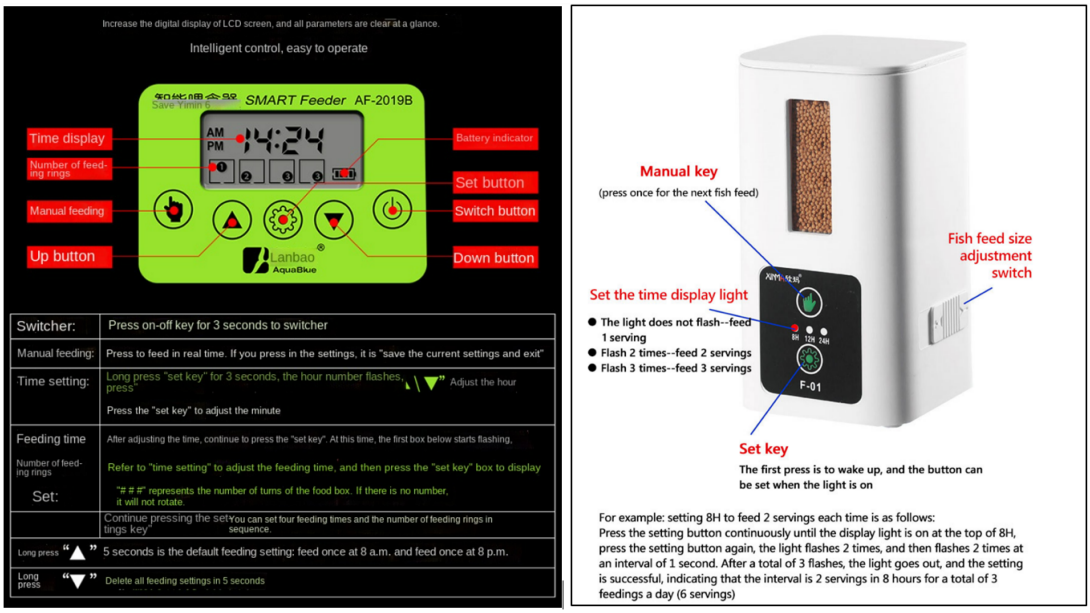


Figure 3.2: Example Interfaces from Existing Market Products

Upon observing the above interfaces, it becomes evident how cluttered they are compared to the proposed user interface. The interface on the left side of the images features five buttons and a display, requiring multiple button presses and constant reference to the display output to achieve a desired option. The selection process may not be intuitive, potentially necessitating consultation of the user manual. On the other hand, the interface on the right side comprises only two push buttons and three LEDs, with only one button utilized for selecting options. While it offers a simpler layout, it still falls short in terms of user-friendliness and intuitiveness.

4 | Variations from Existing Market Products & Extra Features

4.1 | Extra Features

1. Simple User Interface (See Figure 3.1)
   - Inputs/Control by user – 2 rocker switches, Manual key.
   - Outputs/Indications to user – RGB led for level, Blue led for power, Buzzer for alarm, Labels.
2. Level measurement.
   - This product will detect when the level of fish food in the storage compartment goes below a certain level.
   - Then it will turn the RGB led from green to red. Also, a small alarm will go off at regular intervals to notify the user to fill the compartment (this feature is put in place so that user can be notified even if he doesn't notice the red light). The purpose of the alarm is only to notify the user that the fish food level is low and needs refilling. So, it will ring at regular intervals (once every 6 hours) just to notify the user, rather than ringing continuously since it will irritate the user.
   - Once the compartment is refilled, the RGB led turns back to green and the alarm will stop.
3. Mountable on both sides – For both covered fish tanks & uncovered fish tanks (See Figure 4.1)
   - This product will be designed in a way that it can be mounted on the left side of the tank as well as on the right side of the tank (Touch Sensor, RGB Led & Blue Led will be installed on 2 sides).
   - For both mounting directions, there will be indicator LEDs & touch sensors facing the front side of the tank to provide indications to the user & for ease of access to the manual key.
   - For both mounting directions, the control switches and power jack will on the side that is furthest from the water. Control switches do not require frequent access since the users won't be changing options on a daily basis. The power jack is located on the side that is furthest from the water.
4. Touch sensor
   - Manual feed key will be implemented using a touch sensor instead of push buttons that are in existing market products. Having a touch sensor that works with a light touch rather than a button that needs to be pushed will be convenient to the user.
   - There are 2 main reasons for having a manual key,
     [a] If the user wants to feed the fish in real time.
     [b] To reset the timer functionality (The countdown timer that determines when the next feeding takes place gets reset when the manual key is pressed).
   - The manual key can also be used for testing and demonstration purposes.

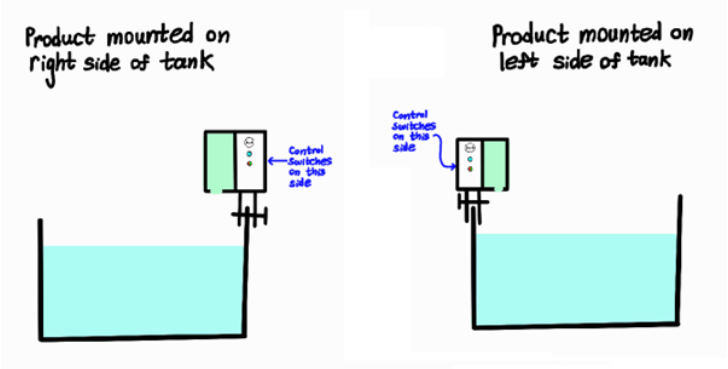


Figure 4.1: Mountable on Both Sides

4.2 | Variations

1. Will be powered using an AC-DC power adapter instead of using battery power.
   Reasons,
   - Can run indefinitely without needing to replace batteries.
   - Simplifies complexity of the circuit and hence reduces cost.
   - Eliminates the space that would be required by batteries within the enclosure.
   - Eliminates the extra weight that needs to be supported if batteries were present inside the enclosure.

2. This device will not use LCD displays along with multiple buttons for control. The device has 3 options for timing and 3 options for quantity. All 6 of these options will be achieved by using 2 rocker switches. Labels will be placed next to the rocker switches to indicate the 6 options relevant to each rocker switch position. (See Figure 3.1)
   Reasons,
   - Significantly reduces circuit complexity and cost of components. Hence reduces the overall cost of the product.
   - The user can obtain the required mode in a much simpler manner – In existing market products, the user has to navigate the control panel displayed on an LCD display using several buttons to achieve the mode of operation he requires (See Figure 3.2 and Figure 4.2). However, in this product, the user only needs to put the 2 rocker switches to the correct position in order to obtain the mode he requires. The options relevant to each position of each rocker switch will be indicated by labels that will be adjacent to the rocker switches. (See Figure 3.1).
   - The currently selected modes will not be indicated using an LCD display or indicator LEDs because it is redundant as the labels adjacent to the switches clearly indicate the selected mode (See Figure 3.1).
   - The Real Time Clock Function (See Figure 4.2) will not be incorporated into this product unlike existing market products since that feature does not seem necessary for an automatic fish food dispenser. The major argument to exclude a real-time clock module from the design was made with cost reduction and user simplicity in mind. Incorporating a real-time clock module would have necessitated the user to initially set the time, requiring additional controls and potentially complicating the user interface. By eliminating this component, the overall design emphasizes a more streamlined and intuitive user experience.

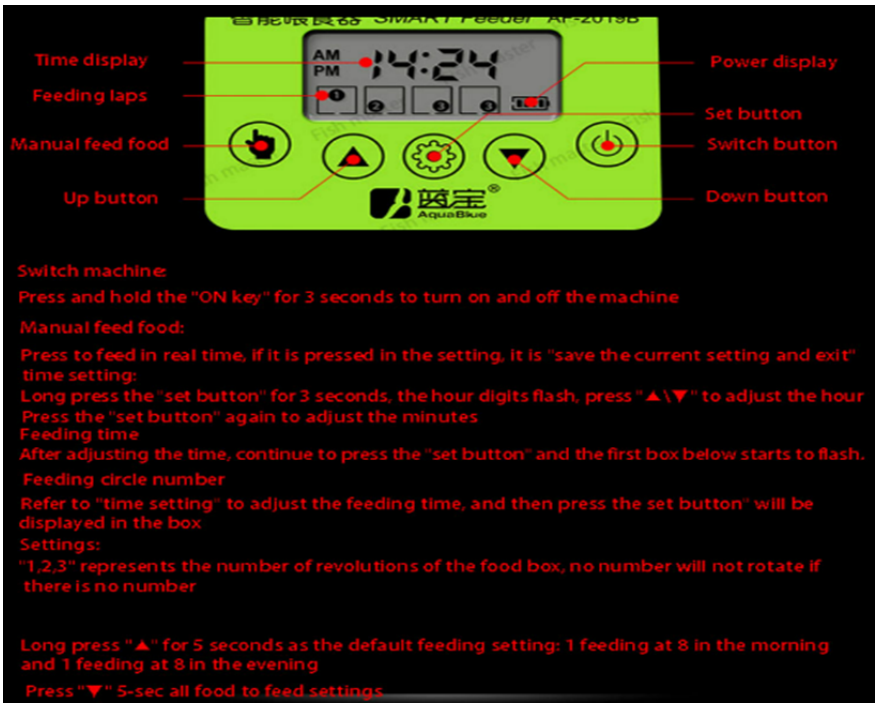


Figure 4.2: Example Interface from a Current Market Product

5 | Functional Block Diagram

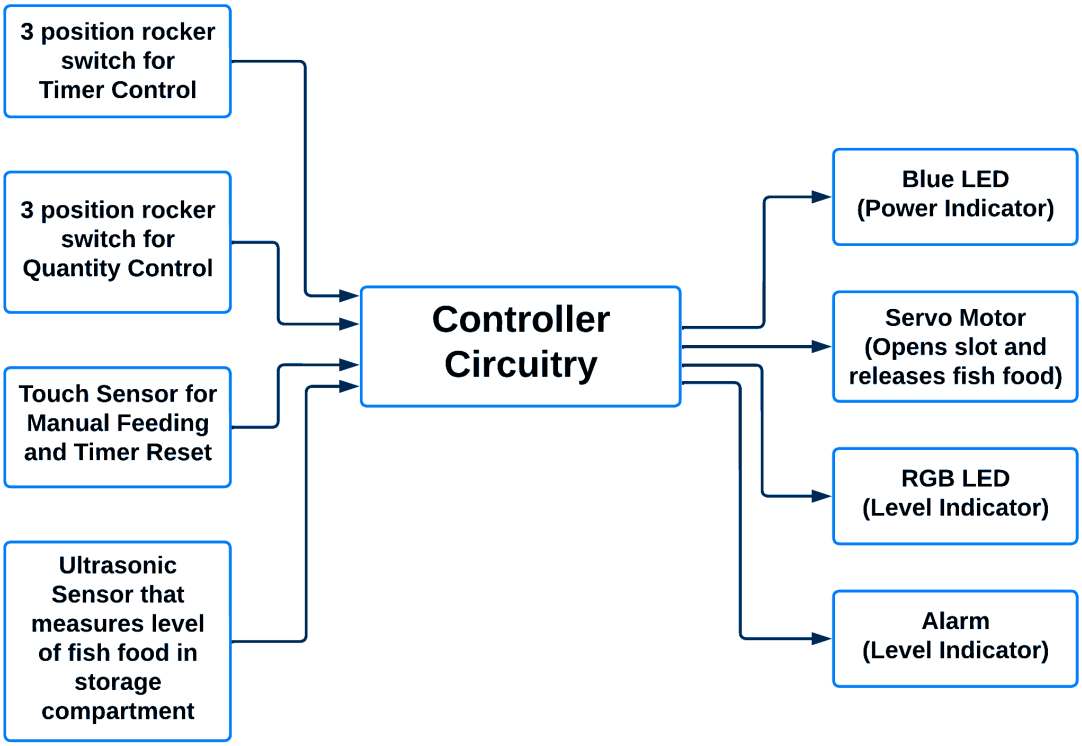


Figure 5.1

6 | Realization of Features & Selection of Components

6.1 | Power Delivery

The device will be powered using an AC-DC power adapter instead of using battery power.

Reasons,

- Can run indefinitely without needing to replace batteries.
- Simplifies complexity of the circuit and hence reduces cost.
- Eliminates the space that would be required by batteries within the enclosure.
- Eliminates the extra weight that needs to be supported if batteries were present inside the enclosure.

A commonly available AC-DC Switch Mode Power Supply Adapter with a 5V DC Output was selected for this purpose. The microcontroller, sensors and servo motor all require a 5V DC source. Hence this power supply is suitable for this device. In addition, the power supply provides features such as over-voltage protection, short-circuit protection and over-current protection. This also simplifies the internal power circuitry of the device.

Specifications of the selected power adapter:

- Input Voltage : 100 - 240V AC
- Output Voltage : 5V DC
- Rated Current : 2A

Figure 6.1

6.2 | Controller Implementation

The control of the device will be carried out using the ATMega328 microcontroller. The microcontroller will carry out the following tasks,

- Keeping track of the time elapsed since the last feeding.
- Sending control signals to the servo motor which opens the slot and releases fish food.
- Detecting what states the 2 control switches are in.
- Detecting when the touch sensor is pressed.
- Measuring the distance from the ultrasonic sensor to the surface of fish food in the storage compartment.
- Keeping track of the time elapsed since the last level measurement.
- Changing the color of the RGB LED as required.
- Sending signals to the alarm(buzzer) when necessary.

The ATMega328 microcontroller will be programmed using an Arduino Development Board and Arduino Programming Language (C++) and will be connected to the PCB using a 28 pin IC socket along with the required peripheral circuitry. It was decided to use a DIP type microcontroller along with IC sockets to enable simple reprogramming, easy repair, and replacement instead of using an SMD type microcontroller. Pin Configuration of the ATMega328 microcontroller is given below.

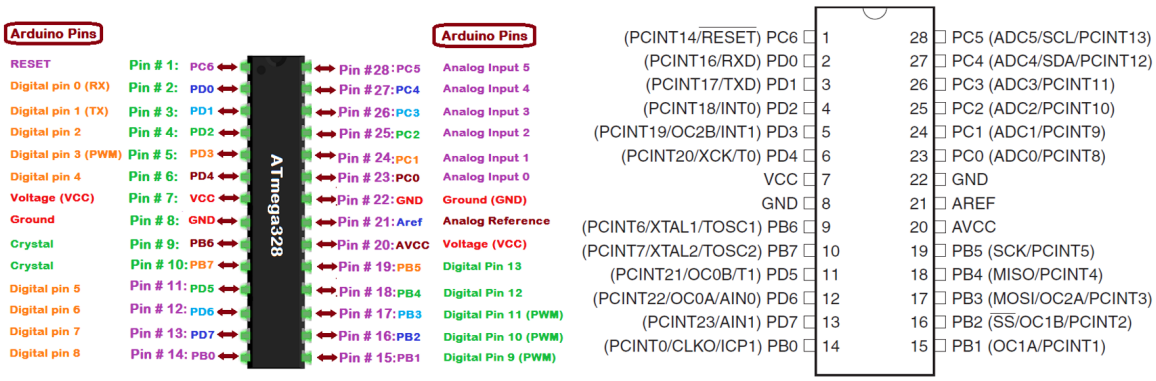


Figure 6.2

When using an ATMega328 microcontroller without a development board, it is important to include a few components in its periphery for proper functionality. The peripheral circuitry used for the ATMega328 is given below. It includes a 16MHz Crystal Oscillator accompanied by two 22pF Ceramic Capacitors. Also, a decoupling capacitor of 100nF will be used between VCC pin and GND pins.

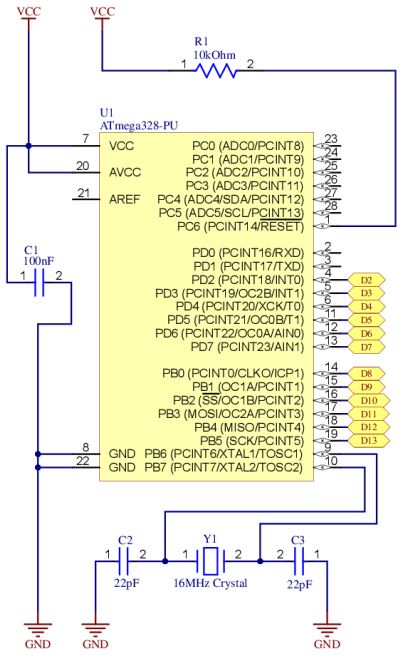


Figure 6.3

6.3 | Level Measurement

There are several possible methods that can be used to detect whether the level of fish food has gone below a certain level.

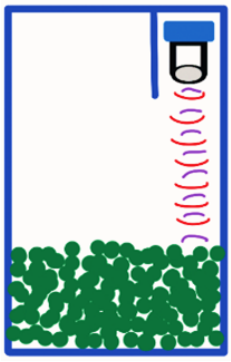


Figure 6.4: USS

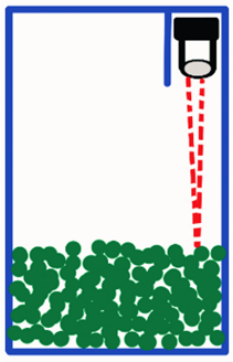


Figure 6.5: LDAR

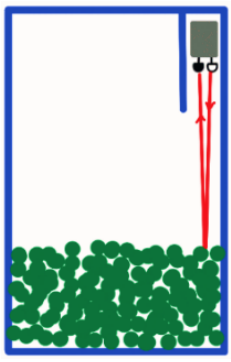


Figure 6.6: IR

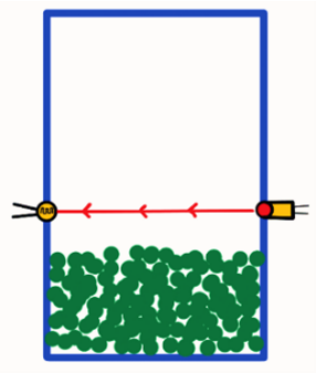


Figure 6.7: LASER

1. Using an Ultrasonic Sensor to measure the distance downwards and detect when distance exceeds a certain value. (See Figure 6.4)
   Advantages:
   - Low cost.
   - Doesn't depend upon lighting conditions.
   - Easy to interface with microcontroller.
   - High accuracy.
   - Low power consumption.

   Disadvantages:
   - Low resolution.
   - Slow refresh rate.

2. Using a LIDAR sensor to measure the distance downwards. (See Figure 6.5)
   Advantages:
   - Excellent max range.
   - Very fast update rate.

   Disadvantages:
   - Expensive.
   - High current draw.
   - Large footprint.

3. Using an IR Sensor to measure the distance downwards. (See Figure 6.6)
   Advantages:
   - Low cost.
   - Decent update rate.
   - Easy to interface with microcontroller.
   - Small footprint.

   Disadvantages:
   - High current consumption.
   - Gets affected by lighting conditions.
   - Low accuracy.
   - Low range.

4. Using a Laser Diode along with an LDR to detect obstacles (in this case, fish food). (See Figure 6.7)
   Advantages:
   - Low cost.

   Disadvantages:
   - High power consumption.
   - Should not allow Laser Diode and LDR to touch fish food. Hence the enclosure should have 2 transparent regions for the Laser and the LDR. Will increase manufacturing complexity.
   - Potential divider circuit with LDR and fixed resistor is needed to get the output. Output will be analog.

After consideration of the above factors, the ultrasonic sensor was decided as the best option for measuring the level of fish food. Selected ultrasonic sensor: **HC-SR04 4 Pin Ultrasonic Sensor Module : Link**

6.4 | Touch Sensor

Manual feed key will be implemented using a touch sensor instead of push buttons which are used in existing market products. Having a touch sensor that works with a light touch rather than a button that needs to be pushed will be a convenient addition to the user. There are 2 main reasons for having a manual key.

1. If the user wants to feed the fish in real time.
2. To reset the timer functionality (All timers get reset when the manual key is pressed).

Manual key will also be used for testing and demonstration purposes.

The following touch sensor will be used: **TTP223 Capacitive Touch Sensor Module : Link**

Figure 6.8

6.5 | Selection of Switches

A switch that has **3 states** is necessary for the proposed interface controls of this product. The 3 states can be mapped to the 3 control options in the program of the device.

3 possible choices were identified:

- Option A: Single Pole-Double Throw Switch with 3 positions (ON-OFF-ON).

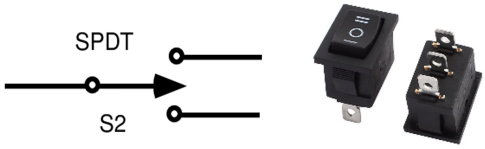


Figure 6.9

- Option B: Single Pole-Triple Throw Switch with 3 positions (ON-ON-ON).

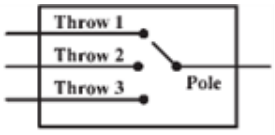


Figure 6.10

- Option C: 3 position Rotary Selector Switch.

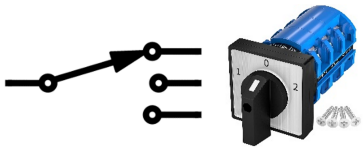


Figure 6.11

Ultimately, the Single Pole-Double Throw Switch (See Figure 6.9) with 3 positions **(Option A)** was selected for timing control and quantity control due to following reasons,

- Option A is widely available in the market.
- Option B is extremely rare in the market.
- Option C is available in the market. However, Most of the rotary selector switches in the market have multiple poles and more than 3 throws.

A simple Single Pole-Single Throw switch will be used as the ON-OFF switch.

Figure 6.12

6.6 | Pull-up Resistors

From each rocker switch, 2 voltage values will be obtained and given as digital inputs to the microcontroller. Depending on the state(position) of the switch, the 2 inputs can be 0V or 5V. To prevent floating inputs at the digital input pins of the microcontroller, that can cause errors to our logic, pull-up resistors will be used at 2 terminals of each switch. External pull-up resistors will be used instead of using the internal pull-up resistors inside the microcontroller due to the following reasons,

- Less heat dissipated by the microcontroller.
- Higher immunity to noise.
- The internal pull-up resistor value is fixed but we can adjust external pull-up resistor according to our requirements. So, external pull-up resistors will be used at 2 switch terminals to avoid floating inputs at the input pins of the microcontroller.

The value selected is 10kΩ. This provides a good balance between a strong pull-up as well as low power dissipation. (If resistance is too high, the pull-up will be weak. If the resistance is too low, the power wasted will be high) The maximum power dissipation across the 10kΩ pull-up resistors will be 2.5mW each.

The way that the 3 rocker switch positions will be mapped to 3 states by sending digital inputs to 2 pins of the microcontroller is shown below,

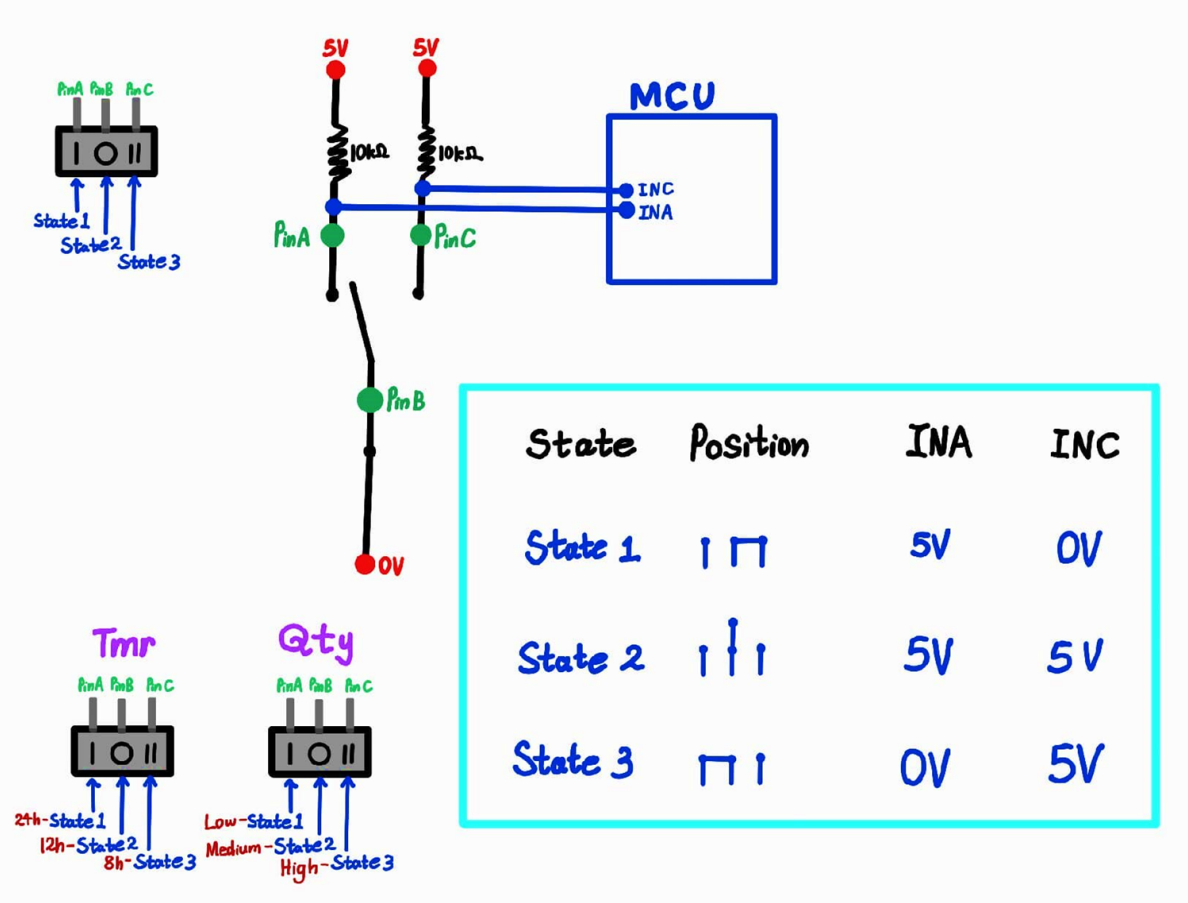


Figure 6.13

6.7 | Slot Mechanism

When it is required, the microcontroller will send control signals to a servo motor which opens and closes the slot. When the slot is open, fish food will be released from the storage compartment, and it stops when the slot is closed. Since opening and closing of the slot does not require a high torque and since power efficiency is important, a relatively low power servo motor can be used. Cost can also be minimized by doing so.

Selected servo motor: **TowerPro SG90 Micro Servo : Link**

A special attachment will be used along with the servo motor in order to open and close the slot to release the fish food. This attachment will be designed using SolidWorks and 3D printed.

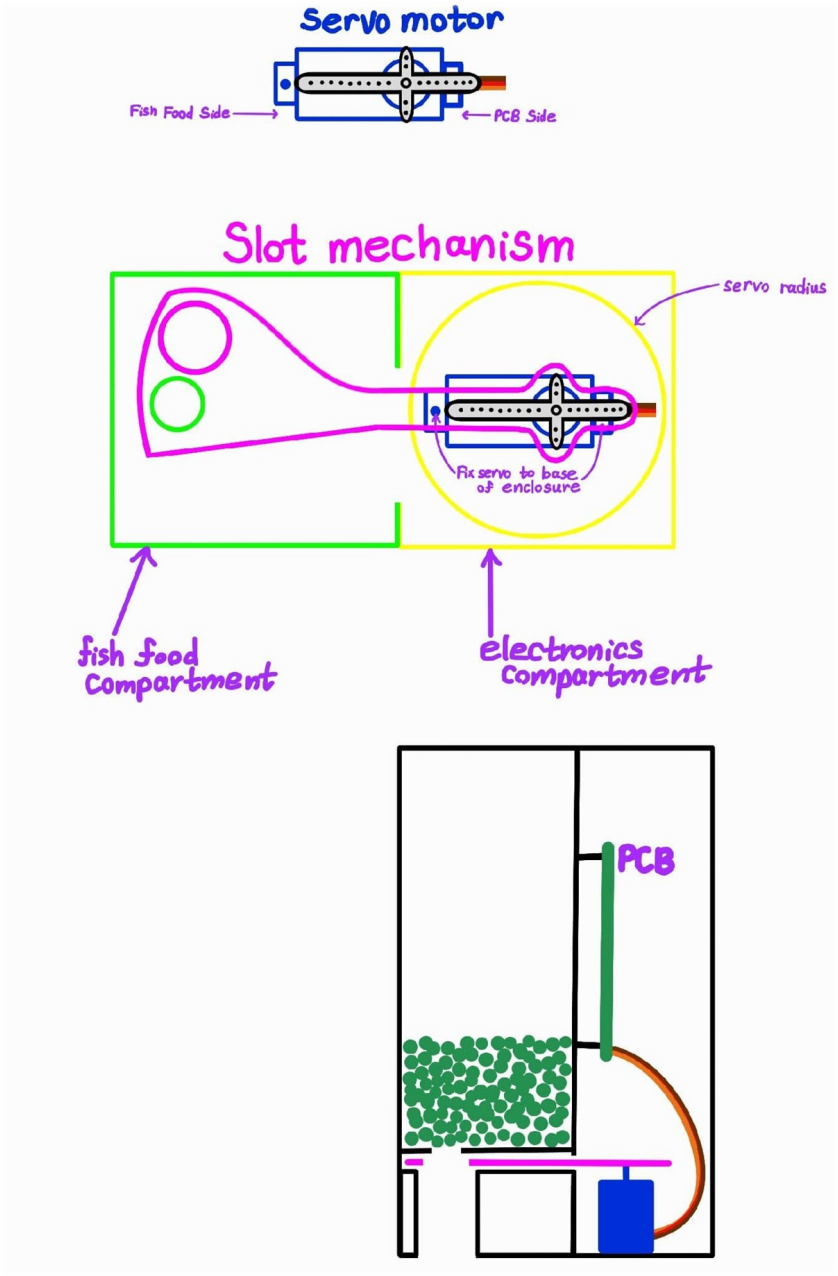


Figure 6.14: Sketches of the Proposed Servo & Slot Mechanism

7 | Programming

7.1 | Algorithm

Device States

- Qty State: Low, Medium, High
- Tmr State: 24h, 12h, 8h
- FF Level State: Low, High
- FF Alarm State: Low, High

Device Controls

- Touch sensor
- Tmr switch position
- Qty switch position

Device Responses

- Slot: Open, Close.
- RGB Led: Red, Green
- Alarm
- Power Led

Device Timers

- Fish Food Dispenser Timer: Depends on Tmr state
- Level Measurement Timer: 6 hours
- FFL Alarm Timer: 6 hours

Device Requirements

1. Check touch sensor input continuously. If touch sensor is pressed, dispense fish food according to currently selected qty state and reset all 3 timers.
2. Detect the position of the tmr switch and qty switch continuously and set states.
3. Control the servo motor to open the slot and release fish food according to fish food dispenser timer and currently selected states.
4. Check the level of fish food in the storage compartment using the ultrasonic sensor and set fish food level state according to level measurement timer. Also set the colour of the RGB Led according to the current fish food level state.
5. As long as level of fish food is low, ring the alarm once every 6 hours. This will be handled using FFL alarm timer.

```
if (FF_level_state==HIGH)
  FF_alarm_state=LOW;

if ((FF_level_state==LOW)&&(FF_alarm_state==LOW))
  FF_alarm_state=HIGH;
  previous_fflalarm_time=millis();

if ((FF_alarm_state==HIGH)&&((current_time-previous_fflalarm_time)>=fflalarm_interval))
  alarm();
  FF_alarm_state=LOW;
  previous_fflalarm_time=previous_fflalarm_time+fflalarm_interval;
```

7.2 | C++ Code

```
/*                  Automatic Fish Feeder Code                    */

/*-----  Importing Libraries and Declaration of I/O Pins  -----*/

#include <Servo.h>
Servo myservo;

const int RGB_green_pin=2;
const int RGB_red_pin=3;
const int buzzer_pin=4;
const int uss_trig_pin=5;
const int uss_echo_pin=6;
const int touch_sensor1_pin=7;
const int touch_sensor2_pin=8;
const int servo_pin=9;
const int tmr_switch_INA_pin=10;
const int tmr_switch_INC_pin=11;
const int qty_switch_INA_pin=12;
const int qty_switch_INC_pin=13;


/*----------  Declaration of variables and constants  ---------*/

const int fish_food_threshold_level=7;
// If distance between the sensor and surface of fish food
// is more than 7cm, fish food level is LOW. Else, HIGH.

int current_fish_food_level;

int distance;
int roundeddistance;
long t;

int tmr_switch_INA=1;  //  1 or 0
int tmr_switch_INC=1;  //  1 or 0
int qty_switch_INA=1;  //  1 or 0
int qty_switch_INC=1;  //  1 or 0

int tmr_state=1;  //  1:24hour, 2:12hour,   3:8hour
int qty_state=1;  //  1:Low,    2:Medium,   3:High

bool FF_level_state=HIGH;  // LOW:Low fish food,  HIGH: Sufficient fish food
bool FF_alarm_state=LOW;   // Alarm can only ring in HIGH state.


//THESE VALUES WERE SELECTED AFTER CALIBRATION OF THE PROTOTYPE

const int servo_slot_closed_pos=87;  //Servo position for closed slot (default).
const int servo_slot_open_pos=66;    //Servo position for opened slot.
int slot_open_delay;                 //Time for which the slot remains opened.
                                     //Depends on the qty_state.


//THESE VALUES WERE SELECTED AFTER CALIBRATION OF THE PROTOTYPE

const int low_duration=150;
//Duration to keep slot open for low quantity    : 0.15s

const int medium_duration=350;
//Duration to keep slot open for medium quantity : 0.35s
```

```
const int high_duration=500;
//Duration to keep slot open for high quantity   : 0.50s

unsigned long current_time=0;
unsigned long previous_fishfooddispense_time=0;
unsigned long previous_levelmeasurement_time=0;
unsigned long previous_fflalarm_time=0;

unsigned long fishfooddispense_interval;
// Depends on the tmr_state.
// For tmr_state 1, 24hours
// For tmr_state 2, 12hours
// For tmr_state 3,  8hours


// THE VALUES INDICATED BELOW HAVE BEEN USED IN THE PROTOTYPE
// SINCE FOR DEMONSTRATION PURPOSES, IT IS NOT POSSIBLE TO
// WAIT FOR LONG PERIODS OF TIME.

// CHANGE THESE INTERVALS AS REQUIRED FOR THE FINAL PRODUCT.

const unsigned long hour_24_interval = 20000;
// In final product:24 hours. Here it is 20s.

const unsigned long hour_12_interval = 10000;
// In final product:12 hours. Here it is 10s.

const unsigned long hour_8_interval  =  5000;
// In final product: 8 hours. Here it is 5s.

const unsigned long levelmeasurement_interval=10000;
// In final product: 6 hours. Here it is 10s.

const unsigned long fflalarm_interval=10000;
// In final product: 6 hours. Here it is 10s.


/*----------------------  Setup & Loop  -------------------------*/

void setup()
{
    pinMode(RGB_green_pin,OUTPUT);
    pinMode(RGB_red_pin,OUTPUT);
    pinMode(buzzer_pin,OUTPUT);
    pinMode(uss_trig_pin,OUTPUT);
    pinMode(uss_echo_pin,INPUT);
    pinMode(touch_sensor1_pin,INPUT);
    pinMode(touch_sensor2_pin,INPUT);
    pinMode(tmr_switch_INA_pin,INPUT);
    pinMode(tmr_switch_INC_pin,INPUT);
    pinMode(qty_switch_INA_pin,INPUT);
    pinMode(qty_switch_INC_pin,INPUT);
    myservo.attach(servo_pin);
    myservo.write(servo_slot_closed_pos);
    digitalWrite(RGB_green_pin,HIGH);
    digitalWrite(RGB_red_pin,HIGH);
}


void loop()
{
    current_time=millis();
```

```
    read_switches_and_set_states();
    manual_key();
    release_fish_food_when_required();
    level_measurement();
    low_fish_food_alarm();
}


/*------------------  User Defined Functions  ------------------*/

void read_switches_and_set_states()
// Reads the rocker switches, determines the qty and tmr states,
// sets slot_open_delay and fishfooddispense_interval.
{
    tmr_switch_INA=digitalRead(tmr_switch_INA_pin);
    tmr_switch_INC=digitalRead(tmr_switch_INC_pin);
    qty_switch_INA=digitalRead(qty_switch_INA_pin);
    qty_switch_INC=digitalRead(qty_switch_INC_pin);

    if ((tmr_switch_INA==HIGH)&&(tmr_switch_INC==LOW))
    {
        tmr_state=1;    //Tmr 24hour
    }
    else if ((tmr_switch_INA==HIGH)&&(tmr_switch_INC==HIGH))
    {
        tmr_state=2;    //Tmr 12hour
    }
    else if ((tmr_switch_INA==LOW)&&(tmr_switch_INC==HIGH))
    {
        tmr_state=3;    //Tmr 8hour
    }

    if ((qty_switch_INA==HIGH)&&(qty_switch_INC==LOW))
    {
        qty_state=3;    //Qty Low
    }
    else if ((qty_switch_INA==HIGH)&&(qty_switch_INC==HIGH))
    {
        qty_state=2;    //Qty Medium
    }
    else if ((qty_switch_INA==LOW)&&(qty_switch_INC==HIGH))
    {
        qty_state=1;    //Qty High
    }

    if (qty_state==1)
    {
        slot_open_delay=low_duration;
    }
    else if (qty_state==2)
    {
        slot_open_delay=medium_duration;
    }
    else if (qty_state==3)
    {
        slot_open_delay=high_duration;
    }

    if (tmr_state==1)
    {
        fishfooddispense_interval=hour_24_interval;
    }
    else if (tmr_state==2)
```

```
    {
        fishfooddispense_interval=hour_12_interval;
    }
    else if (tmr_state==3)
    {
        fishfooddispense_interval=hour_8_interval;
    }
}


void release_fish_food()
//Instantaneously releases fish food through the slot.
{
    myservo.write(servo_slot_open_pos);
    delay(slot_open_delay);
    myservo.write(servo_slot_closed_pos);
}


void release_fish_food_when_required()
//Releases fish food according to the fishfooddispense timer.
{
    if ((current_time-previous_fishfooddispense_time) >=
        fishfooddispense_interval)
    {
        release_fish_food();
        previous_fishfooddispense_time = previous_fishfooddispense_time +
            fishfooddispense_interval;
    }
}


void manual_key()
// Reads both touch sensors continuously. If either is pressed, instantaneously
// releases fish food according to the currently selected qty state.
// Also resets timers.
{
    int tsi_1=digitalRead(touch_sensor1_pin);
    int tsi_2=digitalRead(touch_sensor2_pin);

    if ((tsi_1==HIGH)||(tsi_2==HIGH))
    {
        release_fish_food();

        //All 3 timers will get reset when manual key is pressed.
        previous_fishfooddispense_time=current_time;
        previous_levelmeasurement_time=current_time;
        previous_fflalarm_time=current_time;

    }
}


void level_measurement()
// According to levelmeasurement timer, this measures fish food
// level, sets FF_level_state and sets colour of the RGB led.
{
    if ((current_time-previous_levelmeasurement_time) >=
        levelmeasurement_interval)
    {

        current_fish_food_level=measure_distance_using_uss();
```

```
        if (current_fish_food_level>=fish_food_threshold_level)
        {
            FF_level_state=LOW;
        }
        else
        {
            FF_level_state=HIGH;
        }

        //RGB Leds are common anode type. Output HIGH:OFF, Ouput LOW:ON.
        if (FF_level_state==LOW)
        {
            digitalWrite(RGB_green_pin,HIGH);   //Off green led
            digitalWrite(RGB_red_pin,LOW);      //On red led
        }
        else
        {
            digitalWrite(RGB_red_pin,HIGH);     //Off red led
            digitalWrite(RGB_green_pin,LOW);    //On green led
        }

        previous_levelmeasurement_time = previous_levelmeasurement_time +
            levelmeasurement_interval;

    }
}


int measure_distance_using_uss()
// Measures and returns the distance from ultrasonic
// sensor to surface of fish food (in cm).
{
    digitalWrite(uss_trig_pin,HIGH);
    delayMicroseconds(10);
    digitalWrite(uss_trig_pin,LOW);
    t=pulseIn(uss_echo_pin,HIGH);
    distance=t*330/2/10000;
    roundeddistance=round(distance);
    return roundeddistance;
}


void low_fish_food_alarm()
//Rings the alarm according to fflalarm timer.
{

    if (FF_level_state==HIGH)
    {
        FF_alarm_state=LOW;
    }

    if ((FF_level_state==LOW)&&(FF_alarm_state==LOW))
    {
        FF_alarm_state=HIGH;
        previous_fflalarm_time=millis();
    }

    if ((FF_alarm_state==HIGH)&&((current_time-previous_fflalarm_time)>=
        fflalarm_interval))
    {
        alarm();
        FF_alarm_state=LOW;
        previous_fflalarm_time = previous_fflalarm_time + fflalarm_interval;
```

```
    }
}


void alarm()
//Simulates an alarm using an active buzzer.
{
    digitalWrite(buzzer_pin,HIGH);
    delay(200);
    digitalWrite(buzzer_pin,LOW);
}

/*---------------------------- End of Code ------------------------------*/
```

8 | Schematics

8.1 | Circuit Diagram Sketch with Arduino

The intial circuit design was done using an Arduino UNO Development Board as the controller.

The rough **sketch** of the proposed circuit diagram is given below.

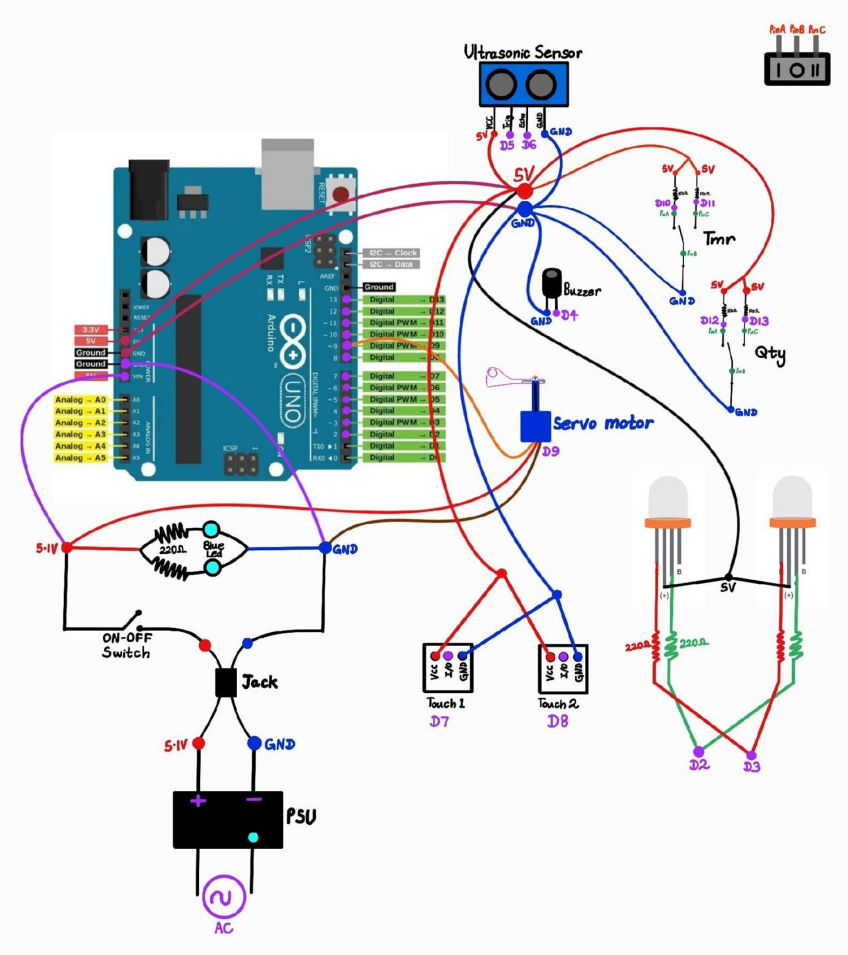


Figure 8.1

8.2 | Circuit Diagram Sketch without Arduino

After verifying that the above circuit works by implementing the above circuit on a protoboard, the circuit was redesigned without the Arduino Board.

Now the schematic uses only the ATMega328 Micro-controller and the necessary peripheral circuitry. See Section 6.2 for further details.

The rough **sketch** of the final circuit diagram is given below.

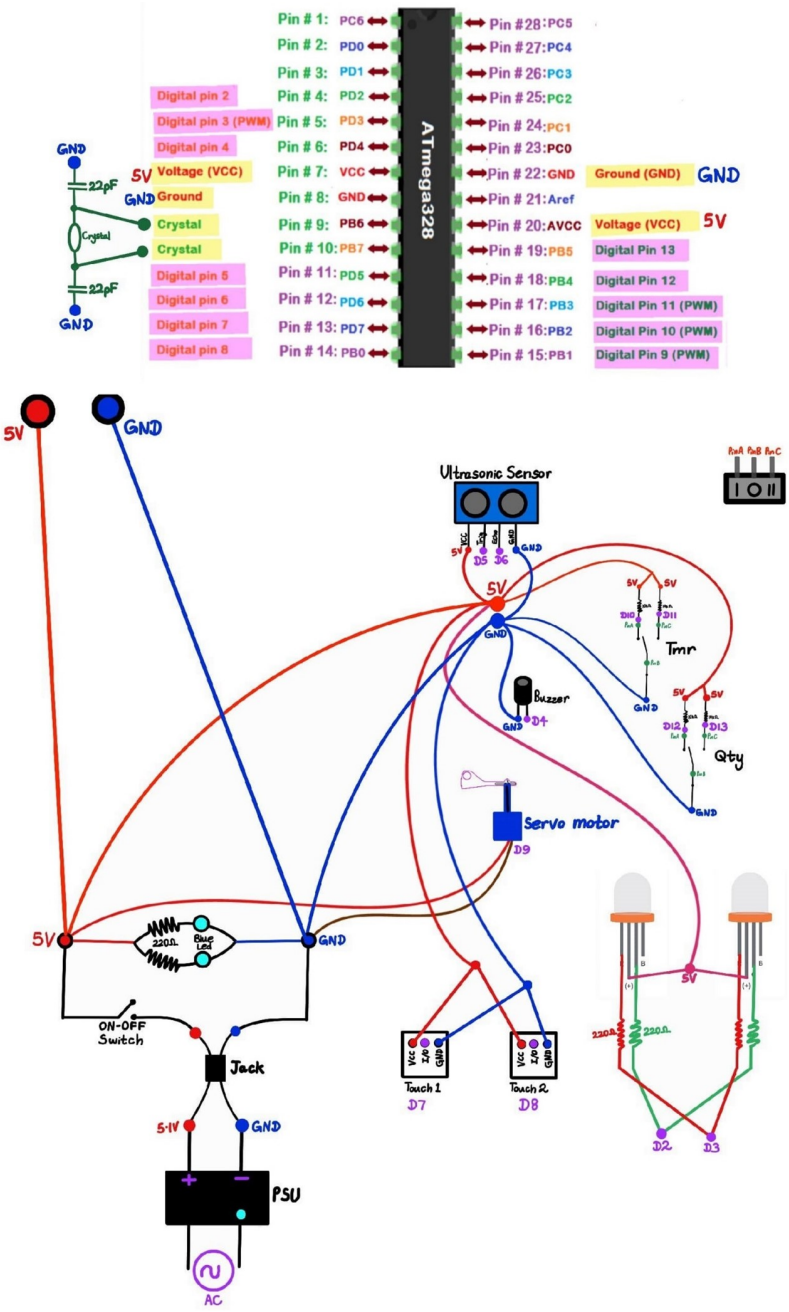


Figure 8.2

8.3 | Final Schematic

Then the above circuit diagram was drawn using a Professional Schematic Capture Software.

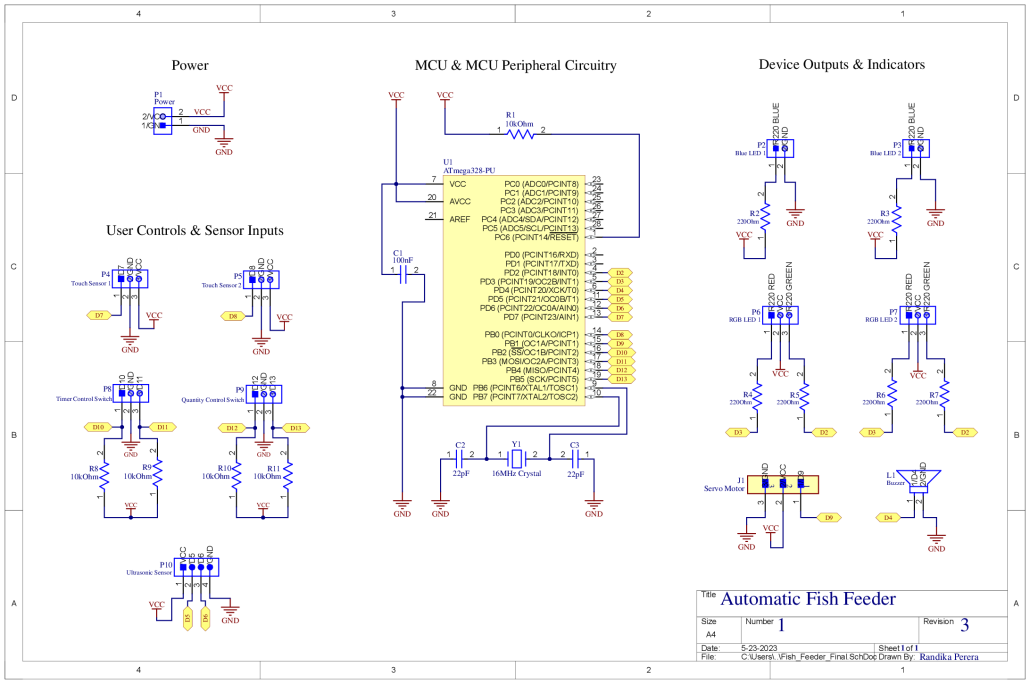


Figure 8.3: Final Schematic Drawn Using Altium Designer

9 | Circuit Verification

Circuit Verification was done in 3 stages:

1. Verification using Proteus Circuit Simulation Software.
2. Verification on a protoboard using physical components **and** an Arduino UNO Development Board.
3. Verification of the final circuit without the Arduino UNO Development Board.

9.1 | Proteus Simulation

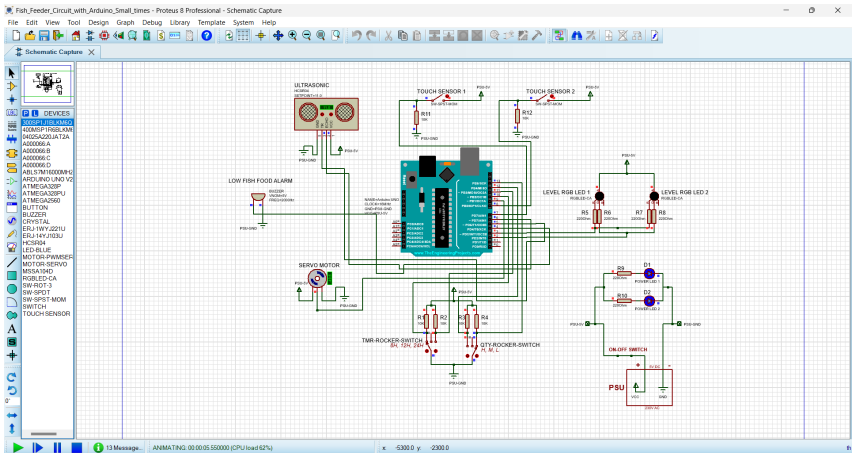


Figure 9.1: Circuit simulation with Arduino UNO Development Board

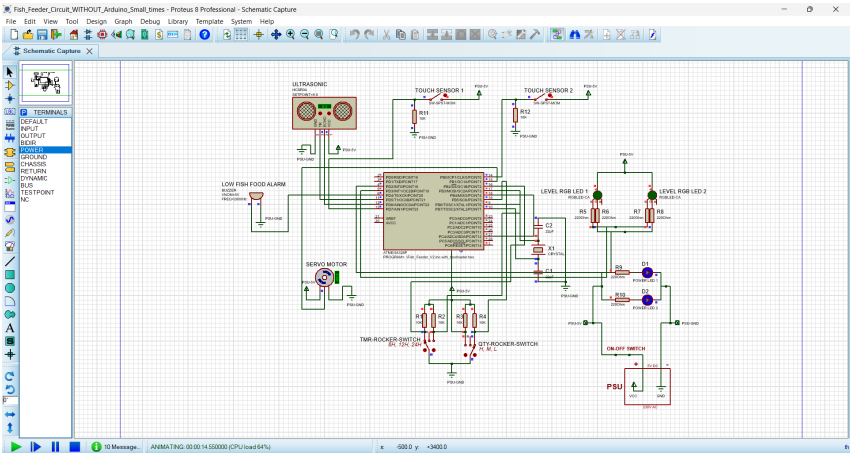


Figure 9.2: Circuit simulation with ATMega328 IC and Required Peripheral Circuitry

9.2 | Protoboard Implementation with Arduino

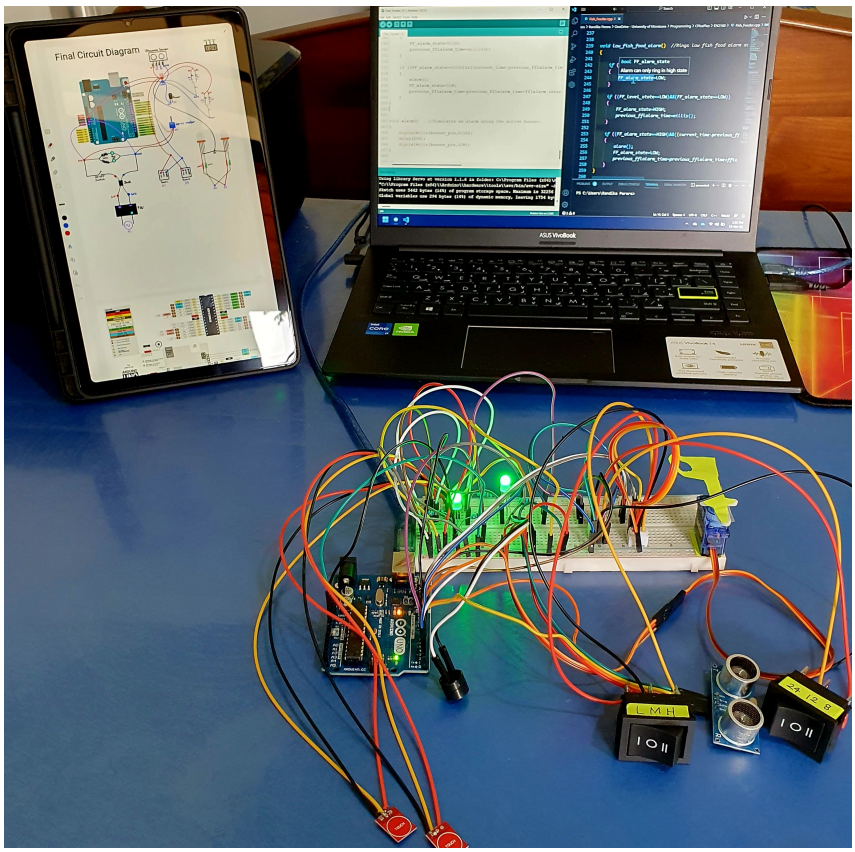


Figure 9.3: Testing the proposed circuit on a protoboard

The proposed circuit was tested on a protoboard with physical components along with an Arduino UNO Development Board as the controller.

After making some minor adjustments to the code, it was possible to achieve all of the expected functions of the device and functionality of the circuit was verified.

Afterwards the final circuit was designed using only the ATMega328 micro-controller.
(Without the Arduino UNO Development Board)

9.3 | Protoboard Implementation without Arduino

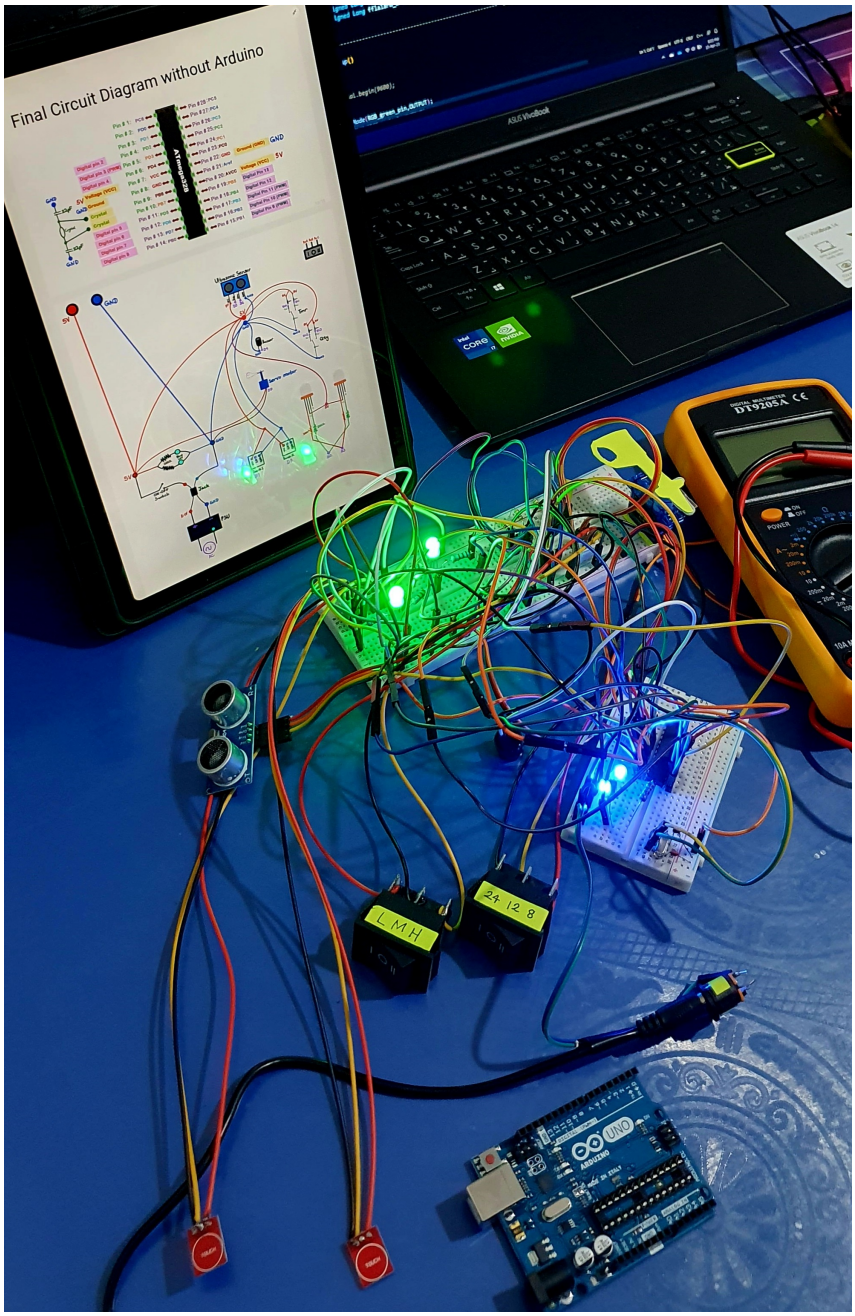


Figure 9.4: Testing the **final** circuit on a protoboard

10 | Production

10.1 | PCB Design

Altium Designer was used for Schematic Capture (See Figure 8.3) and for PCB Design.

Afterwards, the relevant Gerber Files of the PCB were generated and the PCB was manufactured via JLCPCB.

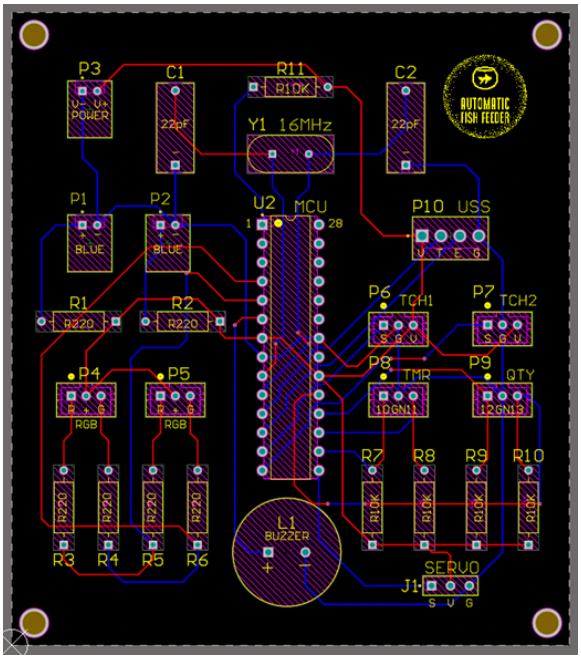


Figure 10.1: 2D View (Before Copper Pour)

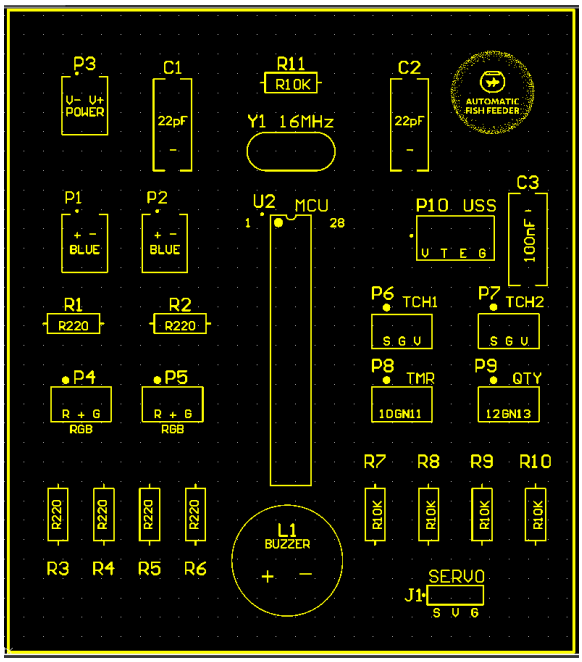


Figure 10.2: Top Overlay

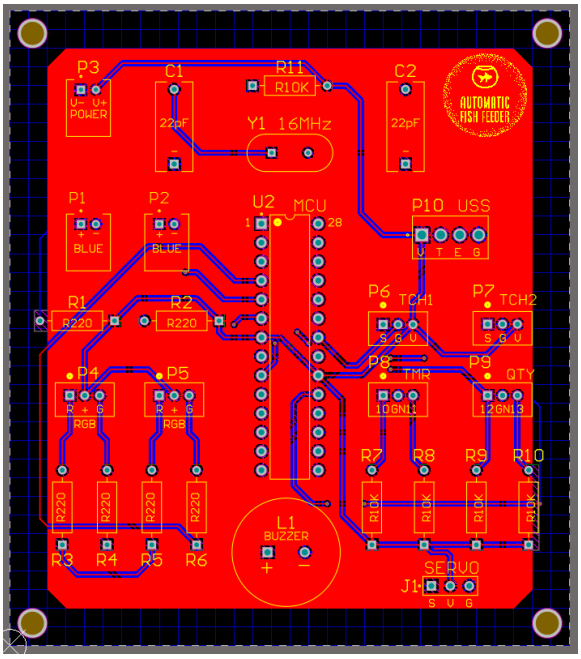


Figure 10.3: Top Layer (After Copper Pour)

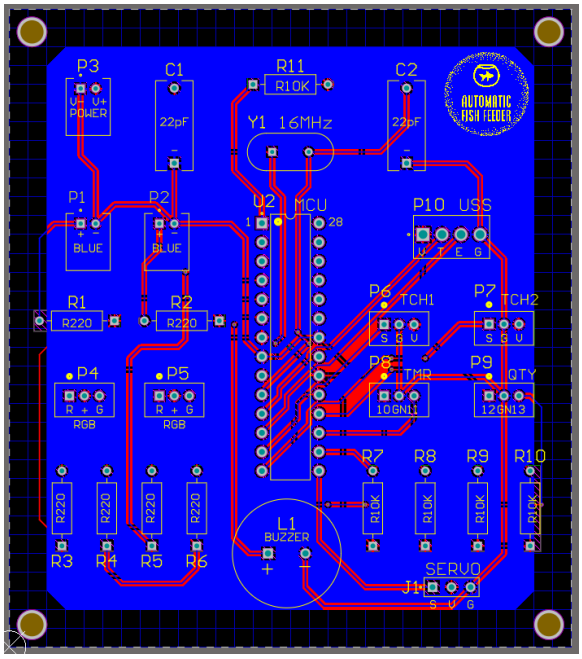


Figure 10.4: Bottom Layer (After Copper Pour)

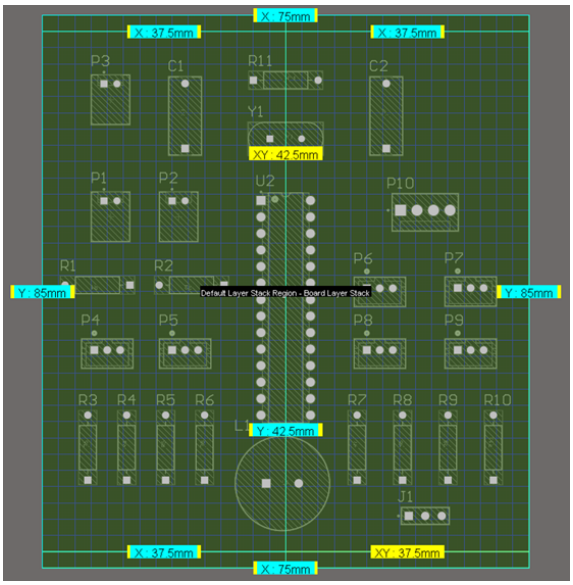


Figure 10.5: PCB Dimensions

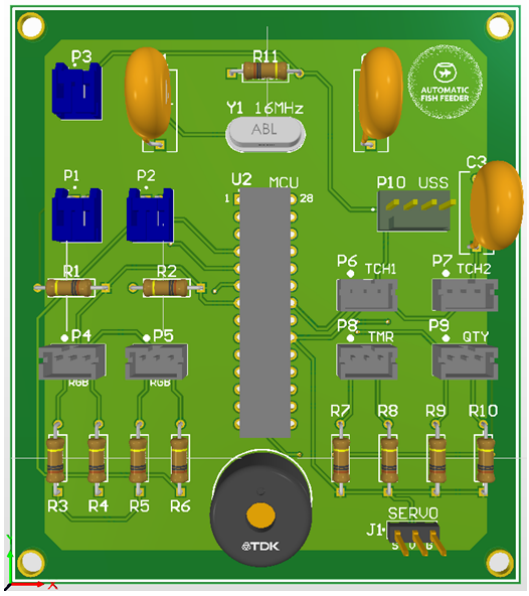


Figure 10.6: 3D View (Top)

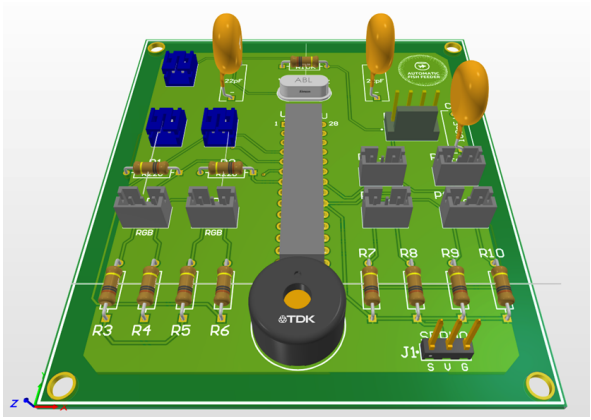


Figure 10.7: 3D View

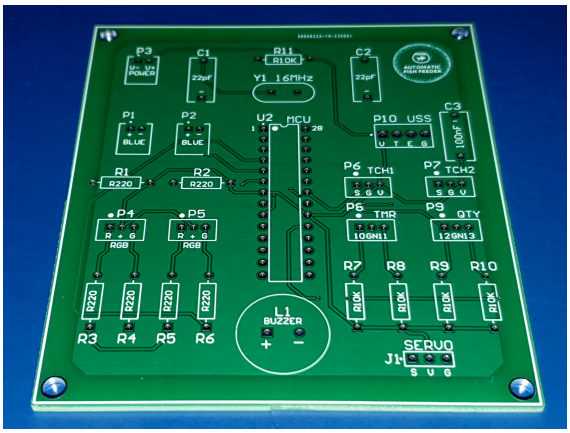


Figure 10.8: After PCB Manufacture

Afterwards, all components were hand soldered onto the printed circuit board and the circuit was tested.

10.2 | Enclosure Design

First, a rough sketch of the enclosure was made considering PCB dimensions and component dimensions. Also, the fish food compartment should have a significant volume since that will directly affect its storage capacity.

Afterwards, each part of the enclosure was designed using SolidWorks solid modeling software. To manufacture each part, 3D printing was used.

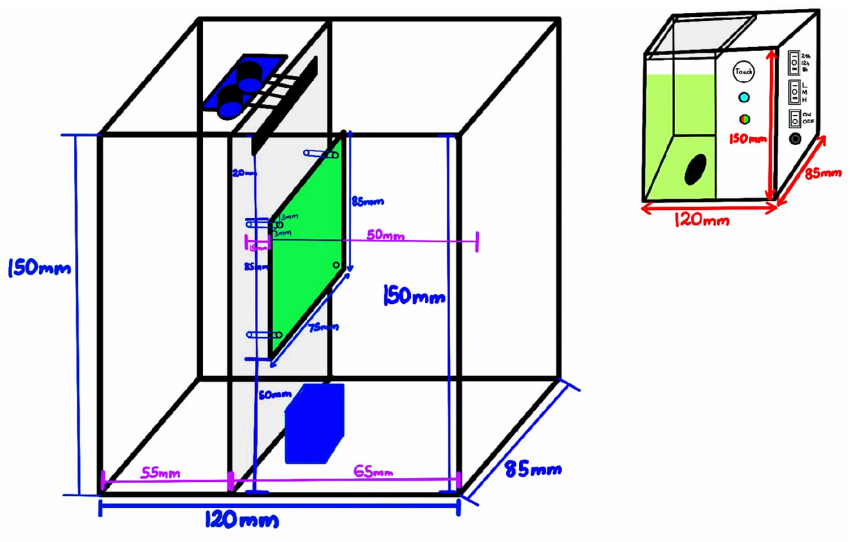


Figure 10.9: Sketch of the Proposed Enclosure

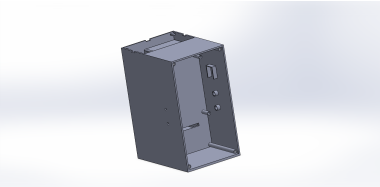


Figure 10.10: Main Body

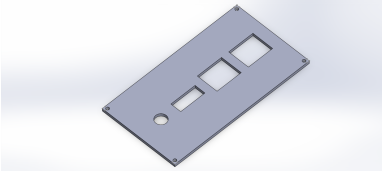


Figure 10.11: Side Panel

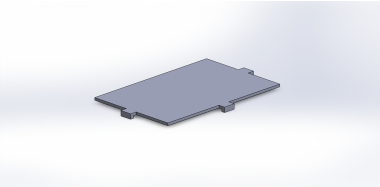


Figure 10.12: Top Lid

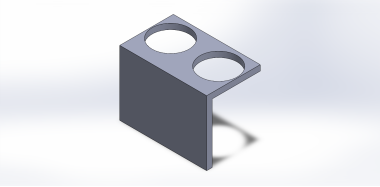


Figure 10.13: Ultrasonic Bracket

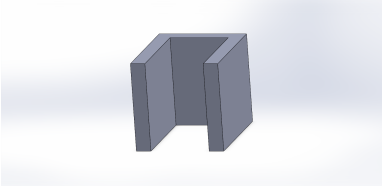


Figure 10.14: Servo Bracket

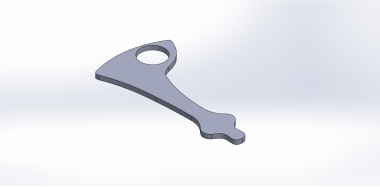


Figure 10.15: Servo Arm

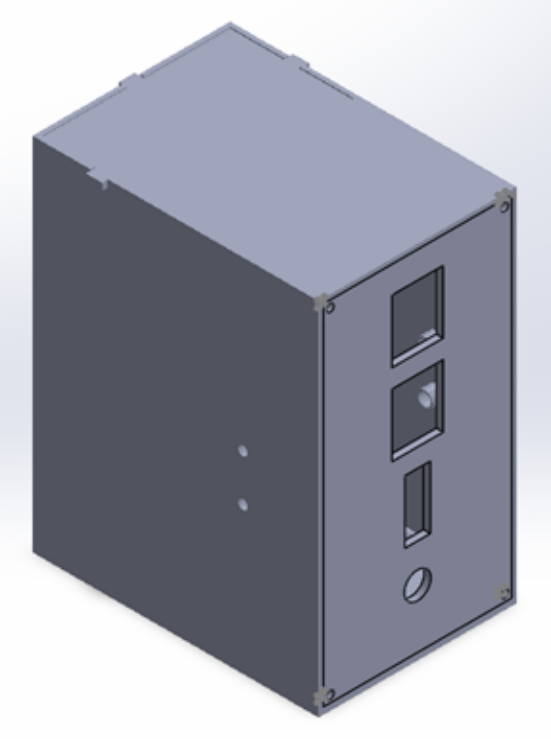


Figure 10.16: Full Assembly

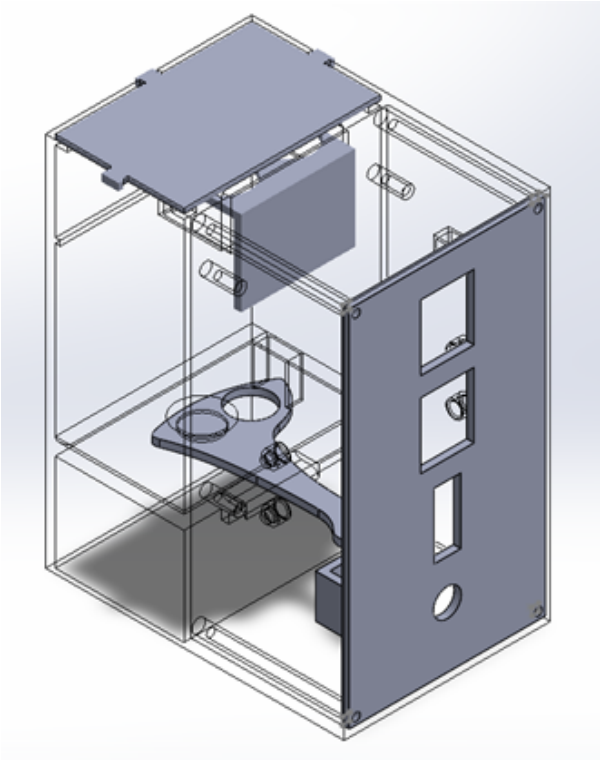


Figure 10.17: Assembly Internal View

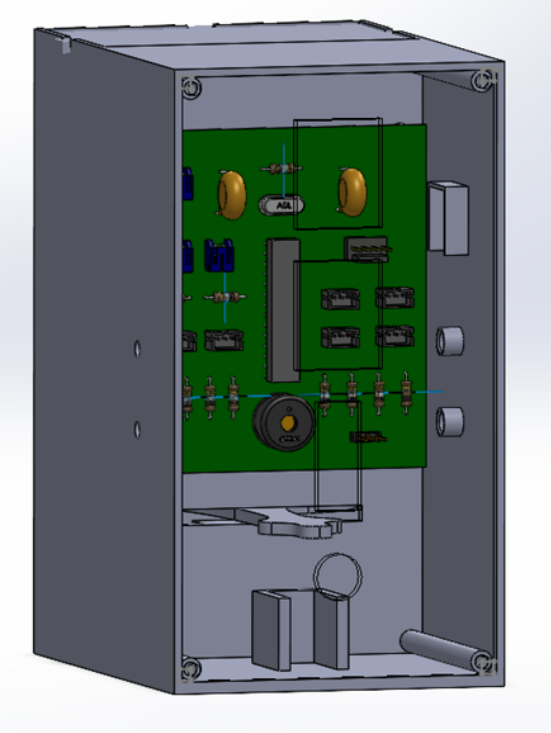


Figure 10.18: Assembly with PCB

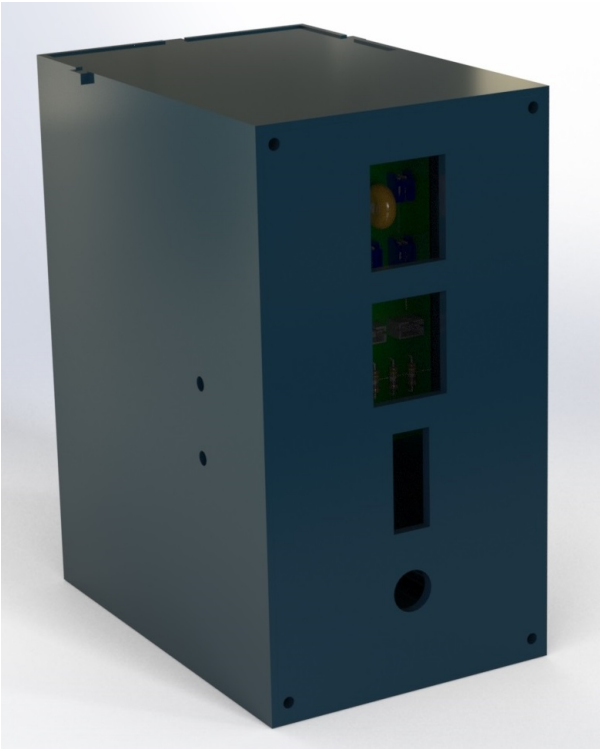


Figure 10.19: Expected Appearance

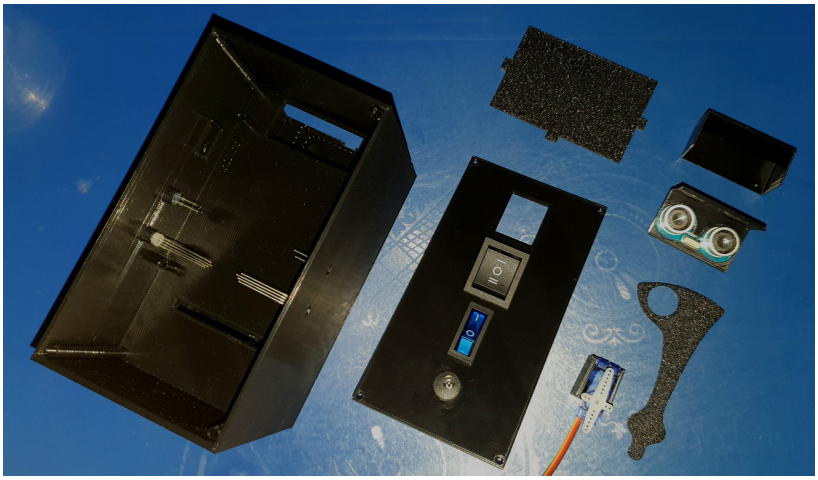


Figure 10.20: 3D Printed Parts of the Enclosure

10.3 | Assembly

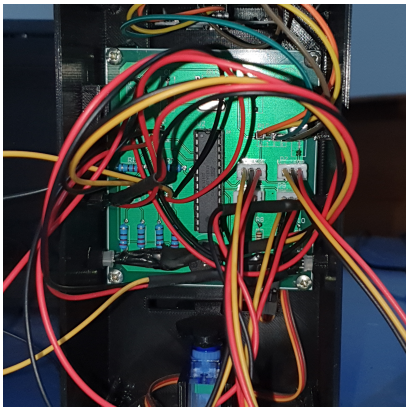


Figure 10.21

Figure 10.22

Figure 10.23: Final Prototype

11 | Bill of Materials

Supplier	Description	Unit Price (in USD)	Unit Price (in LKR)	Quantity	Cost (in LKR)
JLC PCB	Printed Circuit Board	1.2	384	1	384
Xydder Labs 3D	3D Printed Enclosure	-	14650	1	14650
LCSC	ATMega328P MCU	5.64	1804	1	1804
LCSC	16MHz Crystal	0.0861	28	1	28
LCSC	22pF Capacitor	0.2514	80	2	160
LCSC	100nF Capacitor	0.0527	17	1	17
LCSC	10kOhm Resistor	0.0087	3	5	15
LCSC	220Ohm Resistor	0.0177	6	6	36
Arrow	Micro Servo 9g	3.77	1206	1	1206
Arrow	HC-SR04	3.95	1264	1	1264
Tronic.lk	TTP223 Touch Sensor	-	80	2	160
Tronic.lk	RGB LED Diffused	-	30	2	60
Tronic.lk	Blue LED Diffused	-	5	2	10
Tronic.lk	5V Active Buzzer	-	60	1	60
Tronic.lk	IC Base 28 Pin	-	20	1	20
Tronic.lk	JST Connector 3-Pin	-	30	4	120
Tronic.lk	Male Headers	-	20	1	20
Tronic.lk	Female Headers	-	100	1	100
Tronic.lk	ON-OFF Switch	-	50	1	50
Nilambara	3 Position Rocker Switch	-	150	2	300
Tronic.lk	AC/DC Adapter 5V 2A	-	550	1	550
Tronic.lk	DC Jack Female	-	30	1	30
			Net Cost		**21044**

12 | Data sheets

[1] ATMega328-PU Microcontroller

[2] TowerPro SG90 Micro Servo

[3] TTP223 Touch Sensor

[4] HC-SR04 Ultrasonic Sensor Module

[5] JST Connectors

13 | References

[1] "From Arduino to a Microcontroller on a Breadboard — Arduino Documentation," docs.arduino.cc. Available: https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoToBreadboard

[2] D. A. Mellis, "Building an Arduino on a Breadboard — Arduino Documentation," docs.arduino.cc, Oct. 23, 2008. Available: https://docs.arduino.cc/hacking/hardware/building-an-arduino-on-a-breadboard.

[3] "Distance Sensing - SparkFun Electronics," www.sparkfun.com. Available: https://www.sparkfun.com/distance_sensing

[4] A. Halimic, "Choosing the Right Sensor for Measuring Distance," AUTOMATION INSIGHTS, Oct. 19, 2022. Available: https://automation-insights.blog/2022/10/19/choosing-the-right-sensor-for-measuring-distance/

[5] "The Full Arduino Uno Pinout Guide [including diagram]," circuito.io blog, Nov. 18, 2018. Available: https://www.circuito.io/blog/arduino-uno-pinout/

[6] "Programming an AVR ATmega328P with an Arduino," www.brennantymrak.com. Available: https://www.brennantymrak.com/articles/programming-avr-with-arduino

[7] "Pull-up Resistor and Pull-down Resistor Explained," Basic Electronics Tutorials, Aug. 30, 2016. Available: https://www.electronics-tutorials.ws/logic/pull-up-resistor.html

[8] "Using millis() for timing. A beginners guide," Arduino Forum, Oct. 02, 2017. Available: https://forum.arduino.cc/t/using-millis-for-timing-a-beginners-guide/483573

[9] "Secrets of Arduino millis: How it works and how to use it.," Best Microcontroller Projects. Available: https://www.best-microcontroller-projects.com/arduino-millis.html

[10] Proto-Electronics, "Best Rules for PCB Components Placement." Available: https://www.proto-electronics.com/blog/best-rules-for-pcb-components-placement

[11] Proto-Electronics, "Our Top 10 PCB Routing Tips," www.proto-electronics.com. Available: https://www.proto-electronics.com/blog/top-10-pcb-routing-tips